International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 03 | Mar 2020 www.irjet.net p-ISSN: 2395-0072

Efficient Load Balancing in a Distributed Environment

Ankit Kumar Singh[1], Dhairya Kalpeshbhai Patel[2], Kaustumbh Jaiswal[3], Dr. Ram Mohan Babu[4], Shikhar Sharma[5]

[1,2,3,5]*B.E. Student, Dept. of Information Science and Engineering, Dayananda Sagar College of Engineering, Kumaraswamy Layout, Bengaluru, India*
[4]*HOD, Dept. of Information Science and Engineering, Dayananda Sagar College of Engineering, Kumaraswamy Layout, Bengaluru, India*

---***---

Abstract - *Cloud computing is an emerging paradigm in computing. It aims for the transparent sharing of data, calculations and service over a scalable node network. Since cloud computing stores the data in the open environment and disseminates the resources, the quantity of data storage is fast growing. Load balancing is a key issue within cloud storage. Maintaining load information will consume a lot of cost as the network is too large to spread load in a timely manner. Load balancing is one of the main challenges in cloud computing that is required to spread the distributed workload over several nodes to ensure that no single node is overloaded. It helps in optimum resource utilization and thus in improving system performance. With effective job scheduling and resource management techniques, a few existing scheduling algorithms can preserve load balance and provide better strategies as well. To achieve maximum profits with automated load balancing algorithms, it is necessary to make efficient use of resources. This paper addresses some of the current algorithms for load balancing in cloud computing.*

Key Words: cloud computing, load balancing, dynamic workload.

1. INTRODUCTION

Significant advances in computer technology have led to increased demand for high-speed computing and the need for fast scalability, availability and rapid response. It led to the use of parallel and distributed computing systems where the job is performed concurrently by more than one processor. Effective strategy to spread workload across multiple processors is one of the main research issues in parallel and distributed systems. Load balancing is used to minimize response time, optimize the performance and prevent overload. Load balancing is designed to ensure that every processor in the system does about the same amount of work at any time.

2. LOAD BALANCING

Load balancing is a relatively new technique facilitating networks and resources by delivering maximum throughput with minimum response time. Dividing traffic between servers allows data to be sent and received without any significant delay. There are various types of algorithms that help load traffic between available servers. Websites may be linked to a basic example of load balancing in our daily lives. The users could encounter delays, timeouts and possibly long device responses without load balancing. Load balancing solutions usually apply redundant servers which help to better distribute communication traffic so that the availability of the website is certainly settled. There are many different types of algorithms for load balancing available which can be classified primarily into two categories, static and dynamic.

2.1 Static Algorithms

Static algorithms equivalently divide the traffic between servers. Through this strategy the traffic on the servers will be quickly disdained and thus the situation will become more imperfect. This algorithm is named as round robin algorithm, which splits the traffic equally. There have been many problems in this algorithm, however so weighted round robin was described to enhance the crucial round robin challenges. Every server was assigned a weight in this algorithm and obtained more connections according to the highest weight. Servers will receive balanced traffic in a situation where all the weights are equal.

Before you begin to format your paper, first write and save the content as a separate text file. Keep your text and graphic files separate until after the text has been formatted and styled. Do not use hard tabs, and limit use of hard returns to only one return at the end of a paragraph. Do not add any kind of pagination anywhere in the paper. Do not number text heads-the template will do that for you.

2.2 Dynamic Algorithms-

Dynamic algorithms assigned appropriate weights on servers and chose to balance the traffic by finding a lightest server in the entire network. Selecting a suitable server, however, involved real-time contact with the networks, resulting in additional traffic being introduced to the system. Comparing these two algorithms, though round robin algorithms based on simple rule, this resulted in more loads conceived on servers and therefore imbalanced traffic. However; dynamic algorithm based on query that can be done frequently on servers, but sometimes prevailing traffic will prevent these queries from being answered and can be distinguished accordingly by more overhead on network.

