Resource Topology Aware GridSim: A Step Ahead

Attiqa Rehman[1] Kalim Qureshi[2,*] Paul Manuel[3] Haroon Rashid[1]

[1] Department Computer Science
COMSATS Institute of Information Technology
University Road, Tobe Camp, 22060 Abbottabad, Pakistan
{atti_dimple, haroon}@ciit.net.pk

[2] Department of Mathematics and Computer Science
Kuwait University
Safat, 13060, State of Kuwait
qureshi@sci.kuniv.edu.kw

[3]Department of Information Science
Kuwait University
Safat, 13060, State of Kuwait

Received 15 January2008; Revised 30 April 2008; Accepted 1 May 2008

Abstract. Grid computing is a giant leap forward in high performance distributed computing, ushering in a new era of commodity computing. The biggest challenge in this field is the heterogeneity of applications, resources and administrative domains controlling them. New algorithms of job scheduling and load balancing require stringent evaluation involving controlled configurations. Such a controlled environment is very difficult to provide in a real grid environment. Grid simulators can simulate real grid systems as well as provide configurable environments for repetitive experiments. GridSim [1] is a popular grid simulator originally proposed to evaluate resource management and job scheduling schemes. In this paper we propose an enhancement in GridSim architecture "Resource Topology Aware (RTA) GridSim" to increase its potential in resource management, job scheduling and load balancing. Our measured results show that the proposed enhancement improves the flexibility of GridSim by accommodating resource heterogeneity. In addition, it minimizes communication overhead and response time by making load balancing decisions locally through resource topology awareness.

Keywords: Grid computing, distributed computing, load balancing, resource management, GridSim architecture, and Resource topology aware GridSim.

1 Introduction

Booming advancements in computer hardware and high speed communication instigate research in High Performance Distributed Computing (HPDC). HPDC combining computational power of multiple resources has embarked upon new horizons which is Grid Computing. Grid [2] is a flexible, secure and coordinated resource sharing algorithm among dynamic collections of individuals, institutions, and resources termed as Virtual Organization (VoG).

Grid computing has brought new challenges [3] of resource management, job scheduling, load balancing, fault tolerance and privacy such as

- which resources will form VoG,
- who will be responsible for scheduling jobs on grid,
- how to meet grid user's computational and storage requirements,
- how to solve the privacy and security issues at resource and user level,
- how to unite resources from different administrative domains in a VoG,
- how to find appropriate resources meeting grid user requirements.

Traditional HPDC solutions to these problems cannot be scaled up to grid because of three reasons [4]. First resources, applications and users forming a VoG are heterogeneous. Second the resources are controlled by different administrative domains having their own policies. Third the VoG is world-wide involving internet

[*] Correspondence author

instead of enterprise local intranet. Due to these reasons decentralized and scalable algorithms are required for the grid, especially for job scheduling and load balancing [4].

Due to inherent dynamicity, flexibility and heterogeneity, evaluating novel job scheduling and load balancing algorithms in real grid environment is very difficult. Practical experiments with online applications on real resources and particularly reproduction of experimental configurations are difficult due to the following reasons:

(i) the fluctuations of resource and application availability due to multiple administrative domains
(ii) the difficulty in providing specific and controlled configuration of resources to applications and users without time restriction in extensive computational process
(iii) the complexity due to the world-wide span of VoG and involvement of the Internet.

Grid simulators [5] are effectively utilized in many research endeavors [4, 6] to experiment new grid algorithms. Some of the notable simulators are Bricks, SimGrid, GangSim and GridSim [1]. One of the first grid simulators proposed to test grid scheduling algorithms is Bricks [7]. This is an event-driven performance evaluation system for testing scheduling policies based on a global computing system framework and replaceable components. The framework consists of clients, servers and networks. Algorithms and programming modules for resource scheduling, working schemes of networks and servers and their topologies can be simulated here.

SimGrid [8] is a popular discrete event based grid simulator to study single client multi server scheduling in distributed, dynamic and heterogeneous environments. SimGrid provides abstractions to build a customized simulator corresponding to resources, applications and other infrastructure. SimGrid is used to experiment static and dynamic scheduling and load balancing algorithms.

GangSim [9] is an enhancement of Ganglia Monitoring toolkit for VoGs. It is designed to simulate grid computing with the abundance of computational and storage resources. It is a policy-based resource management simulator, where job scheduling and load balancing decisions are made by cooperation of site, VoG and other VoG policies. It is used to simulate workloads and conflicting resource management policies.

GridSim [1] is a famous Java based grid simulator with a clear focus on Grid economy [10] involving producers and consumers of computational and data storage resources bargaining with each other as in a real marketplace. GridSim works on top of a SimJava [1] discrete event simulation framework. This simulator is based on entities who are user and broker. A broker is one who bargains among user and resources. These entities can have customized characteristics. GridSim is generally used to study economy based scheduling decisions taken by distributed brokers with competing requirements of budget and deadlines.

We have chosen GridSim as a grid simulator, and opted to enhance its potentials because of its following inherent advantages:

(i) it allows new grid scheduling and load balancing algorithms to be easily incorporated
(ii) it provides a wide range of features for grid simulations which closely match the performance of real grid.

