International Journal of Applied Engineering Research ISSN 0973-4562 Volume 15, Number 4 (2020) pp. 393-404

Performance Analysis of CBLB Algorithm in Simulation and Real Cloud Environment

Kshama S B [1], Shobha K R[2]

[1,2]Dept. of Electronics and Telecommunication Engineering
Ramaiah Institute of Technology, Bengaluru, Karnataka-560054, India.

Abstract

The advantages of cloud computing over traditional computing has made it more popular among Information Technology (IT) industry. This popularity has forced organizations to move their business from traditional computing to cloud computing. This has increased the rate of cloud adoption leading to various critical issues in cloud usage. The distribution of enormous load among servers efficiently is one of those critical issues. This issue can be handled efficiently by using the load balancer for distributing enormous incoming load among servers. There are many categories of load balancing algorithms in cloud environment. We have proposed an algorithm in the general load balancing category named Capacity Based Load Balancing (CBLB) algorithm. In this paper, we have extended the earlier published CBLB work of ours by analyzing the performance of the algorithm for different group sizes, homogeneous and heterogeneous environment, and implemented in real Amazon Web Services (AWS) cloud platform. The performance of homogeneous, heterogeneous and AWS implementation is analyzed for the parameters, namely make span, throughput, average response time and CPU utilization.

Keywords: Cloud Computing, Load Balancing, Capacity Based Load Balancing, Virtual Machines, Cloudlets, CloudSim, AWS.

1. INTRODUCTION

The digital transformation in the Information technology (IT) has increased the requirement for heavy computing power, more IT infrastructure and a huge amount of storage. These requirements are supported by cloud computing. Cloud computing [1] is a technology that provides all these services to companies based on their requirements. Sometimes, companies need more infrastructures for their changed business strategy. In such a case, they need to invest on infrastructure and their maintenance. Cloud computing saves companies from this hassle. It provides a scalable service model, where companies can buy the required amount of resources and pay only for their usage. It also saves investment and maintenance expenditure. Cloud also provides more security on data by creating multiple copies of it and as well as increases the collaboration. The main goal of cloud computing is to provide cost-effective and dynamic IT services [2][3].

The loads received in cloud computing are dynamic in nature. A massive load may be received during peak hours, which may overwhelm a single server, affect service availability and response times in the system. On the other hand, fewer loads may be received during non-peak hours. The load balancer [4] plays a vital role in handling load distribution in scalable infrastructure. It maintains system firmness and improves performance by distributing load among available servers. It also ensures service availability and improves responsiveness.

The load balancing algorithms may be static or dynamic [5]. As dynamic loads are received in the cloud environment, the dynamic algorithms are well suited for cloud computing. There are many categories of load balancing algorithms [6]. The CBLB algorithm, proposed by us falls in the general load balancing category. In this paper, we have extended the performance analysis of the algorithm.

The rest of the paper is organized as follows: The related works are discussed in section 2. The performance of the CBLB algorithm for different group sizes is analyzed in section 3. The fitness of the algorithm in homogeneous and heterogeneous environments is analysed in CloudSim simulator as well as in real AWS cloud platform in section 4.

2. RELATED WORK

The need to distribute the load among resources in a cloud environment has initiated many researchers to work on load balancing algorithms [7]. The Round Robin (RR), throttled, Active monitoring and Equally Spread Current Execution (ESCE) are some of the popular dynamic general load balancing algorithms. The RR algorithm distributes the load among available server in a circular fashion. The disadvantage of this algorithm is that the allocation is done without checking the server load. This drawback can be overcome by Throttled algorithm. The throttled algorithm maintains a status table which indicates whether a Virtual Machine (VM) is available or busy. During allocation, the table is checked to get the first available VM. If any VM is available, the task is allocated to that VM, else it waits until a VM gets free. In this algorithm, if a recently allocated VM is available immediately, the task is allocated to that VM. This makes that VM to be utilized more and others to be underutilized. An enhanced Throttled Load balancer [8] is an improvement of the Throttled algorithm, which balance the load in case of load imbalance by migrating

load from over loaded VM to suitable under loaded VM. Active monitoring algorithm maintains a VM table which contains information about the status like throttled algorithm but considers the load of the VM as an additional factor. During allocation it monitors the VM table continuously to get this information in order to balance the load. The Modified Active Monitoring [9] algorithm is an improvement over Active Monitoring. In this algorithm, in the VM table in addition to status and load, the priority of VMs to handle the tasks is maintained. During task allocation the parameters in VM table is checked and the task is allocated to the VM which has less load with high priority.

Many researchers have tried to improve the existing algorithms or combine more than one algorithm to get better results. A hybrid load balancing algorithm [10] is attempted by combining RR, Throttled and ECSE algorithms to get their combined benefits. Initially, the tasks are processed using RR and are distributed among datacenters. As soon as the tasks arrive in the datacenter, they are allocated to the available VM using the Throttled algorithm. After allocation, the server utilization is monitored continuously by the ECSE. Upon exceeding the threshold of utilization, in order to balance the system, the load is moved to another server.

Some researchers have tried to balance the load efficiently based on the characteristics of tasks and VMs. The task-based load balancing algorithm [11] groups the tasks into two classes, i.e., upper and lower class to get the better performance. The group is done by comparing with the average of the tasks' lengths. Correspondingly, the VMs are also grouped into upper and lower class by comparing with the average of all VMs' MIPS (Million Instructions Per Second). Then, the lower- and upper-class tasks are assigned to the respective VM classes. Within each class, the task is assigned to more powerful VM (with respect to utilization). If two similar capacity VMs are found, the task is assigned on First-Come-First-Serve (FCFS) basis. Dynamic Load Balancing algorithm [12] is one more algorithm which uses characteristics of tasks and VMs for load balancing. In this algorithm the tasks and VMs are sorted based on length and processing capacity using bubble sort. Then tasks are assigned in the order of First-come-first-serve basis. After assigning tasks to VMs, the status of VMs are monitored to identify the overloaded and under loaded VMs. The identified over and under loaded VMs are sorted, and loads are moved from overloaded to under loaded machines. The optimal cost scheduling algorithm [13] has tried to improve the performance by grouping the user requirement into a single package and running that package on a single VM. Whenever the user needs more than one package, the user must process more than one VM. This costs more for the user as well as the service provider. This problem has been overcome in a package-based approach [14]. In this approach, instead of having a single package in a VM, multiple packages are placed in it. Whenever a user requires more than one package, the VM containing all the required packages are executed. This saves the cost of the user as well as the service provider.

Few researchers have tried to balance the load based on the profile of the resources. Profile-Based Effective Resource Utilization [15] is an algorithm which uses divide and conquer method to divide the requests into multiple queues. Similarly, VMs are divided into two groups based on the parameters, CPU utilization, response time and success rate of the node. Then based on some threshold values, the tasks are assigned to groups and on requirement auto scaling is done.

The CBLB [16] is a general load balancing algorithm, which balances the load based on the CPU utilization. The main objective of the algorithm is to allocate cloudlets (tasks) to VMs efficiently by reducing searching time and by improving resource utilization. The time to search for best suitable VM is reduced by grouping VMs based on their utilization capacity (UC) and arranging them in an array based on their utilization. This makes the less or not yet utilized VMs to be available at the first few positions. It reduces searching time for available VM as well as improves the processing time of cloudlets. In addition to this, in each group all the resources are utilized efficiently by placing the VMs in a list in the order in which they arrive to the group. During allocation, the cloudlets are allocated to the first VM in the group, which avoids a VM to be idle for a longer period and elude the allocation of tasks to the same VM again and again.

