International journal of Computer Science and Engineering (IJCSE)
ISSN 2278-9960
Vol. 2, Issue 2, May 2013, 101-108

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

ANALYSIS OF LOAD BALANCERS IN CLOUD COMPUTING

SHANTI SWAROOP MOHARANA[1], RAJADEEPAN D. RAMESH[2] & DIGAMBER POWAR[3]

[1,2]Research Scholar, Department of Electrical Engineering BITS Pilani, Hyderabad, India

[3]Lecturer, Department of Computer Science BITS Pilani, Hyderabad, India

ABSTRACT

Cloud Computing is high utility software having the ability to change the IT software industry and making the software even more attractive. It has also changed the way IT companies used to buy and design hardware. The elasticity of resources without paying a premium for large scale is unprecedented in the history of IT industry. The increase in web traffic and different services are increasing day by day making load balancing a big research topic. Cloud computing is a new technology which uses virtual machine instead of physical machine to host, store and network the different components. Load balancers are used for assigning load to different virtual machines in such a way that none of the nodes gets loaded heavily or lightly. The load balancing needs to be done properly because failure in any one of the node can lead to unavailability of data.

KEYWORDS: Cloud Computing, Static Load Balancer, Dynamic Load Balancer, Load Balancer Algorithms

INTRODUCTION

Load balancing[1] is a generic term used for distributing a larger processing load to smaller processing nodes for enhancing the overall performance of system. In a distributed system environment, it is the process of distributing load among various other nodes of distributed system to improve both resource utilization and job response time. An ideal load balancing algorithm should avoid overloading or under loading of any specific node. But, in case of a cloud computing environment the selection of load balancing algorithm is not easy because it involves additional constraints like security, reliability, throughput etc. So, the main goal of a load balancing algorithm in a cloud computing environment is to improve the response time of job by distributing the total load of system. The algorithm must also ensure that it is not overloading any specific node.

Load balancers can work in two ways: one is cooperative and non-cooperative. In cooperative, the nodes work simultaneously in order to achieve the common goal of optimizing the overall response time. In non-cooperative mode, the tasks run independently in order to improve the response time of local tasks.

Load balancing algorithms, in general, can be divided into two categories: static and dynamic load balancing algorithm. A static load balancing algorithm does not take into account the previous state or behavior of a node while distributing the load. On the other hand, a dynamic load balancing algorithm checks the previous state of a node while distributing the load. The dynamic load balancing algorithm is applied either as a distributed or non-distributed. The advantage of using dynamic load balancing is that if any node fails, it will not halt the system; it will only affect the system performance. In a dynamic load balanced system, the nodes can interact with each other generating more messages when compared to a non-distributed environment. However, selecting an appropriate server needs real time communication with the other nodes of the network and hence, generates more number of messages in the network. Dynamic load balancer uses policies for keeping track of updated information. Dynamic load balancers have four policies:

- Transfer policy-It is used when a selected job is needed for transfer from a local to a remote node.
- Selection policy-It is used when processors exchange information between them.
- Location Policy-It is responsible for selecting a destination node for the transfer.
- Information Policy-It is used to maintain a record of all the nodes of the system.

The task of load balancing is shared among distributed nodes. In a distributed system, dynamic load balancing can be done in two different ways. In the distributed one, the dynamic load balancing algorithm is executed by all nodes present in the system and the task of scheduling is shared among all nodes. Whereas, in the undistributed one, nodes work independently in order to achieve a common goal. We will be using three criteria for comparing the load balancing algorithms:

- Throughput-It can be estimated by calculating the total number of jobs executed within a fixed span of time without considering the virtual machine creation and destruction time.
- Execution time-It can be estimated by measuring the time taken for completing the following processes time for formation of virtual machines, processing time of a process and destruction time of a virtual machine.

In general, any cloud computing algorithm performs the following decision making steps:

- Client requests the cloud coordinator.
- Cloud coordinator divides the task into cloudlet, and sends it to data centers.
- Data center coordinators work on scheduling the tasks. It also selects the algorithm, which will be used for scheduling tasks.

In this paper, we will analyze the various static and dynamic load balancing algorithms.

STATIC LOAD BALANCERS

Round-Robin Load Balancer [2]

It is a static load balancing algorithm, which does not take into account the previous load state of a node at the time of allocating jobs. It uses the round robin scheduling algorithm for allocating jobs. It selects the first node randomly and then, allocates jobs to all other nodes in a round robin manner. This algorithm will not be suitable for cloud computing because some nodes might be heavily loaded and some are not. Since the running time of any process is not known prior to execution, there is a possibility that nodes may get heavily loaded.