These features include a robust abstraction for computational and data storage resources with variable performance, abstractions for time and space shared scheduling, abstraction for a wide range of network devices and services, extendable abstractions for managing failure of resources, a system for workload traceability from real high performance supercomputers and abstractions for advance reservation of resources. Caminero et. al. [14] have also expressed a similar view for the selection of GridSim as an optimal simualtor to test new job scheduling algorithms.

According to [5] GridSim is higher-level simulator than SimGrid in terms of abstraction flexibility. However, GridSim does not consider network topology. We believe that GridSim should be expanded to include resource topology as well. We propose an enhancement “resource topology aware” in GridSim architecture to include machine and Processing Element (PE) as new active entities to extend the resource topology model. This enhancement makes GridSim aware of resource topology and consequently increases the flexibility and scalability of GridSim. We have conducted a load balancing experiment to evaluate the proposed enhancement.

The rest of the paper is organized as follows. Section 2 discusses the GridSim architecture, interaction protocol and scheduling policy. Section 3 introduces RTA-GridSim architecture, interaction protocol and scheduling policy. Section 4 shows the experimental study and section 5 contains concluding remarks.

2 GridSim Architecture

GridSim provides an abstraction to create resources, users, applications, resource brokers and schedulers [1] with customized capabilities. It can be used to simulate job scheduling and resource management algorithms. GridSim is built on top of Java Virtual Machine JVM layer and SimJava layer [1].

GridSim extends the SimJava concept of entity to provide higher-level abstraction for user, broker, resource, grid information service, input and output GridSim entities [1, 11]. The grid users can vary in the context of number and types of jobs, scheduling strategy, time zone and grid economy factors. Broker and user entities are

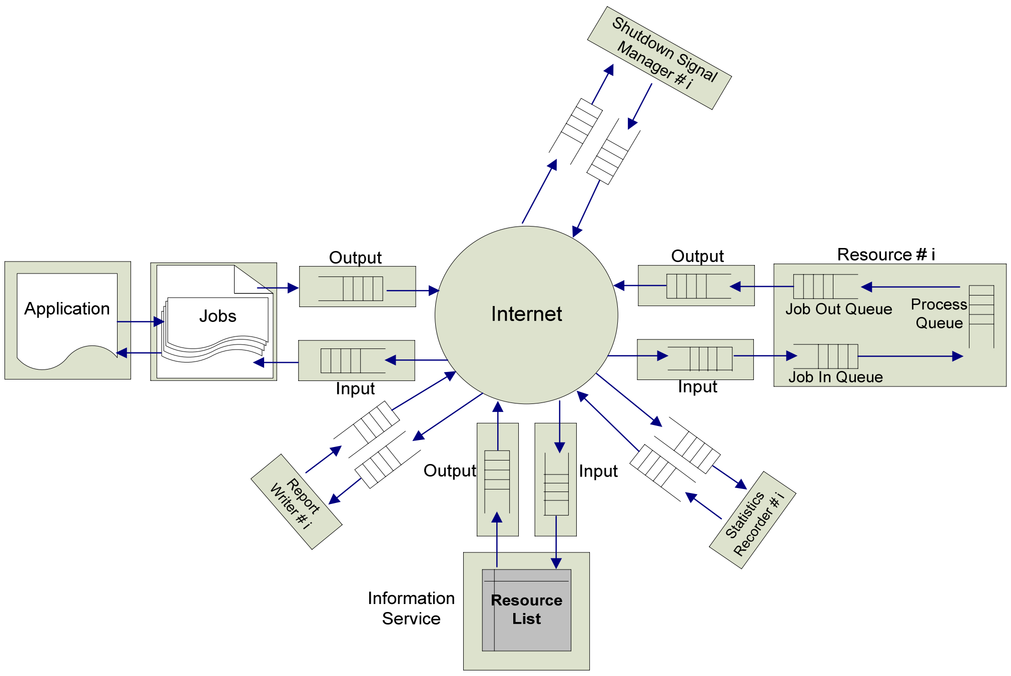


Fig. 1 A flow diagram in GridSim based simulations [1]

in one-to-one correspondence with each other. Every job of user goes through the appropriate broker entity. The broker entity in communication with other broker entities manages the tradeoffs of scheduling policy.

Resource is the main entity to simulate user job executions in GridSim. Resources vary in terms of number of processors, cost and speed of processing, internal process scheduling policy, local load and time zone. Speed of processing elements is measured in Million of Instructions per Second (MIPS). Grid information service (GIS) entity registers resources and provides their list on demand. Input and output are wrapper abstractions of SimJava entities handling information flow. Fig 1 shows the interaction of entities in GridSim simulation.

2.1 GridSim Interaction Protocol

The GridSim interaction protocol is based on SimJava discrete event simulation. GridSim entities handle service request and service delivery events [1]. An entity can schedule an event for itself (an internal event) or for another entity (an external event). Events are identified by unique entity IDs assigned to them. An event can be *synchronous* (source entity waits till the destination entity completes the requested job) or *asynchronous* (source entity raises events and proceeds to other actions instead of waiting).

