IJCSN International Journal of Computer Science and Network, Volume 4, Issue 5, October 2015
ISSN (Online) : 2277-5420 www.IJCSN.org
Impact Factor: 0.417

A New Modified HBB Optimized Load Balancing in Cloud Computing

[1] Mohd Hamza, [2] Satish Pawar , [3] Yogendra Kumar Jain

[1,2,3] Dept. of Computer Science and Engineering, Samrat Ashok Technological Institute
Vidisha, (MP) , INDIA

Abstract - Cloud computing is solely network dependent technology which completely rely on internet. The load balancing is one of the vital parameter for judging the efficiency of cloud computing. It continuously perform check on the VMs and cloudlets, so that none of the VMs get overloaded, while some machines still have space to allocate the cloudlet. In such cases load balancing is done to use resources in efficient manner. Here, we proposed a new approach called modified honey bee behavior approach which is based on HBB-LB. In this modified honey bee algorithm, we keep account of cloudlet size, VM capacity and priority. Knowledge of these entities of cloud environment is very important to allocate cloudlets to VM. The utilization of VM resources should be maximum and cloudlet will be allocated to that VM which is best fit by principle of optimality (on basis of memory allocation) with minimum number of high priority task. Modified HBB-LB keeps track of priorities of cloudlets, so high priority cloudlet should not wait in queue as compared to other cloudlets. We have compared modified honey bee approach with some previous load balancing approaches and got minimum average response time, makespan time and degree of imbalance.

Keywords - *Cloud Computing, HBB-LB, Load Balancing.*

1. Introduction

Cloud computing is newly growing technology which is completely internet based and were different systems are connected with each other in a complex way to share the computing resources, on the methodology of pay and use. The main reason behind this methodology is to diminish the cost and get efficient performance. There are three service model in cloud computing IaaS, PaaS, SaaS. In IaaS (Infrastructure as a service) whole infrastructure is placed in datacenter of cloud, with growing technology trends the organization infrastructure is getting transformed in private cloud. This trend of transforming the infrastructure in cloud helps the organization to diminish the cost and provide flexibility to the consumer to add and remove their task during fluctuation in busy hours. In IaaS environment, resources are assigned at cloud level and infrastructure level. In cloud level the infrastructure is created by VM (Virtual Machine) scheduler. At VM level, job scheduler techniques are used to assign the job to a particular VM at the best possible manner according to techniques to get efficient response. SaaS is an software delivery model that provide access to the software and it function remotely as a web based services. SaaS allows organization to access business functionality at low cost. PaaS is a model which delivers application over the internet. In PaaS model, cloud provider deliver network and software tools, especially those tools which are needed for application development to its user as a service.

The main objective of load balancing methods is to speed up the execution of task on VM whose workload varies at run time in unpredictable manner [1]. Task of processing efficiently is completely depend upon the virtual machines (VM). In cloud computing environment, many virtual machines can be connected with a single host and all VM can executes at the same time, so the resources connected will be utilized in such a way to get best possible result. So, when the virtual machine are executing simultaneously the chaos is created in scheduling of task. In a cloud computing environment, the task is given to virtual machine by the host to which VM is connected and one thing which is to make sure that the task aren't allotted to any particular machine and other machines remain in the free state. Scheduler should ensure that the all the VM's will be allocated with the task to get efficient results. Load balancing is done to get efficient result, there are techniques for load balancing which are depend on the scenario on which balancing is to be done are static and dynamic. Static algorithm work on were

fluctuation in load is not very considerable, we can say this type of algorithm are best suited for homogeneous environment. Dynamic algorithm is used were load is varying randomly in unexpected manner it is best suited for heterogeneous environment.

2. Related Work

Load balancing algorithms can be classified in static and dynamic load balancing. In static load balancing algorithm the decisions related to balancing of load is made at compile time when resource requirements are estimated. Static algorithms work very efficiently as compared to dynamic algorithm when there is a minimum variation in the load for the VMs. Therefore, these algorithms are not well suited for grid and cloud computing environments where the load kept varying. Dynamic load balancing algorithms make changes to the distribution of work load among nodes at run-time. Dynamic algorithm use current load information when making distribution decisions [2]. Fiona et al. says that balancing of load is one of major issues in distributed and parallel system for efficient use of system resources [3]. Load balancing has to be performed when all system are not performing same amount of processing at a particular time. Load balancing is job scheduling process which takes job as whole and sends it to computing node.

Mello et al. proposed a Routing Load Balancing Policy for Grid Computing environments [4]. It uses routing concepts from computer networks to define a neighborhood and search the adequate computers to divide applications workload. This algorithm is designed to equally distribute the workload of tasks of parallel applications over Grid computing environments. Route algorithm is applicable for environments where there are several heterogeneous computers and parallel applications are composed of multiple tasks. When dealing with large scale systems, a minimization of the total execution time is not the only objective of a load balancing strategy. The cost of communication and load redistribution are also some of the important issues.

