A New Approach for Grid Load Balancing among Heterogeneous Resources with Bandwidth Consideration

Deepak Kumar Patel *
Dept. of Computer Science & Engineering
Veer Surendra Sai University of Technology
Burla, Odisha, India
patel.deepak42@gmail.com

Debashreet Das
Dept. of Computer Science & Engineering
Veer Surendra Sai University of Technology
Burla, Odisha, India
debashreetdas12@gmail.com

Prof. C. R. Tripathy
Dept. of Computer Science & Engineering
Veer Surendra Sai University of Technology
Burla, Odisha, India
crt.vssut@yahoo.com

***Abstract*— Grid computing is a service for sharing computer power and data storage capacity over the Internet. In this paper, we propose a load balancing scheme (LBFBS) which presents a new selection method (FNCS) for the scheduling among heterogeneous resources by considering the network communication speed or bandwidth speed and capacity of various resources. So we implement the proposed load balancing strategy (LBFBS) among heterogeneous resources on the GridSim platform. Experimental results prove that the proposed load balancing mechanism can reduce the response time.**

Keywords—Load Balancing, Bandwidth, GridSim, Heterogeneous Resources

I. INTRODUCTION

A Grid belongs to the class of distributed systems. However, we cannot apply the traditional policies of the distributed system into a Grid directly [1,2]. In recent years, various load balancing methods have been proposed present in parallel and distributed systems but those cannot work in Grid architectures as such. Due to the distribution of a large number of resources in a Grid environment and the size of the data to be moved, the traditional distributed approaches are not appropriate for a Grid. The heterogeneity, autonomy, scalability and adaptability of a grid make the load balancing more difficult. These challenges bring significant obstacles to the problem of designing an efficient and effective load balancing system for Grid environments which can integrate all these factors.

There are a wide variety of issues that need to be addressed in a heterogeneous Grid environment. For example, the capacities (in terms of processor speed) of the machines differ because of processor heterogeneity. In Grid computing, as resources are distributed in multiple domains in the Internet, not only the computational nodes but also the underlying network connecting them are heterogeneous. For instance, a typical bandwidth in a local area network (LAN) varies from 10Mbps to 1Gbps, whereas it is of the order of a few kbps to Mbps for a wide area network (WAN). Also, the network bandwidth across resources varies from link to link for large-scale Grid environments. Further, the network topology among resources is also not fixed due to dynamic nature of the Grid.

In the literature, some researchers have proposed several load balancing strategies in Grid environments [3–8]. Cao [3] and Cao et al. [8] use an ant-like self-organizing mechanism to achieve system-wide Grid load balancing through a collection of simple local interactions between Grid nodes. In the said model, multiple resource management agents cooperate to achieve automatic load balancing of distributed job queues.

Yagoubi and Slimani [4] propose a layered algorithm which achieves dynamic load balancing in Grid computing. Based on a tree model, their algorithm possesses the following main features: (i) it is layered; (ii) it supports heterogeneity and scalability; and (iii) it is totally independent of any physical architecture of a Grid.

Lenders [5] uses a field-based routing and service discovery scheme for sensor networks to balance the load. Using a novel anycast routing method called density-based anycast, they show that number of group members can be effective in the routing decision, in contrast to the proximity-based routing techniques, which only consider distance information.

Erdil and Lewis [6] describe information dissemination protocols that can distribute the load in a way without using load rebalancing through job migration, which is more difficult and costly in large-scale heterogeneous grids. Essentially, in their model, nodes adjust their advertising rates and aggressiveness to influence where jobs get scheduled.

Ludwig and Maollem [7] propose two new distributed swarm intelligence inspired load balancing algorithms. One of those algorithms is based on Ant colony optimization (ASO), and the other algorithm is based on Particle swarm optimization (PSO).

In [12], Y. Hao, G. Liu and Na Wen have proposed a load balancing scheme on enhanced GridSim based on deadline control called EGDC. However, the said work does not consider the network communication speed or bandwidth speed of resources and also does not consider the resources having

ISBN No.978-1-4799-3834-6/14/$31.00©2014 IEEE

different capacity. Therefore, it does not present a realistic view of the Grid environment. This is viewed as serious limitation of the EGDC.

