A Sender Initiate Based Hierarchical Load Balancing Technique for Grid Using Variable Threshold Value

Neeraj Rathore
Department of Computer Science and Engineering
Jaypee University of Engineering & Technology
Guna, M.P
neeraj.rathore@juet.ac.in

Inderveer Chana
Department of Computer Science and Engineering
Thapar University
Patiala, Punjab
inderveer@thapar.edu

Abstract- **In this paper, a hierarchical load balancing technique has been presented, which is based on variable threshold value. Through this paper, an attempt has been made to solve the problem of load balancing while maintaining the resource utilization and response time with the help of sender initiative policy. The proposed technique is suitable for dynamic and decentralized Grid environments. The load is divided into different categories like lightly loaded, under-lightly loaded, overloaded, normally loaded, based thresholds values. A threshold value, which can be found out using load deviation, is responsible for transfer the task and flow of workload information. In order to increase the response time and decrease throughput of the Grid, a sender initiated policy has been introduced to reduce the communication overhead. The model has been rigorously examined over the GridSim simulator using various parameters like execution time, response time, makespan etc. Experimental results prove the superiority over existing techniques.**

Keywords: **Grid computing, GridSim, Architecture, Load balancing, Communication overhead.**

I. INTRODUCTION

The explosion of the Internet, the computational power of servers and high-speed computer network gives an idea of the Grid to share computational resources [1]. Grid is a vastly distributed enormous system that builds a global network of computational resources and may span many continents. The use of each small resource is flexible that combines computer resources and information systems for creating a ubiquitous source of computational power [2,3]. Sharing resources between organizational and institutional boundaries need an infrastructure to coordinate the resources across boundary, so-called virtual organizations. These infrastructures have been used to offer an easy management of forming virtual organizations, sharing resources, discovering and consuming services.

The Grid application performance has been degraded due to numerous challenges like fault tolerance, security, heterogeneity of resources, load balancing etc. Load balancing one of the critical features, which ensure that the load on each node should be balanced in the network. Load balancing can be generalized into four basic steps [4, 6] such as load monitoring, synchronization, rebalancing criteria and job migration. In load balancing problem, resources on one Grid site may be heavily loaded while another Grid site may be lightly loaded or idle. This may be the reason for unequal computing and patchy template of task landing. Furthermore, it is desirable to transfer some jobs from the heavily loaded computers to the lightly loaded, aiming to reduce mean job response time and increase the performance of each host and at the same time small jobs will not suffer from starvation. Load balancing has been used to report an evaluated load on each task at each computer during load balancing.

The Grid is based on a distributed network and resources are keep on adding and deleting according to their requirement. Due to the above reason, we need a hierarchical load balancing technique used for dynamic and distributed Grid environment. Several authors have proposed various load balancing techniques in Grid environment has been discussed in below literatures. Yagoubi and Slimani [7] propose a layered algorithm based on a tree model, which achieves dynamic load balancing in Grid computing. It supports heterogeneity, scalability, and total independence of any physical architecture of a Grid. Erdil and Lewis [9] has been described an information dissemination protocol that can distribute the load in a way without using load re-balancing through job migration. Furthermore, it is more difficult and costly in large-scale heterogeneous Grids. The model proposed by Erdil and Lewis [9], mainly deals with nodes adjustment, their advertising rates and aggressiveness to influence scheduling. Ludwig and Maollem [10] have proposed two new distributed swarm intelligence inspired load balancing algorithms. One algorithm is based on ant colony optimization while the other algorithm based on particle swarm optimization. Peng and Xiao [11] has described a Proximity-aware Load balancing to ensure fair load distribution with reduced overhead. It provides rapid convergence on load balance and reduces the load transfer cost in the heterogeneous and dynamic network state. Lin and Shen [12] has been reported a centralized & distributed hierarchical method to schedule a large number of parallel tasks and it can make full use of the Internet resources to solve the high-performance computing problems. Abdi and Mohamadi [13] has proposed a job scheduling policy and data replication mechanism to achieve good network bandwidth utilization and better performances to reduce data access time. This particular technique uses the Hierarchical Replication Strategy (HRS) in distributed networks. Malarvizhi Nandagopal [14] addresses the problem of load balancing using Min-Load and Min-Cost policies while scheduling jobs to the resources in the multi-cluster environment. It determines a resource for an arriving job and distribution of job to the remote clusters for optimizing performance. Heuristic estimate the completion time of executing jobs in remote clusters, improves system

