DOI 10.15622/sp.57.2

M.A.S. MOSLEH, G. RADHAMANI

A NOVEL FUZZY QOS BASED IMPROVED HONEY BEE BEHAVIOR ALGORITHM FOR EFFICIENT LOAD BALANCING IN CLOUD

Mosleh M.A.S., Radhamani G. **A Novel Fuzzy QOS Based Improved Honey Bee Behavior Algorithm For Efficient Load Balancing In Cloud.**

Abstract. Nature inspired algorithm based Load balancing of tasks on virtual machines (VMs) has become an area of greater research interest. Honey Bee Behavior Based Load Balancing (HBB-LB) was introduced to balance the load with a maximum throughput. This approach also balances the priorities of the tasks on the VM to minimize the waiting time of the tasks. However, HBB-LB considers only the VM load for balancing the load, which might not be sufficiently effective. This paper proposes an Improved Honey Bee Behavior Based Load Balancing (IHBB-LB), taking into consideration a few more QoS parameters of VM, such as service response time, availability, reliability, cost and throughput to enhance load balancing. Response time is vital in determining the instant activity of a VM while availability determines available resource and state of VM (idle or active) and Reliability determines the level of trust in a VM. Most importantly, Cost for utilizing a VM and Throughput (capability of VM) are also essential in determining the VM efficiency. But, the inclusion of multiple QoS parameters results in multi-objective optimization problem. As a number of QoS parameters are computed, the Fuzzification of the QoS values was performed through the generated fuzzy rules and multi-objective optimization problem was eliminated. The experiments were performed in terms of makespan, response time, degree of imbalance and the number of tasks migrated and results indicate that the IHBB-LB provides a better level of performance.

Keywords: Optimization, QoS parameters, Cloud Computing, Load Balancing, Fuzzification.

1. Introduction. Cloud computing is an internet-based approach processed using the shared resources on a cloud consisting of multiple computers interlinked together. Cloud includes the concepts of distributed and parallel computing to enable shared resources and applications. This approach focuses on the advantages of cost, flexibility and availability of the service users, which in turn increases the demand for the cloud services. The increase in demand increases technical issues such as high availability and scalability, in Service Oriented Architectures (SOA) and Internet of Services (IoS) applications, which can be overcome by the load balancing concept of allocating dynamic workload evenly across all the machines. In [1] developed adaptive cost based task scheduling (ACTS) for efficient task scheduling in cloud However, task scheduling alone cannot improve cloud computing service to the users; exploring several other factors like balanced load, effective VM migration, etc. also contribute to efficient cloud user experience.

Load balancing in cloud computing is essential to tackle the overload and under-load conditions of virtual machines (VM). Efficient load balancing can be achieved using optimization techniques. Whenever an under-load of VM

or overload of VM occurs, the tasks have to be loaded to achieve optimal machine utilization. The Stochastic Hill Climbing approach [2] selects the VM based on a random selection of the uphill move. Stochastic Hill Climbing maintains a VM status table and assigns a random id to each VM. When a task arrives, a VM is chosen randomly and a request is sent to check the status of task allocation, following which the task is assigned. The Simulated Annealing approach [3] obtains all the load balance requests and selectively drops some less important requests from default users in order to maintain the load balance of the SLA premium users. Genetic algorithm [4] allocates tasks by using natural selection strategy of crossover and mutation on the selected VM in order to determine the optimal VM for efficient load balancing. Ant colony optimization [5] follows the natural ant behavior of food detection to allocate the tasks by estimating the computation of the VMs. While honey bee based optimization algorithms are more common in cloud computing. Honey bee mating algorithm [6] can be used for effective and optimal cloud resource selection. The honey bee behavior based load balancing (HBB-LB) [7] selects the overloaded VM, moves the loads to unallocated VMs and then indicates the status of VM to other tasks so that new high priority tasks could be allocated to the unallocated VMs. However, it considers only one parameter (i.e.) the VM load conditions and this is not enough for effective load balancing. The performance of this honey bee behavior based load balancing techniques can be improved by considering the many QoS factors of VMs.

This paper proposes the Improved Honey Bee Behavior Based Load Balancing (IHBB-LB) based on the consideration of multiple QoS parameters, such as service response time, availability, reliability, cost and throughput. These parameters have been included in IHBB-LB for balancing the load when certain VMs are overloaded and the some under-loaded. Multiple QoS parameters result in the multi-objective optimization problem. In general, simple techniques such as crowding distance or weight estimation can be employed to find the pareto-optimal solutions for the multi-objective optimization problem. However, the use of more number of QoS parameters makes it difficult to utilize the general approaches. Hence, the concept of Fuzzification is employed for generating of fuzzy rules in order to resolve the multi-objective problem. Through this, it is also possible to effectively balance the load in the cloud environment.

The rest of the paper is organized as follows: Section II presents the review of related work. Section III explains the HBB-LB and the proposed IHBB-LB methodologies in detail. Section IV presents the performance evaluation of the proposed methodologies while section V concludes the paper.

2. Related Works. Load balancing is one of the most sought after research area in cloud computing. Many researchers have developed their own version of load balancing technique based on the issues they considered

as the prime focus. Analyzing some of the researches related to our research work can make a mile difference in the proposed solution. For example, Dual Directional Files Transfer Protocol (DDFTP) [8] developed for efficient balancing of the downloading servers without duplicating the same files is concept to watch out for. Though this concept seems alien to our concept, deep analysis shows that DDFTP provides efficient load balancing among the multiple heterogeneous data servers with a minimal overhead.