The present method proposed by us improves the existing EGDC technique [12] by introducing the concept of network communication speed or bandwidth speed of resources. Further, the proposed technique which we call LBFBS is also applicable for heterogeneous resources. Here, we propose a new selection method (FNCS) for the scheduling by considering the network communication or bandwidth speed and capacity of various resources. So, we simulate the proposed load balancing strategy (LBFBS) among heterogeneous resources on the GridSim platform [9]. The proposed method reduces the response time as compared to the EGDC [12].

The rest of the paper is organized as follows. In the next section, we discuss the GridSim and the grid topology. Next we describe the load balancing scheme on enhanced GridSim with deadline control (EGDC) [12]. In Section 3, we propose a new Grid load balancing scheme LBFBS which considers the network communication or bandwidth speed and capacity of various resources for the scheduling. The section 4 presents the simulated results and then compares it with EGDC [12]. Finally, Section 5 concludes the paper.

II. LOAD BALANCING SCHEME FOR RESOURCES

A. GridSim

The GridSim [9] is the most popular Grid simulation tool in comparison to other simulators. Since it simulates a Grid environment well, it has been widely used in studies of Grids [9–12]. The GridSim supports entities for simulation of a single processor, multiprocessors and the heterogeneous resources that can be configured as time-shared or space-shared systems. In addition, it allows setting of the clock to different time zones to simulate the geographic distribution of resources. Furthermore, it supports entities that simulate networks used for communication among resources. During simulation, GridSim creates a number of multi-threaded entities, each of which runs in parallel in its own thread. An entity's behavior needs to be simulated within its body() method, as dictated by SimJava [13]. The various layers of the GridSim are shown in Fig. 1 [12].

The *GridBroker* is the top manager of a Grid environment which is responsible for maintaining the overall Grid activities of scheduling and rescheduling Grid resources. Each user is connected to an instance of the Broker entity. Every job of a user is first submitted to its broker and the broker then schedules the parametric tasks according to the user's scheduling policy. The Grid Broker gets the information of the load from Grid resources and sends the Gridlet to resources for optimization scheduling.

A *Gridlet* is an entity that contains all the information related to the job, and its execution management details such as job length expressed in MIPS (million instructions per second), disk I/O operations, the size of input and output files, and the job originator. These basic parameters in GridSim determine the execution time of a job (Gridlet), the time required to transport input and output files between users and remote resources and returning the processed Gridlets back to the originator along with the results.

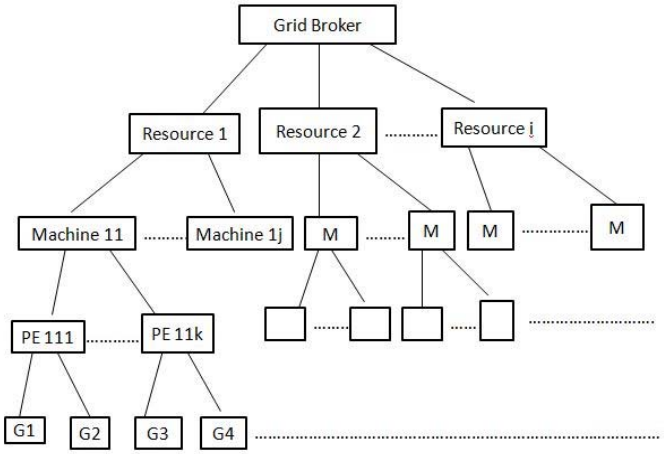

Fig. 1 The Structure of Grid in GridSim

The *GridResource* is next to the Grid Broker in the hierarchy. It is responsible for maintaining the scheduling and load balancing of its machines. In addition, it sends an event to the Grid Broker if it is overloaded. Each resource may differ from the rest of the resources with respect to their different characteristics such as no of processors, cost of processing, speed of processing, internal process scheduling policy (time shared or space shared), local load factor and time zone. The resource speed and the job execution time can be defined in terms of the ratings of standard benchmarks such as MIPS and SPEC (Standard Performance Evaluation Corporation). They can also be defined with respect to the standard machine. Upon obtaining the resource contact details from the Grid information service, Grid Brokers can query resources directly for their static and dynamic properties.

