Implementation of Hierarchical Scheduling Algorithm on Real-Time Grid Environment

Abhishek P. Nachankar[1]
M.Tech-Second year, Department of Information Technology, Yeshwantrao Chavan College of Engineering, Nagpur, India.
abhishek_srpce@rediffmail.com

Prof. R. C. Dharmik[2]
Asst. Prof Department of Information Technology, Yeshwantrao Chavan College of Engineering, Nagpur, India.

Abstract— Grid computing is considered as the upcoming phase of distributed computing. Grid focuses on maximizing the resource utilization of an organization by sharing them across application. In grid computing, job scheduling is an important task. Load balancing and resource allocation are vital issues that must be considered in a grid computing environment. Load balancing is the technique which distributes the workload across multiple computers to reduce the latency of process execution with proper resource utilization. To resolve these issues, we are proposing hierarchical scheduling algorithm for grid environment and implementing it on the real time grid set up on LAN.

Keyword s- Hierarchical Set Up, Job Scheduling, Load Balancing, Resource Utilization, Process Execution.

I. INTRODUCTION

Grid computing environment combines together number of processing cycles of many computers in a network for solving problems that are too intensive for any super computer [11]. It is same as that of traditional networking unlike conventional networks that focus on communication among devices. It uses the unused CPU capacity in all participating machines to be allocated to single application that is very computation intensive and programmed for parallel processing. Grid aims on the virtualization of distributed computing and data resources such as processing, network bandwidth and storage capacity to create a single system image and granting users access to emerge IT capabilities. An internet/end user can view a unified instance of content via the internet; a grid user essentially sees a single, large virtual computer [1].

Grid environment suggests a solution for solving massive computational problems by making use of the idle resources of large numbers of distributed computers. Grid computing environment has unique ability to support computation across working domains apart from traditional computer clusters. Grid computing aims on solving problems that are too big for any standalone machine and versatile to work on multiple smaller problems. Hence grid provides a multiple user environment. Its secondary aims are better exploitation of available computing power for the complex processing computational exercises [2].

A grid is homogeneous or heterogeneous resources to deal with large scale complex problems. There are different issues in grid computing. Appropriate and efficient allocation of the tasks to the available resources is called as job scheduling. Aim of job scheduling is to provide proper load balancing at higher level of nodes and reduce the response time by proper allocation of jobs. In grid computing, resources and tasks are changing time to time. So, no traditional scheduling algorithm works properly for dynamic grid system. The vital thing is to allocate the tasks to proper resources by using good scheduling algorithm to enhance throughput and avoid unnecessary delays. Although many of proposed scheduling algorithms proved that they are suitable for a homogeneous environment, but there are only few algorithms that work for heterogeneous environment.

Load balancing is an important issue that is recently studied very much because of need of high performance computing. The computation of grid involves high performance platforms to heterogeneous and shared environment. Thus how to select a proper load balancing schemes for Grid environment becomes the area of research. In parallel applications, load balancing is achieved by distributing the load across multiple processors as evenly as possible by keeping objective of improving performance.

A load balancing scheme consisting of three phases: information collection, decision making and data migration. In information collection, load balancer gathers the information of workload distribution and the state of computing environment. In decision making, calculating an optimal data distribution, while during data migration, transfers the excess amount of jobs from overloaded processors to under loaded ones [3].

The paper is presented as follows: Section 2 focuses on related work; Section 3 discusses the

978-1-4799-6892-3/15 $31.00 © 2015 IEEE
DOI 10.1109/ICCUBEA.2015.117

proposed algorithm with results; Section 4 we are coming up with future scope.

II. LITRETURE SURVEY

Resource manager or Scheduler is central authority of the system and its main task is to collect requests from different users, match the user's requests to available resources and map it to best the matched resources. Such requests are nothing but the jobs for the Grid environment. A standard scheduling algorithm is needed for enhancing the system performance.

It is very tricky to handle load balancing in grid as that of traditional distributed systems because of the heterogeneous as well as dynamic nature of the grid. Major research has been done only on centralized systems. All of them are having disorders, such as scalability issues specifically for the centralized approaches. Author's considered a hierarchical tree structure for grid computing environment similar to that we are using [4].

The grid computing environment collects, integrates, and uses heterogeneous or homogeneous resources which are distributed in the network. There are two types of grid environment as: data grids and computing grids. In computing grid, importance is given to the job scheduling [5]. An impressive scheduling algorithm can allocate jobs to resources and balance the entire load of the system. A job scheduling algorithm called Hierarchical Load Balancing Algorithm for Grid computing environment is used to balance the entire system load.

