(IJIRSE) International Journal of Innovative Research in Science & Engineering
ISSN (Online) 2347-3207

Effective Load Balancing in Grid Environment

[1]Mr. D. S. Gawande, [2]Mr. S. B. Lanjewar, [3]Mr. P. A. Khaire, [4]Mr. S. V. Ugale

[1,2,3] Lecturer , CSE Dept, DBACER, Nagpur, India

[4] Lecturer, CSE Dept, GWCET, Nagpur, India

Abstract— **The computational grid is a new parallel and distributed computing paradigm that provides resources for large scientific computing applications. It typically consists of heterogeneous resources such as clusters that may reside in different administrative domains, be subject to different access policies and be connected by networks with widely varying performance characteristics. Many researchers have been proposed numerous scheduling and load balancing techniques for locally distributed multiprocessor systems. However, they suffer from significant deficiencies when extended to a grid environment. Computational grids have the potential for solving large-scale scientific computing applications. The main techniques that are most suitable to cope with the dynamic nature of the grid are the effective utilization of grid resources and the distribution of application load among multiple resources in a grid environment. In this paper contain short literature study on generic load balancing model, load balancing policies and propose scheduling and load balancing approach.**

Keywords: Grid computing, load balancer, Response Time

I. Introduction

The rapid development in computing resources has enhanced the performance of computers and reduced their costs. This availability of low cost powerful computers coupled with the popularity of the Internet and high-speed networks has led the computing environment to be mapped from distributed to Grid environments. In fact, recent researches on computing architectures are allowed the emergence of a new computing paradigm known as Grid computing. Grid is a type of distributed system which supports the sharing and coordinated use of geographically distributed and multi owner resources, independently from their physical type and location, in dynamic virtual organizations that share the same goal of solving large-scale applications. In order to fulfill the user expectations in terms of performance and efficiency, the Grid system needs efficient load balancing algorithms for the distribution of tasks. A load balancing algorithm attempts to improve the response time of user's submitted applications by ensuring maximal utilization of available resources. The main goal is to prevent, if possible, the condition where some processors are overloaded with a set of tasks while others are lightly loaded or even idle [2]. Although load balancing problem in conventional distributed systems has been intensively studied, new challenges in Grid computing still make it an interesting topic and many research projects are under way. This is due to the characteristics of Grid computing and the complex nature of the problem itself. Load balancing algorithms in classical distributed systems, which usually run on homogeneous and dedicated resources, cannot work well in the Grid architectures. Load balancing involves assigning job to a resource proportional to its performance, thereby minimizing the response time of a job. However, there are wide varieties of issues that need to be considered for a heterogeneous grid environment. For example, processing capacities of the resources may differ and their usable capacities may vary according to the load imposed upon them. Further, 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. Therefore, in the grid environment it is essential to consider the impact of various dynamic characteristics on the design and analysis of scheduling and load balancing algorithms. Due to uneven job arrival patterns and unequal computing capacities, one resource may be overloaded while others may be underutilized. It is therefore desirable to dispatch jobs to idle or lightly loaded resources to achieve better resource utilization and reduce the mean job response time. The strategy proposed here is to perform scheduling and balancing the application load in the grid environment by taking resource heterogeneity, communication delay and network heterogeneity into consideration.

