Energy Efficient Dynamic Load Balancing Algorithm for Cloud Environment

Akbarali Kharodia
M.E. Student
Information Technology
PIET,Limda,India

Rutvik Mehta
Assistant professor
Information Technology
PIET, Limda, India

Miren Karmta
Project scientist,BISAG
Gujarat
India

M.B.Potdar, Ph.D
Project director,BISAG
Gujarat
India

ABSTRACT
Cloud computing provides information technology resources and services on demand using the internet. It different types of services like IAAS (infrastructure as a service), PAAS (platform as a service) and SAAS(software as a service). Scenario of cloud computing refers to the two entities mainly cloud service provider and cloud service user. Cloud service provider provides the cloud resources and services to the cloud service user. Cloud service provider must have to use its resources efficiently to gain profit, but challenging task for them is maximum optimization of cloud resources. This paper describes approach which deals with saving of energy by optimally utilizing hosts and by minimizing active host using migration techniques.

General Terms
Algorithm for energy efficient dynamic load balancing in cloud environment.

Keywords
Cloud computing, load balancing, distributed solution

1. INTRODUCTION
Cloud computing is type of computing which is based on sharing of multiple computing resources instead of local server or personal server to handle application[1]. There are several issues in cloud computing like security, privacy, compliance, sustainability and load balancing. All of them load balancing is primary and biggest issue[1]. Distribute of load over shared pool of resources over internet is a biggest question. Suppose two or three servers are used to complete a work and one another mechanism is needed to determine which server should be used to assign a work, to determine which server is busy and which server is loaded. Some systems have to deal with the situation where servers are distributed over different geographical area [2]. This paper defines different feasible approaches for load balancing in different cloud systems.

First algorithm defines strategy that is inspired by nature to efficiently balance the load over distributed servers.

Second algorithm takes random samples of system domain and engineered them in such a manner that all nodes can be self-organized.

Third approach makes cluster of node with same type of behavior and balances load within them. Forth algorithm compares the load of current host with others and balances the load according to the current load of system.

Last approach calculates load of host dynamically and performs load balancing according to it[13].

2. CLOUD COMPUTING
Cloud computing was mainly developed to enable computation within geographically distributed and different type of resources. There is not any specific definition of cloud but it can be defined as a collection of distributed computers which are able to provide on demand computational resources and services with the help of internet[1]. As described earlier it provides services like IAAS, PAAS and SAAS to the geographically widespread customers. Well known example is Amazon Elastic Compute cloud which provides virtual computing environment, different configuration of CPU, processor and memory[3].

3. ALREADY EXISTING METHODS
In this section different schemes for cloud load balancing are defined and explained in terms of theirs behavior.

3.1 Honeybee Foraging
A self-organizing and behavior based algorithm was used for the load balancing at application layer[5]. Algorithm was inspired by Honey bees and theirs strategy to find food. Two entities are defined here one is forager and other is scout. A forager bee looks for the appropriate source of food, when they found specific food source they returned to the hive and performs "waggle dance" [5]. The appropriate source is selected according to the quantity, quality and distance of source. This all parameters are noticed by the forager bees and they show it to scout bees via waggle dance. Then scout bees follow the forager to harvest it.

This approach is used as resource searching algorithm for different computing applications. There are different virtual servers and virtual servers processes different requests from the users [4]. Each server acts as a forager or scout and after successfully fulfilling the request server places the advert on advert board. Initially, each server chooses virtual server randomly to serve a request, after completion of request load of current virtual server is calculated and compared with overall virtual servers load. If load of that virtual server is low than incoming requests will be assign to that virtual server. If load of that server is high than scout reads the advert board and will follow listed virtual server from advert board. A load of virtual servers are calculated according to the CPU utilization [5]. Figure shows how server allocation is performed by foraging.

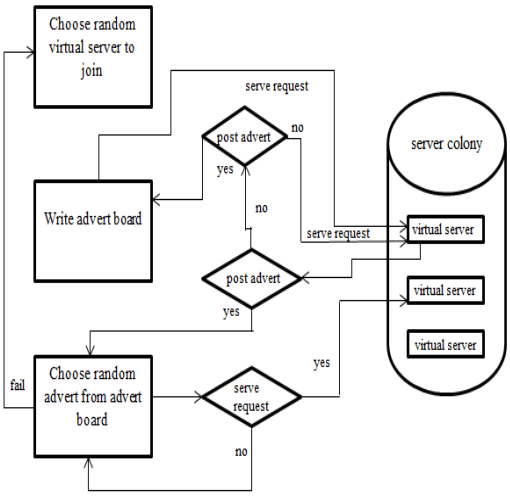

Fig 1.Server allocation in Honeybee foraging [5]

3.2 Random Biased Sampling

In this approach load of server is described according to its connectivity. Assigned links are defines capacity of server. Full working of this approach is found in [6].

