International Journal of Future Generation Communication and Networking
Vol. 13, No. 1, (2020), pp. 1366-1384

Enhanced Artificial Bee Colony Algorithm for Load Balancing in Cloud Computing Environment

Kshama S B[1] *, Dr. Shobha K R[2]

[1,2]*Dept. of Electronics and Telecommunication, Ramaiah Institute of Technology*
[1]*kshamasb08@gmail.com*, [2]*shobha_shankar@msrit.edu*

Abstract

IT companies are moving their business from traditional computing to cloud computing to leverage the benefits like reduced infrastructure cost, business agility, etc. This change has led to an increase in the number of cloud-based service users worldwide from 2.4 billion in 2013 to 3.6 billion in 2018. By the end of 2020, this number will be double of 2013 users. The increased number of users, in turn, has increased the number of accesses done to web servers. Increased number of dynamic requests can create problems like reduced system performance, system break down, inefficient utilization of resources, etc.; All these problems can be avoided by distributing the workload among all resources using the load balancer. In this paper, a bio-inspired load balancing algorithm named Artificial Bee Colony is enhanced and compared with the basic ABC algorithm. The enhancement is done by merging ABC with Capacity Based Load Balancing algorithm. The experiments are carried out in the CloudSim simulator as well as real cloud environment using Amazon Web Services (AWS) platform. The significance of the results is tested by using statistical tests. The performance of the proposed algorithm is analyzed for make span, throughput, response time and the average waiting time.

Keywords: *Cloud computing, Bio-inspired, Artificial Bee Colony, Capacity Based Load Balancing, Virtual Machines, Cloudlets, Utilization Capacity*

1. Introduction

Cloud computing is an emerging technology in which users can use the third party owned or private shared pool of configurable computing resources to store, process, and manage their data remotely. Cloud computing has many advantages over traditional computing like fulltime availability, low cost, scalability, flexibility, automated updates on software, security, mobility, increased collaboration, disaster recovery, etc. To get these benefits, slowly, the IT companies are moving their business elements to the cloud. According to the 2018 International Data Group (IDG) cloud computing study [36], 73% of organizations have a portion of their computing infrastructure or at least one application already in the cloud, and 17% have planned to do so within one year. 38% of respondents have shared that the IT department is hesitant to migrate 100% to the cloud platform. Currently, organizations are utilizing a mix of cloud delivery models in which the average environment is 53% non-cloud, 16% Infrastructure-as-a-Service (IaaS), 23% Software-as-a-Service (SaaS) and 9% Platform-as-a-Service (PaaS). The gradual increase in cloud adoption has made a massive increase in the number of cloud-based service users. Due to this increased number of cloud users, a computing resource may get plenty of user requests at any point of time, resulting in difficulty of handling resources. If this situation is not handled efficiently, it may also result in the system break down. The sinking feeling of the server being down or not accessible may lead to the service provider losing potential customers. The load balancer plays a vital role in balancing this massive amount of workload effectively.

ISSN: 2233-7857 IJFGCN

It provides system firmness, improved performance, and protection against system failures.

The cloud load balancing is an optimization technique that distributes workload and computing resources efficiently in a cloud computing environment. In addition to the proper distribution of workload, the load balancer provides the advantages like better performance, handle sudden traffic burst, flexibility and elasticity to the cloud environment.

The load balancer uses an efficient scheduling algorithm to distribute workload among the resources to provide all the above advantages to the system. The enterprises can achieve high-performance levels at a potentially lower cost with the help of Cloud load balancer than traditional on-premises load balancing technology. The resources in the cloud computing environment may be physical hosts or Virtual Machines (VMs). The workload is distributed by load balancer either among physical hosts or virtual machines.

The load balancing algorithms are classified into two kinds based on the system state: static and dynamic [1] [26]. The static load balancing is the approach in which load distribution decisions are made at the compile-time aiming to minimize communication delays. The load distribution decisions are made based on prior information about the system like memory, processing power, and performance before the commencement of execution. Hence, these static load balancing algorithms are easy to implement and have less overhead. But the disadvantage of these algorithms is that the algorithms are not flexible, i.e., the allocation decisions cannot be changed during the execution. Once the execution starts, the tasks are executed using the allocated resources. It is only suitable for the system that has low variation in the load. Whereas, dynamic load balancing approaches make allocation decisions based on the current status of the system. In such algorithms, allocation decisions are made during execution time. Hence, they are challenging to implement. These algorithms continuously monitor the load, and in the case of imbalance, the load is redistributed among all available resources. This causes an extra overhead in dynamic algorithms. Albeit with more overhead, as allocation decisions can be changed during execution time, these algorithms are more flexible. Due to this flexibility, dynamic load balancing is better suited for the system that has more variation in the load. When the cloud computing environment is considered, there is a high variation in the system load. Hence, dynamic load balancing is best suitable for the cloud computing environment.

The cloud load balancing algorithms are broadly classified into many categories [8]. Among these categories, we have considered the natural Phenomena-based load balancing algorithm. In this paper, a bio-inspired Artificial Bee Colony (ABC) algorithm is enhanced by merging with the Capacity Based Load Balancing (CBLB) algorithm. CBLB [23] is our concept, proposed in the category of general load balancing. Here, we have tried to combine algorithms of two different categories and utilize the advantages of both algorithms. For performance analysis of these algorithms, parameters like make span, response time, throughput, and average waiting time are considered.

The following sections are organized as follows. Some of the bio-inspired methods that are applied for load balancing in cloud computing are discussed in section 2. Section 3 explains the proposed enhanced ABC algorithm. The implementation details and experimental results are discussed in section 4. The conclusion of the paper is specified in section 5.

ISSN: 2233-7857 IJFGCN

2. Related Work

Nature/bio-inspired algorithms are inspired by the behavior of animals or birds to accomplish a task efficiently. They are mainly used to address highly complex problems [2]. Some of the popular bio-inspired algorithms are applied in many cloud computing problems like energy optimization, load balancing, task scheduling, cost optimization, etc., [28].

The Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Artificial Bee Colony (ABC) are the most adapted bio-inspired algorithms in cloud computing applications. These are adapted in many cloud computing applications and modified to get better results. GA with Local Search optimization (GA-LS) [5] is an example of a modified GA used in cloud computing for monitoring and scheduling of virtual machines. The main objective of the algorithm is to reduce memory usage and energy consumption during VM scheduling. This is achieved by using the local search-a heuristic method in the cross over operation of GA. The New GA (N-GA) [3] is another example of improved GA, which is used for task scheduling in a cloud computing environment. The GA is improved by adopting the elitism technique (a small portion of the fittest candidates is copied to the next generation without change) during initialization, employing the Heterogeneous Earliest Finish Time (HEFT) method during subtask initialization, and the method proposed by [34] during mutation operation. The algorithm has a better performance for make span and execution time, but has some logical problems like reachability, fairness, and deadlock.

The PSO algorithm is a population-based stochastic optimization technique, inspired by the movement of organisms in a bird flock or fish school. The improved PSO [9] is used for task scheduling in a cloud computing environment based on adaptive weight. It has shown better resource utilization and task completion time.

The ACO algorithm is inspired by the actions of ant colony, initially proposed to search for an optimal path in a graph. Later, it is diversified to solve many complex problems. The primary ACO has a lack of rapid adaptability, which increases execution time and decreases convergence time. This problem is overcome by MACO [27]. This is achieved by feeding a higher number of tasks to fast processing VMs and a smaller number of tasks to slow processing VMs.

The ABC algorithm is inspired by the intelligent foraging behavior of honeybee swarm, proposed for numerical optimization [16]. Later, it has been proposed for numeric function optimization and constrained optimization problems [4] [17][18] and proved that the performance of ABC is better than the other population-based algorithms [19][17]. Further, the performance of ABC is analyzed for multi-dimensional and multi-model numeric problems and it has shown improved performance compared to Differential Evaluation (DE) Algorithm, Evolutionary Algorithm (EA) and PSO [19] for those problems. The ABC is further modified for constrained optimization problems [20], and applied in different fields to solve complex problems [21] as well as for load balancing in a cloud computing environment.

The basic ABC model consists of three types of populations: Employee, Onlooker and Scout bees. In the algorithm, initially scout bees are sent to initial food sources. The selected scout bees become the employee bees, and they go to food sources and determine the nectar amount. This information is shared with onlooker bees through wangle dance. Based on this information, the onlooker bees calculate the probability value of the sources. Then onlooker bees are sent to the best probability value food

ISSN: 2233-7857 IJFGCN

sources. If the selected food source is empty, the bee is abandoned and it becomes a scout bee. The abandoned scout bee searches for a new food source randomly. The process is repeated until the requirements are met.

