ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

High Performance Dynamic Load Balancing With Inter-Dependent Tasks And Dependent Tasks in Cloud Computing Environments

B.Subramani*, S.Rekha

*Head, Department of Information Technology, Dr.N.G.P. Arts and Science College, Coimbatore, India
M.Phil Scholar, Department of Computer Science, Dr.N.G.P. Arts and Science College, Coimbatore, India.

Abstract

Cloud computing is an entirely internet-based approach where all the applications and files are hosted on a cloud which consists of thousands of computers interlinked together in a complex manner. A load balancing algorithm attempts to improve the response time of user's submitted applications by ensuring maximal utilization of available resources. The main objective of load balancing methods is to speed up the execution of applications on resources whose workload varies at run time in unpredictable way. Load balancing of non preemptive independent tasks on virtual machines (VMs) is an important aspect of task scheduling in clouds. Whenever certain VMs are overloaded and remaining VMs are under loaded with tasks for processing, the load has to be balanced to achieve optimal machine utilization. Previous work Propose an algorithm named honey bee behavior inspired load balancing (HBB-LB), which aims to achieve well balanced load across virtual machines for maximizing the throughput. But the major problem of this work well for independent task and it doesn't work well for dependent task. Improve the results of the existing bee colony optimization methods, in this work proposed a novel modified artificial bee colony algorithm which supports for dependent and independent task with modified artificial bee colony algorithm for load balancing tasks. The effectiveness of the proposed Modified Artificial Bee Colony for load balancing dependent and independent tasks (MABC-LBDIID) algorithms in reducing the operational cost of the cloud system is demonstrated by comparing the results with existing HBB-LB .The proposed algorithm also balances the priorities of tasks on the machines in such a way that the amount of waiting time of the tasks in the queue is minimal.

Keywords: Load balancing , dynamic load balancing ,Static load balancing ,Cloud computing, Modified Artificial Bee Colony(MABC), independent task, dependent task, Force-Directed Scheduling Approach(FDSA).

Introduction

Cloud computing is an entirely internet-based approach where all the applications and files are hosted on a cloud which consists of thousands of computers interlinked together in a complex manner. Cloud computing incorporates concepts of parallel and distributed computing to provide shared resources; hardware, software and information to computers or other devices on demand. With the development of information and communication technologies. The distributed systems are more popular as the computing demand increases. A large scale distributed systems are required with a considerable amount of servers. For efficient use of distributed system it is important to allocate tasks to each node appropriately, if the tasks are allocated randomly, it is possible some nodes gets overloaded while other becomes idle, to avoid this an efficient dynamic load balancing using random walk search on content based distributed clusters are used.

A distributed networks allows large scale resource sharing and system integration, clusters are attractive platforms for deploying applications at large scale for high performance. For the proper distribution of user requests, load balancing is required in the clustered environment.

Load Balancing is a method to distribute workload across one or more servers, network interfaces, hard drives, or other computing resources. Typical datacenter implementations rely on large, powerful (and expensive) computing hardware and network infrastructure, which are subject to the usual risks associated with any physical device, including hardware failure, power and/or network interruptions, and resource limitations in times of high demand.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

Load balancing is the process of improving the performance of the system by shifting of workload among the processors. Workload of a machine means the total processing time it requires to execute all the tasks assigned to the machine [1]. Load balancing is done so that every virtual machine in the cloud system does the same amount of work throughout therefore increasing the throughput and minimizing the response time [2]. Load balancing is one of the important factors to heighten the working performance of the cloud service provider. Balancing the load of virtual machines uniformly means that anyone of the available machine is not idle or partially loaded while others are heavily loaded. One of the crucial issue of cloud computing is to divide the workload dynamically. The benefits of distributing the workload includes increased resource utilization ratio which further leads to enhancing the overall performance thereby achieving maximum client satisfaction [3].

Load balancing methods have been categorized into two ways [4] :Static load balancing and dynamic load balancing

Static load balancing algorithms require aforementioned knowledge about the applications and resources of the system [5]. The decision of shifting the load does not depend on the current state of the system. The performance of the virtual machines is determined at the time of job arrival.
In this type of load balancing algorithms e.g., [6], the current state of the system is used to make any decision for load balancing. It allows for processes to move from an over utilized machine to an underutilized machine dynamically for faster execution. This means that it allows for process preemption which is not supported in Static load balancing approach.