---

978-1-4673-6190-3/13/$31.00 ©2013 IEEE

performance in terms of mean response time and average slowdown. This algorithm uses the Hierarchical Status Information Exchange Scheduling of parallel and distributed computing network. Zheng et al. [15] has proposed a periodic and hierarchical load balancing algorithm, which overcome the scalability challenges and considerably reduces the memory requirements, running time and provides interconnect topology aware strategies that map the communication graph on the processor topology to minimize network contention. Wang and Shen [16] works on P2P System using distributed hashing table. Even node distribution is achieved in key space according to top-down ID allocation. In heterogeneous system, the algorithm performs well in load balancing for applications with even distribution of keys and queries. Grande and Boukerche [17] uses local and cluster monitoring mechanisms to identify imbalances, minimize imbalances to perform reliable and low- latency load transfers, and improves the use of shared resources. The hierarchical architecture minimizes the overhead, formed by the balancing system, and overcome the heterogeneity issues. Boukerche and Grande [19] have reported dynamic techniques for high level architecture based on Large-Scale distributed systems. It minimizes the communication and lead to performance improvement. On the basis of existing load balancing techniques, the hierarchical load balancing technique is better than others because of its layered approach and load calculation criteria.

The load balancing technique reported by literature [20] can be categorized into lightly loaded, overloaded, normally loaded. If load on one node exceeds or lessen or equal to threshold value then it may be treated as overloaded, lightly loaded and normally loaded at each level consequently. Load parameters shown through table-1 uses previous tested simulated work [23-24], which is based on the same hierarchical technique. It uses the static threshold value at each level causes the main drawback of the network scalability. If number of nodes are increasing in the network then response time and communication overheads proportionally increases.

TABLE 1: LOAD PARAMETER [23-24]

Load Parameter	Level
$\partial = 0.6$	PE level Threshold
$\eta = 0.75$	Machine level Threshold
$\rho = 0.8$	Resource level Threshold

In order to overcome the above mentioned problems a dynamic threshold value has been used at each level in proposed technique. The value of the threshold is dynamically changing according to the Grid size in the network. Furthermore, it extended the existing technique of dividing lightly loaded node into lightly and under lightly loaded categories in the context of variable threshold and standard deviation. This approach minimizes the searching time to find out the receiver machine in order to transfer the Gridlets. Through this paper, the design and development of an optimal load balancing algorithm have been proposed for the Grid. Due to the materialization of Grid computing over the Internet, the need for a hierarchical load balancing algorithm is in demand. Communication overhead, execution time, response time, makespan, resource allocation has been recorded fair contribution of our research. In view of existing scenario, dynamic decentralized load balancing consider all the factors pertaining to the characteristics of the Grid computing environment as mentioned above. A well-designed information exchange scheduling scheme is adopted to enhance the efficiency of the load balancing model. The proposed technique is an extended version of the load balancing mechanism using sender initiate policies. In sender initiate policy, overloaded node discover under load or under lightly loaded node to transfer the Gridlets [25]. The proposed load balancing technique is not only reduces the communication overhead of Grid resources but also cuts down the response time of the whole Grid. We rigorously examined proposed algorithm on the GridSim simulator [5] in order to judge the effectiveness of the algorithm. The simulation results of proposed algorithm show significant savings over other existing approaches such as centralized load balancing, no load balancing, min-max etc.

The remainder of this paper is organized as follows. The *second section* describes the structure of the load balancing model along with the design of the system environment and its application to the network environment. The *third section* focuses on implementations of algorithms and analysis of mathematical calculations. Performance evaluation of case studies and simulation results has been presented in *section third*. Finally, *the last section* is dedicated to conclusion and future work.