James et al. proposed load balancing algorithm [5], it use the concept of analyzing virtual machines and according to their capacity a particular number is assigned to it and there is active weight load balancer which keeps account on VM's weight and number of task to be processed under the datacenter. The balancer check for the least loaded VM by parsing the table. It allocates cloudlets to the least loaded VM according to the weight assigned. If there are many least loaded VMs then the first identified is selected. This technique enforces us very well to know the response time of cloudlets.

Houle et al. consider algorithms for static load balancing on trees treating that the total load is a fixed [6]. This approach is considered token-distribution on trees is important since an algorithm for token-distribution on trees can be applied to a spanning tree of an arbitrary network. Dimension exchange algorithm which is used in balancing of tree produces a distribution, each node has knowledge of the number of nodes in the tree. Hu et al. design an optimal data migration algorithm in diffusive dynamic load balancing through the calculation of Lagrange multiplier of the Euclidean form of transferred weight [7]. This work can effectively minimize the data movement in homogenous environments, but it does not consider heterogeneous environments. Genaud et al. states enhanced MPI Scatter primitive to support master-slave load balancing, taking into consideration the optimization of computation and data distribution using a linear programming algorithm [8]. However, this solution is limited to static load balancing. M. Moradi et al. formed a New Time Optimizing Probabilistic Load Balancing Algorithm Grid Computing is presented [9]. This algorithm chooses the resources based on better past status and least completion time. The main purpose of this algorithm is to establish load balancing and reduce the response time.

Ant Colony Optimization (ACO) algorithms for job scheduling arose from the way real ants behave in nature. Real ants initially wander randomly, and when they find food they return to their colony while laying down pheromone trails. If remaining ants find a path which is accessed by earlier ants then they will not travel at random, but will follow the trail instead, and while returning they reinforcing the path for other. If ant find food or a shorter path to a food source then other ants will be going to follow that path, and the positive feedback ensure that all the ants following a same path. There might be two possible paths from the nest to the food source, might be one of them is longer than the other one. Ants will start their journey randomly in beginning to explore and choose one from the available paths. The ants who begin to move on the shorter path will naturally reach the food source before the others ants, and while doing this hunt former group of ants will leave trail for the future coming ants. After finding the food, the ants will return and will try to find the home. Moreover, the ants that perform the round trip faster, strengthen more quickly [10]. Dinesh babu et.al have work on HBB-LB (honey bee behavior load balancing) which aims to achieve well balanced load across virtual machines for

maximizing the throughput [11]. The algorithm also balances the priorities of tasks on the machines in such a way that the waiting time of the tasks in the queue is minimal. The tasks removed from overloaded VMs act as honey bees, upon submission to the under loaded VM, it will update the number of various priority tasks and load of tasks assigned to that VM. This information will be helpful for other tasks i.e., when a task having greater priority is submitted to VMs, it should consider the VM that has minimum number of high priority tasks so that the particular task will be executed earlier. Since VMs are arranged in increasing way, the jobs which are detached from overloaded VMs will be submitted to under loaded VMs. Current workload of all available VMs can be calculated based on the information received from the datacenter. Based on this, standard deviation has to be calculated to measure deviations of load on virtual machine. After finding the workload and standard deviation, the data center broker who makes the policies for load balancing will decide to do load balancing or not.

Need aroused for modified honey bee algorithm is that, there are some less efficient processing way in honey bee algorithm, the desired task by the user can be placed to any data center, users have no knowledge about it where the request is going to be executed. In honey bee algorithms, scheduling policy is based on first come first serve method and because of this first come first serve scheduling policy the processing resources get wasted many time. It also makes imbalance of load between the virtual machines, to get more efficiency for maximum utilization use of resources. We used modified HBB-LB which keep account of space lagging or processing capacity left in machine, number of task and there priority which are assigned to VM. Modified HBB LB allocate the task to VM in which there is minimum wastage of resources and least time to processed the task, to get better response time and more efficient utilization of datacenter resources.

3. Honey Bee Behavior Algorithm

There are various kind of techniques for balancing the load for different task in cloud environment, so in order to solve these problem Honey bee inspired load balancing is proposed. In load balancing, the web servers demand kept fluctuating, the services in cloud computing environment are allotted dynamically, because of the changing demands of the user. In the data centre, the servers are known as Virtual Machines (VM), each VM having its own virtual service queues. Each server processing a request from its queue, which is analogous to the quality that the bee shows [11].

Proposed algorithm is derived from a detailed analysis of the behaviour that honey bees do, to hunt and reap food. In the bee hives, there is a certain kind of categories among bees, one of them is called scout bees. Scott bee forage for food sources, when they find food they return to their beehive to advertise this using a dance called waggle /tremble /vibration dance. The display of this dance, gives the idea of the quality and quantity of food and its distance from the beehive. 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 dance to other bees in the hive. It gives an idea of how much food is left and hence resulting in either more exploitation or abandonment of the food source.