In cloud computing environments, whenever a VM is heavily loaded with multiple tasks, these tasks have to be removed and submitted to the under loaded VMs of the same data center. In this case, when we remove more than one independent task from a heavy loaded VM and if there is more than one VM available to process these tasks, the tasks have to be submitted to the VM such that there will be a good mix of priorities i.e., no task should wait for a long time in order to get processed. Load balancing is done at virtual machine level i.e., at intra-data center level. The some of the task are heavily dependent on one VM, in earlier work HBB-LB only supports load balancing task for independent task only ,if it becomes the dependent task it is not supported by the system in order to solve this problem in this work presents an force-directed scheduling approach is presented that considers the online application workload and limited resource and peak power capacity for dependent task ,then suggests that load balancing in cloud computing can be achieved by modeling the foraging behavior of modified artificial bee colony algorithm . This algorithm is derived from a detailed analysis of the behavior that modified bee adopt to find and reap food. In bee hives, there is a class of bees called the scout bees which forage for food sources, upon finding one, they come back to the bee hive to advertise this using a dance called waggle/tremble/vibration dance. The display of this dance, gives the idea of the quality and/or quantity of food and also its distance from the bee hive. Forager bees then follow the Scout Bees to the location of food and then begin to reap it. They then return to the beehive and do a waggle or tremble or vibration dance to other bees in the hive giving an idea of how much food is left and hence resulting in either more exploitation or abandonment of the food source.
The specific contributions of this paper include:

- An algorithm for scheduling and load balancing of non preemptive independent and dependent tasks in cloud computing environments inspired by modified honey bee behavior
- Correlation of the proposed MABC-LBDID algorithm with actual foraging behavior of modified honey bees and force-directed scheduling approach is presented for dependent task identification for epoch time of the task.
- An analysis and systematic study with mathematical evidence to show how the modified honey bee behavior inspired load balancing can work for cloud computing environments for both dependent and independent tasks.
- Performance analysis of the proposed algorithm and an evaluation of the algorithm with respect to other existing algorithms.

Background study

H. Mahalle et al. [7] discussed this method in which jobs are divided evenly between all processors in a round robin order without considering the work load. Here the time slicing mechanism is used, which divides the time into multiple slices and each node is given a particular time slice or time interval in which they have to perform their task. Though the work load distributions between processors are equal but the job processing time for different processes are not same. Soat any point of time, some nodes may be heavily loaded and others remain idle. The main advantage of Round Robin

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

algorithm is that it does not require inter process communication. However when the jobs are of unequal processing time this algorithm suffers as the some nodes can become severely loaded while others remain idle.

A. Sidhu et al. [8] discussed load balancing algorithm which is completely based on concept of finding the appropriate virtual machines for assigning a particular job. In this the job manager is having a list of all virtual machines, using this indexed list, it allot the desire job given by client to the appropriate machine. As client requests, if the job is well suited for a particular machine on the basis of size and availability of the machine, then that job is assign to the appropriate machine. If no virtual machines are available to accept jobs then the job manager queued the request. This algorithm performs well as compared to round robin algorithm.

M. Nikita et al. [9] proposed a two level scheme for load balancing. The first level scheduling is from user application to the VM, and the second is from the VM to host resources. In this two level scheduling model, the first scheduler create the task description of virtual machine, then the second scheduler finds appropriate resources for the virtual machine in the host resource, hence overall performance is increase. The main disadvantage of this algorithm is it does not improve the response to request ratio.

In [10], A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing is presented. This paper considers three potentially viable methods for load balancing in large scale cloud systems. Firstly, a nature-inspired algorithm may be used for self-organization, achieving global load balancing via local server actions. Secondly, self-organization can be engineered based on random sampling of the system domain, giving a balanced load across all system nodes. Thirdly, the system can be restructured to optimize job assignment at the servers. Recently numerous nature inspired networking and computing models have received a lot of research attention in seeking distributed methods to address increasing scale and complexity in such systems.

The honey-bee foraging solution in [11], is investigated as a direct implementation of a natural phenomenon. Then, a distributed, biased random sampling method that maintains individual node loading near a global mean measure is examined. Finally, an algorithm for connecting simile services by local rewiring is assessed as a means of improving load balancing by active system restructuring. In case of load balancing, as the web servers demand increases or decreases, the services are assigned dynamically to regulate the changing demands of the user. The servers are grouped under virtual servers (VS), each VS having its own virtual service queues. Each server processing a request from its queue calculates a profit or reward, which is analogous to the quality that the bees show in their waggle dance.

In [12], Dynamic Load Balancing Strategy for Grid Computing is presented addressing the problem of load balancing in Grid computing. As in [13-14] this paper also proposes a load balancing model based on a tree representation of a Grid. This load balancing strategy has two main objectives: (i) Reduction of the mean response time of tasks submitted to a Grid; and, (ii) Reduction of the communication costs during task transferring. This strategy deals with three layers of algorithms (intra-site, intra-cluster and intra-grid).

Distributed system load balancing is still an active area of research in which load balancer attempts to improve the performance of a distributed system by using the processing power of the entire system to smooth out periods of high congestion at individual nodes, this is done by transferring some of the workload of heavily loaded nodes to other nodes for processing. Decisions on how to balance loads among the nodes are either static or dynamic[15].

Proposed load balancing using modified bee colony for independent and dependent task methodology

Cloud computing deals with assigning computational tasks on a dynamic resource pool of virtual machines online according to different requirements from user or the system [16]. The service requests from the clients for diverse applications can be routed at any data center to any end server in the cloud. The routing of service requests to the diverse servers is based on cloud management policies depending on load of individual servers, closeness to databases etc. The two frequently used scheduling principles in a non pre-emptive system are the First-in-First-out (FIFO) and Weighted Round Robin (WRR) policies. These policies may end up with different degrees of loads on each and every VM. This may lead to load difference between VMs computing in parallel. This creates additional problems of reduction in response time, wastage of resources and so on.

