Load Balancing in Cloud Computing Using Modified Throttled Algorithm

Shridhar G.Domanal and G.Ram Mohana Reddy
Department of Information Technology
National Institute of Technology Karnataka
Surathkal, Mangalore, India
{shridhar.domanal,profgrmreddy}@gmail.com

Abstract— Load balancing is one of the critical components for efficient operations in the cloud computing environment. In recent years many clients from all over the world are demanding the various services at rapid rate. Many algorithms have been designed to carry out the client's request towards the cloud nodes. Accordingly the cloud computing platform will dynamically configure its servers and these servers may be present physically or virtually in the computing environment. Hence, selecting the virtual machines or servers has to be scheduled properly by using an appropriate load balancing approach. In the present work, a local optimized load balancing approach is proposed for distributing of incoming jobs uniformly among the servers or virtual machines. Further, the performance is analyzed using CloudAnalyst simulator and compared with existing Round Robin and Throttled algorithms. Simulation results have demonstrated that the proposed algorithm has distributed the load uniformly among virtual machines.

Keywords - Load balancing, Cloud Computing, Virtual machine, CloudAnalyst.

I. INTRODUCTION

Recently Cloud Computing has become one of the popular techniques adopted by both industry and academia providing a flexible and efficient way to store and retrieve the data files. However, the main issue is to schedule the incoming requests efficiently with very low response time. Many algorithms like Round Robin, Throttled, Stochastic Hill Climb etc. have been widely used for executing the client's request with a minimal response time [1]. But, constraints such as heterogeneity, reliability and high communication delays need to be addressed while designing efficient scheduling algorithms. Generally load balancing algorithms can be classified into static and dynamic algorithms. Static algorithms are mostly suitable for homogeneous and stable environment but produce poor results whenever attributes are dynamically changing. On the other hand the dynamic algorithms are more flexible and also take different types of system attributes (both prior and during the run-time) into account.

Cloud computing system comprises of several servers, virtual machines, data centers, storage devices etc which are interconnected in an efficient way. Nowadays, computing systems heavily rely on Virtualization technology and thus makes the servers feasible for independent applications. Further, virtualization process improves the power efficiency of the datacenters (consolidation of servers) and thereby enabling the assignment of multiple virtual machines (VMs) to a single physical server [2]. Consequently, some of the servers in the cloud computing system can be turned off (sleep state) and resulting in low power consumption.

In this paper, we present a novel modified Throttled algorithm for assigning all incoming jobs uniformly among the available virtual machines in an efficient way. The response time for serving the incoming jobs is computed and compared with the existing Round Robin and Throttled algorithms.

The rest of the paper is organized as follows: The background and related work is discussed in Section II, in Section III we discuss the proposed algorithm, Section IV deals with an experimental setup, Section V gives the results and analysis; finally the conclusion is given in Section VI.

II. BACKGROUND AND RELATED WORK

In this section, we briefly summarize the load balancing algorithms used in the cloud computing environment. The main focus is on the assignment of all incoming jobs among the available virtual machines with minimal response time. Load balancing is defined as a process of making effective resource utilization by reassigning the total load to the individual nodes of the collective system and thereby minimizing the response time of the job.

Brototi Mondal et al. [3] have developed the Stochastic Hill Climbing algorithm for balancing the load. Stochastic Hill Climbing is one of the incomplete approaches for solving such optimization problems. A stochastic and Local Optimization algorithm is simply a loop that continuously moves in the direction of increasing value, which is uphill. It stops when it reaches the peak value where no neighbor has a higher value. This variant chooses at random from among the uphill moves and the probability of selection can vary with the steepness of the uphill move. Thus it maps assignment of values to a set of other values by making only minor changes to the original value. The best element of the set is made the next assignment. This basic operation is repeated until either a solution is found or a stopping criterion is reached. The results are quite encouraging when compared to Round Robin and FCFS algorithms.

The other load balancing algorithms given by researchers are as follows.

A. Round Robin Algorithm

The algorithm works on random selection of the virtual machines. The datacenter controller assigns the requests to a list of VMs on a rotating basis. The first request is allocated to a VM picked randomly from the group and then the Data Center controller assigns the requests in a circular order. Once the VM is assigned the request, the VM is moved to the end of the list [4]. The major issue in this allocation is this that it does not consider the advanced load balancing requirements such as processing times for each individual requests and it if the VM is not free then incoming job should wait in the queue.

B. Equally Spread Current Execution Load

This algorithm requires a load balancer which monitors the jobs which are asked for execution. The task of load balancer is to queue up the jobs and hand over them to different virtual machines [5]. The balancer looks over the queue frequently for new jobs and then allots them to the list of free virtual server. The balance also maintains the list of task allotted to virtual servers, which helps them to identify that which virtual machines are free and need to be allotted with new jobs. The experimental work for this algorithm is performed using the cloud analyst simulation. The name suggests about this algorithm that it work on equally spreading the execution load on different virtual machine.

