International Journal of Soft Computing and Engineering (IJSCE)
ISSN: 2231-2307, Volume-2 Issue-1, March 2012

A Study on the Assessment of Load Balancing Algorithms in Grid Based Network

Sachin Kumar, Niraj Singhal

Abstract- **Grid computing comprises of distributed computer systems which are geographically dispersed to share the combination of resources in a heterogeneous environment. The ever varying and increasing demands of the computational resources have generated the need for solutions that are more flexible. With the use of a high tech computer that has more and faster processors and auxiliary storage space or more RAM (random access memory), it is not well enough for a solution as the system usage patterns differ. A grid based distributed system can solve this problem by allowing multiple independent jobs to run over a network with heterogeneous computing nodes. A network-aware load balancing algorithms that are dynamic as well as quick are the requirement of a network of computers to keep the workload balanced, represented by these jobs. The purpose of this paper is to review various different load balancing algorithms for the grid based distributed network, identify several comparison metrics for the load balancing algorithms and to carry out the comparison based on these identified metrics between them.**

Keywords- dynamic load balancing algorithms; grid based distributed network; comparison metrics; Heterogeneous node; Load Balancing Policy

I. INTRODUCTION

Grid computing is a distributed infrastructure which allows large scale resource sharing and system integration. Load balancing is the most important factor to improve the efficiency and performance of a multiple nodes in a grid based distributed network. An independent program or a partitioned module of a parallel task or program is called as workload. Grid connected computers are basically a combination of computing resources applied to a common task, usually to a scientific, technical or business problem that requires a major number of processing cycles or the need to process huge amount of data. In a grid some nodes may be heavily loaded while some may be idle.
Therefore, load balancing is the problem of distributing workload among physically separated nodes during run time. Workload distribution is carried out in such a way that a set of independent tasks are distributed among all the computing nodes of the grid so that the jobs are uniformly distributed and none of the nodes are overloaded or under loaded. A load balancing algorithm improves the performance of the system. However, the degree of improvement in the performance of the system depends not only on the specific algorithm used but also on the degree of uneven distribution of load over the nodes. Many scheduling and load balancing solutions have been proposed for conventional distributed computing systems. Dynamic decentralized approach is one of the main load balancing strategies which considers the run time environment before distribution of jobs among the nodes in a grid. The dynamic decentralized approach is preferred because of the varying nature of the elements of the grid in capacity or number during run time and also be heterogeneous in nature giving rise to different loading circumstances. In this paper we have compared the different load balancing algorithms for the grid based on various different metrics such as makespan, load balancing time, average resource utilization rate, communication overhead, reliability, stability, fault tolerance and scalability.
The organization of the paper is as follows: In Section II, an introduction to various methods of performing load balancing for a grid is given. In section III, the policies of dynamic load balancing for heterogeneous resources are considered. In Section IV, the different metrics for comparing the load balancing algorithms is identified. In section V, we discuss about different load balancing algorithms. In section VI, a comparison is made among algorithms in a tabular form. Finally, a conclusion is made.

II. DIFFERENT LOAD BALANCING STRATEGIES

The main objective behind load balancing is to distribute a set of independent jobs among all the computing nodes of a grid such that jobs are uniformly distributed among nodes and there is no under loaded or overloaded nodes in the grid. We can classify load balancing strategies as static and dynamic.
a) Static load balancing- In static load balancing algorithms, information about all the characteristics of the jobs, the computing nodes and the communication network is known in advance. Therefore, load balancing decisions are made at compile time and remain stable at execution time. 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 nodes for performance statistics. However, the major drawbacks are, first, the workload distribution of several applications cannot be predicted before program

Manuscript Received February 26, 2011.
* **Sachin Kumar**, Senior Lecturer, Department of Computer Science & Engineering, Hermes College of Engineering & Management (Guru Nanak Education Trust's Group of Institutions), Roorkee, Uttrakhand, (India), (e-mail: sachinagnihotri16@gmail.com).
****Niraj Singhal**, Assistant Professor, Faculty of Electronics, Informatics and Computer Engineering, Shobhit University, Meerut., Uttar Pradesh (India), (e-mail: sonia_niraj@yahoo.com)

execution; second, it is assumed that all the characteristics of the computing resources and communication network are all known in advance and remain unchanged. Such a hypothesis may not apply to a distributed environment. Because the static approach cannot respond to a dynamic runtime environment, it may lead to load imbalance on some nodes and significantly increase the load balancing time [3]. However, static algorithms only work well when there is not much variation in the load on the workstations [4]. Static load balancing is not a good choice for load balancing in dynamic and ever changing large distributed systems like grid.

