Asian Journal of Information Technology 15 (9): 1425-1430, 2016
ISSN: 1682-3915

A Novel Improved Honey Bee Based Load Balancing Technique in Cloud Computing Environment

Geethu Gopinath and Shriram K Vasudevan
Deptartment of Computer Science and Engineering, Amrita School of Engineering,
Amrita Vishwa Vidyapeetham University, Coimbatore, India

Abstract: To come up with an effective load balancing algorithm in cloud environment which provides sharing the available resources systematically across the network. In this research, an optimized dynamic algorithm is proposed where it makes use of the concept of Honeybee algorithm which is categorized under the dynamic load balancing category. The proposed algorithm provides better makespan which is one of the main features in load balancing.The proposed algorithm is simulated using two simulators namely cloudsim and workflow. Somewhere, it compares both the dependent as well as independent nature of tasks. The main goal of load balancing algorithms in clouds is to reduce the makespan. Here, the algorithm performs with a reduced makespan for both type of tasks. Due to the heterogeneity behaviour of cloud, dynamic algorithms work better when compared to the static methods. The proposed algorithm is dynamic in nature and provides reduced makespan and it uses the resources efficiently. Our future research is based on proposing algorithm considering the other QS factorsof load balancing for both in dependent and independent nature of tasks.

Key words: Load balancing, cloud, honey bee algorithm, makespan, processing time, work_load, load migration, capacity, threshold value

INTRODUCTION

Load balancing is regarded as a process of redistributing the entire accessible workload to each and every individual node so that the entire system is well balanced. As the name suggests, it is to balance the load properly and more efficiently within the nodes available. The increase in the number of the users and their urge in resources lead to cloud computing technology and finally to the efficient load balancing Algorithms. Many load balancing algorithms are put forward to make a better system but due to the network bottlenecks and bandwidth issues balancing of load results in so many complications and is never an easy job. A balanced cloud environment scenario is considered to be an NP-complete problem (Braun *et al*., 2001). Generally, the task submitted to the cloud scheduler is distributed to the various unassigned nodes in the system. But to assign the tasks to the correct node effectively in cloud environment is certainly a daunting task. The difficulty arise due to the high heterogeneity in cloud framework, operating system, cloud providers and the resource consumers (Kokilavani and Amalarethinam, 2011). The intention of any load balancing algorithm should be to reduce the makespan which results in effective resource utilization that leads to the user's utter most satisfaction. But to find such an algorithm which is capable of satisfying all the goals is tiresome. Traditional algorithms like Min-min and Max-min (Kokilavani and Amalarethinam, 2011) doesn't adapt to the cloud environment and they fall under static load balancing algorithms. The improved version of Max-min (Kokilavani and Amalarethinam, 2011) was proposed provides better makespan than the traditional one but due to the heterogeneity in cloud it results in reduced resource utilization. Load balancing algorithms should be well planned and efficient for the proper balancing of the load among the available nodes in the network. A proper Algorithm is needed for the efficient resource utilization, makespan. In this proposed algorithm, it reaches the goal of reduced makespan. The algorithm scans the node before allocating it with the load.

Literature review: Load balancing (Singh and Hemalatha, 2012) can be regarded as a process of allocating unassigned resources to the available nodes present in the system such that the nodes shares almost equal amount of workload, judiciously. Balancing algorithms are mainly classified in to two types (Moharana *et al*., 2013) as static and dynamic load balancing algorithms. In static, the algorithm does not focus on the past behavior of the

Corresponding Author: Geethu Gopinath, Deptartment of Computer Science and Engineering, Amrita School of Engineering, Amrita Vishwa Vidyapeetham University, Coimbatore, India

node during the process of load distribution. On the contrary, dynamic algorithm keeps in mind the past behavior of the node during load distribution. During the load distribution, they take in to account the present state of the node (Malarvizhi and Uthariaraj, 2009). Static algorithms are better than dynamic with respect to its simplicity, since in static there is no need for the continuous scanning of the nodes where it focus only the current behavior. Static algorithms work better if the task is homogenous in nature which means that the load variation is negligible. Dynamic algorithms provide better results when compared with static algorithms in cloud scenario. Due to the heterogeneity in cloud dynamic algorithms are better than static.

