An OSCI TLM-2.0 based Platform for Grid Computing Simulation

Mina Zolfy, Ziaddin Daie Koozehkanani
Faculty of Electrical and Computer Engineering
University of Tabriz
Tabriz, Iran

Zainalabedin Navabi
Electrical and Computer Engineering Department,
Faculty of Engineering, Campus #2
University of Tehran
Tehran, Iran

ABSTRACT
In this paper a grid computing simulation platform, is implemented based on the OSCI TLM-2.0 standard. TLM-2.0 standard, offered as a layer on top of the SystemC library, is becoming a key solution in system level design. The concurrency facility of TLM-2.0 on one side, and its ease of use on the other side, makes it an ideal choice for modeling and simulation of distributed systems. These days, one of the most important subjects in distributed system researches, is the corresponding scheduling algorithms. The simulation platform, implemented in this work can be configured easily for any scheduling algorithm and provides an opportunity for wide exploration in related design space.

Keywords
Grid computing, Modeling, Simulation, Transaction level modeling, SystemC, System level design

1. INTRODUCTION
The rapid growth of computational requirements increases the role of middleware technologies in current researches. Although, the processors are developing so fast, these systems cannot meet the software engineer requirements due to the high growth of computing. On the other hand, improvement of computer networks decreases the connectivity latency of individual connected nodes. Considering the new networking technologies not only increase the speed of networks, but also make the established connections more reliable and secure.

The above discussed advancements- utilizing fast computers on one hand and fast and reliable networks on the other hand- encouraged software engineers to distribute the processes on the network of computers. Accordingly,, distributed systems appeared, in which a group of independent but connected computers would be responsible for a common computing task and act as a single powerful computing engine.

The widespread availability of internet all over the world, and the number of available individual computers, connected to the internet were the key motivations in introducing grid computing systems[1]. A grid computing system is an example of distributed systems, in which, the existent facilities are being used, i.e., in a grid system, processor nodes are the existing computers that are connected to each other with an established network – like internet or intranet. One of the most important related challenges to grid computing is its modeling and simulation.

In this paper, a grid computing simulation platform is presented, using which designers would be able to model their design and explore the design space. The illustrated simulation platform is implemented using the TLM-2.0 standard of Open SystemC Initiative (OSCI)[2]. The Open SystemC Initiative works on the definition and standardization of SystemC, as a system level language.

Recently, . In the Transaction-Level Modeling (TLM), which is frequently discussed in system-level design community, the details of communication and computation components are separated from each other.

OSCI has released two versions for the TLM standard. In TLM 1.0, channels are the basic communication elements, while in TLM 2.0 the scheme of the sockets has been introduced as the communication infrastructure. This new abstraction level brings some research challenges into hardware design communities [3][4].

In this paper the TLM-2.0 standard has been used in which the core interfaces pass transactions between initiators and targets. A module is named an initiator when it can initiate transactions, i.e, when it creates new transaction objects and passes them by calling a method of one of the core interfaces. A target is a module considered as the final destination for a transaction. Sockets on both sides assist these connections, i.e,. the initiator module sends transactions via its initiator socket and the target module receives transactions with the target socket. The TLM-2.0 classes are layered on top of the SystemC class library.

As discussed above, in this paper we deal with a modeling methodology for some simple grid computing structures which provide a flexible simulation environment. First in section 2 some of the similar projects in grid computing simulation are introduced. Then section 3illustrates the presented simulation platform. Section 4 presents the result of an experiment on the simulation platform. In section 5, the advantages of proposed modeling methodology is discussed. Finally, section 6 deals with concluding remarks anmd future works.

2. RELATED WORKS
In this section, we deal with the specification and architecture of Grid simulation tools[5]. *GridSim* [6] is a toolkit for modeling