C. Active Monitoring Load Balancer

Active VM Load Balancer maintains information about each VMs and the number of requests currently allocated to which VM. When a request to allocate a new VM arrives, it identifies the least loaded VM [6]. If there are more than one, the first identified is selected. Active VM Load Balancer returns the VM id to the Data Center Controller the data Center Controller sends the request to the VM identified by that id. Data Center Controller notifies the Active VM Load Balancer of the new allocation.

D. Throttled Load Balancer

Throttled algorithm is completely based on virtual machine. Here the client first requests the load balancer to check the right virtual machine which access that load easily and perform the operations which is given by the client [7]. In this algorithm the client first requests the load balancer to find a suitable Virtual Machine to perform the required operation.

In the present work we are considering Round Robin, Throttled algorithm as a comparison because these algorithms distribute the load uniformly. Our focus is on improving the Throttled algorithm which is achieved significantly.

III. PROPOSED MODIFIED THROTTLED ALGORITHM

This algorithm focuses mainly on how incoming jobs are assigned to the available virtual machines intelligently. The basic methodology of the proposed algorithm is given in the following Fig. 1.

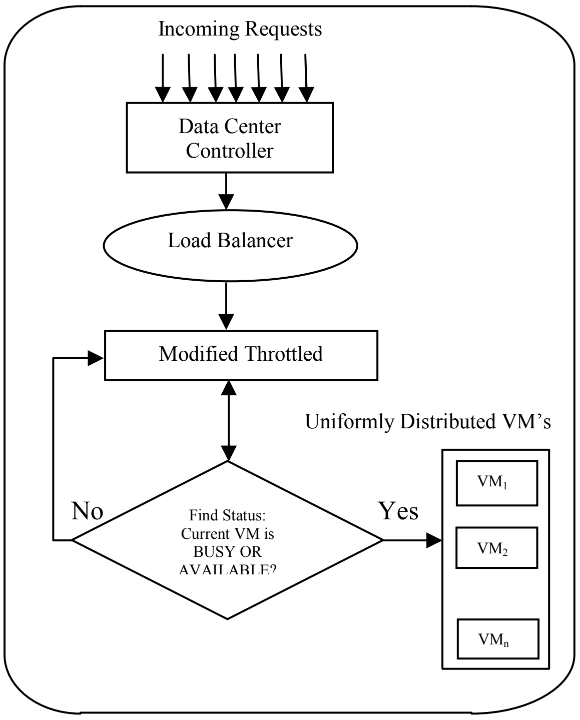

Fig.1: Flow of the Proposed Modified Throttled Algorithm

Algorithm: Modified Throttled Load Balancer

Input: No of incoming jobs $x_1, x_2 \ldots\ldots x_n$
Available VM $y_1, y_2 \ldots\ldots y_n$

Output: All incoming jobs $x_1, x_2 \ldots x_n$ are allocated one by one to the available VM $y_1, y_2 \ldots\ldots y_n$

1: Modified Throttled Load Balancer maintains an index table of VMs and the state of the VM(BUSY/AVAILABLE). At the start all VM's are available.

2: Data Center Controller receives a new request.

3: Data Center Controller queries the Modified Throttled Load Balancer for the next allocation.

4: Modified Throttled Load Balancer starts with the VM at first index, checking for the availability of the VM.

case 1: if found

a. The Modified Throttled Load Balancer returns the VM id to the Data Center Controller

b. The Data Center Controller sends the request to the VM identified by that id.

c. Data Center Controller notifies the Modified Throttled Load Balancer of the new allocation

d. Modified Throttled Load Balancer updates the allocation table accordingly

case 2: if not found

a. The Modified Throttled Load Balancer returns -1.

5: When the VM finishes processing the request, and the

Data Center Controller receives the response cloudlet, it notifies the Modified Throttled Load Balancer of the VM de-allocation.
6: If there are more requests, Data Center Controller repeats step 4 with next index and process is repeated until size of index table is reached. Once the size of index table is reached parsing starts with first index.
7: Continues from step 2.

Modified throttled algorithm maintains an index table of virtual machines and also the state of VMs similar to the Throttled algorithm. There has been an attempt made to improve the response time and achieve efficient usage of available virtual machines. Proposed algorithm employs a method for selecting a VM for processing client's request where, VM at first index is initially selected depending upon the state of the VM. If the VM is available, it is assigned with the request and id of VM is returned to Data Center, else -1 is returned. When the next request arrives, the VM at index next to already assigned VM is chosen depending on the state of VM and follows the above step, unlikely of the Throttled algorithm, where the index table is parsed from the first index every time the Data Center queries Load Balancer for allocation of VM.