This approach do not uses static network to monitor nodes and their available resources. Instead of that dynamic network approach is created which provides virtual machines load distribution status, resource update strategy.

Now, to define dynamic network mechanism node's in-degree (total edges pointing towards node= N) defines free resources of that node. When a node receives request for new task, it removes one incoming edge and decreases own in-degree. Deletion of one in-degree indicates that total available resources are reduced to N-1 [6]. The deletion and addition process of edges is defined by random sampling. Steady state defines that rate of job arrival is equal to the rate of job finishing so dynamic network mechanism is created that connects all the nodes.

The random sampling defines the deletion and addition of edges. Random sampling describes that all nodes in the network are randomly picked up. Sampling of initiator node is fixed, and after each step it moves to neighbor node. So by deletion and addition of edges in-degree of node remains proportional to the free resources of node [6].

3.3 Active Clustering

Similar nodes or services are grouped together to maintain load balancing for cloud systems[8]. This algorithm is considered as self-aggregation algorithm which rewires network. This algorithm can be used where nodes have knowledge of similar types of nodes and they are able to act behalf of it.

Each nodes are iteratively executed by Active clustering as follows[8]:

1. Any one node becomes "initiator" node at random time.

2. Initiator node selects "matchmaker" node.

3. The matchmaker node selects the similar node to the initiator node and creates link between them.

4. The matchmaker deletes node link between itself and selected neighbor.

This approach was studied in detail in [7] which shows formation on steady network from complex network.

3.4 Compare And Balance

Simple algorithm compare and balance was introduced in [9],which compares the load of current host and maintain equilibrium state by comparing current host load to others.

Current host is randomly selected at given time period. Using parameters like no. of virtual machine running, CPU usage and storage usage load of current host is determined. Then current host randomly selects other hosts from network and compare theirs load with own load. If current host load is more than load of selected node than current host transfers its extra load to the selected node. All the nodes in the system perform the same operation in the network.

3.5 Dynamic Compare And Balance

This algorithm (DCABA) was introduced in [10], it takes two threshold values. Current host load (H_load) defines which step will be executed next. Current host load is calculated as below:

- H_load = x (CPU usage) + y(RAM usage) + z(BW usage)

Where x, y and z € [0, 1] and they are weight coefficients.

Here used threshold values are calculated as below:

- Host_limit: it defines maximum capacity of host at which load host can perform with maximum efficiency.
- H_UTD (Upper threshold value of host): Host_limit * β, where β is weight coefficient which is selected by vendor.
- H_LTD (Lower threshold value of host): Host_limit * α, where α is weight coefficient which is selected by vendor.

Algorithms divides server optimization problem into two sections. First section distribute overload of current host using load balancing algorithm and second section reduces number of active host in network to support green computing concept[10].

Part 1: Suppose there is a situation when load of host is more than upper threshold value which is predefined by the vendor, so host is considered as overloaded. So load balancing algorithm is applied to reduce extra load and to transfer extra load to another host. Here it is performed by Adaptive compare and balance algorithm technique. It finds another host which can handle extra load without being overloaded.

Part 2: Suppose there is a situation where load of host is less than lower threshold value of host which is predefined by vendor, so to transfer extra load to the other host server-consolidation algorithm is used. This is achieved by migration of virtual machines. It finds another host which can handle extra load without being overloaded.

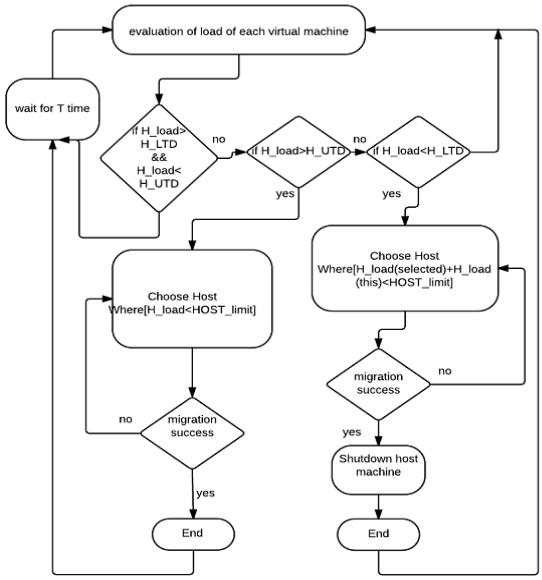

Fig 2.DCABA for server optimization[10]

