Service and Utility Oriented Distributed Computing Systems: Challenges and Opportunities for Modeling and Simulation Communities

Rajkumar Buyya and Anthony Sulistio

Grid Computing and **D**istributed **S**ystems (GRIDS) Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, Australia
Email: {raj, anthony}@csse.unimelb.edu.au

Abstract: Grids and peer-to-peer (P2P) networks have emerged as popular platforms for the next generation parallel and distributed computing. In these environments, resources are geographically distributed, managed and owned by various organizations with different policies, and interconnected by wide-area networks or the Internet. This introduces a number of resource management and application scheduling challenges in the domain of security, resource and policy heterogeneity, fault tolerance, and dynamic resource conditions. In these dynamic distributed computing environments, it is hard and challenging to carry out resource management design studies in a *repeatable* and *controlled* manner as resources and users are autonomous and distributed across multiple organizations with their own policies. Therefore, simulations have emerged as the most feasible technique for analyzing policies for resource allocation.

This paper presents emerging trends in distributed computing and their promises for revolutionizing the computing field, and identifies distinct characteristics and challenges in building them. We motivate opportunities for modeling and simulation communities and present our discrete-event grid simulation toolkit, called *GridSim,* used by researchers world-wide for investigating the design of utility-oriented computing systems such as Data Centers and Grids. We present various case studies on the use of GridSim in modeling and simulation of Business Grids, parallel applications scheduling, workflow scheduling, and service pricing and revenue management.

1 Introduction

The proliferation of the Internet and the Web, and the availability of powerful computers and high-speed networks as low-cost commodity components are changing the way we do parallel and distributed computing. These technological advances have led to the possibility of using networks of computers as a single, unified computing resource, known as *cluster computing* [1]. Clusters appear in various forms: high-performance clusters, high-availability clusters, and high throughput clusters. In addition, computer scientists in the mid-1990s, inspired by the electrical power grid's pervasiveness and reliability, began exploring the design and development of a new IT (Information Technology) infrastructure exhibiting quality of seamless access to computing resources distributed across different organisations [2]. This led to growing interest in coupling geographically distributed resources for solving large-scale problems, leading to what is popularly called the Grid [3] and peer-to-peer (P2P) computing [4] networks. A large number of computing devices ranging from high-end computing systems such as supercomputers, to specialized systems such as visualization devices, storage systems, sensors, and scientific instruments, are logically coupled together in a Grid (see Figure 1) that serves as a *Cyberinfrastructure* supporting collaborative scientific and business applications.

In the business world, cluster architecture-based computing systems, called *data centers*, offering high-performance and reliable hosting services are widely used. The low-cost availability of data center services has encouraged many businesses to outsource their computing needs; thus heralding a new utility computing model.

Utility computing is envisioned to be the next generation of IT evolution that depicts how computing needs of users can be fulfilled in the future IT industry [5]. Its analogy is derived from the real world where service providers maintain and supply utility services, such as electrical power, gas, and water to consumers. Consumers in turn pay service providers based on their usage. Therefore, the underlying design of utility computing is based on a service provisioning model, where users pay providers for using computing power only when they need to use.

The emerging cloud computing systems such as Amazon EC2 (Elastic Compute Cloud) are the recent incarnation of data centers [49]. They have high potential for enabling the creation of market-maker that further virtualizes clouds from different providers. Thus bringing buyers and sellers together, and realizing virtual Grid computing.

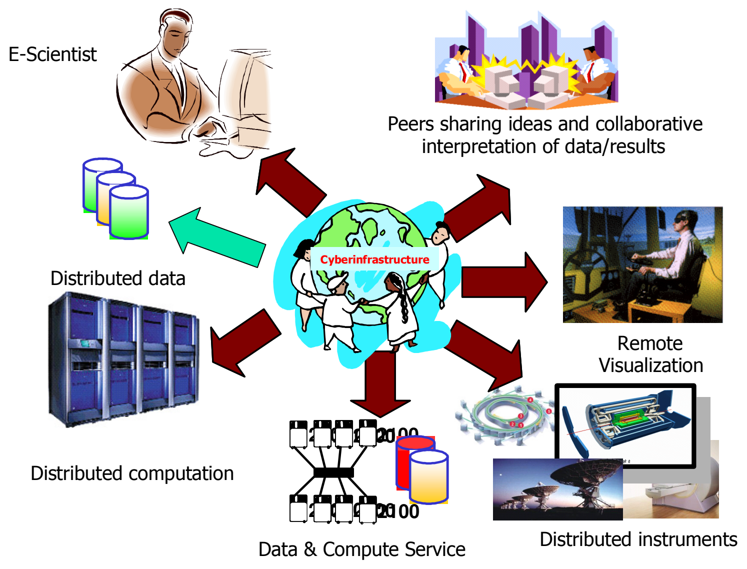


Figure 1: Grid as a Cyberinfrastructure for coupling and sharing distributed resources.

These developments are leading to the realization of the vision of Leonard Kleinrock, one of the chief scientists of the original Advanced Research Projects Agency Network (ARPANET) project which seeded the Internet, who stated in 1969 [6]: "As of now, computer networks are still in their infancy, but as they grow up and become sophisticated, we will probably see the spread of '*computer utilities*' which, like present electric and telephone utilities, will service individual homes and offices across the country."

1.1 Potential of Grids as Service and Utility-Oriented Computing Systems

Grid is defined as a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed "autonomous" resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements. Grid computing systems support coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations (VOs) [24]. A high level view of a global Grid with its key components is shown in Figure 2. A Grid user can easily access globally distributed resources by interacting with a Grid resource broker. The user essentially interacts with a resource broker that hides the complexities of Grid computing [14]. The broker discovers resources that the user can access using information services, negotiates for access costs using trading services, maps tasks to resources (scheduling), stages the application and data for processing (deployment), starts job execution, and finally gathers the results. It is also responsible for monitoring and tracking application execution progress along with adapting to the changes in Grid runtime environment conditions and failures.

Grids offer a number of benefits such as:

- Transparent and on-demand access to distributed and heterogeneous resources.
- Improved productivity with reduced processing time.
- Provisioning of extra resources to solve problems that were previously unsolvable due to the lack of resources.
- A more resilient infrastructure with autonomic management capabilities [50], on-demand aggregation of resources from multiple sites to meet unforeseen demand.
- Seamless computing power achieved by exploiting under-utilized or unused resources that are otherwise wasted.
- Maximum utilization of computing facilities to justify IT capital investments.
- Coordinated resource sharing and problem solving through VOs that facilitates collaboration across physically dispersed departments and organisations.
- Service Level Agreement (SLA) based resource allocation to meet Quality of Service (QoS) requirements.
- Reduced administration effort with integration of resources as compared to managing multiple standalone systems.