Hence, weighted round-robin algorithm [3] was proposed to solve this problem. In this algorithm, each node is assigned a specific weight. Depending on the weight assigned to the node, it would receive appropriate number of requests. If the weights assigned to all the nodes are equal, then each node will receive same traffic. In cloud computing system, precise prediction of execution time is not possible therefore, this algorithm is not preferred.

Min-Min

It is a static load balancing algorithm. So, all the information related to the job is available in advance. Some terminology related to static load balancing:

- **ETC:** The expected running time of the job on all nodes are stored in an ETC (expected time of compute) matrix. If a job is not executable on a particular node, the entry in the ETC matrix is set to infinity.

- **OLB:** It is an opportunistic Load Balancing, in which each job is assigned to the node in an arbitrary order, irrespective of the ETC on the nodes. It provides load balance schedule but it results in very poor make-span.
- **MET:** It is a Minimum Execution Time algorithm. In this, each job is assigned to the node which has the smallest execution time as mentioned in ETC table, regardless of the current load on that processor. MET tries to find the best job-processor pair, but it does not take into consideration the current load on the node. This algorithm improves the make-span to some extent, but it causes a severe load imbalance.
- **MCT:** It is a Minimum Completion Time algorithm which assigns jobs to the node based on their minimum completion time. The completion time is calculated by adding the expected execution time of a job on that node with node's ready time. The node with the minimum completion time for that particular job is selected. But this algorithm considers the job only one at a time.

Min-Min algorithm[4,5] begins with a set of all unassigned jobs. First of all, minimum completion time for all jobs is calculated. The job with minimum completion time is selected. Then, the node which has the minimum completion time for all jobs is selected. Finally, the selected node and the selected job are mapped. The ready time of the node is updated. This process is repeated until all the unassigned jobs are assigned. The advantage of this algorithm is that the job with the smallest execution time is executed. The drawback of this algorithm is that some jobs may experience starvation.

Max-Min

Max-Min[4,5] is almost same as the min-min algorithm except the following: after finding out minimum completion time of jobs, the maximum value is selected. The machine that has the minimum completion time for all the jobs is selected. Finally the selected node and the selected job are mapped. Then the ready time of the node is updated by adding the execution time of the assigned task.

Load Balance Min-Min

LBMM [5,6] is a static load balancing algorithm. This algorithm implements load balancing among nodes by considering it as a scheduling problem. The main aim of this algorithm is to minimize the make-span, which is calculated as the maximum of the completion times of all the jobs scheduled on their respective resources. This algorithm performs the following steps for scheduling the jobs on the nodes.

- Execute the Min-Min scheduling algorithm and calculate the make-span.
- Select the node with the highest make-span value.
- Corresponding to that node, select the job with minimum execution time.
- The completion time of the selected job is calculated for all the resources.
- Maximum completion time of the selected job is compared with the make-span value. If it is less, the selected job is allocated to the node, which has the maximum completion time. Else, the next maximum completion time of the job is selected and the steps are repeated.
- The process stops if all the nodes and all the jobs are assigned.

In scenarios where the number of small tasks is more than the number of large tasks in meta-task, this algorithm achieves better performance. This algorithm does not consider low and high machine heterogeneity and task heterogeneity.

Load Balance Max-Min-Max

S.-C. Wang et al. [7] proposed a two-phase scheduling algorithm that combines OLB (Opportunistic Load Balancing) and LBMM (Load Balance Min-Min) scheduling algorithms. OLB scheduling algorithm keeps every node in working state to achieve the goal of load balancing and LBMM scheduling algorithm is utilized to minimize the execution time of each task on the node, thereby minimizing the overall completion time. This combined approach helps in efficient utilization of resources. This algorithm performs the following steps for load balancing:

- Average completion time of each task for all the nodes is calculated.
- Select the task with maximum average completion time.
- Select an unassigned node with minimum completion time that should be less than the maximum average completion time for the selected task. Then, the tasks dispatched to the selected node for computation.
- If all the nodes are already assigned, re-evaluate by considering both assigned and unassigned nodes. Minimum completion time is computed as :
    - Minimum completion time of the assigned node is the sum of the minimum completion time for all the tasks assigned to that node and minimum completion time of the current task.
    - Minimum completion time of the assigned node is the minimum completion time of the current task.
- Repeat step 2 to step 4, until all tasks are executed. This gives better results than the above discussed algorithms.