In the same manner, the removed tasks from over loaded VMs are considered as the honey bees. Upon submission to the under loaded VM, the task will update the number of various priority tasks and load of that particular VM to all other waiting tasks. This will be helpful for other tasks in choosing their virtual machine based on load and the tasks priorities. Whenever, a high priority task is submitted to the under loaded VM. It should ensure that the VM should have least number of high priority tasks, so that the particular task will be executed at the earliest possible. Since all under loaded VMs are sorted in ascending order based on load, the task removed from the over loaded machine which are sorted in descending order is submitted to under loaded virtual machine. In essence, the tasks are the honey bees and the VMs are the food sources. Loading of a task to a VM is similar to a honey bee foraging a food source (a flower or a patch of flowers). When a VM is overloaded i.e., similar to the honey getting depleted at a food source, the cloudlets moving to an under loaded VM is similar to a foraging bee finding a new food source. This removed task updates the remaining tasks about the VM status, similar to the waggle/tremble/vibration dance performed by the honey bees to inform other honey bees. This task will update the status of the VM i.e., what tasks have been processed by the VM. It also tell about the number and details of high priority tasks currently processed in the VM in a manner similar to the bees who finds an abundant food source and then they update the other bees in the bee hive about that through dance and the updating gives a perception about the hive scenario.

4. Proposed Method

The better your paper looks, the better the Journal looks. Thanks for your cooperation and contribution. Data center broker of cloud environment has responsibility, to which machine the job will be allotted and this decision i.e.

choosing virtual machine is completely based upon the available resources at that machine and number of high priority cloudlets assigned to it. The honey bee algorithm is scheduling policy, which is based on first come first serve method, honey bee also takes care of priority of task but still it makes imbalance of load, to get more efficient results. We proposed modified HBB-LB (Honey Bee Behavior Load Balancing) which keeps account of space lagging and processing capacity left in machine and number of task assigned to it with their priority. To achieve optimal utilization of resources and get better response time and less imbalance degree.

This modified honey bee algorithm will check all the virtual machine which are under loaded, keep knowledge of UVM (Under loaded Virtual Machine) list and when cloudlet arrives it check all the under loaded VM. Unused space or processing capacity is checked in UVM, if it is already allocated with some cloudlet, UVM will be allocated by the removed task load which will best fit to it, for that we need to have information that how much of load is transferred from OVM (Over loaded Virtual Machine). Extra load and free space of VM is calculated by subtraction of VM capacity and VM load. Ratio of OVM's extra load and UVM's free space is taken to know their best fit condition, for a particular OVM extra load. All free spaced UVM with priorities of task which are already in queue of VM are accounted to get desired state of best fit condition. So during the allocation of cloudlet to under loaded VM this modified honey technique not only keep track of under loaded VM but also the queues of task and their priorities of cloudlets which are waiting for VM, after analyzing both of these parameters allocation of task to VM is performed

This honey bee inspired algorithm maximize the throughput as compared to any other load balancing techniques but drawback with honey bee inspired algorithm is that it take every cloudlets in first in first out fashion to virtual machine. Sometimes it may be possible that there are other virtual machines in which there would be less waste of processing resources. Suppose that, 50 MB task has to be assign to the VM and because of honey bee inspired techniques, it assign to first available machine having space of 500 MB and there might be other free virtual machine which have capacity of 75 MB. In order to overcome this drawback, the modified version of honey bee algorithm is proposed to overcome with this problem and priority measures are also checked for each allocation of cloudlets to any VM.

4.1 Mathematical Model

Let take 'm' number of Virtual Machines, and all of the virtual machine here are parallel and they are not related with each other. Virtual Machine is denoted here by VM.

$$VM = \{VM_1, VM_2 \ldots\ldots VM_m\}$$

For 'm' machines there would be 'n' number of cloudlets. Cloudlets (task) which are taken here for processing in VM should be non-preemptive so in order to diminish the interruption while processing the cloudlets.

$$Cloudlet = \{Cloudlet_1, Cloudlet_2 \ldots\ldots Cloudlet_n\}$$

Let the processing time for a particular cloudlet, $cloudlet_i$ over a particular machine VM_j denoted by P_{ij}

and processing time of all task at particular VM is given by

$$P_j = \sum_{i=1}^{n} P_{ij} \qquad j=1, 2 \ldots m$$

At the time of load balancing, the task are moved from one to other virtual machines to minimize makespan because of different processing capabilities of VM the completion time of task may vary according to different VM.

Makespan can be defined as the overall task completion time means time taken from start time of first task to finishing time last task.
The load on virtual machine is calculated by the information which is received from the data center and standard deviation has to calculate the amount of deviation on each virtual machine.

4.1.1. Capacity of a VM

Capacity of VM means that what the actual configuration of VM, number of processing element in VM and processing capacity of VM that how many instructions per second can VM execute.

$$VM_{cap} = Pe_{numj} * Pe_{mipsj} + VM_{bwj}$$

Here, capacity is denoted by VM_{cap}, processing element Pe_{numj} is number of elements in the particular virtual machines, Pe_{mipsj} is amount of processing speed does particular virtual machine have and VM_{bwj} is bandwidth ability that particular VM have.

