ISSN 2278-3091
Volume 9 No.2, March -April 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse34922020.pdf
https://doi.org/10.30534/ijatcse/2020/34922020

Credit Based Scheduling with Load Balancing in Cloud Environment

Abhikriti Narwal[1] Sunita Dhingra[1*]
[1] Department of Computer Science & Engineering, University Institute of Engineering & Technology, Maharshi Dayanand University, Rohtak
sunitadhingramdu@rediffmail.com

ABSTRACT

Cloud computing is the recent advancement in the technology of distributed computing that works on the basis of pay as you use model. It comprises of virtual machines that provides both storage and computational facility. The main aim of cloud computing is to offer access to remotely distributed resources to its users. Scheduling of tasks plays a vital role in the efficient working of cloud computing. Sometimes situation comes where multiple tasks with same priority arrives so a good scheduler is one which handles the situation appropriately with proper load balancing. There are various scheduling algorithms proposed in the past. Most of the scheduling algorithms neglect the concept of load balancing. Load balancing in cloud computing is also as much important as task scheduling. As cloud environment consists of virtual machine, it should take care that no virtual machine remains idle and also no virtual machine is under heavy load of the tasks. So, it is important to balance the load equally among the virtual machines to solve the issue of under loading and over loading of virtual machines. In this paper, a credit based scheduling algorithm with load balancing (CBSA_LB) is proposed that balances the load along with the scheduling of tasks. The results are evaluated on the basis of six parameters: processing time, processing cost, response time, makespan time, Throughput and Execution Time. Experimental results show that the proposed technique outperforms the existing techniques (EMOSA and CBSA).
.

Key words: Cloud computing, Virtual machines, Task scheduling, Load balancing, Credit based scheduling algorithm, Enhanced multi objective scheduling algorithm.

1. INTRODUCTION

Cloud computing is a Distributed Computing worldview that gives administrations to their clients on the premise of as per utilization by specific client. Use to such an extent or less you need to utilize, utilize administrations when you need to utilize and pay just for what you have utilized. Distributed computing is a development that permits you to utilize functions that really live on an area not similar to your machine area and gives a distinctive virtualized stage that encourages client to fulfill their occupations with least finishing time and least costs. Cost adequacy, adaptability, dependability, adaptation to internal failure, benefit introduction, utility based, virtualization and service level agreement (SLA) are some of the striking highlights of Cloud Computing [1].

Cloud computing is a compensation for each utilization benefit instead of an item. The term used in cloud computing for providing such sort of administrations is known as cloud service providers. For Example Google, Amazon, Microsoft are cloud service providers. The two most critical segments of cloud computing architecture are called as back end and front end. The front end is the portion observed by the customer, i.e. the PC client. This joins the client's framework (or PC) and the applications used to get to the cloud by methods for a User Interface, for instance, a web program. The back end of the distributed computing engineering is just the cloud, containing distinctive PCs, servers and data storing contraptions.

Load balancing is one of the principle challenges in cloud processing. It is a component that conveys the dynamic workload equitably above all the hubs in the whole cloud to maintain a strategic distance from a circumstance where some hubs are dynamically stacked while others are performing less work [2]. By ideal usage of assets it accomplishes less reaction time, high throughput, enhanced adaptation to non-critical failure, adaptability, high client fulfilment, less warmth age, ideal power utilization, and less operational cost.

The resolute quality and execution of cloud organizations relies on various components such as booking of tasks in the cloud. Arranging ought to be conceivable at task level or resource level or work process level. Planning of employment and keeping up stack is turning into a principle issue in cloud condition. This can be accomplished by receiving suitable assignment planning calculation. By considering parameters, for example, throughput, asset usage, cost, computational time, need, execution, transfer speed, asset accessibility and some all the more, scheduling calculations are executed. With a specific end goal to give better Quality of Service (QoS), there is need to execute a suitable assignment planning calculation which keeps up a decent harmony between asset usage to empower productivity, lesser make span, simultaneous errand booking, appropriate asset use and administration.

Scheduling algorithms are utilized for the most part to limit the time and cost of execution. A decent planning calculation ought to consider the heap adjustment of the framework and aggregate execution time of the accessible assets. For accomplishing them too it is not good to squander the esteems with elevated abilities to the employments with less length. The scheduler ought to

allot the jobs to assets as per the activity length and assets limits [3].