Load balancing using the Bayes theorem (LB-BC) [9] is another example of how well an heuristic clustering concept utilized for task allocation. Bayes theorem was included with clustering algorithm to provide optimal clustering set of physical hosts from which the efficient hosts are selected for balancing the tasks. Hybrid meta-heuristic algorithm for VM scheduling & load balancing [10] and load balancing model based on cloud partitioning [11] are some of the load balancing schemes focusing on reduced computation time. Cache (C^2) [12] for metadata server clusters implements the adaptive cache diffusion to detecting the overloaded nodes and process adaptive replication scheme to split the load in the overloaded nodes. Agent based load balancing [13] for the cloud data centers utilizes the migration heuristics to provide effective load balancing. Another autonomous agent based load balancing algorithm proposed [14] provides comparatively better balancing than other agent based algorithms. However agent based schemes are more commonly used so it becomes mandatory to analyze other options for better load balancing. Game theoretic static load balancing models [15] utilizes equilibrium game concepts either two-player or multi-player to efficiently balance the load with minimal server overload situations. Though these schemes try to resolve load balancing issues, they are static. Dynamic annexed balance method called Cloud load balancing (CLB) [16] considered both server processing power and computer loading to minimize overload and computation complexities. Although these single methods have ruled the load balancing concept, there are some hybrid load balancing concepts. For example DeMS [17] consisting of On-Demand scheduling, Querying and Migrating Task (QMT) and Staged Task Migration (STM) reduced overhead and balances the load effectively than single methods. Energy aware [18] and resource aware [19] schemes were also been largely employed for load balancing in cloud. Though the above described schemes provides better load balancing than their predecessors, in any application the nature inspired schemes have worked better. The same concept can be applied in cloud load balancing.

The foraging behavior of honey bees [20] was considered to develop of an intelligent optimization technique. The intelligent foraging behavior of the bee swarm focuses on the collection of honey for energy. Foraging begins only when suitable conditions such as temperature, weather, etc., are satisfied.

The bees maintains a thumb rule for foraging area which is be fixed around a hive for two miles, as the food obtained within this area will gain weight while the energy spent beyond this area will be greater than the energy obtained. Foraging at the extreme distances wears out the wings of the individual bees and reduces its life expectancy, thus reducing the efficiency of the bee colony. The foraging behavior depends on the communication of distance and direction of food source by round dance, waggle dance and shaking signals of the individual bees. Based on this intelligent foraging behavior, Artificial Bee Colony (ABC) algorithm [21] has been developed. ABC consists of three bees namely: employed bees, onlookers and scouts. Employed bees search for the food source and return to hive and dance on this area. The employed bee whose food source has been abandoned becomes a scout and starts scouting a new food source. Onlookers watch the dances of employed bees and choose food sources depending on the dances. ABC algorithm has the advantage of global search ability, which is achieved by the introduction of the neighborhood source production mechanism. ABC is used in many fields such as signal processing, image processing, scheduling problems and optimization problems. This approach has been employed for cloud load balancing as stated in HBB-LB [7] but as it considers only a single parameter, it cannot be considered efficiently balanced. Hence we intend to include additional load balancing parameters to enhance the load balancing performance of the HBB-LB in this paper.

3. Improved Honey Bee Behavior Based Load Balancing Algorithm. Honey bee behavior inspired load balancing (HBB-LB) algorithm has been employed for efficient balancing of the tasks. HBB-LB exhibits the basic honey bee foraging (wide search) behavior with respect to the tasks loaded in the VMs. However, HBB-LB algorithm considers only the load conditions in the selection of VM, while the other vital QoS factors which are necessary for enhancing the efficiency of computation in cloud are computing are not taken into consideration. In order to balance the available load to the VMs effectively, the VM characteristics are needed to be determined. The major considerations for the VM are makespan, processing time, capacity and load of a VM. While the HBB algorithm considers these parameters in the objective function for sorting the VMs, from the analysis it is found that some important parameters namely response time, reliability, availability, cost and throughput are equally vital in characterizing a VM.

Hence, an enhanced version of HBB-LB called as Improved Honey bee behavior based Load balancing algorithm (IHBB-LB) has been proposed which considers multiple QoS parameters in the selection of the VMs. The VMs are sorted in the ascending order based on the load conditions and QoS parameters such as service response time, availability, reliability, cost, and

throughput. The tasks removed from the overloaded VMs are considered as the honey bees. When the tasks are submitted to the under-loaded VM, and then would update the number of various priority tasks and the load of the respective VM to all other waiting tasks. Based on this input, those tasks with higher priority are allocated to the under-loaded VMs enabled with better QoS parameters based on the list sorted. As the VMs are arranged in an ascending order based on load conditions and QoS parameters, those tasks removed from overloaded VMs will be submitted only to the under-loaded VMs. The main advantage of this approach is the updating of the priorities and load conditions, which is similar to the dance movements of the scout bees. This proposed IHBB-LB approach was found to be very effective in load balancing of tasks in cloud computing environments. Figure 1 shows the load balancing procedure in cloud using the proposed IHBB-LB.

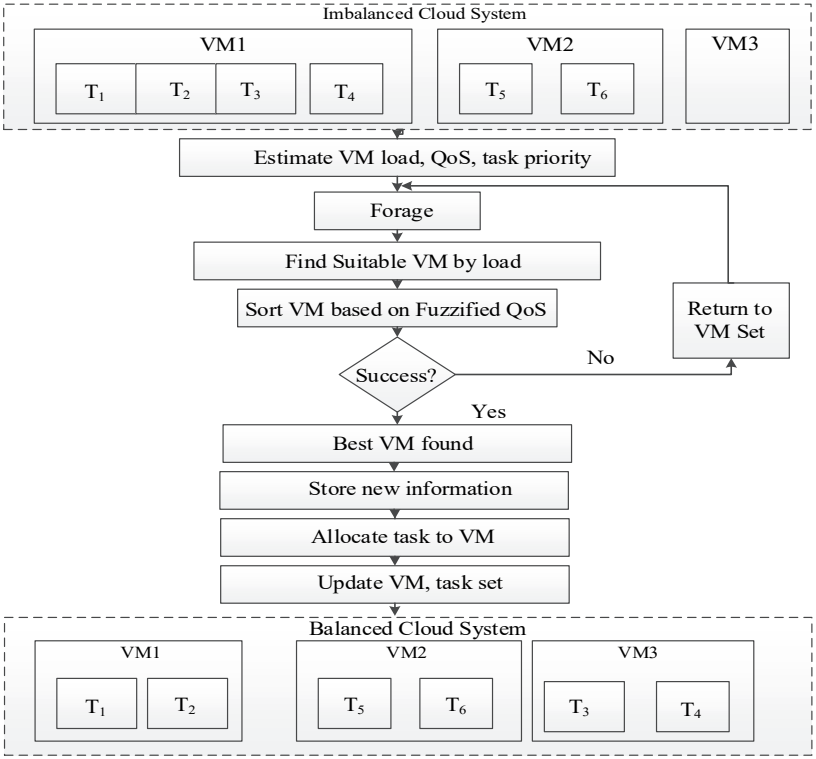

Fig. 1. Procedure of IHBB-LB

Let us analyze the load conditions and the QoS parameters to be considered for VM load balancing in order to clear the air about the importance