4.1.2. Capacity of all VM

It is summation of the capacity of all the VM.

$$VM_{cap\ total} = \sum_{i=1}^{m} VM_{cap\ i}$$

4.1.3. Load on a VM

Load on VM is calculated as the number of cloudlets allotted to VM_i. VM_i is a particular i[th] VM, time 't' divided by servicer rate of VM_i at time t. Load of VM is calculated as the ratio of number of cloudlets at time 't' and service rate at a time.

$$VM_{load}, t = \frac{N(Cloudlet_i, t)}{S(VM_i, t)}$$

4.1.4. Total Load

Summation of load on all VM, to know actually how much load is there with datacenter which has to be manage by processing element called VM. It is just by adding loads on each VM.

$$VM_{load\ total} = \sum_{j=1}^{n} VM_{load}^{j}$$

4.1.5. Processing time of VM

Processing time of VM illustrate that the time which will be taken by any particular VM at the time of processing when cloudlets are assigned to it. and to denote the processing time of any 'i' VM is by $Time_{proc}^{i}$.

$$Time_{proc}^{i} = \frac{VM_{load}^{i}}{VM_{cap}^{i}}$$

Summation of processing time of VM, it calculate the complete time which is taken by all VMs which are allotted with cloudlets at any datacenter at any particular time.

$$Time_{proc} = \frac{VM_{load\ total}}{VM_{cap\ total}}$$

Normalization function is used to check the deviation occurred in the task occurrence in the VM and its processing time. Normalization function is used to conclude the deviations at a single value and is compared with the fixed threshold so as to determine that either situation needs to be balance or it is already balanced.

$$S.D = \sqrt{1 - \sum_{i=1}^{m} \left(\frac{Time_{proc}^{i} - Time_{proc}min}{Time_{proc}max - Time_{proc}min} \right)^2}$$

4.1.6. Load Balancing Strategy

After initializing all element of computing like cloudlets, VMs and datacenter. VM grouping is performed on the basis of the deviation of load which we got after calculating the normalized function of load at the time of processing. Now we set the threshold according to our expecting results and decide which VM is how much balanced

If total load L of virtual machine is more than total capacity then this condition is called cannot be balanced

If $VM_{load\ total} > VM_{cap\ total}$ (system cannot be balanced)

Else

If

S.D <= threshold (system is balanced)

Else

Trying to balance

After that virtual machines are categorized according to their loads into over loaded VM, under loaded VM and balanced VM. Cloudlet are allotted to that VM which have near about exact amount of space and processing capabilities to process that task, so no extra resources get wasted. Hence, we got best fit in current scenario of cloud environment and priority of cloudlets is also accounted which are queued in VM to ensure that our cloudlet which is to be processed have greater priority than cloudlets which are queued in VM. This task of assigning of cloudlet to under loaded VM is gone through for each cloudlet and thus cloudlet are allotted in such a way to get better performance.

VM selection according to space

$$UVM^{i}\ free\ spc = UVM^{i}{}_{cap} - UVM^{i}{}_{load}$$
$$OVM^{j}\text{ext load} = OVM^{j}{}_{load} - OVM^{j}{}_{cap}$$

$$max(fitness) = \left(\frac{OVM^{j}\text{ext load}}{UVM^{i}free\ spc} \leq 1 \right);$$

$\forall$ i = 1, 2 …m-1; j = 1, 2 … m-1;

$OVM\ \text{ext load}$ is the extra load with OVM $UVM\ free\ spc$ is the free space in UVM, ratio should be

maximum to 1, but not more then 1, because approx. to 1 will be the best fit condition for transfer of load to VM which have capacity to process.

VM selection according to priority

$$Cloudlet_h \rightarrow LVM_{free\ spc}, min\left(\sum Cloudlet_h\right) \in UVM_{free\ spc}$$