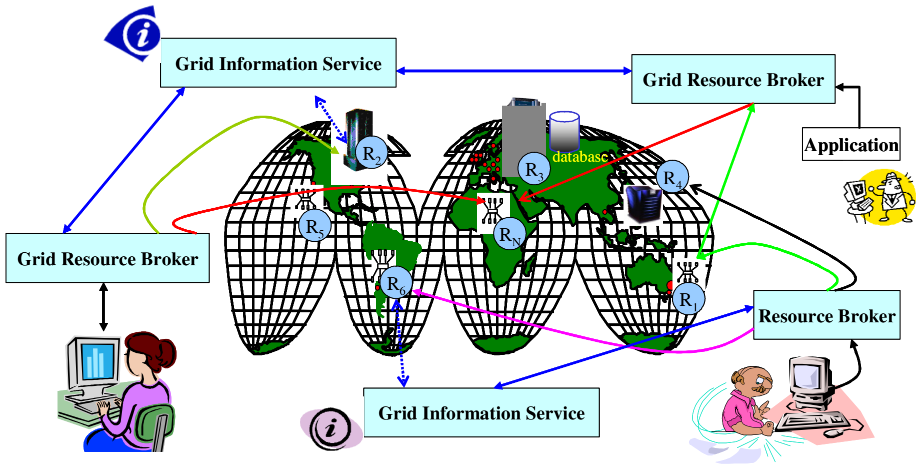


Figure 2: A high level view of a global Grid.

Service-oriented architecture (SOA) and Web-services technologies are extensively used in the construction of Grid middleware and applications [12]. The Grid architecture consists of four key layers: fabric, core middleware, user-level middleware, and applications [11]. 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 environment. The Grid applications include those constructed using Grid libraries or legacy applications that can be Grid-enabled using user-level middleware tools.

A diverse range of applications are explored using Grids, some of which include: aircraft engine diagnostics, earthquake engineering, virtual observatory, bioinformatics, drug discovery, digital image analysis, high energy physics, astrophysics, and multi-player gaming [3]. Grids can be primarily classified into the following types, depending on the nature of the applications they are driving [30]:

- *Computational Grid*: Aggregates the computational power of globally distributed computers (e.g. SETI@Home [9] and TeraGrid [42]).
- *Data Grid*: Emphasizes on a global-scale management of data to provide data access, integration, and processing through distributed data repositories (e.g. LHCGrid [26]).
- *Application Service Provisioning (ASP) Grid*: Focuses on providing access to remote applications, modules, and libraries hosted on data centers or Computational Grids (e.g. NetSolve [34]).
- *Interaction Grid*: Focuses on interaction and collaborative visualization between participants (e.g. AccessGrid [15]).
- *Knowledge Grid*: Aims towards knowledge acquisition, processing, management, and provides business analytics services driven by integrated data mining services (e.g. KnowledgeGrid [20] and EU Data Mining Grid [45]).
- *Utility Grid*: Focuses on providing all the Grid services including compute power, data, and services to end-users as IT utilities on a subscription basis and the infrastructure necessary for negotiation of required QoS, establishment and management of contracts, and allocation of resources to meet competing demands from multiple users and applications (e.g. Utility Data Center [25] at enterprise level and Gridbus [46] at global level).

These types of Grids can be logically realized as a layer of services with one building on top of the other, as shown in Figure 3. A Grid on a higher layer utilizes the services of Grids that operate at lower layers in the design. For example, a Data Grid utilizes the services of Computational Grid for data processing, hence builds on it. Moreover, lower-layer Grids focus heavily on infrastructural aspects, whereas higher-layer ones focus on users and QoS delivery.

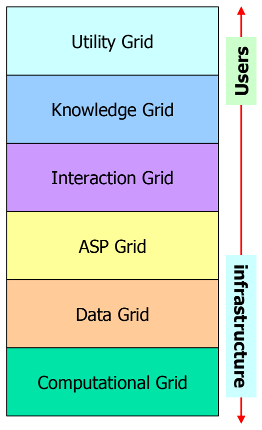


Figure 3: Types of Grids and their focus.

1.2 Grid Characteristics and Challenges

The Grid environments comprise heterogeneous resources, fabric management systems (single system image OS, queuing systems) and policies, and applications with varied requirements (compute, memory, network intensive). The users: *producers* (also called *resource owners*) and *consumers* (also called *end-users*) 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 approaches to resource management that attempt to optimize system-wide measures of performance cannot be employed. This is because traditional approaches use centralized policies that need complete state information and a common fabric management policy, or decentralized consensus based policy. In large-scale Grid environments, it is hard to define an acceptable system-wide performance matrix and common fabric management policy. Apart from the centralized approach, two other approaches that are used in distributed resource management are: *hierarchical* and *decentralized* scheduling or a combination of them [44]. We note that similar heterogeneity and decentralization complexities exist in human economies where market driven economic models have been used to successfully manage them. Therefore, in [16], we investigated on the use of economics as a metaphor for management of resources in Grid computing environments.

The researchers and students, investigating resource management and scheduling for large scale distributed computing, need a simple framework for deterministic modeling and simulation of resources and applications to evaluate scheduling strategies. For most investigators who do not have access to ready-to-use special experimental infrastructures such as France's Grid 5000 and Netherlands's DAS [47], it is expensive and time consuming to build them. Also, even for those who have access, the testbed size is limited to a few resources and domains; and testing scheduling algorithms for scalability and adaptability, and evaluating scheduler performance for various applications and resource scenarios is harder to trace. In addition, it is challenging to create a *repeatable* and *controlled* environment for experimentation and evaluation of scheduling strategies. This is because resources in Grids such as TeraGrid in US are dynamic and span across multiple administrative domains, each with their own policies, users, and priorities.

Simulation appears to be one of the most feasible technique for analyzing algorithms and policies for resource allocation. 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 and resources, which is very hard to coordinate and build at large-scale research environment for investigation purpose.

To support studies in resource management for Grids, we have developed a Java-based simulation toolkit, called GridSim [17], for simulating various types of Grids. The Grid computing researchers and educators also recognized the importance and the need for such a toolkit for modeling and simulation environments [41]. It should be noted that this paper has a major orientation towards Grid, however, we believe that our discussion and thoughts also apply equally well to P2P systems since resource management and scheduling issues in both systems are quite similar.

The GridSim toolkit supports modeling and simulation of a wide range of heterogeneous resources, such as single or multiprocessors, shared and distributed memory machines with different capabilities and configurations. It can be used for modeling and simulation of application scheduling on various parallel and distributed computing systems such as clusters, Grids, and P2P networks. In fact, P2P techniques for resource organization and discovery are being used in building Grids. A set of characteristics that helps distinguish clusters, Grids and P2P systems is listed in Table 1.