The major drawback of the basic ABC algorithm is that it can handle only light loaded nodes efficiently and is not capable of handling newly arrived requests efficiently, which may lead to system imbalance. This problem is overcome by the improved ABC [15]. In this algorithm, the iterative process is introduced at each node in the system to improve efficiency. Similarly, ABC is modified as an Interaction ABC algorithm (IABC) for better load balancing in the cloud computing environment [14] by ensuring the system balance for each selection. During the initial stage of task allocation, VMs are chosen randomly for tasks. If this selection keeps the system balanced, then the task distribution is done according to the selection. Else, the iteration formulation is used to search an efficient VM. The performance of IABC is better than ABC in case of a varying number of cloudlets with a constant number of VMs, varying number of VMs with a constant number of cloudlets, and a varying number of both VMs and cloudlets. In all three situations, the IABC's performance is better for all best, worst and average cases.

During the migration process of load balancing, delay in the execution of the high priority tasks may result in a response delay. In such cases, the priorities of tasks need to be considered with load balancing. The Honey Bee Behavior Inspired Load Balancing (HBB-LB) algorithm [7] is an ABC based algorithm that considers priorities of migrated tasks during execution. This algorithm has less task migration compared to dynamic load balancing algorithm and is having better performance than the Weighted Round Robin, First-in-First-out and dynamic load balancing algorithms with respect to execution time, make span and degree of imbalance. Even though the priorities of tasks are considered after migration, there may be a response delay due to the migration process in HBB-LB. This problem is overcome in the Modified bee colony algorithm [29] by considering the low priority tasks during the migration process. It has given a better performance for make span, and the number of tasks migration is reduced compared to the basic ABC.

The ABC algorithm is enhanced to Improved Efficient ABC (IE-ABC) algorithm [24] to provide Quality of Service (QOS). The QoS is provided by assigning a dedicated employee bee to a data center to update the information and by taking allocation decision based on the load of the VM. This elude the search for the best fitness value and reduce the completion time, cost and task migration. This algorithm has given a better performance than GA, ACO and ABC algorithms. One more enhanced ABC algorithm is the Discrete Artificial Bee Colony (DABC) [35] algorithm, which uses the discrete operators, namely flip, swap and slide. The steps are added in all the three stages of the ABC algorithm to reduce make span time and increase resource utilization. The algorithm has given better performance for make span and average resource utilization ratio.

The above-discussed algorithms are concerned for independent jobs. But the jobs received in the cloud may be dependent or independent. Workflow scheduling is one of the critical issues in cloud computing in which a set of jobs are dependent on each another. This issue is addressed in the ABC based workflow scheduling [10] algorithm. The algorithm's main objective is to reduce the waiting time of the user. It also considers the live migration process. In the algorithm, initially, the ranks of the jobs are defined by workflow scheduling using jobs' burst time. Then the minimum make span of all available queues is estimated. The queues with minimum make span are appended, and the queues with maximum make span are discarded. After scheduling, the ABC algorithm is used for the live migration process. In the

ISSN: 2233-7857 IJFGCN

algorithm, during workflow scheduling, the evaluation of the best-effort queue has resulted in the reduction of make span and execution cost. The algorithm has also achieved higher efficiency, better utilization, and less energy consumption. In the similar way, other nature-inspired algorithms are also used for workflow scheduling in cloud computing environment [30].

Along with GA, ACO, PSO and ABC algorithms, few other bio-inspired algorithms are also adapted for load balancing. A Flower Pollination-based scheduling (FPAS) [13] is used for task scheduling in a cloud computing environment. The tasks are considered as pollens, and VMs are considered as flowers. The local pollination and global pollinations are calculated by using fitness functions to get better solutions. The algorithm is having better performance than Round Robin, FCFS and GA algorithms for make span.

The Multi-Objective Cuckoo Search Optimization (MOCSO) [25] is one more bio-inspired algorithm based on Cuckoo Search Optimization. It is used for resource scheduling in a cloud environment. The Cuckoo Search Optimization is inspired by the obligate brood parasitism of some cuckoo species. The basic CSO has three rules. Among these rules, two rules are modified in MOSCO to integrate the requirements of multi-objective. It has shown better performance than multi-objective ACO, GA and PSO algorithms. The Bat algorithm is another bio-inspired, meta-heuristic algorithm for global optimization, which is inspired by the echolocation behavior of microbats. It is adapted for workflow scheduling in cloud computing environment [31] to minimize the transmission and computation cost incurred during execution. The bat algorithm has exhibited a high convergence rate and balanced load distribution. It has taken less iteration than PSO and Cat Swarm Optimization (CSO) (the algorithm generated by observing the behavior of cats). The execution time of this algorithm is lesser than the PSO algorithm.

To get more benefits of bio-inspired algorithms for cloud computing applications, few researchers have combined two or more bio-inspired algorithms or a bio-inspired algorithm with other category load balancing algorithms. Heuristic ABC (HABC) [22] is an algorithm, in which ABC is combined with the heuristic algorithms like First Come First Serve (FCFS), Largest Job First (LJF) and Shortest Job First (SJF) and performance is compared with each other. The heuristic LJF algorithm (ABC_LJF) has shown better performance than ABC and other two heuristic ABC algorithms. The Modified Particle Swarm Optimization (MPSO) and Modified Cat Swarm Optimization (MCSO) algorithms are together used for task allocation and resource allocation & management [32]. The Modified ACO (MACO) and Modified Bee Colony Optimization (MBCO) algorithms are used together for load balancing [33], whereas basic ABC and ACO are combined as Hybrid Artificial Bee and Ant Colony Optimization (H_BAC) [11] algorithm. In all the hybrid approaches, the algorithms have shown the better performance than the basic algorithms.

3. Proposed Enhanced Artificial Bee Colony Algorithm (ABC_CBLB)

In the previous section, we have discussed many population-based algorithms that are applied for load balancing in a cloud computing environment. [6] has proved that ABC algorithm provides better performance compared to other algorithms. This algorithm is used for load balancing in the cloud computing environment to allocate cloudlets to resources. The mapping of ABC parameters with the cloud environment is as shown in Table I.

We have tried to enhance the performance of the ABC algorithm by merging it with our previously proposed CBLB algorithm. CBLB is the algorithm that works

ISSN: 2233-7857 IJFGCN

based on the utilization of VMs. During allocation of cloudlets, the algorithm arranges VMs in the increasing order of their utilization in an array. This arrangement provides the less utilized and unutilized VMs at the first few positions and the highly utilized VMs at the later positions. This reduces the time required to search for the best suitable VM during allocation. One more advantage of this algorithm is that the cloudlets are allocated first to the VMs, which are idle for a longer time than the others. This improves resource utilization. To take these advantages in the ABC and to improve the performance of the algorithm, we have merged it with the CBLB algorithm.

Table I. Mapping of ABC algorithm parameters with a cloud environment

Honey beehive	Cloud environment
Honeybee	Cloudlet
Food Source	VM
Employee bees	Cloudlets allocated to VMs at the beginning
Onlooker bees	Cloudlets allocated based on prior allocation condition
Scout bee	Cloudlet removed from overloaded VM and allocated to another VM randomly

The CBLB algorithm is used in the ABC for selecting the best-fitted food sources during allocation. Initially, all the available food sources (VMs) are initialized as per the CBLB algorithm. In the CBLB, a group size of 11 is considered. Each group holds the VMs with utilization capacity (UC) range 0-9%, 10-19% … 90-99%, 100% respectively. In the ABC algorithm, during the employee bee's stage, the food sources are selected randomly. But in the ABC_CBLB, the food sources are selected from the array. Whenever the food sources are selected by employee bees from the array, the fitness of the bees are checked and are allocated to them. Once the allocation is done, based on their UC, the food sources are moved to the respective groups by the CBLB algorithm.

