THE IMPACTS OF LOAD BALANCING ON GRID COMPUTING PERFORMANCE - A REVIEW

Amit Kapoor*

Meenakshi Gupta*

Vasudha Rani*

ABSTRACT

Grid technology has emerged as a new way of large-scale distributed computing with high-performance orientation. 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. Grid computing is the next generation IT infrastructure that promises to transform the way organizations and individuals compute, communicate and collaborate. The Objective of the grid environment is to achieve high performance computing by optimal usage of geographically distributed and heterogeneous resources.

But grid application performance remains a challenge in dynamic grid environment. Resources can be submitted to Grid and can be withdrawn from Grid at any moment. This characteristic of Grid makes Load Balancing one of the critical features of Grid infrastructure. There are a number of factors, which can affect the grid application performance like load balancing, heterogeneity of resources and resource sharing in the Grid environment. In this paper we have focused on Load Balancing and tried to present the impacts of Load Balancing on grid application performance.

Keywords: *Grid computing, Load balancing, Job migration*

*Assistant Professor, Maharaja Agrasen Institute of Management and Technology, Jagadhri, Haryana, India

INTRODUCTION

Grid computing is a model of distributed computing that uses geographically and administratively disparate resources [3]. In Grid computing, individual users can access computers and data, transparently, without having to consider location, operating system, account administration, and other details. In Grid computing, the details are abstracted, and the resources are virtualized.

Then the Grid computing comes into existence. Grid computing is the next generation IT infrastructure that promises to transform the way organizations and individuals compute, communicate and collaborate. Sharing in a Grid is not just a simple sharing of files but of hardware, software, data, and other resources [4]. Thus a complex yet secure sharing is at the heart of the Grid. The goal of Grid computing is to create the illusion of a simple but large and powerful self-managing virtual computer out of a large collection of connected heterogeneous systems sharing various combinations of resources.

1. THE GRID

Rajkumar Buyya [1] defined the Grid as: "Grid is type of parallel and distributed system that enables the sharing, selection and aggregation of geographically distributed resources dynamically at run time depending on their availability, capability, performance, cost, user quality-of-self-service requirement"[5].

In other words we can say

• Grid is a service for sharing computer power and data storage capacity over the Internet and Intranet.

• Grid is beyond simple communication between computers but it aims ultimately to turn the global network of computer into one vast computational resource.

• Grid is to coordinate resources those are not subject to centralized control.

• Grid is to use standard, open, general-purpose protocols and interfaces.

• Grid is to deliver nontrivial Qualities of Service.

A computational Grid environment behaves like a virtual organization consisting of distributed resources. A **Virtual Organization** is a set of individuals and institutions defined by a definite set of sharing rules like what is shared, who is allowed to share, and the conditions under which the sharing takes place. A number of VOs exist like the application service providers, storage service providers, etc., but they do not completely satisfy the requirements of the Grid. The needs of the Grid range from client-server to peer-to-peer architecture, from single user to multi-user systems, from sharing files to sharing resources,

etc. and all these in a dynamic, controlled, and secured manner. Current distributed computing technologies do not fulfill all these needs. As Grid computing focuses on dynamic and cross-organizational sharing, it enhances the existing distributed computing technologies. Many of the existing technologies like the World Wide Web (WWW), Internet and Peer-to-Peer computing can be considered as similar to Grid Computing, but differences do exist. Internet and Peer-to-Peer computing have much in common with Grid technologies.

2.1 CHARACTERISTICS OF GRID

There are three main issues that characterize Computational Grids [6, 7]:

• ***Heterogeneity***: A Grid involves a multiplicity of resources that are heterogeneous in nature and might span numerous administrative domains across wide geographical distances.

• ***Scalability***: A Grid might grow from few resources to millions. This raises the problem of potential performance degradation as a Grids size increases. Consequently, applications that require a large number of geographically located resources must be designed to be extremely latency tolerant.

• ***Dynamicity or Adaptability***: In a Grid, a resource failure is the rule, not the exception [8]. In fact, with so many resources in a Grid, the probability of some resource failing is naturally high. The resource managers or applications must tailor their behavior dynamically so as to extract the maximum performance from the available resources and services.

