An Implementation of Cloud Load Balancing Algorithm and Comparison with ACO Load Balancing Algorithm

Zeba Qureshi[1], Dr. Abhay Kothari[2],

[1]Ph.D. Scholar, SAGE University, Indore, , MP, India, 452020

[2] Professor, SIRT, SAGE University, Indore, MP, India, 452020

Abstract: Cloud computing has made a remarkable impact on the world. It has lot of advantages and that is why many companies are shifting to cloud computing paradigm. With cloud computing any one can easily access IT resources over the internet without the hassle of maintaining the whole system. As the cloud services are easily available the more people are getting associated with the cloud vendors for their needs of IT resources. This is eventually increasing burden on cloud service vendorsbecause the load is increasing day by day. The cloud service providers can ensure good performance to the customers only when they have good load balancing strategies. In the cloud environment which is a highly dynamic environment the load cannot be predicted beforehand and this leads to an unbalanced system where some nodes are overloaded whereas others are underloaded. The overloaded and underloaded nodes create issues in the system and this problem needs to be addressed with a good load balancing policy. In our work, we have proposed a Dynamic cloud load balancing algorithm and compared it with Ant Colony load balancing algorithm in terms of response time. It is found that the proposed Dynamic load balancing algorithm is giving better response time than ACO load balancing algorithm. The response time is a crucial parameter of performance. Cloud Analyst tool is used for the simulation and analysis purpose.

Keywords: Cloud Computing, Load Balancing, Response Time, Dynamic load balancing algorithm, ACO

1. Introduction

Load balancing is a crucial issue in Cloud computing that must be handled because it has a direct impact on resource utilization and customer satisfaction [1]. For the purpose of ensuring that no single node is overworked and that other nodes are not left idle, load balancing distributes the dynamic local workload evenly among all of the nodes in the overall Cloud environment. This strategy, by improving overall system performance and resource utilization, contributes to improved levels of user satisfaction as well as more efficient use of system resources [2]. The distributed system also makes certain that each and every computing resource inside it is allocated in an equitable and effective manner [3]. Many different types of load balancing strategies are implemented in order to minimize system bottlenecks caused by an imbalanced load. While load balancing can assist prevent a service from going down completely, it cannot prevent a component from failing completely. An additional advantage is that it optimizes the flow while simultaneously minimizing reaction times and preventing the system from getting overloaded. In addition to helping businesses save money [3], load balancing systems can also help them reduce their environmental impact.

2. Related Work

Radojevic and Zagar[4] proposed a Central Load Balancing Decision Model, which is a variant of the Round-Robin algorithm that is more efficient (CLBDM). The suggested algorithm made use of the round robin principle as its foundation, while it also monitored the duration of communication between the server and the client by calculating the total time it took to complete a job on a certain Cloud resource. The findings indicate that the new algorithm incorporates information from the end user experience and virtualized computer settings, and that it can either proactively influence load balancing decisions or reactively adjust decisions when dealing with crucial situations. However, the proposed method incorporated additional aspects that could jeopardise the overall stability of the computer system, despite the fact that the proposed model resolves many of the concerns that can arise with today's accessible models. It is probable that the introduction of CLBDM as a central management module may have an impact on both the load balancing decisions and the virtual server resources, which will result in a single point of failure in some cases. Furthermore, in the lack of suitable design and engineering in the characteristics such as dependability, resilience, and robustness, CLBDM might result in severe performance issues or even the complete system's inability to function. One more issue that has contributed to the unreliability of CLBDM is that it has a tendency to get stuck in unexpected loops and to flap its choice between nodes, leading in poor performance and end-user experience.

Shah and Farik[5] suggested a hybrid static approach that combined the weighted round robin technique with the maximum-minimum load balancing algorithm to achieve better performance. The capacity of the server's resources was calculated in this study by utilising the weight assignment function of the weighted round robin algorithm before the load assignment was completed. Furthermore, this is paired with the properties of the max-min method, which calculates the minimum and maximum execution times, respectively. As a result, the maximum time value that has been determined is utilized to allocate jobs to the appropriate machines. Because it is known before task scheduling is completed, this strategy can make use of the greatest amount of time for job completion and scheduling the more time-consuming tasks. Furthermore, because all work will be distributed in a round robin format, the strategy is more effective at combating famine than other methods. However, a significant disadvantage of this method was that it was incompatible with complicated infrastructures and constantly changing user requirements.

Nayak and Patel [6] did an analytical investigation on the existing throttled algorithms and their performance in Cloud networking in order to find a means of balancing the load on the cloud infrastructure. For the purpose of balancing the load, the researchers presented a modified throttled algorithm. The findings demonstrated that the proposed improved throttle algorithm performed well despite the fact that the underlying capacity of one VM differed significantly from the other. When making the decision on which VM to utilize, this updated proposed algorithm took into account a greater number of criteria, such as the expected response time and the loading state. The CPU utilization of the virtual machine was used to compute the estimated response time. The results of the analysis revealed that the use of a more efficient throttled load balancing algorithm with less overhead resulted in an increase in the number of user requests handled and better VM allocation, which in turn will result in a reduction in the number of requests denied when they arrive at the Cloud's data center. New static load balancing, presented by Raj [7], is a modified version of the original static load balancing. Findings from the research conducted by Patel and Shah [8] demonstrated that the application of the proposed throttling algorithm reduces response time, datacenter request servicing time, and cost. While conducting an in-depth comparative and analytical research of the existing and suggested throttle algorithms, it was discovered that in order to assess the availability of virtual machines for exact decision making, more load balancing factors should be included. Furthermore, the proposed algorithm's implementation in a variety of various hardware configurations must be tested as well.

