

## A Load Balancing in Grid Environment

Mr. Dinesh S. Gawande \*, Asst. Prof. Rajesh C. Dharmik \*\*, Ms. Chanda Panse \*\*\*

\*(Department of Computer Technology, Yeshwantrao Chavan College of Engineering, Nagpur, India.

\*\* (Department of Information Technology, Yeshwantrao Chavan College of Engineering, Nagpur, India.

\*\*\* (Department of Computer Technology, Yeshwantrao Chavan College of Engineering, Nagpur, India.

### ABSTRACT

Grid computing is being adopted in various areas from academic, industry research to government use. Grids are becoming platforms for high performance and distributed computing. The computational grid is a new parallel and distributed computing paradigm that provides resources for large scientific computing applications. 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 and process of grid implementation by using Gridsim toolkit.

**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 REVUIWE

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 processed 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. Load Balancing Approaches

Load balancing problem has been discussed in traditional distributed systems literature for more than two decades. Various algorithms, policies have been classified [15].

- **Load Balancing Algorithms** Algorithms can be classified into two categories: static or dynamic.

#### (a) Static Load Balancing Algorithm

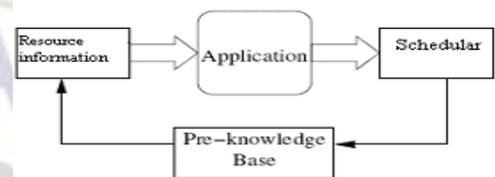


Figure 1: Static Load Balancing

Static load balancing algorithms allocate the tasks of a parallel program to workstations based on either the load at the time nodes are allocated to some task, or based on an average load of our workstation cluster. The decisions related to load balance are made at compile time when resource requirements are estimated. The advantage in this sort of algorithm is the simplicity in terms of both implementation as well as overhead, since there is no need to constantly monitor the workstations for performance statistics. However, static algorithms only work well when there is not much variation in the load on the workstations. Clearly, static load balancing algorithms aren't well suited to a Grid environment, where loads may vary significantly at various times.

#### (b) Dynamic Load Balancing Algorithm

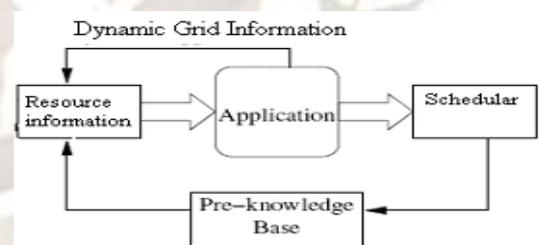


Figure 2: Dynamic Load Balancing

Dynamic load balancing algorithms make changes to the distribution of work among workstations at run-time; they use current or recent load information when making distribution decisions. Multicomputers with dynamic load balancing allocate/reallocate resources at runtime based on no a priori task information, which may determine when and whose tasks can be migrated. As a result, dynamic load balancing algorithms can provide a significant improvement in performance over static algorithms.

### **B. Load Balancing Policies**

Load 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
- 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.

### **C. Scheduling and Load Balancing**

Grid computing enables sharing, selection and aggregation of large collections of geographically and organizationally distributed heterogeneous resources for solving large-scale data and compute intensive problems. Resources are dynamic in nature so the Load of resources varies with change in configuration of Grid and it makes Load Balancing more important in case of Grid environment. 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. The focus of our study is to consider factors which can be used as characteristics for decision making to initiate Load

Balancing. Load Balancing is one of the most important factors which can affect the performance of the grid application. The main objective 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 three policies: Information Policy, Triggering Policy and Selection Policy. For implementation of Information Policy all existing Load Balancing algorithm use periodic approach, which is time consuming. The proposed approach uses activity based approach for implementing Information policy. For Triggering Load Balancing proposed algorithm uses two parameters which decide Load Index. On the basis of Load Index Load Balancer decide to activate Load Balancing process. 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. The choice of a load balancing algorithm for a Grid environment is not always an easy task. Some load balancing strategies work well for applications with large parallel jobs, while others work well for short, quick jobs. Some strategies are focused towards handling data-heavy tasks, while others are more suited to parallel tasks that are computation heavy. While many different load balancing algorithms have been proposed, there are four basic steps that nearly all algorithms have in common:

- Monitoring workstation performance (load monitoring)
- Exchanging this information between workstations (synchronization)
- Calculating new distributions and making the work movement decision (rebalancing criteria)
- Actual data movement (job migration)