of each QoS metric in SLA agreements. Although many QoS aware load balancing concepts are commonly available, no method has achieved the efficiency of HBB algorithm. Hence improving the HBB algorithm, results in maximization of load balancing performance. It is worth mentioning, our proposed concept is suitable for both public and private clouds.

The proposed IHBB-LB is implemented in CloudSim for evaluation. The proposed system follows a data center model (server/client model) to process the cloud user requests. These requests are generally the set-up of different types of tasks. The server is responsible for managing the user requests and allocating the tasks to the VM. The main functions of the cloud server in this aspect are the selection of an efficient physical machine (PM) for VM instance, monitor the task execution in VM through VM monitor and send the computational results to the users. Different tasks have different execution patterns for which they require multiple types of execution resources. A single task can also be split into multiple steps and each allotted to different machines with different I/O bandwidth. Suppose there are n physical machines and a task requires R resources, then the complete set of execution resources are considered in evaluating the performance.

Let us assume the condition that all the PMs are now initiated into VM instances. A set of VMs are initialized as $V = \{V_1, V_2, \ldots . V_m\}$ where m is the total number of VMs. A set of tasks $T = \{T_1, T_2, \ldots . T_n\}$ is initialized where n is the total number of tasks. Each task is assumed to be accepted from the user requests and each task is constructed by multiple sub-tasks. All the sub-tasks are corresponding to the web services where they are data transmission tasks or data read/write through disk. Each task is executed on the respectively allotted VM within the allotted resources (CPU rate and network bandwidth).

The completion time of a task T_i on a virtual machine VM_j is denoted as $CT_{ij}, i = 1, 2, \ldots . n$ and $j = 1, 2, \ldots . m$. Based on CT_{ij}, the makespan is computed. Makespan is an indicator of the general throughput of the system, generally said as the completion time of a VM. Small values of makespan mean that the server is providing good and efficient planning of tasks to resources. It is different from the execution time which means the completion time of tasks in a VM. Makespan is denoted as M and it is given as

$$M = \max \left| CT_{ij} \right|; \qquad i \in T \ . \tag{1}$$

Makespan is generally higher than execution time as it includes all the hold time/ wait time of the VM. However it should be minimized for efficient VM which is done by minimizing the waiting time of a task T_i . The vital condition that should be satisfied in this scenario is that the processing

time must be less than or equal to the maximum time for completion $P_j \le CT_{\max}$. In order to evaluate this condition, P_j is computed as

$$P_j = \sum_{i=1}^{m} P_{ij}, \tag{2}$$

where P_{ij} is the processing time of a task T_i on VM_j.

However, at the time of load balancing, the tasks will be moved from overloaded VMs to other VMs in order to minimize the makespan and reduce the response. The movement of tasks from one VM to another varies the processing time of the tasks based on the VM capacity. $CT_{\max}$ varies with the task migration due to the load balancing.

The tasks removed from the overloaded VMs are considered as honey bees, as in HBB-LB. The main concepts of updation all number of priority tasks and the load allocated to that particular VM. The priority tasks are allocated to the under-loaded VM with better QoS parameters. Therefore, the load conditions and QoS parameters of the VMs are calculated and the VMs are ranked based on these parameters. The priority tasks are allocated to the VMs from the list sorted. In order to balance the workload, the capacity and load are to be calculated. The standard deviation is also calculated to measure the load deviations on VMs.

Capacity and load are considered as the major determinants of the VM characteristics. Capacity determines the total possible resources of a VM that are allotted for task execution. If the minimum number of processors in a VM is denoted as Np with the mips (million instructions per second) for all the processors denoted as Pmips, B denotes the bandwidth requirement of a VM for communication, then the capacity C of a single VM can be calculated as

$$C = Np + Pmips + B. \tag{3}$$

The capacity of a VM denotes the available resources for executing a particular task of length l with sufficient satisfaction of time deadlines. The capacity of a VM server is the summation of all VMs given as $C = \sum_{i=1}^{m} C_i$. As said before, load of a VM determines how much resources are available for newer tasks. The load of a single VM can be estimated based on the total length of the tasks allocated to that VM. If the number of tasks at time t on a service queue of VM is given as T(t) and execution time as S(T,t), then the load $L_{V,t}$ is computed as

$$L_{V,t} = \frac{T(t)}{S(T,t)}. \tag{4}$$

The overall load of all VMs can be obtained by adding the load of all VMs, as $L = \sum_{m} L_{V_m}$. Based on the load and capacity of V_j, the processing time of the VM can be calculated. Processing time of task (given in equation (2)) is different from the processing time of VM. The processing time for V_j is given as

$$PT_j = \frac{L_{V_m}}{C_i}. \quad (5)$$

The same can be computed for as PT for all VMs. Depending upon the processing time, the standard deviation σ of the load of VMs can be computed. σ is the quantity to measure to determine the deviation of load conditions when a new task is accepted in a VM. It is estimated as to determine the effect of a task of length l to the VM j.

$$\sigma = \sqrt{\frac{1}{m}\sum_{j=1}^{m}(PT_j - PT)^2}. \quad (6)$$

Response time is vital in analyzing a VM which determines the readiness of VM to accept the tasks. If the VM is idle or insufficient to process the task, then it does not respond to the cloud user within a desirable time. It is generally the amount of time taken between the submission of a request and the first response which is produced. For a VM j, the response time is denoted as RT_j while the average response time for n VMs is calculated as

$$RT = \sum_{j} \frac{RT_j}{n}. \quad (7)$$

Reliability can be described as the ability of a system or a component essential for performing under steady-state conditions for a specific time period. The reliability of a cloud VM is the measure of success of a task allocated to it. Factors affecting the reliability of VM are Overflow, Timeout, Data resource missing, computing resource missing, Software failure, Database failure, Hardware failure, and Network failure.