Following are the steps involved in GridSim interaction protocol [1]:

1. All the entities are created and initialized.
2. Grid user entity sends a Gridlet (grid jobs) list to broker entity.
3. All the resource entities are registered with GIS.
4. Broker entity demands the resource list from GIS.
5. Broker entity demands the resource characteristics object from the registered resources present in the list.
6. Broker raises an asynchronous event to submit gridlets to the appropriate resource entity as suggested by scheduling policy.
7. Resource predicts the completion time of gridlet and schedules an internal event at the smallest completion time of a gridlet in the execution list.
8. After the execution of the gridlet it is sent back to the broker.
9. Broker sends the gridlet back to the user.
10. When all the gridlets are completed the user sends a finished experiment event to the user.
11. After receiving the finished experiment event the user entity sends an event save statistics to Grid Statistics entity.
12. A shutdown event is sent to the Grid Shutdown entity.

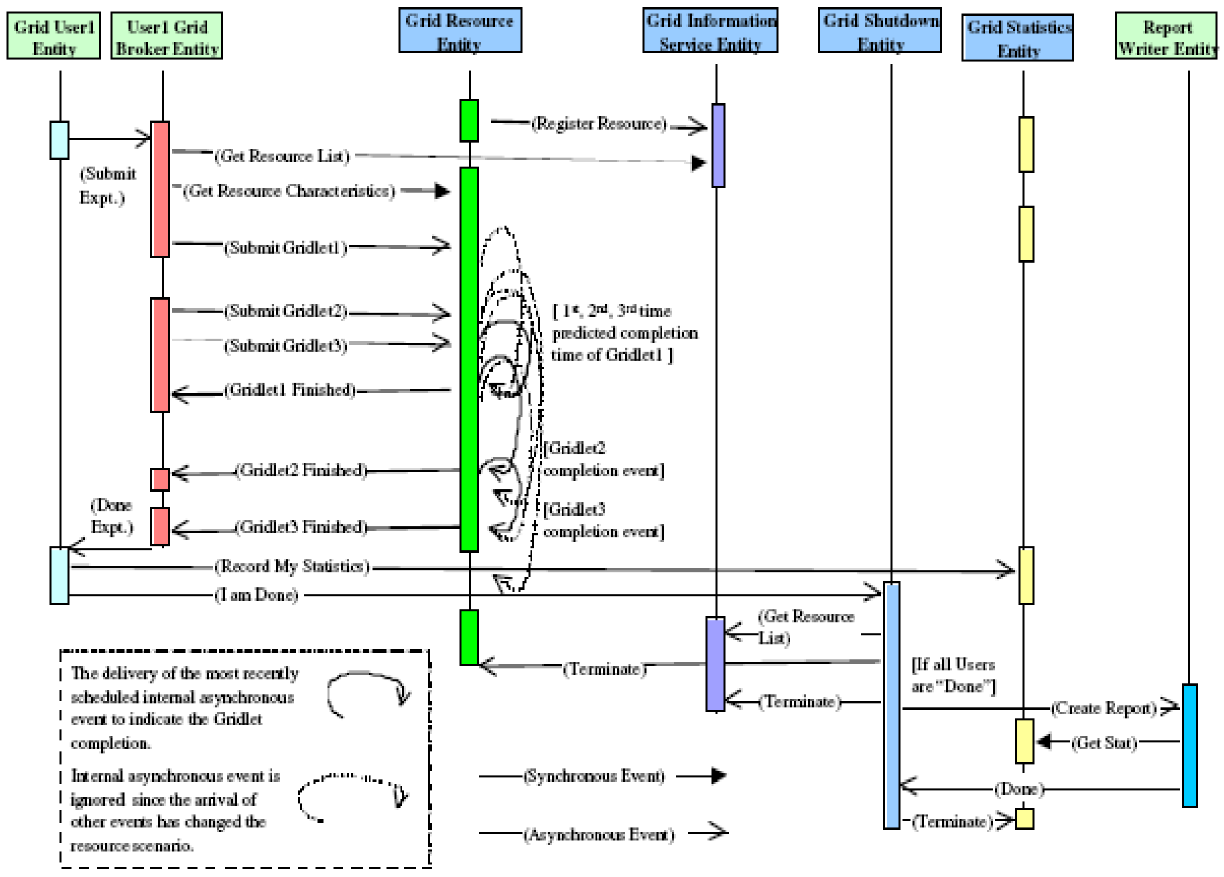


Fig. 2 Event diagram showing interaction of a time-shared resource with other entities [1]

13. Shutdown entity sends a terminate event to the resource entities after getting the resource list from GIS.
14. Shutdown entity sends a write report event to Report writer entity.
15. The report writer entity gets statistics from statistics entity. After finishing, report writer sends a finished event to shutdown entity.
16. Shutdown entity sends a finished event to the grid statistics entity and simulation is ended.

Fig. 2 shows the GridSim interaction protocol. The resource entity in GridSim can be time-shared using a round robin strategy or space-shared using first come first served policy. The time shared policy is implemented in GridSim as described in Fig. 3.

3 Topology Aware GridSim Architecture

Grid is a worldwide distributed system which has huge resources, applications and network heterogeneity [2]. A simulator designed to mimic the real grid should have enough abstractions to cover the inherent heterogeneity. In Grids, a resource is a reusable entity that is employed to fulfill a job or resource request. It could be a machine, network, or some service that is synthesized using a combination of machines, networks, and software.