b) Dynamic load balancing - Dynamic load balancing algorithms make changes to the distribution of work among nodes at run-time; they use current or recent load information when making distribution decisions [5]. Despite the higher runtime complexity, dynamic algorithms can potentially provide better performance than static algorithms. For heterogeneous networks where network elements may vary in capacity or number at runtime, dynamic load balancing strategies are preferred. Dynamic strategies increase the communication overhead due to exchanging information among nodes. Dynamic load balancing strategies can be further categorized as centralized approach and decentralized approach. In the centralized approach, only one node in the grid work as a central manager or master node. Its work is to assign jobs to each of the slave nodes. The slave nodes execute the jobs assigned to them by the master node.

Centralized approach is used mainly for a small size grid. It is a simple approach which gives better results when the communication cost is less significant. The main drawback of this approach is that, this approach depends on the master node as if it fails the entire system is also failed.

In the **decentralized approach**, all nodes in the grid participate in load balancing decision making. The decentralized approach is the preferred choice because the network elements usually are of different capacity or they may vary in number during the run time. The advantage of decentralized algorithms is that they are scalable and have better fault tolerance. The only drawback with decentralized approach is that it increases the communication overhead to a large extent [3] for the network with dynamic heterogeneous resources.

III. THE POLICIES OF DYNAMIC LOAD BALANCING ALGORITHMS FOR HETEROGENEOUS RESOURCES

A typical dynamic load balancing algorithm is defined by their four basic inherent policies which are Transfer policy, Selection policy, Location policy and Information policy [6, 7].

a) Transfer Policy: Transfer policy determines the condition under which a task should be transferred. Atypical transfer policy includes task migration and/or task rescheduling. Migration is suspending an executing task, transferring it to another processor, and resumes its execution from the state of suspension. The transfer policy decides if there is a need to initiate load balancing across the system. By using workload information, it determines when a node becomes eligible to act as a sender or as a receiver.

b) Selection Policy: once the transfer policy decides that a host is a sender, a selection policy selects a task for transfer. The simplest and popular approach is to select the newly arrived task for transfer that just transforms the host into a sender. A selection policy considers several factors in selecting a task:

i. The overhead incurred by the transfer should be minimal. For example, transfer of a small task gain less overhead.

ii. The selected task should be long lived so that it is worthwhile to acquire the transfer overhead.

iii. The number of location-dependent system calls made by the selected task should be minimal. Location dependent calls are system calls that must be executed on the host where the task originated, because they use the resources available on that host only.

c) Location Policy: The location policy determines a suitably under loaded node. In other words, it locates complementary nodes to/from which a node can send/receive workload to improve the overall system performance.

d) Information Policy: The main focus of information policy is to decide the time when information about the status of other hosts in the system is to be collected. There are three types of information policies:

i. Demand driven policy: Decentralized approach, collects host information only when host becomes either a sender or a receiver.

ii. Periodic policy: Either centralized or decentralized approach, collect information periodically.

iii. State-change driven policies: Either centralized or decentralized approach, the hosts circulate information about their states change by a certain degree.

IV. METRICS USED FOR ASSESSMENT OF VARIOUS DYNAMIC LOAD BALANCING ALGORITHMS

The several metrics identified for assessing the load balancing algorithm are- **Communication overhead** – Communication overhead is the status information which every node has to convey to other nodes in the grid.

- **Load balancing time** – Amount of time that elapses between the job arrival time and the time at which the job is finally accepted by a node.
- **Makespan** – Makespan is the total completion time taken to allocate all tasks to a resource. It is the measure of the throughput of the grid.
- **Average resource utilization rate** – This means the usage of all the resources in the grid.
- **Scalability** – It is the ability of the algorithm to perform load balancing for a grid with any finite number of nodes.
- **Fault tolerance** – It is the ability of the algorithm to perform uniform load balancing in spite of arbitrary node or link failure.
- **Reliability** – It is the ability of the load balancing algorithm to schedule job in predetermined amount of time.
- **Stability** – It can be characterized in terms of the delays

in the transfer of information between nodes and the gains in the load balancing algorithm.

V. VARIOUS DYNAMIC LOAD BALANCING ALGORITHMS

The purpose of this section is to study the approaches of dynamic load balancing algorithms. The centralized and decentralized load balancing algorithms can be further classified according to the four above discussed basic policies. They are sender initiated; receiver initiated and symmetrically initiated algorithms. Sender initiated algorithms let the heavily loaded nodes take the initiative to request the lightly loaded nodes to receive the jobs; while receiver initiated algorithms let the lightly loaded nodes invite heavily loaded nodes to send their jobs. Symmetrically-initiated algorithms combine the advantages of both sender and receiver initiated algorithms [3].

