IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.8, Issue.7, July 2020 E-ISSN: 2347-2693

MAHO: Modified Ant Honey Bee Optimization Algorithm for Load Balancing in Cloud Environment

Tony Bayan[1*], Priyanka Sarma[2]

[1]K.C. Das Commerce College, Assam, India,
[2]University of Science and Technology Meghalaya, Meghalaya, India

Corresponding Author: tonybayan@gmail.com

DOI: https://doi.org/10.26438/ijcse/v8i7.122131 | Available online at: www.ijcseonline.org

Received: 05/July/2020, Accepted: 20/July/2020, Published: 31/July/2020

***Abstract*—** Cloud computing mainly does not focus on local resource instead; it uses shared computing resources applications or resources. It has emerged as a new type of computing for accessing present network, managing computer resources and managing distributed computing across the network in order to achieve high degree of precision and reliability various challenges needs to be addressed. One of the challenges in cloud computing is load balancing. Load balancing is important due to the fact that it allows achieving balance in the load by distributing it across the system to all its nodes. Cloud environment allows various ways to achieve load balancing. This includes managing the load on CPU, network load and the load capacity of storage. The greatest impact of balancing the load in cloud computing environment is that it has higher satisfaction of the users as well as it utilizes the resources efficiently. Proper load balancing support substantial improvement of the system, building a fault tolerant system by creating backup and increase flexibility of the system so that it adapts the modification. In cloud computing, there are various algorithms to achieve load balancing and these algorithms behave differently with its some advantages and disadvantages. In this paper we present an Modified Ant Honey Bee based optimization algorithm (MAHO) to achieve load balancing. The results are analyzed with existing load balancing algorithm based on make span metrics.

***Keywords*—**Cloud computing; Load balancing; Static Load balancing; Dynamic Load balancing; Algorithms; Load balancer; Load balancing metrics.

I. INTRODUCTION

Cloud computing is an emerging method for large-scale distributed computing. Cloud computing has moved data and computing to large data centres form mobile and local personal computers [1]. Cloud computing is a need for high performance computing such as scientific and engineering application modelling, simulation and analysis of complex systems like climate, galaxies etc. [2]. It can be termed as an on-demand network access to a shared pool of computing resources [3]. Cloud computing is capable of harnessing the power of Local as well as Wide Area Network (WAN) to use resources remotely, resulting a solution that has efficiency in cost for most of real-life requirement [4]. These resources are allowed to be quickly supplied and withdrawn with the help of minimal request from service provider thus achieving availability [3] [5]. This results in an exponential growth of adapting cloud computing by industries also expanding the data centre. As a result, there is a dynamic increase in energy consumption that hampers the environment in terms of carbon foot prints [3].

A. CLOUD SERVICES

Fig1.1 shows the layered architecture of cloud services. Cloud services are basically divided into six types namely Infrastructure as a Service (IaaS), Network as a Service (NaaS), Platform as a Service (PaaS), Data as a Service (DaaS) and Software as a Service (SaaS). These services fill into Hardware Level, System Level and Application Level. The Hardware Level provides storage, CPU cycle and Network services. The System Level provides platform to run end user's application. The Application Level provides specific applications for dedicated end-users.

***Infrastructure as a Service (IaaS)*:** In IaaS, the consumers are allowed to deploy and use arbitrary software such as operating systems remotely. In this type of service, the consumer only has access and control over the storage and deployed application. Management of the underlying cloud infrastructure is not a concern for the consumer.

***Platform as a Service (PaaS*):** In this type of service the consumers are allowed to create to software using tools or libraries from the cloud provider. The consumer also has control over the software deployment and configuring settings. The cloud provider provides the network servers storage and other services.

Software as a Service (SaaS): This service allows software to be hosted centrally and accessed via a thin client e.g. a browser.

This service forms the layered architecture of Cloud Computing. It is basically the design of software

applications that users on-demand services through internet. Cloud Architecture are based on infrastructure that is used only when its needed that draws the necessary resources on-demand and performs the specific job and then sets it free. This service can be accessed from anywhere though thin client such as web browser etc.

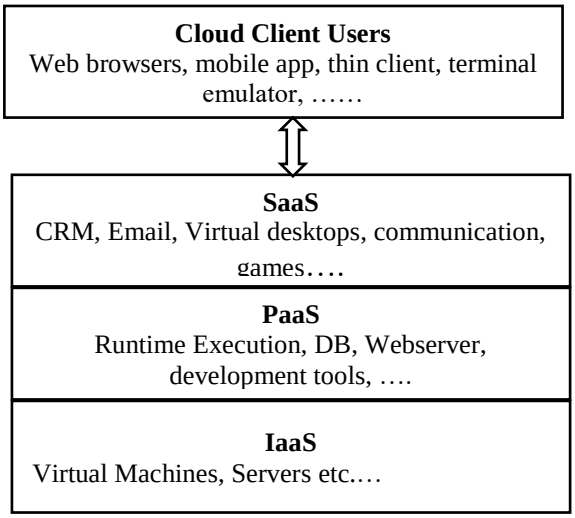

Figure 1. Layered Architecture of Cloud

This service forms the layered architecture of Cloud Computing. It is basically the design of software applications that users on-demand services through internet. Cloud Architecture are based on infrastructure that is used only when its needed that draws the necessary resources on-demand and performs the specific job and then sets it free. This service can be accessed from anywhere though thin client such as web browser etc.

B. TYPES OF CLOUD

Basically, clouds are of the following types:

(a) Private Cloud: In private cloud, data and processes are serviced within the corporate firewall of an organization. The management of the cloud is wholly dependent on local manpower. A Cloud Provider is not used.

(b) Public Cloud: In public cloud, a Cloud Service Provider manages and makes resources available to one over the internet via web applications. The resources are dynamically provisioned and de-provisioned. Resources are basically shared between users.

(c) Hybrid Cloud: Amalgamation of private and public clouds is termed as hybrid cloud. This type of cloud consists of multiple internal or external providers.