For a VM j, if the number of accepted tasks is At while the number of tasks completed is Ct , then the reliability R can be computed as the ratio of Ct to At

$$R = \frac{Ct}{At}. \quad (8)$$

Generally in any cloud service, the resources are said to be accessible or usable if and only if they are free to be utilized at the time needed. Availability is the degree of this usable VM such that when the degree of availability is low, then the VM is either unavailable for users or the VM and resources are in idle state. For a VM which has N tasks to be processed while if At tasks are completed within t time limit, the availability *Avail* is computed as

$$Avail = \frac{At}{N}. \quad (9)$$

Though there are many QoS and other parameters, the foremost one that keeps on haunting the users is the cost. Even if a VM is efficient in all means but not cost, is rejected by the users for affordable VMs. The VM performance is usually compared with the cost parameter to obtain the efficient performance at moderate or even cheaper price. Cost of a VM depends on the cost of one unit of CPU, RAM, network and bandwidth. If a VM is priced at p unit price (p=1), net for network units, data for bandwidth, RAM for memory units, a,b,c,d are weights for each resource attribute and a+b+c+d=1, then the cost of VM is calculated as

$$cost = \frac{p}{cpu^a * net^b * data^c * RAM^d}. \quad (10)$$

Throughput determines whether the estimated capacity of a VM is actually the same and whether it is capable of satisfying the user requests without exceeding the time limits. It is defined as the number of tasks completed by the cloud provider per unit of time. If n tasks are performed on m VMs at completion time CT and T_0 is the time overhead due to various factors such as infrastructure initiation delays and inter task communication delays, then $Throughput$ can be calculated as

$$Throughput = \frac{n}{CT(n,m) + T_0}. \quad (11)$$

Load balancing based on VM characteristics. After computing the load conditions, standard deviation and QoS parameters it is essential to sort the VMs in an ascending order, based on the better values. Initially, the system should decide whether load balancing should be taken up or not. This decision requires the following preliminary: finding whether the system is balanced and also if whether the whole system is overloaded or not. If the whole system is overloaded, then load balancing is not possible. The VMs

then sorted based on the QoS parameters. However, the use of multiple QoS parameters as in our work causes multi-objective optimization problems. Since a number of QoS parameters are involved simple techniques such as crowding distance or weight estimation cannot be employed to find the pareto-optimal solutions for the multi-objective optimization problem. Hence, the Fuzzification process is performed on these QoS values. Using the fuzzy rules generated for the QoS parameters considering the best values for a VM, the VMs are ranked in an order of minimum load and better QoS values with priority are given to load. The fuzzification process generates an affordable solution for calculating the efficiency of each VM in-terms of all QoS parameters without needing to sacrifice performance efficiency. If computes all the parameters to be treated equally in ranking the VM.

When a VM is overloaded and the decision is taken to balance the load, the scheduler triggers the load balancing aspect. The overloaded VMs, load requirement, low-loaded VMs and available load can be determined by analyzing the capacity and the load of VM, along with the supply and demand at time t. Supply S and demand D can be calculated as follows:

$$S = Maximum\,capacity - \frac{L}{C}. \tag{12}$$

$$D = \frac{L}{C} - Maximum\,capacity. \tag{13}$$

The tasks are then removed from the overloaded VMs and the priority of the tasks is found. The task priority can be ranked as follows-: high, medium and low. The tasks are set as scout bees and forager bees and depending on the analysis of scout bee (earlier removed task), the suitable low loaded VMs with better QoS values are found to be effective for the forager bee (current task). The forager bee then becomes the scout bee for the next task and the whole process continues until complete load balancing task is achieved.

Step 1: Initialization

VM initialization $\mathrm{VM} = \{\mathrm{VM}_1, \mathrm{VM}_2, \ldots . \mathrm{VM}_\mathrm{m}\}$

Input Tasks $\mathrm{T} = \{\mathrm{T}_1, \mathrm{T}_2, \ldots . . \mathrm{T}_\mathrm{n}\}$

Step 2: Estimate VM characteristics

For each VM

- Compute M, C, $L_{V,t}$, $\mathrm{PT_j}$& σ
- Set threshold condition set (Ts [0-1]) for VM

//Ts based on M, C, $L_{V,t}$, $\mathrm{PT_j}$& σ

- Check condition for balance VM $\leq$ Ts
- Classify VMs (Overload, Under-load, Normal)
- Check Possibility of balancing

If $L > max\ capacity$
Load balancing aborted
Else
Balancing process begins
End if

End for

Step 3: Estimate QoS parameters for VMs

- Compute *RT, R, Avail, Cost* & $Throughput$
- Fuzzification process
  Generate fuzzy rules
  Fuzzify QoS

Step 4: Load balancing process

For under-load VM

- Compute S
- Rank VM by descending S value
- Adjust VM rank based on Fuzzified QoS values

End for

For overload VM

- Compute D
- Rank VMs by ascending D value
- Adjust VM ranks based on Fuzzified QoS values

End for

- Rank Removed tasks based on priority T_h, T_m, T_l
- Finding suitable VM for each task T

For all tasks$Load_{VM} \leq Capacity_{VM}$; minRT; maxAvail; maxR; minCost; maxThroughput

- Special condition for T_h, T_m, T_l
- $T_h \rightarrow VM \,|\, \min\left(\sum T_h\right) \in VM$
- $T_m \rightarrow VM \,|\, \min\left(\sum T_h + \sum T_m\right) \in VM$
- $T_l \rightarrow VM \,|\, \min\left(\sum T\right) \in VM$

End for

Step 5: Update phase after each iteration

- Update tasks assigned to VM
- Update remaining priority tasks
- Update current load of VM
- Update VM class status (overload, under-load, normal)