The proposed algorithm is mainly inspired from the traditional honey bee foraging algorithm (Vries and Biesmeijer, 1998). Virtual machine scheduling management using artificial bee colony in cloud computing scenario was proposed in study (Kruekaew and Kimpan, 2014) .The study deals with the scheduling of the virtual machines in the dynamic behavior of cloud environment using the artificial bee colony optimization algorithm. Karaboga in study (Vries and Biesmeijer, 1998; Karaboga and Basturk, 2007; Karaboga and Basturk, 2008) put forward an optimization algorithm which is inspired from the foraging activity of honeybee. In 2006, a search algorithm based on the bees foraging strategy was suggested by Pham *et al.* (2011). The mentioned study focus on a new optimization algorithm that collectively includes a neighbor search along with random search which helps in finding an optimal value. The honey bee algorithm has been using in many fields such as Karaboga and Cetinkaya (2011) used this idea for adaptive filtering in digital processing area. Kang *et al.* (2009) got inspired from the foraging behavior and used this idea in inverse analysis. For motion estimation using a block matching algorithm was recommended by Cuevas *et al.* (2013). A improved algorithm was proposed based on the honey bee forging strategy which results in better results than the traditional one. Reduced makespan is obtained in the honey bee inspired algorithm (LD and Krishna, 2013). The proposed algorithm is mainly taking the idea from the HBB-LB which is presented in this study.

MATERIALS AND METHODS

Honey bee foraging algorithm: Honey bee foraging algorithm is one of the traditional dynamic load balancing techniques that exists in the cloud computing environment. Honey bee algorithm is one among the decentralized load balancing techniques that mainly focus to achieve load balancing among the heterogeneous nodes in the cloud computing platform that results in maximizing the overall throughput of the system. Dynamic algorithms mainly focus on the current behavior, here the load over the desired node is calculated and categorized it as overloaded, under loaded and balanced. The node is categorized based on the workload it holds. The task can be migrated to the under loaded node depending upon its priority. Each task that is in the waiting queue is processed based upon the priorities. The lightly loaded node is found out based on the previously removed tasks on that particular node. Honey bee algorithm is actually a nature inspired algorithm that follows the honey bee food finding strategy. In honey bee food finding method the bees will travel a long distance to find the better flower patches just like in cloud environment searching for an under loaded node. After finding the best flower patch the honey bee comes back to their nest and performs a 'waggle dance' which provides information to other bees. In contrary in cloud scenario the number of under loaded nodes is published through a random search. The quality and quantity of the food depends upon the duration of the dance performed by the bees. For the best flower patches more worker bees will get allotted. In this fashion for the node which is more lightly loaded will get assigned with more tasks compared with others. The process will get repeated till either the bee hives get filled or the worker bees are fully busy. Honey bee food finding strategy is selected to balance the overall workload in the cloud computing scenario.

Honey bee inspired load balancing technique: In honey bee inspired load balancing technique, it takes the idea from the food finding behavior of honey bees. The algorithm takes in to consideration about the priorities of jobs that are to be allocated to the under loaded virtual machine. While taking care of the priorities it will results in providing services to the user who needs their work to be done at almost high speed. Setting the priorities to the tasks leads to higher user satisfaction and reducing the waiting time of the tasks. The algorithm improves the overall throughput of processing. The algorithm performs better for non-preemptive independent tasks.

Improved HBB-LB: The improved HBB-LB is mainly inspired from the honey bee behavior inspired load balancing of tasks in cloud computing environments by LD and Krishna (2013). The proposed algorithm provides better makespan than the above one.

In the cloud computing scenario user request will come in random fashion at different time intervals and from different locations. The request are forwarded to the corresponding servers depending upon so many factors

like load on the node, closeness to the client, priority of the request, etc. The scheduling policies in cloud will results in different amount of workloads in each and every node present in the system. A balanced environment is disturbed and an efficient load balancing algorithm is required for reduced makespan and resource usage. Due to the dynamic behavior the dynamic load balancing algorithms performs much better than the static one in the cloud scenario. The proposed algorithm shows a dynamic behavior and comparing with other existing algorithms it performs better. The main aim of the proposed algorithm is to reduce the makespan. Makespan can be explained as the total task completion time.

Let, task = $\{T_1, T_2, T_3..., T_n\}$ be the set of 'n' tasks in a system of nodes = $\{N_1, N_2, N_3..., N_j\}$ of 'j' number of nodes. And T_x be the finishing time for any task where x = 1, 2, ..., n. Then, the makespan can be defined as:

$$\text{Makespan} = \text{max\_value}$$

Where:
x = 1, 2, 3, ..., n
k = 1, 2, 3,..., j

The proposed algorithm, improved HBB-LB is a dynamic behavior that is an extension of the work done in study (LD and Krishna, 2013) by making change in the capacity equation and load. Capacity is given a great importance where the processing time for each individual node as well as the migration time depends upon the capacity of that particular node. Processing time for any task 'T'_x' can be denoted as '$Proc_{xk}$' where, k = 1, 2, 3, ..., j. The overall processing time for all the tasks in a single node can be provided by the Eq. 1:

$$Proc_j = \sum_{i=1}^{n} Proc_{lk} \quad (1)$$

Where, k = 1, 2, 3,...., j. The algorithm also focuses in providing priorities to the tasks that are removed from the overloaded node. The tasks which are removed are considered as worker bees (LD and Krishna, 2013).