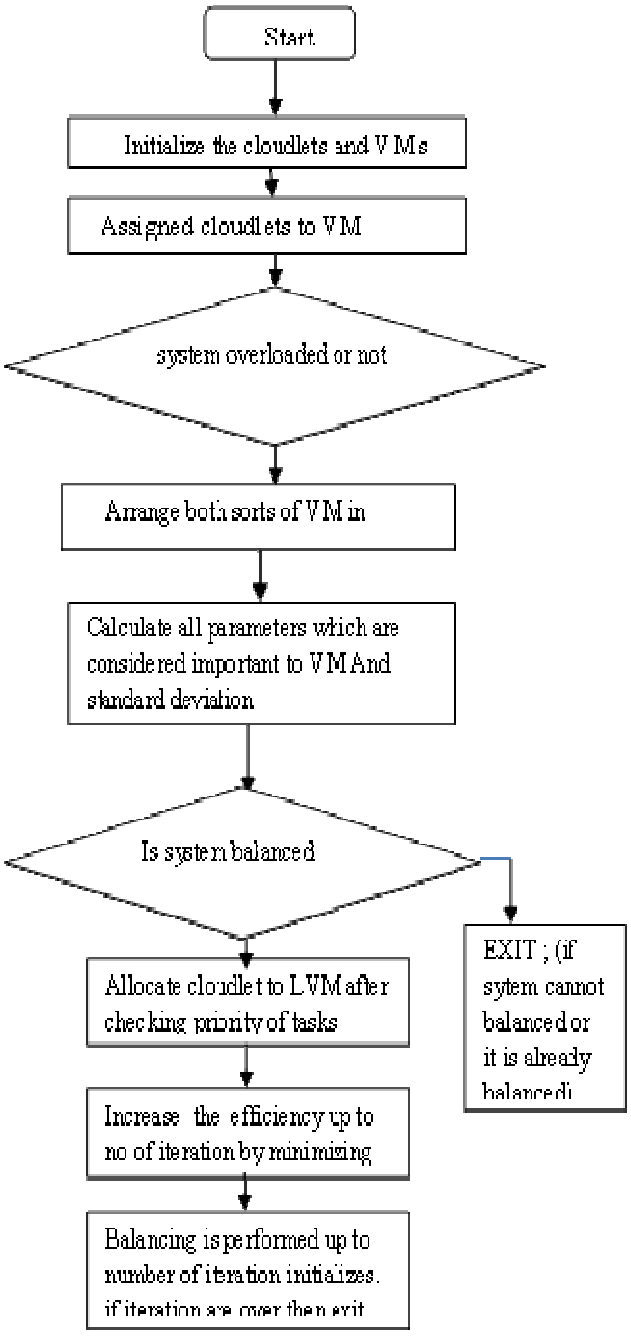
$$Cloudlet_m \rightarrow UVM_{free\ spc}, min\left(\sum Cloudlet_h + \sum Cloudlet_m\right) \in UVM_{free\ spc}$$

$$Cloudlet_l \rightarrow UVM_{free\ spc}, min\left(\sum Cloudlet\right) \in UVM_{free\ spc}$$

$Cloudlet_h$ is the high priority task and $Cloudlet_m$, $Cloudlet_l$ are the middle and low priority task respectively.

When the cloudlet having high priority moves to a VM which already have high priority task hence the cloudlet have to consider that so because of this hurdle the high priority cloudlet will always search for the under loaded VM which have less number of high priority task.

FLOWCHART:



5. Algorithm

1 Initializing iteration value and threshould values

$$iter = 0;$$

$$iter_{max} = 99;$$

$$std_{threhold} = 0.99;$$

2 Declaration and Assigning of the load and capacity to VM and cloudlets.

$$while(iter < iter_{max})$$
$$\{$$
$$VM_{avl} = getAvlVMs();$$
$$CloudLet_{avl} = getCloudLets();$$
$$VM_{cap} = getCapacity(VM_{avl});$$
$$VM_{load} = assignRandom(VM_{avl}, CloudLet_{avl});$$
$$CloudLet_{load} = getLoad(CloudLet_{avl});$$

3 Assigning priority to the cloudlets available

$$CloudLet_{pri} = getPriority(CloudLet_{avl});$$

4 Categorize VM into overloaded and under loaded on the basis of load and capacity

$$VM^{i}_{ovm} = i, \forall i\ VM^{i}_{load} > VM^{i}_{cap};$$
$$VM^{i}_{uvm} = i, \forall i\ VM^{i}_{load} < VM^{i}_{cap};$$

5 Get the priorities of cloudlet for OVM and UVM

$$CloudLet_{pri,ovm} = getPriority(getLoad(VM_{ovm}));$$
$$CloudLet_{pri,uvm} = getPriority(getLoad(VM_{uvm}));$$

6 Sort OVM in descending and UVM in ascending order

$$VM^{descend}_{ovm} = sortDescending(CloudLet_{pri,ovm});$$
$$VM^{ascend}_{ovm} = sortAscending(CloudLet_{pri,uvm});$$

7 Calculating the processing time for each VM

$$time^{i}_{proc} = \frac{\sum VM^{i}_{cap}}{\sum VM^{i}_{load}}$$
$$, i = \{1,2,3, \ldots., numOfVMs\};$$

8 Calculate standard deviation, total capacity of VM and total load with cloudlets

$$std_{proc} = calcStd(time_{proc});$$
$$VM_{cap,total} = \sum VM_{cap};$$
$$CloudLet_{load,total} = \sum CloudLet_{load};$$

9 Decision for load balancing

$$if(VM_{cap,total} < CloudLet_{load,total})\{$$
$$exit(\text{Balancing Not Possible});$$
$$elseif(std_{proc} < std_{threshold})\{$$
$$exit(\text{System is Balanced});$$
$$\}$$
$$else$$

