Received: February 20, 2020. Revised: April 7, 2020.

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

An Efficient Load Balancing Technique based on Cuckoo Search and Firefly Algorithm in Cloud

Kethavath Prem Kumar[1]* Thirumalaisamy Ragunathan[2] Devara Vasumathi[3] Pamulapati Krishna Prasad[2]

[1]*Department of Computer Science & Engineering, ACE Engineering College, Hyderabad, India*
[2]*Department of Computer Science & Engineering, SRM University, Amaravathi, India*
[3] *Department of Computer Science & Engineering, Jawaharlal Nehru Technological University, Hyderabad, India*
* Corresponding author's Email: kpremkumarcs@gmail.com

Abstract: In recent years, cloud Load balancing (CLB) is the significant research area because vast data are stored on the servers leads to increase the loads on cloud servers. A trade-off on servers are maintained by LB techniques by distributing less power with equal load. In this paper, Cuckoo Search with Firefly Algorithm (CS-FA) is proposed for LB in a cloud environment. Initially, the capacity and load of each virtual machine are calculated. If the load of the virtual machine is greater than the balanced threshold value then, the LB algorithm is used for allocating the tasks. The CS-FA algorithm selects the best Virtual Machines (VMs) for assigning the tasks and migrate the overloaded VMs tasks to under-loaded VMs task. This algorithm majorly avoids the imbalanced workload performances in a cloud environment. The performance of proposed CS-FA method is compared with existing LB techniques such as Dynamic LB (DLB), Hybrid Dynamic LB (HDLB) and Honey Bee Behavior LB (HBB-LB) to evaluate the capacity and load of methods. The results showed that the existing HDLB method migrate seven tasks whereas the CS-FA method migrates only two tasks. The experimental result shows that proposed CS-FA method migrate only two tasks when number of loads is 40 and existing method migrate 6 tasks.

Keywords: Cloud computing, Cuckoo search, Firefly algorithm, Honey bee behavior, Load balancing, Virtual machine.

1. Introduction

Cloud Computing (CC) is a well-developed business model for distributed computing environment because of its services to users. According to individual demand, the IT resources are shared, allocated and accessed by users, which is provided by CC model [1, 2]. Moreover, CC provides various kinds of services such as Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS). These services are useful in different domain applications such as scientific, business, industrial, etc. [3]. In general, CC platform includes three significant problems such as Virtualization problem, Distributed framework and Load balance problem [4]. The LB is the heart of cloud system because it sustains the entire system workloads [5]. The major benefit of LB aspect is to improve the performance of large scale computing systems and applications. Subsequently, the tasks are designed to redistribute over the components of the computing system and reduce the response time, increase the resource utilization, throughput and avoid the overload possibilities [6, 7]. The LB methods are significantly classified into two groups such as static and dynamic LB [8]. An existing CLB research work employs several algorithms such as Particle Swarm Optimization (PSO) [9], Genetic Algorithm (GA) [10], Artificial Bee Colony (ABC), Ant Colony Optimization (ACO) [11], etc.

In traditional research work many optimization methods are used for CLB to minimize the task overloading and under loading problems. The GA

algorithms are used to generate the better solutions to the optimization problems but genetic operations are more complicated as well as time consuming. The major problems faced by traditional PSO and ABC algorithms are more number of iteration time and they fall into local extremes [12]. In this research work, an efficient CLB hybrid cuckoo search-firefly optimization algorithm is proposed. The proposed model includes the several major steps such as (i) At first, scheduling the task with the help of Round Robin (RR) method (ii) Calculate the capacity of each VMs. (iii) Based on the capacity calculate the load of the each VM (iv) the load values are passed to the proposed hybrid CS-FA, as a result the under-loaded and overloaded VMs are grouped. Also, if VMs are overloaded then migrate the task to the under-loaded VMs. The proposed algorithm finds the best VM in less time and improves the LB efficiency. The major contribution of the current research work is addressed below.

- The VM loads are randomly selected by CS algorithm, which causes the imbalance in cloud system that is overcome by implementing the firefly approach.
- Design the proposed hybrid cuckoo search-firefly algorithm to decrease the time complexity of Task Scheduling (TS) and improves the performance of LB in a cloud environment.
- Develops the proposed model to handle the multiple objectives in cloud platform such as the capacity of VMs, load of VMs, number of processors, bandwidth and Million Instruction Per Second (MIPS).

This paper is composed as follows. Section 2 presents the survey of several recent research work on CLB techniques. Section 3 represents the system model of the proposed CLB method. Section 4 shows comparative experimental result for proposed and existing segmentation strategies in a cloud environment. The conclusion is made in section 5.

2. Literature review

Numerous methods have been proposed by researchers in LB in cloud environment using optimization technique. In this section, a brief review of some important contributions to the existing literature is presented.

V. Jeyakrishnan, and P. Sengottuvelan, [13] proposed Bacterial Swarm Optimization (BSO) Algorithm for resource allocation and LB in data centers. The proposed BSO algorithm used different set of jobs and calculated the list of resources for each job. The capabilities of local and global search and fast merging optimal points were identified by this study. This algorithm reduces the operational cost, make span and improves the resource utilization. But, BSO algorithm employed number of live migration and cloud data which were not secured.

V. A. Xavier, and S. Annadurai, [14] presented Chaotic Social Spider (CSS) Algorithm for TS in different heterogeneous VMs. This proposed algorithm avoids the local convergence and search the best optimized VM for user tasks. The throughput of the cloud system was increased by CSS method which decreased the overall computational cost and balanced resource utilization. But, CSS algorithm is only flexible to work for assigned tasks not for independent task.

