ISSN:2229-6093

IJCTA

Hareesh M J et al, Int.J.Computer Technology & Applications,Vol 5 (2),640-645

A Review on Load Balancing Algorithms in Cloud

Hareesh M J
Dept. of CSE, RSET, Kochi
Email: hareeshmjoseph@gmail.com

John P Martin
Dept. of CSE, RSET, Kochi
Email: johnpm12@gmail.com

Yedhu Sastri
Dept. of IT, RSET, Kochi
Email: yedhusastri@gmail.com

Anish Babu S
Dept. of IT, RSET, Kochi
Email: anishbabus@gmail.com

Abstract

Load balancing is an important term as far as efficient and effective working of distributed system is concerned. As it requires all dynamic local resources to be distributed evenly to all nodes in a distributed system, It has become one of the main challenges in cloud computing. Many techniques were suggested for the effective and efficient balancing of load among nodes, which are distributed throughout a cloud. This algorithm improves the performance of the whole cloud by providing effective and efficient services to the users. In this paper, different algorithms for load balancing and task scheduling in cloud were compared and discussed, to provide the overview of the latest techniques in this field.

1. Introduction

Cloud computing is emerging as a new paradigm of large scale distributed computing. It has moved computing and data away from desktop to portable PCs into large data centers. It has the capability to harness the power of Internet and wide area network to resources that are available remotely, thereby, providing cost effective solution to the most of the real life requirement. It provides the scalable IT resources such as applications and service, as well as infrastructure on which they operate, over the Internet, as pay-per- use basis to adjust the capacity quickly and easily. It helps to accommodate changes in demand.

Thus cloud computing is a framework for enabling a suitable, and resource utility of the system. It also ensures for the fair distribution of work and resources.

Load balancing algorithms are classified as static and dynamic algorithms. Static algorithms are mostly suitable for homogeneous and stable environments and can produce very good results in these environments. However, they are usually not flexible and cannot match the dynamic changes to the attributes during the execution time. Dynamic algorithms are more flexible and take into consideration different types of attributes in the system both prior to and during run-time. These algorithms can adapt to changes and provide better results in heterogeneous and dynamic environments. However, as the distribution attributes become more complex and dynamic. As a result some of these algorithms could become inefficient and cause more overhead than necessary resulting in an overall degradation of the services performance.

The objective and motivation of this survey is to provide a analytic survey of existing load balancing techniques in cloud computing. The rest of the paper is organized as follows: We discuss the challenges of load balancing in cloud computing in Section II. Then, In Section III we go over the current literature and discuss the algorithms proposed to solve the load balancing issues in Cloud Computing. After that, we discuss and compare the relevant approaches in Section IV. We then conclude the paper and show possible areas of enhancement and our future plan of improving load balancing algorithms in Section V.

2. Challenges in implementing load balancing

Before going to the current load balancing approaches for Cloud Computing, we need to explain about the main problems and challenges that could affect how the algorithm would perform. Here we discuss the challenges to be addressed when attempting to propose an optimal solution to the issue of load balancing in Cloud Computing. These challenges are summarized below.

2.1. Cloud Node's Spatial Distribution

Some algorithms are developed to be efficient only for closely located nodes where communication delays are negligible. However, this is a basic issue to design and implement a load balancing algorithm that can work for nodes which are spatially distributed. This is because to give the service of maximum quality we must take in to account other factors such as the speed of the network links, the distance between the task processing nodes and client node , and the distances between the nodes. Here we must develop a load balancing mechanism among the nodes which are spatially distributed and also must efficiently and effectively tolerate high delays.

2.2. Storage/ Replication

An algorithm which implements full replication does not take in to account efficient storage utilization. This is because all replication nodes contain the same data. It also imposes higher costs since more storage is needed. At the same time, the algorithm which implements partial replication could save parts of the data sets in each node based on each node's processing power and capacity. This provides to better utilization, yet it increases the complexity of the load balancing algorithms as they attempt to take the availability of the data set's parts across the different Cloud nodes into account.

2.3. Complexity of the Load Balancing Algorithm

For implementation and operation of load balancing algorithms, algorithms with less complexity are preferred. The higher complexity would lead to a more complex process which could lead to some negative performance issues. Also, when the algorithms require more information and higher communication for monitoring and control, delays would cause more problems and the efficiency will drop. So, load balancing algorithms must be designed in the simplest possible forms.

