A Survey of Load Balancing in Cloud Computing: Challenges and Algorithms

Klaithem Al Nuaimi, Nader Mohamed, Mariam Al Nuaimi and Jameela Al-Jaroodi
College of Information Technology, UAEU
Al Ain, United Arab Emirates
{klaithem.alnuaimi, nader.m, mariam.alnuaimi}@uaeu.ac.ae, jaljaroodi@gmail.com

***Abstract*—Load Balancing is essential for efficient operations in distributed environments. As Cloud Computing is growing rapidly and clients are demanding more services and better results, load balancing for the Cloud has become a very interesting and important research area. Many algorithms were suggested to provide efficient mechanisms and algorithms for assigning the client's requests to available Cloud nodes. These approaches aim to enhance the overall performance of the Cloud and provide the user more satisfying and efficient services. In this paper, we investigate the different algorithms proposed to resolve the issue of load balancing and task scheduling in Cloud Computing. We discuss and compare these algorithms to provide an overview of the latest approaches in the field.**

Keywords*—*Cloud Computing, Load Balancing, Task Scheduling, Cloud Storage, Replications.

I. INTRODUCTION

Cloud Computing became very popular in the last few years. As part of its services, it provides a flexible and easy way to keep and retrieve data and files. Especially for making large data sets and files available for the spreading number of users around the world. Handling such large data sets require several techniques to optimize and streamline operations and provide satisfactory levels of performance for the users. Therefore, it is important to research some areas in the Cloud to improve the storage utilization and the download performance for the users. One important issue associated with this field is dynamic load balancing or task scheduling. Load balancing algorithms were investigated heavily in various environments; however, with Cloud environments, some additional challenges are present and must be addressed. In Cloud Computing the main concerns involve efficiently assigning tasks to the Cloud nodes such that the effort and request processing is done as efficiently as possible [1], while being able to tolerate the various affecting constraints such as heterogeneity and high communication delays.

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 [2]. 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.

In this paper we present a survey of the current load balancing algorithms developed specifically to suit the Cloud Computing environments. We provide an overview of these algorithms and discuss their properties. In addition, we compare these algorithms based on the following properties: the number of attributes taken into consideration, the overall network load, and time series.

The rest of this 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.

II. CHALLENGES IN CLOUD COMPUTING LOAD BALANCING

Before we could review the current load balancing approaches for Cloud Computing, we need to identify the main issues and challenges involved and 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 in the following points.

A. Spatial Distribution of the Cloud Nodes

Some algorithms are designed to be efficient only for an intranet or closely located nodes where communication delays are negligible. However, it is a challenge to design a load balancing algorithm that can work for spatially distributed nodes. This is because other factors must be taken into account such as the speed of the network links among the nodes, the

978-0-7695-4943-9/12 $26.00 © 2012 IEEE
DOI 10.1109/NCCA.2012.29

distance between the client and the task processing nodes, and the distances between the nodes involved in providing the service. There is a need to develop a way to control load balancing mechanism among all the spatial distributed nodes while being able to effectively tolerate high delays [3].

B. Storage/ Replication

A full replication algorithm does not take efficient storage utilization into account. This is because the same data will be stored in all replication nodes. Full replication algorithms impose higher costs since more storage is needed. However, partial replication algorithms could save parts of the data sets in each node (with a certain level of overlap) based on each node's capabilities such as processing power and capacity [4]. This could lead to better utilization, yet it increases the complexity of the load balancing algorithms as they attempt to take into account the availability of the data set's parts across the different Cloud nodes.

C. Algorithm Complexity

Load balancing algorithms are preferred to be less complex in terms of implementation and operations. The higher implementation complexity would lead to a more complex process which could cause some negative performance issues. Furthermore, when the algorithms require more information and higher communication for monitoring and control, delays would cause more problems and the efficiency will drop. Therefore, load balancing algorithms must be designed in the simplest possible forms [5].

D. Point of 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 (centralized 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 [6]. 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.

III. Load Balancing Algorithms Review

In this section we discuss the most known contributions in the literature for load balancing in Cloud Computing. We classify the load balancing algorithms into two types: static algorithms and dynamic algorithms. We first discuss the static load-balancing algorithms that have been developed for Cloud Computing. Then, we will discuss the dynamic load-balancing algorithms.

A. 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.

Radojevic suggested an algorithm called CLBDM [7] (Central Load Balancing Decision Model). CLBDM is an improvement of the Round Robin Algorithm which is based on session switching at the application layer. Round Robin [8] is a very famous load balancing algorithm. However, it sends the requests to the node with the least number of connections. The improvement done in CLBDM is that the connection time between the client and the node in the cloud is calculated, and if that connection time exceeds a threshold then there is an issue. If an issue is found, the connection will be terminated and the task will be forwarded to another node using the regular Round Robin rules. CLBDM acts as an automated administrator. The idea was inspired by the human administrator point of view.