The priorities as well as the workload of each task will get updated during the submission of these tasks to the under loaded node. A full scanning of the entire nodes in the system will be done before submission of the tasks to any particular nodes. Setting the priorities will help in allocating tasks to the correct node. The node having the less number of high priority tasks will get assigned with any of the removed task. This will help in balancing the system as well as providing high user satisfaction. Capacity of each node is calculated using the formula. Here, virtual machine is treated as node and the CPU is denoting as processor. Capacity = MIPS+memory space allocated to a node+CPU allocated to a node, i.e., capacity of node 'j' is denoted as '$Capacity_j$' then:

$$Capacity_j = Proc_{mips(j)} + CPU_{num(j)} + Space_{mem(j)}$$

Where, '$Proc_{mips(j)}$' denotes the total million instruction per second for all the available number of 'CPU's' in the node 'j'. The $CPU_{num(j)}$ denotes the number of available 'CPU's' in the node 'j'. The $Space_{mem(j)}$' denotes the memory allocated to that particular node. The overall capacity in Eq. 2:

$$Capacity = \sum_{i=1}^{n} Capacity_{(i)} \quad (2)$$

Where, $Capacity_{(i)}$ denotes the individual node capacity. The load equation proposed in study (LD and Krishna, 2013) is as shown below:

$$Load_{vm_i,t} = \frac{N(T,t)}{s(vm_i,t)} \quad (3)$$

Where, load on a node is calculated by the total number of tasks that a node can hold at time 't' divided by the service rate of a node at time 't'. Load can be defined as the total workload that are assigned to a virtual machine in a cloud scenario. The revised load equation is given by Eq. 4:

$$Load_{(vm)} = N(tasks_{((t))}) + CPU_{utilized_{(t)}} + mem_{utilized_{(t)}} \quad (4)$$

Total load of all virtual machine in a system is given by Eq. 5:

$$Load_{(total)} = \sum_{i=1}^{n} load_{vm_{(i)}} \quad (5)$$

The processing time of a node is calculated as Eq. 6:

$$Process_{time_{vm(1)}} = \frac{load_{vm_{(1)}}}{capacity_{vm_{(1)}}} \quad (6)$$

The total processing time is calculated as Eq. 7:

$$Process\_time = \frac{load}{capacity} \quad (7)$$

Based on the equations provided above information about the load and capacity of all the nodes in a system can be obtained. The collected information is capable of finding out the standard deviation that will give an idea

about the amount of deviation in workload for all the available nodes in the system. The deviation value categorizes the node in to overloaded, under loaded or balanced one.

Migration of the tasks to the under loaded node is based upon the priorities of each tasks. All the nodes will be arranged in increasing order which helps in migrating the tasks to the under loaded node which will be positioned in the first place. The standard deviation is calculated using Eq. 8:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\text{Process}_{\text{time}_{vm(1)}} - \text{Process}_{\text{time}})^2} \qquad (8)$$

Load migration decision: The load information as well as the standard deviation helps the system to find out whether the whole scenario is balanced or not. For finding out if the system is balanced it has set two conditions: is to check whether the system is balanced or not; is to check the whole system is overloaded. If the whole system is overloaded then there is no meaning in checking the standard deviation since migration cannot be performed.

Steps in balancing the load

Threshold setting:

A threshold value (μ) is set up which is constant and the value is given as 0.2.The calculated standard deviation value (σ) is checked with the threshold value in Algorithm A. If the value is ≤ 0.2, then the system is said to be balanced otherwise the system will be in unbalanced state and the system will be in either of the two condition: the task has to be migrated to the under loaded nodes in the system to make the whole cloud scenario a balanced one; the whole system will be in overloaded state:

Threshold algorithm:

```
If (σ≤μ)
        Then the system is balanced
    Else if
        Migrate load to under loaded vm
    Else
        Exit
```

Virtual machine classification: The virtual machines are classified in to three categories based on the amount of load that it holds. The amount of load is obtained from the capacity equation. The virtual machines which are classified as overloaded, the tasks will be removed from them and will be migrated to any of the under loaded virtual machines. The balanced virtual machines are not taken in to the migration process.