2. BACKGROUND

Nancy et al. (2013) discussed the load balancing problem solution by integrating the back-propagation approach of NN with genetic algorithm for effective load balancing in grid system [1].

Kaur et al. (2013) proposed a scheduling approach which schedules moldable task in a manner of heterogeneous system and the basic processor unit can be used to calculate the speed of the processors. The technique integrates selection of jobs, site and processor in a single technique to optimize the response time [2].

Mathew et al. (2014) present a definite investigation of different undertaking booking strategies existing for the cloud condition. A short investigation of different planning parameters considered in these strategies is too covered. It thinks of some as heuristic, vitality effective and half strategies for contemplate. A concise examination of each strategy is done and many computations done booking in view of more than two parameters. A better booking computation can be generated from the latest techniques by counting more number of estimations which can achieve incredible execution. [3]

Yash P et al. (2014) thinks about different scheduling algorithms and characterize them on the premise of components, for example, time and expenditure which have been displayed. This near investigation should be supportive in determination of proper planning calculations for utilizing diverse sorts of administrations according to the prerequisites of cloud buyers and additionally cloud service providers. [4]

Javanmardi et al. (2014) show a half breed work planning approach with the guide of hereditary calculation and fluffy hypothesis, which considers the heap adjusting of the framework and decreases add up to time and cost of execution. They attempt to change the standard Genetic calculation and to diminish the cycle of making populace with the guide of fluffy hypothesis. The new calculation appoints the employments to the assets with considering the work length and assets limit [5].

Maddali et al. (2014) Explained the proof of the idea that depends on web protocols and its design includes configuration of context as persons have many mobile gadgets, their social media usage and the number of communities in which they are sharing the information [6].

K R Babu et al. (2015) proposed a honey bee state dependent calculation for proficient load adjusting that depends on the rummaging conduct of bumble bees to adjust stack over VM. In the planned strategy, undertakings expelled from highly stacked VMs are dealt with as bumble bees and below stacked VMs is the sustenance source. The planned technique additionally believe the needs of undertakings in holding up lines of VMs and tried to accomplish least reaction time and lessened quantity of undertaking relocations. [7]

K. RAJA et al. (2016) gives a moved forward credit based booking computation using the parameters such as requirement of client, errand size and due date imperatives [8]

Mittal et al. (2016) gives a savvy computational procedure in which the framework itself adjusts the improved task scheduling plan from the current one as per the situation. Max-min and Min-min are pertinent in little scale appropriated frameworks. At the point when the quantity of substantial errand is more than little assignment or the other way around, both the calculations can't plan the assignment properly and the make span gets generally bigger. RASA utilizes the benefits of both the procedures and try to cover up their hindrances. Dissimilar to this calculation, enhanced Max-min and Max-min endeavors to accomplish the heap adjusting among the assets by booking vast undertaking preceding the little ones. [9]

Akbar et al. (2016) proposed a novel undertaking booking calculation named Median Deviation based Undertaking Scheduling (MDTS), which utilizes Median Absolute Deviation (Distraught) of the Expected Time to Compute (ETC) of an undertaking as a noteworthy ascribe to ascertain positions of the given assignments. They utilize coefficient-of-variety (COV) based system that considers undertaking furthermore, machine heterogeneity to appraise the Ethereum (ETC) of a specific directed acyclic graph (DAG). The proposed calculation is assessed under different conditions utilizing manufactured DAGs and certifiable applications [10].

Pandey et al. (2016) shows the issues of execution, difficulties and asset assignment systems for distributed computing. It likewise concentrates on the issues of existing systems of asset distribution [11].

Azimzadeh et al. (2017) introduced strategy for asset administration and errand task in cloud condition represents the advanced condition of both time diminishment and dependability improvement for the framework. The target of this calculation is the streamlining of the planning of autonomous errands in cloud condition and age of an ideal response for the task of assignments to existing asset [12].

Singh et al. (2017) examines stack adjusting as an instrument to convey the workload equitably to all hubs in the framework to accomplish a higher asset usage and client fulfillment. It assists in allotment and de-assignment of occurrences of utilizations with no disappointment. It reports another heap adjusting method utilizing altered **credit based framework** utilizing undertaking length, assignment need and its expenditure. The planned calculation has been actualized in cloudsim toolbox and its correlation with current calculation has been examined [13].