The proposed algorithm by Kumar [9] is an improvement version of the algorithm presented in [10]. Both algorithms are using the ants' behavior to gather information about the cloud nodes to assign the task to a specific node. However, the algorithm in [10] has the ants synchronization issue and the author in [9] 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 underloaded 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 [9] is that the ant will commit suicide once it finds the target node, which will prevent unnecessary backward movements.

The algorithm proposed in [11] is an addition to the MapReduce algorithm [12]. MapReduce 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*. MapReduce 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. Since several Map tasks can read entities in parallel and process them, this will cause the Reduce

tasks to be overloaded. Therefore, it is proposed in this paper to add one more load balancing level between the Map task and the Reduce task to decrease the overload on these tasks. The load balancing in the middle divides only the large tasks into smaller tasks and then the smaller blocks are sent to the Reduce tasks based on their availability.

Junjie proposed a load balancing algorithm [13] 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.

B. 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 in the Cloud and run-time properties collected as the selected nodes process the task's components. These algorithms assign the tasks and may dynamically reassign them 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.

In [14], the goal is to find an algorithm to minimize the data duplication and redundancy. The algorithm proposed is called INS (Index Name Server) and it integrates de-duplication and access point selection optimization. There are many parameters involved in the process of calculating the optimum selection point. Some of these parameters are the Hash code of the block of data to be downloaded, the position of the server that has the target block of data, the transition quality which is calculated based on the node performance and a weight judgment chart, the maximum bandwidth of downloading from the target server and the path parameter. Another calculation is used to find out whether the connection can handle additional nodes or not (busy level). They classified the busy levels into three main categories B(a), B(b) and B(c). B(a) means that the connection is very busy and cannot handle any additional connections. B(b) means the connection is not busy and additional connections can be added. However, B(c) means that the connection is limited and further study needs to be done to know more about the connection. B(b) is also classified into three categories: B(b1) which means that INS must analyze and establish a backup, B(b2) which means the INS must send the requests to the backup nodes and B(b3) which is the highest level of efficiency required and it means that INS must reanalyze and establish new backups.

Ren [15] presented a dynamic load balancing algorithm for cloud computing based on an existing algorithm called WLC [16] (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 [17] is a dual direction downloading 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 decremental 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, attributes such as network load, node load, network speed are automatically taken into consideration, while no run-time monitoring of the nodes is required.

The paper in [18] proposes an algorithm called Load Balancing Min-Min (LBMM). LBMM has a three level load balancing framework. It uses the Opportunistic Load Balancing algorithm (OLB) [19]. OLB is a static load balancing algorithm that has the goal of keeping each node in the cloud busy. However, OLB does not consider the execution time of the node. This might cause the tasks to be processed in a slower manner and will cause some bottlenecks since requests might be pending waiting for nodes to be free. LBMM improves OLB by adding a three layered architecture to the algorithm. The first level of the LBMM architecture is the request manager which is responsible for receiving the task

and assigning it to one service manager in the second level of LBMM. When the service manager receives the request, it divides it into subtasks to speed up processing that request. A service manager would also assign the subtask to a service node which is responsible for executing the task. The service manager assigns the tasks to the service node based on different attributes such as the remaining CPU space (node availability), remaining memory and the transmission rate.

IV. Discussion and Comparison

In this section we discuss the different algorithms that were discussed in Section III. We also compare these algorithms based on the challenges discussed in Section II.

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 synamic 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.

Table I shows a comparison among the reviewed algorithms. The comparison shows the positives and negative points of each algorithm. For example, the INS algorithm is able to handle the load balancing dynamically. However, the provided algorithm is complicated which could cause high implementation complexity. We foresee that a close examination of the algorithm and changing the overall structure may result in a less complex algorithm. Furthermore, the CLDBM algorithm solves the problem of having a human administrator needed all the time to control the system. Therefore, it provides a centralized controller. However, if the centralized controller fails any time the whole system will not be able to operate which will cause a system failure. Having a backup of the central controller could solve the issue for CLDBM in cases of failure. 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 II illustrates a comparison between the reviewed algorithms in terms of the challenges discussed in Section II. For example, the only algorithm that avoids data redundancy and storage replication is the INS algorithm. However, INS is a centralized algorithm and therefore has a single point of failure. Moreover, it is a complex algorithm. On the other hand, DDFTP relies on replicated resources and does not reduce the storage size required but it has a dynamic decentralized approach to balance the loads. It is also a much simpler algorithm to download stored data. DDFTP can be improved to use less storage by applying partial replication. Generally, each algorithm satisfies a partial set of these challenges, which makes it suitable for specific situation that match the addressed challenges. For example INS, CLBDM and VM Mapping all have a single point of failure, thus they would function very well in a very stable environment where the resources reliability is very high. Moreover, all algorithms except for the Ants Colony and VM Mapping can handle highly distributed environment. Therefore, they are more suitable for the public Cloud than the other two. In addition, all but DDFTP introduce high overhead on the network. As a result, if the network conditions worsen, they would all suffer significantly as more delays will be involved which will delay the overall load balancing process. However, DDFTP would be more capable in handling such delay as it does not need to rely on run-time monitoring and controls.