V. Polepally, K. S. Chatrapati, [15] proposed Dragonfly Optimization (DO) and Constraint Measure LB (CMLB) algorithm for cloud. When VMs were imbalanced, then the tasks were reallocated to the corresponding VMs by using CMLB algorithm. The DO algorithm was employed to generate the optimal threshold values. If the load of the PM was greater than the balanced threshold value, then CMLB algorithm reallocates the tasks to VM. In this research work, if the number of tasks were increased, then variation in system work balance.

G. Reddy, N. Reddy, and S. Phanikumar, [16] presented Modified ACO algorithm for TS in cloud. The major objective of MACO was to reduce the make span and the corresponding relevant VMs performed the Multi-Objective (MO) TS process. This algorithm enhances the performance of TS by decreasing make span and degree of imbalance with the help of MO nature. In this literature, if VM workloads were varied then difficult to handle the entire system so, improvements were required in LB.

K. A. Sultanpure, and L. S. S. Reddy, [17] proposed ABC algorithm for energy efficient job scheduling in VM migration. The ABC algorithm significantly balances the load and to sustain the SLA violation with less execution time by optimal objective function. An energy for the conception of VM must be lower than the energy required for the allocations of jobs. The major drawback of this algorithm was if the number of workloads were increased then gradually increase the SLA violation.

L. D. Dhinesh Babu, and P. Venkata Krishna, [18] proposed Honey Bee behavior inspired Load Balancing (HBB-LB) provides load balancing in the VM for maximizing the throughput. The proposed method balances the priority of task and minimize the waiting time of the method. The developed method is compared with other existing methods

such as Dynamic Load Balancing (DLB), and Hybrid Dynamic Load Balancing (HDLB). The experimental result shows that the developed method has the significant performance than existing method. The developed method convergence rate is low and this affects the performance of the method.

To rectify these issues, hybrid CS-FA is proposed and it's reduced the make span and improves the TS efficiency.

3. Proposed load balancing in cloud

A vast amount of information can be stored in cloud servers which increases the load on servers. The goal of LB is to maintain the trade-off on servers by distributing less power with equal load. In this research paper, hybrid CS-FA is proposed for LB in cloud environment. The proposed CS-FA algorithm decreases the make span and improves the TS policy. Moreover, proposed algorithm considers the multiple parameters such as processing speed, bandwidth, and memory and so on while estimating the VM capacity. The general system model of proposed LB mechanism is shown in Fig. 1.

3.1 System model

To balance the entire system workloads, the task allocation to the VM is a vital process in cloud. In some situations, the VM is overloaded that is Number of Tasks (No. Ts) is allocated to it which increases the response time. Thus the LB algorithm is introduced to allocate the tasks to the VMs. Fig. 1 represents the proposed LB system model which includes various components such as number of data centers, physical machines, VMs and etc.

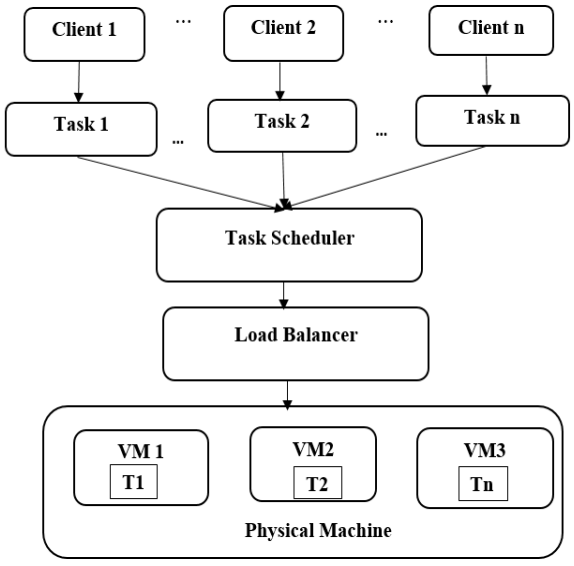


Figure. 1 General structure of LB in cloud

Each data center has some computing resources to perform the user's tasks. The multiple cloud users include numerous tasks and each task is assigned to the different VMs. According to the processing time of each task, the load of VM can be calculated. Hence the processing time of each task differs which leads to variations in load of VM. Suppose, the VM are overloaded, the loads are shared to VM which is under loaded for achieving optimal resource utilization.

Let us consider that C is the cloud system, number of PM is represented as P which includes multiple VMs and it's indicated as V. The n number of PM is represented in Eq. (1).

$$C = \{P_1, P_2, \ldots\ldots P_k \ldots\ldots P_n\} \quad 1 < k \leq n \tag{1}$$

Whereas, P_k is indicated as k^{th} number of PMs and P_n indicated as n^{th} number of PMs. Each PM is consisting of multiple VMs and it's mathematically shown in the Eq. (2).

$$V = \{V_1, V_2, \ldots\ldots V_i \ldots\ldots V_m\} \quad 1 < i \leq m \tag{2}$$

Whereas, m is the total number of VMs in the k^{th} PM. Moreover, cloud system includes multiple users so, l No. Ts are mathematically shown in the Eq. (3),

$$T = T_1, T_2, \ldots\ldots T_j \ldots\ldots T_l \tag{3}$$

Whereas, T is represented as set of tasks, l is indicated as total No.Ts. Each task is assigned to the VM. When the workload status of VM is normal, then tasks are processed without any problem in the cloud system. If VM workload status is overloaded, then LB strategy is required for migrate the task form overloaded VMs to under-loaded VMs. The VMs are depending on the few parameters for LB, which are represented in Eq. (4),

$$V_i = \{r_i, s_i, b_i, g_i, m_i\} \tag{4}$$

Whereas, number of processors are represented as r_i, number of MIPS is indicated as s_i, the variable b_i is indicated as bandwidth, task migration cost is signified as g_i, memory usage is represented as m_i. Every task has various execution time as well as priority value. Additionally, the tasks are assigned to the VMs based on two major key points such as (i) Higher priory task (ii) Minimum execution time related tasks are first assigned to the VM. These two points significantly reduce the task communication cost.