3. PROPOSED ALGORITHM

The goal of task scheduling in cloud computing is to offer scheduling of tasks which is optimal for users, and offer the QoS and throughput of entire system of cloud at the same time. In the existing approach, cloudlets were being scheduled according to multi objective task scheduling [14], considering the sorting of tasks using non-dominating sorting algorithm, thereby mapping the task having highest rank according non-dominating

i.e. the task dominates all other tasks is mapped to highest rank virtual machines, according to their arrival times. But the concept for sorting of the task has taken only task length and file size into consideration there arise many such cases in which two or more tasks are having the same priority after sorting, therefore proposed a

scheduling algorithm, that assigns the credits on the basis of combination of weights assigned to task comprises of task length, priority, cost and deadline. This will reduce the chances of same priority occurrence between the two tasks.

The proposed algorithm uses the concept of load balancing with the credit-based scheduling algorithm to further optimize the scheduling problem solutions. Load Balancing method guarantees the system load balancing by calculating the load on each virtual machine and transferring the loads based on demand and supply values of overloaded and underloaded machines. In the proposed credit-based scheduling with load balancing algorithm, credits are assigned to the tasks based on the four parameters including task length, task priority, cost and deadline of task and then for balancing the load, honey bee optimization algorithm is utilized.

Procedure 1: Credit based on Length of task
For all submitted tasks in the set; Ti
Task length difference (TLD) = absolute value (average length – length of particular task)
If $TLD_i \leq$ value_1
then credit =5
else if value_1 < $TLD_i \leq$ value_2
then credit =4
else if value_2 < $TLD_i \leq$ value_3
then credit =3
else if value_3 < $TLD_i \leq$ value_4
then credit =2
else value_4 > TLD_i
then credit =1
End For
where value_1= high_len / 5;
value_2= high_len / 4;
value_3=value2+value1;
value_4=value3+value2;

Honey bee concept is utilized in the virtual cloud environment as cloudlets being utilized as honey bees and the virtual machines as food sources. As the honey bees search for foodstuff in the same way cloudlets in the cloud environment are searching for the virtual machine for their execution. Each VM has different capacity for execution of tasks. Some VMs might be over-burden while others might be under stacked. So, proper Stack adjustment is required for the better execution of tasks in scheduling. Consequently, when a VM is over-burden with numerous cloudlets then some cloudlets are expelled from that specific VM and it is assigned to an underloaded VM. The basic goal of the proposed technique is to schedule the credited task based on the load calculated by using honey bee algorithm. Credits are assigned on the basis of length of tasks, priority, deadline and cost of tasks. So they are calculated by using the following Procedures:

Procedure 2: Priority credits assigning to task
For each submitted tasks in set; Ti
Search for highest priority task
Choose the divisible factor for priority
$credit\_Priority_i = TaskPriority_i / divisiblefactor$;
End For

Procedure 4: Cost of the task
For each submitted tasks in set; Ti
$Cost_i =$
*Datacentercharacteistics.getCostpermemory*vm$_{Ram}$ + Datacentercharacteristics.getCostperStorage*vm_size*

End For

Procedure 3: Deadline of the task
For each submitted tasks in set; Ti
Search for MAXMIPS of the VM from the virtual machine list
$Deadline\_Task_i = Credit\_Length_i * Credit\_Priority_i / MIPS_{MAX}$
End For

The Proposed Algorithm CBSA_LB is discussed below:

1. Initialize the Cloudsim package by creating the datacenter, broker, virtual machines and cloudlets.
2. Initialize the virtual machines list.
3. Initialize the task list.
4. Sort the virtual machines using QOS parameters (MIPS and Granulaity size).
5. Sort the task list using priorities and assign them credits as follow:
   a) CALL Procedure 1.
   b) CALL Procedure 2.
   c) CALL Procedure 3.
   d) CALL Procedure 4.

$TotalCredit = Credit_{length\ i} * Credit_{priority\ i} * Credit_{deadline\ i} * Credit_{cost\ i}$

6. Calculate the Load of the Virtual machines and CALL Procedure 5.

4. EXPERIMENTS AND ANALYSIS

This section is divided into two sections Parameter Analysis and Experimental Results. Parameter Analysis section explains the various formulas which will be used for carrying out the experiments. A total of six parameters have been discussed in this section. Experimental Results section explains the various outcomes of experiments carried out in cloudsim simulator. In this section comparison of results with previously discussed algorithms is also represented.