II. PRELIMINARIES

Qureshi and Rehman [20], has proposed Grid architecture for load balancing depicted through the figure-1. Poisson process has been used for random job arrival with a random computation length [18]. Considering the jobs are sequenced, mutually independent with the arrival rate and can be executed on any site. Furthermore, the site should have to meet the jobs demand for computing resource and the amount of data transmitted. Each processor can only execute one job at a time and execution of a job cannot be interrupted or moved to another process during execution. This model has been divided into three levels: Level-0 Broker, Level-1 Resource, and Level-2 Machine Level. These entities can have customized characteristics. When a new Gridlet has been arrived at a machine, it may go to under lightly loaded, lightly loaded, overloaded and normal loaded by load calculation being computed at each node. In order to compute the mean job response time analysis one Grid Broker (GB) section as a simplified Grid model has been considered. Grid Broker is the top manager of a Grid environment, which is responsible for maintaining the overall Grid activities of scheduling and rescheduling. It acquires the information of the work load from Grid resources and sends the tasks to resources for optimization of load. Resource that comes next to Grid Broker in the hierarchy, are connected through internet. The resource manager is responsible to maintain the scheduling and load balancing of its machines and it also sends an event to the Grid broker during overload. The machine is a Processing Entity (PE) manager, responsible for task scheduling and load balancing of its PEs connected with various resources via LAN. Processing Entity (PE) manager also sends an event to resource during overload. PE`s next to machines are mainly responsible for calculating workload and threshold values for

load balancing, job migration and passes the load information upwards to machines via buses. Gridlet considered as a load and assigned to any of the PE`s according to their capability (i.e. Computational speed, queue length, etc.). The proposed hierarchical based Load Balancing technique in a heterogeneous Grid environment, assumed that heterogeneity in terms of processing capability and the same for each PEs.

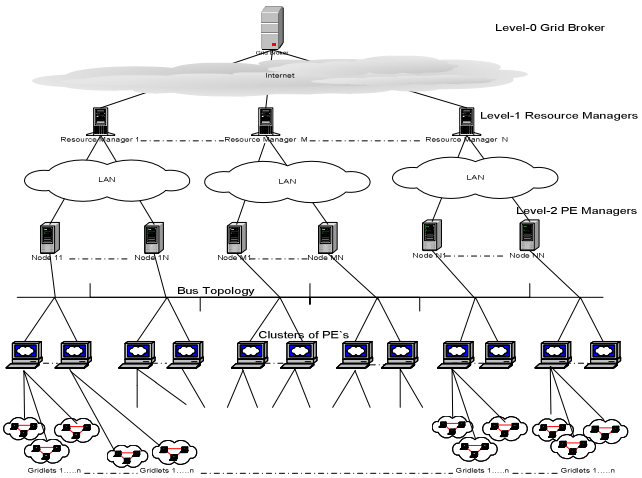

FIGURE 1: HIERARCHICAL LOAD BALANCING MODEL

III. PROPOSED TECHNIQUE

This section deals with a load balancing model that can be classified into three levels like resource, machine and PE`s shown in figure-1. The proposed algorithm has been executed individually, according to the threshold value at each level and only PE level algorithms at level-2 considered. Initially, the load has been calculated according to Gridlet (task) arrive on each node and it is connected to the corresponding PE`s. Furthermore, performed load balancing operation, according to the availability of lightly loaded or under lightly loaded node list and at the same time it passes information to associated machines. At last, the job migration operation has been performed till the overload queue is not empty and same procedure has been followed at machine level and resource level. PE level load threshold, machine level load threshold and the resource level load threshold denoted by ∂, η, and ρ respectively in algorithms 1-4, has been found out according to the task arrival.

Algorithm-1: Load balancing activity algorithm presented in algorithm-1, has been started if any of given four activity will happen. At the PE level 2, we called as lb_act_algo ( ). These activities will change the load condition in a system that as follows as [22]: (i) Arrival of any new job and queuing of that job to any node, (ii) Completion of execution of any job, (iii) Arrival of any new resource, (iv) Withdrawal of any existing resource. Whenever any of those four activities occurred then load balancing condition has been checked.

Algorithm-1: *lb_act_algo ( )*

Notice: This algorithm is executed on each node ($0 \leq i \leq n$)
Input: List of available Gridlets activity ready for execution
Output: Queued into node selected for the job execution

```
1. Begin
2.   for (check all machines belonging to corresponding resources)
3.     Waiting for some activity (load change in any machine) happen
4.     If (activity is occurring)
5.       Call machine_level_lb_algo ( ) to find out the category of PEs
6.     Else go to step 2
7.     End if
8.   End for
9. End
```

In the above algorithm, in step 2 and 3 has been checked for all machines associated with the corresponding resources and waiting for activity. If an activity occurs call Machine_Level_LB_Algo to find out the category of PEs else loop back to step 2.

