ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 7, Special Issue 6, May 2018

2nd National Conference on "RECENT TRENDS IN COMPUTER SCIENCE & INFORMATION TECHNOLOGY (RTCSIT'18)"

13th-14th March 2018

Organized by

Departments of Computer Science & Information Science Engineering, Sri Krishna Institute of Technology, Bangalore, India

FCM-LB: Fuzzy C Means Cluster Based Load Balancing in Cloud

Geetha Megharaj[1], Dr. Mohan G. Kabadi[2], Rajani[3], Deepa M.[4]

Associate Professor, Department of Computer Science & Engineering, Sri Krishna Institute of Technology, Bangalore, India[1]

Professor and Head, Computer Science & Engineering, Presidency University, Bangalore, India [2]

UG Student, Department of Computer Science & Engineering, Sri Krishna Institute of Technology, Bangalore, India [3, 4]

ABSTRACT: Load balancing strategies aim at achieving high user satisfaction by minimizing response time of the tasks and improving resource utilization through even and fair allocation of cloud resources. Cloud data centers require efficient load balancing strategy to reduce excessive work load on some Virtual Machines (VM).The traditional Throttled load balancing algorithm is a good approach for load balancing in cloud computing as it distributes the incoming jobs evenly among the VMs. But the major drawback of the algorithm is that it works well for environments with homogeneous VMS, does not considers the resource specific demands of the tasks and has additional overhead of scanning the entire list of VMs every time a task assigned to datacenter.
In this paper, these issues have been addressed by proposing a Cluster based load balancing algorithm which works well in heterogeneous nodes environment, considers resource specific demands of the tasks and reduces scanning overhead by dividing the Virtual machines into clusters. Experimental results have shown that our algorithm gives better results in terms of waiting time, execution time, turnaround timeand throughput as compared to existing throttled algorithm. The implementation of proposed methodology FCM-LB has been done using MATLAB and comparative analysis done to evaluate the FCM-LB method with a traditional load balancing algorithms such as throttled and experimental results have shown that the proposed method gives better results.

KEYWORDS: Load Balancing Algorithm, Task Scheduling, Fuzzy C Means, Clustering, Cloud Computing.

I.INTRODUCTION

Cloud computing is basically divided into 3 classes, based on the services provided Platform as a Service (PaaS), Infrastructure as a Service (IaaS) and Software as a Service (SaaS). This new computing paradigm has provided new business opportunities for Cloud service providers and service requesters. The Cloud centre provides all the 3 services by abstracting its physical resources. Most of the Cloud centers provide their services through the Virtual Machine (VM), which is a virtual system that provides its computing services through the abstraction of physical resources.The Cloud service centers can be distributed in different geographical locations. The existing Cloud center can be built by the federation of other different Cloud centers. This scenario createsan opportunity to provide different kinds of services.

Load balancing algorithms can be broadly classified into two types: Static algorithms and Dynamic algorithms. In Static Scheduling the assignment of tasks to processors is done before program execution begins i.e. in compile time. Scheduling decision is based on information about task execution times, processing resources, etc., which are assumed to be known at compile time [1]. Static scheduling methods are non-preemptive. The goal of static scheduling methods is to minimize the overall execution time. These algorithms cannot adapt to load changes during run-time [2]. Dynamic scheduling (often referred to as dynamic load balancing) is based on the redistribution of processes among the processors during execution time. This redistribution is performed by transferring tasks from the heavily loaded processors to the lightly loaded processors with the aim to improve the performance of the application.

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 7, Special Issue 6, May 2018

2nd National Conference on "RECENT TRENDS IN COMPUTER SCIENCE & INFORMATION TECHNOLOGY (RTCSIT'18)"

13th-14th March 2018

Organized by

Departments of Computer Science & Information Science Engineering, Sri Krishna Institute of Technology, Bangalore, India

It is particularly useful when the requirement of process is not known a priori and the primary goal of the system is to maximize the utilization of resources. The major drawback of the dynamic load balancing scheme is the run-time overhead due to the transfer of load information among processors and decision-making for the selection of processes and processors for job transfers and the communication delays associated with the task relocation itself. The dynamic load balancing algorithms can be centralized or distributed depending on whether the responsibility for the task of global dynamic scheduling should physically reside in a single processor (centralized) or the work involved in making decisions should be physically distributed among processors [3]. The most important feature of making decisions centrally is simplicity [1]. However, centralized algorithms suffer from the problem of the bottleneck and single point failure. Distributed load balancing algorithms are free from these problems. Again distributed dynamic scheduling can be cooperative or non-cooperative. The last one is simple where individual processors act alone as autonomous entities and arrive at decisions regarding the use of their resources independent of the effect of their decision on the rest of the system. In the former one each processor has the responsibility to carry out its own portion of the scheduling task to achieve a common system wide goal [1], [3]. The load balancing problem can be divided into two sub problems:

1. Submission of new task for VM provisioning and placement of VMs on host.
2. Reallocation/migration of VMs. Different load balancing algorithms are there in the literature which are discussed below.