10 Load balancing

$$\{$$
$$while(VM_{ovm} \neq 0\ \&\&\ VM_{uvm} \neq 0)$$
$$\{$$
$$for(i = 1; i \leq \#(VM_{ovm}); i++)\{$$
$$minimize\left(\left|\frac{getCapacity(VM_{uvm})}{getLoad(VM_{uvm})}\right| - \left|\frac{getCapacity(VM_{ovm})}{getLoad(VM_{ovm})}\right| + \frac{|VM_{uvm,pri} - VM_{ovm,pri}|}{VM_{uvm,pri}}\right)$$
$$\}$$
$$\}$$
$$\}$$

Here in cloud computing context nectar is known as VMs and cloudlets are honey bees, firstly the VMs and cloudlets are initialized the *iter* (iteration value) and set the threshould value. After that we declare the available resources like VM and cloudlet with and assigned random load, capacity to it and priorities are set to the cloudlet configured with particular processing abilities. Afterwards, load and capacity is compared on VMs and is

categories in to UVM and OVM. If load on VM is greater than capacity then it is called OVM (Overloaded Virtual Machine) otherwise it is called UVM (Under loaded Virtual Machine). After that, the priority of cloudlet in both OVM and UVM are specified by $CloudLet_{pri}$. Then processing time of every VM is calculated, where the cloudlet was arranged in priority and particular sorted order to diminish the effect of transferring of cloudlets. After that normalize value is taken by standard deviation of processing time which is compared by standards initialized. If load on VM is greater than capacity then balancing is not possible and if normalize value is less than standards set then system is balanced, if both above condition are not entertained then balanced is done by moving cloudlet to UVM from OVM, for this a minimize function is created which takes the ratio of load and capacity of all UVM as well as of OVM. We check that for which UVM the extra load of OVM will be well suited on cord of minimum value we get by subtracting these ratios and calculation of absolute priority value of OVM extra task and task which are in queue of UVM.

6. Results

A cloud computing system has to handle several hurdles like flow of task, load balancing on virtual machines, reliability, security measure, scalability and many others. Sometimes unexpected behaviour of demands and other various issues are considered that why experiments cannot be done on the real computing environment, if it is done then it will cost to the customer. So there is need of simulator arises, simulator is framework which allows us to perform all those activities which can be performed at real cloud environment for the experiment purpose.

Proposed algorithm is more efficient and it reduce the average processing time, imbalance degree and makespan time as compared to previous (HBB LB) and FIFO.

Fig 6.1, calculates the average processing time to process all cloudlet at a particular configuration. Here, we compare our proposed method Modified Honey Bee Behavior algorithm with FIFO and HBB-LB (Honey Bee Behavior load Balancing). We can evaluate that the results of proposed technique is better than honey bee as well FIFO approach and degree of imbalance is calculated while load balancing is also far much less then

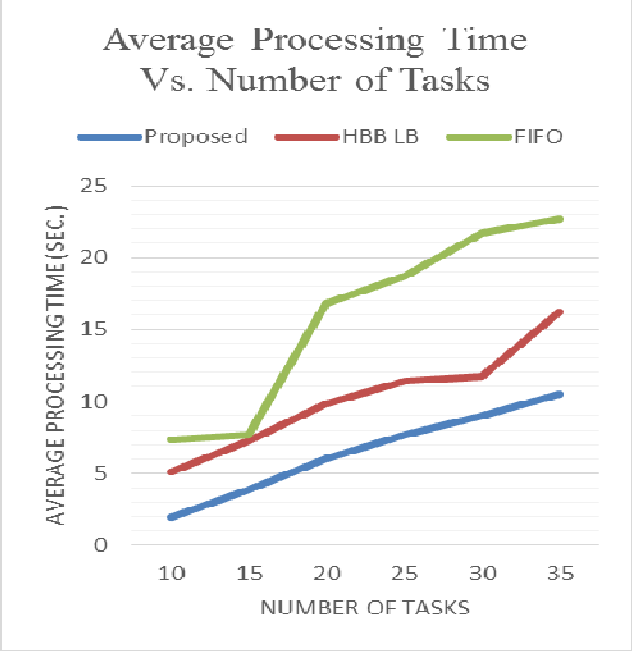


Fig 6.1: Comparision of HBB LB , FIFO and modified HBB Algo average respose time

In the tabular results, we calculate average of processing time taken by cloudlets, includes waiting and executing time to respond for load balancing (average processing time) and degree of imbalance for a datacentre having 10 VM and varying number of cloudlet kept increasing from 10 to 35 on Modified HBB LB technique.