Procedure 5: Honey bee optimized algorithm

For all VM in set; Vm_i

1. While termination condition is false perform
2. Find the load on the virtual machine using

$$Load_i = \frac{N * Cloudlet\_length}{VM\_MIPS}$$

Where N is number of tasks assigned to a VM, *Cloudlet_length* is the length of single tasksand VM_MIPS is the MIPS rate of that VM.

The capacity of a particular VM could be computed using

$Capacity_i = PE_{Num} * PE_{MIPS} + Vm_{BW}$

Where

$Capacity_i$ = Processing capacity of VM_i

PE_{Num} = Number of Processor of VM

PE_{MIPS}= Million Instructions per second of processor of VM

Vm_{BW} =Bandwidth of Virtual Machine

3. By using load and capacity of a VM, processing time can be computed by equation

$$Processing\ Time_i = \frac{Load}{Capacity}$$

Finally, by utilizing the equation below Standard deviation (SD) of load can becomputed as,

$$SD = \sqrt{\frac{1}{N}\sum_{i=0}^{N}(X_i - X)^2}$$

Where X_i is Processing Time and $\bar{X}$ is the average Processing Time of the virtualmachine.

4. On the basis of above calculation mark vm's as under-loaded or over-loaded.

5. In this step underloaded and overloaded VMs will be calculated.

6. Arrange the under-loaded and over-loaded VM sets on the basis of Load.

7. Arrange the tasks in over-loaded VMs on the basis of priority.

8. For every task in all over-loaded VM search an appropriate under-loaded VM.

9. Revise the under-loaded and over-loaded VM sets and repeat step 2.

10. End while.

4.1 Parameter Analysis

The proposed technique is implemented in CloudSim 3.0 test system on windows 7 OS. NetBeans IDE 8.0 is utilized to run CloudSim. The planned technique is implemented by using various datasets of assignments and machines. An arrangement of experiments are carried out to compute parameters such as time of processing, make span time and the cost required for processing etc.

- **Processing Time:** Processing time is the time taken by an algorithm to execute given task. It is calculated by using following formula:

$$Processing_{time} = cloudlet_{length}/vm_{MIPS} * no\_ofPES$$

- **Processing Cost:** Processing cost is the cost required to execute given set of tasks by an algorithm. It is calculated by:

$$Processing_{cost} = Datacenter_{costpermemory} * VM_{Ram}$$

- **Make span Time:** Make span time is the total time elapsed from the start of the tasks to the end of the task i.e. the finish time of the last task. Formula for the calculation of makespan time is:

Makespan time = Finish time of last cloudlet

- **Response Time:** It is the **t**ime a system requires for reacting to start the particular task. It is the measure of difference among the start time of execution of the task and the actual cpu time of task i.e. the time when task arrived.

$$Response\ time = ActualCPUTime\ of\ cloudlet - ExecStartTime\ of\ cloudlet$$

- **Execution Time:** It is the time spent by the system for executing the task. It is the measure of difference between finish time and the start time of the task.

$$Execution\ time = FinishTime\ of\ cloudlet - ExecStartTime\ of\ cloudlet$$

- **Throughput:** It is the number of processes that complete their execution. It is a measure of how many tasks a system can scheduled in a given amount of time

$$Throughput = \frac{Total\ no.\ of\ vms}{FinishTime\ of\ cloudlet - ExecStartTime\ of\ cloudlet}$$

4.2 Experimental Results

The Experimental results of the proposed technique has been taken using various scenario created in cloudsim by varying the quantity of cloudlets and VMs. The results are compared with the existing techniques EMOSA[14] and CBSA

According to the Credit Based Scheduling Algorithm with load Balancing (CBSA_LB), the simulations results will be carried out by using CloudSim- 3.0 simulator and parameter setting on the CloudSim-3.0 is shown in Table 1

Table 1: Parameters Settings Of Cloudsim

TYPE	PARAMETER	VALUE
DATACENTER	Number of Datacenter	9
	Number of Host	2
	Type of manager	Space_Shared Time_Shared
	Number of PE[a] per Host	2
	MIPS[b] of PE	12000
	Host Memory(Ram) MB[c]	5048
VIRTUAL MACHINE	Datacenter Cost	3
	Total Number of VM	10-50

(VMs)	MIPS of VM	500
	Number of PE per VM	2
	VM Memory (RAM) (MB)	512-2048
	Bandwidth (Bit)	250-1500
TASK	Total Number of Tasks	100-500
	Length of Task(MI[d])	5000-30000
	Number of PE per requirement	1
	Type of Manager	Space_Shared

[a] PE is Processing Element, [b] MIPS (Million Instructions Per Second) is computation of speed of Computer, [c] MB is Megabyte, [d] MI is Million Instruction,

4.2.1: Performance based on Processing time

The performance of the proposed technique is evaluated here based on the Total Processing time and is compared with other existing algorithms. Figure 1 shows the Processing time comparison of algorithms.