For the first sub-problem, algorithms has been developed to be applied at load balancer which will decide to which VM aparticular request will be assigned. In the second sub-problem,when a particular host gets overloaded, then some of its VMsget migrated to the hosts which are lightly loaded This research mainly focuses on the second sub problem of load balancing. Mostly the existing algorithms are designed in away that they work well for homogeneous environments. Therefore we develop such a dynamic load balancingalgorithm which will serve heterogeneous environments and will consider the resource specific demands of the tasks. Weaim to improve the throughput, execution time, response timeand turnaround time.The overall architecture of the system model has been shownin figure 1.

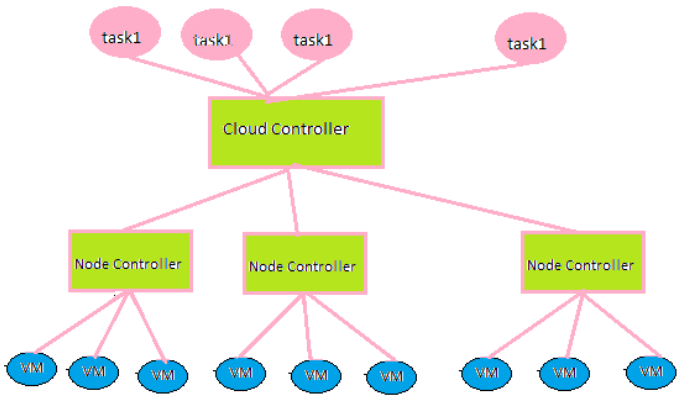

Fig. 1 System model

II. RELATED WORK

In this section, we will be reviewing various load balancing algorithms in the area of cloud computing.In throttled load balancing algorithm, a list of VMs is maintained along with the status of each VM [4]. When a new task arrives, datacenter controller queries the load balancer for the new request. Load balancer starts scanning the list of VMs from the beginning in order to find a suitable VM which is available. If the VM is found, then load balancer return VM id to datacenter controller and if suitable VM is not available, then load balancer returns -1. Once a task is allocated to a VM, its status is changed to busy from Available.

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 7, Special Issue 6, May 2018

2nd National Conference on "RECENT TRENDS IN COMPUTER SCIENCE & INFORMATION TECHNOLOGY (RTCSIT'18)"

13th-14th March 2018

Organized by

Departments of Computer Science & Information Science Engineering, Sri Krishna Institute of Technology, Bangalore, India

In Round Robin [5] algorithm, the requests are assigned VM's in circular order on first come first serve basis. 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 if the VM is not free then incoming job should wait in the queue.

In paper [6], the author has proposed a modified throttled algorithm which overcomes the problem of throttled algorithm of scanning the entire list of VMs from the beginning. In modified throttled algorithm being proposed, the first incoming job is assigned to the first available VM, then for the next incoming request, scanning of list starts from the VMnext to the one allocated to previous task unlike throttledalgorithm in which every time when a task comes, the load balancer starts scanning the VM list from the beginning. In this way, scanning overhead is reduced in this proposed algorithm.

In paper [7], author has proposed a hybrid load balancing approach using ESCE and throttled algorithms. In ESCE algorithm, load balancer maintains an index table of virtual machines along with the number of requests currently allotted to a machine. New request is assigned to the VM which hasthe least load. In throttled algorithm, a list of VMs along with their status whether available or busy is maintained by theload balancer. The incoming request is assigned to the first available VM in the list. The algorithm proposed in [8] is a hybrid of throttled and ESCE algorithms. It combines the positive points of both throttled and ESCE algorithms. It maintains an index list of the VMs with their allocation countas well as status. The status of VM can be available or busy depending upon the number of requests currently being allocated to it.