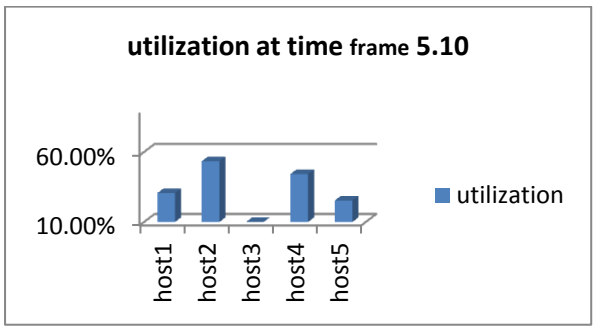

Fig 3.Result of DCABA algorithm

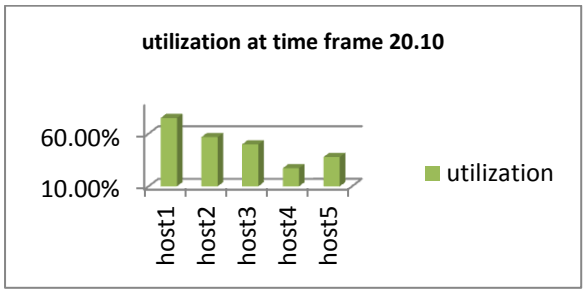

Fig 4.Result of DCABA algorithm

4. PROPOSED WORK

Dynamic compare and balance algorithm does not consider load of server at VM allocation time. It simply allocates the VM to server and then at predefined time interval T it checks load of the server. According to server's overload or under load condition further steps are performed. If server is overloaded then load balancing is applied using adaptive compare and balance algorithm and if server is under loaded then server consolidation algorithm is applied to shutdown server. After both scenarios VM migration is performed to balance the load.

Proposed algorithm works on prediction of host load after VM allocation. If server becomes overloaded according to prediction than VM will be created on different host.Proposed algorithm predicts load of host after VM allocation and if according to prediction host become overloaded than VM will be created on different host.So its minimize the number of migrations due to host overloading conditions. Proposed algorithm also supports green computing by minimizing number of active hosts using server consolidation. Algorithm migrates VM from host if host is under loaded and try to minimize active number of hosts to save energy