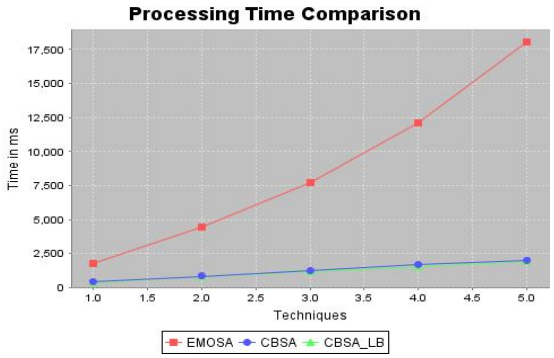

Figure 1: Comparison of EMOSA, CBSA and CBSA_LB with respect to Processing Time parameter**.**

Here, five different cases have been evaluated by varying number of tasks, the processing time of the proposed algorithm (CBSA_LB) is 1892.63 milliseconds in case of 500 tasks ,which is 2023.189 milliseconds in Credit based scheduling algorithm [7] and for EMOSA [14] the processing time is 18036.94 milliseconds. The graph clearly demonstrate that the proposed technique performs better in case of processing time.

4.2.2. Performance based on Processing Cost

The performance of the proposed technique is evaluated here based on the Total Processing cost and is compared with other existing algorithms. Figure 2 shows the Processing cost comparison of algorithms.

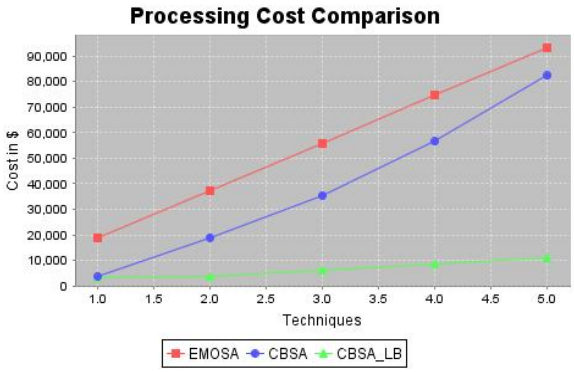

Figure 2: Comparison of EMOSA, CBSA and CBSA_LB with respect to Processing Cost parameter.

Here, five different cases have been evaluated by varying number of tasks, the processing cost of the proposed algorithm(CBSA_LB) is 11211.73 in case of 500 tasks, which is 82273.50 in Credit based scheduling algorithm and for EMOSA [14,15] the processing cost is 93320.75. The figure above clearly shows that the planned algorithm performs better on the basis of Processing Cost metric.

4.2.3. Performance based on Execution Time

The performance of the proposed technique is evaluated here based on the execution time and is compared with other existing algorithms. Figure 3 demonstrate the execution time evaluation of algorithms.

Figure 3: Comparison of EMOSA, CBSA and CBSA_LB with respect to Execution Time parameter.

Here, five different cases have been evaluated by varying number of task, the execution time of the proposed algorithm (CBSA_LB) is 159.13 in case of 500 tasks, which is 285.06 in Credit based scheduling algorithm and for EMOSA [14] the Execution time is 1792.95 milliseconds. The figure above clearly shows that the planned algorithm performs better on the basis of execution time metric.

4.2.4. Performance based on Makespan Time

The performance of the proposed technique is evaluated here based on the makespan time and is compared with other existing algorithms. Figure 4 demonstrate the makespan time comparison of algorithms.

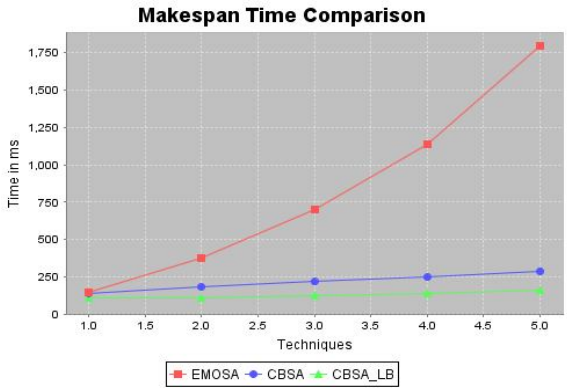

Figure 4: Comparison of EMOSA,CBSA and CBSA_LB with respect to Makespan Time Parameter.