• ***Parallel CPU execution***: One of most important feature of Grid is its scope for massive parallel CPU capacity. The common attribute among such uses is that the applications have been written to use algorithms that can be partitioned into independently running parts. A CPU intensive Grid application can be thought of as many smaller “Subjobs,” each executing on a different machine in the Grid.[8] To the extent that these Subjobs do not need to communicate with each other, the more “scalable” the application becomes. A perfectly scalable application will, for example, finish 10 times faster if it uses 10 times the number of processors.

• ***Virtual organizations***: The users of the Grid can be organized dynamically into a number of virtual organizations, each with different policy requirements [9, 10]. These virtual organizations can share their resources collectively as a larger Grid.

• ***Resource balancing***: A Grid contains a large number of resources contributed by individual machines into a greater total virtual resource. For applications that are Grid enabled, the Grid can offer a resource balancing effect by scheduling Grid jobs on machines with low utilization [11].

• ***Reliability and Management***: High-end conventional computing systems use expensive hardware to increase reliability. A Grid is an alternate approach to reliability that relies more on software technology than expensive hardware [8].

3. LOAD BALANCING IN GRID ENVIRONMENT

Grids functionally combine globally distributed computers and information systems for creating a universal source of computing power and information [2]. A key characteristic of Grids is that resources (e.g., CPU cycles and network capacities) are shared among numerous applications, and therefore, the amount of resources available to any given application highly fluctuates over time. In this scenario load balancing plays key role. For applications that are Grid enabled, the Grid can offer a resource balancing effect by scheduling Grid jobs on machines with low utilization. A proper scheduling and efficient load balancing across the Grid can lead to improved overall system performance and a lower turn-around time for individual jobs.

Load balancing is a technique to enhance resources, utilizing parallelism, exploiting throughput improvisation, and to cut response time through an appropriate distribution of the application [28]. To minimize the decision time is one of the objectives for load balancing which has yet not been achieved.

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.

Classical load balancing algorithms are typically based on a *load index* which provides a measure of the workload at a node relative to some global average, and four policies which govern the actions taken once a load imbalance is detected [29]. The load index is used to detect a load imbalance state. Qualitatively, a load imbalance occurs when the load index at one node is much higher (or lower) than the load index on the other nodes. The length of the CPU queue has been shown to provide a good load index on timeshared workstations when the performance measure of interest is the average response time [24, 27]. In the case of multiple resources (disk, memory, etc.), a linear combination of the length of all the resource queues provided an improved measure, as job execution time may be driven by more than

CPU cycles. The four policies that govern the action of a load balancing algorithm when a load imbalance is detected deal with information, transfer, location, and selection.

The information policy is responsible for keeping up-to-date load information about each node in the system. A global information policy provides access to the load index of every node, at the cost of additional communication for maintaining accurate information [23].

The transfer policy deals with the dynamic aspects of a system. It uses the nodes' load information to decide when a node becomes eligible to act as a sender (transfer a job to another node) or as a receiver (retrieve a job from another node). Transfer policies are typically threshold based. Thus, if the load at a node increases beyond a threshold Ts, the node becomes an eligible sender. Likewise, if the load at a node drops below a threshold Tr, the node becomes an eligible receiver. Load balancing algorithms which focus on the transfer policy are described in [24, 29].

The location policy selects a partner node for a job transfer transaction. If the node is an eligible sender, the location policy seeks out a receiver node to receive the job selected by the selection policy (described below). If the node is an eligible receiver, the location policy looks for an eligible sender node. Load balancing approaches which focus on the use of the location policy are described in [25, 26].

Once a node becomes an eligible sender, a **selection policy** is used to pick which of the queued jobs is to be transferred to the receiver node. The selection policy uses several criteria to evaluate the queued jobs. Its goal is to select a job that reduces the local load, incurs as little cost as possible in the transfer, and has good affinity to the node to which it is transferred. A common selection policy is latest-job arrived which selects the job which is currently in last place in the work queue.

3.1COMPARATIVE ANALYSIS OF EXISTING LOAD BALANCING ALGORITHMS

In this section comparative study of load balancing algorithms (already mentioned in literature survey) has been done on the basis of some parameters like, scalability, security, throughput, migration time, efficiency, fault tolerance, overload rejection, resource utilization, forecasting accuracy and stability etc.

- **Node reconfiguration mechanism** makes application workload migrate from source node to destination node, and then let source node depart from original computing environment. There are two implementation fashions of node reconfiguration, first one is synchronous

method and the other is asynchronous method. Best performance can be obtained by maintaining the order of node reconfiguration messages [16].