These kinds of situations leads us to give more importance to the dynamic load balancing techniques which solves the problem of load imbalance between VMs. Load Balancing techniques are effective in reducing the makespan and response time. These kinds of the problems is also important for dependent and independent task while performing the load balancing task in the cloud computing

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

,existing work HBB-LB not applicable for dependent task ,in order to overcome these problem in this work proposed an modified artificial bee colony algorithm for balancing the dependent and independent task perform load balancing for task with priority level .For that purpose first define the following constraints

Load Balancing techniques are effective in reducing the makespan and response time. First define the condition of maximum lifetime to complete task . Makespan can be defined as the overall task completion time. We denote completion time of task T_iTi on VM_i as CT_{ij}. Hence, the makespan is defined as the following function,

$$Makespan = \max\{VMT_{ij} | i \in t, i = 1, \dots n \; and \; j \in VM \; , j = 1,2, \dots , m \} \quad (1)$$

Response time is the amount of time taken between submission of a request and the first response that is produced. The reduction in waiting time is helpful in improving responsiveness of the VMs.

Let $VM = \{VM_1, VM_2, \dots VM_m\}$ be the set of m virtual machines which should process n tasks represented by the set $= \{T_1, \dots \dots , T_n\}$. All the machines are unrelated and parallel and are denoted as R in the model. Non-preemptive tasks are denoted as npmtn. Non preemption of a task means that processing of that task on a virtual machine cannot be interrupted (assuming that failure does not occur).

Denote finishing time of a task T_i by VMT_i. Our aim is to reduce the makespan which can be denoted as CT_{max}. So our model is $R|npmtn|VMT_{max}$. Processing time of a task T_i on virtual machine VM_j can be denoted as P_{ij}. Processing time of all tasks in a VM_j can be defined by Eq. (2).

$$P_j = \sum_{i=1}^{n} P_{ij} \; , j = 1, \dots m \quad (2)$$

By minimizing VMT_{max}, we get Eq. (3). From Eq. (2) and (3) can imply Eq

$$\sum_{i=1} P_{ij} \leq VMT_{max} \; , j = 1, .. m \quad (3)$$

$$VMT_{max} = \left\{ \max_{i=1} n \; VMT_i \; \max_{j=1} n_i \sum_{i=1}^{n} P_{ij} \right. \quad (4)$$

The thershold function of the CPU utilization load is defined as below

$$VMTS(vmu_t) = t_o(vmu_t)/T_i \quad (5)$$

vmu_t is the VM utilization threshold distinguishing the non-overload and overload states of the host; to is the time, during which the host has been overloaded, which is a function of u_t; and ta is the total time, during which the host has been active. The following conditions need to added to measure the job task when is exceed overload condition,

$$T_j(vmt_m, vmu_t) \rightarrow max \quad (6)$$

$$\frac{t_0(vmt_m, vmu_t)}{t_j(vmt_m, vmu_t)} \leq M \quad (7)$$

where vmt_m is the time when a VM migration has been initiated; u_t is the CPU utilization threshold defining the overload state of the host; $t_0(vmt_m, vmu_t)$is the time, during which the host has been overloaded, which is a function of vmt_m and u_t; ta is the total time, during which the host has been active, which is also a function of vmt_m and vmu_t; and M is the limit on the maximum allowed $VMTS(u_t)$ value. It is necessary to maximize the mean time between VM migrations initiated by the host overload detection algorithm, which can be achieved by maximizing each individual inter-migration time interval. Therefore, we limit the problem formulation to a single VM migration, i.e., the time span of a problem instance is from the end of a previous VM migration and to the end of the next.
The capacity of

$$C_j = pe_{numj} \times pe_{mipsj} + vm_{bwj} + VMTS(VMU_t) \quad (8)$$

where processing element, $pe_{numj}$is the number processors in VM_j, pe_{mipsj} is million instructions per second of all processors in VM_j and vm_{bwj} is the communication bandwidth ability of VM_j,

$$C = \sum_{i=1}^{m} C_i \quad (9)$$

Total length of tasks that are assigned to a VM is called load.

$$L_{vM_i, t = \frac{N(T,t)}{S(VM_i,t)}} \quad (10)$$

Load of a VM can be calculated as the Number of tasks at time t onservice queue of VM_i divided by the service rate of VM_i at time t. To estimate the results of overloaded detection results for VM proposed a Semi hidden markov models with $VM = \{VM_1, VM_2, \dots VM_m\}$ be the set of m virtual

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

machines which should process n tasks represented by the set $T = \{T_1, \dots\dots, T_n\}$ with $S(VMU_t)$.

$$L = \sum_{i=1} L_{vM_i} \quad (11)$$

Processing time of a VM

$$L = \sum_{i=1} L_{vM_i} \quad (12)$$

Processing time of all VMs

$$PT = \sum_{i=1} L/C \quad (13)$$

Standard deviation

$$\sigma = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(PT_i - PT)^2} \quad (14)$$

In order to support dependent task for load balancing ,Starting from the initial solution, instances associated with each task needs to be merged to reduce the number of instances related to each task to one ( ). Merging task instances changes the force in the datacenter instances hosting the task instances and may change the placement (scheduling) of the instances related to the dependent tasks (parent and child tasks) due to scheduling constraints. Based on the FDS terms, the force change (future force minus current force) generated from dependent task instance movements is called dependent-force and the force change directly related to the task movement is called self-force. Task instance merging force (change) is defined as the summation of self-force and dependent-force. Given any task, the task instance merging with minimum force is executed to reduce the total force in the system.