IV. EXPERIMENTAL SETUP

Complete execution is carried out in a simulator. For the experimentation CloudAnalyst simulator has been used [7]. As cloud infrastructure is distributed requests should be handled from different geographical locations.

A. Brief overview of simulator

CloudAnalyst simulator gives the real time scenario with six different geographical locations. i.e no of users from particular locations can be identified depending on the specific application, e.g facebook users from Asia, Africa etc. The simulator is very flexible and it provides data centers, virtual machines, band width and many more for experimentation [7]. A snapshot of the CloudAnalyst architecture is shown in Fig. 2 [3] and simulation toolkit is in Fig. 3.

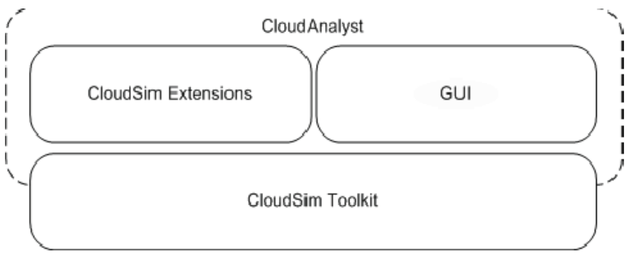

Fig.2: Architecture of CloudAnalyst simulator

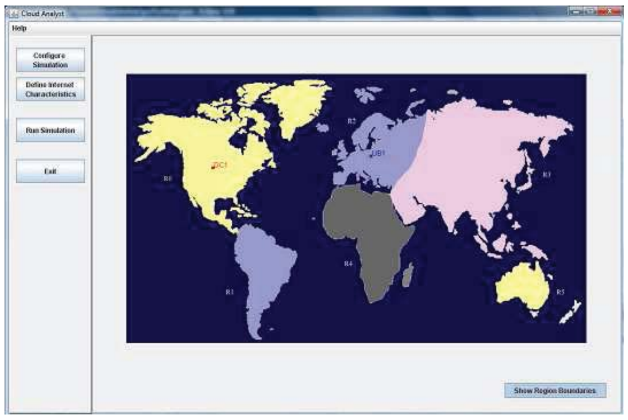

Fig.3: Snapshot of simulator

Hypothetical applications like Facebook users, Twitter users, Internet users are considered for experimentation. Six different geographical locations (six different continents of the world) are considered [3]. A single time zone is considered for all user locations. For simplicity one hundredth of the total users from each continent is considered and it is assumed that only 5% of total users are online during peak hours and in off peak hours, users are one tenth of the peak hours.

For experimentation internet users at six different continents are considered i.e. six user bases and peak and non peak users are given in the table 1. We have considered internet users at different continents from the month of June 2012. The same data is experimented with three different scheduling algorithms and response time of each algorithm is also considered for the result analyses.

TABLE I. Simulation Configuration

User Base	**Region**	**Simultaneous online Users during peak Hours**	**Simultaneous online users during off peak hours**
North America	0	135000	13500
South America	1	125000	12500
Europe	2	255000	25500
Asia	3	535000	53500
Africa	4	30000	3000
Oceania	5	10000	1000

Each Data Center has a capacity to host a no of virtual machines which are needed for particular application. Machines have 100 GB of storage space, 4 GB of RAM, each machine has 4 CPU and a power of 10k MIPS.

V. RESULTS AND ANALYSIS

Results are analyzed in two different aspects. First w.r.t load balancing of VMs and secondly average response time of proposed algorithm compared to existing algorithms.

A. Load balancing of VMs

As mentioned in section III the proposed algorithm will not parse the index table from the beginning every time. But this is not the same case with throttled load balancing algorithm in which irrespective of the no of the request it parses the index table from the beginning. So with this in Throttled algorithm few VMs are overloaded and remaining VMs are underutilized. This results in imbalance of the load on the VMs. But if we use the proposed algorithm all the VMs are utilized completely and properly. Both the algorithms are compared using five VMs and the following table 2 gives the information about how many times each VM has been used efficiently.

TABLE 2. VMs usage

SI. No	**Throttled**	**Round Robin**	**Modified Throttled**
VM0	1182	254	254
VM1	76	254	254
VM2	8	253	254
VM3	2	253	253
VM4	0	254	253

From the table 2 we can observe that using Round Robin and Modified Throttled algorithm we can have efficient utilization of VMs compared to Throttled algorithm. But if we compare Round Robin and Modified Throttled, the results are same but in Round Robin we don't check the state of the VM (BUSY/AVAILABLE) which leads to queuing of the incoming request at the server. Modified Throttled algorithm gives better results by checking the state of the VM and thus avoiding the queuing at server.