Algorithm 2: At PE level, Load calculation has been done at each PE using PELoad_Calc_Algo ( ) algorithm 2 shown below:

Algorithm-2: *PELoad_Calc_Algo*

Notice: This algorithm is executed for each PE till PE! =0
Input: File Size, Rating
Output: Calculated load of each PEs

```
1  Begin
2    Initialized PELoad and check all Gridlets
3    If PELoad= 0
4      Go to step 9
5      Else
         Check Load for all PE`s using corresponding Gridlets
         //Gridlets is an arrival task as a load.
6        Calculate Current PELoad using File_size  // File_size is in bytes.
7        Calculate all PELoad using current PELoad, Average load & rating
         // Rating denotes CPU capacity is in MIPS.
8      Return PELoad
9    End if
10 End
```

Algorithm-3: *Machine_Level_LB_Algo ( )*

Notice: To divide the Gridlets loads into four PE list executed on all the m/c
Input: Threshold Value, PELoad, Algorithm 1
Output: Distributed job according to load in all PE lst

```
1.  Begin
2.    Create OPEList      //PE list with overloaded Gridlets (0 ≤ i ≤ OP) ;
                          OP-Size of the overloaded PE list
3.    Create ULPEList     //PE list with underlightlyloaded Gridlets (0 ≤ i ≤
                          UP); UP-Size of under lightly loaded PE list
4.    Create LPEList      //PE list with lightly loaded Gridlets (0 ≤ i ≤ LP);
                          LP-Size of lightly loaded PE list
5.    Create NPEList     // PE list with normal loaded Gridlets (0 ≤ i ≤ NP);
                          NP-Size of normal loaded PE list
6.    Calculate average machine load (ALM_i) and standard deviation (σ)
7.    For (all PEs associated with Machine which are processed/ empty)
8.      CALL PELoad_Calc_Algo ( ) //calculate PELoad (work load of PEs)
9.        If (ALC_i == ∂ (PELoad threshold value ∂))
                    // Average load on each cluster (collection of PE`s)
10.         Add this PE to NPEList
11.       Else if (ALC_i > ∂)
12.           Add this PE to OPEList
13.         if (ALC_i < 0.5 * ∂)
14.             Add this PE to ULPEList
15.           Else Add this PE to LPEList
16.         End if
17.       End if
18.     End if
19.   End for
20.   Sort OPEList in descending order of loads
21.   Sort ULPEList in ascending order of loads
22.   Sort LPEList in ascending order of loads
23.   CALL Machine_Level_LB_Algo2 ( ) for job migration
24. End
```

In algorithm-2, step 2 has been used to number of Gridlets in any PE queue. If all Gridlets processed or empty then set PELoad zero in step-3 followed by return PELoad in step-9, else check load for all PE`s and the calculate the current

PELoad. After that, calculate all PE`s load for further calculation using current load of PE and CPU in steps 6, 7, 8 respectively.

Algorithm 3: The Load has been categorized into four queues according to the threshold value in the algorithm-3. At the machine level, if any load changes occur, then this algorithm will execute.

Algorithm 4:

Algorithm-4 has been used to demonstrate actual Gridlet migration calculation by checking overloading PE (from which Gridlet will select for migration) and discover under load PEs during any imbalanced occurred.

Algorithm-4: *Machine_Level_LB_Algo2 ( )*

Notice: Load calculation for job migration process
Input: PE load, Task load
Output: Send load information to the resources on upper level for job migration

1 **Begin**
For (Check PE`s from the overloaded PE List (Descending Order) from selected PE)
2 **If** (check for all Gridlets processed or empty) then
3 Go to step 17
4 **Else**
5 Check for all PE`s in the lightly loaded PE list (One by one ascending order)
6 **If** (Calculate and check $((PELoad)_D + (TaskLoad)_L < 0.5 * \partial)$ then
// $(PELoad)_D$ -Load of PE for lightly loaded PE list
7 Shift Gridlet Load of Overload to lightly loaded PE List
// $(PELoad)_E$- Load of PE for under lightly loaded PElist
8 **Else** Check for all PE`s in the under lightly loaded PElist (One by one in ascending order)
If (Calculate and check $((PELoad)_E + (Task\ Load)_L < 0.5 * \partial)$) then *// $(Task\ Load)_L$ – Load of Gridlets*
9 Shift Gridlet Load of Overload PE List to under lightly loaded PE List
10 **Else** Gridlet Load will execute in its originator
11 **End if**
12 **End if**
13 **End if**
14 **End for**
15 Call MLoad_Calc_Algo ( )
16 **If** ($ALC_i >= \eta$) then //Overload machine situation
17 Send information to resource level for further calculation
18 **End if**
19 **End**

Here, describe only machine level load balancing process for identifying under load/ under lightly loaded PE from the network. The same process is going on at the upper levels for selecting unloaded/ under lightly loaded machine and resources. Through above algorithms, it has been proposed an efficient load balancing algorithm using sender initiate policy with variable threshold value.