3.2 Proposed cuckoo search-firefly algorithm

In this section, the proposed hybrid CS-FA is used for LB in cloud system. The major objective of this algorithm is to schedule the tasks and allocate the overloaded VMs tasks to the under-loaded VMs. Hence, it avoids the task imbalanced situations in the entire system. Generally, the cloud system consists of numerous data centers and each data center includes number of PMs also, each PMs includes several VMs. The multiple cloud users have sent the requests (task) which are submitted to the data centers. After that, those tasks are executed by the VMs. Here, the different requests assigned to the VMs by using proposed hybrid CS-FA. The proposed LB system includes several processes such as (i) scheduling the task by RR method (ii) calculate the capacity of VM (iii) calculate load of each VM (iv) Hybrid CS-FA used for identifying the overloaded and under-loaded task and, (v) task migration from overloaded VMs to under-loaded VMs. The block diagram of proposed CS-FA based LB on cloud is described in Fig.2.

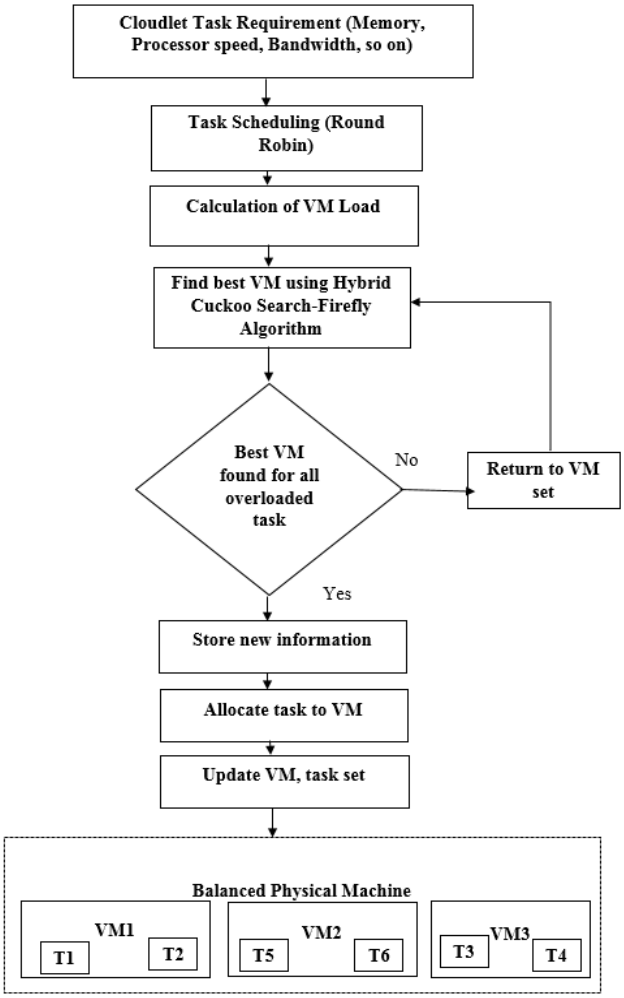


Figure. 2 Block diagram of proposed LB model

3.2.1. Task scheduling

In this proposed CS-FA model, TS is the first step which reads the tasks and assigned to VM by using the RR algorithm. The time quantum is a short time period for planning and scheduling the tasks for execution in each round which plays most significant role in RR algorithm. The RR algorithm allocates the tasks to the processor and providing time quantum to every task. If execution of the task is not completed within the time quantum, then the task will be stopped and tasks stored back to the queue waiting for next turn. Similarly, other tasks are used their time quantum and executes the tasks. The benefit of RR algorithm is that each task will be executed in turn and they don't have to be waited for the previous one to get completed. But if the load is too high, then RR will take a long time to complete all the jobs. The chains RR scheduling strategy is used for internal scheduling of jobs. Here, in order to balance the loads LB algorithm is used namely Cuckoo Search-Firefly Algorithm.

3.2.2. Capacity of VMs

The estimation of VM capacity depends on the number of processors, MIPS, memory and bandwidth. It represented as CV_i which is mathematically shown in the Eq. (5),

$$CV_i = \left[\frac{r_i \times s_i \times m_i}{1000} + b_i\right] \times \frac{1}{2} \tag{5}$$

Whereas, r_i is the number of processors in i^{th} VMs, s_i is represented as MIPS, bandwidth is represented as b_i. The capacity of the CV_i takes a value in the range $[0, 1]$. The highest capacity is required for VM task allocation.

