Hierarchical Load Balancing GridSim Architecture with Fault Tolerance

D.Nanthiya/ PG Student, P.Keerthika M.E.,/Assistant Professor (Sr.G)

Abstract— Grid computing enables sharing, selection and aggregation of large collection of geographically and organizationally distributed heterogeneous resources for solving large-scale data and compute intensive problems. Resources are dynamic and heterogeneous in nature. So the load of each resource varies with change in configuration of grid and also the occurrence of failure of resources is much greater. Moreover the failure of the resources affects the job execution fatally. This makes load balancing and fault tolerance more important in case of computing grid. This paper introduces an algorithm which balances the load among the resources and also increases the reliability of the grid environment. This algorithm consists of two-phases. In the first phase, the fittest resources are selected based on the expected completion time and fault tolerant factor of the resources. In the second phase, the load balancing algorithm is applied to determine the status of the selected resources and then the jobs are scheduled to the resources if and only if the load is balanced. The proposed algorithm is simulated on GridSim simulator for effective modeling of the grid environment.

Index Terms— fault tolerance, fault rate, grid computing, gridsim , load balancing,makespan,resource..

1 INTRODUCTION

In today's pervasive world of needed information anytime and anywhere, the explosive grid computing environments have now proven to be so significant that they are often referred to as being the world's single and most powerful computer solutions. The Grid computing discipline involves the actual networking services and connections of a potentially unlimited number of ubiquitous computing devices within a grid. Grid computing [1] provides highly scalable, highly secure and extremely high performance mechanisms for discovering and negotiating access to remote computing resources in a seamless manner. This makes it possible for the sharing of computing resources, on an unprecedented scale among an infinite number of geographically distributed groups. So, the individual users can access computers and data transparently without having to consider location, operating system, account administration, and other details. Grids tend to be more loosely coupled, heterogeneous and geographically distributed across multiple administrative domains and owned by different organizations.

Grid computing environments must be constructed upon the following foundations coordinated resources-the infrastructure should provide coordination among the resources, based on respective policies and service-level agreements. Open standard protocols and frameworks-The use of open standards provides interoperability and integration facilities. The standards must be applied for resource discovery, resource access and resource coordination.

The grid systems are classified into two types:-Computational grids, in which machines with set-aside resources stand by to crunch data or provide coverage for other intensive workloads and Data grids [2], which provide a unified interface for all data repositories in an organization, and through which data can be queried, managed, and secured.

The grid environment consists of several components such as schedulers, load balancer, grid broker, and portals. The schedulers are the types of applications that are responsible for the management of jobs, such as allocating resources needed for any specific job, partitioning of jobs to schedule parallel execution of tasks, data management, event correlation and a service level management capabilities. The resource broker provides pairing services between the service requester and the service provider. This pairing enables the selection of best available resources from the service provider for the execution of a specific task. The load balancer is responsible for distributing the workload among the resources. This load-balancing feature must always be integrated into the system in order to avoid processing delays and overcommitment of resources. The grid portals help free end users from the complexity of job management and resource allocation.

The performance of the grid environment is strongly affected by the improper distribution of load among resources and the frequent resource failures. This paper proposes an algorithm which considers the failure rate and user deadline as major deciding factor for the distribution of the load among the resources.

2. LITERATURE REVIEW

The grid environment consists of dynamic and heterogeneous resources. These resources changes with time i.e., any new resource can join or any of the old resources can exit the grid environment at any time. So the selection of the available resources meeting the user deadline [19] in such a dynamic and time variant environment is very complex. Similarly designing a proper job scheduling algorithm considering the resource characteristics is very important. Blythe et al.[20] uses Min-min task scheduling algorithm which schedules the job based on the (ECT) Expected Computation Time value. The experimental results show that this algorithm reduces the makespan and the response time of the submitted jobs.

Due to uneven job arrival patterns and unequal computing capabilities, some resources in the grid environment get overloaded or some resources get underloaded or some resources remain idle. The load balancing mechanism helps to distribute the load among the resources from overloaded resource to underloaded resource. This helps to achieve high throughput, reduced response time, optimized utilization of the resources.