DYNAMIC LOAD BALANCERS

Equally Spread Current Execution

Equally Spread current execution [2] is a dynamic load balancing algorithm, which handles the process with priority. It determines the priority by checking the size of the process. This algorithm distributes the load randomly by first checking the size of the process and then transferring the load to a Virtual Machine, which is lightly loaded. The load balancer spreads the load on to different nodes, and hence, it is known as spread spectrum technique.

Throttled Load Balancer

Throttled load balancer [2] is a dynamic load balancing algorithm. In this algorithm, the client first requests the load balancer to find a suitable Virtual machine to perform the required operation.

In Cloud computing, there may be multiple instances of virtual machine. These virtual machines can be grouped based on the type of requests they can handle. Whenever a client sends a request, the load balancer will first look for that group, which can handle this request and allocate the process to the lightly loaded instance of that group.

Honeybee Foraging Algorithm

The main idea behind the Honeybee Foraging algorithm[8] is derived from the behavior of honeybees. There are two kinds of honeybees: finders and reapers. The finder honeybees first goes outside of the honey comb and find the honey sources. After finding the source, they return to the honey comb and do a waggle dance indicating the quality and quantity of honey available. Then, reapers go outside and reap the honey from those sources. After collecting, they return to beehive and does a waggle dance. This dance indicates how much food is left. M. Randles proposed a decentralized honeybee based algorithm for self-organization. In this case, the servers are grouped as virtual server and each virtual server have a process queue. Each server, after processing a request from its queue, calculates the profit which is analogous to the quality

that the bees show in their waggle dance. If profit is high, the server stays else, it returns to the forage. This algorithm requires that each node to maintain a separate queue. This computation of profit on each node causes additional overhead. The disadvantage of this algorithm is that, it does not show any significant improvement in throughput, which is due to the additional queue and the computation overhead.

Biased Random Sampling

Biased Random Sampling [9] is a dynamic load balancing algorithm. It uses random sampling of system domain to achieve self-organization thus, balancing the load across all nodes of system. In this algorithm, a virtual graph is constructed with the connectivity of each node representing the load on server. Each node is represented as a vertex in a directed graph and each in-degree represents free resources of that node.

Whenever a client sends a request to the load balancer, the load balancer allocates the job to the node which has atleast one in-degree. Once a job is allocated to the node, the in-degree of that node is decremented by one. After the job is completed, the node creates an incoming edge and increments the in-degree by one. The addition and deletion of processes is done by the process of random sampling.

Each process is characterized by a parameter know as threshold value, which indicates the maximum walk length. A walk is defined as the traversal from one node to another until the destination is found. At each step on the walk, the neighbor node of current node is selected as the next node.

In this algorithm, upon receiving the request by the load balancer, it would select a node randomly and compares the current walk length with the threshold value. If the current walk length is equal to or greater than the threshold value, the job is executed at that node. Else, the walk length of the job is incremented and another neighbor node is selected randomly. The performance is degraded as the number of servers increase due to additional overhead for computing the walk length.

Active Clustering

Active Clustering [9,10] is a clustering based algorithm which introduces the concept of clustering in cloud computing. In cloud computing there are many load balancing algorithms available. Each algorithm has its own advantages and disadvantages. Depending on the requirement, one of the algorithms is used. The performance of an algorithm can be enhanced by making a cluster of nodes. Each cluster can be assumed as a group. The principle behind active clustering is to group similar nodes together and then work on these groups.

The process of creating a cluster revolves around the concept of match maker node. In this process, first node selects a neighbor node called the matchmaker node which is of a different type. This matchmaker node makes connection with its neighbor which is of same type as the initial node. Finally the matchmaker node gets detached. This process is followed iteratively. The performance of the system is enhanced with high availability of resources, thereby increasing the throughput. This increase in throughput is due to the efficient utilization of resources.

Join-Idle Queue

Join-Idle Queue[9] is basically used for large-scale systems. It uses distributed dispatchers by first load balancing the idle processors across dispatchers and then assigning jobs to processors to reduce average queue length at each processor. The disadvantage of this algorithm is that it is not scalable. Y. Lua et al.[3] proposed this load balancing algorithm for dynamically scalable web services. It effectively reduces the system load, incurs no communication overhead at job arrivals and does not increase actual response time. It can perform close to optimal when used for web services.