II. Literature Review

Previous related work [2] addresses the problem of scheduling and load balancing in a grid architecture where computational resources are dispersed in different administrative domains or clusters which are connected to the grid scheduler by means of heterogeneous communication bandwidths is considered. In this reference [14], the problem of transferring files that are generated during the execution of DAG workflows with interdependent tasks is addressed. The ineffectiveness of advanced file-transfer techniques in these cases is discussed, and a heuristic is proposed for dealing with the scheduling of interdependent and independent tasks that arrive on-line to be processes by grid infrastructure. The Best File-Transfer Time (BFTT) heuristic is proposed in this study as a means to circumventing the problem of reducing the time spent for transferring data files among different resources in the grid, while still ensuring good performance and/or good load balance among the resources. In addition, BFTT is implemented in this work in conjunction with the OLB algorithm, and a few tests for verifying its results are performed and discussed. One of the main load balancing methods mentioned in the reference [7] is dynamic decentralized approach. This approach considers the run time environment before distributing the jobs among the nodes of the grid. The dynamic decentralized approach is preferred because elements of the grid may vary in capacity or number during runtime and also be heterogeneous in nature giving rise to different loading conditions. In this paper we compare the different load balancing algorithms for the grid based on various metrics like communication overhead, load balancing time, scalability, fault tolerance, reliability and stability. [13]The GridSim toolkit provides a comprehensive facility for simulation of different classes of heterogeneous resources, users, applications, resource brokers, and schedulers. It can be used to simulate application schedulers for single or multiple administrative domains distributed computing systems such as clusters and Grids. Application schedulers in the Grid environment, called resource brokers, perform resource discovery, selection, and aggregation of a diverse set of distributed resources for an individual user. This means that each user has his or her own private resource broker and hence it can be targeted to optimize for the requirements and objectives of its owner. In contrast, schedulers, managing resources such as clusters in a single administrative domain, have complete control over the policy used for allocation of resources. This means that all users need to submit their jobs to the central scheduler, which can be targeted to perform global optimization such as higher system utilization and overall user satisfaction depending on resource allocation policy or optimize for high priority users.

III. Load Balancing In Grid Environment

A. *Grid Topology*

This is topological structure for a Grid computing Grid computing is a finite set of G clusters Ck, interconnected by gates gtk, k belongs to {0, ...,G − 1}, where each cluster contains one or more sites Sjk interconnected by switches SWjk and every site contains some Computing Elements CEijk and some Storage Elements SEijk, interconnected by a local area network[1]. This model is based on an incremental tree. First, for each site it creates a two-level subtree. The leaves of this subtree correspond to the computing elements of a site, and the root is a virtual node associated to the site. These sub trees, that correspond to sites of a cluster, are then aggregated to form a three-level sub-tree.

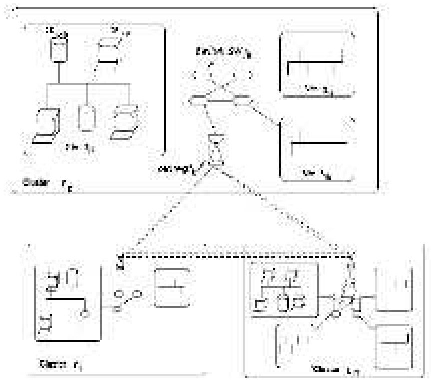


Figure 1: Grid Topology [1]

B. Load Balancing Policies

PoliciesLoad balancing algorithms can be defined by their implementation of the following policies [5]:
• Information policy: specifies what workload information to be collected, when it is to be collected and from where.
• Triggering policy: determines the appropriate period to start a load balancing operation.
• Resource type policy: classifies a resource as server or receiver of tasks according to its availability status.
• Location policy: uses the results of the resource type policy to find a suitable partner for a server or receiver.
• Selection policy: defines the tasks that should be migrated from overloaded resources (source) to most idle resources (receiver).
The main objective of load balancing methods is to speed up the execution of applications on resources whose workload varies at run time in unpredictable way. Hence, it is significant to define metrics to measure the resource workload. Every dynamic load balancing method must estimate the timely workload information of each resource. This is key information in a load balancing system where responses are given to following questions:
• How to measure resource workload?
• What criteria are retaining to define this workload?
• How to avoid the negative effects of resources dynamicity on the workload; and,
• How to take into account the resources heterogeneity in order to obtain an instantaneous average workload representative of the system? Several load indices have been proposed in the literature, like CPU queue length, average CPU queue length, CPU utilization, etc. The success of a load balancing algorithm depends from stability of the number of messages (small overhead), support environment, low cost update of the workload, and short mean response time which is a significant measurement for a user. It is also essential to measure the communication cost induced by a load balancing operation.