Moreover, to determine the order of task instance merging execution, tasks are sorted (non-decreasing) based on their minimum task instance merging force. Task instance merging is performed in multiple stages in order to decrease the complexity of selecting the new host for task instances and avoid drastic changes in the assignment solution. For this reason, first the number of task instances in each epoch is gradually reduced to one (only self-force) and then number of epochs for each task is reduced one by one (self-force plus dependent-force). At the end of the second stage, the schedule of each task is determined. The pseudo-code for task instance merging having at most one task instance per epoch per task is shown in Algorithm 1. In this pseudo-code, denotes the force related to task instance in epoch having.

Algorithm 1:FLBS algorithm for task merging

Input :Task instance placed in static manner on the datacenter and server types (at most one task instance for each period of the task)

Output :One instance solution for each task to determine scheduling solution

For each (Job J)

$T_{ij}^{d} = infinity$

For each $(t \in T_j)$

Let T_S and T_e denotes the starting and the ending time of the task for present task

Total number of the task instance

$N = T_e - T_s + 1$

If $(N == 1)$ continue

$F_t^{r,self} = \sum_{T_s+1}^{T_e} F_t^{N-1} - \sum_{T_s+1}^{T_e} F_t^{N}$ // self force shift to right

$F_t^{r,dep} = \sum_{t' \in C_{j,t}} (F_t^{r,self} + F_t^{r,dep})$ //dependent – force shift to right

$F_t^{r} = F_t^{r,self} + F_t^{r,dep}$ //dependent –force shift to right

$F_t^{l,self} = \sum_{T_s+1}^{T_s-1} F_t^{N-1} - \sum_{T_s}^{T_e} F_t^{N}$ // self force shift to right

$F_t^{l,dep} = \sum_{t' \in C_{j,t}} (F_t^{r,self} + F_t^{r,dep})$ //dependent – force shift to left

$F_t^{l} = F_t^{r,self} + F_t^{r,dep}$

$F_j^{min} = \min(F_j^{min}, F_t^{r}, F_t^{l})$

End

End

ABC has been successfully used for load balancing problems [17] as it is easy to develop and solve many optimization problems with only a few controls of parameters [18]. ABC suggests the intellectual searching behavior of a honey bee swarm. In existing work the basic version of the ABC algorithm for load balancing with overload, underload and balanced is performed for independent task. The basic version of the Artificial Bee Colony algorithm has only one control parameter ‘‘limit” apart from the common control parameters of the population-based algorithms such as population size or colony size (SN) and maximum generation number or maximum cycle number (MCN).

But the major problem of this existing ABC for load balancing, the convergence rate of the algorithm is poorer when working with constrained problems, composite functions and some non-separable functions. In order to improve the speed of the load balancing for feature dependent and independent task. In this work proposes a modified

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

artificial bee colony optimization algorithm (MABC) for load balancing task. There are three general modifications are done in usual ABC algorithm There are introducing the best-so-far solution, inertia weight and acceleration coefficients to modify the search process. So it improves the load balancing speed, the modification forms of the employed bees and the onlooker ones are different in the second acceleration coefficient. The modified ABC algorithm for load balancing for dependent and independent task is called as MABC. Multiple The operation process for load balancing can be modified in the following form,

$$V_{ij} = X_{ij}W_{ij} + 2(\phi_{ij} - 0,.5)(X_{ij} - X_{kj})\Phi_1 + \varphi_{ij}(X_j - X_{kj})\Phi_2 \quad (18)$$

Where V_{ij} is the new optimal load balancing results for dependent and independent task . $X_{ij}W_{ij}$ is the inertia weight which controls impacts of the load balancing solution . $X_{ij}.X_j$ is the j[th] parameter of the best optimal load balanced results solution so-far, ϕ_{ij} and φ_{ij} are random numbers between [0, 1], Φ_1 and Φ_2 are positive parameters that could control the maximum step size of bees for load balancing of static and dynamic tasks. However, if the global fitness of the each task is very large, bees are far away from the best optimal load balanced results. Conversely, if the global fitness values of load balancing are small only a small modification needed for global optimal load balancing results. In this investigation, the inertia weight and acceleration coefficients are defined as functions of the fitness in the search process of ABC. They are proposed as follows:

$$W_{ij} = \Phi_1 = \frac{1}{\left(1 + \exp\left(-\frac{fitness(i)}{ap}\right)\right)} \quad (19)$$

$$\Phi_2 = 1\ ,if\ a\ bee\ is\ employeed\ bee \quad (20)$$

$$\Phi_2 = \frac{1}{\left(1 + \exp\left(-\frac{fitness(i)}{ap}\right)\right)}\ if\ a\ bee\ is\ onlooker \quad (21)$$