3.2.3. Calculation of VM load

The load of VMs is computed based on two approximates, called maximum bound and minimum bound approximates, in addition to the execution time of the task and the capacity of VMs. The VM load calculation is mathematically shown in the Eq. (6).

$$L(VM_i) = \frac{\frac{1}{T}\sum_{m=1}^{T}(E_m \times D_m^i)}{b_{min} + (b_{max} - b_{min}) \times CV_i} \tag{6}$$

Whereas, E_m is the execution time of m^{th} task, CV_i is the capacity of VMs. b_{max} is indicated as the maximum bound approximate, b_{min} is indicated as minimum bound approximate. T is indicated as total No.Ts, D_m^i is a function that indicates all the tasks in

VMs as given in the following condition shown in Eq. (7),

$$D_m^i = \begin{cases} 1 \;\; ; \;\; if\ m^{th}\ task\ in\ i^{th}\ VM \\ 0 \;\; ; \;\;\;\;\; Otherwise \end{cases} \quad (7)$$

Whereas, $L(VM_i)$ is indicated as VM loads and $L(VM_i)$ value is between 0 to 1. Moreover, $L(VM_i)$ preferred minimum value to handle the LB. After the calculation of VM's capacity and load, hybrid CS-FA is applied to select the best VM for task allocation. In next section, the detail description of CS-FA algorithm is presented.

3.2.4. Proposed cuckoo search-firefly algorithm

The proposed technique has to find appropriate under loaded VMs to reallocate the task that is taken out from the overloaded VM. The PMs may have more than one VM that can perform the task. Therefore, it is essential to consider the suitable VM that has the capacity to accept the task. The proposed technique not only performs LB but also considers the communication cost, priority level and execution time of the task. Let considers the VM's load and its capacity, which are the important parameters for determining the under loaded VMs.

The special lifestyle and reproduction strategy are the inspiration for constructing the CS algorithm which contains a population of eggs or nests. The solutions represent each egg in nest, whereas new solution describes the cuckoo eggs. If the host's egg is very similar to cuckoo egg, then host's egg is high which needs to calculate the fitness function in solutions. The solution which is not good in nest is replaced with better solution (cuckoo). A Lévy flight is performed in Eq. (8) to generate new solutions $x^{(t+1)}$ for cuckoo i.

$$x^{(t+1)} = x_i^{(t)} + a \oplus Levy(\lambda) \quad (8)$$

Where $a > 0$ is the step size which should be related to the scales of the problem of interest. The product $\oplus$ is entry wise multiplications. While the random steps of Levy distribution for large steps are drawn, the Levy flights provides random walk and the CS algorithm contains the following rules as below,

- At a time, only one egg can be laid by cuckoo which can dump into a randomly chosen nest.
- The next generations will be carried out by high-quality eggs from the best nests.
- The host species with probability is used to find out the laid egg by a cuckoo and the number of host nests are fixed. In this situation, the host species can either abandon the nest or take the egg and create the new nest.
- Consider $p_a \in [0, 1]$ as a probability of discovery of an alien egg in its nest of a host bird.

The new nests with new random solutions replaced the last assumption can be approximated by a fraction p_a of the m host nests. The major issue in CA algorithm is selected the nests (VMs) in random process to rectify this problem firefly algorithm is used.

The local search is provided by the FA algorithm which is faster than the CA which has only a single parameter along with population size. So a hybrid algorithm is adopted to find better results. The FA algorithm majorly concentrates on its position and fitness value of the objective function. The optimistic position shows the firefly has better objective value and it can attract more fireflies to move towards their direction, as each firefly has its own range of view.

Let's Consider, firefly swarm id N, the i firefly's position x_i, y_i matches the objective function $f(x_i, y_i)$ and the firefly's fluoresce value is T_i, the updating formula of each firefly's range of view in Eq. (9).

$$f_k^i(u+1) = min\{f_t, max\{0, f_k^i(u) + \beta(t_u - |t_u(u)|)\}\} \quad (9)$$

Whereas, $f_k^i(u+1)$ is the i^{th} firefly in $u+1$ range, t_u is the threshold value of the neighbor firefly's number. β is the control constant, $t_u(u)$ is the range in number of fireflies with high fluoresce. Therefore, the formula of $t_u(u)$ is mathematically represented in Eq. (10),

$$t_u(u) = \{j: \|y_j(u) - y_i(u)\| \prec f_k^i l_i(t) \prec l_j(u)\} \quad (10)$$

Whereas, $y_i(u)$ is the position of j firefly in t generation, $l_j(u)$ is the j firefly's fluoresce value in t generation, the neighbor firefly range is in f_k^i. The firefly neighbors selecting the probability are shown in Eq. (11).

$$f_{ij}(u) = \frac{l_i(u) - l_j(u)}{\sum_{k \in n_i(u)} l_k(u) - l_i(u)} \quad (11)$$

The position updating formula of firefly is shown in Eq. (12).

$$f_i(u) = f_i(u-1) + s\frac{f_j(u-1) - f_i(u-1)}{\|f_j(u-1) - f_i(u-1)\|} \quad (12)$$

The value of Fluoresce is shown in the Eq. (13).

$$f_i(u+1) = (1-t)l_i(u) + \gamma k(c_i(u+1)) \quad (13)$$

In above equation, γ is the parameter to measure the function value, $k(c_i(u+1))$ is the fitness value of the function. The Hybrid CS-FA is selecting the best VM for migrate the over task from overloaded VMs to under-loaded VMs.

4. Result and discussion

In this section, the experimental results of the proposed CS-FA and the comparative discussion of the CS-FA method with the existing LB methods, such as DLB, HBB-LB, HDLB and CMLB are described.

4.1 Experimental setup

The proposed LB method is experimented in a personal computer with Intel Core i3 processor and 2GB memory using Windows 8 operating system. The CS-FA method is implemented using Java with cloudsim and the performance is evaluated with various cloud set up for load and capacity.

4.2 Evaluation metrics

The proposed LB method is evaluated for the evaluation metrics load and capacity.

Load: The load can be defined as the total No.Ts assigned to the virtual machines.

