An Efficient Load Balancing Mechanism with Cost Estimation on GridSim

Deepak Kumar Patel *
Dept. of Computer Science & Engineering
Veer Surendra Sai University of Technology
Burla, Odisha, India
patel.deepak42@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*— Due to the rapid technological advancements, the Grid computing has emerged as a new field, distinguished from conventional distributed computing. The load balancing is considered to be very important in Grid systems. In this paper, we propose a new dynamic and distributed load balancing method called "Enhanced GridSim with Load Balancing based on Cost Estimation" (EGCE) for computational Grid with heterogeneous resources. The proposed algorithm EGCE is an improved version of the existing EGDC in which we perform load balancing by estimating the expected finish time of a job on resources on each job arrival and then balance the load by migrating jobs to resources by taking into account the resource heterogeneity and network heterogeneity. We simulate the proposed algorithm on the GridSim platform. From the results, our algorithm is shown to be quite efficient in minimizing the average response time.**

Keywords—Load Balancing, GridSim, Average Response Time, Gridlet

I. INTRODUCTION

The Grid serves as a comprehensive and complete system for organizations by which the maximum utilization of resources is achieved [1]. With its multitude of heterogeneous resources, a proper scheme of scheduling and efficient load balancing across the Grid is required for improving the performance of the system. The load balancing mechanism aims to equally spread the load on each computing node, maximizing their utilization and minimizing the total task execution time [2].

Although there exist many load-balancing algorithms for traditional distributed systems, those cannot be applied as such to a Grid directly. This is due to the unique characteristics of the Grid computing environment such as heterogeneity, autonomy, scalability, adaptability, resource selection and computation-data separation. Thus, it is a challenging problem to design an efficient and effective load balancing scheme for Grid environments which can integrate all the above said factors [13].

There are a wide variety of issues that need to be addressed for a heterogeneous Grid environment. For example, the capacities of the machines differ because of the resource heterogeneity. The job execution time becomes important due to resource heterogeneity which is influenced by the capacity of the resources. In Grid computing, as the resources are geographically distributed and located at different sites, not only the computational nodes but also the underlying networks connecting them are heterogeneous in nature. Due to network heterogeneity, the network bandwidth across the resources varies from link to link in Large Scale Grid environments. The job migration cost is primarily influenced by the available network bandwidth between the sender and receiver nodes and hence becomes an important factor for a Grid environment [12].

In the literature, some researchers have proposed several load balancing strategies in Grid environments [9–12].

Yagoubi and Slimani [9] proposed a dynamic tree based model which represents the Grid architecture. 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. Their simulations show a significant improvement in mean response time with a reduction of communication cost.

In [10], a load balancing algorithm LBEGS which is an improvement over the LBGS [9] is proposed. It gives the details of the load calculation methods of the PE, Machine, and GridResource. At the same time, the model also gives the algorithms of load balancing between PEs, machines, and resources. Their proposed scheme not only reduces the communication overhead of Grid resources but also cuts down the idle time of the resources during the process of load balancing based on GridSim. The effectiveness in terms of communication overhead and response time reduction is gauged.

The researchers in [11] presented a decentralized dynamic load balancing algorithm called ELISA (Estimated Load Information Scheduling Algorithm) for general purpose distributed computing systems. The ELISA uses the estimated state information based upon the periodic exchange of exact state information between the neighbouring nodes to perform load scheduling. The primary objective of their algorithm is to cut down the communication and load transfer overheads by minimizing the frequency of status exchange and by restricting the load transfer and status exchange within the buddy set of a processor.

In [12], Shah et al proposed two algorithms, the MELISA (Modified ELISA) and LBA (Load Balancing on Arrival). The MELISA is applicable to large-scale systems with the resource heterogeneity and network heterogeneity. The other algorithm, LBA is applicable to small-scale systems. It performs load

* *Corresponding Author*

978-1-5090-3584-7/16 $31.00 © 2016 IEEE
DOI 10.1109/ICIT.2016.15

balancing by estimating the expected finish time of a job on buddy processors on each job arrival. Both the said algorithms estimate system parameters such as the job arrival rate, CPU processing rate and load on the processor.

In this paper, our basic aim is to develop a suitable load balancing algorithm adapted to the heterogeneous Grid computing environment. This paper presents a dynamic and distributed load balancing algorithm called EGCE (Enhanced GridSim with Load balancing based on Cost Estimation) which estimates the expected finish time of a job on resources on each job arrival and then balances the load by scheduling jobs by taking in to account the heterogeneity in resources and networks. The proposed load balancing strategy is simulated on the GridSim platform [14]. Our method reduces the average response time.