Tables 1, 2, 3 illustrate the average processing time and degree of imbalance on different configuration of VM and cloudlet, were number of VM remains same but number of cloudlet kept increasing In table 1 - 3, we calculate the average processing time and degree of imbalance of proposed Modified Honey Bee techniques, HBB-LB (Honey Bee Behavior Load Balancing) and FIFO respectively. In all three tables, it is evident that the average processing and degree of imbalance of our proposed technique is comparatively most efficient among the compared techniques

Table 1: Calculating average processing time and their imbalance degree for Modified HBB load balancing approach

Cloudlets / VMs	10	15	20	25	30	35
10	1.03	2.17	11.84	5.86	8.29	10.26
10	2.21	4.56	2.6	5.31	13.16	18.24
10	1.34	5.5	6.3	6.28	11.59	6.11
10	2.03	4.51	6.09	5.19	5.8	5.14
10	3.25	2.41	3.08	15.85	6.3	12.64
Average process time	1.972	3.83	5.982	7.698	9.028	10.478
imbalance	0.38	0.43	0.56	0.4	0.47	0.33

Table 2: Calculating average processing time and their imbalance degree for HBB LB approach

Cloudlets / VMs	10	15	20	25	30	35
10	7.3	8.74	14.16	20.22	12.09	21.7
10	3.58	2.62	6.88	11.58	7.71	15.23
10	4.22	5.54	5.39	5.12	11.79	20.09
10	6.19	9.74	12.09	8.19	15.7	12.34
10	4.02	9.5	10.53	12.09	11.45	11.8
Average process time	5.062	7.228	9.81	11.44	11.748	16.232
imbalance	0.53	0.59	0.62	0.67	0.52	0.49

Table 3: Calculating average processing time and their imbalance degree for FIFO approach

Cloudlets / VMs	10	15	20	25	30	35
10	3.84	3.02	42.03	28.51	28.42	22.65
10	2.93	10.86	8.51	8.04	29.05	17.55
10	2.57	12.63	15.4	19.34	14.38	16.44
10	11.96	7.16	9.2	30.97	16.43	25.6
10	15.42	4.59	8.72	6.78	20.08	31.47
Average process time	7.344	7.652	16.772	18.728	21.672	22.742
imbalance	1.21	1.63	1.89	1.66	1.67	1.53

In fig 6.2, we compare the degree of imbalance for FIFO, previous (HBB LB) and proposed (modified HBB LB) and we can see that imbalance degree for modified HBB LB is lowest as compared to other techniques. Degree of imbalance is to know how much of imbalance is there while doing load balancing. Here is the mathematical formula to calculate degree of imbalance.

$$\text{Degree of Imbalance} = (T_{max} - T_{min})/T_{avg}$$

where Tmax and Tmin are the maximum and minimum time taken by task Ti among all VMs, Tavg is the average time taken by Ti task on VMs. Our load balancing technique reduces the degree of imbalance drastically as we compared.

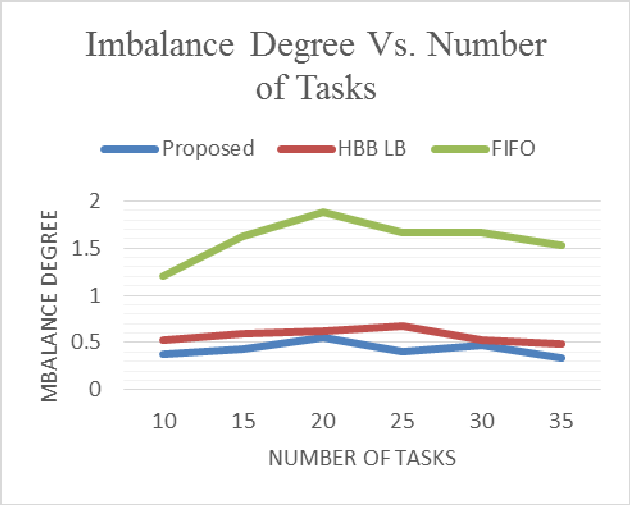


Fig 6.2: Comparison of imbalance degree while balancing on respective approaches

This Fig 6.3, 6.4 and 6.5 illustrates the comparison of makespan time. Comparison is done between FIFO HBB-LB and proposed Modified Honey Bee technique.

The comparison of three techniques is done on three different configurations and every figure illustrates different combination of VMs and cloudlet. In figure 6.3 – 6.5 , X-axis represents the number of cloudlets and Y – axis represents time. In fig 6.3, we compare makespan time on fixed number of VM as the number of cloudlets ranging from 10 - 35