For distributing incoming jobs equitably among the servers or virtual machines, Domanlal and Reddy [9] introduced a locally optimized load balancing solution named Equally Spread Current Execution algorithm (ESCEL) algorithm, which they called the Equally Spread Current Execution algorithm. The technique necessitates the use of a load balancer that can keep track of the jobs that are possible. The basic role of a load balancer is to place jobs in a job pool and assign them to discrete virtual machines (VMs) in a cluster. The new jobs are assigned to distinct virtual machines (VMs) once they have been monitored on a regular basis by the balancer in the work queue. Because of the way the load balancer handles the list of tasks that are given to virtual servers, it is able to identify any free VMs that need to be assigned to new tasks immediately.

For Cloud environments, Singh et al. [10] proposed an Autonomous Agent Based Load Balancing (A2LB) algorithm that provided dynamic load balancing through the use of autonomous agents. The proposed method, which is already in place, ensures an acceptable outcome for all parties involved. The proposed mechanism is based on the performance of two independent applications using two alternative techniques to conveyance, each of which has its own set of characteristics. As illustrated by the experimental results, a small number of apps may be used to modify the distribution of load among machines as a result of this. As a result, in large-scale simulations, it is possible to anticipate better performance when using a distribution strategy.

3. Problem Definition

The cloud computing is an enormous appropriated framework which uses dispersed assets to convey a support of the end clients. Henceforth giving adequate response time to end clients, introduces a significant obstacle in achievement of distributed computing. This can be taken care of through an appropriate load balancing algorithm. This will build the accessibility and will improve the customer satisfaction.

4. Methodology

In this particular section we have proposed a new Dynamic load balancing algorithm.

Input: a set ofdata centers requests and virtual machines

Output:Requests from data centers are allocated to virtual machines

Process:

1. The algorithm maintains a list of all the virtual machines available with their Status(Status will be either BUSY or AVAILABLE), Minimum Response Time and Throughput.
2. If a request is received by the data center controller, the data center controller forwards the request to dynamic load balancing algorithms. Dynamic load balancing algorithm sorts the available virtual machines on the basis of minimum response time and throughput.

For each user request repeat step 3 to 5

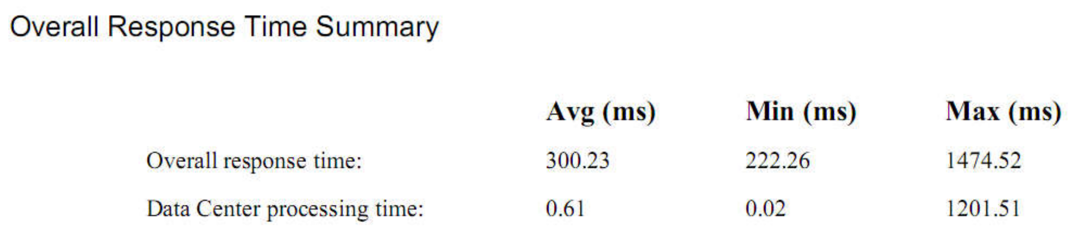
3. If more than one Virtual Machine AVAILABLE, Then Allocate the VM with maximum throughput to user request. If single Virtual Machine AVAILABLE then allocate machine to user request.
    - Then the algorithm sends the VM id of that machine to the data centre controller
    - Data centre controller sends a request to that virtual machine
    - Data centre controller sends a notification of this new allocation to the dynamic load balancing algorithm
    - The algorithm updates the list of virtual machines accordingly.
4. If VM is not available
    - If the priority of the new request is greater than the priority of the executing request, then the executing request is switched by the new request and placed in Queue. Virtual machine with highest possible throughput is allocated to higher priority request.
    - Else request will be placed in waiting queue
5. When the virtual machine finishes the request.

- The data centre controller sends a notification to dynamic algorithm that the vm id has finished the request.
- The algorithm updates the list of virtual machines accordingly.

4. Result Analysis

The Cloud Analyst simulation tool is used for the experimental work.This particular section highlights the results of proposed Dynamic load balancing algorithm and ACO load balancing algorithm.To configure the simulation and carry out the experimental work on cloud analyst we have used 3 data centers, 30 User bases and 20 virtual machines.