Step 6: Termination condition

- If VM class non-empty
- Continue Load balance process
- Else

End the overall process

Listing 1. Algorithm: IHBB-LB

Description: In the above described algorithm, first the VM and tasks are initialized. Each task is constructed by means of several sub-tasks. All the tasks are arithmetic computation programs. For each VM, the makespan, load, processing time and standard deviation of load are estimated. Based on the load conditions, the VMs are classified into overload, under-load and normal load VMs. Then the VMs are checked for the possibility of initiating load balancing. Once the VM load balancing process is possible, the QoS parameters are estimated. As the consideration of different QoS parameters causes pareto-optimal problem, the values are fuzzified. Then the supply for under-load VMs and demand for overload VMs are estimated respectively and then the VMs are ranked. The ranking is re-adjusted based on the fuzzified QoS values. Each task is assigned priority and categorized into high, low and medium priority tasks. Then the tasks are allocated the VMs based on the load and QoS conditions. Finally the remaining tasks, load and VM class structure are updated for the iterative process. Thus the load is balanced efficiently using the proposed IHBB-LB algorithm.

4. Experimental Results. In this section, the performance of the proposed IHBB-LB is evaluated using the CloudSim tool. CloudSim is a library for the simulation of cloud scenarios providing essential classes for describing data centers, computational resources, virtual machines, applications, users, and policies for the management of various parts of the system such as scheduling and provisioning. This performance is compared with that of HBB-LB in terms of makespan, response time, imbalancing degree and number of tasks migrated. The number of VMs is initialized as 10 for the performance evaluation. The results obtained for the proposed IHBB-LB and the existing HBB-LB algorithms are compared in Table 1.

Table. 1. Performance Comparison of HBB-LB & IHBB-LB

No. of tasks	Makespan (in seconds)		Response time (in seconds)		Degree of Imbalance		Number of tasks migrated	
	HBB-LB	IHBB-LB	HBB-LB	IHBB-LB	HBB-LB	IHBB-LB	HBB-LB	IHBB-LB
10	32.576	31.207	12.476	11.107	1.47	1.43	-	-
15	36.378	34.408	16.278	14.308	1.30	1.22	2	1
20	38.576	35.957	18.476	15.857	1.29	1.24	3	1
25	40.424	38.011	20.324	17.911	1.406	1.34	4	2
30	42.253	39.526	22.153	19.426	1.35	1.30	4	3
35	44.115	40.058	24.015	19.958	1.42	1.38	5	2
40	56.915	48.770	36.815	28.670	1.45	1.41	6	3

From the Table 1 it can be seen that for the proposed IHBB-LB has better performance values than HBB-LB in terms of makespan, response time, degree of imbalance and number of tasks migrated. It can be proved that IHBB-

LB balances all the tasks to the VMs with less makespan while also takes less response time for accepting the user requests for the services based on tasks. The degree of imbalance denotes the imbalance in the VM load allocation which is considerably reduced in the IHBB-LB. similarly, the number of tasks migrated from a VM due to lack of resources or inability to handle the task is also minimum in the proposed IHBB-LB algorithm. The graphical comparisons of the evaluation results of HBB-LB and IHBB-LB in terms of the above mentioned performance metrics are given in the following sub-section.

Makespan: Makespan is defined as the overall time taken for task completion. It can be computed as in equation (1). Figure 2 shows the comparison of makespan in HBB-LB and IHBB-LB.

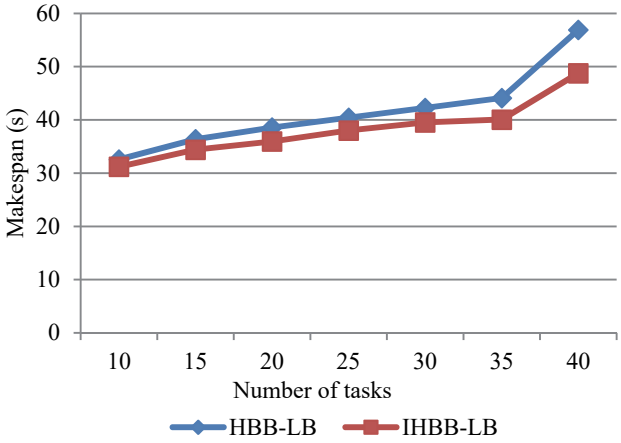

Fig. 2. Makespan

The number of VMs is initialized as 10. The number of tasks is plotted along x-axis while the makespan is plotted along y-axis. Considering the number of tasks as 40, HBB-LB has a makespan of 56.915s and IHBB-LB has a makespan of 48.77s. Thus, from the results it is clear that the proposed IHBB-LB provides better load balancing performance by minimizing the overall makespan of the tasks. The performance of HBB-LB and IHBB-LB in terms of response time is next evaluated and compared as follows:

Response Time: Response time is the amount of time taken between the submission of a request and the first response which is produced. Though makespan evaluates the performance efficiently, it is the time estimated between a task acceptance/allocation to output generation without considering the waiting time for VM response. Response time is computed as in equation (7). Figure 3 shows the comparison of response time in HBB-LB and IHBB-LB.

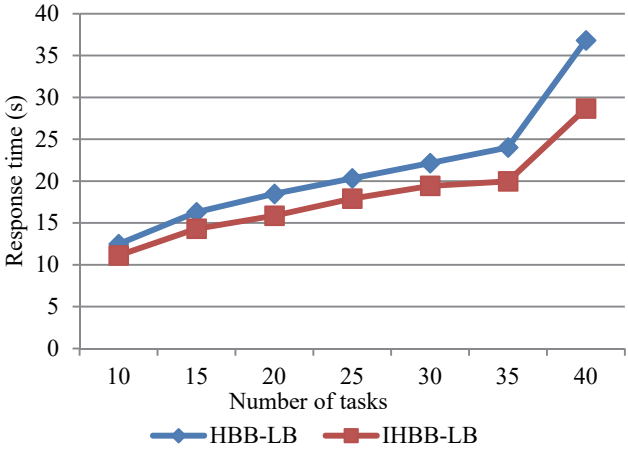

Fig. 3. Response Time

The number of tasks is plotted along x-axis while the response time is plotted along y-axis. Considering the number of tasks as 40, HBB-LB has response time of 36.815s and IHBB-LB has response time of 28.67s. Thus, from the results it is clear that the proposed IHBB-LB provides better load balancing performance. As the evaluation based on makespan and response time is efficient, the proposed scheme balancing rate has to be estimated. Inversely the degree of imbalance is evaluated as in the following sub-section.