The rest of the paper is organized as follows. In the next section, we discuss GridSim and describe the load balancing scheme on enhanced GridSim with deadline control (EGDC) [13]. In Section 3, we propose a new Grid load balancing scheme EGCE. The section 4 presents the simulated results and then compares it with EGDC [13]. Finally, Section 5 concludes the paper.

II. BackGround

A. GridSim

The Grid simulator (GridSim) is a very popular Grid simulation tool [14]. The GridSim toolkit supports modeling and simulation of a wide range of resources, such as single or multiprocessors, shared or distributed memory machines such as PCs, workstations, SMPs, and clusters with different capabilities and configurations. The GridSim is built on a general-purpose discrete event simulation package called SimJava [6], which is implemented in Java. The GridSim's highly specialized grid simulation base classes are developed extending SimJava's simulation foundation classes [3,4,5]. The various layers of the GridSim are shown in Fig. 1 [13]. A brief discussion on each layer of the GridSim follows.

The user dispatches its jobs to the Grid through its own broker called the *Grid Broker*. Each user is connected to an instance of the broker entity. Every job of the user is first submitted to its broker and then the broker schedules the parametric tasks according to the user's scheduling policy. Before scheduling the tasks, the broker dynamically gets a list of available resources from the global directory entity. Every broker tries to optimize the policy of its user and therefore, the brokers are expected to face extreme competition while gaining access to resources. The scheduling algorithms used by the brokers must be highly adaptable to the market's demand-supply situation [3,4].

The *GridResource* is next to the Grid Broker in the hierarchy. Each GridResource may differ from the rest of resources in terms of the number of processors, cost of processing, speed of processing, internal process scheduling policy, local load factor and the time zone [3,4].

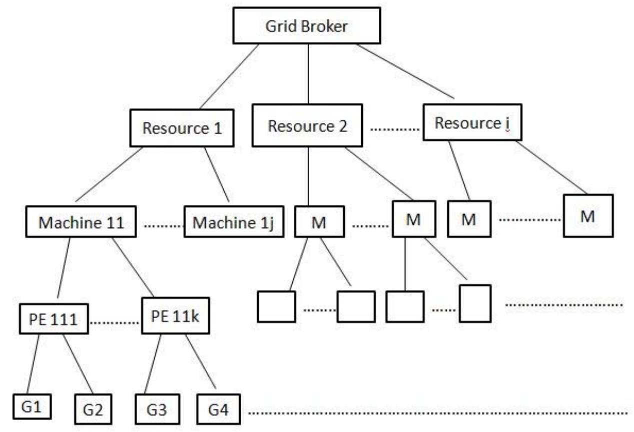

Fig. 1 The Structure of Grid in GridSim

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 [13].

The *Machine* is a processing entity 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, the spaceshared systems start its execution immediately if there is a free PE. Otherwise, it is queued. Whenever a Gridlet job 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 job 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 free PE [13].

A *Gridlet* is an entity that contains all the information related to a job and its execution management details such as the job length expressed in MIPS (million instructions per second), the disk I/O operations, the size of input and output files and the job originator. These basic parameters in the GridSim determine the execution time of a job (Gridlet), the time required to transport input and output files between the users and the remote resources, and returning the processed Gridlets back to the originator along with the results [3,4].

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

There exist many centralized load-balancing schemes for the Grid. However, those as such cannot be used in GridSim due to the differences in their frameworks [13].

However, the EGDC [13] is a distributed load balancing scheme on GridSim which provides the deadline control for tasks. At the outset, the resources check their states and make a request to the Grid Broker according to the change of state in load. Then, the Grid Broker assigns the Gridlets between the resources and scheduling for load balancing under the deadline request. The EGDC is simulated on the GridSim. The EGDC is shown to reduce the response time, improve the finished rate of the Gridlet and reduce the resubmitted time. But it ignores the job migration cost, job execution time, while making a decision on load balancing. This job migration cost which is primarily influenced by the available network bandwidth between the sender and receiver nodes, becomes an important factor where the communication latency is very large. The communication latency becomes large due to network heterogeneity or the higher job size such as for a Grid environment. Also, the job execution time becomes important due to resource heterogeneity which is influenced by the capacity of resources. Therefore, it does not present a realistic view of the Grid environment. This is viewed as a serious limitation of the EGDC and therefore needs research attention for its further improvement.