The author in the paper [9] proposes a Load Balance Improved Min-Min (LBIMM) Scheduling algorithm that takes the characteristic of Min-Min scheduling algorithm as foundation to minimize the completion time of all resources and improves the load imbalance factor of the Min-Min algorithm. It also adds the user priority aware factor to the above algorithm by dividing the user tasks into two groups G 1 and G2 where G 1 consists of VIP users tasks and G2 consists of ordinary users tasks such that VIP users tasks are scheduled first and are assigned to the VIP qualified resources set. Thus paper finally gives a PA-LBIMM algorithm that performs the above functions. The proposed LBIMM and PALBIMM algorithms are evaluated in MATLAB. Both these algorithms perform better than Min-Min algorithm in terms of completion time of tasks and load balancing of resources. PA-LBIMM performs better than LBIMM in terms of completion time of VIP tasks. But some of the other issues are still open such as deadline of each task, QOS requirements, and dependencies oftasks.

The algorithms reviewed have a problem that they do not consider the resource specific demands of the tasks.

III.PROPOSED WORK

In this section, the proposed approach is discussed. The proposed algorithm adds clustering approach so as to divide VMs with similar capacities into groups. Fuzzy C Means clustering [10] approach has been used to divide VMs into cluster. The load balancer will maintain a list of all the clusters with the minimum and maximum resource specific capacities of each cluster. This is range specified list. Also the load balancer will maintain the list of VMS for each cluster. The proposed approach is dynamic, centralized and heterogeneous in nature, considers resource specific demands. It reduces the overhead of scanning the entire list of VMs from the beginning.

The step-wise procedure of the proposed approach is given as follows:

Stepl: Initialize all VMs with their specific resource types, capacities of each resource and status of VMs.
Step2: Cluster the n VMs into n clusters using Fuzzy C Means clustering using the three resource types as parameters i.e. CPU processing speed, Memory and network bandwidth.
Step3: Cloud controller receives a new request
Step4: Cloud controller queries appropriate node controller/load balancer for next allocation.
Step5: Load balancer scans the range specified list of n clusters to see that which cluster can handle the incoming request

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 7, Special Issue 6, May 2018

2nd National Conference on "RECENT TRENDS IN COMPUTER SCIENCE & INFORMATION TECHNOLOGY (RTCSIT'18)"

13th-14th March 2018

Organized by

Departments of Computer Science & Information Science Engineering, Sri Krishna Institute of Technology, Bangalore, India

Step6: Load balancer will then assign the request to the appropriate VM of the chosen cluster by looking into the list of cluster members which will match the specific demands ofthe task and whose status is available. In case more than oneVMs satisfy this, then the first one which is found will get thetask.
Step7: Remaining resource quantities of that VM in the VMlist of that cluster is then updated.
Step 8: Status of that VM is changed from AVAILABLE to BUSY.
Step9: When the VM finish processing the request, the statusof that VM is changed to AVAILABLE.
Step 10: The load balancer also updates the capacity of thatVM in the VMs capacities list.

The FCM algorithm operates by allocating membership to every data point with respect to each cluster centroid based on distance between the cluster centroid & the data point. More the data is close to the cluster centroid, more is its membership headed for the particular cluster centroid. So, summation of membership of each data point should be equivalent to one.

The clustering of VMs is based on minimization of the following objective function presented in equation(1).

$$J = \sum_{i=1}^{N} \sum_{j=1}^{c} \mu ij \|xi - cj\| \quad \text{.....(1)}$$

The number of centroids randomly *C* and it can be updated using the following equation (2) at every iteration. At initial stage of clustering, the membership values between each data points with respect to every clusters can be defined randomly between 0 to 1 and with the constraints of summation of membership of each data point should be equal to one. At every successive iteration the membership values can be updated using the membership update function presented in following equation (3).

$$cj = \frac{\sum_{i=1}^{N} \mu ij * xi}{\sum_{i}^{N} \mu ij} \quad \text{.....(2)}$$

$$\mu ij = \frac{1}{\sum_{k=1}^{c} \frac{\|xi - cj\|}{\|xi - ck\|}} \quad \text{.....(3)}$$

From the above equation 2, where $c_j$indicates the updation of jthcentroid and $x_i$represents the data point and $\|xi - cj\|$signifies the Euclidian distance between the i^{th} data point with respect to jth centroid and$\|xi - ck\|$denotes the Euclidian distance between the i^{th} data point with respect to all other centroids except jthcentroid. With the aid of equation 2 and 3, the centroids and membership values are updated at each iteration. This process is repeated until minimum value of *J* stated in equation (1) is achieved.