III. Proposed System

In grid environments, the shared resources are dynamic in nature, which in turn affects application performance. Workload and resource management are two essential functions provided at the service level of the Grid software infrastructure. To improve the global throughput of these environments, effective and efficient load balancing algorithms are fundamentally important. Load Balancing is one of the most important factors which can affect the performance of the grid application. All Load Balancing algorithms implement five policies. The efficient implementation of these policies decides overall performance of Load Balancing algorithm. The main objective of this thesis is to propose an efficient Load Balancing Algorithm for Grid environment. Main difference between existing Load Balancing algorithm and proposed Load Balancing is in implementation of two policies: Triggering Policy and Selection Policy. For implementation of Triggering Policy all existing Load Balancing algorithm use periodic approach, which is time consuming. The proposed approach uses activity based approach for implementing Triggering policy. For implementation of Selection Policy Proposed algorithm uses Job length as a parameter, which can be used more reliably to make decision about selection of job for migration from heavily loaded node to lightly loaded node.

IV. Experimental Environment

A. GridSim Simulation ToolKit

The simulation was carried out on the excellent grid simulation toolkit GridSim ToolKit 5.0 [13] which allows modeling and simulation of entities in grid computing systems-users, applications, resources, and resource load balancers for design and evaluation of load balancing algorithms. A heterogeneous grid environment by using various resource specifications was built. It proposes the method of creating a user job and different types of heterogeneous resources. The resources differ in their operating system type, CPU speed, RAM memory, MIPS rating.

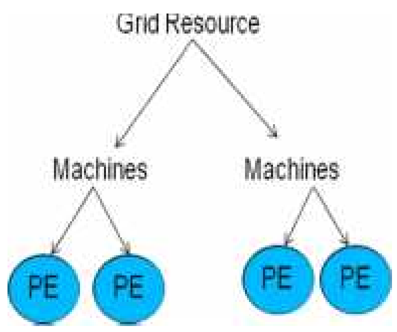
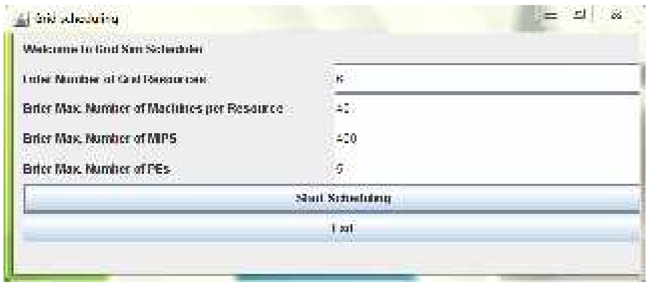


Fig. 2 GridSim Resource Hierarchy

In GridSim, application jobs are modeled as Gridlet objects that contains all information related to the job and the execution management. Details of the available Grid resources are obtained from Grid Information Service (GIS) entity that keeps track of the resources available in the grid environment. The experimental environment consisting of hierarchy of resources used for the evaluation of proposed algorithm is shown in figure 2. A grid resource (GS) maintains information about machines (LS) and each machine contains PEs running at different speeds.

B. Simulation Result

Number of grid resource, number of machine, number of processing element and MIPS is provide as a input to module 1 as shown in Figure 3.



Welcome to Grid Sim Scheduler

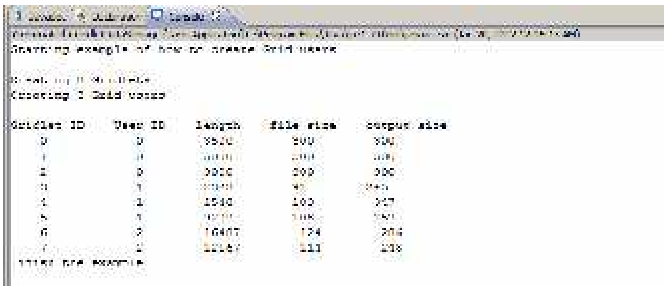
Enter Number of Grid Resources	6
Enter Max. Number of Machines per Resource	40
Enter Max. Number of MIPS	400
Enter Max. Number of PEs	5

Figure 3 : Grid Implementation Input Frame

Gridlet/Tasks Creation

- Gridsim provide facility to create various tasks
- The tasks has various characteristics..
  - Gridlet id
  - Length
  - File size
  - Output size

GridletDataStorage class is used for creating various tasks and creating number of users. In Figure 4. show the result of creating gridlet or tasks and users.



