Global Journal of Engineering Science and Research Management

STUDY OF VARIOUS LOAD BALANCING TECHNIQUES AND CHALLENGES IN CLOUD COMPUTING

Rakesh Patel[*1] Mili patel[2] Proff.Anupam R chaube[3]
[*1]Research Scholar, MIET Gondia.
[2]Research Scholar, MIET Gondia.
[3]Faculty and Research Supervisor of M.E. GH Raisoni Collage of Engineering Nagpur.
*Correspondence Author: Rakeshpatel.kit@gmail.com

Keywords: Load Balancing, Cloud Computing

Abstract

Cloud computing is emerging technology which is a new standard of large scale distributed computing and parallel computing. It provides shared resources, information, software packages and other resources as per client requirements at specific time. As cloud computing is growing rapidly and more users are attracted towards utility computing, better and fast service needs to be provided. For better management of available good load balancing techniques are required. So that load balancing in cloud becoming more interested area of research. And through better load balancing in cloud, performance is increased and user gets better services. Here in this paper we have discussed many different load balancing techniques used to solve the issue in cloud computing environment.

Introduction

It is a process of reassigning the total load to the individual nodes of the collective system to make resource utilization effective and to improve the response time of the job, simultaneously removing a condition in which some of the nodes are over loaded while some others are under loaded. A load balancing algorithm which is dynamic in nature does not consider the previous state or behavior of the system, that is, it depends on the present behavior of the system. The important things to consider while developing such algorithm are : estimation of load, comparison of load, stability of different system, performance of system, interaction between the nodes, nature of work to be transferred, selecting of nodes and many other ones . This load considered can be in terms of CPU load, amount of memory used, delay or Network load Load balancing is one of the main issues related to cloud computing. The load can be a memory, CPU capacity, network or delay load. It is always required to share work load among the various nodes of the distributed system to improve the resource utilization and for better performance of the system. This can help to avoid the situation where nodes are either heavily loaded or under loaded in the network. Load balancing is the process of ensuring the evenly distribution of work load on the pool of system node or processor so that without disturbing, the running task is completed.

Goals of load balancing

As given the goals of load balancing are:

- ✓ To improve the performance substantially
- ✓ To have a backup plan in case the system fails even partially
- ✓ To maintain the system stability
- ✓ To accommodate future modification in the system

Types of load balancing algorithms

Depending on who initiated the process, load balancing algorithms can be of three categories as given in:

- ✓ **Sender Initiated**: If the load balancing algorithm is initialized by the sender
- ✓ **Receiver Initiated**: If the load balancing algorithm is initiated by the receiver
- ✓ **Symmetric**: It is the combination of both sender initiated and receiver initiated

Depending on the current state of the system, load balancing algorithms can be divided into 2 categories as given in

Static algorithm

In static algorithm the traffic is divided evenly among the servers. This algorithm requires a prior knowledge of system resources, so that the decision of shifting of the load does not depend on the current state of system. Static algorithm is proper in the system which has low variation in load.

Global Journal of Engineering Science and Research Management

Dynamic algorithm

In dynamic algorithm the lightest server in the whole network or system is searched and preferred for balancing a load. For this real time communication with network is needed which can increase the traffic in the system. Here current state of the system is used to make decisions to manage the load.

Static load balancing algorithm

Static load balancing policies are generally based on the information about the average behavior of system; transfer decisions are independent of the actual current system state. Static load balancing schemes use a priori knowledge of the applications and statistical information about the system.

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 assigned by the master processor. 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 execution time, minimizing the communication delays.

A general disadvantage of static approaches 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.

There are four types of static load balancing: - Round Robin algorithm, Randomized algorithm, Central Manager Algorithm, and Threshold algorithm.

Round robin algorithm

Round Robin algorithm 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. The main 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.

However when the jobs are of unequal processing time this algorithm suffers as the some nodes can become severely loaded while others remain idle. Round Robin is generally used in web servers where generally HTTP requests are of similar nature and thereby be distributed equally.

Randomized algorithm

Randomized algorithm 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

Central Manager Algorithm, 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.

This algorithm is expected to perform better than the parallel applications, especially when dynamic activities are created by different hosts.

Threshold algorithm

In Threshold algorithm 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 under loaded, medium and overloaded. Two threshold parameters t_under and t_upper can be used to describe these levels.

$$\textbf{Under loaded: } load < t\_under,$$
$$\textbf{Medium : } t\_under \leq load \leq t\_upper,$$
$$\textbf{Overloaded: } load > t\_upper.$$

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.

Global Journal of Engineering Science and Research Management

If the local state is not overloaded then the process is allocated locally. Otherwise, a remote under loaded processor is selected, and if no such host exists, the process is also allocated locally. Thresholds algorithm have low inter process communication and a large number of local process allocations. The later decreases the overhead of remote process allocations and the overhead of remote memory accesses, which leads to improvement in performance.

A disadvantage of the algorithm is that all processes are allocated locally when all remote processors are overloaded. A load on one overloaded processor can be much higher than another overloaded processor, causing significant disturbance in load balancing, and increasing the execution time of an application.

Dynamic load balancing algorithm

In a distributed system, dynamic load balancing can be done in two different ways: distributed and non-distributed. In the distributed one, the dynamic load balancing algorithm is executed by all nodes present in the system and the task of load balancing is shared among them. The interaction among nodes to achieve load balancing can take two forms: cooperative and non-cooperative. In the first one, the nodes work side-by-side to achieve a common objective, for example, to improve the overall response time, etc. In the second form, each node works independently toward a goal local to it, for example, to improve the response time of a local task. Dynamic load balancing algorithms of distributed nature, usually generate more messages than the non-distributed ones because, each of the nodes in the system needs to interact with every other node. A benefit, of this is that even if one or more nodes in the system fail, it will not cause the total load balancing process to halt, it instead would affect the system performance to some extent. Distributed dynamic load balancing can introduce immense stress on a system in which each node needs to interchange status information with every other node in the system. It is more advantageous when most of the nodes act individually with very few interactions with others.