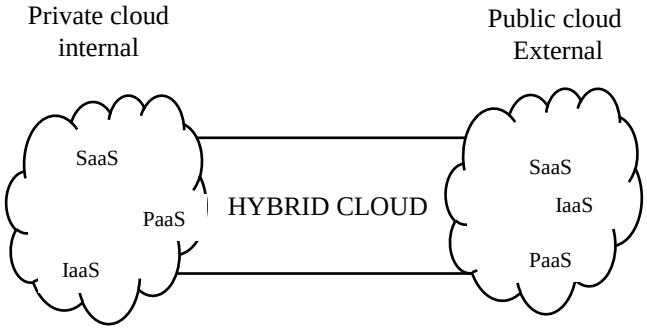

Figure 2: Public, Private and Hybrid Cloud

C. VIRTUALIZATION

Cloud computing is vastly supported by virtualization. Virtualization is separating the underline hardware from the operating system. As a result, the entire operating system along with application can be transferred from one physical machine to another. Virtualization helps in scalability. Virtualization is of two types: **(a)** Client installed **(b)** Hypervisor. In client installed virtualization software an operating system is installed on a piece of hardware and on top of its client virtualization software is installed. Using the client virtualization software, instances of different operating systems can be created and migrated as when needed. In case of hypervisors are types of operating system which are installed on to hardware. E.g. ESxi.Basically, virtualization refers to abstraction of logical resources away from the underlying hardware resources to increase flexibility, reduce costs etc. Under Virtualization computing environments can be created, expanded, shrunken or moved dynamically.

D. SCALABILITY

One of the most important feature of cloud computing is Dynamic Scalability. Dynamic resizing can be termed as a feature that allows server to resize the virtual machine as per requirement of resources. Equally distributing the load amongst the nodes is an important aspect that is too considered for efficient working of the nodes. Cloud scalability mainly is of two types:

a. **Horizontal Scalability**: It is basically connecting multiple resources/servers to work as an individual computing unit. Same computing units are connected together to scale out the computation.

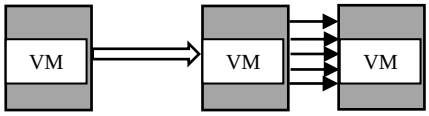

Figure 3. Horizontal Scaling of VM

b. **Vertical Scalability**: The existing hardware or software is increased in order to meet the requirement. It is also known as scaling up. Auto scaling is the ability to scale up or down as per

requirement dynamically. When the demand increases the system is scaled up automatically and with the decrease of demand it shrinks. The scaling infrastructure is shown in Fig 4 which helps in achieving load balancing.

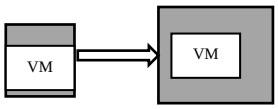

Figure 4. Vertical Scaling

c. **Auto scaling** is the ability to scale up or down as per requirement dynamically. When the demand increases the system is scaled up automatically and with the decrease of demand it shrinks. The auto scaling infrastructure is shown in Figure 5 which helps in achieving load balancing.

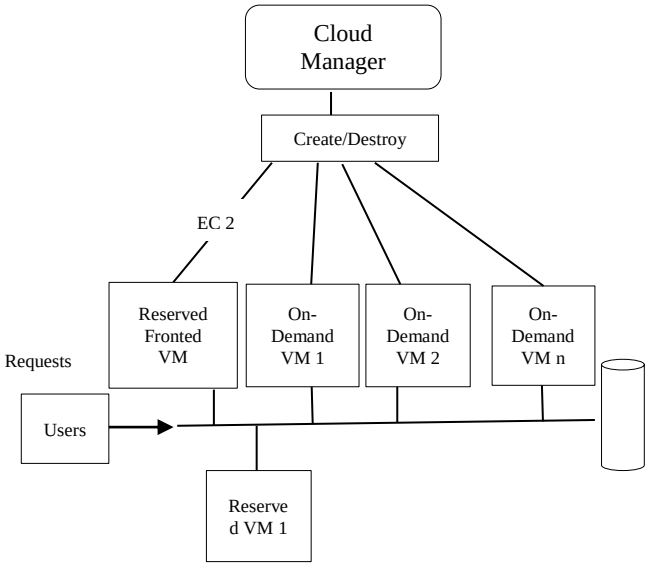

Figure 5. Auto scaling in cloud

This paper is organized as follows. Section II gives an insight to load balancing its types and the policies it incurs. Section III discusses the various existing load balancing algorithms in cloud computing environment. In section IV, proposed work is discussed along with analysis of some existing algorithms. Section V discusses the simulation results using cloudsim finally; the conclusion of this paper is given in section VI.

II. LOAD BALANCING

Large data centre keeps up allocating and de-allocating services as per need. To achieve high degree of utilization and throughput, services need to be balanced. Load balancing is a method for distributing the work load across nodes in the cloud. Workload is the total processing time that is needed to complete all the tasks assigned. The primary goal of load balancing is to avoid situation in which one node is overloaded while others are idle. It also ensures efficient and fair distribution of computing resources. To improve efficiency and to remove overheating of Nodes Load balancing plays an important role. Load balancing is distributing workloads within different nodes available in the cloud.

Load balancing also helps in achieving green computing i.e. reducing the carbon emission. Minimizing the consumption of resources using Load Balancing can result in reduced cost and helps to achieve green computing [3][6]. Scalability, one of the features of cloud computing can also be achieved through load balancing [7]. The basic factors are:

(a) **Limiting Energy Consumption:** Overheating of nodes or Virtual Machines due to excessive overload results in high amount of energy consumption. This energy consumption can be reduced using Load Balancing [8].
(b) **Reducing Carbon Emission:** with the increase in Energy consumption carbon emission also increases equally. Load Balancing helps in reducing the consumption of energy thus reducing carbon emission [8].

The main goal of Load Balancing as discussed by authors of [9] [10] are

(a) Substantial improvement of performance.
(b) Stability maintenance of the system.
(c) Building a fault tolerant system by creating backup.
(d) Increase flexibility of the system so that it adapts the modification.