3. PERFORMANCE ANALYSIS FOR DIFFERENT GROUP SIZES

The CBLB algorithm, discussed in the previous section, used a group size of 11 to categorize the VMs based on their UC. In this section, the performances of the algorithm is checked for other group sizes 6 and 20 and are compared with the group size 11. In group size 11 each group holds the VMs whose UC is in the range of 10% each, i.e., 0-9%, 10-19% … 80-99%, 100% respectively. In group size of 20 each group holds the VMs whose UC is in the range of 5% i.e. 0-4%, 5-9%, 10-14%...90-94% and 95-100%. In group size 6 each group holds the VMs whose UC is in the range of 20% each, i.e. 0-19%, 20-39%, 40-59%, 60-79% and 80-99%, 100% respectively. To consider other group sizes with the UC ranges like 15%, 30%, in these ranges the groups with equal UC ranges is difficult to form. Therefore, the performance of the algorithm is tested only for the group sizes 6 and 20. In the algorithm, the UC of a VM is calculated by using the following formula [17]:

$$UC = L / ((PE * MIPS) + BW) \quad (1)$$

Where,

UC - Utilized capacity

L - Cloudlet length

PE - Number of processing elements of VM

MIPS (Million Instructions Per Second)- MIPS of VM

BW – Bandwidth

Table 1 Parameter Setup

Parameters		Values
Data centre (DC)	No. of DCs	10
	No. of hosts	5
	Host RAM (MB)	2048
	Host storage (MB)	1000000
	Band Width (MBPS)	10000
VM	Image size (MB)	10000
	Memory (MB)	512
	MIPS	1000
	Bandwidth (MBPS)	1000
	No. of processing elements (pes)	2
	VM monitor	Xen
Cloudlets	File Size (Bytes)	300
	Output Size (Bytes)	300
	No. of pes	1

The experiments are carried out in CloudSim 3.0.3 simulator [18] [19] [20], and the algorithm is written in Java. The comparison of group size 11 (CBLB-11) with group sizes 6 (CBLB-6) and 20 (CBLB-20) are done for the parameters make span, throughput and average response time. The experiments with different group sizes are carried out in a homogenous environment with MIPS 6000. The parameter setup for the experiment is given in Table I. The loads received in the cloud environment are dynamic. There may be a traffic burst during peak hours and less traffic during non-peak hours. The experiments are carried out to test the suitability of the algorithm for dynamic requests by varying number of cloudlets in steps of 100 from 100 to 1500. An assumption is made that all the cloudlets are independent of each other. The number of VMs considered here are 60. The performance of the algorithm is also checked for varied ranges of cloudlet lengths i.e.500-1000 (small), 500-2000 (moderate), 500-3000 (large). The graphs for make span against number of cloudlets for different cloudlet length ranges are shown in Fig.1. In the figure, we can notice the average performance of CBLB-20 for small and large cloudlet length ranges, and degraded performance for moderate cloudlet length range. On contrary, CBLB-6 has shown the average performance for moderate cloudlet length, and degraded performance for small and large cloudlet length ranges. Whereas, CBLB-11 has shown a better and stable performance than other group sizes for make span in all cloudlet length ranges.

Fig. 2 shows the graph for throughput against number of cloudlets. The figure shows better performance of the CBLB-11 in all the cloudlet length ranges. Whereas, CBLB-20 has shown the average performance for small and large cloudlet length ranges. But its performance has deteriorated for moderate cloudlet length ranges. On the other hand, compared to other two, CBLB-6 has given very less throughput for small and large cloudlet length range, and average throughput for moderate cloudlet length range. When overall performance is observed, we can notice the unstable performance of CBLB-6 and CBLB-20, and stable and better performance of CBLB-11.

Fig. 3 demonstrates the graphs for average response time versus number of cloudlets for different ranges of cloudlet lengths. In the graphs, we can clearly notice good performance of CBLB-20 than other two group sizes for small and moderate cloudlet length ranges, whereas CBLB-6 has given an average

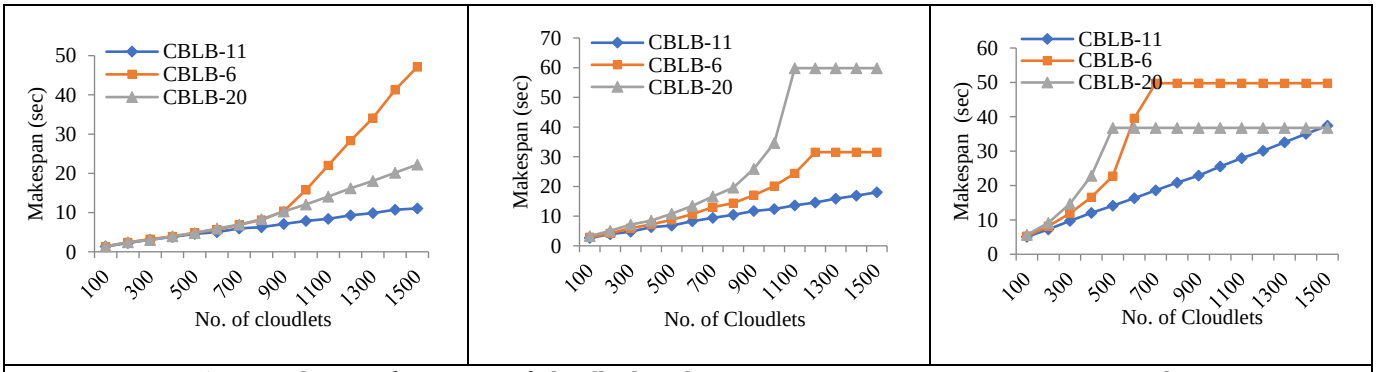


Fig. 1. Make span for a range of cloudlet lengths 500-1000, 500-2000, 500-3000 respectively

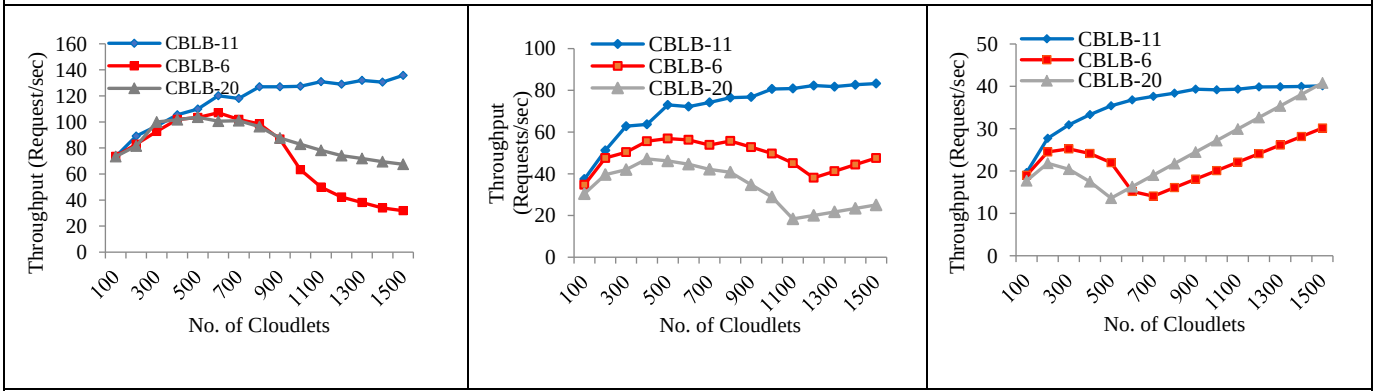


Fig. 2. Throughput for a range of cloudlet lengths 500-1000, 500-2000, 500-3000 respectively

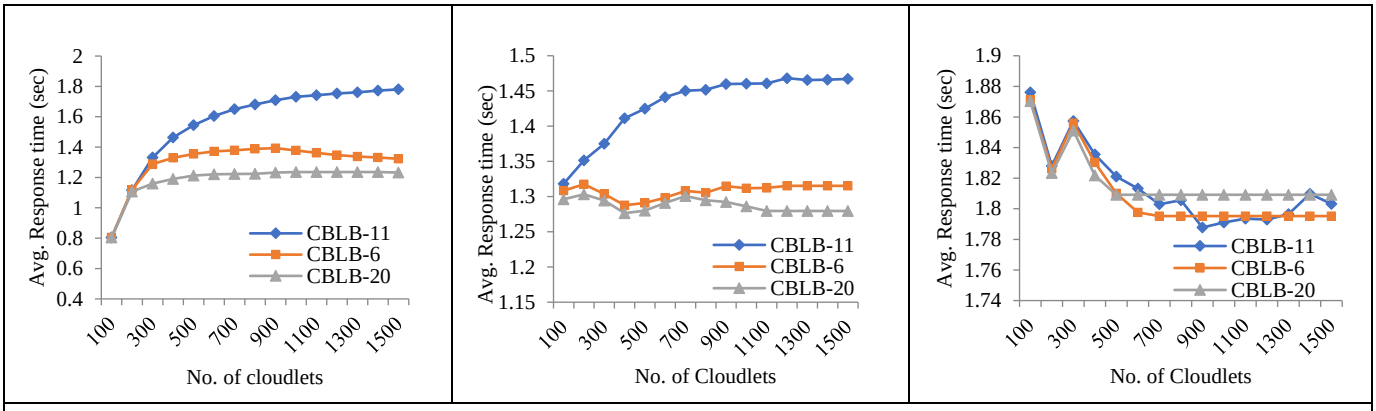


Fig. 3. Average Response time for a fixed range of cloudlet lengths 500-1000, 500-2000, 500-3000 respectively