2.3. Single Point Failure

Controlling the load balancing and collecting data about the different nodes must be designed in a way that avoids having a single point of failure in the algorithm. Some algorithms can provide efficient and effective mechanisms for solving the load balancing in a certain pattern. However, they have the issue of one controller for the whole system. In such cases, if the controller fails, then the whole system would fail. Any Load balancing algorithm must be designed in order to overcome this challenge. Distributed load balancing algorithms seem to provide a better approach, yet they are much more complex and require more coordination and control to function correctly.

3. Load balancing algorithms

In this section we explain the well known load balancing algorithms in Cloud Computing. The load balancing algorithms are mainly classified into two types: static algorithms and dynamic algorithms. The first discussion is about static load-balancing algorithms that have been developed for Cloud Computing. Then, the dynamic load-balancing algorithms are discussed

3.1. Static Load Balancing Algorithms

Static Load balancing algorithms assign the tasks to the nodes based only on the ability of the node to process new requests. The process is based solely on prior knowledge of the nodes' properties and capabilities. These would include the node's processing power, memory and storage capacity, and most recent known communication performance. Although they may include knowledge of the communication prior performance, static algorithms generally do not consider dynamic changes of these attributes at run-time. In addition, these algorithms cannot adapt to load changes during run-time.

The proposed algorithm uses the ants' behaviour to gather information about the cloud nodes to assign the task to a specific node. However, the algorithm in [2] has the ant's synchronization issue and the author

in [1] is trying to solve this by adding the feature 'suicide' to the ants. Both algorithms work in the following way, once a request is initiated the ants and pheromone are initiated and the ants start their forward path from the 'head' node. A forward movement means that the ant is moving from one overloaded node looking for the next node to check if it is overloaded or not.

Moreover, if the ant finds an under loaded node, it will continue its forward path to check the next node. If the next node is an overloaded node, the ant will use the backward movement to get to the previous node. The addition in the algorithm proposed in [1] is that the ant will commit suicide once it finds the target node, which will prevent unnecessary backward movements.

Junjie proposed a load balancing algorithm [5] for the private Cloud using virtual machine to physical machine mapping. The architecture of the algorithm contains a central scheduling controller and a resource monitor. The scheduling controller does all the work for calculating which resource is able to take the task and then assigning the task to that specific resource. However, the resource monitor does the job of collecting the details about the resources availability. The process of mapping tasks goes through four main phases which are: accepting the virtual machine request, then getting the resources details using the resource monitor. After that, the controller calculates the resources ability to handle tasks and the resource that gets the highest score is the one receiving the task. Finally, the client will be able to access the application.

The algorithm in [3] is an addition to the Map Reduce algorithm [4]. Map Reduce is a model which has two main tasks: It Maps tasks and Reduces tasks results. Moreover, there are three methods in this model. The three methods are part, comp and group. Map Reduce first executes the part method to initiate the Mapping of tasks. At this step the request entity is partitioned into parts using the Map tasks. Then, the key of each part is saved into a hash key table and the comp method does the comparison between the parts. After that, the group method groups the parts of similar entities using the Reduce tasks.

3.2. Dynamic Load Balancing Algorithms

Dynamic load balancing algorithms take into account the different attributes of the nodes' capabilities and network bandwidth. Most of these algorithms rely on a combination of knowledge based on prior gathered information about the nodes and run-time properties collected as the selected nodes process the tasks. These algorithms assign and reassign tasks to the nodes based on the attributes gathered and calculated. Such algorithms require constant monitoring of the nodes and task progress and are usually harder to implement. However, they are more accurate and could result in more efficient load balancing.

Ren [6] presented a dynamic load balancing algorithm for cloud computing based on an existing algorithm called WLC [7] (weighted least connection). The WLC algorithm assigns tasks to the node based on the number of connections that exist for that node. This is done based on a comparison of the SUM of connections of each node in the Cloud and then the task is assigned to the node with least number of connections. However, WLC does not take into consideration the capabilities of each node such as processing speed, storage capacity and bandwidth.

The proposed algorithm is called ESWLC (Exponential Smooth Forecast based on Weighted Least Connection). ESWLC improves WLC by taking into account the time series and trials. That is ESWLC builds the conclusion of assigning a certain task to a node after having a number of tasks assigned to that node and getting to know the node capabilities. ESWLC builds the decision based on the experience of the node's CPU power, memory, number of connections and the amount of disk space currently being used. ESWLC then predicts which node is to be selected based on exponential smoothing.