III. The Poposed Work: Enhanced GridSim with Load Balancing based on Cost Estimation (EGCE)

In this section, the EGDC is improved and a new method of load balancing called Enhanced GridSim with Load Balancing based on Cost Estimation (EGCE) is proposed. We use the following notations and terminologies throughout the paper. In the proposed model, the Grid system consists of h heterogeneous resources which are connected by communication channels.

A: Notations and Terminologies

h = number of heterogeneous resources

n = number of Gridlets to be processed

L_r = current load of a resource

C_r = capacity of a resource

B_r = network Bandwidth of a resource

PE = no. of PEs in a machine

CPE = capacity of a PE

m = no. of machines in a resource

L_g = length of the Gridlet

FS_g = Gridlet filesize

D_g = deadline given to the Gridlet

rt = resource load level

rb = machine load level

UnderloadedResourcelist = a list of underloaded resources

OverloadedResourcelist = a list of overloaded resources

NormalResourcelist = a list of normally loaded resources

CT = current system time

ET_g = execution time for a Gridlet

CT_g = communication time for a Gridlet

EFT_g = expected finish time

B: Proposed Algorithm

```
Begin

Step 1:  /* Calculate the current load of the resource */
$C_r = \sum_{i=1}^{m} (PE_i * CPE_i)$
$L_r = L_r + (L_g) / (D_g * C_r)$;

Step 2:  /*Check the state of every resource*/
for all resources do
calculate Lr which is the current load of a resource;
if (Lr < rb)
resource is ''under loaded'' and add this resource to
UnderloadedResourcelist;
else if (Lr > rt)
resource is ''over loaded'' and add this resource to
OverloadedResourcelist;
else
resource is ''normally loaded'' and add this resource to
NormalResourcelist;
end for

Step 3:  /*Transfer Gridlets from overloaded resources to
UnfinishedGridletlist */
While (OverloadedResourcelist is not empty and
UnderloadedResourcelist is not empty)
{
get a gridlet list which we want to transform from
OverloadedResourcelist;

For every Gridlet g belongs to gridlet list
insert g in to UnfinishedGridletlist;

Endfor
Endwhile
}

Step 4:  /* Sorting the underloaded resources present in the
UnderloadedResourcelist */
Sort the underloaded resources by calculating the expected
finish time of the Gridlet g on all the underloaded resources
$EFT_g = CT + \{(ET_g) + (CT_g)\}$
$EFT_g = CT + \{(L_g / CPE) + (FS_g/B_r)\}$

Step 5:  /*Assigning gridlets from UnfinishedGridletlist to the
under loaded resources present in the UnderloadedResourcelist
*/
while (UnderloadedResourcelist and UnfinishedGridletlist is
not empty)
for every Gridlet g in UnfinishedGridletlist
for every resource R in UnderloadedResourcelist
$L_r = L_r + (L_g) / (D_g * C_r)$;
if ($L_r <= rt$)
Assigning Gridlet G to resource R;
else
Move to next underloaded resource present in the
UnderloadedResourcelist for execution of that Gridlet;
```

```
            end if
        end for
    end for
end while
End
```

C: Description of the Proposed Algorithm

In this Subsection, a brief description of the proposed algorithm (EGCE) is presented.

Step 1: First, we calculate the current load of all resources. If the current load of a resource r is L_r and a Gridlet g is assigned to that resource with a deadline, then we calculate the new load of that resource.

Step 2: Then we check the recent state of the resource due to this load. The states are classified into three categories: *underloaded, normally loaded,* and *overloaded.* If the load of the resource is more than rt, we call the state of the resource ‘‘overloaded’’. If the load of resource is less than rb, we call the state of the resource ‘‘underloaded’’. If the load of a resource is in between rt to rb, we call the state of resource ‘‘normally loaded’’. Here rt, rb are known as the *resource level load* and the *machine level load* respectively. After checking the state of the resource, the Grid Broker inserts it into the list that it should belong to. The Grid Broker inserts the resource into the OverloadedResourcelist or the NormalResoucelist or the UnderloadedResourcelist depending upon whether the state is overloaded, normally loaded or underloaded respectively.

Step 3: In the third step, if the state of any resource is over loaded, then the Grid Broker make a list of Gridlets which need to be transfer from that overloaded resource to one of the underloaded resources present in the UnderloadedResourcelist and then insert that list of Gridlets in to the UnfinishedGridletlist for subsequent scheduling.

Step 4: From the beginning of scheduling, our priority always should be assigning the Gridlets present in the UnfinishedGridletlist to one of the best underloaded resources present in the UnderloadedResourcelist.