Table 1: Key Characteristics of Clusters, Grids, and P2P Systems.

Systems / Characteristics	Clusters / Data Centers	Grids	P2P
Population	Commodity Computers	High-end computers (servers, clusters)	Computers at the edge of network (e.g., desktop PCs)
Size / Scalability	100s	1000s	Millions
Ownership	Single	Multiple	Multiple
Discovery	Membership Services	Centralised Indexing & Decentralised Info Services	Decentralized
Service Negotiation	Yes	Yes, SLA based	Lack of enterprise quality support
User Management	Centralised	Decentralised and also VO (virtual organisation)-based	Decentralised
Resource management	Centralized	Distributed	Distributed
Allocation / Scheduling	Centralised	Decentralised	Decentralised
Standards / Inter-Operability	VIA based	Web services-based and Open Grid Forum efforts	No standards
Single System Image	Yes	No	No
Capacity	Stable & Guaranteed	Varies, but high	Varies
Throughput	Medium	High	Very High
Interconnection Network	Dedicated, High-end	Mostly public Internet, Some uses high-end networks	Public Internet
Speed (Latency, Bandwidth)	Low, high	High, Low	High, Low
Application Drivers	Science, business, enterprise computing, web applications, data centers	e-Science, e-Business, multi-party conferencing (e.g., AccessGrid), integration of scientific instruments	Sharing of files (e.g., music files), communication (e.g., Skype)

The resources in clusters are located in a single administrative domain and managed by a single entity whereas, in Grid and P2P systems, resources are geographically distributed across multiple administrative domains with their own management policies and goals. Another key difference between cluster and Grid/P2P systems arises from the way application scheduling is performed. The *schedulers* in cluster systems focus on enhancing the overall system performance and utility, as they are responsible for the whole system. Whereas, schedulers in Grid/P2P systems called *resource brokers*, focus on enhancing the performance of a specific application in such a way that its end-users' QoS requirements are met.

2 Grid Simulation Tools

Simulation has been used extensively for modeling and evaluation of real world systems, from business process and factory assembly line to computer systems design. Accordingly, over the years, modeling and simulation has emerged as an important discipline and many standard and application-specific tools and technologies have been built. They include simulation languages (e.g., Simscript [18]), simulation environments (e.g., Parsec [10]), simulation libraries (e.g., SimJava [27]), and application specific simulators (e.g., NS-2 network simulator [40]). While there exists a large body of knowledge and tools, there are very few well-maintained tools available for application scheduling simulation in Grid computing environments. However, for simulating a Grid, a tool needs to be able to model the interactions of resource brokers, resources and the network. For these purposes, a Grid simulation tool must have at least the following functionalities:

- Able to model heterogeneous computational resources.
- Extensible and modifiable so that various brokering mechanisms and scheduling systems can be implemented and analyzed.
- Able to store and query information regarding to resource properties. This can be achieved by using an indexing service.
- Able to specify an arbitrary network topology in the simulated Grid environment.

Table 2 lists some Grid simulation tools that support one or more of these functionalities.

OptorSim [13] is developed as part of the EU DataGrid project. It aims to mimic the structure of an EU DataGrid Project and study the effectiveness of several Grid replication strategies. It is quite a complete package as it incorporates few auction protocols and economic models for replica optimization. However, it mainly focuses more on the issue of data replication and optimisation.

The SimGrid toolkit [32], developed at the University of California at San Diego (UCSD), is a C language based toolkit for the simulation of application scheduling. It supports modeling of resources that are time-shared and the load can be injected as constants or from real traces. It is a powerful system that allows creation of tasks in terms of their execution time and resources with respect to a standard machine capability.

The MicroGrid emulator [36], undertaken at the UCSD, is modeled after Globus [23], a software toolkit used for building Grid systems. It allows execution of applications constructed using the Globus toolkit in a controlled virtual Grid resource environment. MicroGrid is actually an emulator meaning that actual application code is executed on the virtual Grid. Thus, the results produced by MicroGrid are much closer to the real world as it is a real implementation. However, using MicroGrid requires knowledge of Globus and implementation of a real system/application to study.

GangSim [22], developed at the University of Chicago, is targeted towards a study of usage and scheduling policies in a multi-site and multi-VO environment. It is able to combine discrete simulation techniques and modeling of real Grid components in order to achieve scalability to Grids of substantial size.

Finally, GridSim [17][39] development led by the University of Melbourne, supports simulation of various types of Grids and application models scheduling. The following sections explain GridSim capabilities, architecture, and its usage by the Grids community.

Table 2: Some recent and notable Grid simulators.

Functionalities	GridSim	OptorSim	SimGrid	MicroGrid	GangSim
Resource Extensibility	yes	no	yes	yes	no
Data replication	yes	yes	no	no	no
Disk I/O overheads	yes	no	no	yes	no
Complex file filtering or data query	yes	no	no	no	no
Scheduling user jobs	yes	no	yes	yes	yes
reservation of a resource	yes	no	no	no	no
Workload trace-based simulation	yes	no	yes	no	yes
Differentiated network QoS	yes	no	no	no	no
Generate background network traffic	yes	yes	yes	yes	no
Auction Framework	yes	yes	no	no	no

3 GridSim Toolkit

GridSim is an open-source software platform that enables users to model and simulate the characteristics of Grid resources and networks with different configurations. GridSim is of great value to both students and experienced researchers who want to study Grids, or test new algorithms and strategies in a controlled environment. By using GridSim, they are able to perform repeatable experiments and studies that are not possible in a real dynamic Grid environment.

Some of GridSim's features are outlined below:

- It allows the modeling of different resource characteristics and their failure properties;
- It enables the simulation of workload traces taken from real supercomputers;
- It supports reservation-based or auction mechanisms for resource allocation;
- It allocates incoming jobs based on space- or time-shared mode;
- It has the ability to schedule compute- and/or data-intensive jobs;
- It has a background network traffic functionality based on a probabilistic distribution [39]. This is useful for simulating data-intensive jobs over a public network where the network is congested;
- It provides clear and well-defined interfaces for implementing different resource allocation algorithms;
- It allows the modeling of several regional Grid Information Service (GIS) components. Hence, it is able to simulate a VO scenario;
- It has a visualisation tool for tracing sequences of simulation execution.

In Grids, resources can be part of one or more VOs, as mentioned earlier. The concept of a VO allows users and institutions to gain access to their accumulated pool of resources to run applications from a specific field [24], such as high-energy physics or aerospace design.