IV. EXPERIMENTAL RESULT AND DISCUSSION

Through this part of paper firstly, we introduced the configuration of the simulation set up, then give the experimental design and results.

A. Experimental setup

Windows XP on an Intel Core (2.66 and 2.66 GHz), with 2048 MB of RAM and 360 GB of hard disk has been used during simulation experiments in GridSim version 5.0 [5] with JDK 1.5 updates 6 and JRE 1.3. The goal of this experiment is to compare the performance of a decentralized proposed load balancing algorithms (PLBA) with conventional Without Load Balancing (WLB), Load balancing in enhanced GridSim (LBGES) algorithm.

TABLE 2: SIMULATION PARAMETERS

Parameter	Value
Number of nodes in a cluster	Multiple of 25
Number of nodes (S)	100, 500, 1000, 2000
Reschedule interval (s)	600
Number of jobs	250
Requests intensity/frequency (R_i)ms	50, 100, 200, 400
Arrival time (ms)	200

Grid systems are randomized in various sizes: 100, 500, 1000, and 2000 nodes. In the experiments we change some of test parameters, such as the size of the Grid that is denoted by S on figures, resource requests frequency with 50, 100, 200 and 400 ms are denoted by R_i. The experiment is to randomly submit 250 Grid requests and schedule them to the specific Grid resource based on WLB, LBGES and PLBA. The size of data carried by the Grid task agent that is denoted by *D*, all tasks have the same computation size *D* = 25KB. There are 25 Grid resource agents in the system. The arrival time of each resource request is determined randomly by an exponential distribution with mean of 200 ms, but we will change the values of arrival time when testing effect of requests frequency on response time and resource allocation efficiency. All parameters are summarized in table 2.

Initially, all nodes having a zero load. Throughout the experiment, we use only one Grid user agent for the generation of Grid resource requests. For the simulated scenarios it does not matter how many Grid user agents there are in the system. These experimental configurations are to bring up the performance of the load balancing algorithm as many as possible.

B. Results of the simulation

The first experiment is to compare the performance of WLB, a LBGES algorithm with PLBA in terms of response time and resource allocation efficiency.

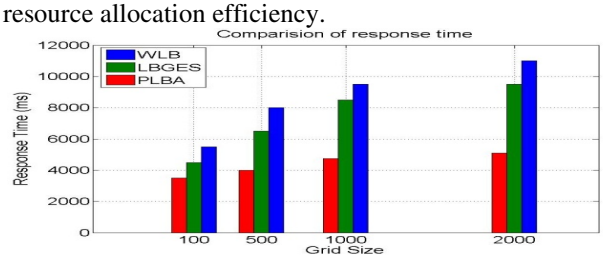

FIGURE 2: COMPARISON OF RESPONSE TIME

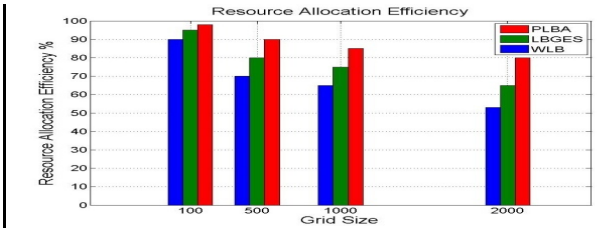

FIGURE 3: RESOURCE ALLOCATION EFFICIENCY

Figure-2 measured the response times that are influenced by Grid size, available connections and bandwidth using request intensity (R_i = 200 ms). For comparing different size Grid, a lower average response time is considered to be better depicted through Figure-2. All algorithms present the best results in the small size Grid but, when the size of the Grid is larger, PLBA is decreasing quickly. The response time using

PLBA can be 50% and 44% shorter than that using the WLB and LBGES respectively.