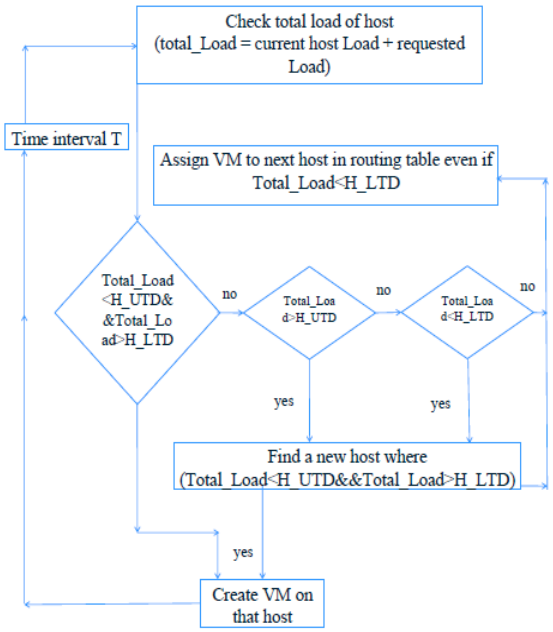

Fig 5.Proposed algorithm

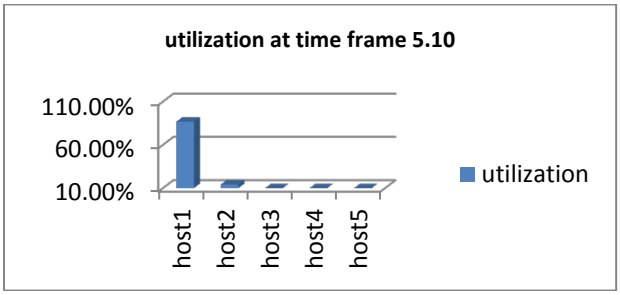

Fig 6.Result of proposed algorithm

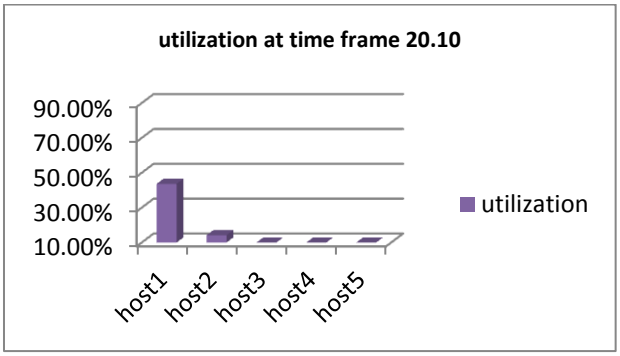

Fig 7.Result of proposed algorithm

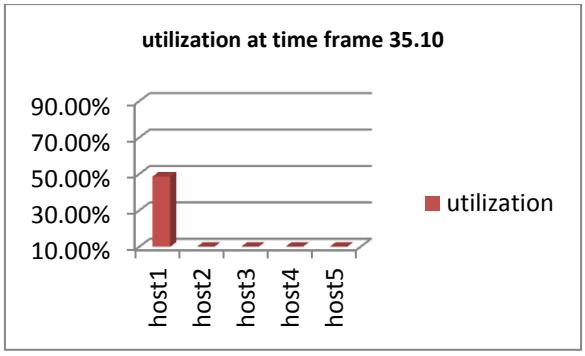

Fig 8.Result of proposed algorithm

5. RESULT ANAYLSIS & COMPARISON

All experiments are conducted in cloudsim simulator tool kit. Results clearly shows that proposed algorithm performs better than DCABA. Proposed algorithm reduces migration by checking host utilization before VM allocation. Proposed algorithm also supports green cloud computing by minimizing active hosts in cloud environment.

DCABA falls short due to the migrations and downtime during the migration. Proposed algorithm reduced migration due to the host overload by checking in advance that host is suitable for VM or not? If host have enough capacity to create and run new VM without being overloaded than olny new VM will be allocated to it so overload condition of host cannot be happen. Proposed algorithm also migrates the VM from least loaded host to ideal host to make under loaded host free to shut down. Results clearly shows that proposed algorithm can save more energy with less time with respect to the DCABA.

6. REFERNCES

[1] Cloud computing bible, 2011. By - Barrie sosinsky, publisher – Wiley

[2] Cloud Application Architectures: Building Applications and Infrastructure in the Cloud, 2009. By- George Reese, publisher - O'Reilly Media

[3] Amazon Elastic Compute Cloud (Amazon EC2) http://aws.amazon.com/ec2/

[4] Martin Randles, A. Taleb-Bendiab and David Lamb, Cross Layer Dynamics in Self-Organizing Service Oriented Architectures. IWSOS, Lecture Notes in Computer Science, 5343, pp. 293-298, Springer, 2008.

[5] S. Nakrani and C. Tovey, On Honey Bees and Dynamic Server Allocation in Internet Hosting Centers. Adaptive Behavior 12, pp: 223-240 (2004).

[6] O. Abu- Rahmeh, P. Johnson and A. Taleb-Bendiab, A Dynamic Biased Random Sampling Scheme for Scalable and Reliable Grid Networks, INFOCOMP - Journal of Computer Science, ISSN 1807-4545, 2008, VOL.7, N.4, December, 2008, pp. 01-10.

[7] F. Saffre, R. Tateson, J. Halloy, M. Shackleton and J.L. Deneubourg, Aggregation Dynamics in Overlay Networks and Their Implications for Self-Organized Distributed Applications. The Computer Journal, March 31st, 2008.

[8] Martin randles, David lamb and taleb-bendiab, A comparative study into distributed load balancing algorithms for cloud computing, 2009 Second International Conference on Developments in E-Systems Engineering

[9] Yi Zhao, Wenlong Huang, "Adaptive Distributed Load Balancing Algorithm based on Live Migration of Virtual Machines in Cloud" Fifth International Joint Conference on INC, IMS and IDC, IEEE, 2009, pp: 6/09.

[10] Yatendra sahu, R.K.Pateriya, Rajeev kumar gupta, Cloud Server Optimization with Load Balancing and Green Computing Techniques Using Dynamic Compare and Balance Algorithm, 2013 5th International Conference on Computational Intelligence and Communication Networks

[11] E. Di Nitto, D.J. Dubois, R. Mirandola, F. Saffre and R. Tateson, Applying Self-Aggregation to Load Balancing: Experimental Results. In Proceedings of the 3rd international Conference on Bio-inspired Models of Network, information and Computing Systems (Bionetics 2008), Article 14, 25 – 28 November, 2008.

[12] Repast Organization for Architecture and Development, http://repast.sourceforge.net

[13] A detailed analysis of distributed load balancing algorithm, Akbarali Kharodia,Rutvik Mehta,Miren Karamta,Dr.M.B.Potdar, International Journal of Scientific & Engineering Research, Volume 5, Issue 12, December-2014