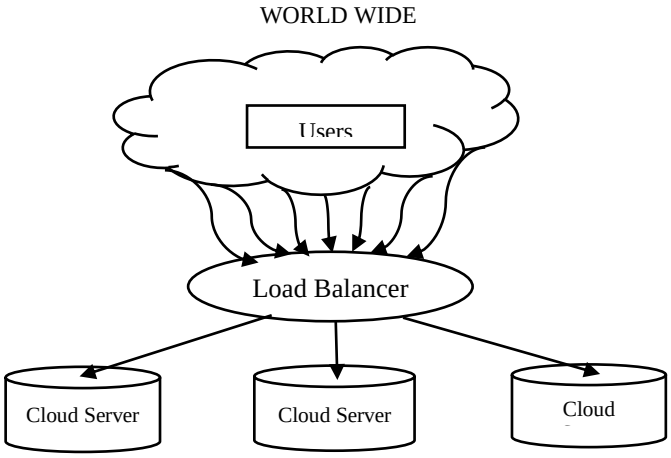

Figure 6. Working of a Load Balancer

The following subsection gives us an idea of the different types of load balancing in cloud environment and the policies used in Dynamic Load Balancing Algorithm.

A. *Classification of Load Balancing Algorithm*

The classification of load balancing Algorithm is mainly based on Process Origin and Current state of the system.
Based on process origin, Load Balancing algorithm is classified as [11] [12]13].

(a) **Sender Initiated**: In this type of load balancing algorithm, a request is initiated by the client and waits until availability of a receiver for accepting the workload
(b) **Receiver Initiated**: In this type a sender ready to share the workload has to wait until a receiver initiates an acknowledged request.
(c) **Symmetric**: It is an amalgamation of both senders initiated and receiver initiated.

There are two types of Load Balancing Algorithm based on the present state of a system

a. Static Algorithm*:* Static Algorithms are best suited for environments that contain resources of same type and where changes are less acceptable [14].

b. Dynamic Algorithm: Dynamic Algorithms are suitable for heterogeneous environment [14].

B. ***Static Load Balancing Algorithm***
This type of load balancing algorithm requires previous knowledge about the resources and applications of the system. The present state of the system is independent of balancing load. These types of load balancing algorithm, allocates a task/jobs to a processor or fix node [15]. In this type of load balancing the work load is distributed to nodes with respect to the evaluated performance of the processors. Since this type of algorithm is non-pre-emptive in nature, task allocated to a virtual machine is always executed at the same [15].

The main disadvantage of Static Load Balancing is that a processor is assigned a task only after the process is created. A task allocated to a particular virtual machine remains the same until it completes its execution. It does not allow changes for choosing virtual machine irrespective of the fact that machine is over loaded or under loaded. Selection of virtual machine is fixed as an when tasks are created.

C. ***Dynamic Load Balancing Algorithm***
The current state of the system plays a vital role in balancing the work load. Dynamic Load Balancing Algorithm has the capability of moving a task/job from over utilized node to an underutilized node for faster execution. In dynamic Load Balancing if any node fails the whole system does not halt. A frequent degree of communication amongst the nodes is always active that generates more messages than non-distributed in the dynamic load balancing algorithm.

D. Dynamic Load Balancing Policies
A Dynamic Load Balancing Algorithm has five major policies [16] and they are as follows:

a. Transfer Policy
This policy used for selecting a task or a process from a local machine to a remote machine.

b. Selection Policy
The selection or identification process of a processor or machine that takes part in load balancing is governed by this policy.

c. Location Policy
The policy used by a node to share the task by an over loaded machine.

d. Information Policy
This policy is used for collecting all information that is needed for load balancing.

e. Load Estimation Policy
The total approximate work load of a processor or machine is decided by this policy.

III. EXISTING LOAD BALANCING ALGORITHMS IN CLOUD COMPUTING

Different load balancing algorithms are available in cloud computing environment, these algorithms are studied and compared based on the predefined metric make span, Following are the load balancing techniques that are widely used in clouds.

Round Robin Algorithm
B Sotomayor et al [17] proposed an algorithm similar to round robin algorithm of scheduling. This is a static load balancing algorithm where the tasks are distributed amongst the virtual machine evenly. Each task after being allocated to a virtual machine, the remaining tasks are then allocated to the virtual machine allocated first in cyclic manner. This process is repeated until all tasks are allocated to virtual machines. This method does not consider the allocation procedure from remote system rather it does it locally. This disadvantage of this method is that irrespective of whether a virtual machine is already loaded or not new tasks might be allocated to the already loaded virtual machine which should not be encouraged. This type of load balancing is not suitable for cloud environment where the number of tasks is dynamic.

Modified Throttled algorithm: Shridhar G. Domanal and G. Ram Mohan Reddy [18] proposed a Modified Throttled algorithm for cloud load balancing. This method is based on round robin algorithm. According to the author, this algorithm basically focuses on how jobs are allocated to available virtual machines. An index table of virtual machines and also state of virtual machines (busy/available) is maintained by this algorithm. Initially, this algorithm selects a virtual machine at first index depending upon the state of virtual machine. If the new request arrives, the virtual machine at the previous virtual machine index+1 is chosen depending on the state of virtual machine [7].

Shortest Job First Based Load Balancing Algorithm
K Suchita et al [19] proposed a static load balancing algorithm based on the shortest job first algorithm for scheduling. In this method the allocation of tasks to a particular virtual machine is done depending upon the length of the task. The task with the shortest length is allocated to the available virtual machine. This method is suitable for environment where the task length does not vary. A task with higher length may have to wait for long, waiting for completion of shortest length tasks. The disadvantage of this method is that priority is not considered for tasks.

