A Dynamic Load Balancing Approach in Grid Environment

Er. Sourabh Budhiraja
M-Tech (Pursuing)

Abstract

The popularity of the Internet and the availability of powerful computers and high-speed networks as low-cost commodity components are changing the way we use computers today. These technical opportunities have led to the possibility of using geographically distributed and multi-owner resources to solve large-scale problems in science, engineering, and commerce. Recent research on these topics has led to the emergence of a new paradigm known as Grid computing.

To achieve the promising potentials of tremendous distributed resources, effective and efficient load balancing algorithms are fundamentally important. Unfortunately, load balancing algorithms in traditional parallel and distributed systems, which usually run on homogeneous and dedicated resources, cannot work well in the new circumstances. In this dissertation, the state of current research on load balancing algorithms for the new generation of computational environments will be surveyed and a new method for A Dynamic load balancing approach in grid Environment is proposed.

1. Intoduction

1.1 Grid Computing

The term grid is increasingly appearing in computer literature, generally referring to some form of system framework into which hardware or software components can be plugged, and which permits easy configuration and creation of new functionality from existing components. Grids enable the sharing, selection, and aggregation of a wide variety of resources including supercomputers, storage systems, data sources, and specialized devices that are geographically distributed and owned by different organizations for solving large-scale computational and data intensive problems in science, engineering, and commerce.

The term grid is chosen as an analogy to the electric power grid that provides consistent, pervasive, dependable, transparent access to electricity, irrespective of its source. Such an approach to network computing is known by several names: meta computing, scalable computing, global computing, Internet computing, and more recently Peer-to-Peer computing.

The concept of grid computing [1] started as a project to link geographically dispersed supercomputers, but now it has grown far beyond its original intent. The grid infrastructure [2] can benefit many applications, including collaborative engineering, data exploration, high throughput computing, distributed supercomputing, and service-oriented computing.

1.2 Load Balancing

The availability of low cost powerful computers coupled with the popularity of the Internet and high-speed networks have led the computing environment to be mapped from classical distributed to grid environments [3]. To improve the global throughput of these environments, effective and efficient load balancing algorithms are fundamentally important. Emerging as new distributed computing environments, computational grids [4] provide an opportunity to share a large number of resources among different organizations. Performance enhancement is one of the most important issues in such grid systems. One obvious way of achieving this goal is to add more computing nodes to the grid. However, in many situations, poor performance is due to uneven load distribution among the nodes in the system. Therefore, to fully exploit the computing power of such grid systems, it is crucial to employ a judicious load balancing strategy for proper allocation and sequencing of tasks on the computing nodes. Load balancing algorithms in classical distributed systems, which usually run on homogeneous and dedicated resources, cannot work well in the Grid architectures. Grids have a lot of specific characteristics [5], like heterogeneity, autonomy and dynamicity, which remain obstacles for applications to harness conventional load balancing algorithms directly. Load balancing is a mapping strategy that efficiently equilibrates the task load into multiple computational resources in the network based on the system status to improve performance [3].

There are many reasons a company would consider upgrading to a load balanced solution, but the most common reasons are scalability, high availability, manageability, ease of use and predictability. The essential objective of a load balancing can be [6], depending on the user or the system administrator, defined by:

- The aim for the user is to minimize the make spans of its own application, regardless the performance of other applications in the system.
- The main goal for administrator is to maximize meet the tasks deadline by ensuring maximal utilization of available resources.

2. The Dynamic Load Balancing Algorithm

The computational grid comprises of GIS, users and resources.Each resource is a computational unit with different processing power.All resources and users register their information to Grid Information Server.

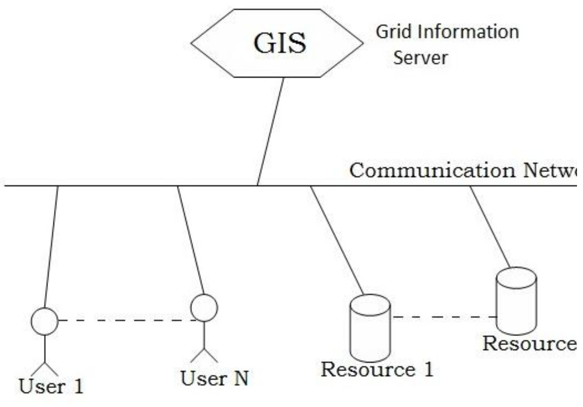

Figure 1: Computational grid model

The proposed algorithm for A Dynamic load balancing approach in grid Environment is as follows:

1. Input the value of number of Users (NU) and Resources (NR).
2. Initialize the gridsim toolkit.
3. Create the gridlength of each User.
4. Get the availability of all registered Resources.
5. Initialize the NextUser = 0 and QueueLength of each resource = 0.
6. Find the resource ‘ **R** ’ with minimum QueueLength.
7. Allocate the job of **NextUser** to **R** and **SubmissionTime_job** = GridSim.Clock( ).
8. **NextUser** = NextUser + 1.
9. **QueueLength_R**=QueueLength_R + 1.
10. Check for the arrival of any job **‘ J ’** from Resource **R’** after completing the execution.
11. If no resource arrival exist then goto step 14.
12. **QueueLength_R’**=QueueLength_R’ – 1.
13. **ExecutionTime_J**=GridSim.Clock ( ) - SubmissionTime_J.
14. If **NextUser <= NU** then goto step 6.
15. Print **ExecutionTime_J** of all jobs.