When user submits task, the load balancer matches taskresource specific requirements with capacity range of clusterto assign the task to appropriate cluster. Then amonga cluster, load balancer will match the suitable availablevirtual machine to which task must be assigned. Thusscanning of list gets divided into two levels. This will reducethe time involved in list scanning and also will assign a betterand more suitable VM to the task as per its requirements.

The parameters that we are considering in this paper are:

1. Response time/Waiting time: The time taken by the loadbalancer to allocate a particular suitable VM to the task.
   Response time= Allocation time-task submission time
2. Execution time: The time taken by the allocated VM toprocess the task.
   Execution time= Task size/CPU speed
3. Turnaround time: waiting time + Execution time

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 7, Special Issue 6, May 2018

2nd National Conference on "RECENT TRENDS IN COMPUTER SCIENCE & INFORMATION TECHNOLOGY (RTCSIT'18)"

13th-14th March 2018

Organized by

Departments of Computer Science & Information Science Engineering, Sri Krishna Institute of Technology, Bangalore, India

IV. EXPERIMENTAL RESULTS

The proposed algorithm is implemented in MATLAB. Arandom task generator and random VM generator have beenused. There is no limit to the number of tasks generated. VMsconsidered in the experiment are 50. Simulation parametershave been mentioned in table 1.

Table 1. Simulation parameters

Number of VMs	50
Number of clusters	5
Number of tasks	100-5000
VM memory	512-2048 MB
VM bandwidth	500-2500bps

Figure 2 shows the graph of waiting time vs number of tasks for throttled and the proposed algorithm. The proposed algorithm gives better results than Throttled algorithm.

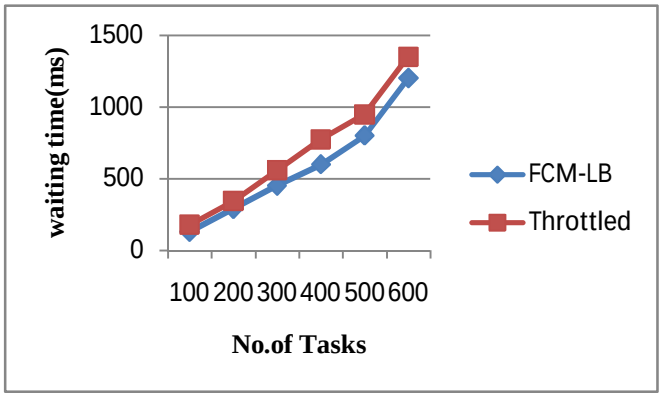

Fig. 2 Waiting time Vs Number of Tasks

Figure 3 shows the plot between execution time and numberof tasks. The execution time ofproposed algorithm is less than Throttled algorithm as theproposed approach is designed in a way that it assigns more accurate VMs to the tasks matching their requirements.

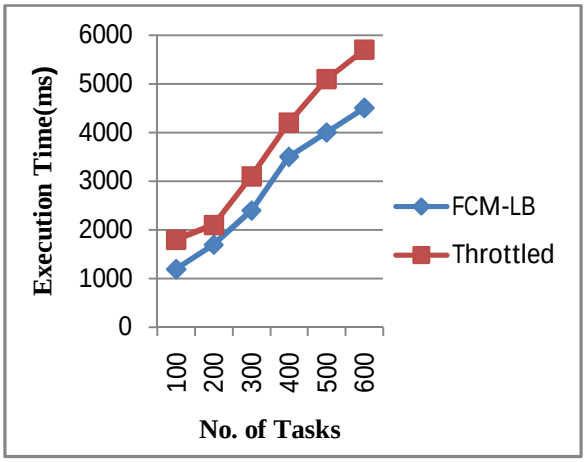

Fig. 3 Execution time Vs Number of Tasks

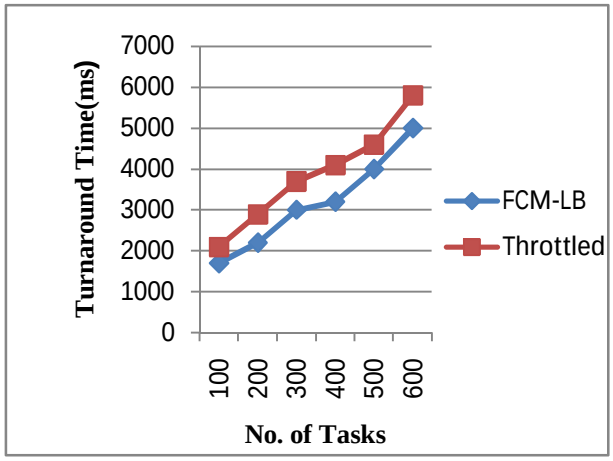

Fig. 4 Turnaround time Vs Number of Tasks

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 7, Special Issue 6, May 2018