With these features, GridSim offers researchers the functionality and the flexibility of simulating Grids for various types of studies such as Grid meta-scheduling [7], workflow scheduling [35], and VO-oriented resource allocation [43].

3.1 GridSim Architecture

We designed GridSim as a multi-layer architecture for extensibility as shown in Figure 1. This allows new components or layers to be added and integrated into GridSim easily. In addition, the layered GridSim architecture captures the model of the Grid computing environment. GridSim is based on SimJava [27], a general purpose discrete-event simulation package implemented in Java. Therefore, the first layer at the bottom of Figure 4 is managed by SimJava for handling the interaction or events among GridSim components.

All components in GridSim communicate with each other through message passing operations defined by SimJava. The second layer models the core elements of the distributed infrastructure, namely Grid resources such as clusters, storage repositories and network links. These core components are absolutely essential to create simulations in GridSim. The third and fourth layers are concerned with modeling and simulation of services specific to Computational and Data Grids [38] respectively. Some of the services provide functions common to both types of Grids such as information about available resources and managing job submission.

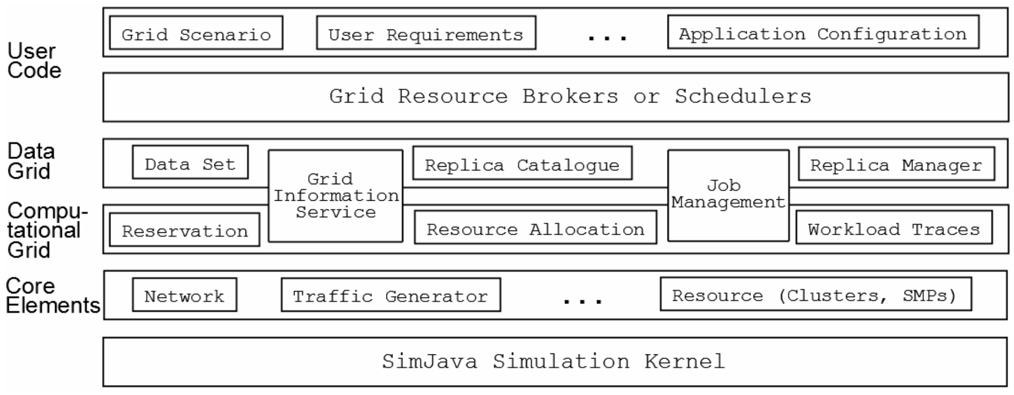


Figure 4: A layered architecture of GridSim and its components.

In case of Data Grids, job management also incorporates managing data transfers between computational and storage resources. Replica catalogs, information services for files and data, are also specifically implemented for Data Grids. The fifth layer contains components that aid users in implementing their own schedulers and resource brokers so that they can test their own algorithms and strategies. The layer above this helps users define their own scenarios and configurations for validating their algorithms.

3.2 Extensible Grid Resource Framework

In Grid computing, any hardware or software component such as a cluster, a supercomputer or a storage repository is called a resource. Computing resources allow users to execute the required application while storage resources allow the users to access datasets and store the results of the computation. GridSim provides well-defined abstractions for configuring the resource management on a resource. Each resource is associated with an *AllocationPolicy* object that allocates internal nodes to the user jobs depending on the policy. Hence, the *GridResource* object in GridSim only acts as an interface between users and the local scheduler, as shown in Figure 5. It is up to the scheduler to handle and to process submitted jobs. This approach gives the flexibility to implement various scheduling algorithms for a specific resource-based system. Currently, GridSim has *TimeShared* and *SpaceShared* objects that use Round Robin and First Come First Serve (FCFS) approaches, respectively, as shown in Figure 3. The advantage of this design is that adding a new scheduler does not require modification of existing resource and/or other scheduling classes. Creating a new scheduler is as simple as extending the *AllocPolicy* class or *ARPolicy* class with advance reservation [37], and implementing the required abstract methods. For an example, *ARSimpleSpaceShared* is a child of *ARPolicy* class that uses FCFS approach to schedule reserved jobs.

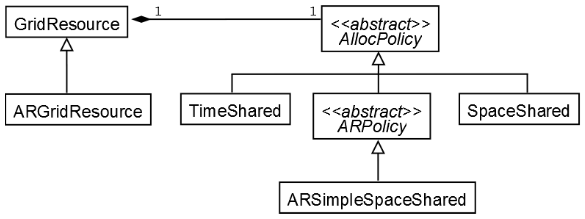


Figure 5: A GridSim resource class diagram.

Figure 6 shows the components of a Grid resource in GridSim. A Grid resource is associated with one or more Storage objects that can each model either a hard disk-based or a tape-based storage device.The resource has a *ReplicaManager* which handles incoming requests for datasets located on the storage elements. In case a new replica is created, it also registers the replica with the replica catalog (RC). The replica manager can be extended to incorporate different replica retention or deletion policies. A *LocalRC* object can be optionally associated with the resource to index available files internally, and handle direct user queries about local files. However, other resources cannot query this RC object.

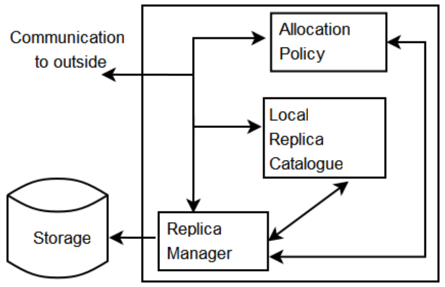


Figure 6: Components of a Grid resource in GridSim.

3.3 Extensions to GridSim

In this section, we highlight some extensions that are built on top of GridSim.

3.3.1 Grid Scheduling SIMulator

Grid Scheduling SIMulator (GSSIM) [31], created by Poznan University of Technology (Poland), is designed as a simulation framework which enables easy-to-use experimental studies of various scheduling algorithms. It is mainly targeting at simulating computational Grids by using various synthetic or real workloads.

3.3.2 Grid Network Buffer

Grid Network Buffer (GNB) [19], developed by University of Castilla La Mancha (Spain), is aimed at making network QoS as an integral part of job scheduling decisions in Grids. Therefore, GNB has both a resource scheduler and an admission control in its architecture.

3.3.3 Alea Grid Simulator

Alea Grid Simulator [29], extended by Masaryk University (Czech Republic), is used to design and test complex scheduling algorithms for various Grid scenarios. Moreover, Alea simulates these algorithms in static and/or dynamic environments.

3.3.4 Grid Agents Platform

Grid Agents Platforms (GAP) [33], proposed by University of Catania (Italy), is targeted at provisioning multimedia contents based on QoS requirements in computational Grids. Therefore, GAP aims at minimising the network latency when transferring multimedia contents to a Grid resource for processing.