As shown in Figure-3, the PLBA outperforms the conventional WLB and LBEGS for different size Grid. Secondly, we measured the resource allocation efficiency of PLBA and WLB and LBEGS using R_i = 200 ms. The WLB and LBEGS achieves to match nearly 98% of all requests in the small size Grid scenario, with PLBA closely behind. However, as Grid size increases, the WLB and LBEGS degraded performance than the PLBA. Under the large size Grid, decrease of results for WLB and LBEGS is lower than in the small size. Resource allocation efficiency using PLBA is 27% and 15% larger than that using the WLB and LBEGS. Varying Grid size, result decreases for both methods similarly. The basic experiments described above have been conducted to show the fundamental differences between the PLBA and the WLB and LBEGS. Next, we devise sensitivity experiments in order to change the request frequency variables. The effects of request frequency to response time and resource allocation efficiency under different size of the Grid through below mentioned experiments. The following figure shows the behavior of two allocation methods when changing the resource requests frequency from 50 to 100 ms and 200 to 400 ms.

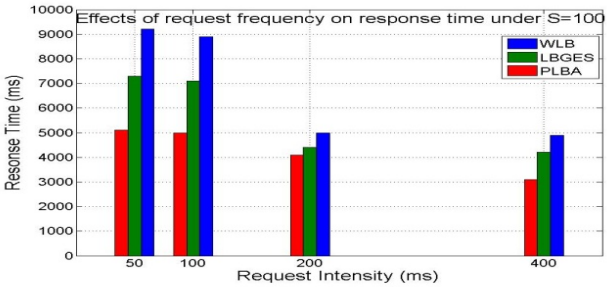

FIGURE 4: EFFECTS OF REQUEST FREQUENCY ON RESPONSE TIME UNDER S = 100

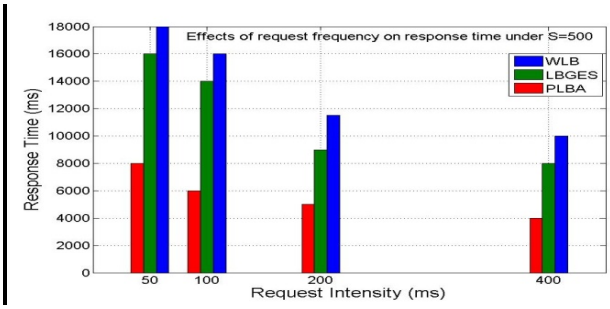

FIGURE: 5: EFFECTS OF REQUEST FREQUENCY ON RESPONSE TIME UNDER S = 500

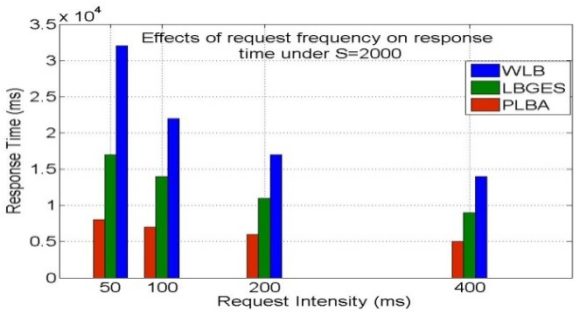

FIGURE 6: EFFECTS OF REQUEST FREQUENCY ON RESPONSE UNDER S = 2000

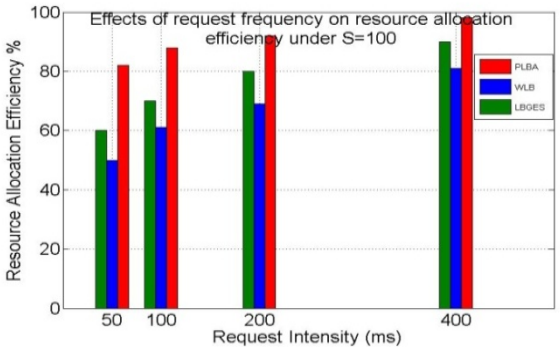

FIGURE 7: EFFECTS OF REQUEST FREQUENCY ON RESOURCE TIME ALLOCATION S = 100

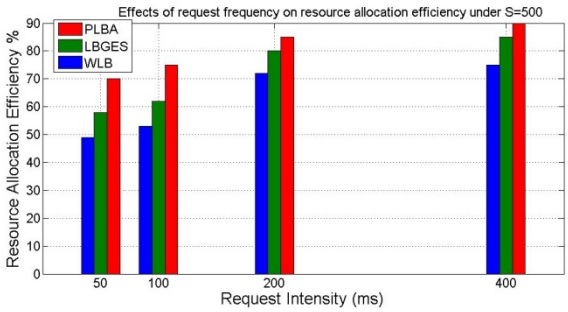

FIGURE 8: FREQUENCY EFFECT ON RESOURCE ALLOCATION IN EFFICIENCY UNDER S = 500

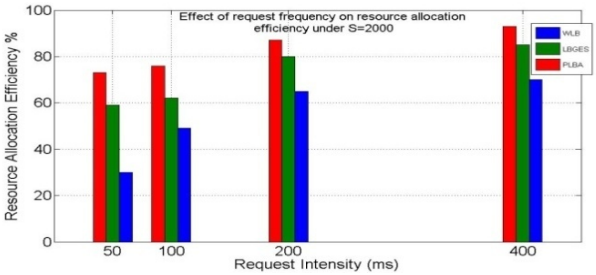

FIGURE 9: FREQUENCY EFFECT ON RESOURCE ALLOCATION EFFICIENCY UNDER S = 2000

For the response time, a lower request frequency leads to faster access times whereas, X-axis shows a change in request frequency. Presented through Figure 4-6, the response time using PLBA can be 21% and 9% shorter than that using the WLB and LBEGS respectively, under small size Grid (S = 100) and low request frequency (R_i = 200 ms). When changing the size of Grid by S = 500 and remain request frequency as R_i = 200 ms, the response time of WLB and LBEGS is 19% & 11% & longer than that using the PLBA. When increasing the size of Grid by S = 2000 and request frequency by R_i = 50 ms, the performance of WLB and LBEGS deteriorates quickly, the response time is 79% & 29% longer than that using the PLBA. Under big size Grid (S = 2000) and high request frequency (R_i = 50, 100 ms), WLB and LBEGS take more time to allocate appropriate resources, the response time is 70% & 43% longer than that using the PLBA. This effect could be caused by an increasing synchronization of the frequency request generation with the negotiations that lead to successful resource provisioning. In other words, the large number of open requests leads to longer queues for processing whereas the queues get shorter with decreasing request frequency, the overall processing gets faster.