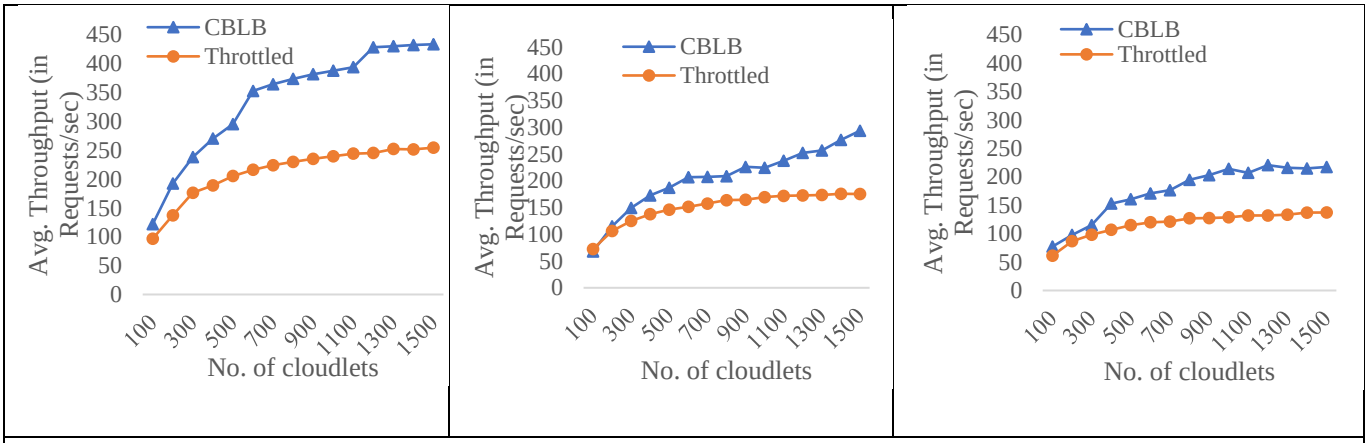


Fig. 4. Throughput for cloudlet lengths ranges 500-1000, 500-2000 & 500-3000 respectively for the homogeneous environment

performance and CBLB-11 has given a low performance. But, as cloudlet length range is increased, the algorithm has shown almost the same performance for all three group sizes. When overall performance of the algorithm is observed; the CBLB algorithm has given better and stable performance with the group size 11.

4. PERFORMANCE OF CBLB IN SIMULATION AND REAL CLOUD ENVIRONMENTS

In the earlier published paper, the CBLB was implemented in the CloudSim simulator with homogeneous environment. Some authors have proven that the throttled algorithm's better efficiency over other general load balancing algorithm [21][22]. Hence, the algorithm's performance is compared with the throttled algorithm. In this section, the performance of CBLB is further analysed for homogeneous and heterogeneous environments in both simulation and AWS real cloud environments.

4.1. Performance analysis in CloudSim simulator

Case 1: Homogeneous Environment

In our previous work, the performance of the CBLB was analyzed for the parameters makespan and average response time. The CBLB has given better make span and average response time than the Throttled. As an extension of this, the performance of the CBLB is checked for the parameter throughput. The parameter setup for the experiment is similar to Table 1. The number of VMs considered is 60, and each VM's MIPS is set to 5000. The readings are taken by varying the number of cloudlets to 100,200 … 1500 to check the performance of the CBLB for the dynamic load. The experiment is repeated for different cloudlet length ranges, 500-1000, 500-2000 and 500-3000, to check the behaviour of the CBLB for varied length cloudlets. The average of multiple readings is taken to get an accurate result. The result has given a better throughput than the Throttled in all three cases as shown in Fig. 4. The CBLB distributes cloudlets among all available VMs based on their utilization. This makes each VM to be utilized efficiently and resulted in better throughput of the algorithm.

In a cloud environment, the requests need to be distributed among multiple datacentres (DC) to achieve resiliency and reliability. The number of datacentres may vary based on customer requirements. The experiment is repeated for the data centre 10,20,30,40 to check the performance of CBLB. The number of VMs, MIPS and cloudlet length range are set to 60, 5000 and 500-2000 respectively. Fig. 5 shows the results for the parameters make span, average response time and throughput, respectively. The results clearly show the varied number of data centers has no effect on the performance of the

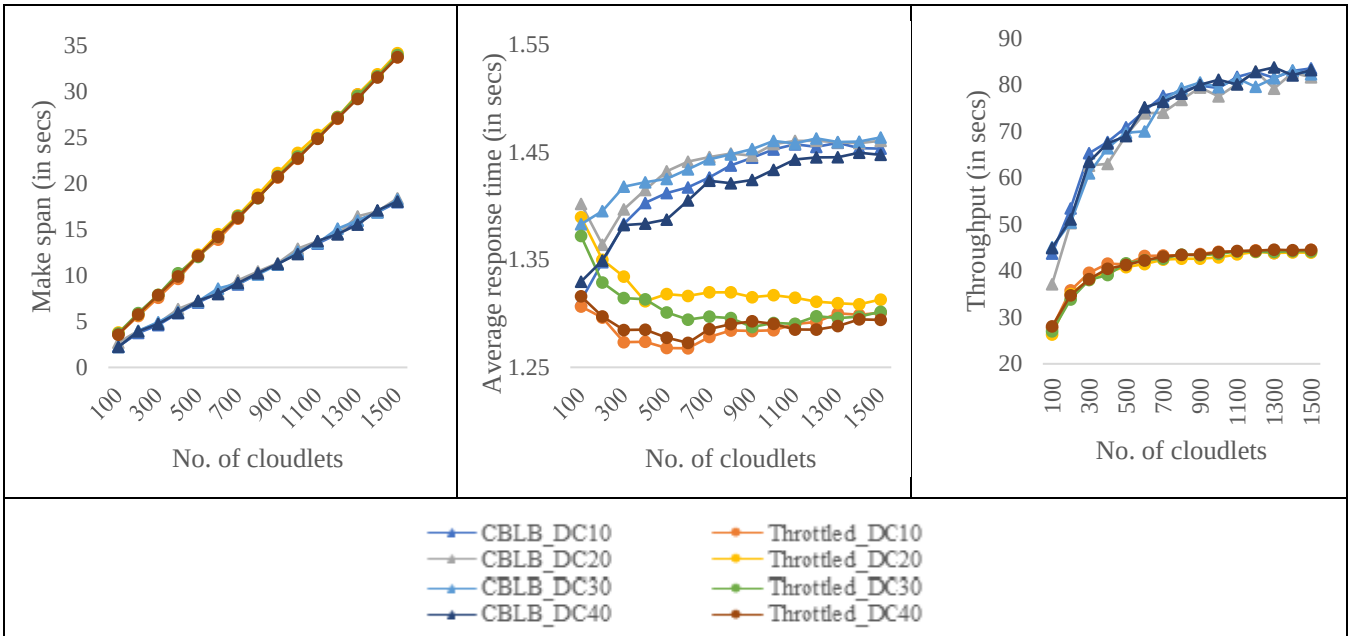


Fig. 5. Make span, average response time and throughput for varied number of DCs in homogeneous environment

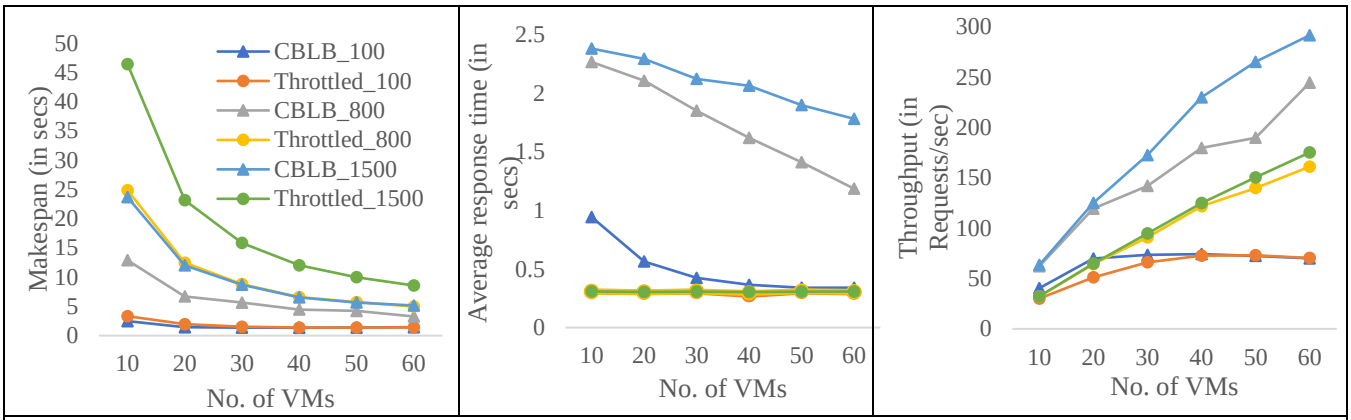


Fig. 6. Make span, average response time and throughput for varied number of VMs in homogeneous environment.

algorithm for the parameters make span and throughput. When average response time is considered, compared to CBLB, the Throttled has shown a small improvement which is negligible.