3.3.5 Web-based Grid Scheduling Platform

Web-based Grid Scheduling Platform (WGridSP) [28], developed by Andong National University (South Korea), is designed to allow users to do resource modelling, testing new algorithms and performance evaluations using GridSim in a web environment. With this approach, WGridSP hides some of the technical complexities of GridSim, hence, makes things easier for the users.

3.4 GridSim Usages

Research students in our GRIDS laboratory are themselves heavy users of GridSim and extend it whenever necessary for their own research needs. They used it for investigating SLA-based resource allocation in clusters, coordinated resource provisioning in federated grids, workflow and data Grid applications scheduling, peering between Grids, and power-aware scheduling. In the last 5 years, GridSim has been continuously extended in this manner to include many new capabilities and has also received contributions from external collaborators—National University of Singapore has contributed a QoS-based network module, and University of Ljubljana has contributed a DataGrid module. Academic and industrial users of GridSim include: IBM, Unisys, HP, University of Southern California, France Telecom, Indian Institute of Technology, and Ume˚a University. Table 3 lists some of the prominent users of GridSim.

4 GridSim Case Studies

In this section, we present selected case studies that demonstrate the usefulness of GridSim for simulating variety of systems, applications, and scenarios from industrial and academic users.

4.1 Meta-Scheduler for Business Grids

IBM India Research Lab has used GridSim to simulate a Grid meta-scheduler, called Data replication and Execution CO-scheduling (DECO) [7]. DECO is responsible for deciding which data and jobs are to be replicated and to be executed, respectively. Then, DECO is in charge of scheduling data transfers and jobs to selected resources according to users' SLA requirements, as shown in Figure 7.

Table 3: Various users of GridSim and their targeted application domain for simulation.

Application Domain	**Organisation**
Scientific Workflows	The University of Southern California, USA
Business Grids	IBM Research Lab
Grid Resource and Virtual Organisation	Umeå University, Sweden
Network modelling	National University of Singapore
Grid Security Studies	France Telecom
Scheduling Studies	University of Malay, Malaysia
Grid economics	Technical University of Catalunya, Spain
Grid Market Studies	Indian Institute of Technology
Semantic Grid Studies	Monash University, Australia
Utility-based Resource Management	The University of Manchester, UK
DataGrid Simulation	The University of Ljubljana, Slovenia
Data Centre Modelling	Unisys, USA
Hierarchical Scheduling	Universidad Complutense de Madrid, Spain
Multi-Criteria Grid Scheduling	Poznan Supercomputing Center, Poland

With this approach, the authors claimed that DECO delivers improvements in both resource revenues and lower jobs' waiting time compared to Earliest Fit (EF). EF uses a greedy approach, where it assigns jobs to available resources at the earliest possible time.

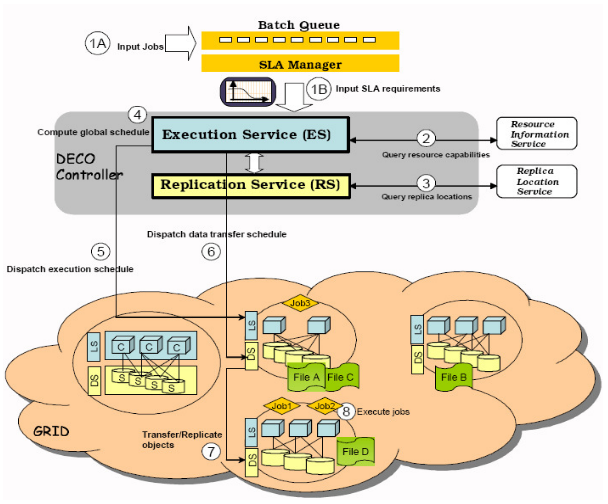

Batch Queue
1A Input Jobs
SLA Manager
1B Input SLA requirements
4 Compute global schedule
Execution Service (ES)
DECO Controller
Replication Service (RS)
2 Query resource capabilities
3 Query replica locations
Resource Information Service
Replica Location Service
Dispatch data transfer schedule
5 Dispatch execution schedule
6
Job3
File A
File C
File B
Job1
Job2
8 Execute jobs
File D
GRID
Transfer/Replicate objects
7
LS
DS

Figure 7: IBM's DECO architecture [7].

4.2 Parallel Applications on CrossGrid

The University of Santiago (Compostella, Spain) has used GridSim to simulate parallel applications in a Grid [8]. The authors compared various scheduling algorithms, such as FCFS, First Fit, EASY Backfilling, and simulated a CrossGrid tested in their performance evaluation, as shown in Figure 8. With GridSim, the authors can analyse the performance of different algorithms using parameters from a real Grid testbed. Hence, reducing complexities and saving time.

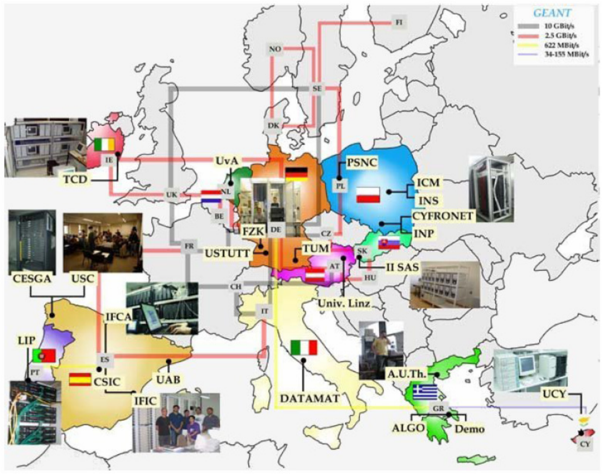


Figure 8: The CrossGrid testbed [21].

4.3 Storage-Aware Workflow Scheduling

The University of Southern California has used GridSim to optimise disk usage when scheduling large-scale scientific workflows in distributed resources, such as Grids [35]. This is because these resources have limited storage capacities and are shared among other users. In the paper, the authors used a Laser Interferometer Gravitational Wave Observatory (LIGO) workflow as a case study. Then, cleanup nodes are added to this workflow structure to remove unwanted files, as shown in Figure 9. As a result, the authors claimed that by optimising the disk usage through a cleanup algorithm, they were able to decrease the storage consumption of the workflow application by up to 57%. This work shows an example that by using GridSim, the system administrators can decide rapidly whether to upgrade their costly infrastructure or improve the performance of the system software.