The load balancing mechanism is generally based on four policies namely information policy, triggering policy, resource type policy, selection policy. The 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. The Resource type 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).

Yagoubi and Slimani [2][3][4] were proposed the layered load balancing algorithm, which is based on the tree model representation .This model transforms any grid architecture into a unique tree with at most four levels. The main features of this algorithm are (i) it is layered. (ii) it supports heterogeneity and scalability. (iii) it is totally independent of any physical architecture of grid.

Kokilavani [18] has proposed a Load Balanced Min-Min algorithm which reduces the makespan and increases the resource utilization. This algorithm consists of two-phases. In the first phase, the Min-Min algorithm is executed and in the second phase, the tasks in the overloaded resources are rescheduled to use the unutilized resources.

Hao et al.[12] used a dynamic, distributed load balancing scheme for a grid environment which provides deadline control for tasks. Initially the resources check their state and make a request to the Grid Broker according to the change of state in load. Then, the Grid Broker assigns Gridlets between resources and scheduling for load balancing under the deadline request. The experimental results shows that this algorithm can reduce the makespan, improve the finished rate of the gridlet and reduce the submitted time.

Samuel et al [13] introduced a Augmenting Hierarchal Load Balancing algorithm .To evaluate the load of the cluster , probability of deviation of average system load from average load of cluster is calculated and checked for the confinement within a defined range of 0 to 1. The fittest resources are allocated to the jobs by comparing the expected computing power of the jobs with the average computing power of the clusters. The evaluation results show that the makespan is reduced and the idle time of the clusters is also reduced.

Balasangameshwara and Raju [14] were proposed a fault tolerant hybrid load balancing algorithm. This algorithm is carried out in two –phases. In the first phase, a static load balancing policy selects the desired effective sites to carry out the submitted job. If any of the sites is unable to complete the assigned job, a new site will be located using the dynamic load balancing policy. By this way the job failure is identified and load is redistributed to the underloaded resource. The experimental results show that this algorithm performs well in large grid environment.

Fathy [21] has proposed a two-level load balancing policy for the multi-cluster grid environment where each cluster is located in different local area networks. This algorithm distributes the load on the resources based on the processing elements capacity. This leads to minimization of the overall job mean response time and maximization of the system utilization and throughput.

Fernandes et al [8] were proposed a route load balancing algorithm, which is designed to equally distribute the load of tasks of parallel applications. This algorithm uses the message routing concepts from the computer networks to define the computer neighbourhood. If any resource is overloaded, then the neighbour's load is evaluated. If the neighbour node is not overloaded then the tasks are transferred to it. The task transfer is based on the migration model.

Cao et al [5] were designed a load balancing algorithm in combination with intelligent agents and multi-agent approaches. This algorithm assumes grid as two-level grid. In the local grid, each agent is responsible for scheduling and load balancing across multiple nodes in grid environment. In the global level grid, each agent is a representative of resources and acts as a service provider.

Due to the resource characteristics in the grid environment, the resource failure may frequently occur in any grid system. This affects the job execution fatally. Detecting the resource failure in advance in grid environment is a complex task. Once the resource failure occurs, the resubmission of the same task to another resource dynamically involves many problems such as selecting appropriate resource meeting the user deadline, increase in response time, etc. So providing reliability in the grid environment is another challenging task which affects the performance of overall grid environment.

Latchoumy and Sheik [16] discussed various fault tolerance techniques, fault management in different systems and related issues. Some of the techniques are check-point recovery, replication and so on.

Hwang and Kesselman [15] proposed a flexible failure handling framework, which addresses some failure recovery requirements. Dabrowski [17] pointed out some important issues and problems to overcome the resource failures in the grid environment.

3 Proposed System

In this section, the brief description of the structure of the grid environment and proposed algorithm is presented. The Grid environment consists of Processing Entities (PE), Machine, Resource and Resource Broker.