One of the major advantages of cloud computing is scalability, in which the number of VMs are varied based on the requirements. The number of VMs are scaled up during peak hours to handle a huge number of requests, whereas VMs are scaled-down to handle nominal requests during non-peak hours. The experiments are carried out by varying the number of VMs to 10, 20, … 60 to check the adaptability of the algorithm. In

this case the number of DCs, MIPS and cloudlet length ranges are set to 10, 5000 and 500-2000, respectively. Fig.6 shows the graphs for a varied number of VMs against makespan, average response time and throughput, for a varied number of VMs respectively. The graphs clearly depict the better make span and throughput of the CBLB. The increase in the number of VMs and cloudlets has shown an improvement in the performance of the algorithm. We can notice in the figure that the makespan of the CBLB for 1500 cloudlets is even lesser than the makespan of throttled for 800 cloudlets. Conversely, the throughput of CBLB for 800 cloudlets is more than the throughput of the Throttled for 1500 requests. But the average response time of CBLB is increased as the number of cloudlets is increased. However, compared to makespan, the increased amount of average response time is negligible.These results indicate that CBLB algorithm is scalable for large and small infrastructure setup.

The homogeneous cloud environment may contain high or low capability VMs. The cloud user may choose the VMs' capacity based on their requirement and budget. Hence, to check the adaptability of CBLB for such a condition, the experiment is carried out for changed capacity VMs. The capacity of VMs is varied by changing the MIPS to 1000, 2000 … 5000. The results for this variation is shown in Fig. 7. The behavior of CBLB is similar to the case of a varied number of VMs. It has given better performance for make span and throughput, and a very slight increase in the average response time for the increased number of cloudlets. When overall performance is observed, the CBLB has given a better performance than the throttled for all cases in homogeneous environment. This shows that the CBLB is suitable for homogeneous environment with low as well as high capability VMs.

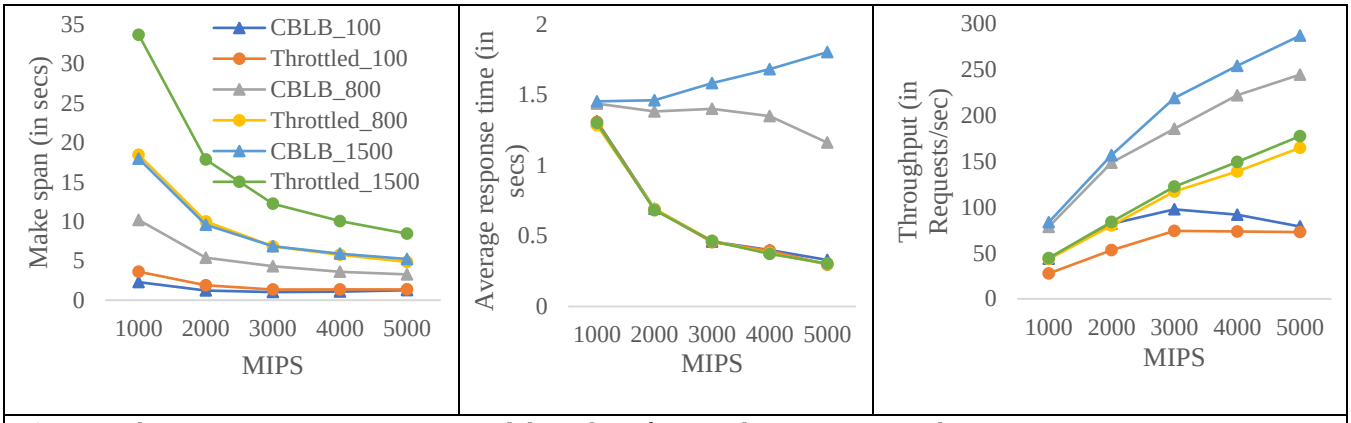


Fig. 7. Make span, average response time and throughput for varied capacity VMs in homogeneous environment

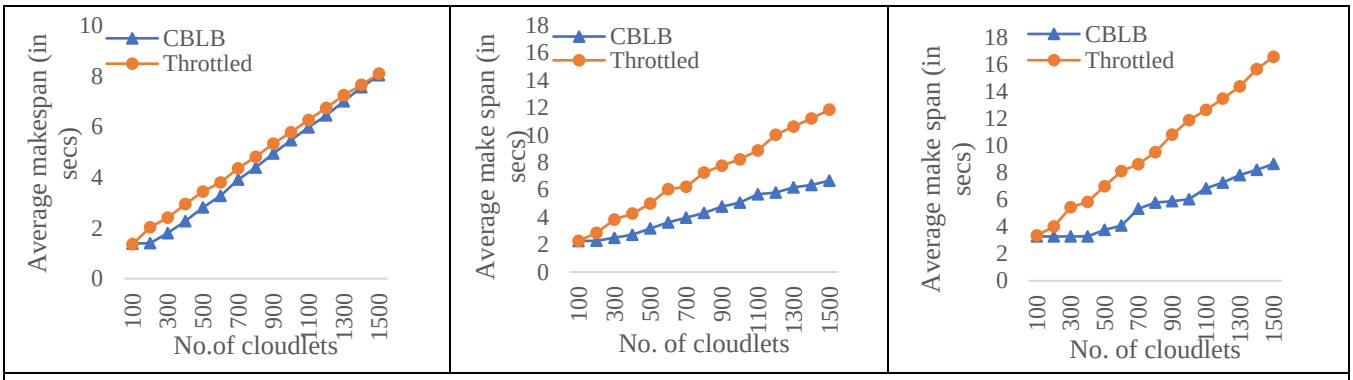


Fig. 8. Average make span for cloudlet lengths ranges 500-1000, 500-2000 & 500-3000 respectively for the heterogeneous environment

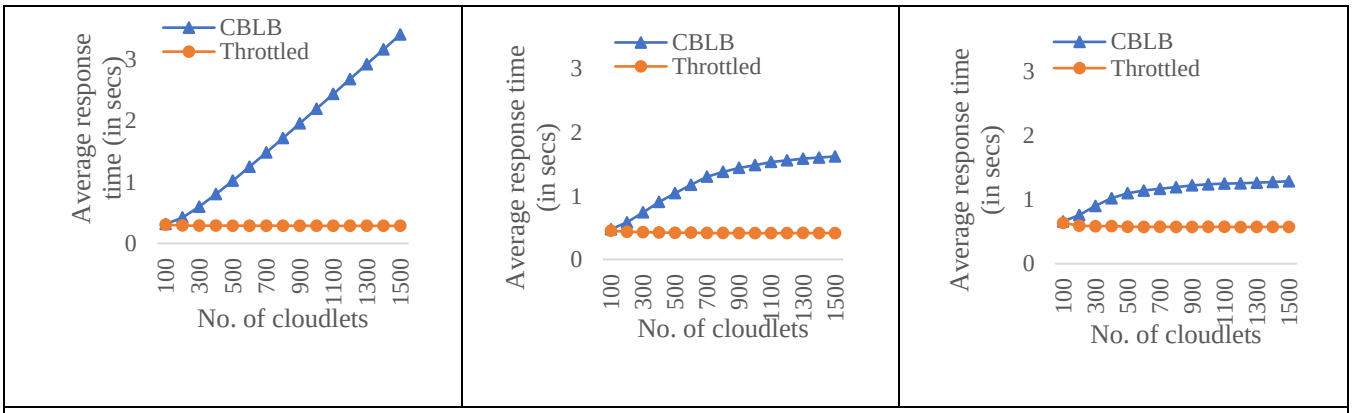


Fig. 9. Average response time for cloudlet lengths ranges 500-1000, 500-2000 & 500-3000 respectively for the heterogeneous environment