The algorithm proposed in [8] is a dual direction download-ing algorithm from FTP servers (DDFTP). The algorithm presented can be also implemented for Cloud Computing load balancing. DDFTP works by splitting a file of size m into m/2 partitions. Then, each server node starts processing the task assigned for it based on a certain pattern. For example, one server will start from block 0 and keeps downloading incrementally while another server starts from block m and keeps downloading in a decrementing order. As a result, both servers will work independently,

but will end up downloading the whole file to the client in the best possible time given the performance and properties of both servers.

Thus, when the two servers download two consecutive blocks, the task is considered as finished and other tasks can be assigned to the servers. The algorithm reduces the network communication needed between the client and nodes and therefore reduces the network overhead. Moreover, no run-time monitoring of the nodes is required, while attributes such as network load, node load, network speed are automatically taken into consideration.

The algorithm proposed in [9] first present a load-balancing algorithm. In this algorithm each node has its own global knowledge about the system, which will make the movement cost very low and convergence very fast. We then extend this algorithm for the situation where without degrading the performance of the algorithm we cannot get the global knowledge. Based on the global knowledge, if node j finds that it is the least-loaded node in the system, j leaves the system by migrating its locally hosted chunks to its successor j+1 and then regions the system as the successor of the heaviest node k.

To immediately relieve node k's load, node j requests min Lk-A, A Ag chunks from k. That is, node j requests A chunks from the heaviest node k if k's load is greater than 2A; otherwise, j requests a load of Lk -A from k to relieve k's load. Node k may still remain as the heaviest node in the system after it has migrated its load to node j. In this case, the current least-loaded node, say node j', departs and then rejoins the system as k's successor. That is, j' becomes node k + 1, and k's original successor i thus becomes node k + 2. Such a process repeats iteratively until k is not at all the heaviest node. Then, this process is done again and again to release the extra load on the other heaviest nodes also in the system. This is repeated until all the heaviest nodes become lighter.

4. Discussions and Comparison

In this section we discuss the different algorithms that were discussed in Section 3. We also compare these algorithms based on the challenges discussed in Section 2. As discussed earlier, the different approaches offer specific solutions for load balancing that suit some situations but not others. The static algorithms are usually very efficient in terms of overhead as they do not need to monitor the resources during run-time. Therefore, they would work very well in a stable environment where operational properties do not change over time and loads are generally uniform and constant. The dynamic algorithms on the other hand offer a much better solution that could adjust the load dynamically at run-time based on the observed properties of the resources at run time. However, this feature leads to high overhead on the system as constant monitoring and control will add more traffic and may cause more delays. Some newly proposed dynamic load balancing algorithms tries to avoid this overhead by utilizing novel task distribution models.

For example As for the Ant Colony approach, we can see that the decentralized approach provides a good solution to the single point of failure issue. However, it could easily cause a network overload due to the large number of dispatched ants. In addition, several operational factors are not being considered which may result in poor performance. This algorithm can be further improved by introducing better evaluation mechanisms that take into consideration the status of the node and its current available resources. In addition, it may also be possible to limit the number of ants being used in the discovery process by introducing search controls that could reduce the branching levels required in the search.

In DDFTP, the control is kept to a minimum and no run-time monitoring is needed to keep up with environment changes, while keeping a very efficient load balancing. As a result, it provides a good approach, yet it still needs some improvements for better utilization of the available resources. One possibility is to find a good model that will reduce the level of replication needed, while maintaining the same level of performance. This may be possible with the consideration of partial replications with a certain level of overlap that will enable more efficient resource utilization and maintain minimum overhead for load balancing.

Table 1 gives a comparison among the reviewed algorithms. The comparison shows the positives and negative points of each algorithm. Table 2 illustrates a comparison between the reviewed algorithms in terms of the challenges discussed in Section 2.

ISSN:2229-6093

Table 1: Pros and Cons of load balancing algorithms [11]

	pros	cons
ANT COLONY	• Best case scenario is that the under loaded node is found at beginning of the search • Decentralized, no single point of failure • Ants can collect the information faster	• Network overhead because of the large number of ants • Points of initiation of ants and number of ants are not clear • Nodes status change after ants visits to them is not taken into account • Only availability of node is being considered, while there are other factors that should be taken in to consideration.
Enhanced MapReduce	• Less overhead for the reduce tasks	• High processing time • Reduce tasks capabilities are not taken into consideration
VM Mapping	• Reliable calculation method	• Single Point of failure • Does not take into account network load, and node capabilities
DDFTP	• Fast • Reliable download of files	• Full replication of data files that requires high storage in all nodes.
WLC	• More accurate results than WLC	• Complicated • Prediction algorithm requires existing data and has long processing time
Load Rebalancing for Distributed File Systems in Clouds	• Fast • Reliable calculation method • Decentralized, no single point of failure	• High processing time • complex

