ISSN:1991-8178

Australian Journal of Basic and Applied Sciences

Journal home page: www.ajbasweb.com

Optimized Load Balancing in Clouds Using Bee Colony Algorithm

M., Kaladevi, A.C., A.P. Meenakshi Sundaram, Guru Vimal Kumar

Department of Computer Science and Engineering, Sona College of Technology, Salem 636005, Tamil Nadu, India

ARTICLE INFO

Article history:
Received 12 November 2014
Received in revised form 26 December 2014
Accepted 29 January 2015
Available online 10 February 2015

Keywords:
Load balancing, Cloud computing, Bee Colony Optimization, Virtual Machines.

ABSTRACT

Cloud computing portends a major change in how we store information and run applications. Instead of running programs and data on an individual desktop computer, everything is hosted in the "cloud". Task balancing in Virtual Machines (VMs) is one of the important aspects in cloud environment. Whenever certain VMs are overloaded and remaining VMs are under loaded for processing the task, the load has to be balanced to achieve optimal machine utilization. In this paper, we use an algorithm named Bee colony optimization, which aims to achieve well balanced load across virtual machines for maximizing the throughput. The algorithm Agglomerative hierarchical clustering works by grouping the data one by one on the basis of the nearest distance measure of all the pair wise distance between the data point. We use two main parameters as distance and capacity to achieve the minimum response time. In our approach, there is a significant improvement in average execution time and reduces the response time.

To Cite This Article: Guru Vimal Kumar, M., Kaladevi, A.C., A.P. Meenakshi Sundaram, Optimized Load Balancing in Clouds Using Bee Colony Algorithm. ***Aust. J. Basic & Appl. Sci.,*** *9(6): 16-19, 2015*

INTRODUCTION

Cloud computing enables sharing of hardware, software and applications according to the requirements of the clients within a minimum time. Load balancing in cloud computing systems is one of the challenging tasks. The distributed solution is required because it is cost efficient for maintaining more idle services and just to fulfil the requirement needs (Kaladevi, 2012). Tasks are not assigned to appropriate servers and components are to be presented throughout a wide spread area. The overall response time should be reduced and this is expected by the client. Virtual Machine is one of the processing units in cloud computing environment. From the business point of view, the VMs should run in parallel. This creates problem in scheduling the tasks. If the tasks are not scheduled properly, the resource utilization will not be efficient. One VM gets high number of tasks and another VM gets low number of tasks. Hence a balanced way of task scheduling should be done among the virtual machines. We therefore use a Bee Colony Scheduler to balance the loads across the VMs. The aim of load balancing is to increase the execution time. Load balancing is of two types (Eager, 1986). We use clustering algorithms for finding the nearest neighbour distance. The rest of the paper is organised as follows: section 2 describes about the related work. Section 3 deals with the Bee's behaviour in nature and how it related to our proposed work. Section 4 mainly deals with the clustering algorithm used for finding nearest neighbour, which is the proposed system and this section also deals with the experimental results and comparisons of previous work calculations. The conclusion and the future enhancement is given in section 5.

Related Work:

In paper (Yagoubi, 2007), the tree representation of grid computing is explained. The tree representation of grid will balances the load but it takes more time for executing a task. In paper (Yagoubi, 2006) dynamic algorithm is applied on servers. It is done by searching the whole network. The server which has low capacity is preferred for balancing the traffic. The research paper (Yagoubi, 2007), explains about the load balancing in grid computing by dynamic algorithms. It takes consideration of previous reference of load balancing is used and the transfer of tasks locally for communication cost reduction is done. Game theory has non-cooperative games and cooperative games are stated in paper (Gaochao, 2013). Here the allocation of resource takes place by noting the idle and normal state. In paper (Randles, 2009), the natural phenomenon on behaviour is investigated and got the solution from the bees' behaviour. Biased

Corresponding Author: Guru Vimal Kumar, M., Department of Computer Science and Engineering, Sona College of Technology, Salem 636005, TamilNadu, India

random sampling method is used in distributed systems. It will examine the global mean measure by maintaining individual node. Load Balancing is done by grouping the server in virtual server. The service queue maintains all the server information and profits will be calculated and the quality of bees is expressed by waggle dance. In paper (Randles, 2010), M.Randles and D.Lamb, 2010, proposed the distributed load balancing algorithm in comparative study for cloud computing. Here three methods for potentially viable are inspired which is used by self organization. Then self organization can be randomly sampled. The node of all system will be balanced equally. The assigning of jobs/tasks is done by some scheduling algorithm. Load balancing provides the minimum response time for maximizing the throughput. In the distributed system, the communication cost is reduced by doing the load balancing. It also reduces the traffic. A policy based load balancing algorithm is explained in paper (Karatza, 1994). In paper (Wang, 2011), cloud computing uses low power system for the high usage. The computer resources is utilized on the network by cloud computing. Thus cloud computing should consider the node selection for task execution and to maximize the response time of the tasks.