2.3 Load Balancing in Distributed Systems-

Today more modern software development methodologies are being used to improve usability of applications embedded in distributed networks on compatible hardware. To achieve this goal and enhance the software infrastructure, middleware has been used to facilitate portability and interpretability of the distributed portion of the application. Middleware is defined as network services and software components enabling the application and networks to be scaled. Middleware has eased the task of designing and developing and handling the distributed applications by providing easy and integrated distributed programming environment.

3. ISSUE IN LOAD BALANCING

The following issues are analyzed during load balancing:

a. The communication channels are of finite bandwidth in the distributed environment and the processing units may be physically distant, therefore load balancing needs to decide whether or not to allow task migration.
b. A computing job may not be arbitrarily divisible which leads to certain dividing tasks constraints.
c. That job consists of several smaller tasks and may have different execution times for each one of those tasks.
d. The load on each processor as well as on the network can differ from time to time, depending on the users' workload.
e. The capacity of the processors in architecture, operating system, CPU speed, memory size and usable disk space that vary from one another.

Taking into account the above factors the load balance can be simplified into four basic steps:

a. Processor load and state control
b. Exchange of load and state data between processors
c. Calculating the distribution of new works
d. Actual movement in the data.

4. ADVANTAGES OF LOAD BALANCING

Some major advantages of load balancing are as follows:

a. It reduces the task waiting time.
b. It minimizes the time required to respond to tasks.
c. It maximizes the use of resources at the system.
d. It maximizes system performance.
e. This improves system reliability, and stability.
f. It's adjusting to future changes.
g. Long hunger for the small work is avoided.
h. The overall performance of the network is improved in load balancing by improving the performance of each node.

5. METRICS FOR LOAD BALANCING

Different measurements found in current load balance techniques are discussed below-

a. Scalability is an algorithm's ability to perform load balancing of any finite number of nodes for a network. The metric needs to be improved.
b. Resource Use is used to test the resources use. For efficient load balancing it should be configured.
c. Quality is used to test device performance. It needs to be improved at a reasonable cost, e.g., while maintaining sufficient delays, reducing task response time.
d. Response Time is the amount of time taken in a distributed system to respond by a given load balancing algorithm. We will minimize this parameter.
e. Overhead Associated determines the amount of overhead involved while a load-balancing algorithm is implemented. It is composed of overhead due to work movement, inter-processor and coordination between processes. This should be reduced to allow for efficient operation of a load balancing technique.

The aim and motivation of this survey is to provide a systematic review of existing load balancing algorithms and to encourage the amateur researcher in this field to contribute to the development of a more effective load balancing algorithm. This will be of benefit to interested researchers in carrying out further research in this area.

6. Load BALANCING ALGORITHMS

6.1 Round Robin

The processes in this algorithm [2] are divided among all processors. In round robin order each cycle is allocated to the processor. The process allocation order is kept locally independent from the remote processor allocations. Although the distribution of workloads between processors is the same, the processing time for different processes is not the same so some nodes can be heavily loaded at any point of time and others remain idle. This algorithm is mostly used in web servers where http requests are similar in nature and are equally distributed.

6.2 Connection Mechanism

Load balancing algorithm [3] can also be based on the least link mechanism that is part of the algorithm for dynamic scheduling. To estimate the load it needs to dynamically count the number of connections for each server. The load balancer records each server's contact number. The number of connections increases when a new link is routed to it, and the number decreases when the connection is terminated or timeout occurs.

6.4 Randomized(RAND)

By fact the randomized algorithm is of a static kind. Within this algorithm [2] a method can be treated with a probability p by a particular node n. For each processor, the process allocation order is maintained independently of allocation from remote processor. This algorithm works well in case of similarly loaded procedures. Problems arise when loads are of different computational complexities. Randomized algorithms do not uphold deterministic approaches. As Round Robin algorithm generates overhead for process queue it works well.

6.4 Equally Spread Current Execution Algorithm

Unit manage with priorities fairly distributed current execution algorithm [4]. Distribute the load randomly by testing the size and moving the load to a virtual machine that is easily loaded or manages the job, taking less time and optimizing the throughput. It is the method of spreading the range in which the load balancer spreads the load of the job in hand over multiple virtual machines.