resources and simulating network connectivity with different configurations. It is implemented as a Java API and is part of the GridBus [7] Project. The GridSim toolkit supports the capability for simulating different classes of heterogeneous resources, users, applications and resource brokers. *OptorSim* [8] is also a Java-based simulator that is mainly used to test data replication strategies in a generic simulation environment for grid applications. OptorSim has a modular architecture which is directly based on the DataGrid project architecture. *SimGrid* [9] is another toolkit which provides core functionalities for the simulation of distributed applications in heterogeneous distributed environments. The goal of SimGrid is to support research in the area of distributed and parallel application scheduling on distributed computing platforms in the range of workstations to grids. The toolkit is a C API published under the GNU LGPL. *JFreeSim* [10], is a grid simulation tool based on the MTMSMR (Multi Task, Multi Scheduler, Multi Resource) model. JFreeSim provides a set of universal and extensible modular libraries. *NSGrid* [11] is an ns2-based grid simulator which supports modeling of different types of grid components, resources and network interconnections. NSGrid can simulate both CPU- and data-intensive jobs using network-aware scheduling algorithms. *GrenchMark* [12] is a Python-based framework for synthetic grid workload generation and submission. GrenchMark centers on generating synthetic grid workloads, which can then be used on a real grid infrastructure or on the simulator. *GridNet* [13] is a simulator used to analyze data replication strategies in grids. The GridNet toolkit focuses on simulating data replication and the underlying network instead of simulating distributed and parallel applications. *Bricks* [14] allows the simulation of various behaviors such as network topology of client server, resource scheduling algorithms and processing schemes for network and servers. It is an event driven simulator built as a framework of a set of replaceable components. *ChicSim* [15] (The Chicago Grid Simulator) is a grid simulator developed using the Parsec simulation language. ChicSim is useful for evaluating scheduling and replication algorithms.

3. SIMULATION PLATFORM

The abstract diagram of the platform, implemented for grid simulation in this work, is shown in Fig. 1. A grid computing model must cover some requisites, for being simulated in the presented platform. Such a model should use a central broker for managing the clients' requests on one side and the resources' services, on the other side. In other words, every request must pass through the central broker, in order to being processed. In the presented platform, each grid model consists of three main modules; a *ClientsGroup*, a *Broker* and a *ResourcesGroup*

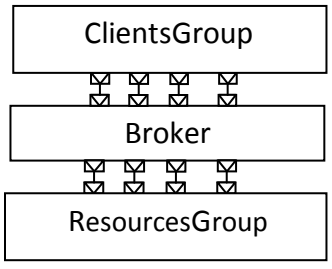


Figure 1- Abstract diagram of entire Platform

ClientsGroup is a module including all of the system's clients. Similarly, *ResourceGroup* includes all of the system's resources. Finally, the *Broker* is a module which manages the entire system. This module is responsible for assigning resources to clients and establishing the required connections. The platform modules are explained in more details in the following sections.

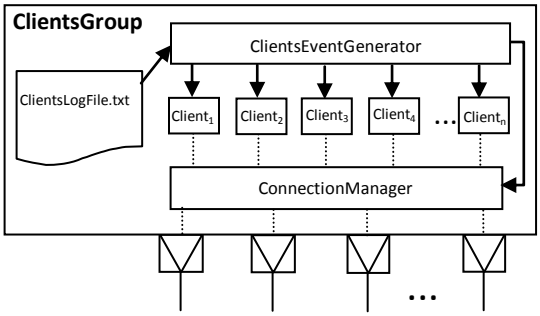


Figure 2- Block diagram of ClientsGroup module

3.1 ClientsGroup

In the SystemC core, presented with OSCI, all module instantiations and bindings are performed in the elaboration phase accordingly; no instantiation and dynamic binding are allowed during the simulation. Therefore, a pool of instantiated *clients* is considered inside the *ClientsGroup* module. Fig. 2 shows the components of *ClientsGroup* module. The major components are discussed below.

3.1.1 ClientsLogFile

ClientsLogFile is a text file, in which the simulation input data – about client's requests – is declared. Each line of this file includes a single request using the format shown below:

RequestID	ClientID	ClientName	RequestTime	ServiceTime

3.1.1.1 RequestID

RequestID points to the the unique ID of the request.

3.1.1.2 ClientID

The unique ID of the client, sending the request is shown by the Client ID. A client may send several requests during the simulation.

3.1.1.3 AppliedTime

The AppliedTime field indicates the moment in which the request in sent. The value of the AppliedTime field includes the time differences between preceding requests.

3.1.1.4 ServiceTime

The ServiceTime field indicates the length of the requested service.

3.1.2 ClientsEventGenerator

This ClientsEventGenerator component is implemented with a SC_THREAD SystemC process. ClientsEventGenerator reads the requests from ClientsLogFile and apply them in the determined

timing order. Accordingly, ClientEventGenerator activates a client and oblige it to send the related service request.