Table 2: Comparison of load balancing algorithms [11]

	Replication	Speed	Heterogeneity	SPOF	Network Overhead	Spatially Distributed	Implementation Complexity	Fault Tolerance
Ants Colony	Full	Fast	No	No	Yes	No	No	Yes
Mapreduce	Full	Slow	Yes	No	Yes	Yes	High	Yes
VM Mapping	Full	Fast	Yes	Yes	Yes	No	High	Yes
DDFTP	Full	Fast	Yes	No	No	Yes	Low	Yes
WLC	Full	Fast	Yes	No	Yes	Yes	High	Yes
Load rebalancing DFS in Cloud	Full	Fast	Yes	No	No	Yes	High	Yes

5. Conclusion and Future work

In this paper we surveyed multiple algorithms for load balancing for Cloud Computing. We discussed the challenges that must be addressed to provide the most suitable and efficient load balancing algorithms. We also discussed the advantages and disadvantages of these algorithms. Then, we compared the existing algorithms based on the challenges we discussed. Our research on Load Rebalancing for Distributed File Systems in Clouds concentrates on efficient load balancing and provides us with the basis to further improve it and reach more efficient load balancing and better resource utilization.

The current design of Load Rebalancing for Distributed File Systems in Clouds can tolerate high delays, handle heterogeneous resources, efficiently adjust to dynamic operational conditions, offer efficient task distribution, and provide minimum node idle time. However, it relies on full replication of the files on multiple sites, which wastes storage resources. Therefore, as our future work, we are planning to improve Load Rebalancing for Distributed File Systems in Clouds to make it more suitable for Cloud environments and more efficient in terms of storage utilization.

6. References

[1] Nishant, K. P. Sharma, V. Krishna, C. Gupta, KP. Singh, N. Nitin and R. Rastogi,"Balancing of Nodes in Cloud Using Ant Colony Optimization." In proc. 14th International Conference on Computer Modelling and Simulation (UKSim), IEEE, pp: 3-8, March 2012.

[2] Zhang, Z. and X. Zhang, "A load balancing mechanism based on ant colony and complex network theory in open cloud computing federation." In proc. 2nd International Conference on. Industrial Mechatronics and Automation (ICIMA), IEEE, Vol. 2, pp:240-243, May 2010.

[3] Kolb, L., A. Thor, and E. Rahm, E, "Load Balancing for MapReducebased Entity Resolution," in proc. 28th Interna-tional Conference on Data Engineering (ICDE), IEEE, pp: 618-629, 2012.

[4] Gunarathne, T., T-L. Wu, J. Qiu and G. Fox, "MapReduce in the Clouds for Science," in proc. 2nd International Conference on Cloud Computing Technology and Science (CloudCom), IEEE, pp:565-572, November/December 2010.

[5] Ni, J., Y. Huang, Z. Luan, J. Zhang and D. Qian, "Virtual machine mapping policy based on load balancing in private cloud environment,"in proc. International Conference on Cloud and Service Computing (CSC), IEEE, pp: 292-295, December 2011.

[6] Ren, X., R. Lin and H. Zou, "A dynamic load balancing strategy for cloud computing platform based on exponential smoothing forecast" in proc. International Conference on. Cloud Computing and Intelligent Systems (CCIS), IEEE, pp: 220-224, September 2011.

[7] G. Holmberg and M. Torrens. Musicstrands: A platform for discovering and exploring music.University of Michigan Li-brary, 2005.

[8] Lee, R. and B. Jeng, "Load-balancing tactics in cloud," in proc. International Conference on Cyber-Enabled Distributed Com-puting and Knowledge Discovery (CyberC), IEEE, pp:447-454, October 2011.

[9] Al-Jaroodi, J. and N. Mohamed. "DDFTP: Dual-Direction FTP," in proc. 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), IEEE, pp:504-503, May 2011.

[10]Hung-Chang Hsiao, Hsueh-Yi Chung, Haiying Shen, Yu-Chang Chao "Load Rebalancing for Distributed File Systems in Clouds". IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 24, NO. 5, IEEE pp. 951-962 MAY 2013.