The *Grid Information Service* provides resource registration services and keeps track of a list of resources available in the Grid. The brokers can query this for resource contact, configuration, and status information.

The *Machine* is a processing entity (PE) manager. It is responsible for task scheduling and load balancing of its Processing Elements (PEs). The GridSim resource simulator uses internal events to simulate the execution and allocation of PEs to Gridlet jobs. When a job arrives, spaceshared systems start its execution immediately if there is a free PE. Otherwise, it is queued. During the Gridlet assignment, the job-processing time is determined and the event is scheduled for delivery at the end of the execution time. Whenever a Gridlet finishes, an internal event is delivered to signify the completion of the scheduled Gridlet job. The resource simulator then frees the PE allocated to it and checks if there are any other jobs waiting in the queue. If there are jobs waiting in the queue, then the

resource simulator selects a suitable job depending on the selection policy and assigns it to the PE which is free. If a newly arrived event happens to be an internal event whose tag number is the same as the most recently scheduled event, then it is recognized as a Gridlet completion event. If there are Gridlets in the submission queue, then depending on the allocation policy, GridSim selects a suitable Gridlet from the queue and assigns it to the PE or a suitable PE if more than one PE is free. After that, the completed Gridlet is sent back to the broker or user and removed from the execution set [12].

B. Grid Topology

From the topological point of view, we regard a Grid as a collection of clusters. Each cluster contains a number of PE's. The PE's within a cluster are interconnected together by LAN. Every cluster is connected to the global network or WAN by a switch. The Fig.2 describes a general grid topology [14].

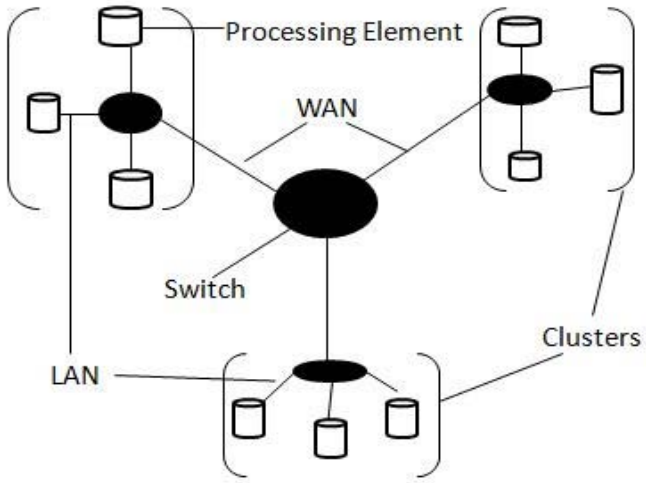

Fig. 2 General Grid Topology.

C. Load Balancing Scheme on Enhanced GridSim Based on Deadline Control

Although there exist many centralized load-balancing techniques in Grid, those as such cannot be used in GridSim due to their different frameworks.

However, the EGDC [12] is a suitable method which can be implemented in GridSim, which pays attention towards the deadline of tasks and presents a load balancing mechanism based on deadline control. First, the said model uses a new calculation method for the load, and then it classifies the resources into three types: over loaded, normally loaded and under loaded. Then it makes a request to the Grid Broker according to the change of load state. Then, the Grid Broker periodically schedules Gridlets. However this load balancing technique targets only resources having same capacity. Their scheme though reduces the makespan and the resubmitted times. Though it enhances the number of finished Gridlets, it does not consider the network communication speed or bandwidth speed of resources and also does not consider the resources having different capacity.

III. POPOSED LOAD BALANCING SCHEME: ENHANCED GRIDSIM WITH LOAD BALANCING USING FASTEST BANDWIDTH SPEED (LBFBS)

In this section, we propose a new load balancing scheme (LBFBS) which presents a new selection method (FNCS) for the scheduling so as to improve the limitations of the existing method EGDC [12]. Our method considers the network communication speed or bandwidth speed of resources and the heterogeneous resources in which all resources have different capacity. The said features of the proposed method make it attractive in Grid applications.

A: Proposed Algorithm

The following notations are used in our algorithm.