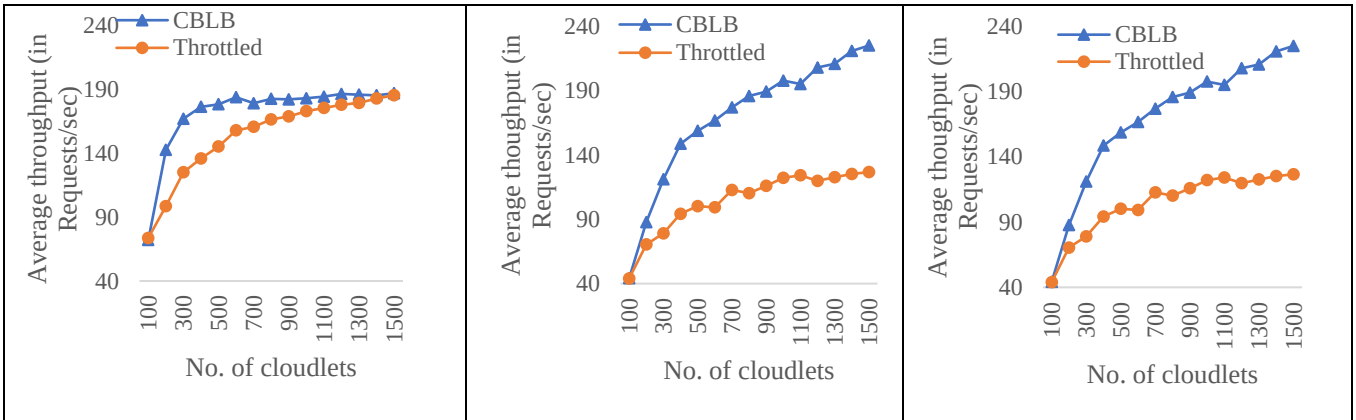


Fig. 10. Average throughput for cloudlet lengths ranges 500-1000, 500-2000 & 500-3000 respectively for the heterogeneous environment

Case 2: Heterogeneous Environment

The experiments in the heterogeneous environment are carried out with the evaluation parameter similar to that used in the homogeneous environment. The heterogeneous environment is set up by keeping VMs' MIPS values in the range 1000-5000. The performance of the CBLB is analyzed for the parameters average make span, average response time and average throughput. The results for average make span is shown in Fig. 8. In the figure, compared to the Throttled, we can notice a slight improvement in the make span of the CBLB for small cloudlet length range. However, as cloudlet length range and cloudlets are increased, the Throttled has taken more time to process the submitted cloudlets than the CBLB. In that case, the make span of the Throttled is almost twice the make span of the CBLB.

The average response time of the algorithms are compared in Fig. 9. In the figure we can observe that the average response time of CBLB is slightly more than the Throttled for small cloudlet length range. However, the increase in the cloudlet length range has reduced the difference between the algorithms' average response time. Although, the Throttled has given better result for average response time, the improvement is very less. Even for more number of cloudlets the improvement value is less than 2 secs. When makespan and throughput are considered, this value is insignificant.

Fig. 10 shows the graphs for average throughput. The CBLB has given the better average throughput for all cloudlet length ranges. Even though all ranges have shown better performance, the results of increased cloudlet length range are perceptible. In these cases, the throughput of CBLB is almost twice the throughput of the Throttled.

Similar to homogeneous, the performance of the CBLB is checked for varied number of datacentres (with no. of VMs=60) and varied number of VMs (with number of DCs=10) in the heterogeneous environment as well. The experiments are carried out for the cloudlet length range 500-2000. Fig. 11 shows the performance of the CBLB for a varied number of datacentres for the parameters make span, average response time and throughput, respectively. In these figures, we can notice the overlap of datacentre lines of similar algorithms. For any number of datacenter, the make span, average response time and throughput of both the algorithms are almost same. The varied number of DC as well the CBLB's make span and throughput is better and average response time is slight more than the Throttled. Hence, the varied number of DCs has no effect on the performance of the algorithm.

The make span, average response time and throughput of the CBLB for a varied number of VMs is depicted in Fig. 12. On comparing to the Throttled, for a varied number of VMs the algorithm has shown similar behavior as homogeneous environment. For the varied number of VMs, the make span

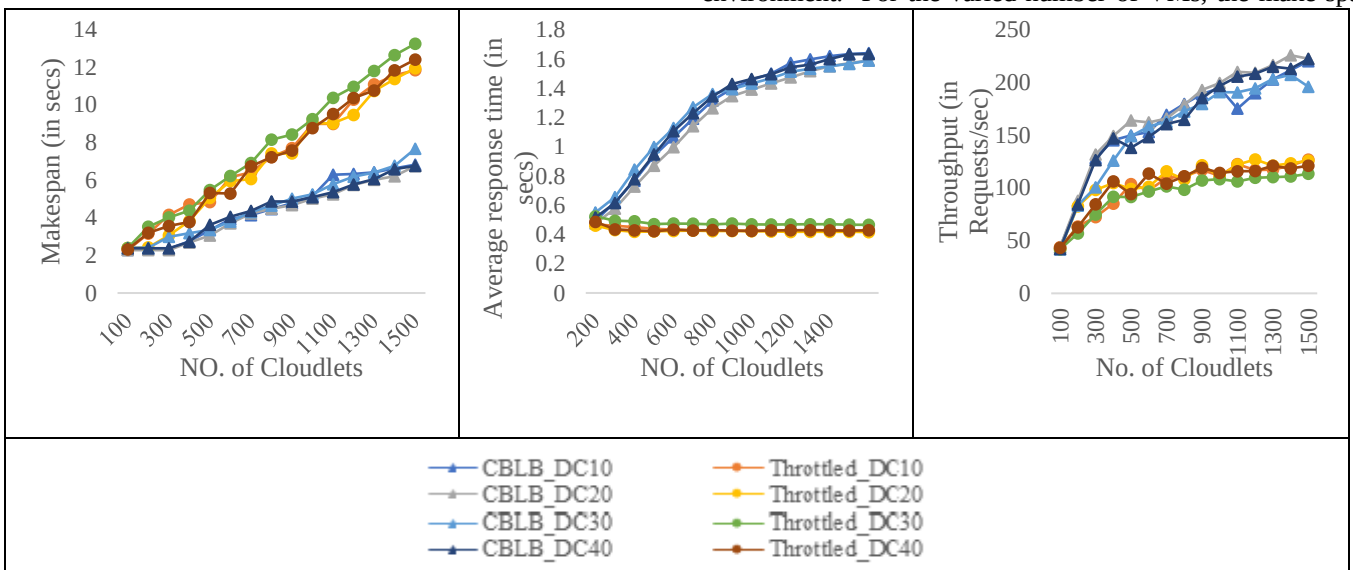


Fig.11. Make span, average response time and throughput for varied number of DCs in heterogeneous environment

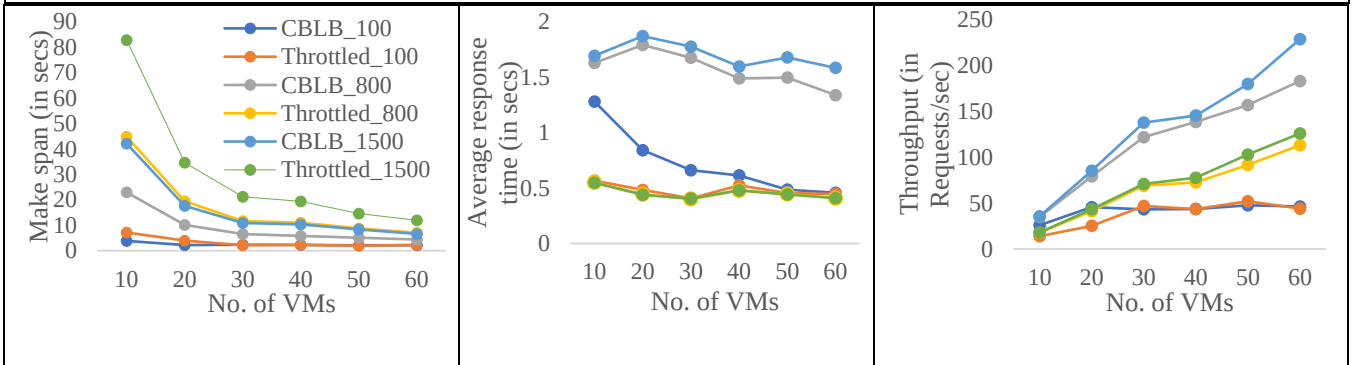


Fig. 12. Make span, average response time and throughput for varied number of VMs in a heterogeneous environment

and throughput of CBLB is better the throttled, whereas the average response time of Throttled is slightly less than the CBLB, which is imperceptible. Here as well the increase in the number of VMs and cloudlets has improved the makespan and throughput of the CBLB. This indicates the load balancing ability of the CBLB in the scalable heterogeneous infrastructure setup.

In the simulation environment, the CBLB has given a better performance in both homogeneous and heterogeneous environments. In both environments, the increased number of cloudlets and cloudlet length ranges has no effect on the performance of the CBLB. It has given good results for dynamic and longer length requests as well. The varied number of VMs and varied capability VMs also has not affected the performance of the CBLB. This has proved the suitability of the algorithm in the scalable cloud environment. These results show the best behavior of CBLB in both environments.