Efficient Load Balancing algorithm makes Grid Middleware efficient and which will ultimately leads to fast execution of application in Grid environment. In this paper, an attempt has been made to formulate a decentralized, sender-initiated load balancing algorithm for Grid environments which is based on different parameters. One of the important characteristics of this algorithm is to estimate system parameters such as queue length and CPU utilization of each participating nodes and to perform job migration if required. The propose Load balancing should take place when the load situation has changed. There are some particular activities which change the load configuration in Grid environment. The activities can be categorized as following:

- Arrival of any new job and queuing of that job to any particular node.
- Completion of execution of any job.
- Arrival of any new resource
- Withdrawal of any existing resource.

Whenever any of these four activities happens activity is communicated to master node then load information is collected and load balancing condition is checked. If load balancing condition is fulfilled then actual load balancing

activity is performed. Here actual load distribution is performed at a centralized controller or manager node. The central controller polls each workstation and collects state information consisting of a node's current load as well as the number of jobs in the node's queue. The polling is done on basis of occurrence of some defined activity. In the proposed algorithm information is collected only if there is a change in configuration of Grid. This information is used to perform load balancing. First of all it initializes different parameters. Whenever any of four activities which are required to start information policy of load balancing occurs, it starts collecting load balancing information. Once information has been gathered then it is decided that load balancing is required or not. For this purpose application uses CPU utilization and queue length parameters. With help of these parameters we decide which resource is heavily loaded and which resource is lightly loaded. After selection of resource the application selects job out of n-jobs running on that resource. This selection is based upon on CPU consumption of different jobs. Least CPU consumed job will be selected for migration. When job is selected, application checks for available lightly loaded resource. If lightly loaded resource is available then migrate selected job from heavily loaded resource to lightly loaded resource. If no lightly loaded resource is available then add selected job to pending job list. This job will be executed later when some lightly loaded resource will be available.

#### IV. EXPERIMENTAL ENVIRONMENT

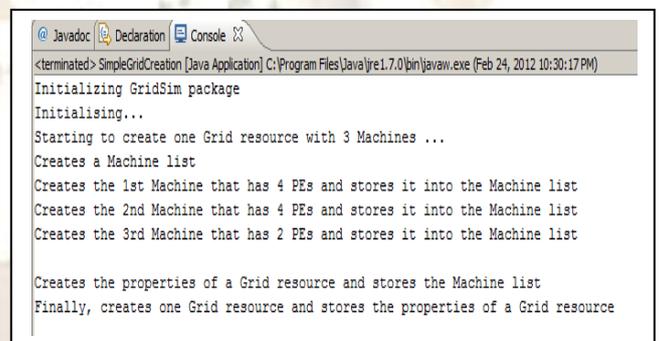
##### A. GridSim Simulation ToolKit

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.

##### B. Methodology

The GridSim [11] toolkit provides a comprehensive facility for simulation of different classes of heterogeneous resources, users, applications, resource brokers, and schedulers. For that first the toolkit is installed on the computer. Then Gridsim package is imported to simulate Gridsim environment. The users can be created in Gridsim and there can be more than one user at a time using the grid. Below are the steps to be followed in JAVA code to create a Grid Resource.

- Create an object of type MachineList to store one or more Machines  
MachineList mList = new MachineList();
- A Machine contains one or more PEs/ CPUs. So we create an object of type PEList to store this PEs before creating a Machine.  
PEList peList1 = new PEList();
- Create PEs and add these into the object of PEList created in step 2. We have to specify the unique ID of the PE as first parameter and its MIPS (Millions of Instruction per Second) rating as second parameter.  
peList1.add( new PE(0, 377) );
- Create a Machine with its unique ID and the PEList associated with it.  
mList.add( new Machine(0, peList1) );
- Repeat the steps from step 2 to step 4 to create additional number of machines.
- Create a Resource Characteristics object which will store the properties of a Grid Resource.  
ResourceCharacteristics resConfig = new  
ResourceCharacteristics(arch ,osm List,  
ResourceCharacteristics.TIME\_SHARED,  
time\_zone, cost);



```
<terminated> SimpleGridCreation [Java Application] C:\Program Files\Java\jre1.7.0\bin\javaw.exe (Feb 24, 2012 10:30:17 PM)
Initializing GridSim package
Initialising...
Starting to create one Grid resource with 3 Machines ...
Creates a Machine list
Creates the 1st Machine that has 4 PEs and stores it into the Machine list
Creates the 2nd Machine that has 4 PEs and stores it into the Machine list
Creates the 3rd Machine that has 2 PEs and stores it into the Machine list

Creates the properties of a Grid resource and stores the Machine list
Finally, creates one Grid resource and stores the properties of a Grid resource
```