In a computational Grid, as resources are geographically distributed and located at different sites, the job transfer cost from one site to another site is considered as an important factor for load balancing. Further, the underlying networks are heterogeneous in nature through which the Grid resources are normally connected. Moreover, due to network heterogeneity, the network bandwidth varies from one link to another. Also, the job execution time becomes important due to resource heterogeneity. Our algorithm considers these facts.

Therefore, the GridBroker calculates the expected finish time of the largest Gridlet present in the UnfinishedGridletlist on all the underloaded resources present in the UnderloadedResourcelist by estimating the job transfer time and execution time of that job on the underloaded resources. Then, the GridBroker sorts the underloaded resources present in the UnderloadedResourcelist by giving priority to the underloaded resource having the least execution time than the other resources.

Step 5: Now, the scheduling of the Gridlets from the UnfinishedGridletlist to the underloaded resources present in the UnderloadedResourcelist is done using a scheduling mechanism. This scheduling mechanism guarantees that after assigning a gridlet to an underloaded resource, the state of that underloaded resource will not be overloaded. If the state of that underloaded resource is observed to be *overloaded*, then the GridBroker will move to the next *underloaded resource* present in the UnderloadedResourcelist for execution of that Gridlet.

IV. Simulation

The proposed algorithm was implemented using the GridSim5.0 simulator [14]. The details of the experimental setup and results are described below. Finally, the results of the simulation are compared with EGDC [13]. The comparison of the proposed algorithm EGCE is based on average response time. We use Windows 7 on an Intel Core (1.73 and 1.73 GHz), with 3 GB of RAM and 1000 GB of hard disk for the simulation purpose.

Here, we consider the ART of the Gridlets processed in the system as the performance metric to measure the algorithm's efficiency. If n no. of Gridlets is processed by the system, then the average response time (ART) is given by

$$\text{Average Response Time}=\frac{1}{n}\sum_{i=1}^{n}(Finish_i - Arrival_i) \quad (6)$$

Where $Arrival_i$ is the time at which the ith Gridlet arrives, and $Finish_i$ is the time at which it leaves the system.

We conducted our simulations for three different but interesting cases: Case 1: Simulation with constant number of PEs, Case 2: Simulation with constant number of Gridlets, Case 3: Simulation with varying Job size.

A. Simulation with constant number of PEs

In this case, the simulation is conducted assuming the number of PEs to be constant and Gridlets to vary. Without loss of generality, we assume that there are 60 resources in the system and every resource has two machines. Every machine has five PEs. So, the Grid system has total 600 PEs. Due to the resource heterogeneity, every PE has the 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 within a range of 1 to 6 seconds. 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. The resource/machine/PE threshold is 0.8/0.75/0.6 [13]. The Fig. 2 shows the results of comparison of our proposed method with other for the case 1.

The Fig. 2 presents a comparative look of the average response time (ART) Vs. number of Gridlets of our proposed algorithm EGCE with the existing method EGDC [13]. The Fig. 2 shows that the average response time (ART) is lower in the case of EGCE as compared to the EGDC [13]. In Fig. 2, the observations are taken by varying the number of Gridlets starting from 0 and ending at 2500 with a step of 200, keeping the number of PEs constant at 300.

B. Simulation with constant number of Gridlets

Here, we assume that each resource has two machines and every machine has five PEs. Due to the resource heterogeneity, every PE has the rating between 1 and 5 MIPS. The Grid is created by varying the number of resources while keeping the number of Gridlets constant. We express the Gridlet length in PEs (1 million instructions). The length of each Gridlet lies between 1 and 5 million instructions. Every Grid has a deadline within a range of 1 to 6 seconds. 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. At first, we connect the user entity and a router with 1 Mbps connection. Next, we connect all the resource entities to another router with various connections with in the said range. Then, we connect both the routers with 1 Mbps connection. The resource/machine/PE threshold is 0.8/0.75/0.6 [13]. The results plotted in Fig. 3 compare the proposed method with the existing ones for the case 2.

In Fig. 3, the observations are taken by varying the number of resources from 30 to 150 with a step of 30, while the number of Gridlets is kept constant at 1000. From the Fig. 3, it is evident that the average response time (ART) is lower in the case of EGCE than the EGDC [13] with the increase of resources.