Notations:

rload= current load of a resource

m= number of Machines of a resource

numPE= number of Processing Elements of a machine

ratingPE= rating of a Processing Element

rcapacity= the capacity of a resource

glength= length of the Gridlet

gdeadline= deadline given to the Gridlet

rt= resource load level

rb= machine load level

Underloadlist= a list of under loaded resources

Overloadlist= a list of over loaded resources

Normallyloadlist= a list of normally loaded resources

unassignedgridletlist= a list of Gridlets waiting for resources to be executed

Algorithm:

Begin

Step 1: /* Calculate the current load of the resource */

$$rcapacity = \sum_{t=1}^{m} numPE_t * ratingPE_t;$$

```
Newload (Resource r, Gridlet g)
{
rload = rload + (glength / gdeadline) / rcapacity;
}
```

Step 2: /*Check the state of every resource*/

```
Checkstate (Resource r)
{
If (rload < rb)
```

Resource r is ‘‘under loaded’’ and insert it into Underloadlist;

```
 Else
If (rload > rt)

Resource r is ‘‘over loaded’’ and insert it into Overloadlist;

Else

Resource r is ‘‘normally loaded’’ and insert it into
Normallyloadlist;

Endif
Endif
}

Step 3:  /*Transfer Gridlets from overloaded resources to
unassignedgridletlist*/
While (Overloadlist is not empty and Underloadlist is not
empty)
{
get a gridletlist which we want to transform from Overloadlist;

For every Gridlet g belongs to gridletlist
insert g in to unassignedgridletlist;

Endfor
Endwhile
}

Step 4:  /*Assigning gridlets from unassignedgridletlist to
under loaded resources*/
Value1 = -∞;  Value2 = -∞;
While (Underloadlist is not empty and unassignedgridletlist is
not empty)
{
Arrange the under loaded resources present in the
Underloadlist according to the fastest network communication
speed or fastest bandwidth speed     //1st arrangement (FNCS)

Again rearrange the under loaded resources having same
bandwidth or network communication speed present in the
Underloadlist according to the highest capacity of the
resource                                  //2nd arrangement (FNCS)

For every resource r in Underloadlist
For every Gridlet g in unassignedgridletlist

add = (glength / gdeadline) / rcapacity;
total = rload + add;
Capacity1 = rt - total;
Capacity2 = total - rt ;

If (rb < total < rt  & Capacity1 > Value1)
Value1= Capacity1;
Assigning gridlet g to resource r;
Endif

If (total < rb & Capacity2 >Value2)
Value2 = Capacity2;
Assigning gridlet g to resource r;
Endif

Endfor
Endfor
End while
}
End
```

B: Description of the Algorithm

In *Step1*: we calculate the current load of resources. If the load of one resource r is rload and a Gridlet g is assigned to resource r with a deadline in seconds, then we have to calculate the new load of that resource r.

In *Step2*: we check the state of the resource periodically. The states are classified into three categories: under loaded, normally loaded, and over loaded. Assuming that 1 >= rt > rb >= 0, if the load of the resource is more than rt, we call the state of the resource ‘‘over loaded’’. If the load of resource is less than rb, we call the state of the resource ‘‘under loaded’’. If the load of resource is in between rb to rt , we call the state of resource ‘‘normally loaded’’. Here rt is known as “resource level load” and rb is known as “machine level load”. After checking the state of every resource, the Grid Broker inserts it into the list that it should belong to. The Grid Broker inserts the resource into Overloadlist / Normallyloadlist / Underloadlist whose state is over loaded / normally loaded/ under loaded, respectively.

In *Step3*: we transform Gridlets from the over loaded resources to the unassignedgridletlist for subsequent scheduling. If the state of a resource is under loaded, we can assign more Gridlets from unassignedgridletlist to the under loaded resource.