- In **Virtual machine migration** snapshots of machine are sent to other machine by using two migration method, live migration and regular migration. An important aspect of this mechanism is to make the run-time job migration with non-dedicated shared resources in dynamic grid environment and also provides high isolation, and security [16].
- **Robin Hood algorithm** is totally non-centralized mechanism, using the proactive library for the migration of jobs, and a multicast channel for node coordination. It improves the decision time in non-centralized environment and uses the noncentralized architecture and non- broadcasting of the balance of each node to reduce the overload in network [15].
- **Load based graph method** is based on network graph where each node is represented with its load, whereas load can be the number of users, average queue length or the memory utilization. It uses analytic model and single load determination policy throughout the system and load is determined on the basis of memory utilization and average queue length. The major parameter, network partitioning issues along with inter-cluster and intra-cluster transfers for decision making of load balancing for the transfer is considered here[17].
- In **Data Consolidation** algorithms data replication technique is considered here which provide fast access and reliability by reducing the scheduling of task and data management. This is the main optimization technique; provide fast data access and reliability. Data consolidation (DC) is the combine name of task scheduling and data migration. DC can be considered as the dual of the data replication problem, where from one site the data are scattered to many repository sites. Through this dual relationship we will show that DC can also provide, by reversing the procedure, data replication services [18]. Table-1 shows the comparative study of all the existing algorithms described above:

Para-meters / Algorithm	**Effic ienc y**	**Over load Rejec t-ion**	**Scala bility**	**Throu-ghput**	**Reso urce Utili za-tion**	**Fore-casting Accu-racy**	**Fault Tole-rance**	**Securit y**	**Per-formance**	**Stabil ity**
Node Reconfigurat ion by User Level Thread	Med-ium	No	High	Low	Less	Less	Medium	Not good	High Perfor-mance in intra cluster	Less

Migration									scenario	
Virtual Machine Migration (live Migration)	Good	No	High	High for small jobs	Less	Medium	High	High security	High perfor-mance in inter cluster scenario	Less
Robin Hood: An Active Objects Load Balancing Mechanism	Not good	No	Not good	Medium	Less	More	Less	High security	Based on percentage loading at node	Large
Load Graph Based Transfer Method	Good	Yes	Med-ium	low	High	More	Medium	No security	High	Large
Data Consoli-dation	Medi um	No	High	Medium	High	More	High	No security	High due to data replication	Large

Table 1: Comparative study of existing migration algorithms

Load balancing algorithms can also be categorized on the bases of adaptability i.e. Static or Dynamic load balancing algorithms. [20]. Static load balancing algorithms are Round Robin algorithm, Randomized algorithm, Central Manager algorithm, and Threshold algorithm. Dynamic load balancing algorithms are Central Queue algorithm, and Local Queue algorithm. In the following table we have compared different load balancing algorithms to analyze their performance w.r.t. different parameters.

Para-meters / Algorithm	**Over load Rejec tion**	**Fault Tole-rance**	**Fore-casting Accu-racy**	**Stabil ity**	**Central ized / Decentr alized**	**Dynami c / Static**	**Cooper ative**	**Process Migration**	**Resource Utilization**
Round Robin	No	No	More	Large	D	S	No	No	Less
Random	No	No	More	Large	D	S	No	No	Less
Local Queue	Yes	Yes	Less	Small	D	Dy.	Yes	Yes	More

Central Queue	Yes	Yes	Less	Small	C	Dy.	Yes	No	Less
Central Manager	No	Yes	More	Large	C	S	Yes	No	Less
Threshold	No	No	More	Large	D	S	Yes	No	Less

Table 2: Parametric Comparison of Load Balancing Algorithms

Round Robin algorithm [19] distributes jobs evenly to all slave processors. All jobs are assigned to slave processors based on Round Robin order, meaning that processor choosing is performed in series and will be back to the first processor if the last processor has been reached. Processors choosing are performed locally on each processor, independent of allocations of other processors. Advantage of Round Robin algorithm is that it does not require inter-process communication. In general Round Robin is not expected to achieve good performance in general case.

Randomized algorithm [19] uses random numbers to choose slave processors. The slave processors are chosen randomly following random numbers generated based on a statistic distribution. Randomized algorithm can attain the best performance among all load balancing algorithms for particular special purpose applications.