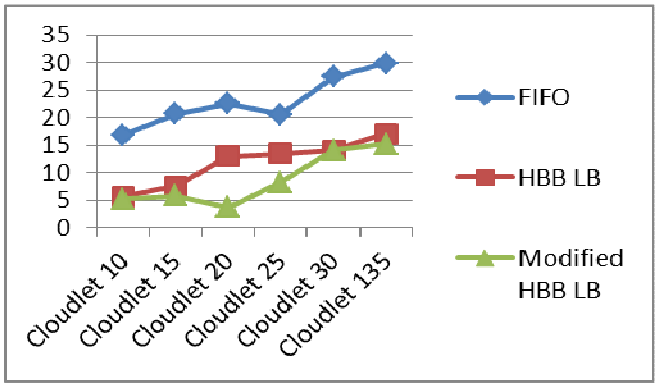


Fig 6.3 Comparison of makespan time of FIFO, HBB LB and Modified HBB LB on 5 VM

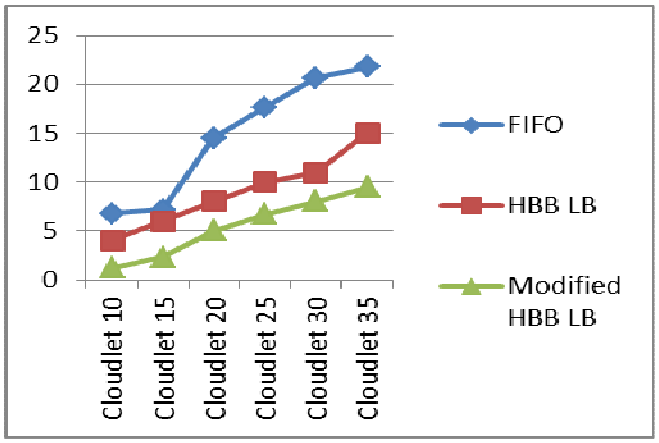


Fig 6.4 Comparison of makespan time of FIFO, HBB LB and Modified HBB LB on 10 VM

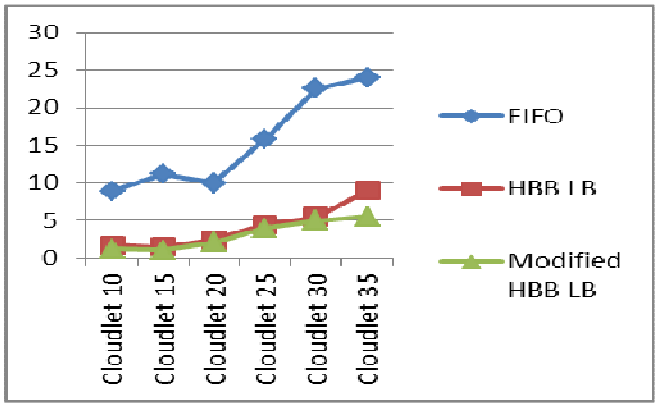


Fig 6.5 Comparison of makespan time of FIFO, HBB LB and Modified Honey bee on 15 VM

7. Conclusion

In this paper, we designed modified honey bee load balancing technique inspired by honey bee behavior and HBB-LB. In modified honey bee approach, it keep track of load of all VMs, its capacity and cloudlet size, to ensure that the assignment of jobs to best feasible VMs. Allocation process of jobs should be in such a way that VM resources should be utilized in optimized way to reduce the wastage of resources. Proposed technique also consider the priority of cloudlets while assigning. Modified Honey Bee load balancing technique diminish the wastage of resource as it was main drawback of HBB-LB, because of prior information of cloudlet size , VM capacity and available free processing capacity left with VM. Priority based allocation is there to give particular importance to particular cloudlets. The results, in which proposed approach is compare with other balancing techniques on average processing time, makespan time

and degree of imbalance parameter and this techniques works well with heterogeneous cloud computing environment and with non-preemptive tasks.

References

[1] D.L. Eager, E.D. Lazowska, J. Zahorjan Adaptive load sharing in homogeneous distributed systems, The IEEE Transactions on Software Engineering, vol. 12, 1986, pp. 662–675.

[2] N. Malarvizhi, V. Rhymend Uthariaraj Hierarchical load balancing scheme for computational intensive jobs in Grid computing environment, in: Advanced Computing, 2009 ICAC 2009. First International Conference on, 13– 15, , 2009, pp.97-204.

[3] J. Rex Fiona, Roshni Thanka, Parallel Genetic Load Balancing with Competency Rank in Computational Grid Environment, International Journal of Engineering Research and Applications, Vol. 3, Issue 1, 2013, pp.1814-1817.

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

[5] Jasmin James efficient VM load balancing algorithm for cloud computing environment, in International Journal on Computer Science and Engineering (IJCSE), Vol. 4 No, 2012, pp. 1658-1663.

[6] 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, 2002, pp. 181–196.

[7] Y. Hu, R. Blake, D. Emerson, An optimal migration algorithm for dynamic load balancing, Concurrency: Practice and Experience 10, 1998, pp. 467–483.

[8] S. Genaud, A. Giersch, F. Vivien, Load balancing scatter operations for grid computing, in: Proceedings of the 12th Heterogeneous Computing Workshop (HCW'2003), Nice, France, 2003, pp. 101–110.

[9] M. Moradi, M.A. Dezfuli, M.H. Safavi, Department of Computer and IT, Engineering,Amirkabir University of Technology, Tehran, Iran, A New Time OptimizingProbabilistic "Load Balancing Algorithm in Grid Computing" IEEE Vol.1 ,2010, pp. 232-237.

[10] Elina Pacini a, Cristian Mateos b,c,⇑, Carlos García Garino a,dBalancing throughput and response time in online scientific Clouds via Ant Colony Optimization (SP2013/2013/00006) Vol. 84, 2015, pp. 31–47.

[11] Dhinesh Babu L.D. , P. Venkata Krishnab "Honey bee behavior inspired load balancing of tasks in cloud computing environments", ELSEVIER Applied Soft Computing 13, 2013, pp.292– 303 .