Where ap is the Fitness value of the tasks in the first iteration. In order to further balance the process of the examination of load balancing results and the utilization, the modification forms of the employed bees and the onlooker ones are different in the acceleration coefficient Φ_2.The fitness of each tasks is assigned randomly depends on the task completion time and the priority level of the tasks . The individual fitness condition for each task is calculated as follows :

$$fit_i = \frac{1}{1 + fit_i} \quad (22)$$

An artificial onlooker bee selects best load balanced VM rely on the probability value associated with that task p_i, calculated by the following expression**,**

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \quad (23)$$

wherefit_irepresents the fitness value of the tasks for each VM i in the location and SN is the number of task that run on the VM ,which is equivalent to the number of employed bees.

Alternative process of the velocity update

Once the optimal load balanced results are found for current population samples in ABC algorithm ,the velocity values of the bees need to update to perform next optimal load balancing results .The velocity values of the bees are updated using global best guided bees for load balancing of static and dynamic tasks ,it is called as Gbest-guided ABC (GABC). The solution search equation of GABC is given by the following form,

$$V_{ij} = X_{ij} + 2(\phi_{ij} - 0.5)(X_{ij} - X_{kj}) + \varphi_{ij}(X_j - X_{kj}) \quad (24)$$

Where V_{ij} is the new load balanced results that is modified depending on its previous load balancing results $X_{ij}.X_j$ is the j[th] parameter of the best load balanced results - so-far solution, ϕ_{ij} is a random number between $[0,1]$, $\phi_{ij}[0,c]$, c is a nonnegative constant, which is set to 1 .In MABC algorithm; there are three different solution search equations. The first one is general velocity equation is modified, which is solution modification form of the original ABC algorithm for load balancing. Second one is solution search equation of GABC as given in Equation (24). In MABC, these load balancing results operations are replicated by producing a new VM for each task position randomly based on the balanced VM results and changing it with the discarded one. In MABC, if a current task for each VM position doesn't improve the load balancing result within a pre-specified number of iterations (MCN), and then the current load balanced task position is assumed to be neglected.

$$\chi_i^j = \chi_{min}^j + rand(0,1)(\chi_{max}^j - \chi_{min}^j) \quad (25)$$

In initialization, MABC like ABC starts by associating all employed bees (Multiple static features) with randomly generated food sources for load balanced task for each VM results. After initialization of task population the food sources of the load balanced subject to repeated cycles for load balanced tasks for VM through employed bees, onlooker bees and scout bees. MABC, an employed bee firstly finds best optimal Load balanced VM

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

results based on the three modification mentioned above ,then chooses and determines the load balanced results as best load balanced candidate solution. When all employed bees in the population have finished load balanced process, they share the task information for each VM fitness information to the next stage of the bee called as onlookers, here each of the bee balance the VM based on the probability value given in Equation (23). Providing that the fitness value in equation (22) is better than that of the previous load balanced population fitness values for each VM, the bee would memorize the new load balanced VM position and forget the previous load balanced results and kept current multiple static feature selection results as best so for . Then, feature selection samples position $v_{ij}$is estimated then its performance is compared with that each one of the previous load balanced VM results. If the new load balanced VM result is better than old balanced VM results for each task, it is replaced with the old balanced VM results in the memory. Or else, an old Balanced VM results is kept as same. In other words, a greedy selection system works for the load balancing between new founded load balancing task and the old load balancing results

Modified Artificial Bee Colony (MABC) Optimization algorithm 2

1. Initialize the population of solutions $x_i, i = 1, \ldots \ldots \ldots SN$,each population as a number of task
2. Evaluate the population with tasks
3. Set cycle = 1
4. Repeat
5. Produce new load balancing solution $V_{ij}$for the employed bees (features) by using (1) and evaluate them best multiple static feature and acceleration ,weight parameters
6. Calculate the W, Φ_1, Φ_2 using equation (19) ,(20) and (21)
7. Calculate fitness value to each number of the tasks by using equation (22).
8. Apply the greedy selection process for the employed bees
9. Calculate the probability values P_i for each task $x_i$by (23)
10. Produce the new load balanced solutions V_{ij} in equation (24) for the onlookers from the solutions X_i selected depending on P_i and evaluate them
11. Apply the greedy selection process for the onlookers
12. Determine the abandoned load balanced solutions for the scout, if exists, and replace it with a new randomly produced solution $\chi_i^j$by (25)
13. Memorize the best solution achieved so far
14. cycle = cycle + 1
15. until cycle = MCN

After finding the workload and standard deviation, the system should decide whether to do load balancing or not,the dependent task F_j^{min} . For this, there are two possible situations i.e., (1) Finding whether the system is balanced (2) Finding whether the whole system is saturated or not (The whole group is overloaded or not). If overloaded, load balancing is meaningless.

