International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 6 (2018) pp. 3683-3687
© Research India Publications. http://www.ripublication.com

A Novel Approach to Improve the Performance of Grid Systems by Analyzing Attentive Dynamic Load Balancing Algorithm

Nilesh Korde and Abhijeet Thakare

[1,2]*Assistant Professor, Department of Computer Application, Shri Ramdeobaba College of Engineering and Management, Gittikhadan, Katol Road, Nagpur-440013, India.*

[1,2]*Orcid Id: 0000-0001-8820-6983, 0000-0003-1096-6418*

Abstract

Grid computing is considered as future of the distributed systems. Mostly discussed issues in distributed systems are proper resource utilization and reduction in process execution time. These issues can be handled by implementation proper load balancing algorithm. A proper load balancing algorithm will improve the overall performance of the distributed systems. To solve the above mentioned issues, an efficient load balancing algorithm is proposed which allocates the load hierarchically across virtual organization in the set up on the basis of their least threshold value. The same algorithm is also compared with the already existing interest attentive dynamic load balancing algorithm. The tag line is to reduce the complexity of the algorithm to reduce the process execution time.

Keywords - Load balancing, Resource Utilization, Process execution time, Grid systems, Threshold, Cluster organization.

INTRODUCTION

The current era of distributed systems has been dominated by Hi-speed distributed system environment. Such environment is called as grid computing environment. There are mostly two issues that are frequently discussed in distributed systems that are proper resource allocation and reduction in process execution time. If proper load balancing mechanism is implemented at grid level then it becomes very simple to get rid of the above mentioned things. Though there is lot many load balancing mechanisms are proposed but very few of them are standard. Research is constantly going on to settle down the things. To contribute the same, this paper proposes and analyzes interest attentive dynamic load balancing algorithm.

Load balancing mechanism is implemented at the top node in the hierarchical structure of grid computing environment. Role of the load balancing mechanism is to divide the tasks evenly based on the number of nodes in an environment and distribute to the cluster organization based on their current threshold value [1]. This mechanism is responsible to reduce the process execution time and resource utilization. To implement this, load balancing mechanism should be very strong to distribute the load in the most sufficient way without too much processing [1, 2].

Load balancing algorithm is basically implemented by using two different types of techniques. The techniques are Static Load balancing and Dynamic Load balancing. In Static Load balancing, load is assigned to every commuting node before the actual execution process [3]. Load allocation is done by ignoring the current situation of the system. However in dynamic load balancing, load is allocated by considering the present state of the system. Thus, we can say that the threshold value varies time to time according to the current load that is the reason why such algorithms are dynamic in nature. The algorithms are divided into two categories centralized and distributed [2, 4]. In centralized approach topmost node controls all the other nodes and also the execution process. While in distributed approach individual node played its part in the execution process. In this paper we are proposing centralized approach with distributed process execution by the cluster organization.

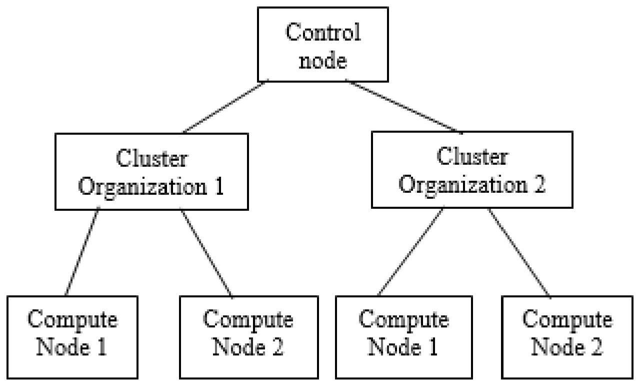

Figure 1.1: Hierarchical Structure of Grid Environment

The execution process goes like this: The request arrives at the control node; it computes the Average Threshold Value (ATV) for the whole system, at next hierarchical level cluster organization exists, which has its own Threshold Value (ATVC); before allocating the request or load it is always checked that the Threshold Value of the Cluster Organization (ATVC) should be less than Average Threshold Value (ATV) for system if it is so, then the load is allocated, otherwise not.