V. 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 DDFTP [20] 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 DDFTP 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 DDFTP to make it more suitable for Cloud environments and more efficient in terms of storage utilization.

TABLE I. PROS AND CONS OF LOAD BALANCING ALGORITHMS

	Pros	***Cons***
INS	•Initially proved to handle some sort of dynamic load balancing	•No forecasting algorithm to identify the future behavior of the nodes. •Complicated in terms of implementation. •Only certain parameters are considered such as distance and time
ESWLC	•More accurate results than WLC	•Complicated •Prediction algorithm requires existing data and has long processing time
CLDBM	•Solves issues of Round Robin Algorithm •Automated tasks forwarding reduces the need for a human administrator	•Inherits Round Robin issues such as not taking into consideration node Capabilities •Single point of failure (if CLBDM fails, the whole process fails) •The threshold might not be applied to all cases.
ANT COLONY	•Best case scenario is that the underloaded 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 into 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.
LBMM	•Reliable tasks assignment to nodes	•Slower than other algorithms because Work must pass through three layers to be processed.

TABLE II. COMPARISON OF LOAD BALANCING ALGORITHMS

	Replication	Speed	Heterogeneity	SPOF	Network Overhead	Spatially Distributed	Implementation Complexity	Fault Tolerance
INS	Partial	Moderate	YES	YES	YES	YES	HIGH	NO
ESWLC	Full	FAST	YES	NO	YES	YES	HIGH	YES
CLBDM	Full	SLOW	YES	YES	YES	YES	LOW	NO
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

REFERENCES

[1] Randles, M., D. Lamb and A. Taleb-Bendiab, "A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing," in Proc. IEEE 24th International Conference on Advanced Information Networking and Applications Workshops (WAINA), Perth, Australia, April 2010.

[2] Rimal, B. Prasad, E. Choi and I. Lumb, "A taxonomy and survey of cloud computing systems." In proc. 5th International Joint Conference on INC, IMS and IDC, IEEE, 2009.

[3] Buyya R., R. Ranjan and RN. Calheiros, "InterCloud: Utility-oriented federation of cloud computing environments for scaling of application services," in proc. 10th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP), Busan, South Korea, 2010.

[4] Foster, I., Y. Zhao, I. Raicu and S. Lu, "Cloud Computing and Grid Computing 360-degree compared," in proc. Grid Computing Environments Workshop, pp: 99-106, 2008.

[5] Grosu, D., A.T. Chronopoulos and M. Leung, "Cooperative load balancing in distributed systems," in Concurrency and Computation: Practice and Experience, Vol. 20, No. 16, pp: 1953-1976, 2008.

[6] Ranjan, R., L. Zhao, X. Wu, A. Liu, A. Quiroz and M. Parashar, "Peer-to-peer cloud provisioning: Service discovery and load-balancing," in Cloud Computing - Principles, Systems and Applications, pp: 195-217, 2010.

[7] Radojevic, B. and M. Zagar, "Analysis of issues with load balancing algorithms in hosted (cloud) environments." In proc.34th International Convention on MIPRO, IEEE, 2011.

[8] Sotomayor, B., RS. Montero, IM. Llorente, and I. Foster, "Virtual infrastructure management in private and hybrid clouds," in IEEE Internet Computing, Vol. 13, No. 5, pp: 14-22, 2009.

[9] Nishant, K. P. Sharma, V. Krishna, C. Gupta, KP. Singh, N. Nitin and R. Rastogi, "Load 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.

[10] 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.

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

[12] 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.

[13] 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.

[14] , T-Y., W-T. Lee, Y-S. Lin, Y-S. Lin, H-L. Chan and J-S. Huang, "Dynamic load balancing mechanism based on cloud storage" in proc. Computing, Communications and Applications Conference (ComComAp), IEEE, pp:102-106, January 2012.

[15] 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.

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

[17] 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.

[18] Wang, S-C., K-Q. Yan, W-P. Liao and S-S. Wang, "Towards a load balancing in a three-level cloud computing network," in proc. 3rd International Conference on. Computer Science and Information Technology (ICCSIT), IEEE, Vol. 1, pp:108-113, July 2010.

[19] Sang, A., X. Wang, M. Madihian and RD. Gitlin, "Coordinated load balancing, handoff/cell-site selection, and scheduling in multi-cell packet data systems," in Wireless Networks, Vol. 14, No. 1, pp: 103-120, January 2008.

[20] Mohamed, N. and J. Al-Jaroodi, "Delay-tolerant dynamic load balancing," in proc. 13th International Conference on High Performance Computing and Communications (HPCC), pp:237-245, September 2011.