6.5 Throttled Load Balancing Algorithm

Throttled algorithm [4] is entirely virtual machine based. First, this client asks the load balancer to check the right virtual machine that can easily access that load and perform the operations that the client or user is giving. The client first asks the load balancer to find a suitable Virtual Machine to perform the necessary operation in this algorithm.

6.6 A Task Scheduling Algorithm Based on Load Balancing

Y. Fang et al. [5] addressed a two-level task scheduling system focused on load balancing to meet users ' complex requirements and achieve a high utilization of resources. By first mapping tasks to virtual machines and virtual machines to host resources, it enables load balancing, thus enhancing task response time, resource utilization and overall environmental performance.

6.7 Biased Random Sampling

M. Randleset al. [6] explored a distributed and flexible load balancing approach that uses random system domain sampling to achieve self-organization, thus balancing the load across all network nodes. Here a virtual graph is built, reflecting the load on the server with the connectivity of each node (a server is viewed as a node). Every server is symbolized as a node in the graph, each being directed in degree to the server's free resources. The load balancing scheme used here is fully decentralized, making it ideal in a cloud for such large network systems. With an increase in population diversity the efficiency is reduced.

6.8 Min-Min Algorithm

It begins with a collection of unassigned tasks. First of all, minimum completion time is to be found for all activities. Then the minimum value is chosen from these minimum times which is the minimum time on any resource among all the tasks. Then the job on the corresponding computer is scheduled according to that minimum time. The execution time for all other tasks on that machine is then changed by adding the execution time of the assigned task to the execution times of other tasks on that machine and assigned task is removed from the list of tasks to be assigned to the machines. The same process is then followed again until all the tasks are performed on the capital. But this strategy has a major drawback which can lead to starvation [7].

6.9 Max-Min Algorithm

Max-Min is almost the same as the min-min algorithm except for the following: after finding the minimum execution times, the maximum value is chosen which is the maximum time on any resource between all tasks. Then the job on the corresponding computer is scheduled according to that maximum time. The execution time for all other tasks on that machine is then changed by adding the execution time of the assigned task to the execution times of other tasks on that machine, and the assigned task is removed from the list of tasks to be assigned to the machines [7].

6.10 Token Routing

The main aim of the algorithm [9] is to reduce the cost of the system by shifting the tokens around it. Yet due to connectivity bottleneck, agents in a distributed cloud system cannot have the necessary information to spread the workload. So the distribution of workload between the agents is not fixed. The token routing algorithm's drawback can be removed with the help of token based load balancing heuristic approach. This algorithm makes routing decision quick and efficient. In this algorithm agent need not have an idea of the complete knowledge about the working load of their global state and neighbors. This method does not involve any communication overhead.

6.11 Nearest Neighbor Algorithm

Each processor considers only its immediate neighbor processors to execute load balancing operations with nearest neighbor algorithm. Based on the load it has and the load details to its immediate neighbors a processor takes the balancing decision [11]. Through successively swapping the load for the adjacent nodes the network maintains a globally distributed state of operation. The nearest neighbor algorithm is split primarily into two groups which are method of diffusion and method of sharing of dimensions. With this method a processor that is strongly or lightly loaded balances its load concurrently with all its closest neighbors at a time, while a processor balances its load successively with its neighbor one at a time in dimension swap form.

6.12 Adaptive Contracting with Neighbor (ACWN)

Once the workload is newly created, it is transferred to the nearest neighbor processor which is least loaded. The processor which accepts the load holds the load in its local heap. If the load in its heap is smaller than its threshold level, otherwise it will transfer the load below the threshold load to the neighboring processor. ACWN thus requires retaining local load information as well as adjacent load information to regularly swap the load. RAND is therefore different from the ACWN in that ACWN always finds the target node that is least loaded in the neighborhood [11].

6.13 Prioritized Random (PRAND) Algorithm

The workload is expected to be consistent in both RAND and ACWN in the light of their statistical specifications. Changes are made to the non-uniform workload on RAND and ACWN to get RAND (PRAND) prioritized and ACWN (PACWN) prioritized respectively. In these equations, index numbers are allocated to the workloads based on the weight of their heaps. PRAND is similar to RAND except that it chooses from the heap the second largest weighted load and passes it to a chosen neighbor at random. On the other hand, PACWN chooses the second largest weighted workload and moves it to the neighbor who is least loaded[11].