Capacity: The capacity of the virtual machine is calculated by the following Eq. (14),

$$Capacity\ (VM) = Y \times I + B \quad (14)$$

Where, Y represents the number of processors, I represents the number of instructions, and B represents the bandwidth.

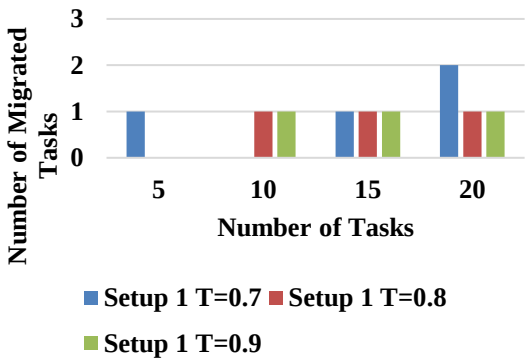
4.3 Analysis for the maximum task size of 20

In this section, the experimental result of the proposed CS-FA LB method is described for setup one and setup two. The first set up contains five physical machines and the second set up contains 10 physical machines which are running with seven virtual machines. Here, maximum number of the task size is the 20 and task migration performance is shown in the Table 1.

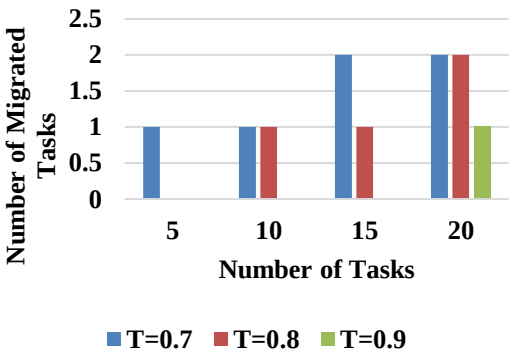
Fig. 3 describes the number of migrated tasks and the load value of VM, when the No.Ts are 5, 10, 15 and 20 for three threshold values such as 0.7, 0.8 and 0.9. Fig. 3 (a) represents the number of migrated tasks when the No.Ts are 5, 10, 15, and 20 for the threshold value of 0.7, 0.8, and 0.9. When the numbers of tasks are five, one task is migrated for the threshold value 0.7. When the No.Ts are ten, one task is migrated while applying the threshold value 0.8 and 0.9. For the all the three threshold values, one task is migrated when the No.Ts are 15, and two tasks are migrated when the No.Ts are 20. The proposed method has the higher efficiency due to convergence rate is high and has simple parameter tuning.

Table 1. Performance of number of task migration

Number of Tasks	Setup 1			Setup 2		
	T=0.7	T=0.8	T=0.9	T=0.7	T=0.8	T=0.9
5	1	0	0	1	0	0
10	0	1	1	1	1	0
15	1	1	1	2	1	0
20	2	1	1	2	2	1



(a)



(b)

Figure. 3 Illustration of number of migrated tasks of: (a) setup 1 and (b) setup 2

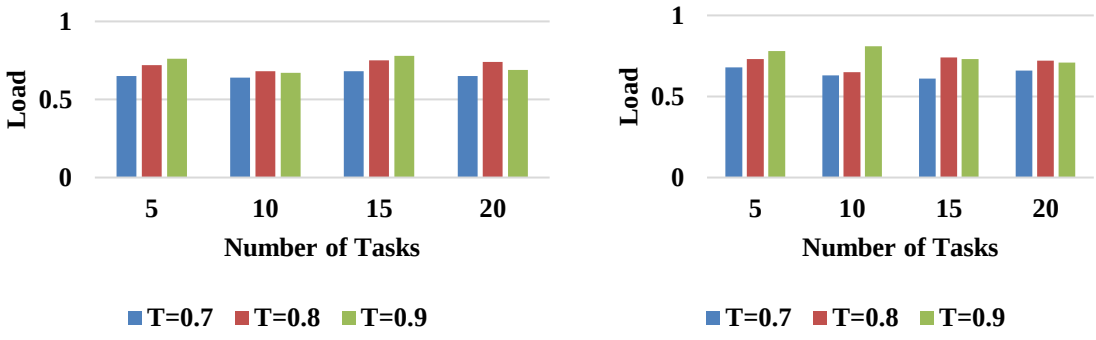


Figure. 4 Illustration of number of migrated loads of: (a) setup 1 and (b) setup 2

Fig. 3 (b) represents the performance of the different No.Ts migration of setup 2. When the No.Ts are five, one task is migrated for the threshold value 0.7. When the No.Ts are 10, one task is migrated while applying the threshold value 0.7 and 0.8. When the number of tasks is 15, two and one task is migrated for the threshold value 0.7 and 0.8 respectively. In two setups, the proposed method has the capacity to migrate task in three different threshold value. When the No.Ts are 20, two tasks are migrated for the threshold value 0.7 and 0.8 respectively also, one task migrate for the threshold value 0.9.the number of different load performance is shown in the Table 2.

Fig. 4 (a) shows the load of the VM when the No.Ts. are 5, 10, 15, and 20. When the No.Ts are 5, 10, 15, and 20, loads of the VM are 0.65, 0.64, 0.68, and 0.65 for the threshold value of 0.7. For the threshold value of 0.8, loads of the VM are 0.72, 0.68, 0.75, and 0.74 when the numbers of tasks are 5, 10, 15, and 20. For the threshold value of 0.9, loads of the VM are 0.76, 0.67, 0.78, and 0.69 when the numbers of tasks are 5, 10, 15, and 20.