In *Step4*: we assign Gridlets from unassignedgridletlist to under loaded resources. Unassignedgridletlist is used to store the new arriving Gridlet and the unfinished Gridlets coming from the overloaded resource. This step gives details for scheduling Gridlets. In EGDC [12], the Grid Broker periodically schedules Gridlets. But here, the Grid Broker schedules Gridlets under a new selection method, called FNCS (fastest network communication speed). In this selection method, there are 2 arrangements. First, we arrange the underloaded resources present in the Underloadlist according to the fastest network communication speed or fastest bandwidth speed (1st arrangement). Second, we rearrange the under loaded resources having the same bandwidth or network communication speed according to the highest capacity of the resource (2nd arrangement). This 2nd arrangement is possible only in heterogeneous case it means that all resources have different capacity. So the under loaded resource having the highest network communication speed (highest bandwidth speed) and highest capacity, get the first chance to execute gridlets from the unassignedgridletlist. This significantly decreases the response time.

IV. SIMULATION

The proposed algorithm is implemented using GridSim5.0 simulator. The details of the experimental setup and results are described below. Finally, the results of the simulation are compared with the EDGC [12]. We compare their response times. The response time is the total simulation time which is

measured from the time the first job is sent to the Grid until the last job comes out of the Grid.

There are 30 resource providers and every resource provider has two machines. Every machine has five PEs. So, the Grid system has 300 PEs in total. Due to resource heterogeneity, every PE has rating between 1 and 5 MIPS. Here, we express the Gridlet length in PEs (1 million instructions). Every Gridlet length is between 1 and 5 million instructions. Every Grid has a deadline with a range [1 6] s.

In our simulation model, we have considered the heterogeneous resources that are connected by communication channels. The network bandwidth connecting two resources varies from 0.5 Mbps to 10 Mbps. First, we connect the user entity with a router with 1 Mbps connection. Second, connect all resource entities with another router with various connections (0.5 Mbps to 10 Mbps). Then, connect both the router with 1 Mbps connection.

Table 1: Parameters values

Parameter	Value
Resource Threshold (rt)	0.8
Machine Threshold (rb)	0.75
PE Threshold	0.6
Gridlet Length	[1,5] PEs
Deadline	[1,6] s

A. Comparison of Response Time when all resources have the same capacity

Here in the proposed method (LBFBS), each PE has a same processing power like EGDC [12]. But all resources have different bandwidth speed. So here only 1st arrangement is possible. We execute the simulation a number of times to get the exact results. But in all cases, the simulation time of LBFBS is found to be less than EGDC [12]. So for the purpose of comparison, we take one of the results. As shown in Fig. 3, after 600 Gridlets supplied to the resources, we check the state of the resources and schedules new Gridlets and Gridlets coming from overloaded resources to the under loaded resources using FNCS selection method. Therefore, until 600 Gridlets, the simulation time is nearly the same but after that we can see the huge difference in time till 800 Gridlets are executed.

B. Comparison of Response Time when all resources have different capacity

Here, in both EGDC [12] and LBFBS, each PE has different processing power and all resources have different bandwidth speed. So here the both 1st and 2nd arrangements are possible. As shown in Fig. 4, due to different resource speed, there is also difference in response time of initial Gridlets. But after 600 Gridlets supplied to the resources, we check the state of the resources and schedules new Gridlets and Gridlets coming from overloaded resources to the under loaded resources using the FNCS selection method and we can see the huge difference in response time till 800 Gridlets are executed.