6.14 Cyclic Algorithm

This is the product of the slight modification of the random algorithm. Cyclically the task is allocated to a remote device. This algorithm also tells of the last device a procedure was sent to.

6.15 Probabilistic

Every node carries a vector of load including the load of a subset of nodes. The first half of the load vector which also carries the local load is sent to a randomly selected node periodically. When information is updated in this manner and without transmission, the information can be distributed over the network. Nevertheless, this algorithm's efficiency isn't optimal, its extensibility is low, and insertion is delayed [1].

6.16 Threshold and Least

THRESHOLD and LEAST, both use a partial knowledge gained through the exchanging of messages. In THRESHOLD, a node is randomly selected to accept a migrated load. If the load is below the load level, the load acknowledged by that load. Then, polling with another node is replicated to find a suitable node to pass the load. If no suitable receiver has been identified, the procedure is executed locally after a limited number of attempts. LEAST is an instant of THRESHOLD, and is chosen to obtain the migrated load after polling the least loaded machine [1]. THRESHOLD and LEAST perform well, and are basic in design. In fact, these algorithms use the up-to-date load values.

6.17 Reception

In this algorithm, nodes below the threshold load consider the overloaded node for migration load from overloaded node by random polling.

6.18 Centralized Information and Centralized Decision

In this class of algorithms the system information is stored in a single node, and that single node also takes the decision. CENTRAL is a subset of that algorithm. When a node that is heavily loaded needs to move a task, it asks a server for a node that is lightly loaded. The server machine is informed by each node in the system whether or not a lightly loaded node is available. CENTRAL delivers very efficient results on performance [1]. But this algorithm suffers from a very serious problem, that this algorithm will not provide any facilities if the server crashes.

6.19 Centralized Information and Distributed Decision

In GLOBAL, information gathering is centralized while decision making is distributed [1]. Server broadcasts the load situation on the nodes. Through this knowledge an overloaded processor identifies the lightly loaded node without going through the server from its load vector. Due to the lower inclusion of message information, this algorithm is very efficient and robust in nature, because the system remains alive even when the server is in recovery. GLOBAL algorithm gathers large amounts of information but the information is not up to date. As a result, there are greater overheads in the program.

6.20 Distributed information and Distributed Decision

Each node in the system regularly transmits its load condition, and each node maintains a global load vector. This algorithm performs poorly. Both the information and the judgment are transmitted in RADIO, and without coercion there is no broadcast. A distributed list of lightly loaded nodes in this algorithm in which each computer is aware of its successor and predecessor. In addition, each node knows the head of the available list, which is called the manager. Migration of a network from a heavily loaded node through the manager to the lightly loaded node is performed directly or indirectly. Broadcasting happens when manager crashes or when a node is added to the list available[1].

6.21 The Shortest Expected Delay (SED) Strategy

This strategy efforts to reduce the anticipated delay of completion of each job so that the destination node is chosen to minimize the delay. This is a greedy technique in which each work fulfills its best interest and enters the queue that can reduce the anticipated completion delay. This method minimizes the average delay of any given batch of jobs with no further subsequent delivery. SED does not reduce the average time period for an ongoing process of arrival. To evaluate the destination node the source node must collect state information for location policy from other nodes.

6.22 Modified SED

In the SED strategy communication delay is introduced. A job experiences communication delay because of the transfer of jobs from the source node to the destination node. Another modification occurs because of a node's limited input queue size [6]. If a job is sent to a node whose input queue is saturated, the job cannot enter the queue and will be transferred to the second-best node, which may also saturate the input queue, and so forth. If there is no node open for a job then it is always moved to the original node. This work will be temporarily held at this node if the queue remains filled and tries to re-enter after a set interval of time. This delay is considered in the modifies SED.

6.23 The Never Queue (NQ) Strategy

NQ policy is a different technique in which the source system calculates the cost of sending a job to each final destination or to a subset of final destinations, and the work is put on the server at reduced cost [10]. This algorithm often places a job on a server which is quickest available. This algorithm minimizes the extra delay in successive arriving workers, so that NQ policy minimizes the total delay. In addition, a server does not transfer incoming work to servers until the server is the fastest one available.