Finding State of the VM group

1. If the standard deviation of the VM load (σ) is under or equal to the threshold condition set (Ts) [0–1] then the system is balanced [13].
2. Otherwise system is in an imbalance state. It may be overloaded or under loaded.
3. If $\sigma \leq T_s, F_j^{min}$
4. System is balanced
5. Exit
6. Finding Overloaded Group

When the current workload of VM group exceeds the maximum capacity of the group, then the group is overloaded. Load balancing is not possible in this case.

1. If $L > maximum\ capacity, S > F_j^{min}$
2. Load balancing is not possible
3. Else
4. Trigger load balancing.

The virtual machines will be grouped based on their loads. The groups are Overloaded VMs, under loaded VMs

and balanced VMs. Each set contains the number of VMs. Task removed from one of overloaded VM set has to a make decision to get placed in one of several low loaded VMs based on the load and tasks available in the under loaded VM. In our technique, this task is considered as a honey bee and low loaded VMs are considered as the destination of the honey bees. The information the bees (tasks) update are load on a VM, load on all VMs, number of tasks in each VM, the number of VMs in each VM group (under loaded VM, over loaded VM, etc.,) and task priorities in each VM. Load balanced VMs are not used in switching of tasks. Once the task switching is over, the balanced VMs are included into the load balanced VM set. Once this set has all the VMs, the load balancing is successful i.e., all tasks are balanced.

VM Selection of different prioritized tasks

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

$$T_h \rightarrow VM_d \left|\min\left(\sum T_h\right)\right. \in Vm_d, VM_d \left|\min\left(\sum F_t^{r,dep}\right)\right. \quad (26)$$

$$T_m \rightarrow VM_d \left|\min\left(\sum T_h + \sum T_m\right)\right. \in Vm_d, VM_d \left|\min\left(\sum F_t^{r,self}\right)\right. \quad (27)$$

$$T_l \rightarrow VM_d \left|\min\left(\sum T\right)\right. \in Vm_d, VM_d \left|\min\left(\sum F_j^{min}\right)\right. \quad (28)$$

where T_h, T_m, T_l are the tasks of high, middle and low priority cadres respectively. The priorities of tasks can be categorized in 3 cadres (high, middle, and low). When a high priority task has to be submitted to one of the under loaded machines, it has to consider the high priority tasks already submitted to that machine. This will ensure that the high priority task will find the machine which has less number of high priority tasks.

Experimentation results

A cloud computing system has to handle several hurdles like network flow, load balancing on virtual machines, federation of clouds, scalability and trust management and so on. Research in cloud computing generally focus on these issues with varying importance. Clouds offer a set of services (software and hardware) on an unprecedented scale. Cloud Services have to handle the temporal variation in demand through dynamic provisioning or deprovisioning from clouds. Considering all these, we cannot directly use the cloud computing system. Experimenting new techniques or strategies in real cloud computing operations is not practically possible as such experiments will compromise the end users QoS requirements like security, cost, and speed. CloudSim [19-20] simulator is a generalized simulation framework that allows modeling, simulation and experimenting the cloud computing infrastructure and application services [20]. In this section, have analyzed the performance of our algorithm based on the results of simulation done using CloudSim. Then extended the classes of CloudSim simulator to simulate our algorithm.

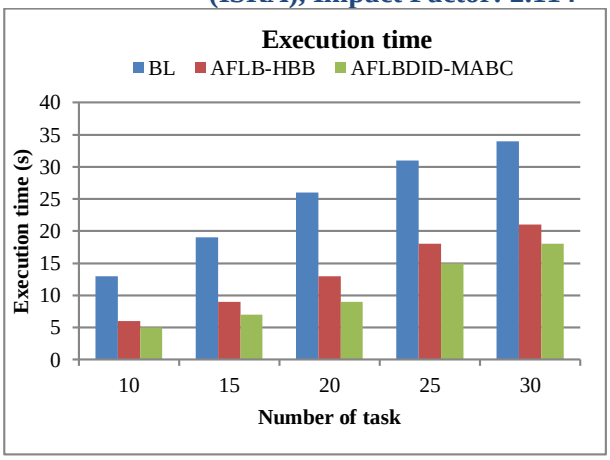

Figure 1: Comparison of makespan before and after load balancing using HBB and MABC-LBDID.

Figure. 1 illustrate the comparison of Makespan before and after Load balancing using HBB-LB, MABC-LBDID. The X-axis represents the number of tasks and the Y-axis represents the Makespan (task execution and completion time) in seconds, it shows that proposed MABC-LBDID are less execution time than the HBB-LB