In [12], parallel and distributed systems are characterized as multi-processor and multi-computer systems. Multiprocessor systems, also termed as Parallel Processing Systems (PPS) are defined as computer systems that have several processors sharing a single set of peripherals including the memory. These processors are not autonomous. Multi-computers, also known as Distributed Computing Systems (DCS), consist of several autonomous processors having their own operating system, with their own job scheduling and load balancing policies.

This classification is not exhaustive, as loosely-coupled PPS having some processors with their own memory also exist. A general DCS [12] consists of well separated autonomous processors connected via communication links as shown by Fig. 4.

```
Algorithm: Time-Shared Grid Resource Event Handler()
1. Wait for an event
2. If the external and Gridlet arrival event, then:
BEGIN /*a new job arrived*/
      a. Allocate PE Share for Gridlets Processed so far
      b. Add arrived Gridlet to Execution Set
      c. Forecast completion time of all Gridlets in Execution Set
      d. Schedule an event to be delivered at the smallest completion time
END
3. If event is internal and its tag value is the same as the recently
      scheduled internal event tag,
BEGIN /*a job finish event*/
      a. Allocate PE Share for Gridlets Processed so far
      b. Update finished Gridlet's PE and Wall clock time parameters and
         send it back to the broker
      c. Remove finished Gridlet from the Execution Set and add to Finished
         Set
      d. Forecast completion time of all Gridlets in Execution Set
      e. Schedule an event to be delivered at the smallest completion time
      f. END
Repeat the above steps until the end of simulation event is received

Algorithm: PE Share Allocation(Duration)
1. Identify total MI per PE for the duration and the number of PEs that
            process one extra Gridlet
         a. TotalMIperPE = MIPSRatingOfOnePE()*Duration
         b. MinNoOfGridletsPerPE = NoOfGridletsInExec / NoOfPEs
         c. NoofPEsRunningOneExtraGridlet = NoOfGridletsInExec % NoOfPEs
2. Identify maximum and minimum MI share that Gridlet get in the Duration
If(NoOfGridletsInExec <= NoOfPEs), then:
BEGIN
         a. MaxSharePerGridlet = MinSharePerGridlet = TotalMIperPE
         b. MaxShareNoOfGridlets = NoOfGridletsInExec
Else /* NoOfGridletsInExec > NoOfPEs */
         a. MaxSharePerGridlet = TotalMIperPE/MinNoOfGridletsPerPE
         b. MinSharePerGridlet = TotalMIperPE/(MinNoOfGridletsPerPE+1)
         c. MaxShareNoOfGridlets = (NoOfPEs -
            NoOfPEsRunningOneExtraGridlet)* MinNoOfGridletsPerPE
```

Fig. 3 Time-shared resource scheduling policy [1]

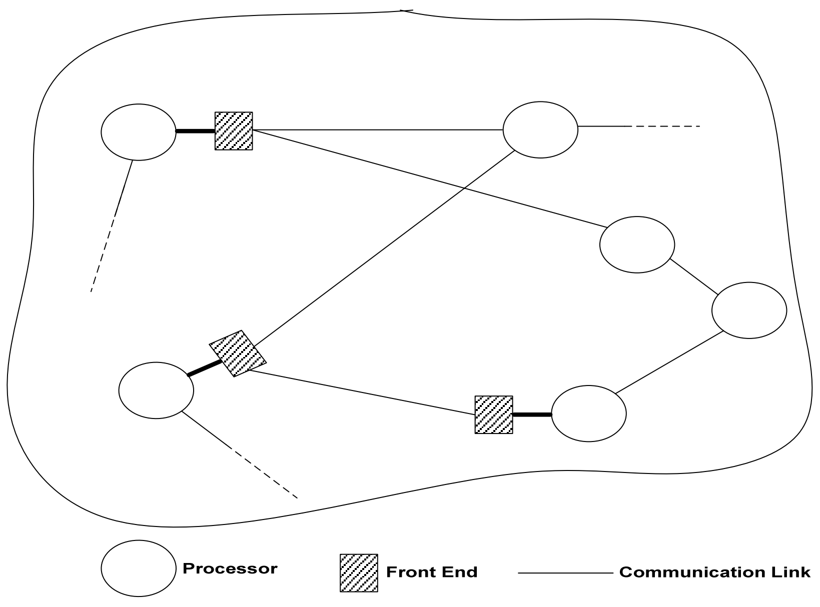


Fig. 4 Distributed computing system [12]

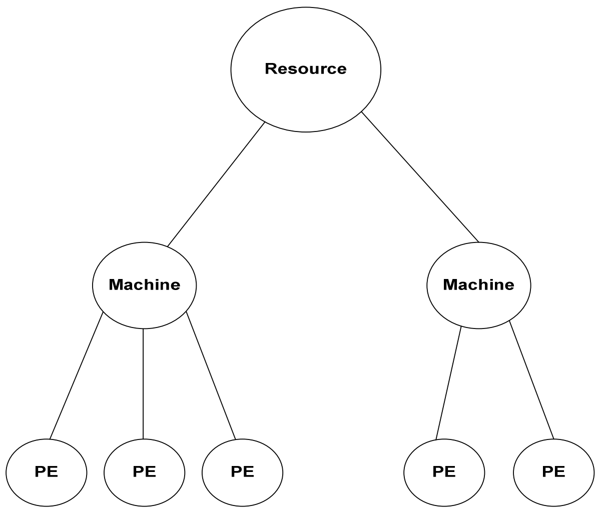


Fig. 5 Tree based resource topology

Resource topology [4] is a graph structure showing resource elements, specific configurations and relationships among these elements. Nodes in the resource topology represent elements of resource, while arcs represent relationship between the elements. We have included machine and PE as new autonomous GridSim entities, as shown in tree resource topology of Fig. 5.