4.2. Performance analysis in the AWS cloud environment

Case 1: Homogeneous environment

The CBLB algorithm is implemented in AWS free tier platform

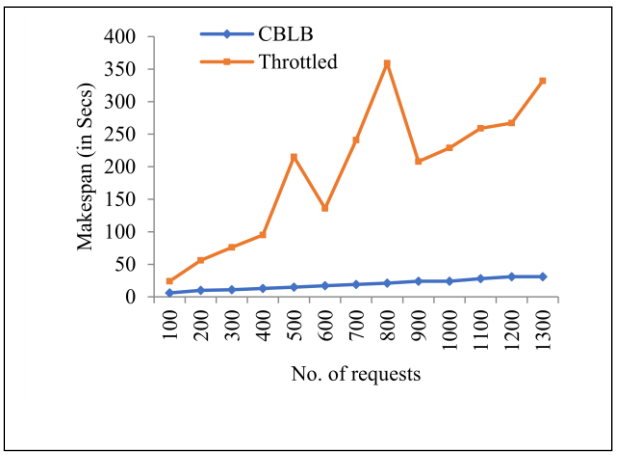


Fig. 13. Make span v/s No. of Requests for homogeneous AWS platform

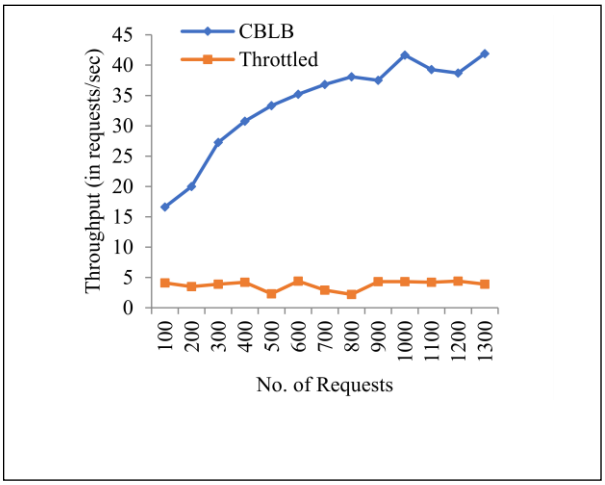


Fig. 14. Throughput v/s No. of Requests for homogeneous AWS platform

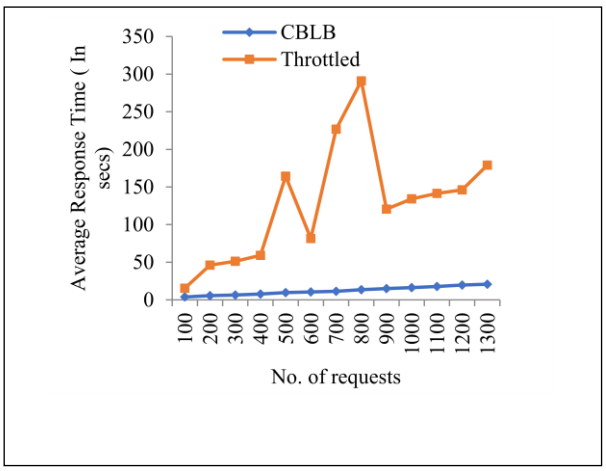


Fig. 15. Average Response Time v/s No. of Requests for homogeneous AWS platform

to check the performance of the algorithm in a real cloud environment, and the performance of the algorithm is checked for the parameters make span, throughput, average response time and CPU utilization. The algorithm is compared with the Throttled algorithm. The algorithms are written using Spring Boot java application in the Eclipse framework with the AWS Java tool kit. A dedicated Elastic Compute Cloud (EC2) is used as a load balancer. The load balancing jar files are run in this EC2 instance, and the different length requests are generated using REST API and are directed to this dedicated instance to distribute load efficiently among all available EC2 instances. The EC2 instances to which the load gets distributed is set with a storage capacity of 8GiB. The experiments were carried out for the requests 100, 200 … 1300 to check the capability of the algorithm to handle the dynamic load. These requests' lengths are in the range 400-3650.

Fig 13-15 show graphs for the parameters make span, throughput and average response time against the number of requests, respectively. In the figures, we can clearly notice the better performance of the CBLB algorithm than the Throttled algorithm. The make span of the CBLB algorithm is very much lesser than the Throttled algorithm. The algorithm has given a better throughput as compared to Throttled. As the number of requests is increased, the throughput of the CBLB is also increased. In the figures, we can also notice the stable performance of the CBLB algorithm for an increased amount of load, whereas there is a lot of variation in the performance of the Throttled. This indicates the robustness of the CBLB for the dynamic load in the cloud environment.

Fig. 16 and 17 shows the graphs for CPU utilization of the CBLB and throttled algorithms, respectively. These graphs are obtained from the AWS cloud watch metrics. The different colours indicate the EC2 instances with their instance ID for which graphs are plotted. In Fig. 16, we can observe almost equal utilization of all EC2 instances. This indicates the equal distribution of load among all available resources by the CBLB algorithm. Whereas in Fig. 17, we can notice more utilization of few EC2 instances and less utilization of remaining

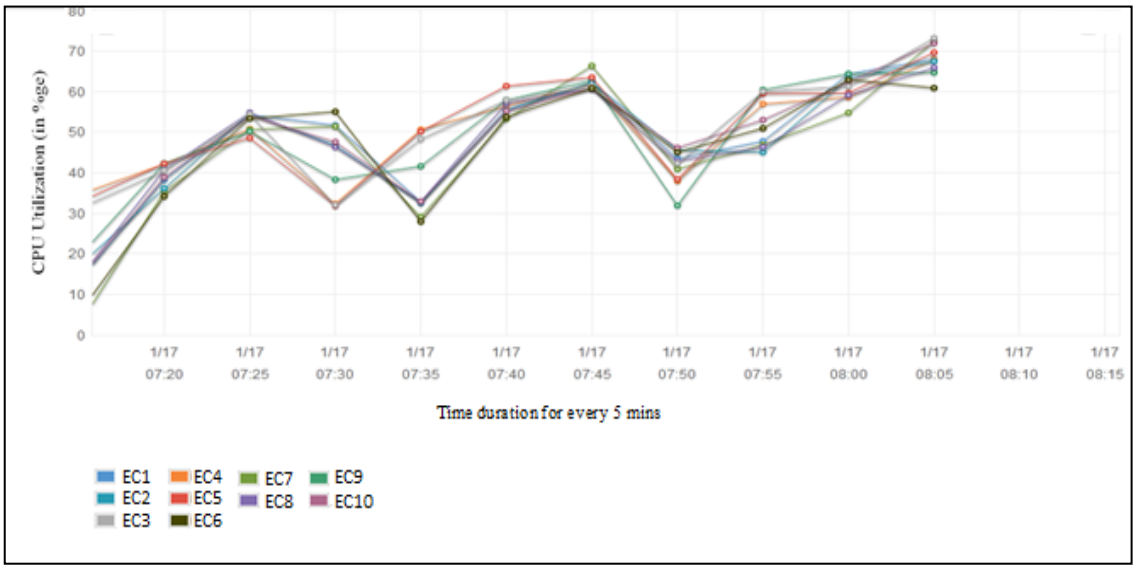


Fig. 16. CPU Utilization of CBLB for homogeneous AWS platform

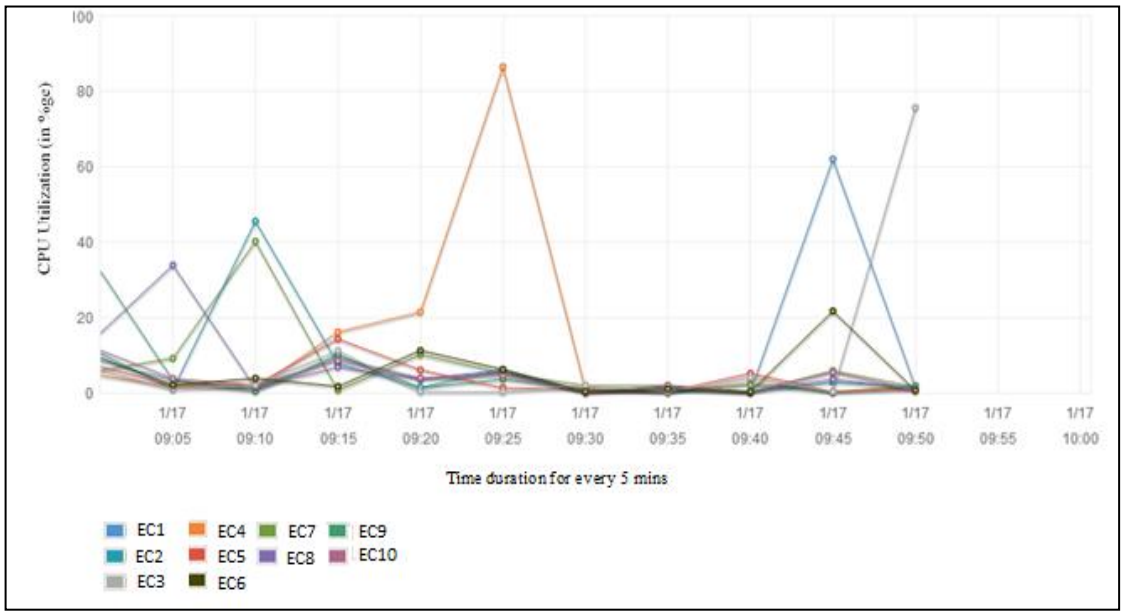


Fig. 17. CPU Utilization of Throttled for homogeneous AWS platform

instances. This indicates the repeated assignment of requests to a few EC2 instances (overutilization) and less/no requests to other instances (underutilization). This shows the unequal distribution of load among instances by the Throttled.