In QoS priority based scheduling algorithm [8], resources are first granted to high priority jobs and then it assigns the low priority job for scheduling. It allocates the job to resource based on their complexity. High complex jobs are allocated to high processing speed system and low complex jobs are allocated to hybrid system. So the makespan and resource utilization for all the tasks are ultimately improved. It assigns tasks to all the resources uniformly, so the load balancing problem is solved in the algorithm. This algorithm shows better makespan and resource utilization rate results than QWMTM, Min Min heuristic algorithms and Max Min heuristic algorithms. In the receiver broadcasting algorithm [9], it is proposed that whenever a processor becomes idle, it broadcasts a request message in the grid. Upon receiving this request, the most heavily loaded node sends a task to the request generated processor. The algorithm does not require an information process to collect the load information of other nodes, which acquires heavy communication traffic. This algorithm shows better total execution time of jobs and better node utilization with a smaller number of job migrations. The nodes which are located far away from each other can communicate quickly. This algorithm requires additional hardware for implementing the single bus structure and arbitration logic.

Decentralized dynamic scheduling scheme [10] for P2P desktop grids, improves total system throughput, by improving load balance across heterogeneous nodes and lowering average job queue wait times. The algorithm also avoids starvation of large jobs through the backfilling counter mechanism. This approach also shows performance competitive with that of greedy online centralized algorithm, and does not acquire high messaging costs.

In [11], a concept of load balancing for mobility-enabled Mobile Agent Systems has introduced and dynamic approach to a runtime allocation of load via agent migrations in a network of hosts is implemented. When resources on a node is running short, a certain agent who is responsible for the load balancing coordinates and tries to find a remote host with enough capacity to take over one of the local agents. This approach is suitable for small scale agent based applications. The decentralized approach provides for stability and sturdiness, due to the fact that there is no single point of failure. The algorithm proposed in [1] correlates the scheduling of incoming jobs and balancing of the loads at each node in a multi cluster. It eliminates the time delay due to arbitration in the receiver broadcasting algorithm in their dynamic scheduling using weights algorithm by proposing that the jobs are assigned to those nodes which have minimum memory utilization or maximum elapsed time. The main target of this algorithm is to develop a new methodology to schedule incoming jobs as well as to balance the load in the system. Job size is taken into account by the algorithm. The method used being centralized in nature increases the load balancing time. The individual nodes are also homogeneous in nature.

In [2], an approach is proposed which overcomes the restrictions imposed by homogeneous nature of nodes. The controller node first tries to distribute the jobs uniformly among its own nodes. If it is unable to distribute load uniformly then it sends the jobs to some other controller node. The controller broadcasts the load balancing requests to other controllers in the grid. Each controller node submits a bid. The controller which submits the highest bids then takes the responsibility of load balancing. This algorithm is sender initiated as well as receiver initiated. The performance improves as the number of nodes in the system increases, but if it exceeds the number of jobs, no improvements results.

These load balancing algorithms take the dynamic run time environment into consideration before assigning jobs to the node. They are fault tolerant in the sense that the faulty nodes do not provide status information. Therefore jobs will not be sent to that node. They are also scalable but at a certain cost in terms of load balancing time or communication overhead.

VI. COMPARISON OF VARIOUS DYNAMIC ALGORITHMS ON DIFFERENT METRIC

A comparison has been made among various dynamic load balancing algorithms on different metric in the following table:

Algorithm/ Metric	QoS priority based scheduling	Receiver Broad-Casting Algorithm	De-centralized Scheme for P2P grids	Load balancing for mobility Enabled systems	Dynamic scheduling algorithm with weights	Competition based dynamic scheduling algorithm
Communication Overhead	More	Very Less	More	More	Less	Less
Makespan	Less	More	more	Less	More	More
Load balancing time	Less	More	more	Less	More	More
Scalability	Scalable	Scalable	Scalable	Scalable	Scalable	Scalable
Avg. Resource Utilization rate	Average	Less	More	More	More	More
Fault Tolerance	Integrated	Integrated	Integrated	Integrated	Integrated	Integrated
Reliability	Integrated	Integrated	Integrated	Integrated	Integrated	Integrated
Stability	Incorporated	Not Incorporated	Incorporated	Incorporated	Not Incorporated	Not Incorporated