Machine is new independent autonomous entity composed of one or more PEs to mimic multi-computer distributed computing systems. Machine is registered with GIS as other resource entities are registered. Machine can be a part of a resource (SMP abstraction) or it can be an independent part of the grid directly (DCS abstraction).

PE is autonomous to the extent that it can only communicate with its machine via the local bus. It is the main entity to simulate job execution. It has its own job scheduling and load balancing policies. User, broker, GIS, input and output behave exactly in the same fashion as in the original GridSim. When a job arrives at RTA-GridSim resource, it schedules them to an appropriate machine based on its job scheduling and load balancing policy.

.

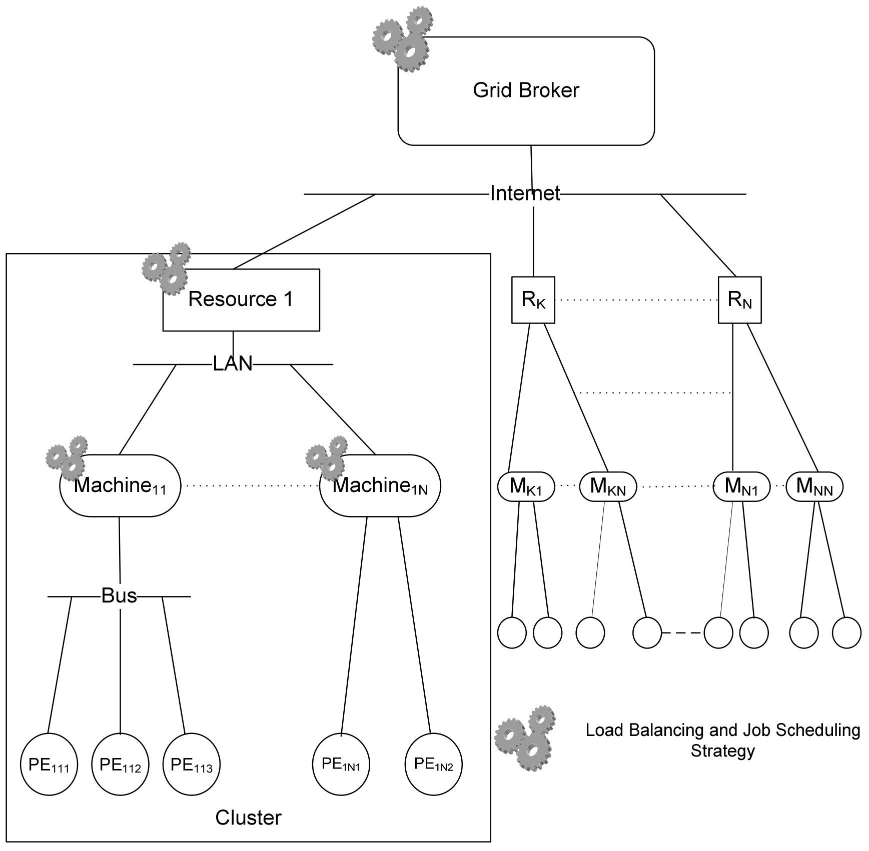


Fig. 6 RTA-GridSim hierarchical structure

RTA-GridSim machine schedules the jobs submitted to it to an appropriate PE, according to its policy. When the job finishes, PE sends it back to its machine. Machine can send the finished job back to the resource if it is a part of resource, or send it back to broker entity if it is an independent part of the Grid. If the resource entity receives a finished job, it sends it back to broker entity. Fig. 6 shows the hierarchical structure of RTA-GridSim.

3.1 RTA-GridSim Interaction Protocol

Following are the steps involved in RTA GridSim interaction protocol:

1. User submits its set of gridlets to broker.
2. Broker gets the available list of resources from GIS entity.
3. Broker then queries the resource characteristics from the resource entities.
4. Broker submits the gridlet to a suitable resource.
5. Resource selects a machine for gridlet submission.
6. Machine submits gridlet to an appropriate PE.