Degree of Imbalance: The degree of imbalance is the imbalancing rate of the VM depending on the imbalance in the VM load. It can be calculated as follows

$$DI = \frac{T_{max} - T_{min}}{T_{avg}}, \tag{14}$$

where T_{max} and T_{min} are the maximum and minimum available tasks among all VMs, while T_{avg} is the average of tasks of VMs. Figure 4 shows the comparison of degree of imbalance in HBB-LB and IHBB-LB.

The number of tasks is plotted along x-axis while the DI is plotted along y-axis. Considering the number of tasks as 40, HBB-LB has DI of 1.45 and IHBB-LB has DI of 1.41. Thus, from the results it is clear that the proposed IHBB-LB provides better load balancing performance with less degree of imbalance. Degree of imbalance can also efficient when most tasks are balanced but there should be equal utilization of all VMs. If some VMs are incapable of handling a task, it is due to inefficient load conditions

while the task is migrated. Hence the efficiency of load balancing also depends upon number of migrated tasks.

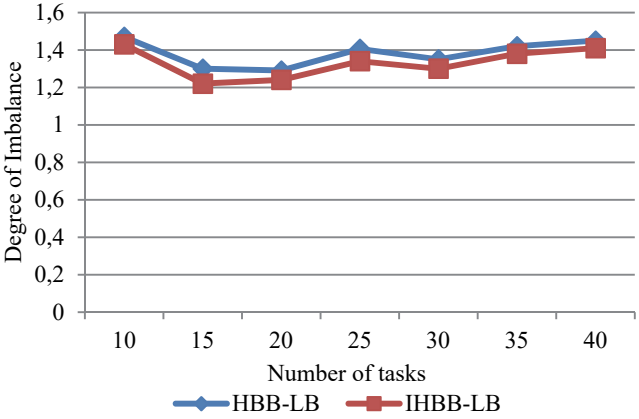

Fig. 4. Degree of Imbalance

Number of tasks migrated: The number of tasks migrated from a VM during load balancing determines the change in completion time. When more number of tasks is migrated, the VMs from which the tasks are migrated have unfavorable load conditions. This scenario must be minimized in order to provide efficient load balancing. Figure 5 shows the comparison of number of tasks migrated in HBB-LB and IHBB-LB.

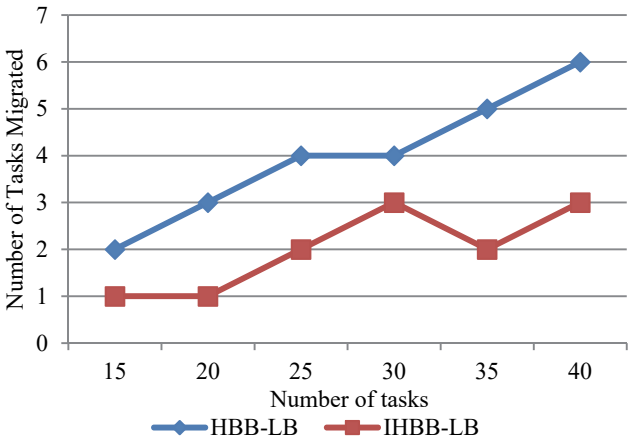

Fig. 5. Number of Tasks Migrated

The number of tasks is plotted along x-axis while the number of tasks migrated is plotted along y-axis. Considering the number of tasks as 40, in HBB-LB 6 tasks are migrated and in IHBB-LB 3 task is migrated. From the comparison results in terms of makespan, response time, degree of imbalance and number of tasks migrated it is clear that the proposed IHBB-LB provides better load balancing performance then HBB-LB. Thus it can be concluded that IHBB-LB can enhance the load balancing performance in cloud computing with high level of user satisfaction.

5. Conclusion. As described in the earlier section Improved Honey Bee Behavior based Load Balancing (IHBB-LB) approach to provide efficient load balancing has been proposed in this paper based on the consideration of multiple QoS parameters. The parameters such as service response time, availability, reliability, cost and throughput are included in IHBB-LB for balancing the load when some VM are overloaded and the remaining VM are under-loaded. This approach was found to perform better than the HBB-LB, which considers only the VM load conditions for the selecting VM for priority tasks. The performance evaluation results show that the proposed IHBB-LB approach more effective in terms of makespan, response time, imbalancing degree and number of migrated tasks which in turn seems to indicate that the IHBB-LB resolves the load balancing problem more effectively.

For more reliable load balancing, it is important for the VMs to successfully execute the tasks, the high priority tasks in particular which require adequate resources than other tasks. Thus, the resources for the VMs needed to be adaptively allocated as per the demands of application, which could be the research direction in future.

References

1. Mosleh M.A., Radhamani G., Hazber M.A., Hasan S.H. Adaptive Cost-Based Task Scheduling in Cloud Environment. *Scientific Programming*. 2016. 9 p.
2. Mondal B., Dasgupta K., Dutta P. Load balancing in cloud computing using stochastic hill climbing-a soft computing approach. *Procedia Technology*. 2012. vol. 4. pp. 783–789.
3. Boone B. et al. SALSA: QoS-aware load balancing for autonomous service brokering. *Journal of Systems and Software*. 2010. vol. 83. no. 3. pp.446–456.
4. Dasgupta K. et al. A genetic algorithm (ga) based load balancing strategy for cloud computing. *Procedia Technology*. 2013. vol. 10. pp. 340–347.
5. Keskinturk T., Yildirim M.B., Barut M. An ant colony optimization algorithm for load balancing in parallel machines with sequence-dependent setup times. *Computers & Operations Research*. 2012. vol. 39. no. 6. pp. 1225–1235.
6. Niknam T. et al. A modified honey bee mating optimization algorithm for multiobjective placement of renewable energy resources. *Applied Energy*. 2011. vol. 88. no. 12. pp. 4817–4830.
7. Dhinesh Babu L.D., Venkata Krishna P. Honey bee behavior inspired load balancing of tasks in cloud computing environments. *Applied Soft Computing*. 2013. vol. 13. no. 5. pp. 2292–2303.