When job is assigned to a cluster with the maximum Computing Power, selected cluster will be checked whether it is overloaded or not. If cluster's load is larger than balance threshold, the cluster is considered as overloaded and the job will seek out the next cluster with the highest ACP. This is called as local update. Local update reflects the situation changes at the local level. The computing power and the load of other clusters remain unaffected.

Load balancing technique in grid environment is used to improve the scalability of the entire system. Most work has been proposed on the area of process migration and load balancing theory [6]. A research has been done in the current work to focus on reliable load balancing approach which gives priority to the processes. This approach has established a Process Migration Server ensuring that the latency time of migrated process execution is minimised with no starvation for any process.

For task allocation in grid computing environment static and dynamic techniques are used [7]. Maximum work and research has been done on dynamic load balancing approach. Few of them are already come up with the results. Not a standard scheduling algorithm exists till now. Little work has been done on scheduling algorithm with multiple parameters [8]. However in this area hierarchical scheduling load balancing algorithm works successfully with decent results.

III. PROPOSED SYSTEM AND RESULTS

However, to satisfy end user expectations with improved performance and efficiency, the Grid computing environment require standard scheduling algorithms for task distribution. A proper scheduling algorithm tries to reduce the response time of end user's submitted jobs by maximizing the utilization of the available resources. Here we are focusing on hierarchical organizational structure. However, In hierarchal set up systems at the same level can communicate directly with the systems above or below them, or at a second level cluster node [7]. Many of the current Grid systems use such type of organizational structure because it is highly scalable. Hence main aim is to improve the performance of the environment by reducing process execution time. To check how efficient our proposed scheduling algorithm, different parameters are under monitoring [4], which are:-

Idle Nodes: Total no. of available nodes to allocate the job.

Overload Nodes: List of overloaded nodes in the complete set up.

IP Address: Unique identification of each node.

N: Total number of nodes.

J: No. of Jobs.

$\sum_{i=1}^{n} \boldsymbol{Exe\ Ti}$: Total time for execution of all jobs.

Through the grid interface the user will be able to make the request to the scheduler node. Scheduler node have status of each and every node with itself and keeps the information about each and every resource, such as CPU and memory utilization, less loaded and overloaded cluster nodes, CPU idle time, etc. The role of the scheduler is to accept the job from the end user and uses the scheduling algorithm with a good effect to choose the cluster with most available resources and compare its load against the whole grid environment. After that it selects the resources with the highest computing power to the cluster to execute the submitted end user request. Once the job gets done, the results and the updated status of the clusters will be sent back to the scheduler for future execution of the job.

3.1 HIERARCHICAL SET UP

In grid computing environment, a number of machines are connected to each other as workstations as shown in fig1. It is a hierarchical structure which is grouped together with the help of LAN. From the topology this system is divided into three levels: L1 – Grid or Scheduler Node, L2 – Master Level or Clustering Nodes, L3 – Computing Nodes to the clustering nodes. Each and every node will communicate amongst the hierarchy with the LAN connectivity.

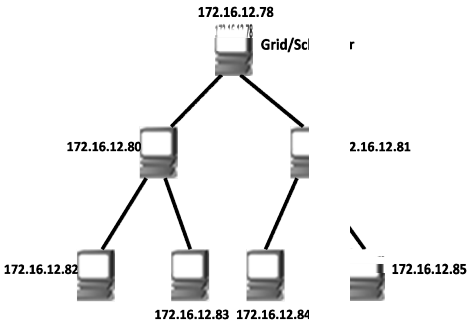

Fig1: Structured Hierarchy of Grid Environment

The Hierarchical structure shown in the fig1 is completely dependent on LAN as they are connected. A Hierarchy of 3-level or more can also be formed by joining many numbers of workers or computing nodes. This can be implemented in two possible ways. One is, to create the hierarchy logically on one machine by using Grid Simulator or any middleware [12]. Second is, to implement it physically on LAN by using as many nodes as possible. In this paper, we prefer the second option to implement our algorithm on physical real time hierarchy i.e. LAN. In the above discussed fig1 grid or scheduler node is called as primary load distributor node and is only static node throughout hierarchy and rest of cluster level nodes and worker nodes are created according to their availability. And the cluster nodes are originated from the scheduler node based on the requirement of distributing available resources form the available list and map the efficient nodes as second-level-cluster nodes. Another role of scheduling node is to distribute the request amongst the second-level-cluster node based on the parameters of total number of resources available on at that instance and total number of second-level cluster node. The equation of second level nodes is- total number of clustering nodes depends on the total number of nodes available in grid factory and total number of jobs available to distribute; this is a simple mechanism to make resources most available to the incoming request from the end user.