In the beginning, as all the food sources are available completely, the possibilities of employee bees to be allocated to the selected food sources are more. But in the case of onlooker bees, the food sources are selected based on the information provided by employee bees, and fitness calculated. If no fitness is found that selection is rejected, and those bees become scout bees. Again, a new food source is selected, and fitness is checked for allocation. This process is repeated until all bees fit into any of the food sources. In the onlooker bees' stage, if the selected food source is already utilized more, the possibility of a new bee fitting into it is very less. This increases the rejection rate. Whenever the rejection rate is high, the time taken for allocation is also more. This affects the overall processing time. Here, we have tried to reduce the processing time by reducing the rejection rate. This is accomplished by making the onlooker bees select the less utilized food sources first and increase the probability of bees to fit into the first few selected sources.

Once the employee bees are allocated, the onlooker bees select the food sources for processing. During this selection process, the food sources are chosen by onlooker bees based on the CBLB algorithm, and fitness of the selected bees are checked. The onlooker bees with fitness values less than or equal to 1 are allocated to the selected food sources. The rest are rejected, and they become scout bees. The scout bees select the food sources from the array. The search for food sources is done from the beginning of the array. Once the selection is made, the fitness of the

ISSN: 2233-7857 IJFGCN

bee is checked for the allocation. If no fitness is found, the process is repeated by scout bees until it fits into a food source.

In the onlooker bees' stage, the CBLB algorithm returns the less or not utilized VMs first. Hence, the probability of getting a fitness of less than or equal to 1 for onlooker bees are more. This reduces the number of scout bees, and most of the bees are allocated during the onlooker bees' stage itself. This reduces the time to select a new food source and to calculate fitness for the allocation. This speed up the allocation process and reduces the processing time. In the CBLB algorithm, in each group, the food sources are kept in a list in the order in which they arrive in the group. The newly arrived food source in the group is kept at the end of the list. During the selection process, they are selected based on a first-in-first-out basis. It avoids the selection of the same food source again and again.

Pseudo code of ABC_CBLB algorithm:

Input: Array (0-10), VMs (v0, v1...vm-1), Cloudlets (c0, c1 ... cn-1)
Output: Cloudlets allocated to VMs

1. Initialize the zeroth position of the array with a list of all VMs.
2. Allocate employee bees to VMs in the array and update the fitness values of employee bees by using fitness function:

$$fitness_j = \sum_{i=0}^{K} cloudlet\ length_{ji} / capacity\ of\ VM_j$$

Where,
fitnessj=fitness of VMj
cloudlet lengthji= Length of cloudlet allocated to VM
capacity of VM =processing capacity of VMj
The processing capacity of VMj is calculated by using the formula:

$$capacity\ of\ VM_j = num\_of\_pes_j \times MIPS_j + BW_j$$

Where,
num_of_pesj= Number of processing elements of VMj
MIPSj=Million Instruction Per Second of VMj
BWj=Bandwidth of VMj

3. Move the positions of allocated VMs based on fitness values.
4. The best food sources are selected by the onlooker bees by calculating the fitness. The fitness is calculated by adding the length of cloudlet to be allocated to the total number of cloudlet length of VM:

$$fitness_j = \frac{(\sum_{i=0}^{K} cloudlet\ length_{ji} + current_{cloudlet_{length}})}{capacity\ of\ VM_j}$$

5. Allocate the best selected VMs to the onlookers. Move the positions of allocated VMs in the array based on their fitness values.
6. Select the food source for abandoned bees from the array. If the allocation is done, then move the positions of VMs based on fitness values.
7. Repeat step 2 until all the cloudlets are allocated.

The above pseudo-code contains two fitness functions. The first fitness function is to calculate the fitness after allocation, and another fitness function is to calculate

ISSN: 2233-7857 IJFGCN

the fitness before allocation. In the proposed algorithm, M number of VMs, N number of cloudlets and a single-dimensional array of size 11 are taken as inputs. As an output the cloudlets are allocated to VMs in an efficient way.

4. Implementation and Experimental Results

The implementations are initially carried out in CloudSim 3.0.3 simulator [12], which provides a framework for the cloud environment. The algorithms are implemented using Java. The performance of the ABC_CBLB is compared with the ABC w.r.t four performance metrics make span, average response time, average waiting time and throughput. Then the significance of the results is tested by statistical analysis. Further, the performance of the algorithm is checked on real time cloud platform, Amazon Web Services (AWS).

4.1. Implementation of ABC_CBLB in CloudSim simulator

In the simulator, the proposed algorithm's performance is compared with the basic ABC. The parameter setup for the experiment is given in Table II. To check the performance of the algorithms in homogeneous and heterogeneous environments, the experiments are carried out in both the environments. For the homogeneous environment, the MIPS are kept at 5000 for all the VMs, whereas for heterogeneous environment, the MIPS of VMs are varied in the range 1000-5000.

Table 2. Parameter setup for ABC_CBLB implementation

Datacentre (DC) parameters	MIPS	6000
	No. of hosts	5
	Host RAM (MB)	10240
	Host storage (MB)	1000000
	Band Width (MBPS)	20000
VM parameters	Image size (MB)	10000
	Memory (MB)	512
	Bandwidth (MBPS)	10000
	No. of processing elements	2
	VM monitor	Xen
Cloudlet Parameters	File Size (Bytes)	300
	Output Size (Bytes)	300
	No. of pes	1

Case 1: Homogeneous Environment

To check the adoptability of our ABC_CBLB algorithm for dynamic cloud environment, the experiment is repeated for the different number of cloudlets, i.e., 100,200..., 1500 by keeping the cloudlet length (Number of instructions of a request to be processed in CPU) range constant. In addition to this, to check the performance of the algorithm for varying (small/large) length requests, the experiment is conducted for different cloudlet length ranges, i.e., 500-1500, 500-5500 and 500-10500. By varying the number of requests and the cloudlet length range, the performance of the algorithm is observed for all the four performance parameters. The results of several iterations are averaged and tabulated for better accuracy. These experiments are carried out using the parameter values indicated in Table II.

ISSN: 2233-7857 IJFGCN

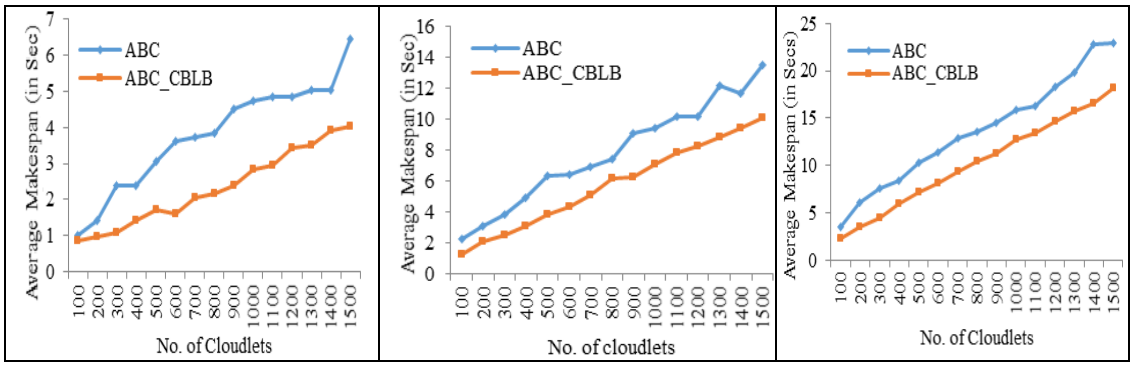

Figure 1. Makespan for cloudlet lengths ranges 500-1500, 500-5500 & 500-11000 respectively for the homogeneous environment

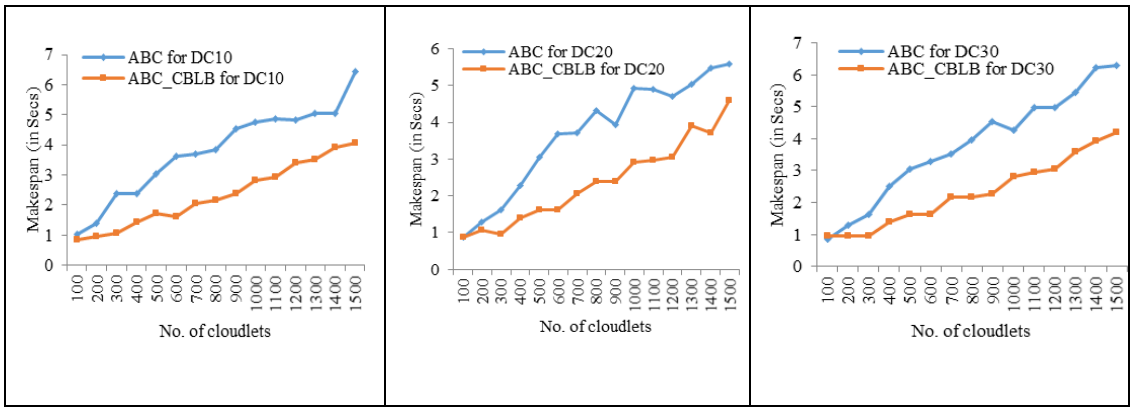

Figure 2. Make span for the number datacentres (DC) 10, 20 and 30 in homogeneous environment

Figure 1 shows the performance analysis of the algorithm for variable load and different cloudlet length ranges w.r.t. average make span. In this case, the number of datacentres, no. of VMs and MIPS of VMs are 10, 60 and 5000 respectively. The figure contains the graphs plotted for average make span against the number of cloudlets (requests) for three cloudlet length ranges, respectively. The results have shown that the ABC_CBLB has taken less time to process the set of submitted jobs than the ABC for all the cloudlet length ranges. This shows that the proposed algorithm handles dynamic requests more efficiently than the ABC.