Central Manager algorithm [20], in each step, central processor will choose a slave processor to be assigned a job. The chosen slave processor is the processor having the least load. The central processor is able to gather all slave processors load information, thereof the choosing based on this algorithm are possible to be performed. The load manager makes load balancing decisions based on the system load information, allowing the best decision when of the process created. High degree of inter-process communication could make the bottleneck state.

In **Threshold algorithm** [20], the processes are assigned immediately upon creation to hosts. Hosts for new processes are selected locally without sending remote messages. Each processor keeps a private copy of the system's load. The load of a processor can characterize by one of the three levels: underloaded, medium and overloaded. Two threshold parameters t_under and t_upper can be used to describe these levels. Under loaded: load < t_under , Medium : t_under ≤ load ≤ t_upper , and Overloaded: load > t_upper. Initially, all the processors are considered to be under loaded. When the load state of a processor exceeds a load level limit, then it sends messages regarding the new load state to all remote processors, regularly updating them as to the actual load state of the entire system.

In **Central Queue algorithm** [22] the decision of allocating a job to a processing node will be made on the spot during execution. However, the processing nodes will share a common central queue due to this the task allocated to one node can not rescheduled to another processing node.

The basic idea of load balancing through **Local Queue algorithm** [22] is that the local scheduler is more effective in utilizing the resources of the local server, if the total relative resource requirements of all jobs in a local work queue match the relative capacities of the server. Hence the resource utilization is much higher than in any other load balancing policy.

In static load balancing, the performance of the processors is determined at the beginning of execution. Then depending upon their performance the work load is distributed in the start by the master processor [21]. The slave processors calculate their allocated work and submit their result to the master. A task is always executed on the processor to which it is assigned that is static load balancing methods are non-preemptive. The goal of static load balancing method is to reduce the overall execution time of a concurrent program while minimizing the communication delays. A general disadvantage of all static schemes is that the final selection of a host for process allocation is made when the process is created and cannot be changed during process execution to make changes in the system load. Whereas in dynamic algorithms workload is distributed, assigned and even migrated on the bases of decision taken by the server node at run time. The main advantage of dynamic load balancing algorithms is their adaptable behavior in the grid environment and improved fault tolerance. Further the Dynamic load balancing algorithms have the ability to reject the load in case of heavily loaded node. Table-2 shows that in resource utilization is maximum in case of local Queue algorithm. A general disadvantage of dynamic algorithms is that they are less scalable than the static load balancing algorithms. Further in case of central queue algorithm the processes can not be migrated from the one node to another as individual nodes do not have their internal queues. Moreover the job migration characteristic of dynamic load balancing algorithms will ultimately increase the communication cost.

4. CONCLUSION AND FUTURE SCOPE

On the bases of comparison made in section 3 we can conclude that to enhance resources, utilizing parallelism, improve throughput, and to cut response time through an appropriate distribution of the applications is a great challenge in dynamic and heterogeneous environment like Grids. From the analysis of Table-1 and Table-2 we can conclude that the static dynamic load balancing policies are fairly simple as they have very less implementation

issues. Besides, they are less preferable as they do not possess those features which are most desirable for improving the performance of Grid Computing. On other hand, the dynamic load balancing policies are more preferable due to their adaptability in the Grid. Further some dynamic algorithms do not have process migration feature which is again a very important characteristics for the load migration in Grid environment. From the above analysis we concluded that there is no single algorithm which possesses all feature necessary for performance improvement in Grid. Hence there is further scope to find such algorithm.