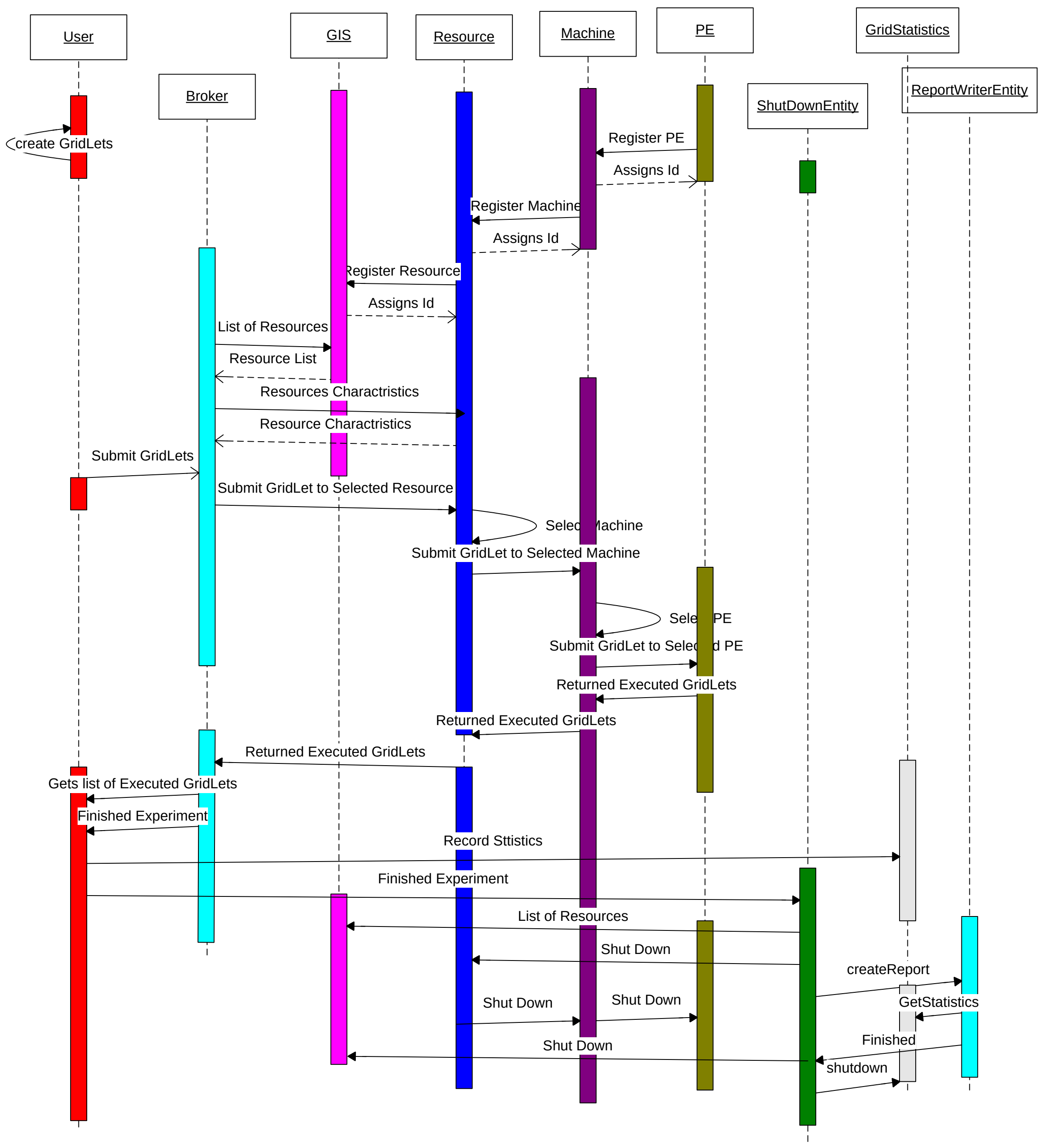


Fig. 7 RTA-GridSim interaction protocol

a

```
Algorithm: New Time-Shared Resource Event Handler()
1. Wait for an event
2. If the external and Gridlet arrival event, then:
BEGIN
a. For all Machine belonging to this Resource
   Do
i.  Calculate the load of this Machine
ii. Save the load
End For
b. Find a Machine with smallest load
c. Assign it the new coming job
d. Get the smallest time from the Machine on which it has predicted the GridLet
      to be completed
e. Schedule an event to be delivered at the got time
End If
3. If event is internal and its tag value is the same as the recently scheduled
internal event tag,
/*a job finish event*/
BEGIN
a. Recalculate the loads of all Machines
b. Save their loads
c. Update finished Gridlet's Machine and Wall clock time parameters and send it
      back to the Resource
End If
4. Repeat the above steps until the end of simulation event is received
```

b

```
Algorithm: New Time-Shared Machine Event Handler()
1. Wait for an event
2. If the external and Gridlet arrival event, then:
BEGIN
a. For all PE belonging to this Machine
   Do
i.  Calculate the load  of this PE
ii. Save the load
End For
b. Find a PE with smallest load
c. Assign it the new coming job
d. Get the smallest time from the PE on which it has predicted the Gridlet
      to be completed
e. Schedule an event to be delivered at the got time
End If
3. If event is internal and its tag value is the same as the recently scheduled
internal event tag,
/*a job finish event*/
BEGIN
a. Recalculate the loads of all PE
b. Save their loads
c. Update finished Gridlet's PE and Wall clock time parameters and send it back
      to the Resource
End If
4. Repeat the above steps until the end of simulation event is received
```

c

```
Algorithm: New Time-Shared PE Event Handler()
1. Wait for an event
2. If the external and Gridlet arrival event, then:
BEGIN
a. Calculate the execution list size of this PE
b. Add arrived Gridlet to Execution Set
c. Calculate the Mipsavailable= MIPS/Execution List size
d. Forecast completion time of all Gridlets in Execution Set
e. Schedule an event to be delivered at the smallest completion time
END If
3. If event is internal and its tag value is the same as the recently scheduled
internal event tag,
/*a job finish event*/
BEGIN
a. Allocate MIPS for Gridlets Processed so far
b. Update finished Gridlet's PE and Wall clock time parameters and send it back
      to the broker
c. Remove finished Gridlet from the Execution Set and add to Finished Set
d. Forecast completion time of all Gridlets in Execution Set
e. Schedule an event to be delivered at the smallest completion time
END If
4. Repeat the above steps until the end of simulation event is received
```