3.1.3 ConnectionManager

The ClientsGroup module includes some instances from client module and also some dedicated TLM-2.0 initiator sockets. ConnectionManager is responsible for establishing connections between client modules and initiator sockets.

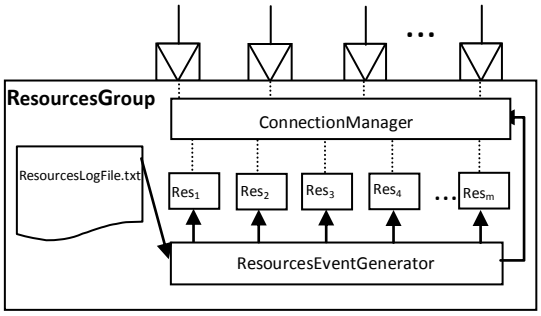


Figure 3- Block diagram of ResourceGroup module

3.2 ResourcesGroup

Similar to ClientsGroup module, ResourcesGroup includes a pool of resource instances and a number of dedicated sockets. However, they are different in terms of socket types. In ResourceGroup, the sockets are some target-sockets which get the service requests from the broker. What follows discusses the main components of ResourcesGroup.

3.2.1 ResourcesLogFile

ResourcesLogFile is a text file including the simulation data regarding the resources availability. Each line of ResourcesLogFile contains the fields shown below:

ServiceID	ResourceID	ResName	StartTime	StopTime

3.2.1.1 ServiceID

ServiceID is the unique ID of the available service.

3.1.1.2 ResourceID

The unique ID of the resource, which is ready to serve a request is shown by ResourceID. A resource may offer several services during the simulation.

3.1.1.3 StartTime

The StartTime field indicates the moment in which the resource is ready to serve a request of a client. The value of StartTime field includes the time differences between preceding ready services.

3.1.1.4 StopTime

The StopTime field indicates the moment in which the resource stops its grid service.

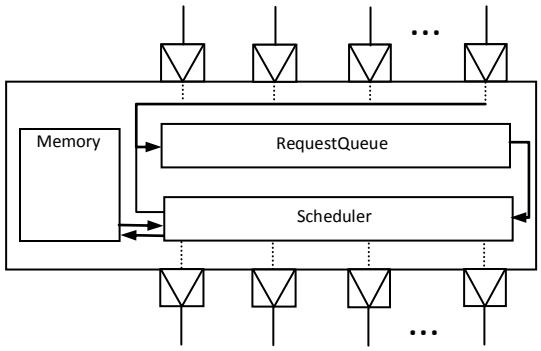


Figure 4- Block diagram of Broker module

3.3 Broker

The Broker module is the manager of entire system. On one hand, all of the clients send their requests through initiator sockets of ClientsGroup module, and on the other hand, resources send their readiness, to the Broker. The Broker decides about the proper resource for any request, based on its scheduling algorithm and sends the corresponding request to its proper resource. The main components of the broker module are described below:

3.3.1 Memory

The Memory component stores and retrieves the required data of the available resources and also the information about currently scheduled requests.

3.3.2 RequestQueue

The RequestQueue is a FIFO (First In First Out) queue, which contains the unscheduled requests, received from the ClientsGroup.

3.3.3 Scheduler

The scheduler pops requests from the RequestQueue. Then considering the available resources and based on the implemented scheduling algorithm, allocates a resource to the popped requests. If the scheduling process can not been done successfully, the scheduler pushes the request back to the RequestQueue.

4. EXPERIMENTAL RESULT

This section presents the result of an experiment on the presented simulation platform. In this experiment a simple scheduling algorithm, which is FCFS (First Come First Serve) has been implemented for the implementation of the scheduling process in Broker module.

Figure 5 shows the input files of the experiment. The ClientsLogFile.txt includes the information of the service requests which is sent by the clients to the broker during the experiment and ResourcesLogFile.txt includes the information of the available system resources in the experiment.