Load migration: The tasks which are removed from the overloaded virtual machines will get migrated depending upon the priorities. The tasks will be put in its waiting queue depending upon the priorities. The load is calculated for each under loaded virtual machine. The node has which has less number of tasks assigned to it will get loaded with the high priority task that are waiting in the queue. The high priority task will always tries to find out the virtual machine that has got less number of tasks. The process continues till the system becomes balanced.

RESULTS AND DISCUSSION

Simulation overview: In this module, the simulation of the proposed algorithm is compared with the existing honey bee behavior inspired algorithm. To evaluate the performance of the algorithms one of the measures is its makespan. The makespan can be defined as the time difference between the last task and the first task.

For the simulation purpose we have used two simulators namely cloudsim and workflowsim (Chen and Deelman, 2012). Cloudsim is a simulating toolkit that is used for simulating the cloud environment where the tasks are independent in nature.workflowsim is again a simulator which is an extension of the cloudsim providing a workflow management (Hoffa *et al*., 2008). The workflowsim is used for the tasks that are dependent in nature. In both the simulators cloudlets is the term used for tasks and nodes of the system as the virtual machines.

The simulation starts by keeping the node count to be five and the cloudlet count varying from 25, 50,100 and 1000. All the four cases are simulated for both dependent as well as independent tasks. All the five resources are located in a single datacenter.

Simulation result with cloudsim: From Table 1, we can note that the makespan of the improved HBB-LB is better when compared with that of the existing algorithm. The assignment of tasks to the machines is not the same every time it gets changing depending upon the current status of the load in the vm and the balanced condition of the system.

For the three cases the performance of the two algorithms is shown in Fig. 1. The 'x' axis denotes the number of tasks and the makespan is denoted in 'y' axis.

Table 1: Makespan table for cloudsim

No. of tasks	HBB-LB	Improved HBB-LB
25	350	325
50	522	500
100	745	720

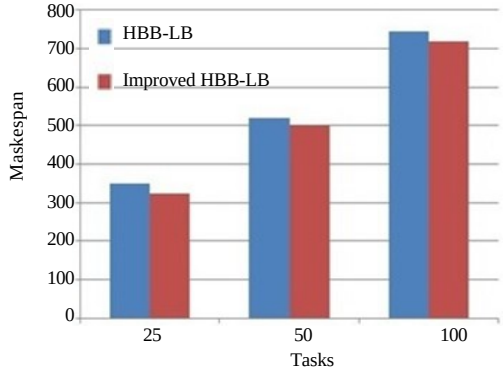

Fig. 1: Makespan bar graph for cloudsim

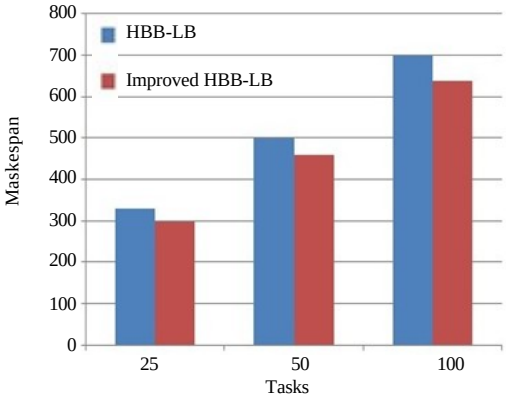

Fig. 2: Makespan bar for workflowsim

Table 2: Makespan for workflowsim

No. of tasks	HBB-LB	Improved HBB-LB
25	330	300
50	500	460
100	700	640

From the simulation result obtained for the independent nature of task set it explains that the proposed one has a reduced makespan compared with the existing one.

Simulation result with workflowsim: From Table 2, we can note that the makespan of the improved HBB-LB is better when compared with that of the existing algorithm. The nature of the task is that it is dependent on each other. The task is capable of migrating from one node to other during execution time.

The 'x' axis denoting the three cases that is 25, 50 and 100 tasks and the 'y' axis denote the makespan value (Fig. 2). The task set for the three cases is taken as Montage_25, Montage_50 and the Montage_100 xml files. The node count is kept constant to be five. The graph shows that the improved HBB-LB works better than the existing one for the dependent nature of task too.