Considering the resource allocation efficiency, from the results in figures 7-9, under small size Grid and low request

frequency (*Ri* = 200 ms), the resource allocation efficiency using PLBA is nearly the same as that using the WLB and LBEGS. When increasing the size of Grid by S = 100 and remain request frequency as Ri = 200 ms, the resource allocation efficiency of PLBA as much as 23% &12% larger that using the WLB and LBEGS respectively. When increasing the size of Grid by S = 500 and request frequency by R_i = 50 ms, the resource allocation efficiency of WLB and LBEGS is as much as 43% & 11% less than that using the PLBA. Under big size Grid (S = 2000) and high request frequency (R_i = 50), the resource all location efficiency of WLB and LBEGS is as much as 43% less than that using the PLBA. Resource allocation efficiency of 93% can be achieved for the PLBA with a 200 ms pulse even in high request density. Complete saturation of request in WLB and LBEGS can be attained in a small size Grid/high request density scenario. Increasing the request frequency implies a high charge to the system and in case of 50 ms pulses reduce the resource allocation efficiency to nearly 50% in all scenarios. In most of the test cases, the PLBA is more efficient than the WLB and LBEGS to allocate Grid resources in the test application. When Grid size is creasing, it has more merits to use the PLBA to schedule Grid resource; the PLBA has better performance than usual WLB and LBEGS.

CONCLUSIONS

This paper provides a sender initiate based dynamic load balancing algorithm for reducing communication overhead, response time and solving task allocation problem, wherein resources and the Grid broker participate in the load balancing operations. A dynamic threshold values has been used at each level according to the Grid size in the network. Furthermore, it extended by dividing lightly loaded node into lightly and under lightly loaded categories in the context of variable threshold and standard deviation. This approach minimizes the searching time to find out the receiver machine in order to transfer the Gridlets. We rigorously examined proposed algorithm on the GridSim simulator in order to judge the effectiveness of the algorithm. The simulation results have been provided to depict the effectiveness of the PLBA over WLB and LBEGS. It can enhance the execution time, resource allocation capacity, reduce the communication delay and response time. In the future, we will adjust our function of the balance threshold and make it more adaptive to differing environments.

REFERENCES

[1] Rehman A, Qureshi K, Manuel P, Rashid H, "Resource topology aware GridSim: a step ahead". J Comput 19 (2): special issue on Grid and Cluster Computing, 2008, pp. 13–22.

[2] Nazir B, Qureshi K, Manuel P, "Adaptive fault tolerant job scheduling strategy for economy based Grid", J Sup Comp, Oct-2008: pp. 116–134.