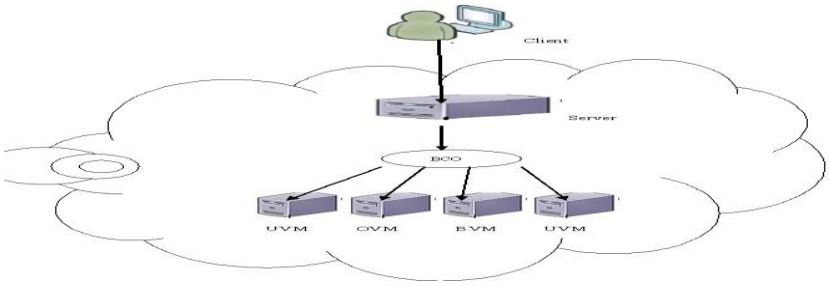

Fig.1: Architecture of Bee Colony based Load Balancing in Cloud.

Existing System:

Foraging Behavior of Bees:

The Honey Bee Foraging Behaviour as the name implies depicts the nature of honey bees to find and reap food forms the core to obtain behaviour of this algorithm. Dance of waggle is a kind of dance by forager bees to advertise their victory in foraging for food sources (Dhinesh Babu, 2013). The foraging behaviour denotes the quality/quantity/distance. These can be identified by the way of dance. These will be identified by forager bees. These then follow the scout bees for food located area and then start to reap it. Again they go to beehive and do the waggle dance which indicates the same as before, about the food left details will be indicating by waggle dance. The profit calculation is done on the queues in server. The request is to be processed based on user demands. After the creation of the virtual machines in cloud, the load balancing starts: when a job arrives at the system, the main server decides which Virtual machine should receive the job. The tasks in the overloaded VM have to be assigned to the under loaded VM. We use bee-colony optimization for sorting the tasks based on ascending and descending order. When the load status of a VM is balanced, this VM can be accomplished locally. OVM: Overloaded VM's, UVM: Under loaded VM's, BVM: Balanced VM's. If the VM load status is not balanced, this job should be transferred to another VM. The whole process is shown in Fig.1.

Bees algorithm:

1. Initialize population with random search.
2. Evaluate the fitness of the population from the random search.
3. While forming a new population. Identify the task, which is located nearest
is based on capacity and distance.
4. Selected tasks are to be recruited and fitness is evaluated.
5. Fittest bee is to be identified from each patch.
6. Then tasks of remaining bees are to be assigned for random search and
fitness is evaluated.
7. End the task.

Sort VMs in UVM by ascending order of capacity and sort the VMs in OVM
by descending order of capacity.

Proposed system:

Hierarchical Agglomerative Clustering (AGNES):

Each cluster will generate a priority queue of the distances to all the other clusters in the cloud. Determine the closer of two clusters. Agglomerate the clusters accordingly. Find the nearest neighbour for newly agglomerated cluster. Update the priority queue of only nearest neighbour. Nearest neighbour priority queue is scanned to determine the cluster nearest to it. If newly agglomerated machine is closer to nearest neighbour machine then it is agglomerated with it otherwise it is agglomerated with machine closer to it. Moreover, if a clustering process terminates, and the distance between nearest clusters exceeds an arbitrary threshold, then this is called a single linkage algorithm. AGNES places the each object into a cluster by own. The clusters are then merged step-by-step together according to nearest

neighbouring criteria. Thus, an agglomerative clustering algorithm uses the minimum distance measure is also called as a minimal spanning tree algorithm.

Selection of Optimized VM:

Measures for distance between machines are as denoted, where |***p***-***p***'| is the distance between two points, ***p*** and ***p***'; ***min*** is the mean for cluster *Ci* and *ni* is the number of points in *Ci*.

$$dmin\ (Cj, Ci) = min_{p \in Ci,\ p' \in Cj} |p-p'| \quad (1)$$

The cluster centre is the centroid of the points. The distance between two cluster center points is the increase in the sum of squared distances from each centre points of its cluster caused by the cluster.

$$H(J*I) = 1/|J|\ |I| \sum_{i \in I} (e_{ij}-e_{iJ}-e_{Ij}-e_{IJ})^2 \quad (2)$$

Where i= Distance
j= Capacity
I= Row
J= Column
From the above equation 2, we will get these

$$e_i = 1/|J| \sum_{j \in J} (e_{ij}) \quad (2.1)$$

$$e_j = 1/|I| \sum_{i \in I} (e_{IJ}) \quad (2.2)$$

Residue:

$$I*J = 1/|I|\ |J|\ (e_i e_j) \quad (2.3)$$

The above formula is used to calculate the distance for 2units and substituted for the capacities 5 , 6, 4, and 8 respectively and presented in table 1