Here, five different cases have been evaluated by varying number of tasks, the makespan time of the proposed algorithm (CBSA_LB) is 159.23 milliseconds in case of 500 tasks, which is 285.16 milliseconds in Credit based scheduling algorithm [7]. And for EMOSA[14] the makespan time is 1793.05 milliseconds. The graph clearly demonstrate that the proposed technique performs better in case of makespan time.

4.2.5. Performance based on Response Time

The performance of the proposed technique is evaluated here based on the response time and is compared with other existing algorithms. Figure 5 shows the response time comparison of algorithms.

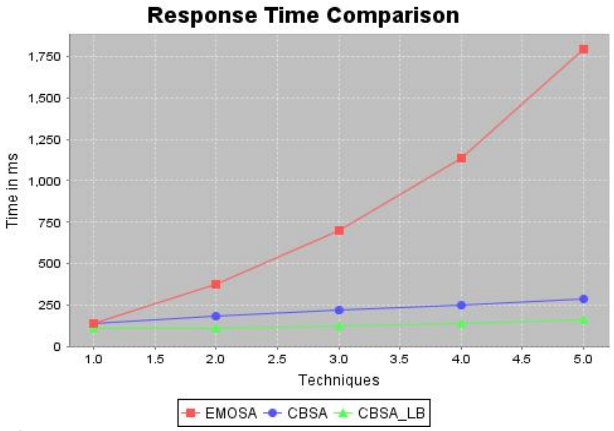

Figure 5: Comparison of EMOSA,CBSA and CBSA_LB with respect to Response Time Parameter

Here, five different cases have been evaluated by varying number of tasks, the response time of the proposed algorithm(CBSA_LB) is 159.03 milliseconds in case of 500 tasks ,which is 284.96 milliseconds in Credit based scheduling algorithm [7] and for EMOSA [14] the response time is 1792.85 milliseconds. The graph clearly shows that the proposed technique performs better in case of response time.

4.2.6. Performance based on Throughput

In this section, the performance of the proposed algorithm is evaluated based on the throughput and is compared with other existing algorithms. Figure 6 demonstrate the throughput comparison of algorithms.

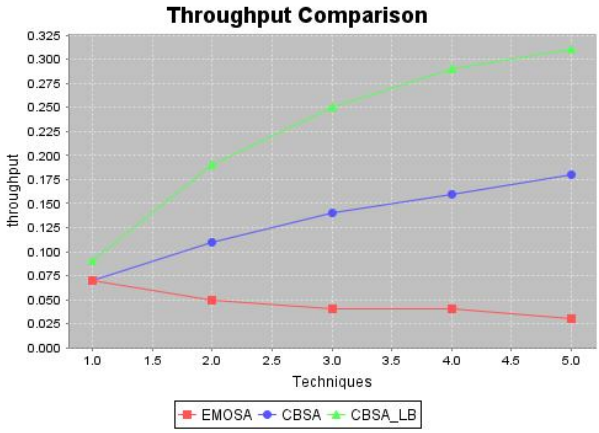

Figure 6: Comparison of EMOSA,CBSA and CBSA_LB with respect to Throughput Parameter.

Here, five different cases have been evaluated by varying number of tasks, the throughput of the proposed algorithm(CBSA_LB) is 0.31 in case of 500 tasks ,which is 0.18 in Credit based scheduling algorithm [7] and for EMOSA[14] the throughput is 0.03. The graph clearly depicts that the proposed technique performs better in case of throughput.

5. CONCLUSION

A credit based scheduling algorithm with load balancing (CBSA_LB) has been proposed. The objective of this algorithm is to efficiently balance the load in cloud environment with appropriate scheduling of tasks. This technique performs scheduling by sorting the cloudlets based on the credits assigned comprising of task length difference, priority execution cost and deadline parameters. Also for managing the load on the virtual machines honey bee optimization algorithm is used to improve the results. This technique is implemented in CloudSim Simulator. It is executed for variable number of machines with variable number of tasks. The performance of the proposed technique (CBSA_LB) is compared on the basis of processing time, processing cost, response time ,throughput, execution time and makespan with the existing techniques enhance multi objective scheduling algorithm (EMOSA) and credit based scheduling algorithm (CBSA).It is found that the proposed technique CBSA_LB outperforms CBSA and EMOSA.