Grid Broker: The Grid Broker is the top manager in the grid environment. It is responsible for the overall activities like

- *Nanthiya D is currently pursuing masters degree program in Computer Science Engineering inKongu Engineering College,India. E-mai:nanthiyaratha@gmail.com*
- *Keerthika P is currently workingas Assistant Professor in Computer Science Engineering in Kongu Engineering College,India. E-mail:keerthikame@gmail.com*

scheduling and load balancing in the grid environment. This gets the workload information from the resources, which lies in next level to the Grid Broker. This sends the tasks to the Resources to optimize the load.

Resource: This is next to grid Broker in the hierarchy which is characterized in fig.1. It is responsible for maintaining the scheduling and load balancing of its machines. Also, it sends an event to grid broker if it is overloaded.

Machine: This is a Processing Entity (PE) manager. It is responsible for task scheduling and load balancing of its PEs. Also, it sends an event to resource if it is overloaded.

The proposed algorithm combines load balancing concept with the fault tolerance concept, which works at three levels: Broker, Resource and Machine. When a job is submitted to the Machine, the algorithm works at two phases namely:-Selection of fittest resource and load balancing algorithm

3.1 Selection of Fittest Resource

Each job is submitted to the Machine in the grid environment with the user deadline. Then for each available job, ECT (Expected Completion Time) value for each resource is found. Then the resources that are satisfying the user deadline (less than or equal to user deadline of the job) alone are considered.

$$ECT = EET + EAT + ECOT \quad (1)$$

Where *ECT* is the Expected Completion Time, *EET* is the Expected Execution Time, *EAT* is the Expected Availability Time and *ECOT* is the Expected Communication Time.

Now, the fittest value for each selected resource is calculated to find the fittest resource for each job. The fittest value for each job is calculated is based on the Fault Tolerance value and the Expected Completion Time of the job in that particular resource. The Fault tolerance value is defined as the ratio of number of submitted jobs failed in that node to the total number of jobs submitted. Finally, the resource with highest fittest value is considered to be the fittest resource for that job.

3.2 Load Balancing algorithm

The load balancing mechanism is performed at all the three levels. After the selection of resources in the first phase, the load balancing mechanism is performed. The load balancing algorithm classifies the nodes at three levels into three lists:-overloaded list, normally loaded list and underloaded list based on the load threshold values namely:-PE level threshold, Machine level threshold and Resource level threshold. Overloaded list means set of nodes with overcommitment of jobs. Normally loaded list is a list of nodes, if any job is further submitted to these nodes then they are shifted to the overloaded list. The underloaded list is a list of nodes which can be submitted with jobs.

If the selected node is present in the underloaded list, then the job is scheduled to the same resource else check for the presence of the next fittest resource in the underloaded list. Before scheduling the job, expected load of the selected resource should be calculated by adding the load of the job with current load of that particular resource. If the expected load exceeds the load threshold value then it is assumed that the submission of job may lead to the overloaded node condition. If the expected load value below the load threshold, then the job can be submitted. So, each time before the submission of jobs the expected load of the node is verified and submitted.

Finally if the load is unbalanced at the PElevel, then few jobs are selected and forwarded to the Machine level. If the load is unbalanced at the Machine level, then few jobs are selected and forwarded to the Resource level. Almost the load is balanced at the Machine level itself.

By this way the load balancing mechanism is performed at each level considering the fault tolerance factors

4 Performance Analysis

The proposed algorithm can be simulated on the GridSim Simulation toolkit. Because it has a complete set of features for simulating realistic grid testbeds. Such features are modeling heterogeneous computational resources of variable performance, differentiated network service, and workload trace-based simulation from a real supercomputer.

More importantly, GridSim allows the flexibility and extensibility to incorporate new components into its existing infrastructure. The performance of the proposed algorithm can be evaluated by the parameters such as Makespan, Communication overhead, Resource Utilization and Hit rate.