Ant Colony Optimization (ACO):
Medhat A.Tawfeed et.al [20] in 2013 presented a dynamic cloud task scheduling based Ant Colony Optimization Approach for allocation of incoming tasks to virtual machine's efficiently to minimize the make span. The author performed simulation of the method using Cloudsim simulation tool kit. In the paper, it compares task scheduling based Ant colony optimization approach with different scheduling algorithm, and finds that Ant colony gives better result as compared to RR, FCFS scheduling algorithms. Mainly ACO algorithm [21] is proposed for load balancing of nodes and its aim is to efficiently distribution of workload among the nodes [22]. The idea is that ants will start approaching towards the food source starting from the head node when the request is initialized and it maintains a table for its route. This is basically the pheromone table. This helps the Ants to records their data which helps in making the decision in future, keeping record of every node visited. As and when ants are created the pheromone table is updated with the latest value. Each ant has a result set associated with it that is recorded to find a complete solution. This result set is continuously updated. The main aim of the ant is to find a new food source using this result set. Thus, the main aim of this algorithm is that it efficiently distributes the work load among the nodes.

ACCLB- An algorithm based on ant colony and complex network theory is proposed by Z. Zhang et al. [23]. Load balancing is achieved by using the feature from small-world and scale-free. The advantage of this method is that it is suitable for overcoming heterogeneity. This method also supports dynamic environment. This method promises fault tolerance at its best. Apart from support of fault tolerance it also is scalable. As a result, it helps in improving the overall performance of the system.

Honey Bee Behaviour inspired Load Balancing (HBB-LB): Dinesh B. L.D, P.V. Krishna [24] proposed a Honey Bee Behaviour inspired Load Balancing (HBB-LB) Technique, according to [1][25], HBB-LB helps to distribute the load evenly for the virtual machines maximizing the overall throughput of the system. This method also accepts priority of tasks to be executed in virtual machine. Overloaded and under loaded or balanced virtual machines are identified by examining the workload of virtual machine. Based on these VMs are grouped [22]. According to the load on virtual machine the task is scheduled on virtual machines, which is removed earlier. To find the least loaded virtual machine for the current task, tasks that are removed from the system from over loaded virtual machine are helpful. In the next steps forager bee is used as a Scout bee [26].

Particle Swarm Optimization (PSO): Kennedy and Eberhart [27] proposed a load balancing algorithm named as Particle Swarm Optimization (PSO). It is a Meta heuristic algorithm based on optimization technique. The PSO is a self-adaptive global search optimization technique. It is same as Genetic algorithm (GA). Bit allocation of tasks is done in this method. It is seen that at least three times cost savings is possible with PSO if compared to best resource selection (BRS) based mapping for application workflow.

IV. PROPOSED WORK

Load Balancing based on Ant Colony Optimization (ACO)
The Ant Colony Optimization algorithm simulates the foraging behaviour of ant. When a group of ants initiates search for food, in order to communicate with other ant's it uses pheromone (a special kind of chemical). Initially ants start searching for foods randomly. On finding a food source it discharges pheromones along the path. This helps other ants to locate the food source following this pheromone trail on ground. The nearest food source is chosen by the other ants as the intensity of pheromone to the nearest food source is maximum. This pheromone depositing along the path and pheromone following behaviour of ants becomes the inspiration source for Ant Colony Optimization Algorithm.

ACO Algorithm approach:
Initialization of algorithm: Initially the pheromone values along with its parameters are initialized.
Initialization of ants: N number of tasks is initialized to M number of ants. Each ant builds solution to M number of resources. Ants are randomly selected to build a constructive direction for each iteration.

Local Pheromone Updating: After initialization of the ants and mapping, pheromone value is updated by local pheromone updating rule. The local pheromone update is performed by all the ants after each construction step. Each ant applies it only to the last edge traversed.

Limitations of ACO
1. Recruitment Strategies (methods used to communicate previous search experiences to other members of the colony) are indirect
2. Insufficient Exploration.
3. It is adaptive in nature.
4. Its convergence is guaranteed but time to convergence is uncertain.

5. Coding is not straightforward.
6. It is prone to falling in the local optimal solution.

Basic Honey Bee Load Balancing Algorithm (HBL)

The basic idea behind the Bee Colony Optimization is to successfully solve difficult combinatorial Optimization problems. A colony of honey bee extends to its surroundings for sources e.g. flower patches and then as a result these bees harvests nectar or pollen from these sources. Once a particular food source is found, a part of this honey bee cluster moves out to find a new food source. Upon finding a new food source the scout bees goes in the field surrounding the hive and check for quality beneficial. When they return to the hive, the scouts collect the food harvested. There is an area in the hive called as the "dance floor", where waggle dance is performed by the bees that found a very beneficial food. Through the waggle dance a scout bee passes the position of its search to idle spectator, which helps in the using of the flower patch. Here the duration of the dance is according to the scout's rating of the food source, to harvest the best rated flower patches more foragers get recruited. When dance is done, the scout return to the food source it found to see more food. Till the food is profitable, food sources will be posted by the scouts when they return to their hive. Foragers who are recruited recently may waggle dance as well, which will step-up the recruitment for highly profitable flower patches. This auto catalytic process will go on to find most beneficial flower patches. The Honey Bee behaviour inspired load balancing algorithm was proposed, which aims to achieve well balanced load across VMs to maximize the throughput and to balance the priorities of tasks on the VMs. Hence, the amount of waiting time of the tasks in the queue is minimal. Using this algorithm average execution time and reduction in waiting time of tasks on queue were improved. This algorithm works for heterogeneous type of systems and for balancing non pre-emptive independent tasks.

Algorithmic Approach

1. Cloud Setup: The cloud is set up by using data centres, physical machines and virtual machines

2. Initialization: All virtual machine loads are initialized to zero, the jobs need to be assigned are send as cloudlets. The threshold value is chosen based on the virtual machine capacity.

3. Load Balancing: The algorithm checks periodically load on each virtual machine and migration or transfer of load takes place according to the following approach. This algorithm identifies the highly loaded VM and finds the less loaded VM then transferring followed, so that the loads are distributed evenly, the response time minimizes and resource utilization increases. The task can be considered as a bee and it is searching for a less loaded VM (food source), when it finds the suitable VM assignment of task to VM takes place, and the next task also tries to assign to the same VM, this assignment continues until the load on the VM reaches threshold.