Fig. 4 (b) represents the performance of the workload with respect to the different No.Ts. Fig. 4 (b) shows the load of the VM when the No.Ts are 5, 10, 15, and 20 while applying the threshold values 0.7, 0.8, and 0.9. For the threshold value of 0.7, loads of the VM are 0.68, 0.63, 0.61 and 0.66 when the No.Ts are 5, 10, 15, and 20. For the threshold value of 0.8, loads of the VM are 0.73, 0.65, 0.74, and 0.72 when the No.Ts are 5, 10, 15, and 20. For the threshold value of 0.9, loads of the VM are 0.78, 0.81, 0.73 and 0.71 when the No.Ts are 5, 10, 15, and 20.

4.4 Analysis for the maximum task size of 40

This section describes the performance of maximum task size is 40 with respect to the various threshold value. The two different setup values are tabulated in the Table 3. Fig. 5 depicts the number of migrated tasks and the load of the virtual machine when the No.Ts are 25, 30, 35, and 40 while providing the threshold value of 0.7, 0.8, and 0.9 for setup 1 and setup 2 respectively.

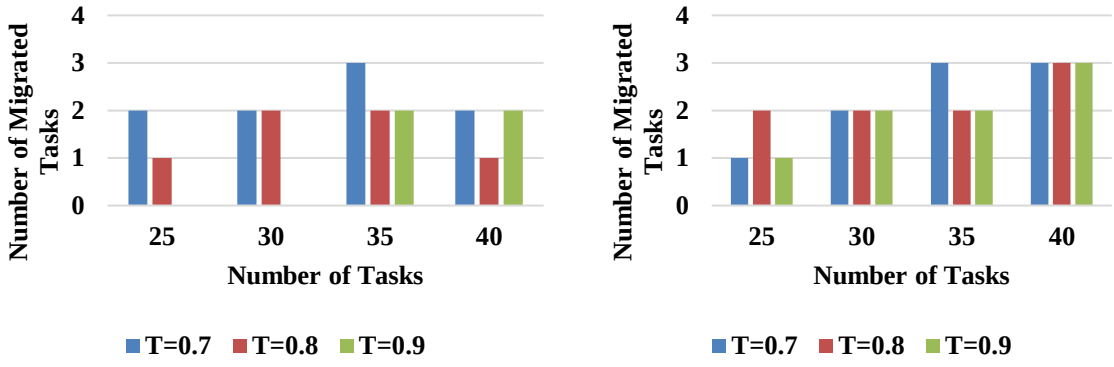
Fig. 5 (a) illustrates the number of migrated tasks when the No.Ts are 25, 30, 35, and 40 for the threshold values of 0.7, 0.8, and 0.9. When the No.Ts are 25, two tasks are migrated for the threshold values of 0.7. For the threshold value of 0.7, three tasks are migrated when the No.Ts is 35. For the threshold value of 0.8 and 0.9, two tasks are migrated when the No.Ts is 35. When the No.Ts is 30, two tasks are migrated for the threshold value of 0.7 and 0.8, zero tasks are migrated for the threshold value of 0.9. When the No.Ts is 40, two tasks are migrated for the threshold value of 0.7,and 0.9.The convergence rate of the proposed method is high and has less number of tuning in parameter. This helps in increases the efficiency of the developed method.

Table 2. Performance of different Load

Number of Tasks	Setup 1			Setup 2		
	T=0.7	T=0.8	T=0.9	T=0.7	T=0.8	T=0.9
5	0.65	0.72	0.76	0.68	0.73	0.78
10	0.64	0.68	0.67	0.63	0.65	0.81
15	0.68	0.75	0.78	0.61	0.74	0.73
20	0.65	0.74	0.69	0.66	0.72	0.71

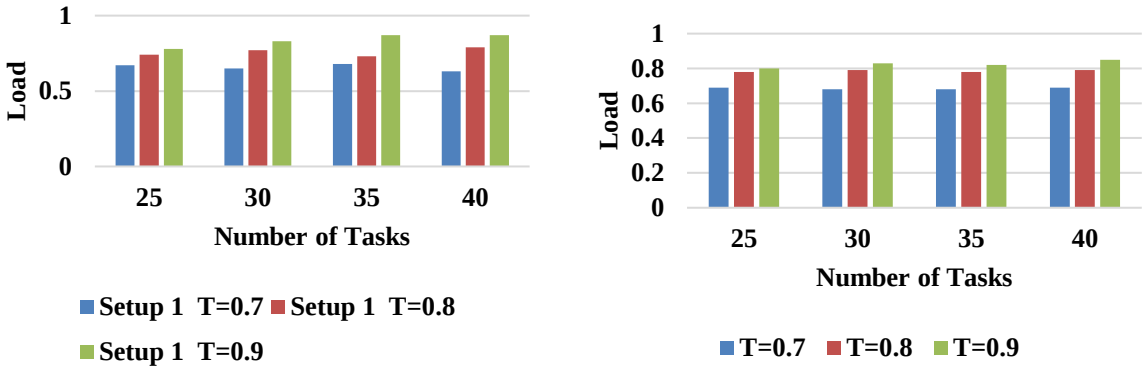
Table 3. Performance of different No.Ts

Number of Tasks	Setup 1			Setup 2		
	T=0.7	T=0.8	T=0.9	T=0.7	T=0.8	T=0.9
25	2	1	0	1	2	1
30	2	2	0	2	2	2
35	3	2	2	3	2	2
40	2	1	2	3	3	3



(a) (b)

Figure. 5 Illustration of number of migrated loads of: (a) setup 1 and (b) setup 2



(a) (b)

Figure. 6 Illustration of number of migrated loads of: (a) setup 1 and (b) setup 2

Table 4. Performance of different loads

Number of Tasks	Setup 1			Setup 2		
	T=0.7	T=0.8	T=0.9	T=0.7	T=0.8	T=0.9
25	0.67	0.74	0.78	0.69	0.78	0.8
30	0.65	0.77	0.83	0.68	0.79	0.83
35	0.68	0.73	0.87	0.68	0.78	0.82
40	0.63	0.79	0.87	0.69	0.79	0.85

Fig. 5 (b) depicts the No.Ts migration performance of the setup 2. When the No.Ts are 25, two tasks are migrated for the threshold values of 0.8. For all the threshold value, two tasks are migrated when the No.Ts is 30. For the threshold value of 0.7, three tasks are migrated when the No.Ts is 35. When the number of the tasks is 40, three tasks are migrated for all the threshold values. The number of load migration performance is shown in the Table 4.