2nd National Conference on "RECENT TRENDS IN COMPUTER SCIENCE & INFORMATION TECHNOLOGY (RTCSIT'18)"

13th-14th March 2018

Organized by

Departments of Computer Science & Information Science Engineering, Sri Krishna Institute of Technology, Bangalore, India

Figure 4 shows the graph plotted between turnaround time and number of tasks. The proposed approach performs better than the two existing algorithms and theperformance becomes better as the number of tasks increase.

V. CONCLUSION

In this paper, we have studied that aspect of load balancingwhich deals with assigning tasks to appropriate VMs in orderto maintain the load on all the VMs. Various existing algorithms have been reviewed for this work. The existing throttled algorithm does not consider resource specific demands of the tasks, are not suitable for heterogeneous VMs environment and have additional overhead of scanning the entire list of VMs. These problems have been addressed in the proposed approach by clustering the VMS into groups such that VMs in same group have similar capacities. Experimental results have shown that our proposed approach gives better results than throttled when compared on the basis of waiting time, execution time and turnaround time.

REFERENCES

[1]. X. Evers, W. H. CSG, CR. B. SG, I. S. Herschberg, D. H. J. Epema, and J. F. C. M. de Jongh, A literature study on scheduling in distributed systems, Delft University of Technology, 1992.
[2]. K. A. Nuaimi, N. Mohamed, M. A. Nuaimi, and J. Al-Jaroodi, A survey of load balancing in cloud computing: Challenges and algorithms, Proc. 2012 Second Symposium on Network Cloud Computing and Applications (NCCA), 2012, 137-142.
[3]. Thomas L. Casavant, and Jon G. Kuhl, A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems, IEEE Trans, on Software Eng., 14(2), 1988, 141-154.
[4]. Jasmin James, Dr. BhupendraVerma, "Efficient VM Load Balancing Algorithm for a Cloud Computing Environment ", International Journal on Computer Science and Engineering (UCSE) , Vol. 4 No. 09 Sep 2012 , pp. 1658-1663.
[5]. [5] M. M. D. Shah, M. A. A. Kariyani, and M. D. L. Agrawal, Allocation of Virtual Machines In Cloud Computing Using Load Balancing Algorithm, IJCSITS, 2013, 2249-9555.
[6]. Domanal, S.G. Reddy, G.R.M., "Load Balancing in Cloud Computingusing Modified Throttled Algorithm," Cloud Computing in Emerging Markets (CCEM), 2013 IEEE International Conference on , vol., no.,pp.I-5, 16-18 Oct. 2013
[7]. Bagwaiya, V.,Raghuwanshi, S.K., "Hybrid approach using throttled andESCE load balancing algorithms in cloud computing," Green ComputingCommunication and Electrical Engineering (ICGCCEE), 2014International Conference on , vol., no., pp.I-6, 6-8 March 2014 doi:10.1109IJCGCCEE.2014.6921418
[8]. Bourguiba, M. Haddadou, K. EI Korbi, I. Pujolle, G., "ImprovingNetwork VO Virtualization for Cloud Computing," Parallel andDistributed Systems, IEEE Transactions on , vo1.25, no.3, pp.673-681,March 2014 doi: 10.II09rrPDS.2013.29
[9]. Huankai Chen; Wang, F.; Helian, N.; Akanmu, G., "User-priority guidedMin-Min scheduling algorithm for load balancing in cloud computing,"Parallel Computing Technologies (PARCOMPTECH), 2013 NationalConference on , vol., no., pp.I-8, 21-23 Feb. 2013
[10]. Bezdek, James & Ehrlich, Robert & Full, William. (1984). FCM—the Fuzzy C-Means clustering-algorithm.Computers & Geosciences. 10. 191-203. 10.1016/0098-3004(84)90020-7.