During the performance analysis of the make span parameter, in addition to the dynamic load, experiment is carried out to test the effect of varied number of datacentres on the performance of ABC_CBLB. The readings are taken for the number of datacentres 10, 20 and 30. The number of VMs and MIPS considered are 60 and 5000 respectively. The results for different datacentres are shown in Figure 2. The graphs are plotted for the total processing time of the set of jobs against the number of cloudlets. The graphs indicated that the proposed algorithm has lesser make span compared to ABC. This proves that the algorithm supports the scalability of infrastructure.

ISSN: 2233-7857 IJFGCN

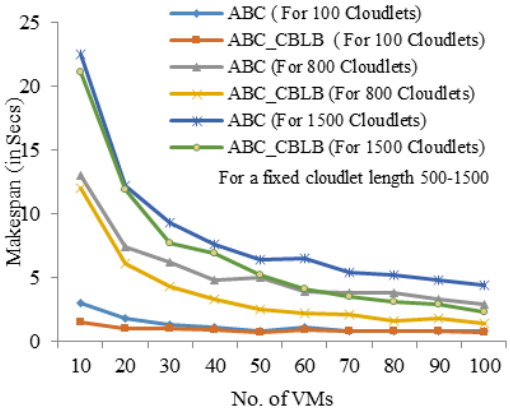

Figure 3. Make span for different number of VMs

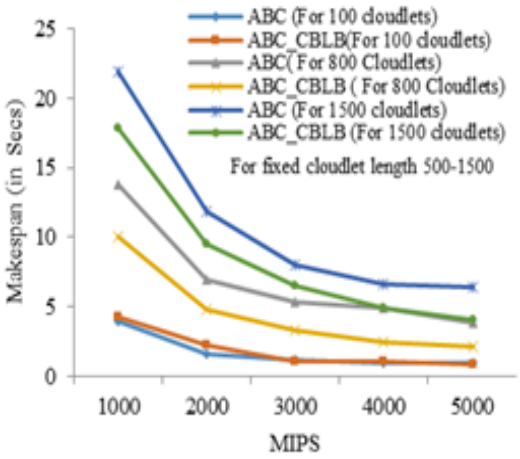

Figure 4. Make span for varied capability of VMs

In the cloud environment, a huge number of requests is received during peak hours. In this situation, the number of VMs is increased to handle these requests efficiently. During non-peak hours, only a few numbers of VMs can handle the incoming requests efficiently. Consequently, the number of VMs is reduced for non-peak hours. Hence, to check the performance of the algorithm for varied number of VMs, the experiment is repeated for different number of VMs, i.e., 10, 20 … 100. All the VMs' MIPS are set to 5000. The number of datacentres considered is 10. Figure 3 shows the plot for the number of VMs against make span. The number of cloudlets considered here are 100, 800 and 1500. The graphs clearly denote that the ABC_CBLB has lesser make span than the ABC for fewer VMs. When the number of VMs is increased, both the algorithms have the same make span for 100 cloudlets. But for the cloudlets 800 and 1500, the make span of the ABC_CBLB is very much lesser than the ABC. In the graphs, we can clearly notice that the make span of the ABC_CBLB for 1500 cloudlets is even lesser than the make span of the ABC for 800 cloudlets. During both peak and non-peak hours, the ABC_CBLB has given less make span. This indicates that the ABC_CBLB is capable of handling requests in a scalable infrastructure more efficiently than the ABC.

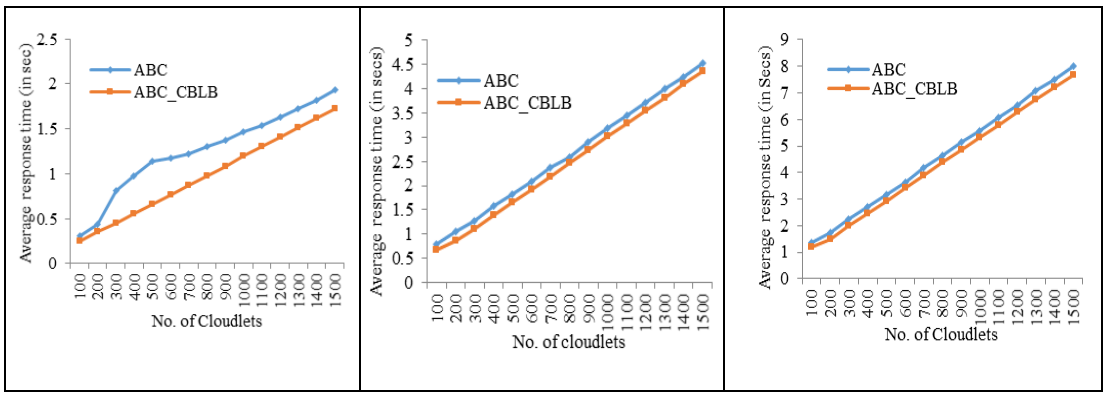

Figure 5. Average response time for a fixed range of cloudlet lengths 500-1500, 500-5500 & 500-10500 in homogeneous environment

ISSN: 2233-7857 IJFGCN

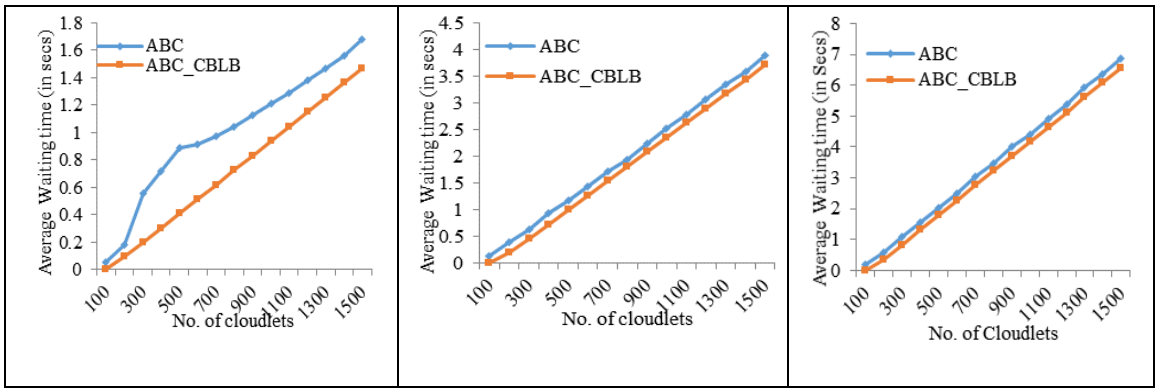

Figure 6. Average waiting time for a fixed range of cloudlet lengths 500-1500, 500-5500 & 500-11000in homogeneous environment