Limitation of Honey Bee Load Balancing Algorithm:

1. It is adaptive in nature.
2. Traditional Honey Bee is triggered only when system balance becomes unbalanced.
3. Task migration is a continuous process.
4. Choosing a VM is random**.**

Modified Ant-Honey Bee Optimization (MAHO) Algorithm

Ant colony algorithm is a solution to optimization problem by selecting the shortest path from their nest to food source. The basic idea is: when an ant moves along a path from their nest to a food source or vice versa, they release pheromone along the path. Initially the path is selected randomly. With concentration being high, the ants follow the path to get to their food source from their nest.

Relating the traditional ACO to load balancing in cloud environment, ants continuously updated single result set and this pheromone i.e. trailing mechanism is used to find out overloaded and under loaded virtual machines in the system. The working of Ant Colony Optimization algorithm along with the foraging behaviour is an inspiration of the ants finding the food source. This has become an inspiration for many researchers, and application of the same has been seen in various fields to solve problems. And in case of Honeybee algorithm, the Foraging behaviour of honey bee gives information to the other bees about the position of food source. Initially a bee starts its journey as a worker i.e. it has no information about the food source. When it finds a food source, it stores the information about its source when sharing the knowledge within the dance space. A special dance form called the Waggle dance is performed by the bees to describe the amount of food source available and the quality. Through this dance the bees knows about the distance from the bee hive.

The main goal of Honeybee Load balancing is distributing the workload such that underutilization and over utilization of the resources is avoided. In order to achieve this the incoming task is allocated the VM where the number of tasks currently processing by this VM is less than number of tasks currently processing by other VMs.

We have noticed that during the execution of algorithm, when a VM is overloaded ACO creates random number of Ants with same pheromone values and to wander in the nodes to find the available VM. This incurs some amount of time.

We propose a load balancing algorithm named as "Modified Ant Honey Bee Optimization (**MAHO**)" algorithm for efficient load balancing of the tasks to virtual machines by hybridization of Ant Colony optimization Algorithm (ACO) and Honey Bee Load Balancing Algorithm (HBL) for Load Balancing in cloud environment. This algorithm proposes two main functions:

1. To choose a virtual machine effectively and

2. To allocate task to a virtual machine which is likely to under load?

Table.1 Mapping parameters of Proposed MAHO Algorithm with Cloud Environment

Ant Nest, Honeybee hive	Cloud Environment
Ants, Honeybees	Tasks (Cloudlets)
Food Source	Virtual Machine (VM)
Ant /Honeybee /searching/foraging a food source	Loading of a task to VM (Virtual Machine)
Honeybee getting exhausted at a food source	VM is overloaded condition
Foraging bee finding a new food source	Task removed will be rescheduled to an under loaded VM having Highest Capacity
Calculating pheromone value	Calculating capacity/load of VM's
Updating pheromone values	Maintain Index table

Stages of the proposed "MAHO" Load Balancing Algorithm are described as follows:

STAGE A: Initialize Pheromone

The amount of virtual pheromone trail $\tau_{i,j}$(t) on the edge connects task i to VM j. The initial amount of pheromone on edges is assumed to be a small positive constant τ_0(homogeneous distribution of pheromone at time t = 0).
Where,
$\tau_{i,j}$(t) shows the pheromone concentration at the time 't' on the path between task i and VM j.

The value of pheromone is initialized to the capacity of each Virtual Machine (VM).

STAGE B: VM choosing rule for next Task using the behaviour of Honey Bee (Input is VM List)

B.1 The fitness of the bee is evaluated in [29]

$$fit_{i,j} = \frac{\sum_{i=1}^{n} tasklength\ i,j}{Evaluate\ Capacity\ of\ VM\ j (Capacity\ j)} \text{ ----------(1)}$$

Where, $fit_{i,j}$ is that the fitness of the bee's population of i in VMj. Task length is that the length of the task that has been submitted in VMj and capability is that the capability of VMj calculating supported the subsequent

Capacity of a virtual machine is defined by [29]

$$C_j = P_{e_{numj}} * P_{e_{mips}} + Vm_{bwj} \text{ ----------(2)}$$

Where,
$P_{e_{num\ j}}$ is processing element i.e. number of processors in VM_j
$P_{e_{mips}}$ is million instruction per second of all processor in VM_j
$P_{m_{bw\ j}}$ is the bandwidth of VM_j

B.2 Select m sites for neighborhood

Scout bees with the best fitness are chosen as choose Bee" and therefore the sites visiting by them are chosen from neighbourhood of m VMs.

B.3 Recruit bees for selected sites
Send a lot of bees to neighbourhood of the most effective VM, then judge the fitness supported [29]

$$fit_{i,j} = \frac{\sum_{i=1}^{n} task\ length\ i,j + input\ file\ length}{Evaluate\ Capacity\ of\ VM\ j\ (Capacity\ j)} \text{ ----------(3)}$$

Where,' input file length' is that the length of the task before execution.

B.4 Calculate the best fitness
choose the most effective Fitness Bees from every Patch and Assign Task to Virtual Machine.

B.5 Return the best available VM

STAGE C: Pheromone Updating

After the completion of a tour, each ant k lays a quantity of pheromone $\Delta\tau_{i,j}^{k}(t)$ computed by Eq. (4) on each edge (i, j) that it has used [17].

$$\Delta\tau_{i,j}^{k}(t) = \begin{cases} \frac{Q}{L^k(t)}, & if(i,j) \epsilon T^k(t) \\ 0, & if(i,j) \notin T^k(t) \end{cases} \text{ ----------(4)}$$

Where $T^k(t)$ is the tour done by ant k at iteration t, L^k(t) is its length (the expected makespan of this tour) that is computed by Eq. (5), and Q is an adaptive parameter [17].

$$L^k(t) = arg\ max_{j \epsilon\ J}\{sum_{i \epsilon\ IJ}(d_{ij})\} \text{ ----------(5)}$$

Where, IJ is the set of tasks that assigned to the VMj.
After each iteration pheromone updating which is applied to all edges [17] is refreshed by Eq. (6).

$$\tau_{ij}(t) = (1-\rho)\,\tau_{ij}(t) + \Delta\tau_{ij}(t) \text{ ----------(6)}$$

Where ρ is the trail decay, 0 < ρ < 1 and $\Delta\tau_{ij}(t)$ is computed [17] by Eq. (7).