0	0	C0	0	80
1	1	C1	0	20
2	2	C2	10	40
3	3	C3	20	150
4	1	C1	40	30
5	4	C4	10	100
6	5	C5	20	40
7	4	C4	30	50

a) ClientsLogFile of the experiment

0	0	R0	0	50
1	1	R1	0	200
2	2	R2	40	250
3	0	R0	60	100
4	3	R3	70	80

b) ResourcesLogFile of the experiment

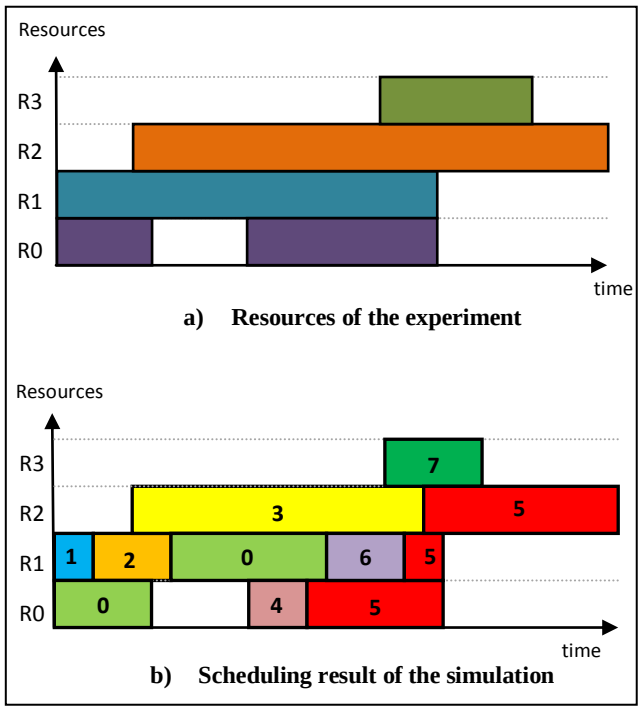
Figure 5 – Inputs of the experiment

In this experiment, 6 different clients have been connected to the broker. As shown in Fig. 5.a , C1 and C4 clients have two requests during the simulation while other clients have only one request. Therefore the total number of service requests in this experiment is 8. On the other hand, 4 individual resources have announced their readiness for providing service. As shown in Fig 5.b, the R0 resource is disconnected for a while, but again becomes available after 100 time unit.

Considering the concurrency facilities provided in SystemC-and therefore in OSCI TLM-2.0- the implemented simulator analyzes the ResourcesLogFile of the experiment –shown in Fig 5.a- and recognizes the availability of the resources shown in Fig 6.a. In this figure the horizontal axis of the chart represents the simulation time and the vertical axis illustrates the individual resources. In a real grid computing system, the resources get disconnected almost randomly, i.e., their disconnection times are not identified in advance. In the presented simulator, in order to have more determined experiments, the disconnection time of the resources must be given in the corresponding input file. But these information couldn't be used in scheduler algorithm, because in a real system such disconnections couldn't be predicted.

As mentioned above, in this experiment the scheduler is using a FCFS algorithm and no priority is defined for the coming requests. On the other hand, we have homogeneous resources in this experiment, so each available resource could serve any request.

The simulation result of this experiment is shown in Fig 6.b. This figure represents the resource scheduling for the received requests. In Fig. 6, the horizontal axis shows the simulation time while the vertical axis shows the individual resources. As shown in this figure, a number of requests are scheduled only once while others are scheduled twice or even more. For example the request with ID equal to 6 is scheduled only once. According to the corresponding ClientsLogFile (Fig 5.a), this request is sent from client named C5, and as shown in Fig 6.b it is served with R1. But Request number 5, which is sent from C4 has been scheduled three times. The reason for these rescheduling, is the disconnection of related resource during the scheduled service. When a resource gets disconnected from the broker, the scheduler component of the broker pushes the uncompleted request back to the RequestQueue. Therefore this uncompleted request (for example request 0 or 5 in our experiment) will be scheduled later.



a) Resources of the experiment

b) Scheduling result of the simulation

Figure 6 – Simulation result of the experiment

5. ADVANTAGES

In this section, advantages of the proposed simulator are discussed.

5.1 Meeting All Essential Requirements

One of the most important advantages of the proposed simulator is its flexibility, which facilitates of the assembling all essential requirements [16] of a grid computing simulator. In the rest of this section, some of these requirements and the corresponding solutions are explained.

5.1.1 Multi-tasking IT resources

Most of the resources in a grid computing system are preemptive multi tasking resources. In these resources, submitted tasks go through processing queue, which includes all ofthe tasks that are being processed.

The convenience concurrency of SystemC kernel enables designers to implement multi-tasking resources with less

simulation overhead. For implementing such resources, using sc_fifo and its sc_event members simplifies the related design process.