The proposed algorithm is simulated with 512 jobs and 16 resources for 4 different inputs. The Hierarchical load balancing with fault tolerance algorithm (HLBFT) is compared with Load balancing on Enhanced GridSim (LBEGS) in [3].

4.1 Makespan

The Makespan can be estimated by the following equation.

$$Makespan = \max_j \{ \sum cj + wj \} \quad (2)$$

where *cj* is the completion time for performing the task in the *jth* machine, *wj* is the previous workload of the machine j and $\sum cj + wj$ shows the time required for machine j to complete the tasks included in it. Fig. 1 it is inferred that the HLBFT has low makespan comparing with LBEGS.

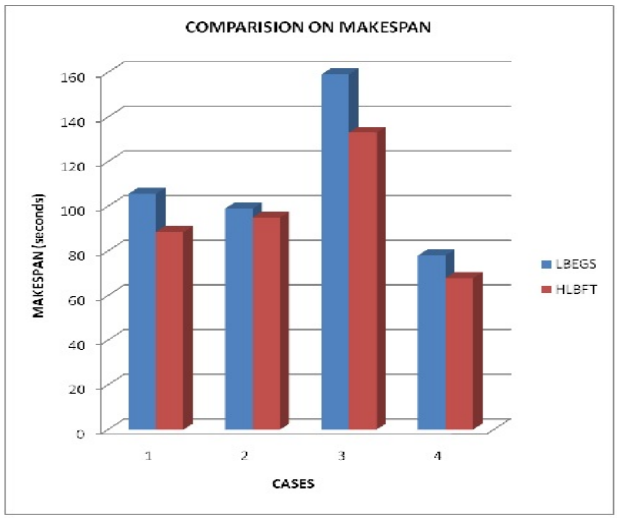

Fig. 1 .Graphical representation to show the improvement of HLBFT over LBEGS

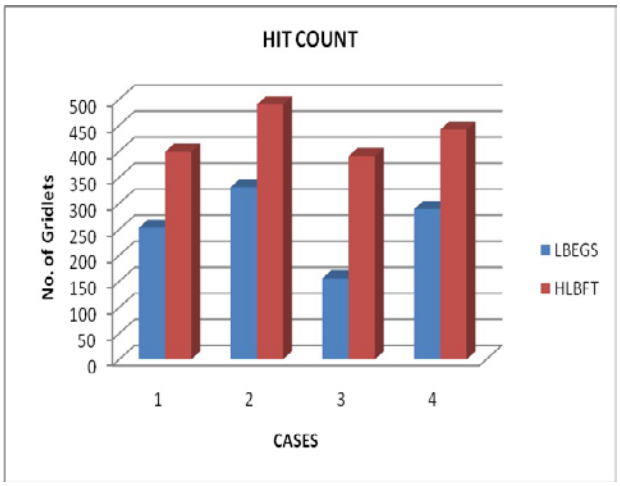

Fig. 3. Graphical representation to show the improvement of HLBFT over LBEGS.

4.2 Communication Overhead

The communication overheads are calculated based on the communication time.Fig.2 it is inferred that the HLBFT algorithm has low communication overhead.

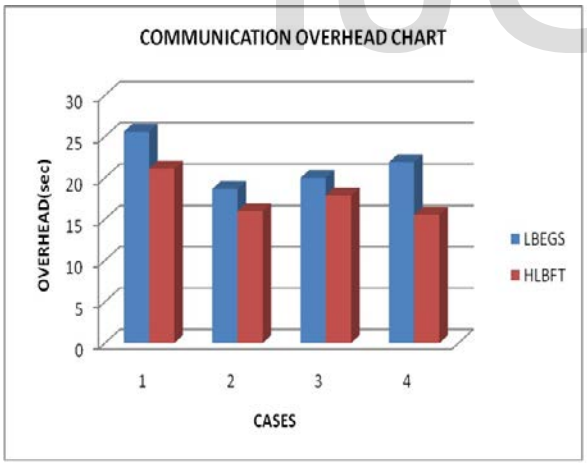

Fig. 2. Graphical representation to show the improvement of HLBFT over LBEGS

4.3 Hit Rate

The hit rate can be defined as the number of jobs that are successfully completed on the first schedule. From fig.5 and Table III it is inferred that due to fault tolerant factor in HLBFT the hit rate is very high comparing with LBEGS.