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \text{ ----------(7)}$$

When all ants complete a traverse, an elitist is an ant which reinforces pheromone on the edges belonging to the best tour found from the beginning of the trial (T^+), by a quantity Q/L^+, where L^+ is the length of the best tour (T^+). This reinforcement is called global pheromone update [17] and computed by Eq. (8).

$$\tau_{ij}(t) = \tau_{ij}(t) + \frac{Q}{L^+} \text{ if (i, j) } \epsilon\ T^+ \text{ ----------(8)}$$

Pseudo Code of Proposed "Modified Ant Honeybee Optimization Algorithm" (MAHO)

Input: List of Cloudlets and list of VM's
Step 1. *Initialization*
set current_ iteration = 0;
set optimal_solution = 0;
set initial value of pheromone = cc for each path between tasks and VM
Step 2. *Place m ants on all starting VMs randomly*
Step 3. *For k =1 to m DO*
place the starting VM of the kth ant in table_pheromone.
optimal_VM () *Choosing the next VM for task allocation.*
Insert the VM in table_pheromone.
Step 4. *For k = 1 to m*
compute the length of the tour according to equation …….. (5)
update current_optimal with the best solution
Step 5. *apply the local pheromone according to equation …….. (6)*
Step 6. *apply global pheromone update according to equation …….. (8)*
Step 7. *increment current_iteration+=1*
Step 8. *if(current_iteration<t_{max})*
empty table_pheromone
repeat Step 2
else
optimal_solution
Step 9. *Allocate task to VM*

*** optimal_VM ()**

Input: VM List
Step 1. *Set vm_id = -1*
Step 2. *Get food source i.e. recruit bees for finding available VM*
Step 3. *Calculate the fitness of VM's using equation …….. (1)*
Step 4. Based on the best fitness, the VM is identified.
Step 5. vm_id of the suitable VM is returned.

V. SIMULATION RESULTS

This section discusses the different load balancing algorithms based on simulation done in cloudsim [28]. Cloudsim is a tool for modelling and simulating cloud environment. In order to simulate we considered the number of Data Centre as **50** and the number of cloudlets in the range **100** to **1000.** We compute the make span of the system. Make span is the maximum completion time for all tasks. The model was reconfigured according to the following parameters:

Table 2. VM parameters

Parameters	Value
Data Centre OS	Linux
Memory of VM	256mb
MIPS	500
No. Of Processors	1
Size	1000

Comparison of Round Robin (RR) and Shortest Job First (SJF) Load Balancing Algorithm
In figure 7, the comparison of Round Robin and Shortest job First algorithm is shows based on the parameters as stated in Table 1.

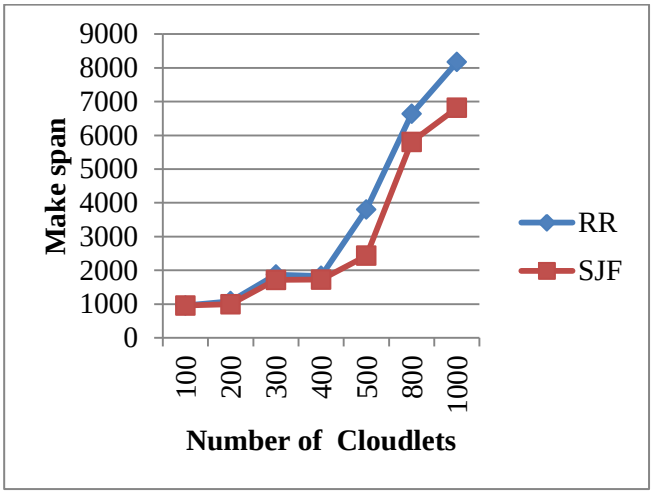

Figure 7. RR vs SJF

As it can be seen that for fewer number of cloudlets/tasks both Round Robin and Shortest Job First performs equally in terms of make span. But with the increase in the number of cloudlets SJF performs better than Round Robin. This is due to the fact that RR allocates tasks to a virtual machine in cyclic manner and does not consider whether a VM is overloaded or not. But in case of SJF the task with shortest length is allocated to a virtual machine thus reducing the overall make span.

6.2 Comparison of Ant Colony Optimization (ACO) and Honey Bee (HBL) Load Balancing Algorithms
Figure 8 shows the comparison of Ant Colony Optimization Algorithm and Honey Bee load balancing algorithm based on the parameters as stated on table 1

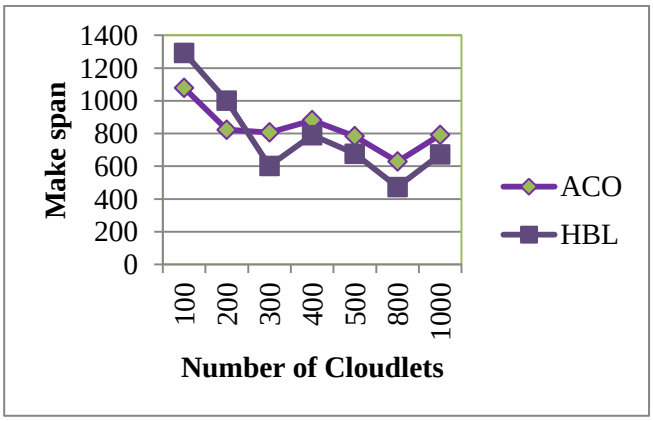

Figure 8. ACO vs. HBL

As seen from the figure ACO performs better than HBL for less number of cloudlets. But as the number of cloudlets increases HBL outperforms ACO in terms of make span. This is due to the fact in case of HBL; initially

tasks are allocated to virtual machine randomly. Depending on the current state of virtual machine all upcoming tasks are allocated to under loaded virtual machine thus minimizing the make span of the system.