3. Sequence Diagram

Sequence Diagram showing working of dynamic load balancing algorithm is explained as:

- Firstly, all resources register (REG) their information to GIS.
- Then all users send request (REQ) to GIS for Resource characteristics.
- After that GIS sends resource characteristics (RC) to each user.
- Now each user starts determining queue length (DQL) of each resource and then send gridlet to resource having minimum queue length (QL).
- After gridlet execution is over, it is send back to user sending that gridlet.
- When all gridlets execution is over then each user print execution time (ET) of gridlets.

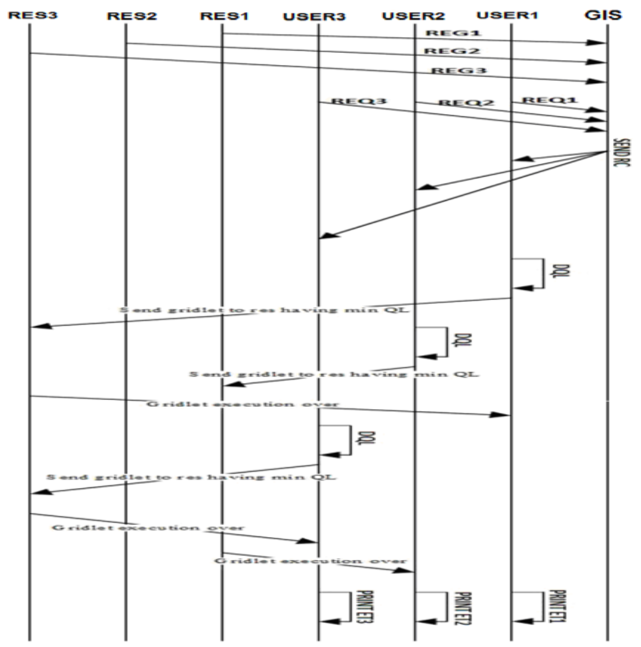

Figure 2: Sequence diagram showing working of proposed algorithm

4. Simulation Results

Following parameters are used during simulation of Dynamic load balancing algorithm:

Table 1: Simulation parameters

Simulation Runs	**4**
No. of Resources	**5 – 20**
No. of Users	**10 – 75**
No. of Jobs	**10 – 75**
Gridlet Size (In MI)	**10,000,000 - 750,000,000**
Processing Power of Resources (In MIPS)	**200 – 400**

The execution time of jobs corresponding to different users using ALBA (Adaptive Load Balancing Algorithm) and NALBA (Non Adaptive Load Balancing Algorithm) is shown in Figure 3 and Figure 4. The graph shows the execution time of jobs under NALBA is more than that of execution time of jobs with ALBA.

Table 2: Execution times of various users when number of resources = 3

Number of Resources = 3		
Number of Users	**Execution Time (ALBA)**	**Execution Time (NALBA)**
10	**2260186**	**2780202**
25	**10610471**	**14210527**
50	**37860946**	**44021172**
75	**81751425**	**87672115**

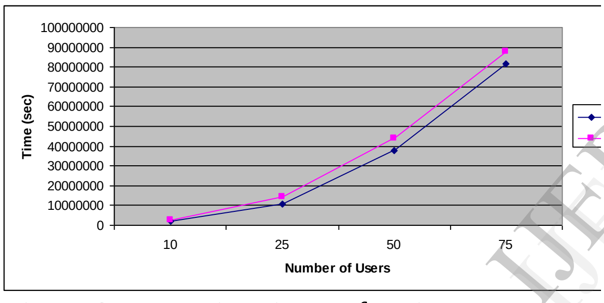

Figure 3: Execution times of various users when number of resources = 3

Table 3: Execution times of various users when number of resources = 5

Number of Resources = 5		
Number of Users	**Execution Time (ALBA)**	**Execution Time (NALBA)**
10	**1700336**	**2140365**
25	**7250840**	**8010944**
50	**24501682**	**29461994**
75	**51752522**	**56793571**

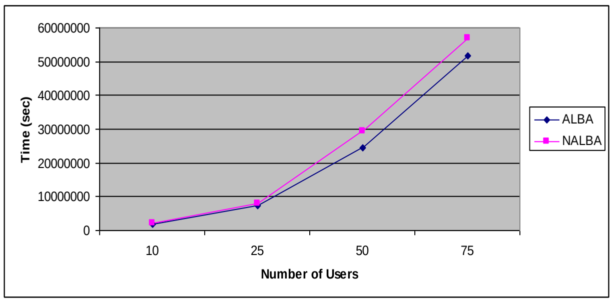

Figure 4: Execution times of various users when number of resources = 5

The execution time of jobs corresponding to different resources using ALBA and NALBA is shown in Figure 5 and Figure 6. The graph shows that execution time of jobs under ALBA is still less as compared to NALBA even when number of resources are increased due to selection of only those resources which has minimum load.