6.24 Modification of NQ strategy

NQ strategy is always keen to find the best idle server. If no idle server is available it will pick the server with the shortest expected delay. If a server is unable to accept the job, it will be moved to another system, with the shortest estimated wait, etc. If no server is finally found with the shortest anticipated time, the job will be backed up to the original server anyway. Because of the congestion of the input queue the job may have to wait there temporarily. So the job will get the chance after a certain interval of time when the input queue has enough space to accommodate it. This delay in is considered in modified NQ.

6.25 Greedy Throughput (GT) Strategy

This approach is distinct from Strategies for SED and NQ. GT approach deals with the system's throughput which is the number of jobs done per unit time would be maximum before the arrival of new job instead of optimizing only the throughput rate at the balancing moment. That is why this program is called Greedy Throughput (GT) [10].

7. CONCLUSIONS

In this paper different load balancing techniques are discussed thoroughly in detail. Load balancing in distributed environment is one of the hottest areas of research as the demand for heterogeneous computing is increasing with the wise utilization of the web. The performance of the computing system increases with the load balancing algorithm. Here in this paper we have drawn a comparison between Static Load Balancing(SLB) and Dynamic Load Balancing(DLB) by introducing some parameters. Different facilities of load balancing algorithms are enumerated in the paper. Finally, we studied some of the most important DLB algorithms and compared them to focus their importance in different situations. There does not exist any algorithm which is absolutely perfect but one can use one of the algorithms mentioned above based on the situation. The comparative study not only provides an overview of the load balancing algorithms, but also offers practical guidelines to researchers in designing the most efficient load balancing algorithm for distributed environment.

REFERENCES

[1] Bernard G., Steve D. and Simatic M. “A Survey of Load Sharing in Networks of Workstations”. The British Computerm Society, The Institute of Electrical Engineers and IOP Publishing Ltd, 75-86,1993.

[2] Zhong Xu, Rong Huang, (2009) “Performance Study of Load Balanacing Algorithms in Distributed Web Server Systems”, CS213 Parallel and Distributed Processing Project Report.

[3] P.Warstein, H.Situ and Z.Huang(2010), “Load balancing in a cluster computer” In proceeding of the seventh International Conference on Parallel and Distributed Computing, Applications and Technologies, IEEE.

[4] Ms.NITIKA, Ms.SHAVETA, Mr. GAURAV RAJ; “Comparative Analysis of Load Balancing Algorithms in Cloud Computing”, International Journal of Advanced Research in Computer Engineering & Technology Volume 1, Issue 3, May 2012.

[5] Y. Fang, F. Wang, and J. Ge, “A Task Scheduling Algorithm Based on Load Balancing in Cloud Computing”, Web Information Systems and Mining, Lecture Notes in Computer Science, Vol. 6318, 2010, pages 271-277.

[6] T.R.V. Anandharajan, Dr. M.A. Bhagyaveni” Co-operative Scheduled Energy Aware Load-Balancing technique for an Efficient Computational Cloud” IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 2, March 2011.

[7] T. Kokilavani J.J. College of Engineering & Technology and Research Scholar, Bharathiar University, Tamil Nadu, India” Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing” International Journal of Computer Applications (0975 – 8887) Volume 20– No.2, April 2011.

[8] Peter Mell, Timothy Grance, “The NIST Definition of Cloud Computing”, NIST Special Publication 800-145, September 2011.

[9] Zenon Chaczko, Venkatesh Mahadevan, Shahrzad Aslanzadeh, Christopher Mcdermid (2011)“Availabity and Load Balancing in Cloud Computing” International Conference on Computer and Software Modeling IPCSIT vol.14 IACSIT Press,Singapore 2011.

[10] Weinrib and Shenker S. “Greed is not Enough: Adaptive Load Sharing in Large Heterogeneous Systems”. INFOCOM, 986-994, 1988

[11] Xu C. and Lau F.C.M. “Load Balancing in Parallel Computers: Theory and Practice”. Kluwer Academic Press, 1997.