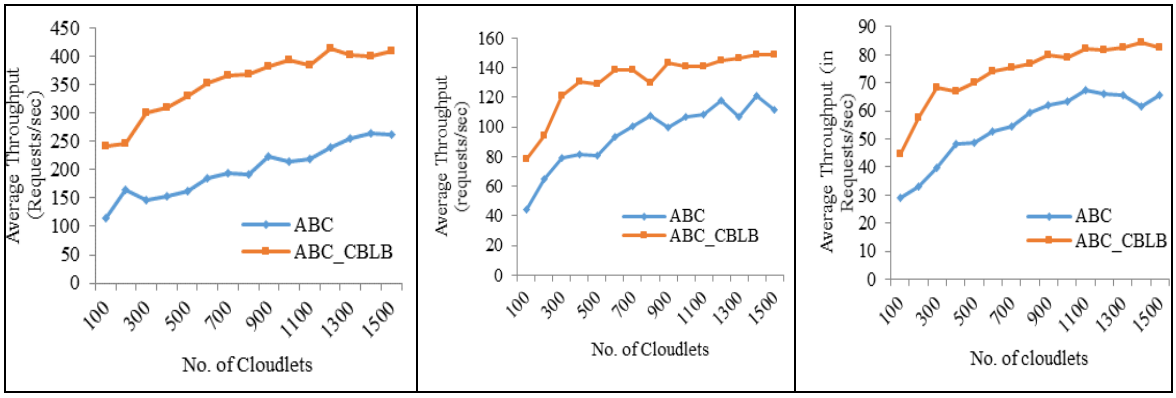

Figure 7. Average throughput for a fixed range of cloudlet lengths 500-1500, 500-5500 & 500-11000in homogeneous environment

The performance of the algorithm is checked for different capacity VMs as well. For this, the readings are taken for the VMs with MIPS 1000, 2000 … 5000. The no. of VMs and datacentres considered are 60 and 10 respectively. The graphs are plotted for MIPS against make span in Figure 4. The number of cloudlets for which the graphs are plotted are 100, 800 and 1500. The graphs indicate that the ABC_CBLB algorithm gives almost the same make span as that of ABC for the low capacity VMs, but as the capacity increases, the performance of ABC_CBLB is better than that of ABC. This clearly shows that the algorithm is suitable for both low capacity as well as high capacity cloud environments.

The results for average response time, average waiting time and throughput are shown in Figure 5,6,7, respectively. The readings are taken with the same parameter's setup as in Figure 1. The graphs are plotted for average response time, average waiting time and throughput against the number of requests. The graphs in Figure 5 and 6 show a similar kind of performance of both the algorithms with respect to average response time and average waiting time. The ABC_CBLB has shown improved performance than the ABC for small cloudlet length range. But there is a minor improvement in the performance for increased cloudlet length range. When throughput is considered, the ABC_CBLB has processed many cloudlets per second than the ABC for all three cloudlet length ranges. The increased cloudlet length range has not affected the performance of the algorithm.

ISSN: 2233-7857 IJFGCN

Case 2: Heterogeneous Environment

Like the homogeneous environment, the experiments are carried out for the heterogeneous environment also. Fig. 8 shows the graphs for the average make span in a heterogeneous environment. Here also the performance of the algorithm is checked for dynamic load and increased cloudlet length ranges. When the load is increased, the ABC_CBLB algorithm has given a better performance than the ABC. The make span of the ABC_CBLB algorithm is very much lesser than the ABC. When the cloudlet length range is increased, there is a large variation in the performance of the ABC algorithm, whereas the ABC_CBLB algorithm has given a stable performance for the increased ranges of cloudlet length. These results indicate that the ABC_CBLB algorithm is capable of handling dynamic requests in the heterogeneous environment compared to basic ABC algorithm. It has maintained the stability for both light loads as well as heavy loads requests.

The results of average response time and the average waiting time are plotted in Fig. 9 and 10, respectively. The performance of the algorithm in a heterogeneous environment is same as the homogeneous environment for these parameters. It has given better performance for a small range of cloudlet length. But the improvement in the performance of the ABC_CBLB algorithm is in small-scale for the increased

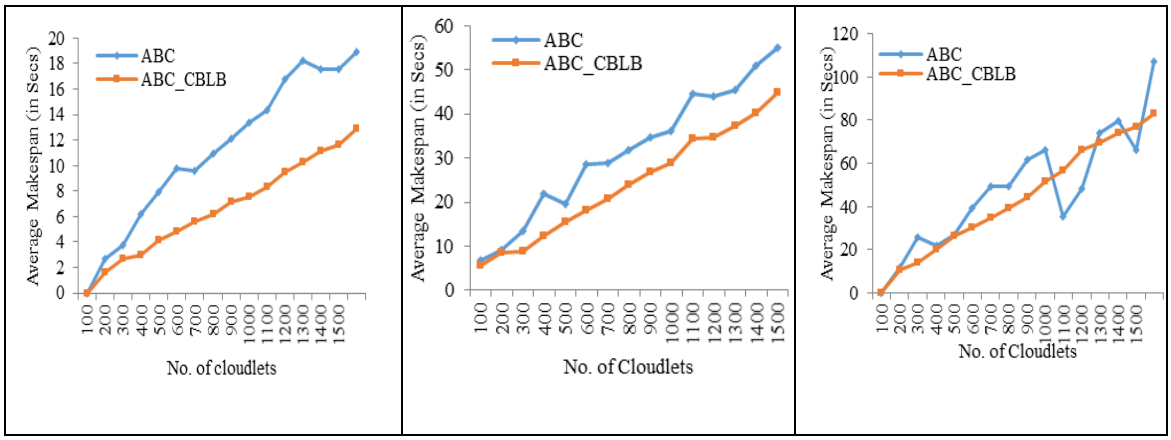

Figure 8. Average Make span for a fixed range of cloudlet lengths 500-1500, 500-5500 & 500-11000 respectively in heterogeneous environment

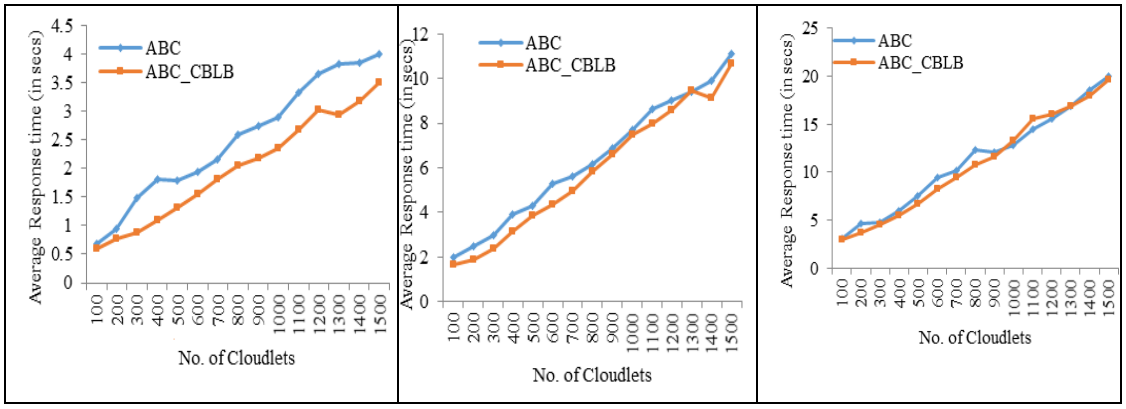

Figure 9. Average response time for a fixed range of cloudlet lengths 500-1500, 500-5500 & 500-11000 for the heterogeneous environment

ISSN: 2233-7857 IJFGCN

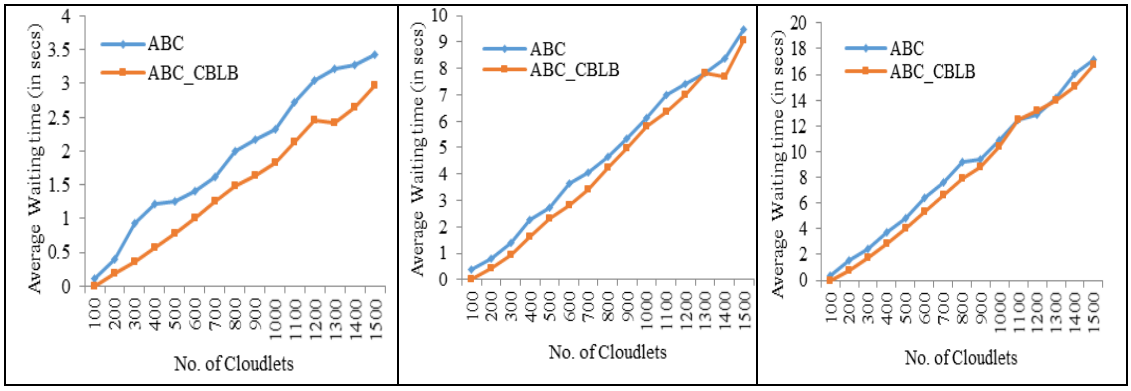

Figure 10. Average waiting time for a fixed range of cloudlet lengths 500-1500, 500-5500 & 500-11000 for the heterogeneous environment