The results of ACO and Dynamic load balancing algorithms in terms of response time are shown below:

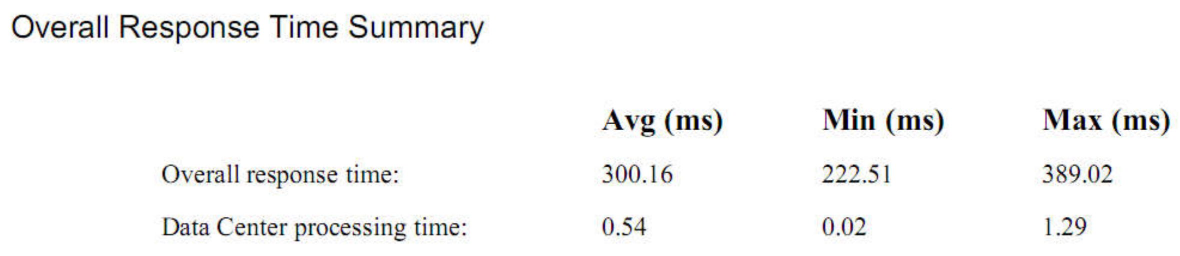


Overall Response Time Summary

	Avg (ms)	Min (ms)	Max (ms)
Overall response time:	300.23	222.26	1474.52
Data Center processing time:	0.61	0.02	1201.51

Figure 1: Result Summary of Ant colony Optimization

The results of Dynamic Proposed algorithm are shown below in figure 5



Overall Response Time Summary

	Avg (ms)	Min (ms)	Max (ms)
Overall response time:	300.16	222.51	389.02
Data Center processing time:	0.54	0.02	1.29

Figure 2: Result Summary of Dynamic ProposedAlgorithm

The result summary of both the algorithms can be seen in the table below:

Algorith m	Overall Response Time Avg (ms)	Overall Response Time Min (ms)	Overall Response Time Max (ms)
Ant Colony	300.23	222.26	1474.52
Dynamic	300.16	222.51	389.02

Table 1: Comparative Results of Dynamic & ACO Load Balancing Algorithms

The results above shows that the Dynamic load balancing algorithm is performing better in terms of response time as compared to ACO load balancing algorithm.

5. Conclusion

Cloud Computing provides IT resources via internet and the basis of pricing is pay as you go which means you have to pay according to the usage of resources. The less resources you use, the less you pay and the more resources you use, the more you pay. This is just like we use electricity at our homes. As cloud computing has many benefits, more and more customers are availing the cloud services. With the greater number of customers, the greater is the strain on cloud service providers. To cope up with this load on cloud environment it is very important to incorporate efficient load balancing policies. In this particular work we had proposed a new Dynamic load balancing algorithm for cloud environment and compared it with ACO load balancing algorithm. The experimental study is carried out on Cloud Analyst tool. In this study we found that the proposed Dynamic load balancing algorithmis performing better in terms of response time.

References

[1]Zhang,Z.,&Zhang,X.(2010,May).Aloadbalancingmechanismbasedonantcolony and complexnetwork theory in opencloud computing federation. *In Proceeding ofIndustrialMechatronics and Automation(ICIMA), 20102ndInternationalConferenceon* (Vol.2,pp.240-243).IEEE.

[2]Sreenivas,V.,Prathap,M.,&Kemal,M.(2014,February).Loadbalancingtechniques: MajorchallengeinCloudComputing-asystematicreview.*InProceedingofElectronics and Communication Systems (ICECS), 2014 International Conference on* (pp. 1-6). IEEE.

[3]Alakeel, A. M. (2010). Aguideto dynamic load balancing indistributed computer systems. *InternationalJournalofComputerScienceandInformationSecurity,10*(6),153-160.

[4]Radojevic,B.&Zagar,M.(2011).Analysisofissueswithloadbalancingalgorithmsin hosted (cloud) environments. *In proceedings of 34th International Convention on MIPRO,IEEE.*

[5]Shah,N.,&Farik,M.(2015).Staticloadbalancingalgorithmsincloudcomputing: Challenges&solutions. *International Journal Of Scientific&TechnologyResearch,4*(10), 365-367.

[6] Nayak,S.,&Patel,(2015)M.P.ASurveyonLoadBalancing AlgorithmsinCloud Computingand ProposedamodelwithImprovedThrottled Algorithm. *International Journalfor ScientificResearch&Development,3*(1).|

[7]Raj, A (2015). A New Static Load Balancing Algorithm in Cloud Computing. *InternationalJournalofComputer Applications,132*(2).

[8]Patel.N,H.,Shah.J(2016)Improved ThrottlingLoadBalancingAlgorithmWithRespect ToComputingCostand ThroughputForCloudBasedRequests,*ijariie,2*(3), 2192-2198.

[9]Domanal,S.G.,&Reddy,G.R.M.(2014,January).Optimalloadbalancingincloud computingbyefficientutilizationofvirtualmachines.*InProceedingofCommunication SystemsandNetworks(COMSNETS),2014SixthInternationalConferenceon*(pp.1-4). IEEE.

[10] Huankai Chen, Professor Frank Wang, Dr. Na Helian and GbolaAkanmu, User-Priority Guided Min-Min Scheduling Algorithm For Load Balancing In Cloud Computing, IEEE, 2013.