5 Conclusion

This paper addresses the issues of load balancing and resource failures. The proposed algorithm helps in reducing the makespan and achieving high reliability. The first phase of the algorithm selects the appropriate resource for the jobs and in second phase, the load of the entire system is balanced and the jobs are scheduled properly. So, the proposed algorithm may reduce the communication overhead and increases the performance of the entire system.

6 End Sections

6.1 References

[1] Mark Baker, Rajkumar Buyya and Domenico Laforenza, Grids and Grid technologies for wide-area distributed computing, Softw. Pract. Exper. 2002; (in press) (DOI: 10.1002/spe.488)

[2] Anthony Sulistio , Uros Cibej , Borut Robi, and Rajkumar Buyya , A Toolkit for Modelling and Simulation of Data Grids with Integration of Data Storage, Replication and Analysis, jan 2006

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

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

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

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

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

[8] 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.

[9] Yajun Li, Yuhang Yang, Maode Ma, Liang Zhou, A hybrid load balancing strategy of sequential tasks for grid computing environments, Future Generation Computer Systems (ISSN: 0167-739X) 25 (8) (2009) 819–828.

[10] Fernandes de Mello R, Senger LJ, Yang LT (2006) A routing load balancing policy for grid computing environments. In: Proceedings of the 20th international conference on advanced information networking and applications (Aina'06), vol 1, 18–20 April, 2006

[11] Krauter K, Buyya R, Maheswaran M (2002) A taxonomy and survey of grid resource management systems for distributed computing. Softw Pract Exp 32:135–164.

[12] Li Y, Lan Z (2005) A survey of load balancing in grid computing. In: Lecture notes in computer science, vol. 3314. Springer, Berlin, Heidelberg, pp 280–285.

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

[14] Yongsheng Haoa,, Guanfeng Liu , Na Wen(2012) An enhanced load balancing mechanism based on deadline control on GridSim Future Generation Computer Systems 28 (2012) 657–665.

[15] Joshua Samuel Raj, Hridya K. S and V. Vasudevan Augmenting Hierarchical Load Balancing with Intelligence in Grid Environment International Journal of Grid and Distributed Computing Vol. 5, No. 2, June, 2012.

[16] Jasma Balasangameshwara, Nedunchezhian Raju , A hybrid policy for fault tolerant load balancing in grid computing environments, journal of Network and Computer Applications35(2012)412–422 413.

[17] Joshua Samuel Raj, Hridya K. S and V. Vasudevan, Augmenting Hierarchical Load Balancing with Intelligence in Grid Environment, International Journal of Grid and Distributed Computing Vol. 5, No. 2, June, 2012.

[18] P. Latchoumy and P. Sheik Abdul Khader, Survey On Fault Tolerance In Grid Computing, International Journal of Computer Science & Engineering Survey (IJCSES) Vol.2, No.4, November 2011.

[19] Christopher Dabrowski(2009), Reliability in grid computing Systems, National Institute of Standards and Technology, 100 Bureau Drive, Stop 8970, Gaithersburg, MD 20899-8970, U.S.A.

[20] T. Kokilavani, Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing, International Journal of Computer Applications (0975 – 8887) Volume 20– No.2, April 2011.

[21] Klaus Krauter, Rajkumar Buyya and Muthucumaru Maheswaran, A taxonomy and survey of grid resource management systems for distributed computing, Softw. Pract. Exper. 2002; 32:135–164 (DOI: 10.1002/spe.432)

[22] Jim Blythe, Sonal Jain, Ewa Deelman, Yolanda Gil, Karan Vahi, (2007) Task Scheduling Strategies for Workflow-based Applications in Grids, Journal of Computer Science and Technology, 18, 2003, pp.442-451.