However, it cannot be used for today's dynamic-content web services due to the scalability and reliability.

ADDITIONAL ALGORITHMS

Compare and Balance

Compare and Balance- Y. Zhao et al. [10] addressed the problem of intra-cloud load balancing amongst physical hosts by adaptive live migration of virtual machines. A load balancing model is designed and implemented to reduce virtual machines migration time by shared storage, to balance load amongst servers according to their processor or IO usage, etc. and to have zero-downtime of virtual machines in the process. A distributed load balancing algorithm compare and balance is also proposed that is based on sampling and reaches equilibrium very fast. This algorithm assures that the migration of VMs is always from high-cost physical hosts to low-cost host but assumes that each physical host has enough memory which is a weak assumption.

Cartron

CARTRON- R. Stanojevic et al. [10] proposed a mechanism CARTRON for cloud control that unifies the use of LB and DRL. LB (Load Balancing) is used to equally distribute the jobs to different servers so that the associated costs can be minimized and DRL (Distributed Rate Limiting) is used to make sure that the resources are distributed in a way to keep a fair resource allocation. DRL also adapts to server capacities for the dynamic workloads so that performance levels at all servers are equal. With very low computation and communication overhead, this algorithm is simple and easy to implement.

LBVS

LBVS- H. Liu et al. [10] proposed a load balancing virtual storage strategy (LBVS) that provides a large scale net data storage model and Storage as a Service model based on Cloud Storage. Storage virtualization is achieved using an architecture that is three-layered and load balancing is achieved using two load balancing modules. It helps in improving the efficiency of concurrent access by using replica balancing further reducing the response time and enhancing the capacity of disaster recovery. This strategy also helps in improving the use rate of storage resource, flexibility and robustness of the system.

CONCLUSIONS AND FUTURE WORK

This paper is based on cloud computing technology which has a very vast potential and is still unexplored. The capabilities of cloud computing are endless. Cloud computing provides everything to the user as a service which includes platform as a service, application as a service, infrastructure as a service.

One of the major issues of cloud computing is load balancing because overloading of a system may lead to poor performance which can make the technology unsuccessful. So there is always a requirement of efficient load balancing algorithm for efficient utilization of resources. Our paper focuses on the various load balancing algorithms and their applicability in cloud computing environment.

We first categorized the algorithms as static and dynamic. Then we analyzed the various algorithms which can be applied in static environments. After that we described the various dynamic load balancing algorithms. For solving any particular problem some special conditions need to be applied. So we have discussed some additional algorithms which can help in solving some sub-problems in load balancing which are applicable to cloud computing. In our future work we will analyze the algorithms with numerical analysis and simulation.

REFERENCES

1. N. Ajith Singh, M. Hemalatha, "An approach on semi distributed load balancing algorithm for cloud computing systems" International Journal of Computer Applications Vol-56 No.12 2012.
2. Nitika, Shaveta, Gaurav Raj, International Journal of advanced research in computer engineering and technology Vol-1 issue-3 May-2012
3. Zenon Chaczko, Venkatesh Mahadevan, Shahrzad Aslanazadeh, and Christopher, IPCSIT Vol-14, IACSIT Press Singapore 2011
4. T. Kokilavani, Dr. D. I. George Amalarethinam "Load Balanced Min-Min Algorithm for Static Meta Task Scheduling in Grid computing" International Journal of Computer Applications Vol-20 No.2, 2011
5. Graham Ritchie, John Levine, ""A fast effective local search for scheduling independent jobs in heterogeneous computing environments" Center for Intelligent Systems and their applications School of Informatics University of Edinburg.
6. Shu Ching Wang, Kuo-Qin Yan Wen-pin Liao, and Shun Sheng Wang Chaoyang University of Technology, Taiwan R.O.C.
7. Che-Lun Hung, Hsiao-hsi Wang, and Yu- ChenHu "Efficient Load balancing Algorithm for cloud computing network"
8. Yatendra sahu, M. K. Pateriya "Cloud Computing Overview and load balancing algorithms", Internal Journal of Computer Application Vol-65 No.24, 2013.
9. Nayandeep Sran, Navdeep kaur "Comparative Analysis of Existing Load balancing techniques in cloud computing", International Journal of Engineering Science Invention, Vol-2 Issue-1 2013
10. Nidhi Jain Kansal, Inderveer Chana "Existing Load balancing Techniques in cloud computing: A systematic review" Journal of Information system and communication Vol-3 Issue-1 2012