Fig. 8 Designed event handlers for (a) Resource (b) Machine and (c) PE

3.2 Designed Gridlet Scheduling Algorithm

GridSim provides two scheduling policies for the execution of the gridlet at resource namely time shared and space shared. As in RTA-GridSim architecture, the main job execution entity is PE. We have redesigned the time-shared scheduling policy and deployed at PE. Also we have re-designed the event handlers of machine and resource. Figures 8a, 8b and 8c show the algorithms for the new entities of RTA-GridSim. Here resource has an internal scheduling policy. It assigns the new coming gridlet to a machine whose load was minimum at that time. The machine, as an independent entity of GridSim assigns the gridlet having smallest load in simple smallest queue length.

4 Results and Discussion

4.1 Flexibility and Heterogeneity

The ultimate goal of a grid simulator is to provide enough abstraction flexibility so that every type of resource and network topology can be translated. The RTA-GridSim is a first step towards accommodating resource heterogeneity in grid simulation. RTA-GridSim is more flexible and heterogeneous than GridSim because now at any level, PE arrival and departure of grid elements are relatively easy.

Furthermore, novel resource management, job scheduling and load balancing policies can be implemented at every level. A machine interacts with its PE entities by a local bus, a resource communicates to its machine entities via a local LAN, while broker communicates with resource and GIS over worldwide internet link. The grid is formed on the internet by communication of broker and user entities.

Now, if for example, a load balancing algorithm is implemented at the resource and machine level and load imbalance can be handled locally at machine only (involving local bus), there can be a considerable saving of communication overhead.

4.2 Load Balancing Experiment

To appreciate the proposed enhancements to GridSim, we have developed a load-balancing algorithm based on [4, 6]. We have simulated Load Balancing algorithms presented in [4, 6] on GridSim (LBGS) and Load Balancing on Enhanced RTA-GridSim architecture (LBEGS) [4, 6].

To evaluate and compare LBGS and LBEGS, we have used the following system configurations. Windows 2000 Professional with SP2 on an Intel PIV 2.4 GHz, with 256 megabytes of RAM and 40 GB of hard disk. GridSim version 4.0 with JDK 1.5 updates 6 and JRE 1.3.

4.3 Communication Overhead

In table 1, the readings are taken by varying the number of PE, starting at 10 and ending at 50 with a step of 10 and the numbers of gridlets (grid jobs) are kept constant at 100. The communication overheads are calculated by counting the number of messages over Internet, LAN and Machine. The readings are the average of 5 runs.

Table 1 Communication overhead vs. No of PE

No of PE	Communication Overheads	
	LBGS	**LBEGS**
10	6.46 sec	3.51 sec
20	4.00 sec	2.20 sec
30	3.32 sec	1.94 sec
40	1.98 sec	1.17 sec
50	1.62 sec	0.84 sec

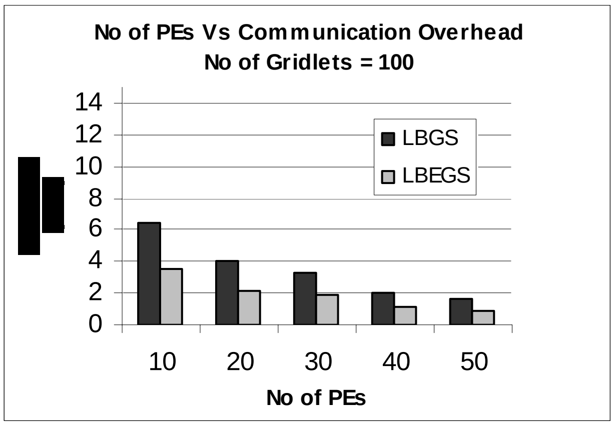


Fig. 9 Communication overhead vs. Communication Overhead

Figure 9 shows that communications overhead of LBEGS is lower when the gridlets are kept constant. This is because most of load balancing decisions are made locally at the resource and machine levels and broker is not contacted. This minimizes the number of control messages which are going all the way to broker and thus reduces the communication overheads.