3.2 PROCESS EXECUTION CYCLE

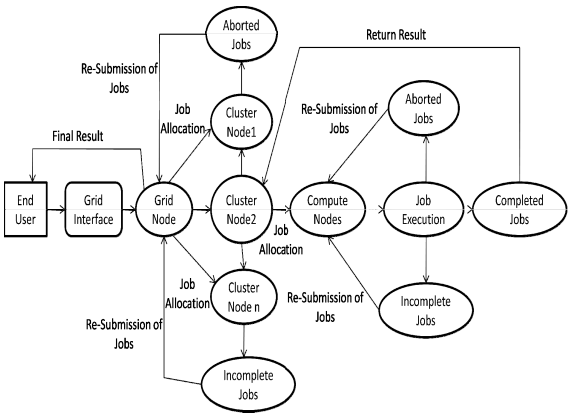

Fig2: Process Execution in Hierarchical Grid

Figure 2 explains the job execution cycle in a hierarchical grid environment and the sequence of steps shows that the life spans of hierarchical grid. In the figure, Grid node, Cluster node and its computing nodes showing their job execution cycle with their own working respectively.

Working of Middle Level Nodes:

At the time of job allocation, Grid node distributes the job to cluster nodes according to the hierarchical structure and total number of second level cluster nodes. With the available resources it allocates the job to the cluster nodes. At the same level the job is get divided and distributed to their worker node. This job distribution approach is based on multiple different parameters of efficient load distribution. The vital feature of distribution is based on the amount of available resources like CPU idleness, heap memory and total number of jobs on dynamic approach [4].

3.3 ALGORITHM

Step 1: Receive the request from end user.

Step 2: For all received request sort the servers in ascending order of their current load.

Step 3: Identify the server with the lowest load.

Step 4: Allocate the request to the server with the lowest load.

Step 5: For every request, allocate the load amongst all the servers in such a way that no server should be overloaded.

Step 6: Once request is served to the server its file contents are modified which represents the load.

Step 7: For every request served to the server its respective file contents are modified.

Step 8: Execution stops when user stop making request.

Step 9: At completion all the servers are equally loaded.

Main aim of the algorithm is to minimize process execution time and improve resource utilization by proper load balancing. Hence the implementation is divided in two phases. The first phase implemented for process execution time. In this, the request is submitted to the grid node by the end user through the grid interface. For this we created the url for the end user which is accessed on the remote desktop and a simple login application. Once this url is accessed login page appears as follows-

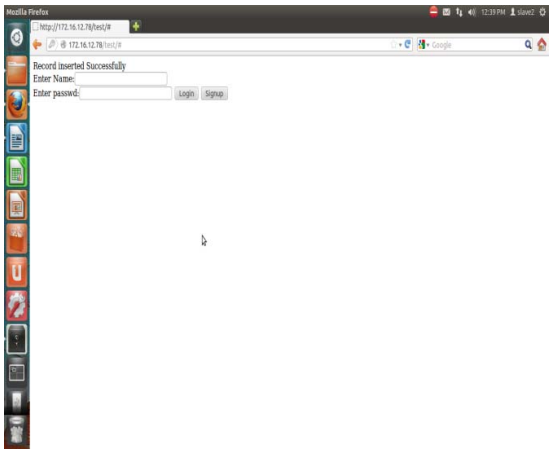

Fig3: Snapshot of Interface to the End user

This login page is accessed on a remote desktop by the end user. Through this user will do the login i.e. job submission to the grid node and based on the available resources it will allocate the task amongst the servers and do the proper load balancing. After job submission new page appears which shows that to which server load is distributed. Here load is given to that server which is less loaded amongst all. The records of all the servers are kept at scheduler node in the form of file. This server gives the information about the overall latency of grid environment. If user wants to make a request again to scheduler node then he should click on the link 'send request again' multiple times to see load distribution approach with the overall latency. This is the second phase of the algorithm i.e. load distribution. This window appears as follows-

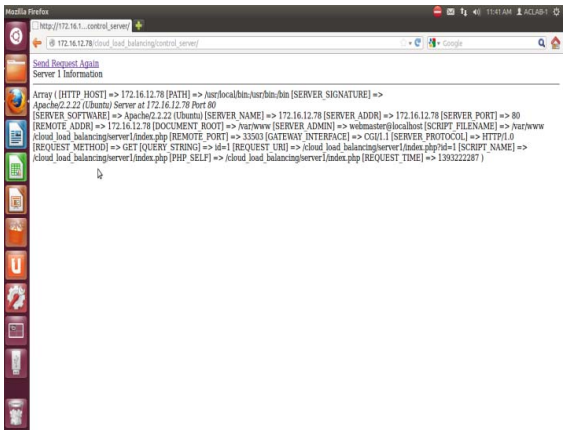

Fig4: Snapshot of Load balancing servers

This shows that server 1 is selected for task distribution as it has minimum of load amongst all servers. Once load is allotted to the server its file contents are updated at scheduler node.