In non-distributed type, either one node or a group of nodes do the task of load balancing. Non-distributed dynamic load balancing algorithms can take two forms: centralized and semi-distributed. In the first form, the load balancing algorithm is executed only by a single node in the whole system: the central node. This node is solely responsible for load balancing of the whole system. The other nodes interact only with the central node. In semi-distributed form, nodes of the system are partitioned into clusters, where the load balancing in each cluster is of centralized form. A central node is elected in each cluster by appropriate election technique which takes care of load balancing within that cluster. Hence, the load balancing of the whole system is done via the central nodes of each cluster

Centralized dynamic load balancing takes fewer messages to reach a decision, as the number of overall interactions in the system decreases drastically as compared to the semi distributed case. However, centralized algorithms can cause a bottleneck in the system at the central node and also the load balancing process is rendered useless once the central node crashes. Therefore, this algorithm is most suited for networks with small size.

Policies or Strategies in dynamic load balancing

- ***Transfer Policy***: The part of the dynamic load balancing algorithm which selects a job for transferring from a local node to a remote node is referred to as Transfer policy or Transfer strategy.
- ***Selection Policy***: It specifies the processors involved in the load exchange (processor matching)
- ***Location Policy***: The part of the load balancing algorithm which selects a destination node for a transferred task is referred to as location policy or Location strategy.
- ***Information Policy***: The part of the dynamic load balancing algorithm responsible for collecting information about the nodes in the system is referred to as Information policy or Information strategy.

Challenges for load balancing

There are some qualitative metrics that can be improved for better load balancing in cloud computing.

Throughput: It is the total number of tasks that have completed execution for a given scale of time. It is required to have high through put for better performance of the system.

Associated Overhead: It describes the amount of overhead during the implementation of the load balancing algorithm. It is a composition of movement of tasks, inter process communication and inter processor. For load balancing technique to work properly, minimum overhead should be there.

Fault tolerant: We can define it as the ability to perform load balancing by the appropriate algorithm without arbitrary link or node failure. Every load balancing algorithm should have good fault tolerance approach.

Migration time: It is the amount of time for a process to be transferred from one system node to another node for execution. For better performance of the system this time should be always less.

Global Journal of Engineering Science and Research Management

Response time: In Distributed system, it is the time taken by a particular load balancing technique to respond. This time should be minimized for better performance.

Resource Utilization: It is the parameter which gives the information within which extant the resource is utilized. For efficient load balancing in system, optimum resource should be utilized.

Scalability: It is the ability of load balancing algorithm for a system with any finite number of processor and machines. This parameter can be improved for better system performance.

Performance: It is the overall efficiency of the system. If all the parameters are improved then the overall system performance can be improved.

Reference

1. *Sandeep Sharma, Sarabjit Singh, and Meenakshi Sharma, "Performance Analysis of Load Balancing Algorithms", World Academy of Science, Engineering and Technology, 2008.*
2. *Hisao Kameda, El-Zoghdy Said Fathyy and Inhwan Ryuz Jie Lix, "A Performance Comparison of Dynamic vs. Static Load Balancing Policies in a Mainframe { Personal Computer Network Model", Proceedings of the 39th IEEE Conference on Decision and Control,2000.*
3. *Daniel Grosua, Anthony T. and Chronopoulosb,"Non-cooperative load balancing in distributed systems", Elsevier, Journal of Parallel and Distributed Computing, 2005.*
4. *M. Nikravan and M. H. Kashani, "A Genetic Algorithm for Process Scheduling in Distributed Operating Systems Considering Load balancing", Proceedings 21st European Conference on Modelling and Simulation (ECMS), 2007.*
5. *Hendra Rahmawan, Yudi Satria Gondokaryono, "The Simulation of Static Load Balancing Algorithms", 2009 International Conference on Electrical Engineering and Informatics, Malaysia.*
6. *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.*
7. *S. Malik, "Dynamic Load Balancing in a Network of Workstation", 95.515 Research Report, 19 November,2000.*
8. *Ali M. Alakeel, AGuide to Dynamic Load Balancing in Distributed Computer Systems,IJCSNSInternational Journal of Computer Science and Network Security, VOL.10 No.6, June 2010.*
9. *William Leinberger, George Karypis, Vipin Kumar, "Load Balancing Across Near Homogeneous Multi-Resource Servers", 0-7695-0556-2/00, 2000 IEEE.*
10. *Anthony T.Velte, Toby J.Velte, Robert Elsenpeter, Cloud Computing a Practical Approach, TATA McGraw-Hill Edition 2010.*
11. *Martin Randles,DavidLamb, A. Taleb-Bendiab,A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing, 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops.*
12. *Mladen A. Vouk, Cloud Computing Issues, Research and Implementations, Proceedings of the ITI 2008 30th Int. Conf. on Information Technology Interfaces, 2008, June 23-26.*
13. *Ali M.Alakeel, A Guide to Dynamic Load Balancing in Distributed Computer Systems, IJCSNS International Journal of Computer Science and Network Security VOL.10 No.6, June 2010.*
14. *http://www03.ibm.com/press/us/en/pressrelease/22613.wss*