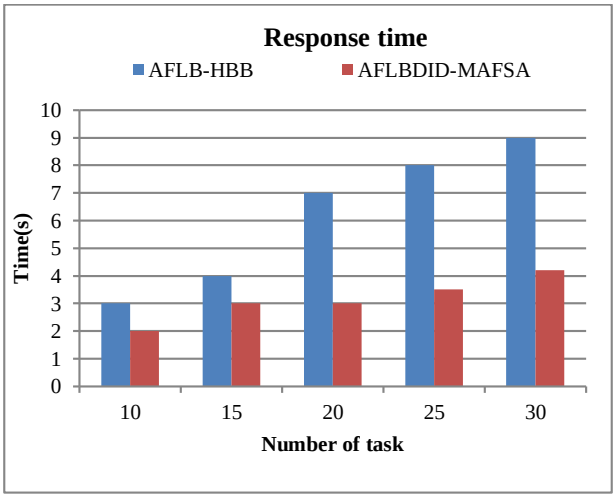

Figure 2: Response time of VMs in seconds for MABC-LBDID and HBB-LB

Figure 2 illustrates the response time of VMs in seconds for MABC-LBDID and HBB-LB Algorithms. The X-axis represents number of tasks and the Y-axis represents time in seconds. It is evident that MBAC-LBDID is more efficient compared with other HBB-LB.

$$DI = \frac{T_{max} - T_{min}}{T_{avg}} \quad (29)$$

Where T_{max} &T_{min} are the maximum and minimum $T_i$among all VMs, T_{avg} is the average T_i of

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

VMs. Our load balancing system reduces the degree of imbalance drastically.

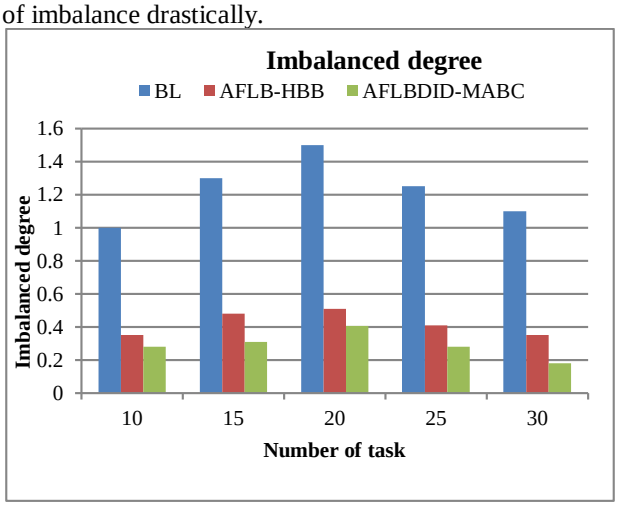

Figure 3: Degree of imbalance between VMs before and after

Figure.3 shows the degree of imbalance between VMs before and after load balancing with HBB-LB and MABC-LBDID . The X-axis represents number of tasks and the Y-axis represents the degree of imbalance. It is clearly evident that after load balancing with HBB-LB and proposed MABC-LBDID, the degree of imbalance is greatly reduced.

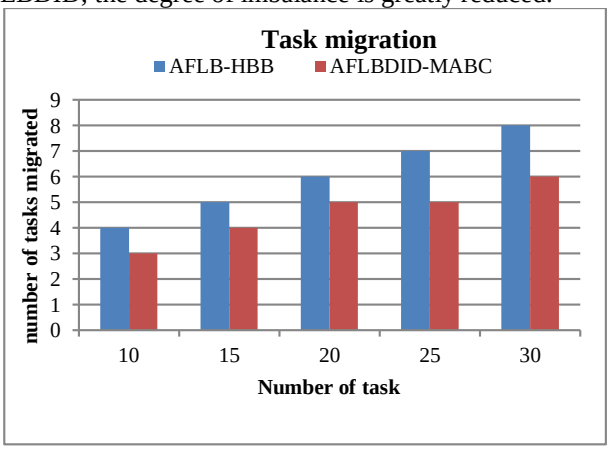

Figure 4: Comparison of number of task migrations

Figure 4 shows the comparison of degree of imbalance between HBB-LB and MABC-LBDID. The X-axis represents number of tasks and the Y-axis represents the degree of imbalance. MABC-LBDID is more efficient and has a lesser degree of imbalance when compared with other three algorithms.

Conclusion

In this paper, we have proposed a load balancing technique for cloud computing environments based on behavior of modified artificial bee colony algorithm. This algorithm not only balances the load, but also takes into consideration the priorities of tasks for both dependent and independent tasks that have been removed from heavily loaded Virtual Machines. The tasks removed from these VMs are treated as bees, which are the information updaters globally. This algorithm also considers the priorities of the tasks. The proposed MABC-LBDID tasks supports both static and dynamic load balancing improves the overall throughput of processing and priority based balancing focuses on reducing the amount of time a task has to wait on a queue of the VM. Thus, it reduces the response of time of VMs. Experimental results were compared with existing honey bee behavior load balancing algorithm. Results show that our algorithm stands good without increasing additional overheads. This load balancing technique works well for heterogeneous cloud computing systems and is for balancing non preemptive independent tasks In present work improves QOS result by considering priority only, other types of QOS factors also important to improve QOS result. In future, plan to improve this algorithm by considering other QoS factors also.