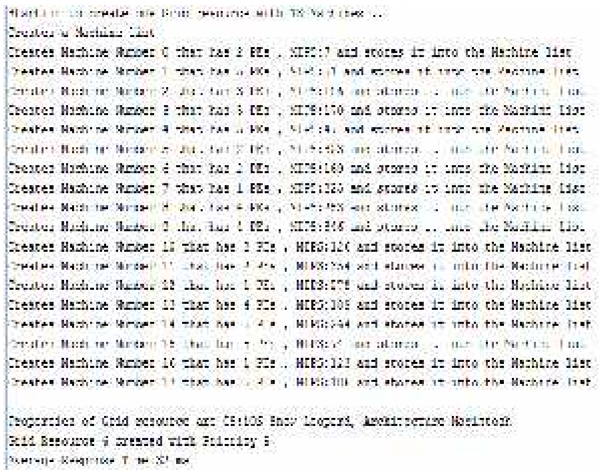
```
[illegible]
Starting example of how to create Grid users

Creating 8 Gridlets
Creating 3 Grid users

Gridlet ID    User ID    length    file size    output size
    0            0        3500        300          300
    1            0        [illegible] [illegible]  [illegible]
    2            0        5000        300          300
    3            1        [illegible] 4[illegible] [illegible]
    4            1        1540        100          207
    5            1        [illegible] 108          [illegible]
    6            2        16407        124          204
    7            2        12167        111          238
 Finish the example
```

Figure 4: Result of creating gridlet and user

Here 8 Gridlets and 3 Grid users are created and each gridlet assign their properties. Tasks are randomly generated and divide among the resources and simulate by using gridsim simulator toolkit.

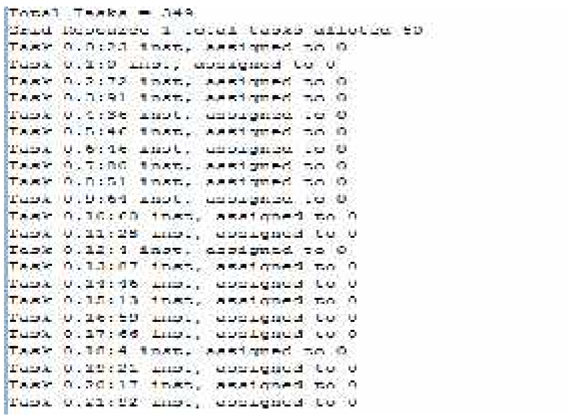


```
Starting to create one Grid resource with [illegible] ...
Creates a Machine List
Creates Machine Number 0 that has 2 PEs , MIPS:7 and stores it into the Machine list
Creates Machine Number 1 that has [illegible] PEs , MIPS:[illegible] and stores it into the Machine list
Creates Machine Number 2 that has 3 PEs , MIPS:114 and stores it into the Machine list
Creates Machine Number 3 that has 5 PEs , MIPS:150 and stores it into the Machine list
Creates Machine Number 4 that has [illegible] PEs , MIPS:[illegible] and stores it into the Machine list
Creates Machine Number 5 that has 2 PEs , MIPS:[illegible] and stores it into the Machine list
Creates Machine Number 6 that has 2 PEs , MIPS:160 and stores it into the Machine list
Creates Machine Number 7 that has 1 PEs , MIPS:220 and stores it into the Machine list
Creates Machine Number 8 that has 4 PEs , MIPS:[illegible] and stores it into the Machine list
Creates Machine Number 9 that has 1 PEs , MIPS:[illegible] and stores it into the Machine list
Creates Machine Number 10 that has 3 PEs , MIPS:120 and stores it into the Machine list
Creates Machine Number 11 that has 2 PEs , MIPS:[illegible] and stores it into the Machine list
Creates Machine Number 12 that has 1 PEs , MIPS:[illegible] and stores it into the Machine list
Creates Machine Number 13 that has 4 PEs , MIPS:105 and stores it into the Machine list
Creates Machine Number 14 that has [illegible] PEs , MIPS:244 and stores it into the Machine list
Creates Machine Number 15 that has [illegible] PEs , MIPS:[illegible] and stores it into the Machine list
Creates Machine Number 16 that has 1 PEs , MIPS:123 and stores it into the Machine list
Creates Machine Number 17 that has [illegible] PEs , MIPS:[illegible] and stores it into the Machine list

Properties of Grid resource are OS:IOS Snow Leopard, Architecture Macintosh
Grid Resource 4 created with Priority 5
Average Response Time:[illegible] ms
```