The whole paper is summarized as follows:

1. Introduction.
2. Related Work.
3. Components of Dynamic Load Balancing Algorithm
4. Analysis of Attentive Load balancing Algorithm.
5. Conclusion.

RELATED WORK

Many of the Load balancing algorithms has been proposed and implemented but even after that there is need to discuss the important issues of the distributed systems that are reduction in process execution time and proper load balancing. Almost all the papers discuss the implantation of load balancing algorithms on simulators. But on the simulators we can the judge the results what can be produced on real time set up. Hence our paper analyzes the attentive dynamic load balancing algorithm and also suggests some the modification's in the same algorithm.

The Load Balancing algorithm is either designed for homogeneous or heterogeneous system. Homogeneous system offers same hardware same processing capabilities and failure and recovery times. That is the reason that load balancing in such system is easy to achieve. But, the scenario is quite different today; nodes in networks of distributed system differ in their configuration. When large scientific applications runs in such environments, performance is affected by a number of factors such as variation in workload, variation in network latency and other factors mentioned in [1]. Thus, it is important to consider the heterogeneity and to decide while designing a DLB algorithm. Parameters for load balancing is as follows-

1. *CPU Utilization* - CPU utilization means an extent upto which CPU is utilized. The CPU utilization varies depending on the amount of tasks is evaluted and managed computing tasks [1, 5].
2. *Load Balancing* – Divides the number of tasks according to number of cluster organization and allocate to it. This allocation is based on the Threshold value, if Threshold of cluster is higher side the request is not allocated[4].
3. *Job Queue* – All the Jobs or Requests are initally received at control node which has a queue. All these jobs are scheduled form the queue by using any of the scheduling algorithm[5].
4. *Cluster Organization* – Set of computing machine is called as cluster. And collection of some cluster organization constituites a Grid[4].

5. *Compute Nodes* – An independent machine which belongs to a cluster is called as compute node. Role of these nodes are to execute the tasks allotted to them by the cluster nodes [4].

COMPONENTS OF DYNAMIC LOAD BALANCING ALGORITHM

A. Information Policy -

The information policy is responsible for collecting the system state information. First local state information is collected from the neighboring nodes by cluster organizational nodes; information collected at local level is responsible for calculating the global state information. Based on this, decision for the scheduling of the job has been taken.

More accurate is the system state information more effective is the load balancing decision taken by control node [6]: the accuracy of processor load estimates, the aging of information due to communication latency, and the frequency of load exchange. Following are the various information policies as in [7]: *periodic or batch policy, demand-driven policy, state change driven policy, request-reply policy, event-driven policy, and index variability based information policy*.

B. Transfer Policy –

The transfer policy is responsible to identify the state of the node whether it is lightly loaded or heavily loaded. Widely used transfer policy is threshold policy. Threshold values are classified as single and double Threshold policy [3]. In single-threshold policy, when the load at the node exceeds the threshold T, the transfer policy identifies that the node is an overloaded node. If the load at the node

Falls below T, the node is identified as under loaded node. In double-threshold policy, two threshold values T1 and T2 are used to describe node status as follows.

• If load = 0 load=idle

• Load < T1 _load= under loaded

• T1 < load < T2 load = Normal/Medium

• Load > T2 load = overloaded

C. Selection Policy -

The selection policy decides the best job to be transferred. There are several factors to consider in the selection of a job: (1) the overhead occurred by the transfer of job should be minimal; (2) the selected job should be long lived so that it should be worth to incur the transfer overhead as in [8]; (3) a job should be selected for transfer only if its response time will be improved upon transfer.