CONCLUSION

In this research, we put forward an algorithm that was inspired from the existing work named, Honey bee behavior load balancing technique in the cloud scenario. The Algorithm takes the concept of the foraging food finding method of the honeybees. Due to the heterogeneity behavior of cloud dynamic algorithms work better when compared to the static methods. The proposed algorithm is dynamic in nature and provides reduced makespan and it uses the resources efficiently. The cloud system should be a stable one even in the changing scenario. The improved HBB-LB shows that it works better with reduced makespan for both dependent as well as independent tasks. The algorithm also focuses on the priorities of the task that has to be allocated to the virtual machines. The priority concept helps in reducing the response time of the virtual machine as well as it enhances the overall throughput also. Our future work is based on proposing algorithm considering the other QoS factors of load balancing for both in dependent and independent nature of tasks.

REFERENCES

Braun, T.D., H.J. Siegel, N. Beck, L.L. Boloni and M. Maheswaran *et al*., 2001. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. J. Parallel Distrib. Comput., 61: 810-837.

Chen, W. and E. Deelman, 2012. Workflowsim: A toolkit for simulating scientific workflows in distributed environments. Proceedings of the 2012 IEEE 8th International Conference on E-Science (e-Science), October 8-12, 2012, IEEE, Chicago, Illinois, USA., ISBN: 978-1-4673-4467-8, pp: 1-8.

Cuevas, E., D. Zaldivar, M.P. Cisneros, H. Sossa and V. Osuna, 2013. Block matching algorithm for motion estimation based on Artificial Bee Colony (ABC). Appl. Soft Comput., 13: 3047-3059.

Hoffa, C., G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman and J. Good, 2008. On the use of cloud computing for scientific workflows. Proceedings of the IEEE Fourth International Conference on eScience, December 7-12, 2008, Indianapolis, IN., USA., pp: 640-645.

Kang, F., J. Li and Q. Xu, 2009. Structural inverse analysis by hybrid simplex artificial bee colony algorithms. Comput. Struct., 87: 861-870.

Karaboga, D. and B. Basturk, 2007. Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems. In: Foundations of Fuzzy Logic and Soft Computing. Patricia, M., O. Castillo, L.T. Aguilar, J. Kacprzyk and W. Pedrycz (Eds.). Springer Berlin Heidelberg, Berlin, Germany, ISBN: 978-3-540-72917-4, pp: 789-798.

Karaboga, D. and B. Basturk, 2008. On the performance of Artificial Bee Colony (ABC) algorithm. Appl. Soft Comput., 8: 687-697.

Karaboga, N. and M.B. Cetinkaya, 2011. A novel and efficient algorithm for adaptive filtering: artificial bee colony algorithm. Turk. J. Electr. Eng. Comput. Sci., 19: 175-190.

Kokilavani, T. and D.I.G. Amalarethinam, 2011. Load balanced min-min algorithm for static meta-task scheduling in grid computing. Int. J. Comput. Appli., 20: 42-48.

Kruekaew, B. and W. Kimpan, 2014. Virtual machine scheduling management on cloud computing using artificial bee colony. Proceedings of the International MultiConference of Engineers and Computer Scientists 2014 Vol. I, IMECS 2014, March 12-14, 2014, MECS Publisher, Hong Kong, ISBN: 978-988-19252-5-1, pp: 12-14.

LD, D.B. and P.V. Krishna, 2013. Honey bee behavior inspired load balancing of tasks in cloud computing environments. Appl. Soft Comput., 13: 2292-2303.

Malarvizhi, N. and V.R. Uthariaraj, 2009. Hierarchical load balancing scheme for computational intensive jobs in Grid computing environment. Proceedings of the First International Conference on Advanced Computing, December 13-15, 2009, IEEE, Chennai, India, ISBN: 978-1-4244-4786-2, pp: 97-104.

Moharana, S.S., R.D. Ramesh and D. Powar, 2013. Analysis of load balancers in cloud computing. Int. J. Comput. Sci. Eng., 2: 101-108.

Pham, D.T., A. Ghanbarzadeh, E. Koc, S. Otri and S. Rahim *et al*., 2011. The bees algorithm-a novel tool for complex optimisation. Proceedings of the 2nd I* PROMS Virtual International Conference on Intelligent Production Machines and Systems, July 3-14, 2006, Elsevier, Oxford, England, UK., ISBN: 978-0-08-045157-2, pp: 442-454.

Singh, A. and M. Hemalatha, 2012. An approach on semi-distributed load balancing algorithm for cloud computing system. Intl. J. Comput. Appl., Vol. 56,

Vries, H.D. and J.C. Biesmeijer, 1998. Modelling collective foraging by means of individual behaviour rules in honey-bees. Behav. Ecol. Sociobiol., 44: 109-124.