Case 2: Heterogeneous environment

The heterogeneous environment is set up in the AWS by varying the storage capacity of the EC2 servers. Each EC2 server has a capacity of 8,9 or 10 GiB. The experiments are carried out in a similar way as in the homogeneous environment. The readings are taken for the number of requests 100,200 … 1300 for the parameters make span, average response time, throughput and CPU utilization.

Fig. 18-20 shows the results for the parameters make span, throughput and average response time, respectively. The make span and average response time of throttled are more than the CBLB. As a result, the throughput of CBLB is very much higher than the Throttled. Fig. 21 and 22 shows the CPU utilization of the CBLB and Throttled for the heterogeneous environment respectively. Fig. 21 clearly shows the equal utilization of all the servers, whereas Fig. 22 shows the over utilization of few servers and underutilization of others. This proves the proper utilization of resources by the CBLB algorithm.

When the homogeneous and heterogeneous results are observed, the throttled has shown much variation in the homogeneous environment and stable performance in a heterogeneous environment. In spite of this, it is not capable of handling an increased number of requests efficiently. Its make span and average response time are very much higher than the CBLB. This resulted in a reduced amount of throughput for the

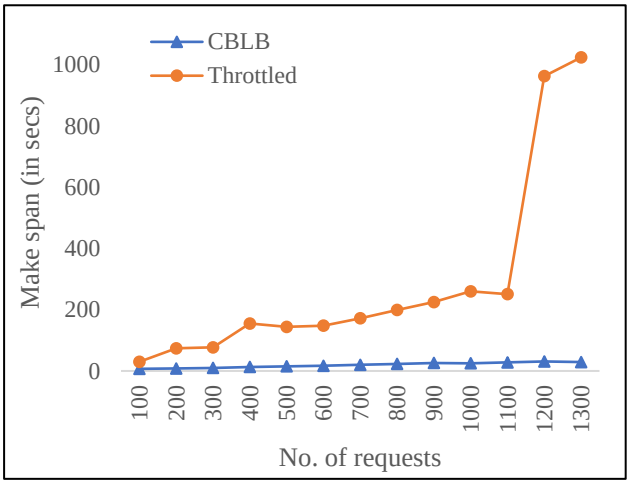


Fig. 18 Make span v/s No. of Requests for heterogeneous AWS platform

Throttled. Whereas CBLB has shown better and stable

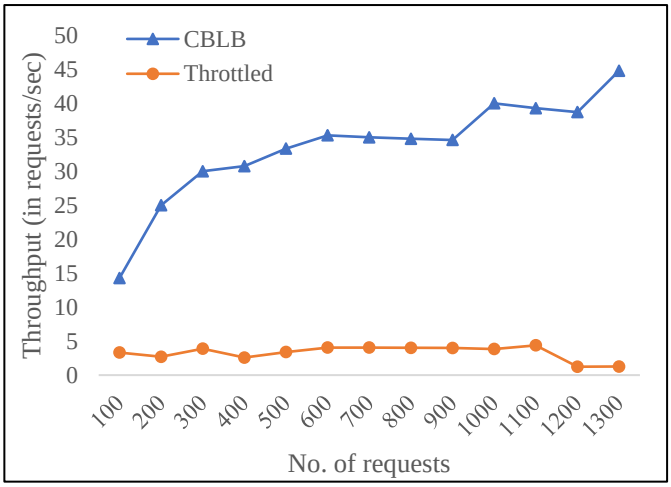
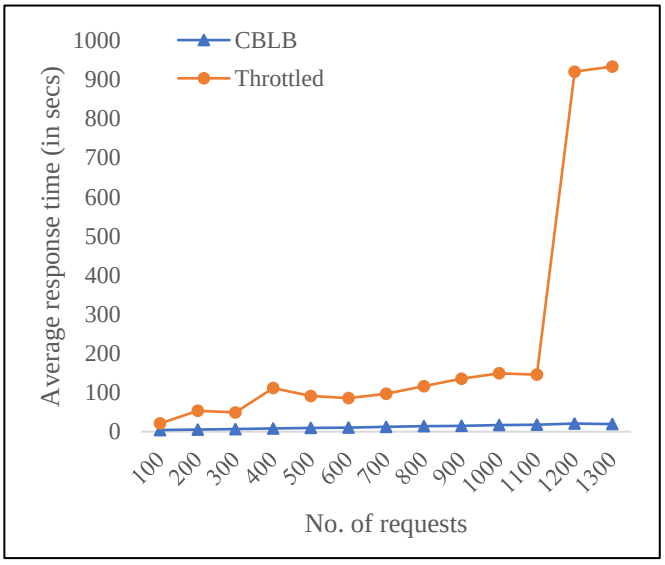


Fig. 19. Throughput v/s No. of Requests for heterogeneous AWS platform



Fig, 20. Average response time v/s No. of Requests for heterogeneous AWS platform

performance in both homogeneous and heterogeneous environments. The real cloud platform results as well supported the suitability of CBLB for both the environments

The similar behavior of CBLB can be observed in simulation, and AWS platform results for the parameter make span and throughput. There is an imperceptible amount of deterioration in the performance of the CBLB for parameter average response time in the simulation environment. The increased range of cloudlet length has shown the improvement in the performance of the CBLB for the parameter. In addition, [16] as well showed the improved average response time for the increased cloudlet length. Hence, the CBLB has given good results for average response time in the AWS platform.

5. CONCLUSION

The performance of the CBLB algorithm is analysed in the paper for group sizes 6 and 20, and the results are compared with CBLB-11. The parameters considered are make span, throughput and average response time. When overall performance is considered, the algorithm has given a good performance for group size 11. By keeping group size 11, the algorithm's performance is further analyzed for both homogeneous and heterogeneous environments in the simulator as well as in the AWS cloud platform. The previously published paper has proved the better performance of the CBLB in a homogeneous simulation environment for the parameters makespan and average response time. As an extension of this, the performance is checked for the parameter throughput. The algorithm has given good improvement for this parameter as well. The results obtained in heterogeneous environment shows that CBLB provides improved performance for make span and throughput with a slight increase in the average response time. The increase in the average response time is not very significant as make span and throughput are good. Along with this, the adaptability of the algorithm for resiliency and scalability of cloud environment is checked by varying number DCs, VMs and varying capacity of VMs. The varied number of DCs has no effect on the performance of the algorithm, whereas the varied number of VMs and VMs' capacity has shown the improved performance of the CBLB than the throttled.