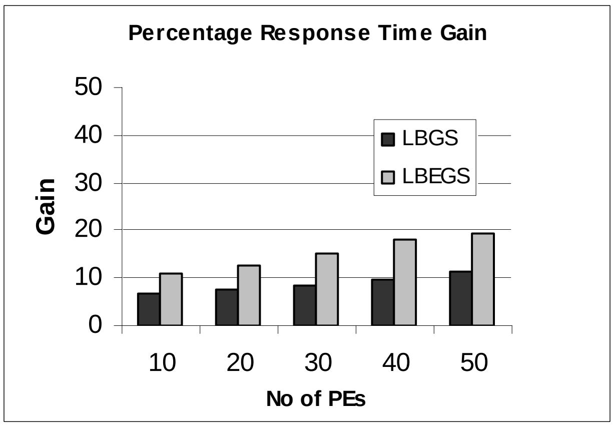
4.4 Response Time

The response time is defined as the total aggregate simulation time. The results presented here show the percentage response time gain of LBGS and LBEGS over a grid simulation Without Load Balancing (WLB). The readings are taken by varying the number of PE starting at 10 and ending at 50 with a step of 10. The numbers of gridlets are kept constant at 100.

The Fig. 10 shows that LBEGS has a considerable percentage response time gain over LBGS. It is because in LBEGS i) the gridlets are assigned to a resource considering the current load, ii) load imbalanced situations are mostly handled locally iii) the control overheads are kept minimum.

Tabel 2 Gain in Response time

No of PE	**Percentage of Gain**	
	LBGS	**LBEGS**
10	6.60 %	10.77 %
20	7.46 %	12.47 %
30	8.21 %	14.99 %
40	9.81 %	17.95 %
50	11.33 %	19.47 %



Gain LBEGS = (RT WLB / RT LBEGS)
Gain LBGS = (RT WLB / RT LBGS)
RT = Response time

Fig. 10 Percentage Response Time Gain

5 Conclusion

Resource management, job scheduling and load balancing algorithms cannot be evaluated on real grid systems. Grid simulators are effectively used to mimic the real grid systems because they provide abstractions for a controllable configuration of grid elements for extendable period of time. GridSim is a popular grid simulator, used to evaluate grid economy based job scheduling algorithms. GridSim does not provide abstractions for accommodating resource and network topology. In this study, we have proposed an enhancement RTA-GridSim, which solves this problem. We have proved that the proposed architecture can emulate resource heterogeneity and is more flexible than GridSim.

We have evaluated a tree-based resource topology, and conducted a load balancing experiment on RTA-GridSim. Our measured results show a considerable reduction in communication overhead and roughly 80% improvement in response time gain of RTA-GridSim over GridSim. In future, we will test validity of our propose enhancement and algorithm on other topologies and scheduling policies like time shared and space shared.

References

[1] R. Buyya, and M. Murshed. “GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing”, *Concurrency and Computation: Practice and Experience,* Vol. 14, pp. 1175-1220, 2002

[2] I. Foster, J. Kesselman, M. Nick, and S. Tuecke. “Grid Services for Distributed System Integration” *IEEE Computer*, Vol. 35, No. 6, pp. 37-46, 2002.

[3] R. Krishnan and K. Hosanagar. “Challenges in Designing Grid Marketplaces”, *Proceedings of the 3rd International Workshop on Grid Economics and Business Models (GECON 2006),* pp. 61-69, 2006.

[4] B. Yagoubi,Y. Slimani. “Task Load Balancing Strategy for Grid Computing” *Journal of Computer Science,* Vo.3, No.3, pp. 186-194, 2007.

[5] B. Quetier, and F. Cappello. “A survey of Grid research tools: simulators, emulators and real life platforms”, *Proceeding of 17th IMACS World Congress (IMACS 2005),* 2005.

[6] B.Yagoubi, and Y. Slimani. “Dynamic Load Balancing Strategy for Grid Computing” *Transactions on Engineering, Computing and Technology,* Vol. 13, pp. 260-265, 2006.

[7] A. Takefusa, S. Matsuoka. “Overview of a Performance Evaluation System for Global Computing Scheduling Algorithms”, *High Performance Distributed Computing,* pp. 97-104, 1999.

[8] H. Casanova. "Simgrid: A Toolkit for the Simulation of Application Scheduling", *Proceeding of 1st International Symposium on Cluster Computing and the Grid (CCGRID2001),* pp. 430-437, 2001.

[9] Dumitrescu I. Foster. "GangSim: A Simulator for Grid Scheduling Studies", *Proceeding of 2005 IEEE International Symposium on Cluster Computing and the Grid,* Vol.2, pp. 1151- 1158, 2005

[10] R. Buyya, *Economic-based Distributed Resource Management and Scheduling for Grid Computing*, Ph.D. Thesis, Monash University, Melbourne, Australia, 2002.

[11] F. Howell, and R. McNab. "Simjava: a discrete event simulation package for Java with applications in computer systems modeling", *Proceeding of First International Conference on Web-based Modeling and Simulation,* 1998.

[12] V. Bharadwaj, Thomas G. Robertazzi, and D. Ghose, *Scheduling Divisible Loads in Parallel and Distributed Systems,* IEEE Computer Society Pres, Los Alamitos, CA, USA, 1998.

[13] S. Graupner, and A. Sahai. “Policy-based resource topology design for enterprise grids”, *Proceedings of the Fifth IEEE international Symposium on Cluster Computing and the Grid (Ccgrid'05),* Vol. 1, pp. 390-397, 2005.

[14] A. Caminero, A. Sulistio, B. Caminero, C. Carrion and Rajkumar Buyya “Extending GridSim with an architecture for Failure Detection”, *International Conference on Parallel and Distributed Systems,* Vol. 1, pp. 1-8, 2007.