REFERENCES

1. Nancy, Bhuvnesh Kumar, Sandeep Waraich and Navroop Kaur, "**Hybridization of GA and Back-Propagation for Load Balancing in Grid System**", *International Journal of Advanced Trends in Computer Science and Engineering (IJATCSE)*, Vol.2 , No.3, Pages : 36-38 , 2013.
2. Navroop Kaur, Amit Chhabra, Nancy and Bhuvnesh Kumar, "**A Hybrid Algorithm For Moldable Jobs Scheduling in Heterogeneous Multi-Cluster System**", *International Journal of Advanced Trends in Computer Science and Engineering (IJATCSE)*, Vol.2 , No.3, Pages : 13-17, 2013.
3. Teena Mathew, K. Chandra Sekaran, John Jose, "**Study and Analysis of Various Task Scheduling**

Algorithms in the Cloud Computing Environment", *IEEE International Conference on Advances in Computing, Communications and Informatics*, pp. 658-664, 2014. https://doi.org/10.1109/ICACCI.2014.6968517

4. Yash P. Dave, Avani S. Shelat, Dhara S. Patel, Rutvij H. Jhaveri, "**Various Job Scheduling Algorithms in Cloud Computing: A Survey**", *IEEE*, 2014.
5. Saeed Javanmardi, Mohammad Shojafar, Danilo Amendola, Nicola Cordeschi, Hongbo Liu, and Ajith Abraham, "**Hybrid Job Scheduling Algorithm for Cloud Computing Environment**", *Springer Conf. on Innov. In Bio-Inspired Comput. and Appl.*, pp. 43-44, 2014. https://doi.org/10.1007/978-3-319-08156-4_5
6. Maddali M. V. M. Kumar and Rajesh Ghanta," **Cloud based Structure Approach of Content-As-AService for Supplier Impartial of Mobile Gadgets**", *International Journal of Advanced Trends in Computer Science and Engineering*, Vol.3 , No.5, Pages : 282 - 287 , 2014
7. K R Remesh Babu, Amaya Anna Joy, Philip Samuel, "**Load Balancing of Tasks In Cloud Computing Environment Based On Bee Colony Algorithm**", *International Conference on Advances in Computing and Communications*, pp. 89-93,2015.
8. K. RAJA, G. SEKAR, "**An Algorithm for Credit Based Scheduling in Cloud Computing Environment Depending Upon Deadline Strategy**", *International Journal of Current Trends in Engineering & Research*, Volume 2 Issue 8, pp. 70 – 76, 2016.
9. Shubham Mittal, Avita Katal, "**An Optimized Task Scheduling Algorithm in Cloud Computing**", *IEEE 6th International Conference on Advanced Computing*, pp. 197-202, 2016.
10. Muhammad Fasih Akbar, Ehsan Ullah Munir, M. Mustafa Rafique, Zaki Malik, Samee U. Khan, Laurence T. Yang, "**List-Based Task Scheduling for Cloud Computing**", *IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 652-659, 2016.
11. Pratibha Pandey, Sarvpal Singh, "**Job Scheduling Techniques in Cloud Environment: A Survey**", *IEEE International Conference on Green Engineering and Technologies*, 2016. https://doi.org/10.1109/GET.2016.7916643
12. Fatemeh Azimzadeh, Fatemeh Biabani, "**Multi-Objective Job Scheduling Algorithm in Cloud Computing Based on Reliability and Time**", *IEEE 3th International Conference on Web Research*, pp. 96-101, 2017
13. Jagdeep Singh, Mr. Pawan Luthra, "**Enhanced Credit Based Load Balancing in Cloud Environment**", *International Journal of Computers & Technology*, Volume 15 Number 1 4, pp. 7444-7452, 2017. https://doi.org/10.24297/ijct.v15i14.5153
14. Abhikriti Narwal, Sunita Dhingra, "**Enhanced Task Scheduling Algorithm Using Multi-Objective Function for Cloud Computing Framework**", *Springer 3rd International Conference on Next Generation Computing Technologies*,2017.
15. Abhikriti Narwal, Sunita Dhingra, "**Task Scheduling Algorithm Using Multi-Objective Functions for Cloud Computing Environment**", *International Journal of Control theory and Applications*, pp.227-238, 2017.