[3] Qureshi K, Hussain SS, "A comparative study of parallelization strategies for fractal image compression on a cluster of workstations". Int J Comput Methods 5(3): 2008, pp. 463–482.

[4] Zhu, Y., "A survey on Grid scheduling systems", Technical report, Department of Comp Sci, Hong Kong Univ.of Sci & Technology, 2003.

[5] 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, pp 13-15.

[6] Rathore N, Channa I, "A Cognitive Analysis of Load Balancing and job migration Technique in Grid" WICT-IEEE conference, Univ. of Mumbai, 2011.

[7] B. Yagoubi, Y. Slimani, "Dynamic load balancing strategy for Grid computing", Engg & Technology-2006, pp.90–95.

[8] Jerrell Watts, Stephen Taylor, "A Practical Approach to Dynamic Load Balancing", IEEE transaction on parallel and distributed system", (vol. 9 no. 3), March 1998, pp. 235-248.

[9] D. Erdil, M. Lewis, "Dynamic Grid load sharing with adaptive dissemination protocols", The J Super Comp 2010, pp. 1–28.

[10] S. Ludwig, A. Moallem, Swarm intelligence approaches for Grid load balancing, J of Grid Computing, 2011, pp. 1–23.

[11] PENG Limin, XIAO Wenjun, "A Binary-Tree based Hierarchical Load Balancing Algorithm in Structured Peer-to-Peer Systems", Journal of Convergence Information Tech, Volume 6, Number 4, April 2011.

[12] Weiwei Lin,Wuyao Shen, "Tree-Based Task Scheduling Model and Dynamic Load-Balancing Algorithm for P2P Computing", South China University of Technology, Guangzhou, China, 10th IEEE International Conference on Computer and Information Technology-CIT 2010.

[13] Somayeh Abdi and Somayeh Mohamadi, "The Impact of Data Replication on Job Scheduling Performance in Hierarchical Data Grid", International journal on applications of graph theory in wireless ad hoc networks and sensor networks (GRAPH-HOC) Vol.2, No.3, Sep 2010.

[14] Malarvizhi Nandagopal, Rhymend V Uthariaraj, "Hierarchical Status Information Exchange Scheduling and Load Balancing For Computational Grid Environments", IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.2, February 2010.

[15] Gengbin Zheng, Abhinav Bhatele, Esteban Meneses, Laxmikant V. Kale, "Periodic Hierarchical Load Balancing for Large Supercomputers", 2010.

[16] Bin Wang and Qing-guo Shen, "ID Management and Allocation Algorithm for P2P Load Balancing", IEEE -2010.

[17] Robson E. De Grande, Azzedine Boukerche, "Dynamic balancing of communication and computation load for HLA-based simulations on large-scale distributed systems", J. Parallel Dis. Comp, 2011, pp. 40–52.

[18] Srinivas Mandalapu, "Dynamic Load Balancing Design and Modeling in MPIAB", General Architecture PDPTA: 2005, pp. 710-716.

[19] Azzedine Boukerche and Robson Eduardo De Grande, "Dynamic Load Balancing Using Grid Services for HLA-Based Simulations on Large-Scale Distributed Systems", 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications, 2009.

[20] Qureshi k., Rehman A., "Enhanced GridSim architecture with load balancing", J. Supercomputing 57: 2011, pp. 265-275.

[21] Chunlin Li, Layuan Li, "Competitive proportional resource allocation policy for computational Grid", FGCS 20, 2004, pp. 1041–1054.

[22] M.Kamarunisha, S.Ranichandra, T.K.P.Rajagopal, "Recitation of Load Balancing Algorithms In Grid Computing Environment Using Policies And Strategies - An Approach", Int Journal of Scientific & Engineering Research Volume 2, Issue 3, ISSN 2229-5518, March-2011.

[23] B. Yagoubi, Y. Slimani, "Task Load Balancing Strategy for Grid Computing", Journal of Computer Science 3 (3): ISSN 1546-9239, Science Publications-2007, pp. 186-194.

[24] Payli RU, Yilmaz E, Ecer A, Akay HU, Chien S, "DLB—a dynamic load balancing tool for Grid computing, scalable computing". Pract Exp 7(2): 2006, pp. 15–23.

[25] Said Fathy El-Zoghdy, Shebin El-Koom, Egypt, "A Load Balancing Policy for Heterogeneous Computational Grids", (IJACSA) International J of Advanced Computer Science and Applications, Vol. 2, No. 5, 2011.