8. Mohamed N., Al-Jaroodi J., Eid A. A dual-direction technique for fast file downloads with dynamic load balancing in the cloud. *Journal of Network and Computer Applications*. 2013. vol. 36. no. 4. pp. 1116–1130.
9. Zhao J. et al. A heuristic clustering-based task deployment approach for load balancing using Bayes theorem in cloud environment. *IEEE Transactions on Parallel and Distributed Systems*. 2016. vol. 27. no. 2. pp. 305–316.
10. Cho K.M., Tsai P.W., Tsai C.W., Yang C.S. A hybrid meta-heuristic algorithm for VM scheduling with load balancing in cloud computing. *Neural Computing and Applications*. 2015. vol. 26. no. 6.pp. 1297–309.
11. Xu G., Pang J., Fu X. A load balancing model based on cloud partitioning for the public cloud. *Tsinghua Science and Technology*. 2013. vol. 18. no. 1. pp. 34–39.
12. Xu Q. et al. Adaptive and scalable load balancing for metadata server cluster in cloud-scale file systems. *Frontiers of Computer Science*. 2015. vol. 9. no. 6. pp. 904–918.
13. Gutierrez-Garcia J.O., Ramirez-Nafarrate A. Agent-based load balancing in cloud data centers. *Cluster Computing*. 2015. vol. 18. no. 3. pp. 1041–1062.
14. Singh A., Juneja D., Malhotra M. Autonomous agent based load balancing algorithm in cloud computing. *Procedia Computer Science*. 2015. vol. 45. pp. 832–841.
15. Siar H., Kiani K., Chronopoulos A.T. An effective game theoretic static load balancing applied to distributed computing. *Cluster Computing*. 2015. vol. 18. no. 4. pp. 1609–1623.
16. Chen S.L., Chen Y.Y., Kuo S.H. CLB: A novel load balancing architecture and algorithm for cloud services. *Computers & Electrical Engineering*. 2017. vol. 58. pp. 154–160.
17. Liu Y., Zhang C., Li B., Niu J. DeMS: A hybrid scheme of task scheduling and load balancing in computing clusters. *Journal of Network and Computer Applications*. 2017. vol. 83. pp. 213–220.
18. Paya A., Marinescu D.C. Energy-aware load balancing and application scaling for the cloud ecosystem. *IEEE Transactions on Cloud Computing*. 2017. vol. 5. no. 1. pp. 15–27.
19. Wang Z. et al. Workload balancing and adaptive resource management for the swift storage system on cloud. *Future Generation Computer Systems*. 2015. vol. 51. pp. 120–131.
20. Randles M., Lamb D., Taleb-Bendiab A. Experiments with Honeybee Foraging Inspired Load Balancing. Proceedings of IEEE Second International Conference on Developments in E-Systems Engineering (DESE). 2009. pp. 240–247.
21. Pan J.S., Wang H., Zhao H., Tang L. Interaction artificial bee colony based load balance method in cloud computing. Proceedings of Genetic and Evolutionary Computing. 2015. pp. 49–57.

Mosleh Mohammed Abdullatef Saeed — Ph.D., research scholar of School of IT & Science, Dr. G.R. Damodaran College of Science. Research interests: Cloud Computing, Data mining, Networking. The number of publications — 4. ma.mosleh2010@gmail.com; Civil Aerodrome Post, Avinashi Road, Peelamedu, Coimbatore, Tamil Nadu 641014, India; office phone: 8122846484, Fax: 8122846484.

Radhamani Govindaraju — Ph.D., director of School of IT & Science, Dr. G.R. Damodaran College of Science, professor of School of IT & Science, Dr. G.R. Damodaran College of Science. Research interests: Internet of things, cloud computing, computer security, databases and mobile computing. The number of publications — 28. radhamani@grd.edu.in; Civil Aerodrome Post, Avinashi Road,Peelamedu, Coimbatore, Tamil Nadu 641014, India; office phone: +91 422-2572719.

УДК 004 **DOI** 10.15622/sp.57.2

М.А.С МОСЛЕХ, Г. РАДХАМАНИ

БАЛАНСИРОВКА ЗАГРУЖЕННОСТИ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ НА ОСНОВЕ УЛУЧШЕННОГО АЛГОРИТМА ПОВЕДЕНИЯ ПЧЕЛИНОЙ КОЛОНИИ

Мослех М.А.С, Радхамани Г. **Балансировка загруженности облачных вычислений на основе улучшенного алгоритма поведения пчелиной колонии.**

Аннотация. На данный момент применение алгоритма балансировки нагрузки задач на виртуальных машинах представляет большой исследовательский интерес. Для балансировки нагрузки с максимальной пропускной способностью была введена балансировка нагрузки на основе поведения медоносных пчел в колонии — Honey Bee Behavior Based Load Balancing (HBB-LB). Этот подход также устанавливает приоритеты выполнения задач на виртуальной машине с целью минимизации времени ожидания задач. Однако он рассматривает только один параметр — нагрузку виртуальных машин, что может оказаться недостаточно эффективным для балансировки. В работе предлагается улучшенный подход к балансировке нагрузки на основе пчелиного поведения, в котором дополнительно учитываются такие параметры качества обслуживания (QoS) виртуальных машин, как время отклика службы, доступность, надежность, стоимость и пропускная способность для улучшения балансировки нагрузки. Время отклика является критически важным для определения мгновенной активности виртуальной машины, доступность определяет доступный ресурс и состояние виртуальной машины (пассивное или активное), а надежность определяет уровень доверия к виртуальной машине. Затраты на использование виртуальной машины и пропускная способность виртуальных машин также необходимы для определения их эффективности. Однако включение нескольких параметров качества обслуживания приводит к многоцелевой оптимизации. По мере вычисления нескольких параметров фаззификация значений качества обслуживания выполнялась с помощью генерируемых нечетких правил, и была устранена проблема многоцелевой оптимизации. Эксперименты проводились с точки зрения времени разрешения задач, времени отклика, степени дисбаланса и количества перенесенных задач, а результаты показывают, что балансировка нагрузки на основе пчелиного поведения обеспечивает лучший уровень производительности.