B. Response time for the execution of the client's request

Simulation scenario was setup as explained in the section IV. We have considered a single Data Center (DC) with 100 VMs. Simulation was repeated for Round Robin, Throttled and proposed Modified Throttled algorithm respectively. With the proposed algorithm response time for request has been improved compared to other two algorithms. The following table 3 gives the information about average response time of all three algorithms.

TABLE 3. Response time for all algorithms

Algorithm	**Average Response Time**
Round Robin	364.85 ms
Throttled	364.76 ms
Modified Throttled	363.52 ms

The following Fig. 4 shows that the proposed Modified Throttled algorithm gives better average response time compared to other two algorithms.

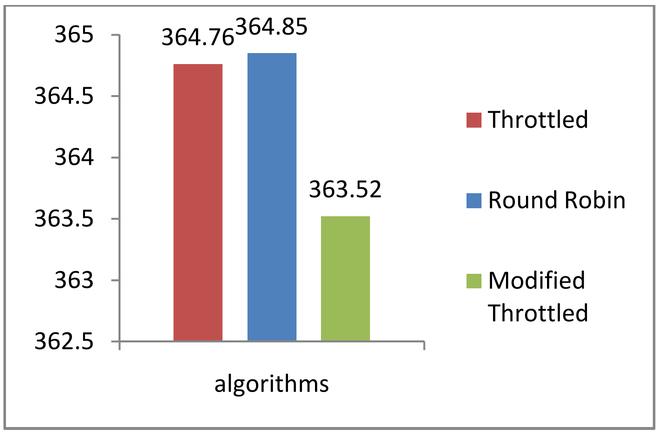

Fig.4: Average Response Time of all three algorithms

VI. CONCLUSION

In this paper, an efficient approach to handle the load at servers by considering both availability of VMs for a given request and uniform load sharing among the VMs for the number of requests served. The work aimed at efficient method for load balancing, depicted from its two different objectives. One being the response time required to serve the requests and other being the distribution of load among the existing VMs. When compared to existing Round-Robin and Throttled algorithms, the response time for proposed algorithm has improved considerably, evident from the results presented in section V. Distribution of load among the virtual machines in Round-Robin algorithm was nearly uniform, but was found less efficient considering response time. Throttled algorithm with better response time than Round-Robin failed to distribute load uniformly, overloading initial VMs and leaving others underutilized. Proposed algorithm distributes load nearly uniform among VMs, with improved response time compared to existing algorithms.

As a future scope the present work need to be focused on changing the data structures used for maintaining the index table and also by incorporating the paradigms of parallel and high performance computing the response time and utilization of VMs may be further optimized.

REFERENCES

[1] Klaithem Al Nuaimi, Nader Mohamed, Mariam Al Nuaimi and Jameela Al-Jaroodi, "A Survey of Load Balancing in Cloud Computing: Challenges and Algorithms", in Second Symposium on Network Cloud Computing and Applications, 978-0-7695-4943-9/12, IEEE 2012.

[2] Hadi Goudarzi, Mohammad Ghasemazar, and Massoud Pedram, "SLA-based Optimization of Power and Migration Cost in Cloud Computing" in 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2012.

[3] Brototi Mondal, Kousik Dasgupta and Paramartha Dutta, "Load balancing in cloud computing using stochastic hill climbing-a soft computing approach" in Procedia Technology 4 (2012) 783 – 789, ELSEVIER C3IT-2012.

[4] Mr.Manan D. Shah, “Allocation Of Virtual Machines In Cloud Computing Using Load Balancing Algorithm” in International Journal of Computer Science and Information Technology & Security (IJCSITS), ISSN: 2249-9555 Vol. 3, No.1, February 2013.

[5] Ms.Nitika, Ms.Shaveta, Mr. Gaurav Raj, “Comparative Analysis of Load Balancing Algorithms in Cloud Computing” in International Journal of Advanced Research in Computer Engineering & Technology Volume 1, Issue 3, May 2012.

[6] Dr. Hemant S. Mahalle, Prof. Parag R. Kaveri, Dr.Vinay Chavan, “Load Balancing On Cloud Data Centres” in International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, issue 1, January 2013.

[7] B.Wickremasinghe, R.N.Calheiros,R.Buyya, Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing in: Proceedings of the 24th International Conference on Advanced Information Networking and Applications (AINA 2010), Perth, Australia,, 2010.

[8] Zehua Zhang and Xuejie Zhang, “A Load Balancing Mechanism Based on Ant Colony and Complex Network Theory in Open Cloud Computing Federation” in 2nd International Conference on Industrial Mechatronics and Automation, IEEE 2010.

[9] Suraj Pandey, LinlinWu, Siddeswara Mayura Guru, Rajkumar Buyya, “A Particle Swarm Optimization-based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments” in 24th International Conference on Advanced Information Networking and Applications, IEEE 2010.