Figure 5: Image of creating grid Resource

In this image number of grid resource are created with properties and priority and store into machine list. Each grid resource having number of machines, PEs and MIPS is assign and each resource having different properties associated with it.



```
Total Tasks = [illegible]
Grid Resource 1 total tasks allotted [illegible]
Task 0.0:23 inst. assigned to 0
Task 0.1:9 inst. assigned to 0
Task 0.2:72 inst. assigned to 0
Task 0.3:91 inst. assigned to 0
Task 0.4:36 inst. assigned to 0
Task 0.5:46 inst. assigned to 0
Task 0.6:16 inst. assigned to 0
Task 0.7:86 inst. assigned to 0
Task 0.8:51 inst. assigned to 0
Task 0.9:64 inst. assigned to 0
Task 0.10:60 inst. assigned to 0
Task 0.11:25 inst. assigned to 0
Task 0.12:1 inst. assigned to 0
Task 0.13:87 inst. assigned to 0
Task 0.14:46 inst. assigned to 0
Task 0.15:13 inst. assigned to 0
Task 0.16:59 inst. assigned to 0
Task 0.17:66 inst. assigned to 0
Task 0.18:4 inst. assigned to 0
Task 0.19:21 inst. assigned to 0
Task 0.20:17 inst. assigned to 0
Task 0.21:22 inst. assigned to 0
```

Figure 6: Image of Tasks Scheduling

In this image show that the number of tasks are assigned to resource with length of that tasks.

C. Performance Parameter

The following metrics were selected to evaluate the performance of the proposed model:

- Average Response Time: Response time rj of job j is the time period from the job arrival to the completion time of the job. i.e., the time spent in the resource queue plus the job service (execution) time.

The Average response time ART:

$$ART=1/N \sum_{j=1}^{N} rj$$

Where, N is the total number of processed jobs.

Here Average response time is used for analyze the system performance before load balancing or no load balancing and after load balancing by using proposed load balancing algorithm. Number of samples is collected by performing the simulation number of times and note down the Average response time, before load balancing and after load balancing the system load. All time units are in milliseconds so the performance metrics are also measured in milliseconds.

Table 1: Average Response Time

No. of Tasks	No Load Balancing	Proposed Load Balancing
250	32	16
500	31	16
1000	35	15
1500	31	15
2000	47	32
2500	47	16

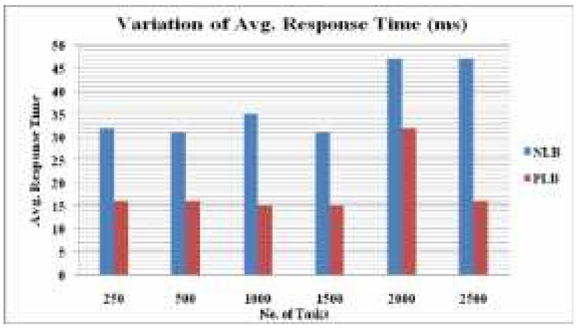


Figure 7: ART versus No. of Tasks

In the simulation, it is observed that Average response time with no load balancing is much higher than load balancing with proposed algorithm.