D. Location Policy –

The location policy is responsible for selecting the best node to allocate the task amongst set of all the available nodes. It finds suitable node to transfer the load. The factors to be considered while selecting the node may include resource availability as in [9]. The selected node should have the correct environment to run the process. Various location policies have been presented in the literature as in [2] [7] [8] that are *random policy, polling policy: threshold, greedy and* *shortest, negotiation policy: low load policy, broadcast a query, main and secondary locations policy.*

ATTENTIVE DYNAMIC LOAD BALANCING ALGORITHM

A. Calculation of Threshold-

Threshold value exists for an individual node. First the threshold is calculated for an individual node then average threshold value of complete system is evaluated. Threshold is calculated from the job queue that exists for each node. A manual value is set for the job queue if number of tasks/jobs in the queue exceeds that value then node is called as overloaded node, else, it is under loaded node. This value is calculated on 10-point scale like, if 4 is the current value and manually set threshold value is 7 that mean still 3 more tasks can be allocated to the same node. Average threshold (ATV) is calculated as-

$$\text{ATV} = \left(\sum_{i=1}^{n} \text{Individual Node}\right) / n$$

B. Calculation of Average Computing Power(ACP)-

Logic to calculate the ACP remains the same as that of previous method. First computing power of individual node is calculated and then an average is calculated. Computing power of every single machine is dependent on CPU utilization. This metric is always present for standalone node in terms of percentage. Again a manual percentage is assumed as a limit, if CPU utilization exceeds that limit again node is said to be overloaded. For example, if current CPU utilization is 40% and limit is 75% that means still some more tasks can be allocated to the node nearby that 75% mark.

$$\text{ACP} = \left(\sum_{i=1}^{n} \text{CPU Utilization of Individual node}\right) / n$$

Vital Conditions for execution of algorithm

At Control node

```
If (ACP<limit of CPU Utilization && ATV< limit of Job/Task Queue) then
    Allocate the job to the cluster organization
else
    Keep it in job queue of control node
```

At Cluster Organization Level

```
If (ACPC<Limit of CPU Utilization for Cluster Organization && ATVC for Cluster Organization)
    Allocate the job to the its compute node
else
    Keep it in job queue of selected Control organization.
```

Algorithm-

Step 1: Identify all the cluster organizations first and receive the request at the control node.

Step 2: For every received request calculate the ATV and ACP mentioned as above-

Step 3: For every received request apply the BFS algorithm to find out the best suited cluster organization so that the request can be allocated to it.

Step 4: Once the best cluster organization is found allocate a load to it.

Step 5: When cluster organization receives the load calculates it own ATVC and ACPC for itself.

Step 6: According to previous calculated values it allocate the job/task to its compute nodes.

Step 7: Compute node will execute the task and return the results back to the nodes hierarchically.

PROPOSED ALGORITHM

The algorithm discussed in the previous section executes the BFS algorithm for the arrival every so that the request to be allocated to the best cluster organization amongst the given set up. But execution of the BFS algorithm for every arrival of request is not a good idea as it produces additional processing overhead. Existence of this processing overhead increases the process execution time, which decreases the overall system performance. So, to overcome this drawback we are proposing few modifications in the same algorithm. Basically, we have to reduce this processing overhead this can be done by implementing any sorting algorithm instead of BFS algorithm. As it is clearly known that the time complexity of sorting algorithm is less as compared to that of BFS algorithm.

In our proposed algorithm we are sorting the cluster organizations in the ascending order of their current threshold value. This sorting is implemented by bubble sort which as best case time complexity as $O(n)$ and worst case is $O(n^2)$. Rest of the algorithms remains the same. The algorithm is as follows –

Algorithm-

Step 1: Identify all the cluster organizations first and receive the request at the control node.

Step 2: For every received request calculate the ATV and ACP mentioned as above-

Step 3: For every received request, sort the cluster organization according to their current load in the ascending order. This can be maintained by using queue data structure at control node.

Step 4: Entry of node at the first position in the queue indicates that the Cluster organization has minimum threshold value. So, cluster organization in front of the queue will be selected to allocate the load.