Fig. 6 represents the number of migrated tasks and the load of the virtual machine when the numbers of tasks are 25, 30, 35, and 40 while applying the threshold value of 0.7, 0.8, and 0.9 for setup two. Fig. 6 (a) shows loads of the virtual machine when the numbers of tasks are 25, 30, 35, and 40 for the threshold values of 0.7, 0.8, and 0.9. For the threshold value of 0.7, loads of the virtual machines are 0.67, 0.65, 0.68 and 0.63 when the numbers of tasks are 25, 30, 35, and 40. For the threshold value of 0.8, loads of the virtual machines are 0.74, 0.77, 0.73, and 0.79 when the numbers of tasks are 25, 30, 35, and 40. For the threshold value of 0.9, loads of the virtual machines are 0.78, 0.83, 0.87, and 0.87 when the numbers of tasks are 25, 30, 35, and 40. Similarly Fig.6 (b) shows the performance of the setup 2 with respect to load migration.

4.5 Worst performance

In this section, Table 5 shows the worst performance for both setup 1 and 2 with worst number of migrated tasks for the threshold values of 0.7, 0.8, and 0.9. For the setup 1, the worst performance occurs when the threshold value is 0.7. For the threshold value of 0.7, there are three migration and the load value of VM is high with 0.689. The load value of the CS-FA at the threshold 0.7, and 0.8 are 0.68, and 0.73 respectively.

Table 5. Performance of worst value of migrated tasks and the load for setup 1 and setup 2

Threshold	Setup 1 Number of Migrated Tasks	Load of the VMs	Setup 2 Number of Migrated Tasks	Load of the VMs
0.7	3	0.689	3	0.691
0.8	2	0.736	3	0.782
0.9	2	0.754	3	0.834

Table 6. Performance of average value of migrated tasks and the load for setup 1 and setup 2

Threshold	Setup 1 Number of Migrated Tasks	Load of the VMs	Setup 2 Number of Migrated Tasks	Load of the VMs
0.7	1	0.64	1	0.66
0.8	0	0.79	1	0.73
0.9	0	0.74	0	0.85

For the setup 2, the worst performance occurs when the threshold value is 0.9. For the threshold value of 0.9, there are three migrations besides the load value is high with the value of 0.834. The load value of the CS-FA at the threshold 0.7, and 0.8 are 0.691, and 0.782 respectively.

4.6 Average performance

In this section, Table 6 shows the average number of migrated tasks and the average load of the VMs for setup one and setup two for the threshold values of 0.7, 0.8, and 0.9. In setup one, the average number of migrated tasks is one for the threshold value of 0.7 and there is no need of task migration for the threshold values 0.8 and 0.9. The average load of the virtual machine is 0.64, 0.79, and 0.74 for the threshold values 0.7, 0.8, and 0.9 respectively.

In setup two, the average number of migrated tasks is one for the threshold value of 0.7 and 0.8, and no migrated task for the threshold value of 0.9. The average load of the virtual machine is 0.66, 0.73, and 0.85 for the threshold values 0.7, 0.8, and 0.9 respectively.

4.7 Best performance

In this section, Table 7 shows the best number of migrated tasks and the corresponding load of the VM for setup one and set up two. For the setup 1, the best performance occurs when the threshold value is 0.7. For the threshold value of 0.7, there zero migration since the load of the system is balanced with the value of 0.63. The load value of the CS-FA at the threshold 0.8, and 0.9 are 0.66 and 0.68 respectively.

Table 7. Performance of Best value of migrated tasks and the load for setup 1 and setup 2

Threshold	Setup 1 Number of Migrated Tasks	Load of the VMs	Setup 2 Number of Migrated Tasks	Load of the VMs
0.7	0	0.63	0	0.61
0.8	0	0.66	0	0.63
0.9	0	0.68	0	0.650

Table 8. Comparative analysis

Number of Task	HBB-LB [18]	DLB [18]	HDLB [18]	CMLB [15]	Proposed CA-FA
10	0	1	1	0	0
20	1	1	2	1	0
30	3	4	4	2	2
40	4	6	7	3	2

For the setup 2, the best performance occurs when the threshold value is 0.7. For the threshold value of 0.7, the load is balanced with the value of 0.61. The load value of the CS-FA at the threshold 0.8, and 0.9 are 0.63 and 0.65 respectively.

4.8 Comparative discussion

Table 8 shows the comparative discussion of the proposed CA-FA LB with the existing methods, such as HBB-LB, DLB, HDLB and CMLB. At 10^{th} task, the existing methods such as DLB and HDLB migrates one task, whereas the proposed CA-FA need no task migration. The proposed CA-FA LB migrates zero when the No.Ts is 20 while the existing HDLB migrates two tasks for the same No.Ts.