5.1.2 Task parallelization

A grid computing task may be parallelizable. Parallelization could be of two types: Embarrassingly parallel tasks can be simply split up to as many portions as available resources. The second type of parallel tasks is more constrained and consists of a specific number of parallel paths, regardless of the number of available resources.

Implementing both of these parallelization types, are feasible in proposed simulation platform. For this purpose, the C++ code of the scheduling algorithm must be manipulated. For example, in the parallelization of the second type, the scheduler needs to split a popped task and send it to a number of resources.

5.1.3 Heterogeneous resources

A grid computing architecture may include heterogeneous resources.

In the proposed simulator, the only obligation for implementing this feature is adapting the memory component and the scheduling algorithm, in order to take account of resources category in the related scheduling decision.

5.1.4 Resource scheduling

The grid simulator must be able to model the scheduling policies used by resource brokers, to determine which resource should process an arriving task. Typical policies are task dispatch rules and are discussed below.

5.1.4.1 Load leveling

In the Load leveling method, the task is sent to the least utilized resource to be able to balance utilization across all resources compatible to that task.
Implementation of this rule in the proposed simulator is achievable by forcing the resource broker to keep track of services that all resources have already done. For this purpose, it is enough to update a data array whenever a successfully completed transaction is sent to the broker on backward path.

5.1.4.2 Greedy

The task is sent to the resource using which it can be completed faster: a function of its processing power and current utilization.
Applying this rule is as easy as implementing the decision algorithm of resource scheduler in a greedy form (According to the described parameters).

5.1.4.3 Round robin

In the Round robin method, the task is sent to the next compatible resource in a defined sequence.

For the implementation of this method, the scheduler component of the broker must keep track of defined sequence using some SystemC sc_fifo objects.

5.1.4.4 Threshold-based

In the Threshold-based method, then tasks are sent to a preferred resource until a certain performance parameter threshold is passed.
Implementation of this rule could be done similar to the greedy one, but it will be enough for the selected resource to reach the define threshold.

5.1.5 Resource provisioning

The ability of resource provision is another feature to be simulated. The provisioning policies can be either based on a calendar or based on a more dynamic policy which monitors workload arrivals and resource usage and reacts accordingly. Simulation of these provisioning policies in conjunction with the resource scheduling policies would allow grid designers to determine whether the respective policies are aligned and consistent with each other and if they provid the desired grid performance in terms of resource availability, workload throughput and processing times.

As described earlier, the proposed methodology supports dynamic configuration of the grid computing structure using pool of predefined components during the elaboration phase. Accordingly calendar based and dynamic provisioning policies can be implemented.

5.1.6 Non-programmer user interface

The expected users of the simulation are grid designers and consultants, who are unfamiliar with simulation languages. Therefore, the simulator must provide a graphical modeling environment and user interface. It should also allow the grid designers to write new resource scheduling and provisioning policies and incorporate them into the simulation.

Although in our methodology all coding will be done in C++, but for the junior users who are not familiar with the required concepts, it could be possible to use some ready to use components through a GUI. As one of the most important features in the recent TLM standard is its interoperability, designers will be able to use such components in conjunction with other modules and systems. But this solution will restrict the amenity of the entire methodology.

5.2 Simulation speed

Simulation time is one of the most important factors in simulation method evaluation. In the proposed methodology, the implemented C++ Class, models the systems directly and there is no interface tool to decrease simulation speed. All other GRID computing simulators spend some time for compiling, analyzing

and simulating each model. But using TLM-2.0, each model is being declared with a C++ code and its simulation includes only compile and run time of the programming code. Therefore using TLM-2.0 will decrease total simulation latency.

5.3 Extensibility

As mentioned in section 1, OSCI TLM-2.0 standard consists of open source C++ classes. Therefore, all designers can manipulate and extend those classes based on their own modeling requirements and goals.

Moreover, most of the modeling experiences require exploring design space with several parameters and different implementations. TLM-2.0 allows designers to define all design aspects such as resource scheduling algorithms, data base managements and so on in each unobligatory way. But in other simulators, only a restricted set of parameters can be defined. In addition in those simulators for changing any design feature, the simulation kernel must be changed or adjusted.

5.4 Ease of use

Programming is one of the inevitable bases of computer engineering field. Accordingly, all designers are familiar with at least one object oriented programming languages.