V. Conclusion

The proposed load balancing model takes into account the heterogeneity of the computational and network resources. i.e., the resources are with different processing capacity and network bandwidth. The load balancing policies at various levels of hierarchy are proposed to optimize various performance metrics. This thesis focuses on load balancing and presents factors due to which load balancing is initiated, and finally proposes an efficient load balancing algorithm for Grid environment.
Some of the limitations of this work and present some possible directions for future research. In this work, we assume that there is no precedence constraint among different jobs or different tasks of a job. Usually, the jobs are independent of each other in the grid, but different tasks of a job may have some precedence constraints. Hence, it is an interesting direction for future research. Such dependencies will not only make the problem extremely difficult to solve, but would also require estimating a very large number of parameters. In future we should also consider some fault tolerant measures to increase the reliability of our algorithm.

REFERENCES

[1].Belabbas Yagoubi and Yahya Slimani, “Dynamic Load Balancing Strategy for Grid Computing” Proceedings of World Academy of Science, Engineering and Technology Volume 13 May 2006 ISSN 1307-6884.

[2] Malarvizhi Nandagopal1 and Rhymend V Uthariaraj2 "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.

[3].S.Rips "Load Balancing Support for Grid-enabled Applications" NIC Series, Vol. 33, ISBN 3-00- 017352-8, pp. 97-104, 2006.

[4].Po-Cheng Chen,Cheng-I Lin,Sheng-Wei Huang,Jyh- Biau Chang, Ce-Kuen Shieh,Tyng-Yeu Liang, "A Performance Study of Virtual Machine Migration vs Thread Migration for Grid Systems".

[5].S Kalaiselvi Supercomputer Education and Research Centre (SERC), Indian Institute of Science, Bangalore V Rajaraman Jawaharlal Nehru Centre for Advanced Scientific Research, Indian Institute of Science Campus, Bangalore "A Survey of Check-Pointing Algorithms for Parallel and Distributed Computers" vol. 25,Part 5,October 2000,pp.489±510.

[6] A. Abed, G. Oz, A. Kostin, Competition-Based Load Balancing for Distributed Systems, Proceedings of the Seventh IEEE International Symposium on Computer Networks (ISCN' 06),pp 230 – 235.

[7] D. Acker, S. Kulkarni, "A Dynamic Load Dispersion Algorithm for Load-Balancing in a Heterogeneous Grid System" Sarnoff Symposium IEEE, May 2007, pp 1- 5.

[8] Dobber, M., Koole, G., Mei, R.: Dynamic load balancing experiments in a Grid. In: Proceedings of IEEE International Symposium on Cluster Computing and the Grid, Cardiff, 2005.

[9] Kai Lu, Riky Subrata and Albert Y. Zomaya, "An Efficient Load Balancing Algorithm for Heterogeneous Grid Systems Considering Desirability of Grid Sites", IEEE International Performance, Computing, and Communications Conference, 2006. IPCCC 2006. Vol 25Page(s):9 pp. – 320.

[10] Kimura, K., Ichiyosi, N.: Probabilistic analysis of the optimal efficiency of the multi-level dynamic load balancing schemes. In: Proceedings of the 6th Distributed Memory Computing Conference, Portland, (1991)

[11] Kameda, H., Li, J., Kim, C., Zhang, Y.: Optimal Load Balancing in Distributed Computer Systems. Springer, London (1997).

[12].Shahzad Malik, "Dynamic Load Balancing in a Network of Workstations", 95.515F Research Report, November 29, 2000.

[13] Rajkumar Buyya1,∗,† and Manzur Murshed2 "GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing" CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE *Concurrency Computat.: Pract. Exper.* 2002; **14**:1175–1220 (DOI: 10.1002/cpe.710)

[14] Elaine C. Machtans1,2; Liria M. Sato2; Airton Deppman3"Improvement on Scheduling Dependent Tasks for Grid Applications" 2009 International Conference on Computational Science and Engineering