Table 1: Calculation of Nearest neighbour with its capacity using del cluster

Dist/capacity	5	6	4	8
2	2.876	2.889	2.8695	2.8901
2	2.876	2.889	2.8695	2.8901
2	2.876	2.889	2.8695	2.8901
2	2.876	2.889	2.8695	2.8901

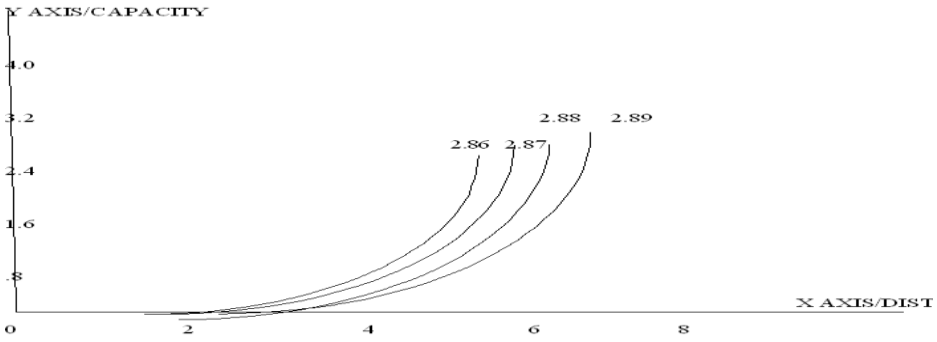

Fig. 2: Choosing of the Optimized VM.

As the number of priority queue updates, the time complexity for the algorithm also decreases, so our proposed work performs better than the k –means algorithm. Disadvantage of nearest neighbour calculation in load balancing using K-Means is that, it will consider only distance as the main factor. Capacity of under loaded machine is not given importance. So we move on to the del cluster algorithm. This algorithm uses two factors as we needed and is expected. We calculated the priority for the machines by using the two factors, distance and the machine capacity. The effective load balancing is acheived by del cluster algorithm which is given in formula 2. As we expected, the machine with distance 2 units and machine with capacity 8 got the preference for load balancing. By doing so we expect minimum difference that is .00x in response time when compared with the response time calculated by nearest neighbour algorithm.

Conclusions and Future Work:

In this paper, the agglomerative del cluster algorithm considers two parameter such as capacity and distance. Bee foraging behaviour includes clustering algorithm for reducing the response time. Here we compared the K-Means algorithm with the del cluster algorithm. This brings a response time reduction. Load balancing is done by the behaviour of bees. It transfers overloaded tasks to under loaded tasks. In this work, we had proposed a del cluster algorithm for hierarchical clustering which reduces number of priority queue updates by updating only the nearest neighbour. The proposed approach makes parallel and simultaneous task transmission possible through these algorithm which leads to the reduction of task transmission time. We also plan to extend this kind of load balancing of workflows with dependent tasks in future. In future there is also a chance of expanding this algorithm to large data sets. We have also planned to improve this algorithm by considering other QoS factors.

REFERENCES

Dhinesh Babu, L.D. and P. Venkata Krishnab, 2013. Honey bee behavior inspired load balancing of tasks in cloud computing environments, *Applied Soft Computing, Elsevier*, pp: 2292–2303

Eager, D.L. and J. Zahorjan, 1986. "A Adaptive load sharing in homogeneous distributed systems", *IEEE Transactions on Software Engineering*, 12(5): 662–675.

Gaochao, Xu, Junjie Pang and Fu. Xiaodong, 2013. "A Load Balancing Model Based on Cloud Partitioning for the Cloud " , *IEEE TRANSACTIONS ON CLOUD COMPUTING and DISTRIBUTED SYSTEM* in the YEAR.

Kaladevi, A.C. and M.V. Srinath, 2012. ECLB: A Novel Exhaustive Criterion Based Grid Load Balancing Algorithm for E-Learning Platform, *European Journal of Scientific Research* ISSN 1450-216X, 88(1): 171-183.

Karatza, H.D., 1994. Job scheduling in heterogeneous distributed systems, *Journal of Systems and Software*, 56: 203–212.

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

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

Wang, S.C. and W.P. Liao, 2011. "A Study of Load Balance Enhancement in a Hierarchical Cloud Computing Environment", *International Journal of Advanced Information Technologies (IJAIT)*, 5(2).

www.iete.org/miller-lou/del-clusteralg

Yagoubi, B. and Y. Slimani, 2006. Dynamic load balancing for Grid Computing, Transactions on Computing and Technology, 13: 260–265.

Yagoubi, B., M. Medebber, 2007. A load balancing model for grid environment,computer, iscis, pp: 1–7.

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