6.3 Comparison of RR vs. SJF vs. ACO vs. HBL

Figure 9 cumulatively shows the comparison of all the load balancing algorithms.

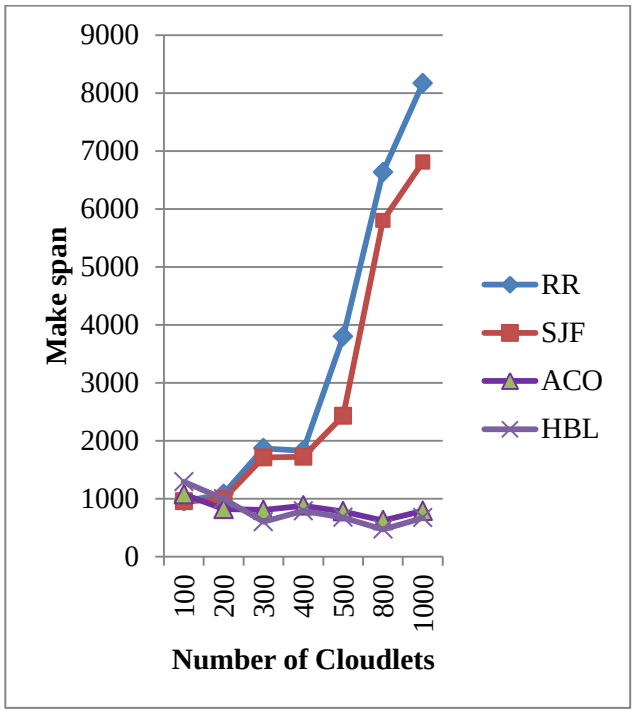

Figure 9 RR vs. SJF vs. ACO vs. HBL

As it can be seen that HBL outperforms all load balancing algorithms in terms of make span thus improving the efficiency of the system.

Comparison of our proposed Load Balancing Algorithm i.e. Modified Ant- Honey Bee Optimization Algorithm (MAHO) and basic ACO and HBL Algorithm:

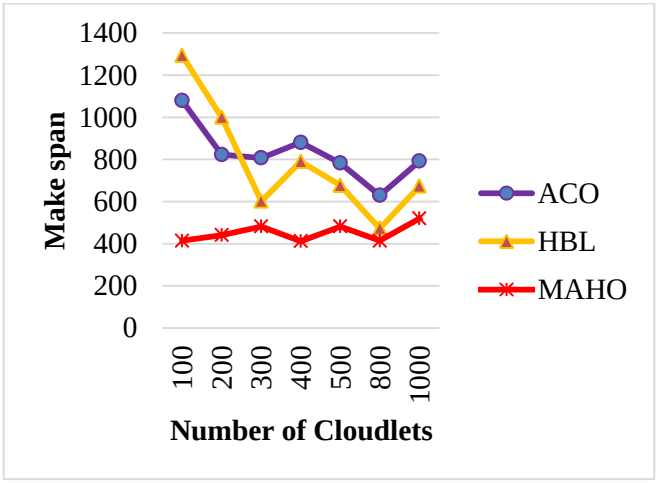

Figure 10 ACO vs. HBL vs. MAHO

MAHO as seen from the results outperforms both Honey Bee and Ant Colony optimization algorithm. This is due to the fact that MAHO chooses the optimal Virtual machine from the list of available VM for allocation of task instead of choosing a random virtual machine as in the case of Ant Colony and Honey Bee optimization Algorithm. Choosing a Virtual Machine for task allocation is a very important aspect of load balancing. As seen from the results, MAHO is well suited for varying number of tasks.

VI. CONCLUSION

Cloud computing allows wide range of users to access distributed, scalable, virtualized, software and hardware resources over the Internet. In cloud computing, load balancing is one of the important issues that are required to dynamically distribute local workload to all the available nodes in cloud. This improves the performance and utilization of resources. This paper discusses cloud load balancing, its types, components of dynamic load balancing algorithms, load balancing metrics and implements various load balancing algorithms and compares it with the proposed 'MAHO' algorithm. The proposed algorithm Modified Ant Honey bee Optimization (MAHO) which is derived from Ant Colony and Honey Bee optimization algorithm is compared with existing load balancing algorithms using CloudSim[28] and we conclude that MAHO gives better efficiency by reducing the make span of the overall system. Future works for this algorithm requires study in the field of achieving better results in terms of all other load balancing metrics.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A.Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "*Above the Clouds: A Berkeley View of Cloud Computing*", EECS Department, University of California, Berkeley, Technical Report No., UCB/EECS-**2009-28**, pages **1-23**, February **2009.**

[2] Wang, S.C., Yan, K.Q., Liao, W.P. and Wang, S.S., 2010, July. Towards a load balancing in a three-level cloud computing network. In ***2010*** *3rd international conference on computer science and information technology* (Vol. **1,** pp. **108-113**). IEEE.

[3] Nidhi Jain Kansal, Inderveer Chana, "*Cloud Computing Techniques: A Step Towards green Computing"*,IJCSI,Vol. **9**, Issue **1**, January **2012.**

[4] A. Bhadani, and S. Chaudhary, "*Performance evaluation of web servers using central load balancing policy over virtual machines on cloud*", Proceedings of the Third Annual ACM Bangalore Conference (COMPUTE), January **2010.**

[5] G. Pallis, "*Cloud Computing: The New Frontier of Internet Computing*", IEEE Journal of Internet Computing, Vol. **14**, No. **5,** September/October **2010**, pages **70-73.**

[6] J. Baliga, R. W. A. Ayre, K. Hinton, and R. S. Tucker, "*Green Cloud Computing: Balancing Energy in Processing, Storage, and Transport*", Proceedings of the IEEE, Vol. **99**, No. 1, January **2011**, pages **149-167**.