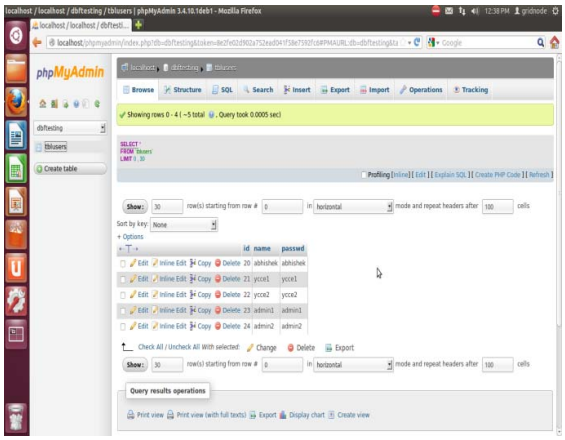

Fig5: Snapshot of Scheduler's Database

This is the scheduler's database that will be accessed through that login interface. Once the query executed successfully it will return the result. The result is shown at the scheduler's database that is displayed in above fig in the yellow patch and query took 0.0005sec to execute.

Few results are calculated by sending multiple requests to grid node which identifies the server to which task is distributed with overall latency and time taken to execute that query. As there are seven servers so initially first seven requests are distributed amongst seven servers and later from request 8 again it starts distributing request from server1. Once request is served, file contents of the respective servers at scheduler node are updated. The results shown below are calculated when all the resources are idle.

Request	Load Distributed to	Overall Latency in (nano seconds)	Execution Time in (seconds)
Request 1	Server 1	1393222287	0.0003
Request 2	Server 2	1393222316	0.0005
Request 3	Server 3	1393222494	0.0002
Request 4	Server 4	1393202511	0.0005
Request 5	Server 5	1393247399	0.0006
Request 6	Server 6	1393202553	0.0005
Request 7	Server 7	1393202555	0.0004
Request 8	Server 1	1402220525	0.0005

The complete server set up is based on the LAMP server i.e. Linux Apache MySql PHP and the platform for implementation is Ubuntu 12.04.

IV. CONCLUSION

In this paper, we implemented hierarchical scheduling algorithm which based on two parameters i.e. process execution time and resource utilization for grid environment which allocates the job amongst the hierarchy through scheduler node. The complete implementation is based on real time set up. From the above results we conclude that we implemented scheduling algorithm successfully with proper load balancing mechanism and minimal process execution time.

VII. REFERENCES

1. M. Irving, G. Taylor, Peter Hobson, "Plug in to Grid Computing," IEEE power and energy magazine, 2004.

2. N. Malarvizhi and V.R.Uthariaraj, Hierarchical Load Balancing Scheme for Computational Intensive Jobs in Grid Computing Environment, *ICAC, IEEE*, 2009.

3. J. Watts, S. Taylor, A Practical Approach to Dynamic Load Balancing, *IEEE transactions on parallel and distributed systems,* 1998, *vol. 9, no. 3, pp. 235-248.*

4. B.Pradhan, A. Nayak and D.S. Roy, An Elegant Load Balancing Scheme in Grid Computing Using Grid Gain, *International Journal of Computer Science and Its Applications,* 2011, *Vol. 1, Issue 1, pp.254-257.*

5. Yun-han Lee, Seiven Len and Ruay-Shiung Chang, Improving job scheduling algorithms in a grid environment, *Future Generation computer systems,* 2011*, vol.27, pp.991-998*

6. Leyli Mohammad Khanil, Shiva Razzaghzadehb and Sadegh vahabzadeh Zargaric, A new step towards load balancing based on competency rank and transitional phases in grid networks*, Future Generation Computer systems,* 2012, *vol.28.pp.682-688.*

7. E. Ajaltouni, A. Boukerche and M. Zhang, An Efficient Dynamic Load Balancing Scheme for Distributed Simulations on a Grid Infrastructure, *12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications,* 2008, *pp. 61-68.*

8. A.Touzene, S. Al-Yahai, Hussien, AlMuqbali, A. Bouabdallah and Y. Challal., Performance Evaluation of Load Balancing in Hierarchical Architecture for Grid Computing Service Middleware, *International Journal of Computer Sciences,* 2011, *pp. 1694-1702.*

9. K. Hemant Kumar Reddy and Shina Diptendu Roy, A Hierarchical Load Balancing Algorithm for Efficient Job Scheduling in a Computational Grid Test bed, 1st Int'l Conf. on Recent Advances in Information Technology, RAIT-2012.

10. R. Manimala and P.Suresh, Load Balanced Job Scheduling Approach for Grid Environment, IEEE, 2012.

11. http://www.wolfram.com/gridmathematica

12. Grid Gain: www.gridgain.com