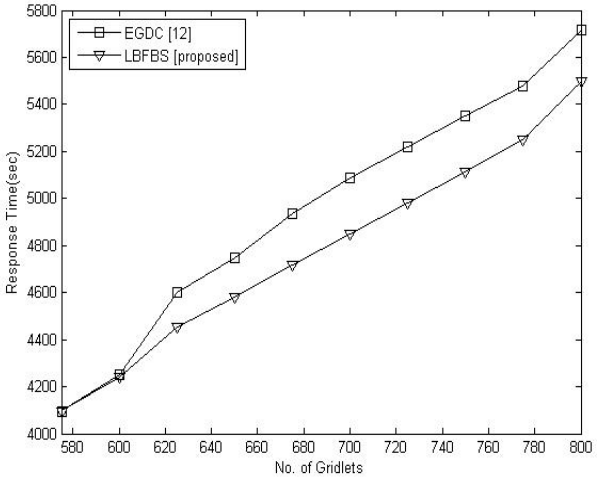

Fig. 3. Response Time versus No. of Gridlets

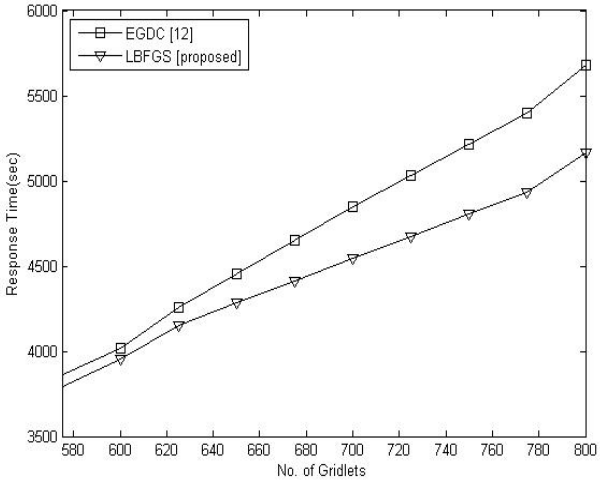

Fig. 4. Response Time versus No. of Gridlets

V. CONCLUSION

In this paper, we proposed a new selection method (FNCS) for load balancing among heterogeneous resources by considering network communication speed or bandwidth speed and capacity of resources for a Grid. Also we presented the proposed load-balancing algorithm (LBFBS) for the heterogeneous Grid computing environment. Extensive simulations are conducted using GridSim5.0. From the simulation results, we could conclude that the proposed method (LBFBS) has better performance than EGDC [12] as it reduces the response time.

The proposed work can be extended further for implementation of various other efficient scheduling, load

balancing, and fault tolerance techniques on enhanced GridSim with a tree structure. More characters, like the resource processing ability, the requirement of Gridlet can also be taken into consideration.

REFERENCES

[1] B. Yagoubi, Y. Slimani, Task Load balancing strategy in grid environment, Journal of Computer Science 3 (3) (2007) 186–194.

[2] B. Yagoubi, Y. Slimani, Load balancing strategy in grid environment, Journal of Information Technology Applications 4 (2007) 285–296..

[3] J. Cao, Self-organizing agents for grid load balancing, in: Proceedings of the fifth IEEE/ACM International Workshop on Grid Computing, GRID'04, Pittsburgh, PA, November 2004.

[4] B. Yagoubi, Y. Slimani, Dynamic load balancing strategy for grid computing, Engineering and Technology (2006) 90–95.

[5] http://www.lenders.ch/publications/

[6] D. Erdil, M. Lewis, Dynamic grid load sharing with adaptive dissemination protocols, The Journal of Supercomputing (2010) 1–28.

[7] S. Ludwig, A. Moallem, Swarm intelligence approaches for grid load balancing, Journal of Grid Computing (2011) 1–23.

[8] Junwei Cao, Daniel P. Spooner, Stephen A. Jarvis, Graham R. Nudd, Grid load balancing using intelligent agents, Future Generation Computer Systems (ISSN: 0167-739X) 21 (1) (2005) 135–149. doi:10.1016/j.future.2004.09.032.

[9] R. Buyya, M. Murshed, GridSim: a toolkit for the modeling and simulation of distributed management and scheduling for Grid computing, The Journal of Concurrency and Computation: Practice and Experience 14 (2002) 13–15.

[10] K. Qureshi, A. Rehman, P. Manuel, Enhanced GridSim architecture with load balancing, The Journal of Supercomputing (2010) 1–11.

[11] http://www.buyya.com/GridSim/

[12] Y. Hao, G. Liu, Na Wen, An enhanced load balancing mechanism based on deadline control on GridSim. Future Generation Computer Systems 28 (2012) 657-665.

[13] F. Howell, R. McNab, SimJava: a discrete event simulation package for Java with applications in computer systems modeling, in: Proceedings of the 1st International Conference on Web-based Modelling and Simulation, Society for Computer Simulation, San Diego, CA, 1998.

[14] N.Malarvizhi, Dr. V. Rhymend Uthariaraj, Hierarchical Load Balancing Scheme for Computational Intensive Jobs in Grid Computing Environment, In: Proceedings of the 1st IEEE international conference on advanced computing, 13–15 Dec, 2009