If the No.Ts is 30, the proposed LB method migrates two tasks while the existing LB methods, such as HBB-LB, DLB, and HDLB migrate three, four, and four tasks respectively. The scheduling method in the HBB-LB, DLB and HDLB techniques tends to migrate more task due to weighting method. The CML method has the disadvantages of low convergence rate. In the 40^{th} task, the proposed CA-FA migrates two task and other existing method migrates the 4, 6 and 7 tasks respectively. The convergence rate of the proposed method is high and has less number of tuning parameter than existing method. So, the proposed method has the higher performance than existing method. Hence, compare to the traditional LB algorithm proposed CA-FA algorithm shows the better performances.

5. Conclusion

In this research work,LB algorithm is proposed namely CA-FA. The LB algorithm calculates the deciding factor of all the virtual machines and fills the deciding list. Then, it calculates the selection factor of each task and fills the selection list and checks the load of the virtual machine. If the load of the virtual machine is greater than the balanced threshold value then, the LB algorithm assigns the task which has better selection factor to the virtual machine which has a better deciding factor. Then the allocated task is removed from the selection list, and the virtual machine is removed from the deciding list. Similarly, all the tasks are assigned to the virtual machines by the proposed LB method. The proposed LB method is evaluated with the existing LB methods, such as HBB-LB, DLB, CMLB and HDLB for the evaluation metrics load and capacity. From the experimental results, understand that the proposed CA-FA has the better performance by migrating only two tasks while the existing LB methods, such as CMLB, HBB-LB, DLB, and HDLB migrates more than two tasks. The proposed method has the advantages of high convergence rate and less number of tuning parameter.The proposed method migrate 2 task, while existing method requires 7 task for migration.In future work is extended as energy consumption and SLA violation will be reduced using an efficient multi objective optimized strategy.

References

[1] M. Adhikari and T. Amgoth, "Heuristic-based load-balancing algorithm for IaaS cloud", *Future Generation Computer Systems*, Vol. 81, pp. 156-165, 2018.

[2] S. Suresh and S. Sakthivel, "A novel performance constrained power management framework for cloud computing using an adaptive node scaling approach", *Computers & Electrical Engineering*, Vol. 60, pp. 30-44, 2017.

[3] M. Kumar and S. C. Sharma, "Deadline constrained based dynamic load balancing algorithm with elasticity in cloud environment", *Computers & Electrical Engineering*, Vol. 69, pp. 395-411, 2018.

[4] L. Tang, Z. Li, P. Ren, J. Pan, Z. Lu, J. Su, and Z. Meng, "Online and offline based load balance algorithm in cloud computing", *Knowledge-Based Systems*, Vol. 138, pp. 91-104, 2017.

[5] T. Tamilvizhi and B. Parvathavarthini, "A novel method for adaptive fault tolerance during load balancing in cloud computing", *Cluster Computing*, Vol. 5, No. 5, pp. 10425-10438, 2019.

[6] A. S. Milani and N. J. 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.

[7] R. K. Naha and M. Othman, "Cost-aware service brokering and performance sentient load balancing algorithms in the cloud", *Journal of Network and Computer Applications*, Vol. 75, pp. 47-57, 2016.

[8] G. Xu, J. Pang, and X. Fu, "A load balancing model based on cloud partitioning for the public cloud", *Tsinghua Science and Technology*, Vol. 18, No. 1, pp. 34-39, 2013.

[9] F. Ramezani, J. Lu, and F. K. Hussain, "Task-based system load balancing in cloud computing using particle swarm optimization", *International Journal of Parallel Programming*, Vol. 42, No. 5, pp. 739-754, 2014.

[10] K. Dasgupta, B. Mandal, P. Dutta, J. K. Mandal, and S. Dam, "A genetic algorithm (ga) based load balancing strategy for cloud computing", *Procedia Technology*, Vol. 10, pp. 340-347, 2013.

[11] L. Wang, Z. Wang, S. Hu, and L. Liu, "Ant colony optimization for task allocation in multi-agent systems", *China Communications*, Vol. 10, No. 3, pp. 125-132, 2013.

[12] M. Lawanyashri, B. Balusamy, and S. Subha, "Energy-aware hybrid fruitfly optimization for load balancing in cloud environments for EHR applications", *Informatics in Medicine Unlocked*, Vol. 8, pp. 42-50, 2017.

[13] V. Jeyakrishnan and P. Sengottuvelan, "A hybrid strategy for resource allocation and load balancing in virtualized data centers using BSO algorithms", *Wireless Personal Communications*, Vol. 94, No. 4, pp. 2363-2375, 2017.

[14] V. A. Xavier and S. Annadurai, "Chaotic social spider algorithm for load balance aware task scheduling in cloud computing", *Cluster Computing*, Vol. 22, No. 1, pp. 287-297, 2019.

[15] V. Polepally and K. S. Chatrapati, "Dragonfly optimization and constraint measure-based load balancing in cloud computing", *Cluster Computing*, Vol. 22, pp.1099-1111, 2019.

[16] G. Reddy, N. Reddy, and S. Phanikumar, "Multi Objective Task Scheduling Using Modified Ant Colony Optimization in Cloud

Computing", *International Journal of Intelligent Engineering and Systems,* Vol. 11, No. 3, pp. 242-250, 2018.
[17] K. A. Sultanpure and L. S. S. Reddy, "Job Scheduling for Energy Efficiency Using Artificial Bee Colony through Virtualization", *International Journal of Intelligent Engineering and Systems,* Vol. 11, No. 3, pp. 138-148, 2018.
[18] P. V. Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments", *Applied Soft Computing,* Vol. 13, No. 5, pp. 2292-2303, 2013.