Ключевые слова: оптимизация, параметры качества обслуживания, облачные вычисления, балансировка нагрузки, фаззификация.

Мослех Мохамед Абдель Латиф Саид — Ph.D., научный сотрудник школы информационных-технологий и науки, Научный колледж им. доктора Г.Р. Дамодарана. Область научных интересов: облачные вычисления, интеллектуальный анализ данных, networking. Число научных публикаций — 4. ma.mosleh2010@gmail.com; Пост гражданского аэродрома, Авинаши роуд, Пиламеду, Коимбатур, Тамилнад, 641014, Индия; р.т.: 8122846484, Факс: 8122846484.

Радхамани Говиндараджу — Ph.D., директор школы информационных-технологий и науки, Научный колледж им. доктора Г.Р. Дамодарана, профессор школы информационных-технологий и науки, Научный колледж им. доктора Г.Р. Дамодарана. Область научных интересов: интернет вещей, облачные вычисления, компьютерная безопасность, базы данных и мобильные вычисления. Число научных публикаций — 28. radhamani@grd.edu.in; Пост гражданского аэродрома, Авинаши роуд, Пиламеду, Коимбатур, Тамилнад, 641014, Индия; р.т.: +91 422-2572719.

Литература

1. *Mosleh M.A., Radhamani G., Hazber M.A., Hasan S.H.* Adaptive Cost-Based Task Scheduling in Cloud Environment // Scientific Programming. 2016. 9 p.
2. *Mondal B., Dasgupta K., Dutta P.* Load balancing in cloud computing using stochastic hill climbing-a soft computing approach // Procedia Technology. 2012. vol. 4. pp. 783–789.
3. *Boone B. et al.* SALSA: QoS-aware load balancing for autonomous service brokering // Journal of Systems and Software. 2010. vol. 83. no. 3. pp.446–456.
4. *Dasgupta K. et al.* A genetic algorithm (ga) based load balancing strategy for cloud computing // Procedia Technology. 2013. vol. 10. pp. 340–347.
5. *Keskinturk T., Yildirim M.B., Barut M.* An ant colony optimization algorithm for load balancing in parallel machines with sequence-dependent setup times // Computers & Operations Research. 2012. vol. 39. no. 6. pp. 1225–1235.
6. *Niknam T. et al.* A modified honey bee mating optimization algorithm for multiobjective placement of renewable energy resources // Applied Energy. 2011. vol. 88. no. 12. pp. 4817–4830.
7. *Dhinesh Babu L.D., Venkata Krishna P.* Honey bee behavior inspired load balancing of tasks in cloud computing environments // Applied Soft Computing. 2013. vol. 13. no. 5. pp. 2292–2303.
8. *Mohamed N., Al-Jaroodi J., Eid A.* A dual-direction technique for fast file downloads with dynamic load balancing in the cloud // Journal of Network and Computer Applications. 2013. vol. 36. no. 4. pp. 1116–1130.
9. *Zhao J. et al.* A heuristic clustering-based task deployment approach for load balancing using Bayes theorem in cloud environment // IEEE Transactions on Parallel and Distributed Systems. 2016. vol. 27. no. 2. pp. 305–316.
10. *Cho K.M., Tsai P.W., Tsai C.W., Yang C.S.* A hybrid meta-heuristic algorithm for VM scheduling with load balancing in cloud computing // Neural Computing and Applications. 2015. vol. 26. no. 6.pp. 1297–309.
11. *Xu G., Pang J., Fu X.* A load balancing model based on cloud partitioning for the public cloud // Tsinghua Science and Technology. 2013. vol. 18. no. 1. pp. 34–39.
12. *Xu Q. et al.* Adaptive and scalable load balancing for metadata server cluster in cloud-scale file systems // Frontiers of Computer Science. 2015. vol. 9. no. 6. pp. 904–918.
13. *Gutierrez-Garcia J.O., Ramirez-Nafarrate A.* Agent-based load balancing in cloud data centers // Cluster Computing. 2015. vol. 18. no. 3. pp. 1041–1062.
14. *Singh A., Juneja D., Malhotra M.* Autonomous agent based load balancing algorithm in cloud computing // Procedia Computer Science. 2015. vol. 45. pp. 832–841.
15. *Siar H., Kiani K., Chronopoulos A.T.* An effective game theoretic static load balancing applied to distributed computing // Cluster Computing. 2015. vol. 18. no. 4. pp. 1609–1623.
16. *Chen S.L., Chen Y.Y., Kuo S.H.* CLB: A novel load balancing architecture and algorithm for cloud services // Computers & Electrical Engineering. 2017. vol. 58. pp. 154–160.
17. *Liu Y., Zhang C., Li B., Niu J.* DeMS: A hybrid scheme of task scheduling and load balancing in computing clusters // Journal of Network and Computer Applications. 2017. vol. 83. pp. 213–220.
18. *Paya A., Marinescu D.C.* Energy-aware load balancing and application scaling for the cloud ecosystem // IEEE Transactions on Cloud Computing. 2017. vol. 5. no. 1. pp. 15–27.
19. *Wang Z. et al.* Workload balancing and adaptive resource management for the swift storage system on cloud // Future Generation Computer Systems. 2015. vol. 51. pp. 120–131.
20. *Randles M., Lamb D., Taleb-Bendiab A.* Experiments with Honeybee Foraging Inspired Load Balancing // Proceedings of IEEE Second International Conference on Developments in E-Systems Engineering (DESE). 2009. pp. 240–247.
21. *Pan J.S., Wang H., Zhao H., Tang L.* Interaction artificial bee colony based load balance method in cloud computing // Proceedings of Genetic and Evolutionary Computing. 2015. pp. 49–57.