References

[1] *Sandeep Sharma, Sarabjeet Singh, Meenaksshi Sharma, "Performance Analysis of Load Balancing Algorithms", World Academy of Science, Engineering and Technology, 2008.*

[2] *Hisao Kameda, EL-Zoghdy Said Fathyy and Inhwan Ryuz Jie Lix, "A Performance Comparison of Dynamic vs Static Load Balancing Policies in a Mainframe, Personal Computer Network Model", Proceedings Of The 39th IEEE Conference on Decision & Control, 2000.*

[3] *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.*

[4] *U.Chatterjee, "A Study on Efficient Load Balancing Algorithms in Cloud computing Environment",International Journal of Current Engineering and Technology, Vol.3, 11 November 2013*

[5] *M. Houle, A. Symnovis, D. Wood, Dimension-exchange algorithms for load balancing on trees, in: Proc. of 9th Int. Colloquium on Structural Information and Communication Complexity, Andros, Greece, June, 2002, pp. 181–196.*

[6] *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.*

[7] *A.khiyait, H.Bakkali, M.Zbakh, D.Kettani, Load Balancing Cloud Computing: state of art", UniversityMohammed V Souissi Rabat Morocco, 2012.*

[8] *H.Mahalle, P.Kaveri, V.Chavan, "Load Balancing on Cloud Data Centers", international Journal of Advance Research in computer Science and Software Engineering, vol. 3, Jan. 2013.*

[9] *A.Sidhu, S.Kinger, "Analysis of Load Balancing techniques in Cloud Computing", Council for innovativeresearch international Journal of Computer & Technology, vol.4, March-April 2013.*

[10] *M. Randles, D. Lamb, A. Taleb-Bendiab, A comparative study into distributed load balancing algorithms for cloud computing, in: Proceedings of 24th IEEE International Conference on Advanced Information Networking and Applications Workshops, Perth, Australia, April, 2010, pp. 551–556*

[11] *M. Randles, A. Taleb-Bendiab, D. Lamb, Scalable self governance using service communities as ambients, in: Proceedings of the IEEE Workshop on Software and Services Maintenance and Management (SSMM 2009) within the 4th IEEE Congress on Services, IEEE SERVICES-I 2009, July 6–10, Los Angeles, CA (to appear), 2009.*

[12] *B. Yagoubi, Y. Slimani, Dynamic load balancing strategy for grid computing, transactions on engineering, Computing and Technology 13 (May) (2006) 260–265.*

[13] *B. Yagoubi, Y. Slimani, Task load balancing strategy for grid computing, Journal of Computer Science 3 (3) (2007) 186–194.*

[14] *B. Yagoubi, M. Medebber, A load balancing model for grid environment,computer and information sciences, 2007. iscis 2007, in: 22nd International Symposium on, 7–9 Nov, 2007, pp. 1–7.*

[15] *Ali, A.D., 2001. A dynamic cluster constructor for load balancing in big heterogeneous distributedsystems. Proceeding of the Symposium on Performance Evaluation of Computer and Telecommunication Systems, July 15-19, Orlando, Florida, USA., pp: 47-55.*

[16] *N.Suguna and K.G.Thanushkodi, "An Independent Rough Set Approach Hybrid with Artificial Bee Colony Algorithm for Dimensionality Reduction", American Journal of Applied Sciences 8 (3): 261 – 266, 2011*

[17] *Li Bao and Jian-chao Zeng, Comparison and Analysis of the Selection Mechanism in the Artificial Bee Colony Algorithm, Proc. IEEE Ninth International Conference on Hybrid Intelligent Systems, 2009, 411-416.*

[18] *R.F. de Mello, L.J. Senger, L.T. Yang, A routing load balancing policy for grid computing environments, in: 20th International Conference on, vol. 1, 18–20 April, Advanced Information Networking and Applications, 2006. AINA 2006. (2006) 6.*

[19] *R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F. De Rose, R. Buyya, CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Software: Practice and Experience 41 (2011) 23–50, http://dx.doi.org/10.1002/spe.995.*

[20] *R.N. Calheiros, R. Ranjan, C.A.F.D. Rose, R. Buyya, CloudSim: a novel framework for modeling and simulation of cloud computing infrastructures and services, Computing Research Repository, vol. abs/0903.2525, 2009.*

Author Biography

S Rekha
Done her Post graduation in G.R.Damodaran College of science and currently pursuing master of philosophy and also working as a Assistant professor in Dr. N.G.P. arts and science college, Coimbatore. She has more than 4 years of teaching experience and her research interest in cloud computing and Networking.
Email:
sampathrekha@yahoo.com

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

B Subramani

He has around 20 years of postgraduate teaching experience in Computer Science.His Research interest includes Advanced networking, Mobile computing, and cloud comuting. He is currently pursuing his Ph.D in Computer Science in Bharathiar university, Coimbatore. He is a Life member in C S I : Computer Society of India , board of studies(bharathiar university), Commission Member Course Approvals (Bharathiar university), BoS - Madurai kamaraj University, PG BoS-Ayya Nadar Janaki Ammal College, Sivakasi, UG Bos-Kongu Nadu Arts and Science College. He has been keenly involved in organizing training programmes for students and faculty members and also he guides number of research scholars in Computer Science. Currently he is Heading the Department of information technology, Dr. N.g.p. arts and science college, Coimbatore.

Email:
subramaningp@gmail.com