Step 5: Once the best cluster organization is found allocate a load to it.

Step 6: When cluster organization receives the load calculates it own ATVC and ACPC for itself.

Step 7: Again the cluster organization executes the bubble sort algorithm and sort the compute nodes in the ascending order of their threshold value. This can be done by maintaining a queue at cluster organizational level.

Step 8: In the same way as previous, first node in the queue indicates that it is least loaded node. And load is allocated accordingly.

A vital condition that exists in the attentive load balancing algorithm remains the same. Their role is to support more accurate load balancing. Thus we can logically say that time complexity of any algorithm is directly proportional to process execution time. So, it is always better to select

As proposed algorithm mentions that by implementing bubble sort algorithm we can get efficient results. The only difference between the two algorithms is the time complexity of BFS and Bubble sort algorithm which are executed after arrival of every request. However it is not mandatory to use Average computing Power and Average threshold value both at the time of implementation. We can also use any one of them for implementation and execution. But using them together would generate more accurate results.

CONCLUSION

In this paper, we discussed a novel approach for several components of load balancing mechanism based on which proper allocation should be done. Although from the related work we can say that more and more work is done on load balancing mechanism that too by using the simulators. More and more work is required to be done on the real time environment to obtain the real time results. Secondly, load balancing algorithm needs to be more accurate, if we choose two parameters for load balancing then mechanism will be much accurate. And last, but the most important is time complexity of the algorithm makes an impact on the process execution time of the job. Logically, if we reduce the complexity then the process execution time decreases. And the basic issues in the load balancing that we discussed in the beginning are reducing process execution time and proper resource allocation to achieve the load balancing.

REFERENCES

[1] http://www.igiglobal.com

[2] Emmanuel Jeannot, "A Practical Approach of Diffusion Load Balancing Algorithms", Lecture Notes in Computer Science, 2006.

[3] www.infosys.usyd.edu.au

[4] Niranjan Shivratri and Mukesh Singhal: Advanced Concepts in operating Systems:Distributed Database and multiprocessor opearting Systems.

[5] Avi Silberschatz, Peter Baer Galvin, Greg Gagne: Operating System Concepts, *Eight Edition*

[6] Z. Khan, R. Singh, J. Alam, and R. Kumar, “Performance Analysis of Dynamic Load Balancing Techniques for Parallel And Distributed Systems,” Int. J. of Computer and Network Security, vol. 2, Feb. 2010, pp. 123-127.

[7] P. K. Chandra and B. Sahoo, “Prediction Based Dynamic Load Balancing Techniques in Heterogeneous Clusters,” Prod. Int. Conf. Computer Science and Technology, Apr. 2008, pp. 189-192.

[8] R. Mukhopadhyay, D. Ghosh, and N. Mukherjee, “A Study on the Application of Existing Load Balancing Algorithms for Large, Dynamic, Heterogeneous Distributed Systems,” Proc. 9th WSEAS Int. Conf. Software Engineering, Parallel and Distributed Systems (SEPADS), 2010, pp. 238-243.

[9] A. Hac and T. J. Johnson, “A Study of Dynamic Load Balancing in a Distributed System,” Proc. ACM Conf. Communications, Architectures and Protocols, vol. 16, Aug. 1986, pp.348-356.

[10] Mohd. Haroon and Mohd. Hussain,” Interest Attentive Dynamic Load Balancing in Distributed Systems”, 2nd International Conference on Computing for Sustainable Global Development (INDIACom), 2015.

[11] Mayuri A. Mehta,” Designing an Effective Dynamic Load Balancing Algorithm Considering Imperative Design Issues in Distributed Systems”, International Conference on Communication Systems and Network Technologies, 2012.

[12] https://www.linux.com/community/blogs/133-general-linux/9401

[13] Ajay Tiwari and Priyesh Kanungo, “A Model for Dynamic Load Balancing in Open Source Software for Distributed Computing Environment”, IEEE, 2012.

[14] 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.

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