The results of AWS implementation are also checked for the parameters, make span, average response time, throughput and CPU utilization. The real cloud implementation has also shown good performance for all these parameters. Hence, we can say th at CBLB algorithm supports dynamic load in both homogenous and heterogeneous environments and provides

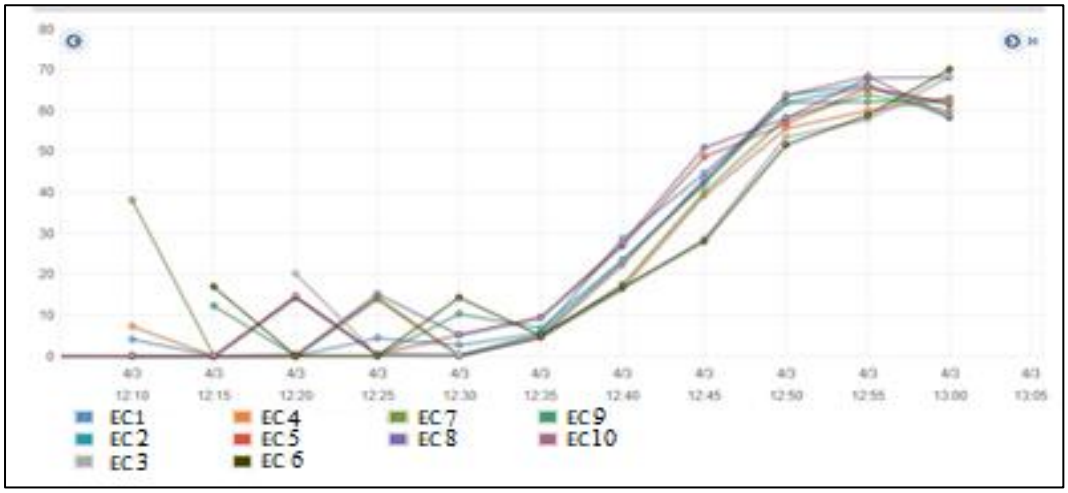


Fig. 21. CPU Utilization of CBLB for heterogeneous AWS platform

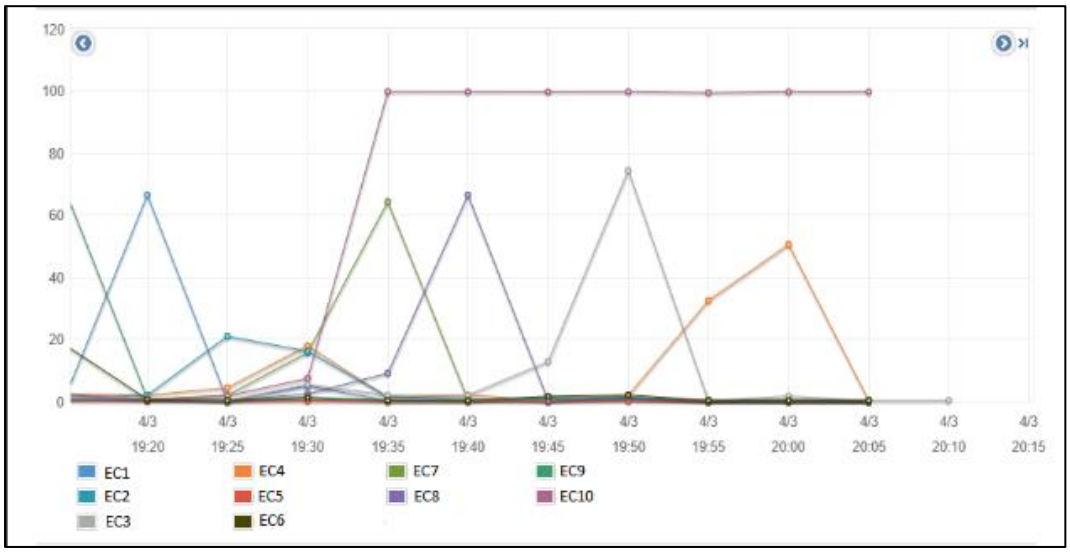


Fig. 22. CPU Utilization of Throttled for heterogeneous AWS platform

proper utilization of all the available resources.

REFERENCES

[1] Mell, Peter, and Grance, Tim.: "The NIST definition of cloud computing." Special Publication 800-145, 2011

[2] Armbrust, M., et al.: Magazine "A view of cloud computing". Commun. ACM Vol. 53, No. 4, pp. 50–58, 2010.

[3] https://aws.amazon.com/waht-is-cloud-computing/

[4] Pradhan A., Bisoy S.K., Mallick P.K., "Load Balancing in Cloud Computing: Survey", In: Sharma R., Mishra M., Nayak J., Naik B., Pelusi D. (eds) Innovation in Electrical Power Engineering, Communication, and Computing Technology. Lecture Notes in Electrical Engineering, vol 630, pp. 99-111, Springer, Singapore, 2020.

[5] Milani, A.S., Navimipour, N.J., "Load balancing mechanisms and techniques in the cloud environments: systematic literature review and future trends", J. Netw. Comput. Appl. Vol. 71, pp. 86–98, August 2016.

[6] Ghomi, E.J., Rahmani, A.M., Qader, N.N., "Load-balancing algorithms in cloud computing: a survey", J. Netw. Comput. Appl. Vol. 88(C), pp. 50–71, 2017.

[7] Arunima Hota, Subasish Mohapatra and Subhadarshini Mohanty, "Survey of Different Load Balancing Approach-Based Algorithms in Cloud Computing: A Comprehensive Review", H. S. Behera et al. (eds.), Computational Intelligence in Data Mining, Advances in Intelligent Systems and Computing 711, pp. 99-110, Springer 2019. https://doi.org/10.1007/978-981-10-8055-5_10

[8] G. J. Mirobi and L. Arockiam, "Dynamic Load Balancing Approach for Minimizing the Response Time Using an Enhanced Throttled Load Balancer in Cloud Computing," 2019 International Conference on Smart Systems and Inventive Technology (ICSSIT),

[9] Tirunelveli, India, pp. 570-575, 2019. doi: 10.1109/ICSSIT46314.2019.8987845

[10] Pawan Kumar, Rakesh Kumar, "Improved Active Monitoring Load-Balancing Algorithm in Cloud Computing", Proceedings of 2nd International Conference on Communication, Computing and Networking, Lecture Notes in Networks and Systems 46, 2019 https://doi.org/10.1007/978-981-13-1217-5_101

[11] Rani, S., Saroha, V., Rana, S., "A hybrid approach of round robin, throttle & equally spaced technique for load balancing in cloud environment", Int. J. Innov. Adv. Comput. Sci. (IJIACS), Vol. 6(8), pp. 2347–8616, 2017.

[12] Kaur, R., Ghumman, N.S.: "Task-based load balancing algorithm by efficient utilization of VMs in Cloud Computing", In: Aggarwal, V., Bhatnagar, V., Mishra, D. (eds.) Advances in Intelligent Systems and Computing, vol. 654. Springer, Singapore 2018. https://doi.org/ 10.1007/978-981-10-6620-7_7

[13] Mohit Kumar and S. C. Sharma, "Dynamic load balancing algorithm tominimize the makespan time and utilize the resources effectively in cloud environment", InternationalJournal of Computers and Applications, Vol. 42:1, pp. 108-117, 2020 DOI: 10.1080/1206212X.2017.1404823

[14] Shahapure, N.H., Jayarekha, P., "Load balancing with optimal cost scheduling algorithm", In: 2014 International Conference on Computation of Power and Energy, Information and Communication (ICCPEIC), IEEE 2014.

[15] Amanpreet Chawla and Navtej Singh Ghumman, "Package-Based Approach for Load Balancing in Cloud Computing", Big Data Analytics, Advances in Intelligent Systems and Computing, Vol. 654, pp. 71-77, https://doi.org/10.1007/978-981-10-6620-7_9

[16] Shah N.B., Shah N.D., Bhatia J., Trivedi H., "Profiling-Based Effective Resource Utilization in Cloud Environment Using Divide and Conquer Method", In: Fong S., Akashe S., Mahalle P. (eds) Information and Communication Technology for Competitive Strategies. Lecture Notes in Networks and Systems, vol 40, pp. 495-508, Springer, Singapore, 2019.

[17] Kshama S.B., Shobha K.R., 'A Novel Load Balancing Algorithm Based on the Capacity of the Virtual Machines', Advances in Computing and Data Sciences. ICACDS 2018. Communications in Computer and Information Science, vol 905, pp. 185-195, Springer, Singapore, 2018.

[18] Kimpan, W., Kruekaew, B., "Heuristic task scheduling with artificial bee colony algorithm for virtual machines", In: 8th International Conference on Soft Computing and Intelligent Systems and 2016 17th International Symposium on Advanced Intelligent Systems, pp. 281-286, 2016.

[19] Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A.F., Buyya, R.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw: Pract. Exper. Vol. 41, pp. 23–50, 2011. https://doi.org/10.1002/spe. 995

[20] Goyal, T., Singh, A., Agrawal, A.: Cloudsim: simulator for cloud computing infrastructure and modeling. Procedia Eng. Vol. 38, pp. 3566–3572, 2012.

[21] Nayyar, A.: The best open source cloud computing simulators (2016). Available: http://opensourceforu.com/2016/11/best-open-source-cloud-computing-imulators/

[22] Ahmad, E.I., Ahmad, E.S., Mirdha, E.S. "An enhanced throttled load balancing approach for cloud environment", Int. Res. J. Eng. Technol. (IRJET), 4(6) (2017). e-ISSN: 2395-0056

[23] Subalakshmi, S., Malarvizhi, N., "Enhanced hybrid approach for load balancing algorithms in cloud computing", Int. J. Sci. Res. Comput. Sci., Eng. Inf. Technol. IJSRCSEIT, 2(2) (2017). ISSN: 2456-3307