REFERENCES

1. Rajkumar Buyya , Grid Computing and Distributed Systems (GRIDS) Lab., Department of Computer Science and Software Engineering, University of Melbourne, Australia and Manzur Murshed ,Gippsland School of comp and IT, Monash University, Gippsland Campus , GridSim: a toolkit for the modeling and simulation of distributed resource mgnt and scheduling for Grid computing,
2. Dazhang Gu, Lin Yang, Lonnie R. Welch ,Center for Intelligent, Distributed and Dependable Systems ,School of Electrical Engineering & Computer Science ,Ohio University, A Predictive, Decentralized Load Balancing Approach.
3. Akshay Luther, Rajkumar Buyya, Rajiv Ranjan, and Srikumar Venugopal, “Peer-to-Peer Grid Computing and a .NET-based Alchemi Framework”, GRIDS Laboratory, The University of Melbourne, Australia
4. Ian Foster , Carl Kesselman Steven Tuecke , The Anatomy of the Grid Enabling Scalable Virtual Organizations , Intl J. Supercomputer Applications, 2001
5. Francois Grey, Matti Heikkurinen, Rosy Mondardini, Robindra Prabhu, “Brief
6. Hans-Ulrich Heiss and Michael Schmitz, Decentralized Dynamic Load Balancing: The Particles Approach.
7. Junwei Cao1, Daniel P. Spooner, Stephen A. Jarvis, and Graham R. Nudd, Grid Load Balancing Using Intelligent Agents.
8. Ian Foster, Argonne National Laboratory & University of Chicago, What is the Grid? A Three Point Checklist.
9. Jennifer M. Schopf, Mathematics and Computer Science Division, Argonne National Lab, Department of Computer Science, Northwestern University, Grids: The Top Ten Questions.
10. Karl Czajkowski, Ian Foster and Carl Kesselman, Resource Co-Allocation in Computational Grids.
11. Ann Chervenak, Ian Foster, Carl Kesselman, Charles Salisbury and Steven Tuecke, The Data Grid:Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets
12. Javier Bustos Jimenez, Robin Hood: An Active Objects Load Balancing Mechanism for Intranet.
13. Manish Arora and Sajal K. Das and Rupak Biswas, A De-centralized Scheduling and Load Balancing Algorithm for Heterogeneous Grid Environment
14. [14] H.D. Karatza. Job scheduling in heterogeneous distributed systems. Journal. of Systems and Software, 56:203–212, 1994.
15. Javier Bustos Jimenez, Robin Hood “An Active Objects Load Balancing Mechanism for Intranet”.

16. 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".
17. Parag Kulkarni & Indranil Sengupta Department of Computer Science & Engg. Indian Institute of Technology, Kharagpur, "Load Balancing With Multiple Token Policy", ICTA'07, April 12-14, Hammamet, Tunisia.
18. P. Kokkinos, K. Christodoulopoulos, A. Kretsis, and E. Varvarigos Department of Computer Engineering and Informatics, University of Patras, Greece and Research Academic Computer Technology Institute, Patras, Greece, "Data Consolidation: A Task Scheduling and Data Migration Technique for Grid Networks" Eighth IEEE International Symposium on Cluster Computing and the Grid.
19. Hendra Rahmawan, Yudi Satria Gondokaryono, "The Simulation of Static Load Balancing Algorithms", 2009 International Conference on Electrical Engineering and Informatics, Malaysia.
20. Sandeep Sharma, Sarabjit Singh, and Meenakshi Sharma, "Performance Analysis of Load Balancing Algorithms", academy of science, engineering and technology, issue 38, February 2008, pp. 269-272.
21. Derek L. Eager, Edward D. Lazowska , John Zahorjan, "Adaptive load sharing in homogeneous distributed systems", IEEE Transactions on Software Engineering, v.12 n.5, p.662-675, May 1986.
22. William Leinberger, George Karypis, Vipin Kumar- "Load Balancing Across Near-Homogeneous Multi-Resource Servers" Army High Performance Computing and Research Center Department of Computer Science and Engineering, University of Minnesota February 16, 2000
23. D. L. Eager, E. D. Lazowska, and J. Zahorjan. Adaptive load sharing in homogeneous distributed systems. IEEE Trans. on Software Engineering, SE-12(5):340–353, May 1986.
24. D. L. Eager, E. D. Lazowska, and J. Zahorjan. A comparison of receiver-initiated and sender-initiated adaptive load sharing. Performance Evaluation, 6:53–68, 1986.
25. L. V. Kale. Comparing the performance of two dynamic load distribution methods. In Proc. Intl. Conference on Parallel Processing, pages 77–80, August 1988.
26. [26] V. Kumar, A. Gramma, and V. Rao. Scalable load balancing techniques for parallel computers. Journal of Parallel and Distributed Computing, 22(1):60–79, July 1994.
27. M. Livny and M. Melman. Load balancing in homogeneous broadcast distributed systems. In Proc. ACM Computer Network Performance Symposium, pages 47–55, April 1982.
28. M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. In 8th IEEE Heterogeneous Computing Workshop (HCW'99), April 1999.
29. N. G. Shivaratri, P. Krueger, and M. Singhal. Load distributing for locally distributed systems. IEEE Computer, 25(12):33–44, December 1992.