Table 4: Execution times of various resources when number of users = 25

Number of Users = 25		
Number of Resources	**Execution Time (ALBA)**	**Execution Time (NALBA)**
5	**7250840**	**8011072**
10	**4851694**	**5691870**
15	**4052547**	**4892494**
20	**3653400**	**4573562**

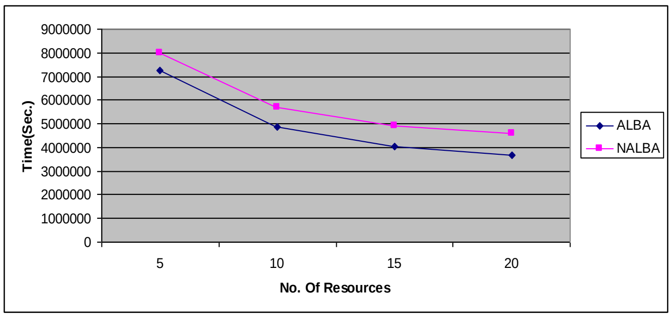

Figure 5: Execution times of various resources when number of users = 25

Table 5: Execution times of various resources when number of users = 50

Number of Users = 50		
Number of Resources	**Execution Time (ALBA)**	**Execution Time (NALBA)**
5	**24501682**	**29781949**
10	**14503492**	**18823936**
15	**11305201**	**15865760**
20	**9706906**	**14187526**

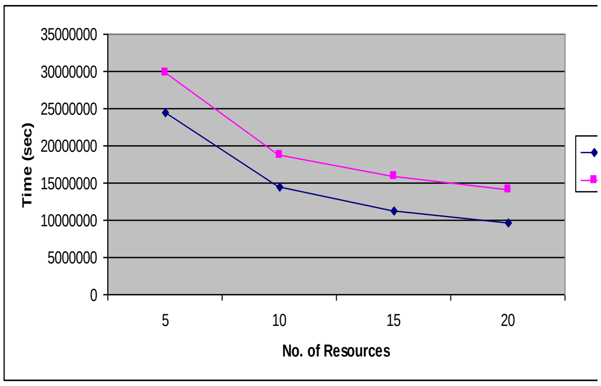

Figure 6: Execution times of various resources when number of users = 50

The results show that ALBA is better than the NALBA in all scenarios.

5. CONCLUSION

In this, effect of load balancing on job in terms of execution time is analyzed. Results show that the execution time of ALBA is less as every time the resource with minimum queue length is selected for execution as compared to the execution time with NALBA. The algorithm is tested under various load conditions in terms of job length varying from 10,000,000 - 750,000,000 (MI). The performance of ALBA is also better when the system is lightly loaded in terms of increasing the resources and keeping the number of user as fixed.

References

[1] Yulai Yuan, Yongwei Wu, Guangwen Yang, and Weimin Zheng," Adaptive Hybrid Model for Long Term Load Prediction in Computational Grid," 8th IEEE International Symposium on Cluster Computing and the Grid, pp.340-347, August 2008.

[2] Youchan Zhu, Lei An, Shuangxi Liu," A Resource Discovery Method of Grid Based on Resource Classification," Proceedings of First International Conference on Intelligent Networks and Intelligent Systems, pp. 716-719, August 2008.

[3] C. Xu and F. Lau, "Load Balancing in Parallel Computers: Theory and Practice," Kluwer, Boston, MA, 1997.

[4] Yajun Li, Yuhang Yang, and Rongbo Zhu," A Hybrid Load Balancing Strategy of Sequential Tasks for Computational Grids," International Conference on Networking and Digital Society (ICNDS), 2009.

[5] M. Baker, R. Buyya, and D. Laforenza, "Grids and grid technologies for wide area distributed computing," International Journal of Software: Practice and Experience (SPE), vol. 32(15), 2002.

[6] B. Yagoubi, and M. Medebber, "A load balancing model for grid environment," Proceeding of 22nd

International Symposium on Computer and Information Sciences (ISCISC 2007), pp. 1-7, 7 November 2007.

[7] C. Kim and H. Kameda," An algorithm for optimal static load balancing in distributed computer systems," IEEE Transaction on Computers, vol. 41(3), pp. 381-384, March 1992.

[8] K. Lu, R. Subrata, and A. Zomaya, "An Efficient Load Balancing Algorithm for Heterogeneous Grid Systems Considering Desirability of Grid Sites," Journal of Computer and System Sciences, vol. 73(8), pp. 1191-1206, December 2006.

[9] K. Lu, and A. Zomaya, "A Hybrid Policy for Job Scheduling and Load Balancing in Heterogeneous Computational Grids," Proceeding of 6th International Symposium on Parallel and Distributed Computing, pp. 19-26, 5 July 2007.