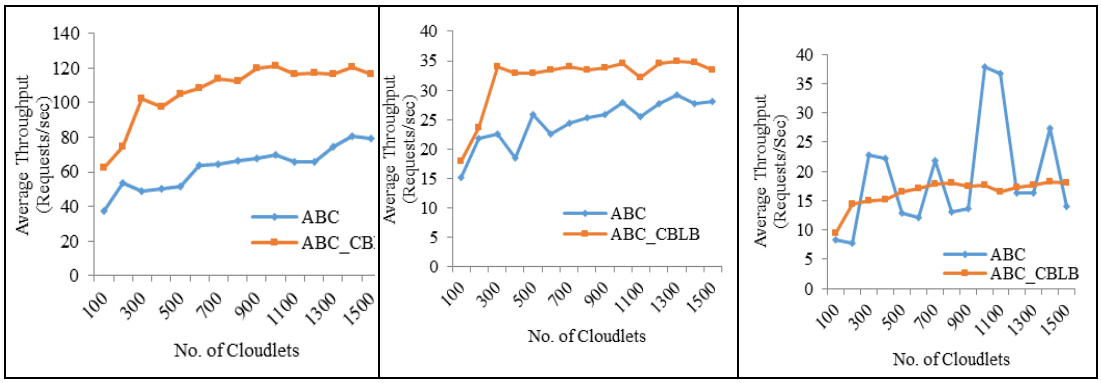

Figure 11. Average throughput for a fixed range of cloudlet lengths 500-1500, 500-5500 & 500-11000 for the heterogeneous environment

range of cloudlet lengths. Fig. 11 graphs plotted for average throughput against the number of cloudlets. The graphs clearly indicate the better performance of the ABC_CBLB algorithm than the ABC for light and average load requests. But there is a drastic variation in the performance of the ABC algorithm for the heavy load requests. For the heavy load requests, the ABC_CBLB algorithm has given a stable performance. In the heterogeneous environment also, the ABC_CBLB algorithm has given a better and stable throughput.

When the results of the algorithms are observed, for the parameters average make span and average throughput, there is a high variation in the performance of the ABC for heavy load requests in the heterogeneous environment. The algorithm is not able to maintain stability for the heavy load requests. But the ABC_CBLB algorithm has given a better and stable performance than the ABC in both heterogeneous and homogeneous cloud environments. This shows the suitability of the ABC_CBLB algorithm is for both the environments.

4.2. Statistical Analysis of the performance of the ABC_CBLB and ABC algorithm

In the previous section, the graphical representations of the results have shown better performance of ABC_CBLB compared to basic ABC. In this section, the significance of the obtained result is tested using statistical test. The results obtained

ISSN: 2233-7857 IJFGCN

under the same condition with different methods can be compared by the statistical tests like F-test, T-test, and ANOVA. Here, we have considered F-test for the analysis. The mean, standard deviation, variance and F-values are calculated for the results discussed in the previous section for the parameters make span, average response time and average waiting time for the cloudlet length range 500-1500 (Case 1), 500-5500 (Case 2) and 500-11000 (Case 3). As the value of throughput is dependent on the value of make span, the statistical values are not calculated for this parameter. The number of samples considered are 15. Hence, the degree of freedom is (14,14). The standard tabular values for the degree of freedom (14,14) is 1.94 for the significance level α=0.10. The calculated F-values are compared this value.

The calculated values for the results of the homogeneous environment are shown in Table III. In the table, we can observe that the mean values of the ABC_CBLB are lesser than the ABC for all three parameters and all cloudlet length ranges. When the standard deviation (SD) and variance are considered, the values of the ABC are greater than the ABC_CBLB for average make span. The F-values obtained for this parameter for cases 1, 2 and 3 are 2.276341, 1.462884, and 1.386884, respectively. The calculated value for case 1 is higher than this tabular value, which shows that both the algorithms have taken different time to process a set of jobs. The lesser deviation, lesser variance and lesser mean value of ABC_CBLB indicate that the algorithm has taken lesser processing time than the ABC algorithm. In cases 2 and 3, the calculated values are lesser than the tabulated value, i.e., the hypotheses are accepted. This indicates that both the algorithms have given a similar kind of results for these cases. Nonetheless, when the deviation and variance are considered, there is a less deviation and variance in the ABC_CBLB. When the parameters average response time and average waiting time are considered, in all three cases, the SD and the variance are almost similar for both the algorithms. In some places, the values of the ABC_CBLB are greater than the ABC which is imperceptible. For these parameters, the F-values are also lesser than the tabular value. This means that both the algorithms have given the same performance for the parameter's average response time and the average waiting time.

Like homogeneous environment, F-values are calculated for the heterogeneous environment and are shown in Table IV. For all three cases in all the parameters, the ABC_CBLB has given less mean value than the ABC. Except few values, all the SD and variance values of the ABC_CBLB are lesser than the ABC. Whereas, the F-value of make span in case 1 is greater than the tabular value and other F-values are lesser than that. This shows performance difference only in case of case 1 of make span. Even though similar performance is obtained for remaining cases, the SD and variance are lesser than the ABC. But the difference in values in case of average response time and average waiting time are very less. This shows the minor variation in the performance of the algorithms for these two parameters.

When the results of the homogeneous and heterogeneous environments are observed, the algorithms have given a similar kind of performance for the parameters average response time and average waiting time. This shows that the enhancement of the ABC algorithm has not affected the response time of the requests. In the algorithm, the movement of VMs among the groups has not made requests to wait for a longer time in the queue. But the algorithm has taken lesser processing time than the ABC. In case of 2 and 3, the algorithm has shown less deviation and variance than the basic algorithm. This shows that it is capable of handling dynamic requests more efficiently than the basic ABC algorithm.

ISSN: 2233-7857 IJFGCN

Table 3. F-Test Values for Homogeneous Environment (VM MIPS=5000)

Parameters	Cloudlet Length	Algorithm Name	Mean	Standard deviation	Variance	Calculated F value
Make span	500-1500	ABC	3.927505	1.514663	2.294203	2.276341
		ABC_CBLB	2.128156	1.003916	1.007847	
	500-5500	ABC	7.82692	3.416005	11.66909	1.462884
		ABC_CBLB	5.746862	2.824318	7.976772	
	500-10500	ABC	13.59826	5.865327	34.40206	1.386884
		ABC_CBLB	10.23895	4.980491	24.80529	
Average Response Time	500-1500	ABC	1.258929	0.473683	0.224376	1.005825
		ABC_CBLB	0.983706	0.472309	0.223076	
	500-5500	ABC	2.639035	1.196784	1.432291	1.005477
		ABC_CBLB	2.472648	1.200057	1.440137	
	500-10500	ABC	4.634702	2.139996	4.579581	1.024444
		ABC_CBLB	4.375323	2.114311	4.470311	
Average Waiting Time	500-1500	ABC	1.002686	0.473516	0.224217	1.005121
		ABC_CBLB	0.727179	0.472308	0.223075	
	500-5500	ABC	1.166616	1.198149	1.435562	1.006147
		ABC_CBLB	1.820762	1.201826	1.444386	
	500-10500	ABC	3.481655	2.149536	4.620504	1.023818
		ABC_CBLB	3.481655	2.124386	4.513014	

Table 4. F-Test Values for Heterogeneous Environment (VM MIPS=1000-5000)

Parameters	Cloudlet Length	Algorithm Name	Mean	Standard deviation	Variance	Calculated F value
Average Make span	500-1500	ABC	11.97946	5.309866	28.19468	2.294534
		ABC_CBLB	7.075167	3.50539	12.28776	
	500-5500	ABC	31.39651	15.01459	225.4379	1.433616
		ABC_CBLB	24.01594	12.53999	157.2512	
	500-10500	ABC	50.87821	25.49239	649.8617	1.13813
		ABC_CBLB	46.55542	23.89541	570.9907	
Average Response Time	500-1500	ABC	2.512255	1.075122	1.155888	1.290222
		ABC_CBLB	1.99297	0.946511	0.895883	
	500-5500	ABC	6.352804	2.883833	8.316493	1.050196
		ABC_CBLB	5.865782	2.955326	8.733949	
	500-10500	ABC	11.23527	5.316811	28.26848	1.088583
		ABC_CBLB	10.8958	5.547305	30.77259	
Average Waiting Time	500-1500	ABC	1.942774	1.061872	1.127573	1.248909
		ABC_CBLB	1.450029	0.950182	0.902846	
	500-5500	ABC	4.763419	2.897295	8.394319	1.020376
		ABC_CBLB	4.300205	2.926663	8.565359	
	500-10500	ABC	8.610822	5.322055	28.32427	1.063806
		ABC_CBLB	8.000174	5.489219	30.13152	