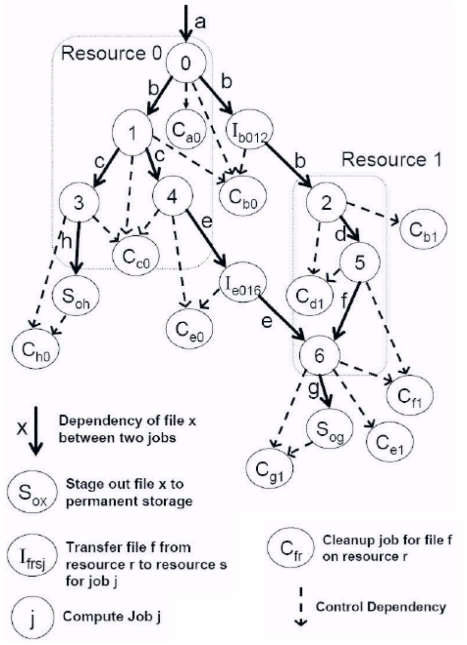

Resource 0
Resource 1
Dependency of file x between two jobs
Stage out file x to permanent storage
Transfer file f from resource r to resource s for job j
Cleanup job for file f on resource r
Compute Job j
Control Dependency

Figure 9: Executable workflows with cleanup nodes mapped to two resources [35].

4.4 Revenue Management for Data Centers and Grids

We have leveraged GridSim's functionalities to investigate the effectiveness of using Revenue Management (RM) for determining pricing of advance reservations in Grids [48]. The main objective of RM is to maximize profits by providing the right price for every product to different customers, and periodically update the prices in response to market demands. Hence, in the model shown in Figure 10, each resource has a Revenue Management System (RMS), which is responsible for handling future bookings and determining their prices based on VOs, demands, and time periods (e.g., peak or off-peak). Moreover, in the model, each user has a broker that is responsible for scheduling jobs based on QoS, such as deadline or budget (in G$). The RMS is created by extending the *GridResource* class, whereas the RM techniques are implemented by extending the *ARPolicy* class. The broker of a user is developed by extending the *GridUser* class.

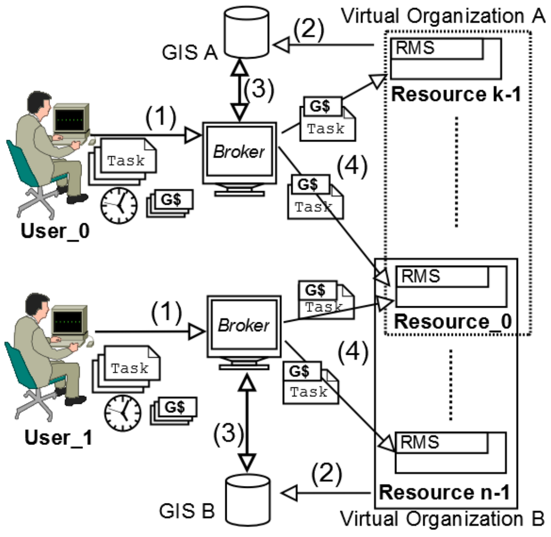


Figure 10: An overview of the model, where resources are assigned to different VOs.

For the performance evaluation, we simulated the EU DataGrid Testbed 1 [26], as shown in Figure 11. In the evaluation, resources are partitioned into four VOs based on their location, and each of them has different characteristics (e.g., the number of CPUs and their processing capability and access costs). Moreover, synthetic workloads are used to model these resources.

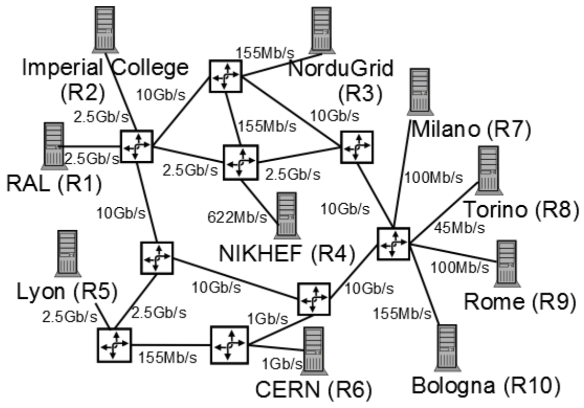
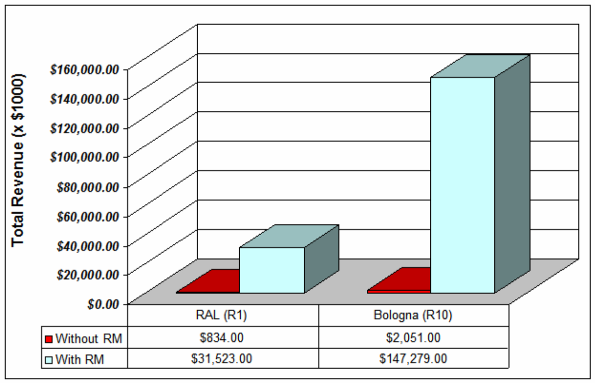


Figure 11: The simulated topology of EU DataGrid TestBed.

Figure 12 shows how *RAL (R1)* and *Bologna (R10)* benefited from using RM techniques as part of their system, instead of using static pricing. However, other resources used RM in the first place. The result shows that both resources gained more than 3,600% in profits by using RM.



	RAL (R1)	Bologna (R10)
🟥 Without RM	$834.00	$2,051.00
□ With RM	$31,523.00	$147,279.00

Figure 12: Total Revenue for RAL (R1) and Bologna (R10).

In the evaluation, we segment users into three categories: *Premium*, *Business* and *Budget* users, based on VOs, time of bookings and set of restrictions or QoS requirements. From Figure 13, both the *Premium* and *Business* users are a major source of revenue for a resource. Therefore, in a competitive demand and supply market, a resource needs to differentiate itself among others to attract these users. Moreover, using RM techniques ensure that resources are allocated to applications that are highly valued by the users.

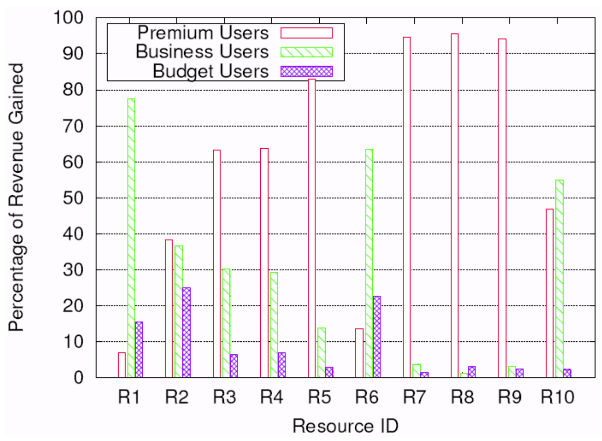


Figure 13: Percentage of income revenue, where all resources using RM.