[7] R. Mata-Toledo, and P. Gupta, "*Green data center: how green can we perform*", Journal of Technology Research, Academic and Business Research Institute, Vol. **2**, No. **1**, May **2010**, pages **1-8**.

[8] Ishfaq Ahmad, Arif Ghafoor, "*Semi-Distributed Load Balancing for Massively Parallel Multicomputer Systems*", **1991.**
[9] David Escalnte and Andrew J. Korty, "*Cloud Services: Policy and Assessment*", EDUCAUSE Review, Vol. **46**, July/August **2011**.
[10] Parin. V. Patel, Hitesh. D. Patel, Pinal. J. Patel, "*A Survey on Load Balancing in Cloud Computing*" IJERT, Vol. **1,** Issue **9**, November **2012.**
[11] Ali M Alakeel, "A *Guide to Dynamic Load Balancing in Distributed Computer Systems*", International Journal of Computer Science and Network Security, Vol. **10** No. **6**, June **2010**.
[12] Ram Prasad Padhey, P. Goutam Prasad Rao, "*Load Balancing in Cloud Computing Systems*", Department of Computer Science and Engineering, National Institute of Technology, May **2011**.
[13] Abhijit A Rajguru, S.S. Apte, "*A Comparative Performance Analysis of Load Balancing Algorithms In Distributed Systems Using Qualitative Parameters*", International Journal of Recent Technology and Engineering, Vol. **1**, Issue **3**, August **2012**
[14] Rajani Sajjan," *Load Balancing and its Algorithms in Cloud Computing: A Survey*", JCSE International Journal of Computer Sciences and Engineering, Volume-**5**, January **2017.**
[15] Aarti Vig, Rajendra Singh Kushwah and Shivpratap Singh Kushwah, "*An Efficient distributed Approach for Load Balancing in Cloud Computing*", **2015** International Conference on Computational Intelligence and Communication Networks.
[16] Anurag, K., Rakesh, K., Rupesh, K. and Prashant, Y.. "*SLA driven load balancing for web applications in cloud computing environment*." **2011**
[17] Borja Sotomayor, Ruben S. Montero, Ignacio M. Llorente, and Ian Foster, "*An Open Source Solution for Virtual Infrastructure Management in Private and Hybrid Clouds*", IEEE Internet Computing, **July, 2009**.
[18] Shridhar G.Domanal and G.Ram Mohana Reddy,"*Load Balancing in Cloud Computing Using Modified Throttled Algorithm ",* **2013** IEEE International Conference on Cloud Computing in Emerging Markets (CCEM).
[19] SuchitaKhare, Asst. Prof. Shatendra Dubey, "*An Efficient Load Balancing in Public Cloud using Priority based SJF Scheduling*", International Journal of Scientific & Engineering Research, Volume 6, Issue 4, April-**2015** 1834
[20] Medhat A. Tawfeek, Ashraf E1-SiSi,Arnabi E.KeshK,Fawzy A.Torkey,"*Cloud Task Scheduling Based on Ant Colony Optimization*".978-1-4799-0080-0/13@**2013** IEEE.
[21] Nishant, K. P. Sharma, V. Krishna, C. Gupta, KP. Singh, N. Nitin and R. Rastogi, "*Load Balancing of Nodes in Cloud Using Ant Colony Optimization.*" In proc. 14th International Conference on Computer Modelling and Simulation (UKSim), IEEE, pp: **3-8**, March **2012**.
[22] Dharmesh Kashyap, Jaydeep Viradiya, "*A Survey of Various Load Balancing Algorithms In Cloud Computing*", International Journal of Scientific & Technology Research, Vol. **3,** Issue **11**, November **2014**
[23] Zhang Z. and Zhang X. "*A Load Balancing Mechanism Based on Ant Colony and Complex Network Theory in Open Cloud Computing Federation*", 2nd International Conference on Industrial Mechatronics and Automation, **240-243**, **2010**
[24] Dhinesh B. L.D , P. V. Krishna, "*Honey bee behavior inspired load balancing of tasks in cloud computing environments*", in proc. Applied Soft Computing, volume **13,** Issue **5**, May **2013**.
[25] Ganesh, Amal, M. Sandhya, and Sharmila Shankar. "*A study on fault tolerance methods in Cloud Computing*." In *Advance Computing Conference (IACC), **2014** IEEE International*, **pp. 844-849**. IEEE, **2014.**
[26] Sushil Kumar, Deepak Singh Rana and Sushil Chandra Dimri, "*Fault Tolerance and Load Balancing algorithm in Cloud Computing: A survey*", International Journal of Advanced Research in Computer and Communication Engineering, July **2015.**
[27] Pandey, Suraj, Linlin Wu, SiddeswaraMayura Guru, and Rajkumar Buyya. "*A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments*." In Advanced Information Networking and Applications (AINA), **2010** 24th IEEE International Conference on, pp. **400-407**. IEEE, **2010.**
[28] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A. F. De Rose and Rajkumar Buyya, "*CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms*", Software practice and experience, August, **2010.**
[29] Rathore, M., Rai, S. and Saluja, N.,. "*Load balancing of virtual machine using honey bee galvanizing algorithm in cloud*." *IJCSIT*, ***6***(4), pp.**4128-4132**. **2015**

AUTHORS PROFILE

Tony Bayan is an Assistant professor in the Department of BCA, K C Das Commerce College. Assam, India. He completed his MSc in Information technology from Gauhati University. His area of interest includes Cloud Computing, Big data.

Priyanka Sarma is an Assistant Professor in the Department of Computer Science and Electronics, University of Science and Technology, Meghalaya. She had done her M. Tech in Information Technology from Gauhati University. Her area of interest is Cloud Computing.