4.3. Implementation of the proposed algorithm in the AWS cloud platform

The proposed algorithm is implemented in the Eclipse Java Integrated development environment (IDE) with the AWS tool kit to test the performance of the proposed algorithm in a real cloud environment. Here, we have considered the web requests for the experiment. The web application is developed by using a spring

ISSN: 2233-7857 IJFGCN

boot framework. The Elastic Compute Cloud (EC2) instances are considered to run the web application to process web requests (REST queries). One more EC2 instance is considered to run a load balancing application. Initially, a request is sent to an instance which runs the load balancing application. That request contains the number of requests to be generated. Upon receiving the request, the application running in the load balancing server generates the specified number of requests. These request lengths are in the range 400-3650. The requests are distributed among all the available EC2 instances according to the load balancing algorithm. During the experiment, the readings are taken by running the load balancing application for the proposed ABC_CBLB and the ABC algorithms separately. The changes in the performance are observed by increasing the load of 100 requests every time to check the efficiency of the algorithms for heavy load. Initially, a load of 100 requests is considered, and the experiment is repeated until 3400 requests.

Figure 12-15 shows the graphs plotted w.r.t the number of requests for make span, average response time, average waiting time and throughput respectively. In these graphs, we can observe the better performance of ABC_CBLB algorithm for make span and throughput. The ABC_CBLB algorithm has given almost the same performance as ABC for average response time and average waiting time. When the simulation results and real environment results are compared, similar results are obtained from both the environments. The request length range in the AWS environment lies between the small and moderate cloudlet length ranges. In the simulator environment for that cloudlet length range, the proposed algorithm ABC_CBLB has given a better performance than the ABC algorithm (Figure 1 & 4).

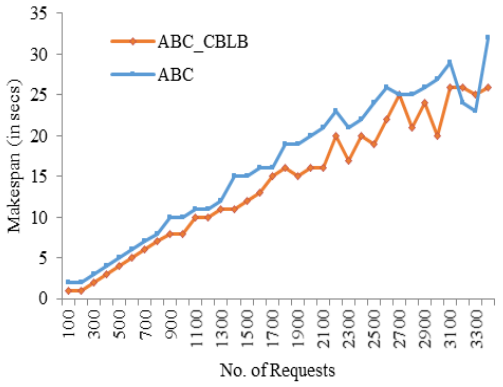

Figure 12. No of requests v/s Make span

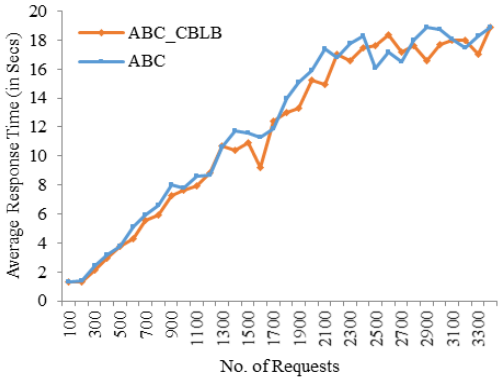

Figure 13. No of requests v/s Average Response Time

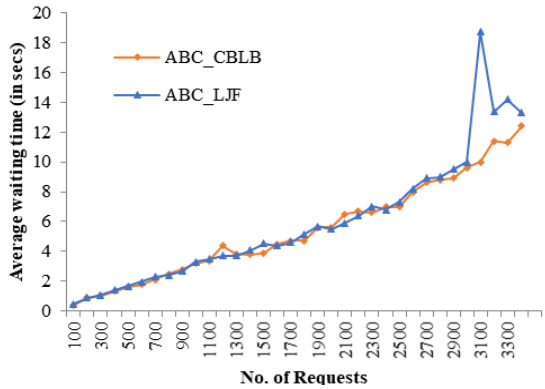

Figure 14. No of requests v/s Average Waiting Time

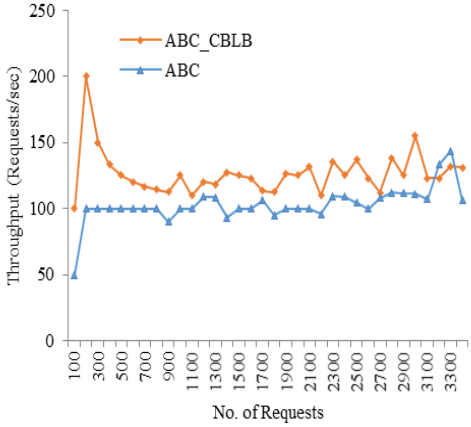

Figure 15. No of requests v/s Throughput

ISSN: 2233-7857 IJFGCN

In the AWS environment also the ABC_CBLB algorithm has given a better performance than the ABC (Figure 12& 15). In the simulation environment, there is a very slight improvement in the performance of the ABC_CBLB algorithm with respect to average response time and average waiting time. The same kind of behavior can be noticed in the AWS environment also. When the overall performance of the ABC_CBLB algorithm is considered, it has given better performance for smaller range dynamic requests and a slight improvement in the performance for the longer-range dynamic requests. However, there is no deterioration in the performance of the algorithm compared to ABC.

5. Conclusion

An enhancement of the Artificial Bee Colony algorithm is done in this work by merging the ABC algorithm with Capacity Based Load Balancing algorithm. Experiments were carried out to compare the performance of ABC_CBLB algorithm with the basic ABC algorithm in the simulator environment as well as the real cloud environment (AWS platform). The significance of these results is tested by the statistical F-test.

The performance of the algorithm is tested for both homogeneous and heterogeneous environments. The ABC_CBLB and ABC algorithms have shown a similar kind of performance in both environments. Initially, the total time taken by both the algorithms to process a set of submitted jobs is analyzed for dynamic number of cloudlets with small, moderate, and higher range of cloudlet length. ABC_CBLB algorithm has taken lesser time than the ABC algorithm to process small and moderate range cloudlet length requests, and both the algorithms have taken almost the same time to process longer-range cloudlet length requests. The variation in the performance of the algorithms is also tested by increasing the number of datacenters, the number of VMs and the capacity of VMs. In these cases, also the ABC_CBLB algorithm has given a better performance than the ABC. The performance of the proposed algorithm has been checked for the parameters make span, average waiting time, average response time and throughput. The graphs show improvement in the performance of the ABC_CBLB algorithm for these parameters. In the statistical test, ABC_CBLB shows improved performance for make span and same performance with respect to average response time and average waiting time but with improved stability. This indicates that the movement of VMs among the UC groups has no impact on the response and waiting time of the cloudlets. But in the case of throughput, the ABC_CBLB algorithm has shown increase in processing number requests per second compared to ABC algorithm. Hence, ABC_CBLB supports dynamic infrastructure, dynamic loads as well as homogenous and heterogeneous capabilities of VMs.

References

[1]. Alireza Sadeghi Milani, Nima Jafari Navimipour, “Load balancing mechanisms and techniques in the cloud environments: Systematic literature review and future trends”, Journal of Network and Computer Applications, vol. 71, pp. 86–98 (2016).

[2]. Arpan Kumar Kar, “Bio-Inspired Computing – A Review of Algorithms and Scope of Applications”, Expert Systems with Application, DOI: 10.1016/j.eswa.2016.04.018 (2016).

[3]. Bahman Keshanchi, Alireza Souri, Nima Jafari Navimipour, “An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: formal verification, simulation, and statistical testing”, The Journal of Systems & Software, (2016).

ISSN: 2233-7857 IJFGCN

[4]. Basturk, B. and Karaboga, D. "An artificial bee colony (ABC) algorithm for numeric function optimization", Proceeding of the IEEE Swarm Intelligence Symposium, Indianapolis, USA, pp.12–14, IEEE, (2016)

[5]. Basu S., Kannayaram G., Ramasubbareddy S., Venkatasubbaiah C., "Improved Genetic Algorithm for Monitoring of Virtual Machines in Cloud Environment", Smart Intelligent Computing and Application, Smart Innovation, Systems and Technologies, vol 105. pp.319-326, Springer, Singapore (2019).