5 Conclusions and Future Directions

In this paper we have provided emerging trends in parallel and distributed computing with emphasis on Grid Computing. We have presented promises of Grid computing and discussed various types of Grids and their challenges. These systems enable the sharing of geographically distributed, autonomous resources owned by different organisations to support collaborative science and business applications. Designing and evaluating Grid resource managements policies that (a) allocate resources to competing user applications to meet their quality of service requirements and (b) provide mechanisms for service providers to profit from sharing their capabilities with external users is a challenging task. Although several Grid infrastructures such as TeraGrid in US and Garuda in India have been setup, it is difficult to carry out *repeatable* and *controlled* experiments on them for evaluating resource management policies as they are shared and no single user has full control over all the resources. In addition, special experimental infrastructures such as Grid 5000 [47] in France exist, but they are very expensive (financially) to setup and maintain. Hence, simulations appear to be one of the most feasible techniques for analyzing algorithms and policies for resource allocation.

We have discussed various Grid simulation tools available with special emphasis on GridSim, which is used by academic and industrial researchers and developers world-wide. We then discussed architectural elements of GridSim and its extensible features as demonstrated by various new tools developed around it. We illustrated the potential of GridSim for modeling and simulation through selected case studies drawn from industrial and academic users. IBM Research Lab has used GridSim for simulating Business Grids, University of Santiago (Spain) for parallel applications scheduling on European CrossGrid, University of Southern California for Gravitational Wave workflow application scheduling, and our own usage for pricing of reservations in Data Centers and their service revenue management.

In addition to service and utility-oriented Grids, other emerging parallel and distributed computing paradigms such as cloud computing offer several challenges and opportunities for modeling and simulation communities. They include (a) SLA-based monitoring and resource allocation techniques in VMs (virtual machines) and clouds, (b) negotiation protocols and polices for resource reservations, (c) economy models for service pricing and revenue management, (d) policies for power-aware "green" computing, (e) strategies for federation of services of clouds and Data Centers from different vendors, (f) coordinated resource provisioning in VO-based Grids, (g) mechanisms and algorithms for autonomic management of distributed systems, and (h) policies for peering among different types of Grids [51].

Software Availability

The GridSim toolkit software with source code can be downloaded from the project website:

http://www.gridbus.org/gridsim/

Acknowledgements

We would like to thank all contributors to GridSim. They include Manzur Murshed, Gokul Poduval, Chen-Khong Tham, Marcos Dias de Assuncao, Uros Cibej, Borut Robic, Agustin Caminero, Blanca Caminero, Carmen Carrion, Hui Li, and James Broberg. We thank users of GridSim especially case studies reported in Section 4. We also thank Mustafizur Rahman and Charity Lourdes for proof reading the paper.

This work is partially supported by the Australian Research Council and the Department of Innovation, Industry, Science and Research under Discovery Project and International Science Linkage Program grants respectively.

References

[1] G. F. Pfister. *In Search of Clusters.* Second Edition, Prentice Hall, 1998.

[2] M. Chetty and R. Buyya. Weaving Computational Grids: How Analogous Are They with Electrical Grids?. *Computing in Science and Engineering*, 4(4): 61-71, 2002.

[3] I. Foster and C. Kesselman (editors). *The Grid 2: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, USA, 2003.

[4] A. Oram (editor), *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, O'Reilly Press, USA, 2001.

[5] M. Rappa, "The utility business model and the future of computing services," *IBM Systems Journal*, vol. 43, no. 1, pp. 32-42, 2004.

[6] L. Kleinrock. A vision for the Internet. *ST Journal of Research*, 2(1):4-5, November 2005.

[7] V. Agarwal, G. Dasgupta, K. Dasgupta, A. Purohit, and B. Viswanathan. DECO: Data replication and Execution CO-scheduling for Utility Grids. *Proceedings of International Conference on Service Oriented Computing*, Chicago, USA, Dec. 4-7 2006.

[8] J. L. Albin, J. A. Lorenzo, J. C. Cabaleiro, T. F. Pena and F. F. Rivera. Simulation of Parallel Applications in GridSim. *Proceedings of the 1st Iberian Grid Infrastructure Conference*, Santiago de Compostela, Spain, May 14-16, 2007.

[9] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer. SETI@home: an experiment in public-resource computing. *Communications of the ACM*, 45(11):56-61, ACM Press, 2002.

[10] R. Bagrodia, R. Meyer, M. Takai, Y. Chen, X. Zeng, J. Martin, B. Park, and H. Song. Parsec: a parallel simulation environment for complex systems. *IEEE Computer*, 31(10):77-85, October 1998.

[11] M. Baker, R. Buyya, and D. Laforenza. Grids and Grid Technologies for Wide-Area Distributed Computing. *Software: Practice and Experience*, 32(15), pp.1437-1466, Wiley Press, USA, December 2002.

[12] I. Foster, C. Kesselman, J. Nick, S. Tuecke. Grid Services for Distributed System Integration. *Computer*, 35(6), 2002.

[13] W. H. Bell, D. G. Cameron, L. Capozza, A. P. Millar, K. Stockinger, and F. Zini. Simulation of dynamic grid replication strategies in OptorSim. *Proceedings of the 3rd International Workshop on Grid Computing*, Baltimore, USA, Nov. 2002.

[14] R. Buyya, D. Abramson, and J. Giddy. Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid. *Proceedings of the 4th International Conference and Exhibition on High Performance Computing in Asia-Pacific Region*, Beijing, China, May 14-17, 2000.

[15] L. Childers, T. Disz, R. Olson, M. E. Papka, R. Stevens, and T. Udeshi. Access Grid: Immersive Group-to-Group Collaborative Visualization. *Proceedings of the 4th International Immersive Projection Technology Workshop*, Ames, IA, USA, June 19-20, 2000.

[16] R. Buyya, D. Abramson, and S. Venugopal. The Grid Economy. In *Proceedings of the IEEE*, 93(3):698-714, March 2005.

[17] R. Buyya and M. Murshed. GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing. *Concurrency & Computation: Practice & Experience*, 14:1175-1220, Nov-Dec 2002.

[18] CACI Products Company. Simscript: a simulation language for building large-scale, complex simulation models. http://www.simscript.com, February 2008.

[19] A. Caminero, B. Caminero and C. Carrion. Designing an Entity to Provide Network QoS in a Grid System. *Proceedings of the 1st Iberian Grid Infrastructure Conference*, Santiago de Compostela, Spain, May 14-16, 2007.

[20] M. Cannataro and D. Talia. The Knowledge Grid. *Communications of the ACM*, 46(1):89-93, 2003.