Using our methodologies, the only required capability for each designer is the ability of manipulating some program classes for defining and verifying an optional design in every desired way. Therefore designers don't have any problem in their design space exploration.

5.5 Availability

Another advantage of this methodology is its availability. TLM-2.0 library is an open source package and is available via systemc.org.

6. CONCLUSION

In this work, the TLM-2.0 based simulation platform for grid computing systems is implemented. Because of the wide spread configuration space of Grid computing architecture, none of the available simulators support all of its possible features. While OSCI TLM-2.0 standard is presented with a C++ library on top of SystemC class library, it takes advantage of the flexibility of the entire C++ language. The correctness and flexibility of the proposed simulation platform is approved with the configured experiment. Although the experiment includes homogeneous resources, it can be easily generalized for heterogeneous resources. The proposed platform discussed in this paper, presents a new idea leading to some original research works in near future.

7. REFERENCES

[1] Rajkumar Buyya, Srikumar Venugopal, "A gentle introduction to grid computing and technologies", Computer Society of India,CSI Communications, 2005

[2] John Aynsley, Doulos, "OSCI TLM-2.0 Language reference manual" by the Open SystemC Initiative (OSCI),2009

[3] M. Zolfy, Z. D. Koozehkanani, M. Hashemi, and Z. Navabi,"Test Strategi in OSCI TLM-2.0", Proceeding of East West Design and Test Symposium, pp. 438-441, Russia, 2009

[4] M. Zolfy, Z. D. Koozehkanani, L. Mohammadkhanli, and Z. Navabi,"Investigation of OSCI TLM-2.0 Employment in Grid Computing Simulation", Proceeding of VALID 2010France, 2010

[5] Y. El-khatib,"Survey of Grid Simulators ,Network-level Analysis of Grid Applications," Europe-China Grid Internetworking, European Sixth Framework STREP

[6] R. Buyya and A. Sulistio,"Service and Utility Oriented Distributed Computing Systems:Challenges and Opportunities for Modeling and Simulation Communities",Proceeding of Simulation Symposium 2008, pp. 68-81, Ottawa, USA

[7] R. Buyya and S. Venugopal,"The Gridbus Toolkit for Service Oriented Grid and Utility Computing: An Overview and Start Up report", Grid Economics and Business Models, pp.19-66, 2004

[8] G. Belalem and Y. Slimani, "Consistency Management for Data Grid in OptorSim Simulator", Proceeding of International Conference on Multimedia and Ubiquitous Engineering, pp. 554-560, Seoul, 2007,

[9] M. Quinson, "SimGrid: A Generic Framework for Large-Scale Distributed Experminets", Proceeding of IEEE 9th International conference on Peer-to-peer computing 2009, pp. 95-96, Seattle, USA

[10] H. Jin, J. Huang, X, Xie, and Q. Zhang, "JFreeSim: A Grid Simulation Tool Based on MTMSMR Model", Proceeding of APPT 2005, pp. 332-341, 2005

[11] B. Volckaert, P. Thysebaert , M. De Leenheer, F. De Turck, B. Dhoedt, and P. Demeester, "On the Use of NSGrid for Accurate Grid Schedule Evaluation". Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA '04), 2004, USA, Volume 1, pp. 200-206, CSREA Press 2004, ISBN 1-892512-23-8.

[12] A. Iosup and D. Epema, "Grenchmark:A Framework for Analyzing, Testing and Comparing Grids", Proceeding of International Symposium on Cluster Computing and the Grid, pp. 313-320,2006

[13] H. Lamehamedi, Z. Shentu, B. K. Szymanski, and E. Deelman ,"Data Replication strategies in Grid Environment ". Proceedings of the 5th International Conference on Algorithms and Architecture for Parallel Processing (ICA3PP 2002), pp. 378-383, China, pp. 378-383, 2002.

[14] S. Naqvi and M. Riguidel, "Grid Sequrity Services Simulator – A Simulation Tool for Design and Analysis of Grid Security Solutions", Proceeding of the 1st IEEE International Conference on e-science and Grid Computing, Australia, pp. 421-428, 2005

[15] K. Ranganathan and I. Foster, "Computation Scheduling and Data Replication Algorithms for Data Grids ",Kluwer Academic Publishers, pp. 359-373,2004

[16] S. Bagchi, "Simulation of Grid computing infrastructure:Challenges and solutions",Proceeding of the 2005 Winter Simulation Conference, pp. 1773-1780.