.Figure 1. Comparison of load balancing algorithms

VII. CONCLUSION

The decentralized approach for load balancing is more robust than the centralized approach. The only drawback is that it increases communication overhead. Centralized approach is mainly used for small size grids because of its simplicity and less communication overhead. In order to balance the load uniformly in a grid one has to choose a combination of centralized, decentralized, sender-initiated and receiver initiated approach. The communication overhead and load balancing time depends upon the approach selected in the algorithm. In order to perform the load balancing identical, a choice between the communication overhead and load balancing time has to be done. 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 future aspect is to develop a decentralized approach which can balance the tradeoff between communication overhead and load balancing time in order to make it scalable.

REFERENCES

1. M. V. Gopalachari, P. Sammulal, A. V. Babu, "Correlating Scheduling and Load balancing to achieve optimal performance from a cluster", WEE International Computing Conferences (IACC 2009), March 2009.
2. A. Abed, G. Oz, A. Kostin, "Competetion based Load Balancing for Dstributed Systems", Proceedings of the seventh IEEE International Symposium on Computer Networks (ISCN' 06).
3. K. Lu, R. Subrata, A. Zomaya, An Efficient Load Balancing Algorithm for Heterogeneous Grid Systems considering Desirability of Grid Sites, 25th IEEE International Conference on performance, Computing and Communication, April 2006.
4. B. Lee, *Dynamic Load Balancing in a Message Passing Virtual Parallel Machine*. Technical Report, Division of Computer Engineering, School of Applied Science, Nanyang Technological University, Singapore, 1995
5. S. Dandamundi, *Sensitivity Evaluation of Dynamic Load Sharing in Distributed Systems*, Technical Report TR 97-12, Carleton University, Ottawa, Canada.
6. N. G. Shivratri, P. Krueger, M. Singhal; "Load Distributing for Locally Distributed Systems", Computer, vol. 25, no. 12, Dec. 1992.
7. R. Mukhopadhyay, D. Ghosh, N. Mukherjee; "A Study of Existing Load Balancing Algorithms for Large, Dynamic, heterogeneous Distributed Systems", Proceedings of the 9th WSEAS International Conference on SOFTWARE ENGINEERING, PARALLEL and DISTRIBUTED SYSTEMS (SEPADS '10)
8. T. Amudha, T T Dhivyaprabha; "QoS Priority Based Scheduling Algorithm and Proposed Framework for Task Scheduling in a Grid Environment" IEEE-International Conference on Recent Trends in Information Technology, MIT, Anna University, Chennai, June, 2011
9. W. Lee, S. Hong, J. Kim, "Dynamic Load Distribution on a mesh with a single bus", IEEE International Conference on Parallel and Distributed Systems, Dec. 1997.
10. J. Lee, P. Keleher, A. Sussman; "Decentralized Dynamic Scheduling across Heterogeneous Multi-core Desktop Grids", IEEE, May-2010.
11. A. Singh; "An Efficient Load Balancing Algorithm for Grid Computing using Mobile Agent", International Journal of Engineering Science & Technology (IJEST), June-2011.

AUTHOR DETAILS

Sachin Kumar received his B Tech in Computer Science & Engineering from Uttar Pradesh Technical University, Lucknow & now pursuing his M Tech from Shobhit University, Meerut,(U.P.). Currently, He is working as a Senior Lecturer in Computer Science Department in Hermes College of Engineering & Management (Guru Nanak Education Trust's Group of Institutions), Roorkee. His area of interests including Data Structures & Algorithms, Distributed Operating Systems, Computer Networks and Network Security. His current research focus is Load Balancing in Distributed Computing.

Niraj Singhal received his M.Tech. in Computer Engineering (First class with honours), and currently pursing Ph.D. in Computer Engg from Shobhit University Meerut. He is member of International Association of Computer Science and Information Technology (IACSIT) Singapore, International Association of Engineers (IAENG) HONG KONG, life member of Computer Society of India (CSI) and Indian Society for Technical Education (ISTE). He has more than twenty five research publications to his credit in International and National journals/conferences of repute. He has the privilege of conducting several courses /workshops for undergraduate students for enhancing their programming skills. He is the author of two handbooks for undergraduate students, and a series of computer science books for school students as per CBSE/NCERT guidelines. Presently he is working as Assistant Professor in the Faculty of Electronics, Informatics and Computer Engineering, Shobhit University, Meerut.. His area of interest includes System programming, Web information retrieval and Knowledge based systems.