C. Simulation with varying Job Size

The proposed method EGCE takes into account the job migration cost, and the job size largely affect the job migration cost. Therefore, it is very interesting to measure the performance of the algorithms by varying the job size. In this

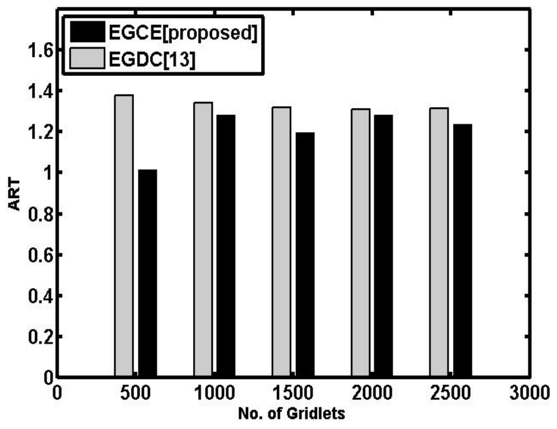

Fig. 2 ART Vs. Number. of Gridlets

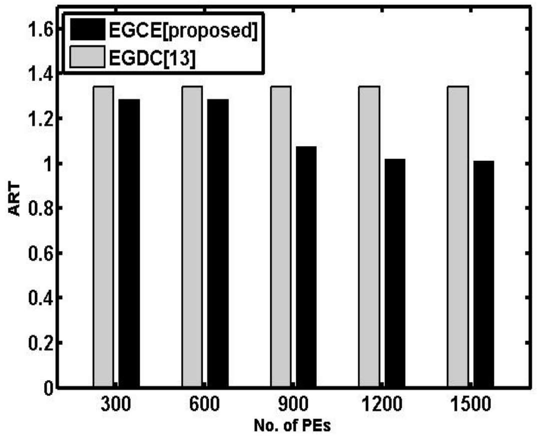

Fig. 3 ART Vs. Number. of PEs

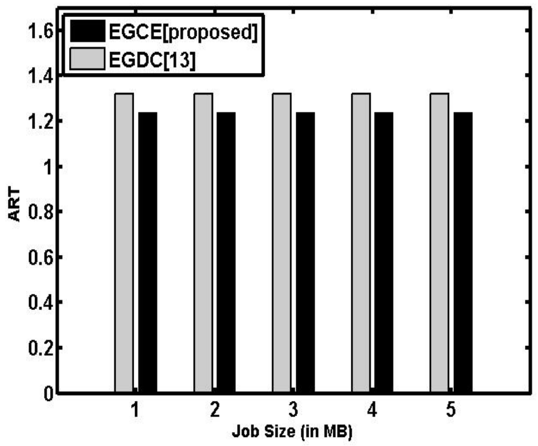

Fig. 4 ART Vs. Job Size

simulation, we vary the job size from 1 MB to 5 MB. As we increase the job size, the performance of our proposed method becomes much better than EGDC [13] in terms of the decrease in average response time (ART) (Fig. 4).

V. Conclusion

In this paper, we proposed a new enhanced load balancing method (EGCE) and scheduling technique for the heterogeneous Grid computing environment. We considered three cases for simulation, one with the constant number of PEs, second with the constant number of Gridlets and third with the varying Job Size. Extensive simulations are conducted using GridSim5.0 for all these cases. From the simulation results, the proposed method (EGCE) is observed to have better performance than the EGDC [13] in terms of average response time. The proposed work with modification can be extended further for fault tolerance and security on GridSim.

References

[1] L.M. Khanli, S. Razzaghzadeh, S.V. Zargari, A new step toward load balancing based on competency rank and transitional phases in Grid networks, Future Generation Computer Systems 28 (2012) 682-688.

[2] R. Subrata, A. Y. Zomaya, B. Landfeldt, Artificial life techniques for load balancing in computational grids, Journal of Computer and System Sciences 73 (2007) 1176–1190.

[3] M. Murshed, R. Buyya, D. Abramson, GridSim: A toolkit for the modeling and simulation of global grids. Technical Report (2001), Monash, CSSE

[4] 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.

[5] A. Sulistio, G. Poduval, R. Buyya, C.K. Tham, On incorporating differentiated levels of network service into GridSim, Future Generation Computer Systems 23 (2007) 606–615.

[6] 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.

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

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

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

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

[11] L. Anand, D. Ghose, V. Mani, ELISA: An Estimated Load Information Scheduling Algorithm for Distributed Computing Systems, Computers and Mathematics with Applications 37 (1999) 57-85.

[12] R. Shah, B.Veeravalli, M. Misra, On the Design of Adaptive and Decentralized Load-Balancing Algorithms with Load Estimation for Computational Grid Environments, IEEE Transactions on Parallel and Distributed systems, Vol. 18, no. 12, December 2007, pp 1675– 1687.

[13] 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.

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