[21] M. Bubak, M. Malawski, and K. Zajac. The CrossGrid Architecture: Applications, Tools, and Grid Services. *Grid Computing: First European Across Grids Conference (Santiago de Compostela, Spain)*, LNCS 2970, Springer, Germany, 2004.

[22] C. Dumitrescu and I. Foster. GangSim: A Simulator for Grid Scheduling Studies. *Proceedings of the 5th International Symposium on Cluster Computing and the Grid*, Cardiff, UK, May 9-12, 2005.

[23] I. Foster and C. Kesselman, Globus: A Metacomputing Infrastructure Toolkit, *International Journal of Supercomputer Applications*, 11(2): 115-128, Sage Publications, USA, 1997.

[24] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *Journal of High Performance Computing Applications*, 15(3):200-222, 2001.

[25] S. Graupner, J. Pruyne, and S. Singhal. Making the Utility Data Center a Power Station for the Enterprise Grid. *HP Labs Technical Report*, HPL-2003-53, Palo Alto, USA, 2003.

[26] W. Hoschek, J. Jaen-Martinez, A. Samar, H. Stockinger, and K. Stockinger. Data Management in an International Data Grid Project. *Proceedings of the 1st International Workshop on Grid Computing*, Bangalore, India, 2000.

[27] F. Howell and R. McNab. SimJava: A Discrete Event Simulation Package For Java With Applications In Computer Systems Modelling. *Proceedings of the 1st International Conference on Web-based Modelling and Simulation*, San Diego, CA, January 1998.

[28] O. Kang and S. Kang. Web-based Dynamic Scheduling Platform for Grid Computing. *International Journal of Computer Science and Network Security*, 6(5B): 67-75, May 2006.

[29] D. Klusacek, L. Matyska, and H. Rudova. Alea - Grid Scheduling Simulation Environment. Workshop on Scheduling for Parallel Computing. *Proceedings of the 7th International Conference on Parallel Processing and Applied Mathematics*, Poland, Sep. 9-12, 2007.

[30] K. Krauter, R. Buyya, and M. Maheswaran. A taxonomy and survey of grid resource management systems for distributed computing. *Software: Practice and Experience*, 32(2):135-164, February 2002.

[31] K. Kurowski, J. Nabrzyski, A. Oleksiak and J. Weglarz, Grid Scheduling Simulations with GSSIM, 3rd Workshop on Scheduling and Resource Management for Parallel and Distributed Systems, *Proceedings of the 13th International Conference on Parallel and Distributed Systems*, Hsinchu, Taiwan, Dec. 5-7, 2007.

[32] 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*, Brisbane, Australia, May 15-18, 2001.

[33] F. Messina, G. Novelli, G. Pappalardo, C. Santoro, and E. Tramontana. A QoS-Aware Architecture for Multimedia Content Provisioning in a Grid

Environment. *Proceedings of the 7th Workshop from Objects to Agents*, Catania, Italy, September 26-27, 2006.

[34] K. Seymour, A. YarKhan, S. Agrawal, and J. Dongarra, NetSolve: Grid Enabling Scientific Computing Environments**,** *Grid Computing and New Frontiers of High Performance Processing*, Grandinetti, L. eds. Elsevier, Advances in Parallel Computing, 14, 2005.

[35] A. Ramakrishnan, G. Singh, H. Zhao, E. Deelman, R. Sakellariou, K. Vahi, K. Blackburn, D. Meyers, and M. Samidi. Scheduling data-intensive workflows onto storage-constrained distributed resources. *Proceedings of the International Symposium on Cluster Computing and the Grid*, Rio, Brazil, May 14-17, 2007.

[36] 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) Conference*, Dallas, USA, Nov. 4-10, 2000.

[37] A. Sulistio, R. Buyya. A Grid simulation infrastructure supporting advance reservation. *Proceedings of the 16th International Conference on Parallel and Distributed Computing and Systems*, Cambridge, USA, November 9–11, 2004.

[38] A. Sulistio, U.Cibej, S. Venugopal, B. Robic and R. Buyya. A Toolkit for Modelling and Simulating Data Grids: An Extension to GridSim. *Concurrency & Computation: Practice and Experience*, Wiley Press, New York, USA, 2008.

[39] A. Sulistio, G. Poduval, R. Buyya, and C.-K. Tham. On incorporating differentiated levels of network service into GridSim. *Future Generation Computer Systems*, 23(4):606. 615, May 2007.

[40] Ns-2 network simulator. http://www.isi.edu/nsnam/ns, February 2008.

[41] J. B. Weissman, Grids in the Classroom, *IEEE Concurrency*, Volume 8, No. 3, July 2000, pp. 6-9.

[42] V. Welch, I. Foster, T. Scavo, F. Siebenlist, C. Catlett, J. Gemmill, and D. Skow. Scaling teragrid access: A testbed for identity management and attribute-based authorization. *TeraGrid 2007 Conference*, Madison, WI, USA, June 4-6, 2007.

[43] E. Elmroth and P. Gardfjall. Design and evaluation of a decentralized system for grid-wide fairshare scheduling. *Proceedings of the 1st International Conference on e-Science and Grid Computing*, Melbourne, Australia, December 2005.

[44] R. Buyya, D. Abramson, and J. Giddy. An Economy Driven Resource Management Architecture for Global Computational Power Grids. *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, Las Vegas, USA, June 2000.

[45] EU Data Mining Grid, http://www.dataminінggrid.org, February 2008.

[46] R. Buyya and S. Venugopal. The Gridbus Toolkit for Service Oriented Grid and Utility Computing: An Overview and Status Report. *Proceedings of the 1st International Workshop on Grid Economics and Business Models*, Seoul, Korea, April 23, 2004.

[47] F. Cappello and H. Bal. Toward an International Computer Science Grid. *Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid*, Rio, Brazil, May 14-17, 2007.

[48] A. Sulistio, K. H. Kim and R. Buyya. Using Revenue Management to Determine Pricing of Reservations. *Proceedings of the 3rd International Conference on e-Science and Grid Computing*, Bangalore, India, Dec. 10-13, 2007.

[49] Amazon, "Amazon EC2, Amazon Elastic Compute Cloud, Virtual Grid Computing", http://aws.amazon.com/ec2, February 2008.

[50] H. Liu, V. Bhat, M. Parashar and S. Klasky, An Autonomic Service Architecture for Self-Managing Grid Applications, *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, Seattle, WA, USA, November 2005.

[51] M. Dias de Assunção, R. Buyya and S. Venugopal. InterGrid: A Case for Internetworking Islands of Grids. *Concurrency and Computation: Practice and Experience*, ISSN: 1532-0626, Wiley Press, New York, USA, 2008.