[6]. Dervis Karaboga, BahriyeAkay, "A comparative study of Artificial Bee Colony algorithm", Journal Applied Mathematics and Computation Volume 214 Issue 1, August, pp. 108-132, (2009).

[7]. Dhinesh Babu, P. Venkata Krishna, 'Honey bee behavior inspired load balancing of tasks in cloud computing environments', Applied Soft Computing, Vol. 13, Issue 5, pp. 2292-2303, (2013).

[8]. Einollah Jafarnejad Ghomi, Amir Masoud Rahmani and Nooruldeen Nasih Qader, "Load-balancing Algorithms in Cloud Computing: A Survey", Journal of Network and Computer Applications, vol. 88 Issue C, pp. 50-71, (2017).

[9]. Fei Luo, Ye Yuan, Weichao Ding, Haifeng Lu,'An Improved Particle Swarm Optimization Algorithm Based on Adaptive Weight for Task Scheduling in Cloud Computing', CSAE '18, October 22–24, 2018, Hohhot, China, (2018)

[10]. Gagandeep Kaur, Manoj Agnihotri, 'Artificial Bee Colony Based Live Migration Technique for Cloud Data Centers', Proceedings of the International Conference on Intelligent Sustainable Systems (ICISS 2017).

[11]. Gamal M., Rizk R., Mahdi H., Elhady B., 'Bio-inspired Load Balancing Algorithm in Cloud Computing', Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2017. Advances in Intelligent Systems and Computing, vol 639. pp. 579-589, Springer, Cham, (2018).

[12]. Goyal, T., Singh, A., Agrawal, A., 'Cloudsim: simulator for cloud computing infrastructure and modeling', Procedia Eng. 38, 3566–3572, (2012).

[13]. Jaspinder Kaur, Brahmaleen Kaur Sidhu, 'New Flower Pollination based Task Scheduling Algorithm in Cloud Environment', 4th IEEE International Conference on Signal Processing, Computing and Control, Sep 21-23, 2017, Solan, India, (2017).

[14]. Jeng-Shyang Pan, Haibin Wang, Hongnan Zhao, and Linlin Tang, "Interaction Artificial Bee Colony Based Load Balance Method in Cloud Computing", Genetic and Evolutionary Computing, Advances in Intelligent Systems and Computing Vol. 329, Springer, (2015).

[15]. Jing Yao, Ju-hou He (2012), 'Load balancing strategy of cloud computing based on artificial bee algorithm', 8th International Conference on Computing Technology and Information Management (NCM and ICNIT), pp. 185-189, (2012).

[16]. Karaboga, Dervis (2005), 'An Idea Based on Honey Bee Swarm for Numerical Optimization', Technical Report-TR06, October 2005, Erciyes University, (2005).

[17]. Karaboga, D. and Basturk, B. (2007) 'A powerful and efficient algorithm for numeric function optimization: artificial bee colony (ABC) algorithm', Journal of Global Optimization, Vol. 39, pp.459–471, (2007).

[18]. Karaboga, D. and Basturk, B. (2007) 'Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems', in lecture notes in ArtificialIntelligence, Vol. 4529, pp.789–798, Springer-Verlag, Berlin, (2007).

[19]. Karaboga, D. and Basturk, B. (2008) 'On the performance of artificial bee colony (ABC) algorithm', Applied SoftComputing, Vol. 8, pp.687–697, (2008).

[20]. Karaboga, D. and Akay, B. (2011) 'A modified artificial bee colony (ABC) algorithm for constrained optimization problems', Applied Soft Computing, Vol. 11, pp.3021–3031, (2011).

[21]. Karaboga, D., Gorkemli, B., Ozturk, C. and Karaboga, N. (2014) 'A comprehensive survey: artificial bee colony (ABC)algorithm and applications', Artificial Intelligence Review, Vol. 42, No. 1, pp.21–57, (2014).

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

ISSN: 2233-7857 IJFGCN

[23]. Kshama S.B., Shobha K.R. (2018) '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).

[24]. M. Roshni Thanka, P. Uma Maheswari, E. Bijolin Edwin (2017), "An improved efficient: Artificial Bee Colony algorithm for security and QoS aware scheduling in cloud computing environment", Cluster Computing, Springer, (2017).

[25]. Madni, S.H.H., Latiff, M.S.A., Ali, J. et al. (2018), 'Multi-Objective-Oriented Cuckoo Search Optimization-Based Resource Scheduling Algorithm for Clouds', Arab J Sci Eng, pp 1-18, (2018).

[26]. Mishra, S. K., et al., (2018) 'Load balancing in cloud computing: A big picture', Journal of King Saud University - Computer and Information Sciences, Vol. X, No. Y4, pp.000–000, (2018).

[27]. Narendrababu Reddy G., Phani Kumar S., 'Modified Ant Colony Optimization Algorithm for Task Scheduling in Cloud Computing Systems', Smart Intelligent Computing and Applications, Smart Innovation, Systems and Technologies, vol 104. pp 357-365 Springer, Singapore, (2019).

[28]. Nayak J., Naik B., Jena A.K., Barik R.K., Das H., "Nature Inspired Optimizations in Cloud Computing: Applications and Challenges", Cloud Computing for Optimization: Foundations, Applications, and Challenges, Studies in Big Data, vol 39. Springer, Cham, (2018)

[29]. Ramesh Babu K.R., Samuel P, "Enhanced Bee Colony Algorithm for Efficient Load Balancing and Scheduling in Cloud", Innovations in Bio-Inspired Computing and Applications, Advances in Intelligent Systems and Computing, vol 424. Springer, Cham, (2016).

[30]. Richa Jain, Neelam Sharma, Pankaj Jain, 'A Systematic Analysis of Nature Inspired Workflow Scheduling Algorithm in Heterogeneous Cloud Environment', 2017 International Conference on Intelligent Communication and Computational Techniques (ICCT) Manipal University Jaipur, (2017).

[31]. Sagnika S., Bilgaiyan S., Mishra B.S.P., 'Workflow Scheduling in Cloud Computing Environment Using Bat Algorithm', Proceedings of First International Conference on Smart System, Innovations and Computing. Smart Innovation, Systems and Technologies, Vol.79. pp 149-163 Springer, Singapore, (2018).

[32]. Shridhar Domanal, Ram Mohana Reddy Guddeti, and Rajkumar Buyya (2017), 'A Hybrid Bio-Inspired Algorithm for Scheduling and Resource Management in Cloud Environment', IEEE Transactions on Services Computing, pp. 1-14, (2017).

[33]. Tripathi A., Shukla S., Arora D, 'A Hybrid Optimization Approach for Load Balancing in Cloud Computing', Advances in Computer and Computational Sciences, Advances in Intelligent Systems and Computing, vol 554, pp 197-206, Springer, Singapore, (2018).

[34]. Xu, Y., Li K., Hu J, Li K., 'A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues', Information Sciences, 270(0): pp. 255- 287, (2014).

[35]. Neha Thakkar, Rajender Nath, "Discrete Artificial Bee Colony Algorithm for Load Balancing in Cloud Computing Environment", International Journal of P2P Network Trends and Technology (IJPTT), vol. 8 Issue-6, (2018)

[36]. Research Report (14 August 2018). https://www.idg.com/tools-for-marketers/2018-cloud-computing-survey/

ISSN: 2233-7857 IJFGCN

Authors

Mrs. Kshama S B, completed her M Tech in Computer Science and Engineering in the year 2013 from Bangalore Institute of Technology (BIT), Bangalore, under Visvesvaraya Technological University (VTU). She is currently pursuing her Ph.D. under VTU, in the department of TCE, RIT, Bengaluru. Her research interest is cloud computing, software engineering. She has presented her research papers in several international conferences and Journals.

Dr. K. R. Shobha, received her M.E. degree in Digital Communication Engg from Bengaluru University, Karnataka, India, and Ph.D. from Visvesvaraya Technological University. She is currently working as an Associate Professor in the Department of Telecommunication Engineering, Ramaiah Institute of Technology, Bengaluru. Her research areas include Mobile Adhoc Networks, IoT and Cloud Computing. She has more than 25 Papers publications to her credit. She is a Senior IEEE Member serving as execom member of WIE and IEEE Communication Society, Bengaluru Section. She is also an active member IETE, ISoc and IAENG

ISSN: 2233-7857 IJFGCN