Figure: 3 Grid Resource creation in Gridsim

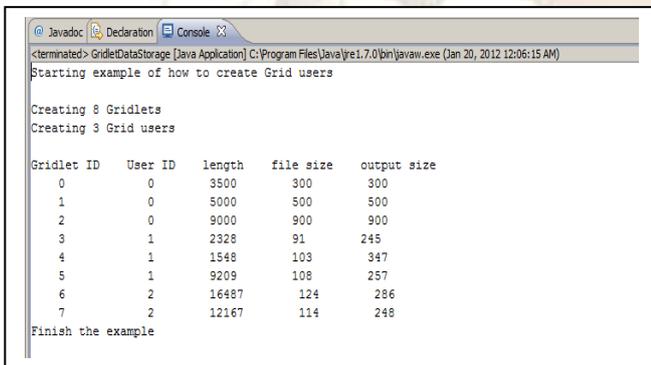
In the final step we create an object of Grid Resource specifying its name, communication speed, peak load, off peak load, holiday load and list of holidays along with the Resource Characteristics object which was created in the step 6. Resource creation is shown in Fig. 3

The steps to create Grid User(s) in GridSim are straightforward, but each User must have a unique ID. The steps to create a Grid User are as follows:

1. Create an object of type ResourceUserList.  
ResourceUserList userList = new ResourceUserList();
2. Add to this list Grid users specifying a unique ID, first user has to have ID equal to 0.  
Just keep on adding Grid users to this list if we wish to create to more of them.  
userList.add(0);

In the terminology of GridSim, a job which can run sequentially and independently on a Grid Resource is called a Gridlet. After the creation of Grid Resource we create Gridlets in the GridSim which can then be submitted to the latter. We need to specify the length of Gridlet, its output file size, its input file size, its unique ID for simulation. Gridlet creation can be done in two modes first is the manual option and second is with the use GridSim Random functions to take care of the statistical needs of highly unpredictable Grid environment simulation. The steps to create Gridlet(s) in manual modes are:

1. Create an object of type GridletList  
GridletList list = new GridletList();
2. Create an object of type Gridlet specifying its unique id, length, input file, output file size in types integer, double, long integer and long integer respectively.  
Gridlet gridlet1 = new Gridlet(id, length, file\_size, output\_size);
3. Add the object created in step 2 to the Gridlet list created in step 1.  
list.add(gridlet1);
4. To create more Gridlets repeat from step 1.



```

<terminated> GridletDataStorage [Java Application] C:\Program Files\Java\jre1.7.0\bin\javaw.exe (Jan 20, 2012 12:06:15 AM)
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         5000     500         500
2            0         9000     900         900
3            1         2328     91          245
4            1         1548     103         347
5            1         9209     108         257
6            2         16487    124         286
7            2         12167    114         248

Finish the example
    
```

**Figure 4: Gridlet and user creation in Gridsim**

## V. CONCLUSION

This paper propose method used for load estimation and load distribution effectively. The focus of paper is to consider factors which can be used as characteristics for decision making to initiate Load Balancing. The load balancing algorithm for the grid can be made more robust by scheduling all jobs irrespective of any constraints so as to balance the load perfectly. The grid resource, gridlet and grid users are created by using Gridsim toolkit. After performing load balancing the efficiency of algorithm is calculated in terms of resource utilization, throughput and response time of system in further work.

## REFERENCES

- [1]. Belabbas Yagoubi and Yahya Slimani, "Dynamic Load Balancing Strategy for Grid Computing"
- [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 25 Page(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 Buyya<sup>1,\*</sup>,† and Manzur Murshed<sup>2</sup> "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. Mactans<sup>1,2</sup>; Liria M. Sato<sup>2</sup>; Airton Deppman<sup>3</sup>"Improvement on Scheduling Dependent Tasks for Grid Applications" 2009 International Conference on Computational Science and Engineering.
- [15] Javier Bustos Jimenez, Robin Hood: An Active Objects Load Balancing Mechanism for Intranet.