Dynamic Chunk Allocation and Migration in Cloud Environment

Deepali R. Joshi
PG Department
MBES's College of Engineering
Ambajogai, India, 431 517.

B. M. Patil
PG Department
MBES's College of Engineering
Ambajogai, India, 431 517.

ABSTRACT
Cloud computing allows the all users to upload and download the resources as per their need. For allocating the resources cloud uses the virtualization concept. It allocates the data center resource to users on demand and minimizes the number of servers. If a load on server increases at that time user cannot get the required result within a time. For that purpose load rebalancing must be done. In this work we are applying the concept of "skewness" to measure the unevenness in utilization of servers. To minimizing the skewness, here we introduce the concept of load Re-balancing in cloud framework. It consider The CPU usage as well as memory for migrate the data object in to the server.

Keywords
Cloud computing, Virtualization, dynamic resource allocation, Data center, Load Re-balancing

1. INTRODUCTION
Cloud computing model allocates the resources dynamically. It automatically scales up and down the resources according to load variation. It optimizes the hardware cost, electricity and other expenses in large data centers. Most of the servers in data centers are under-utilized in cloud model due to excessed provisioning [2] [3]. Distributed file system is the basic building block of cloud computing. In distributed file system large files are fragmented into chunks and allocate each chunk to number of servers. In cloud, allocated files and number of servers are increased then the central node creates an obstacle. Virtual machine monitors (VMMs) like Xen hypervisor provide a mechanism for mapping virtual machines (VMs) to physical resources [4]. The mapping is hidden from the users. Amazon EC2 service does not know where their VM instances run [5]. Virtual Machine (VM) technology is used for resource provisioning. Virtualization reduces the average response time as well as according to the availability of resources it performs the task [12], [13]. VM live migration technology reduces the loads on servers and balances the load according to cpu utilization. During the load balancing cpu utilization is also increased and it overcomes the threshold value. To overcome this situation here introduces the concept of load Re-balancing which balances the load by calculating the cpu usage as well as memory utilization of server.

The load balancing cloud computing across the virtual machine maximizes the throughput. It uses the concept of skewness to measure the unevenness in server utilization. During load balancing it will first predict the load and then allocate the resources dynamically. The proposed load re-balancing model introduced here is aimed at the public clouds which divide the public cloud into number of cloud partition. It reduces the decision time and enhances the utilization of servers.

Our main focus on two concepts:

(a) Overload avoidance: PM should be capable to handle the VM running on it but if it overloaded then it degrade the performance of a system. Hence, to avoid overload on PM migration of resources should be takes place.

(b) Green computing: Dynamically allocated Resources can be handled by the VM so the Number of PM used should be minimized and Idle PM can turn off to save energy.

For overload avoidance we should keep the utilization of PM Low so that the resources can manage easily. To achieve green computing we should keep the utilization of some servers high. Here we present the resource management system to achieve these two goals.

We uses following concept to achieve results.

- Uses the concept of "skewness" for measuring the utilization of resources.
- Uses the load prediction for minimizing the migrations and data lost issues.
- The load Re-balancing concept to minimize the number of migration ie. Reduces the response time and increases the availability of resources.

2. LITERATURE SURVEY
Chase and Anderson [7] have proposed system for data center that perform the automatic scaling of web application for data centers. Here the replicas of web applications are stored by each server and hence the load of each server had increased suddenly. Tang [8] has proposed the load dispatch algorithm that performs the load distribution among all running machines. While minimizes the number of servers under utilization work uses network flow algorithm which allocate the load of web application among all running instances.

Chen [9] has presented an integrated approach for load dispatching and server storing technique for connection oriented services. Dynamic provisioning that dynamically turns on a minimum number of servers required to satisfy the quality services of web applications. Load dispatching distributes load on active servers. Above all work do not use virtual machines. A VM is just like a black box technique. Resource management is done in whole VMs. Zaharia [1] has proposed the mapreduce technique specially preserving data locality i.e. computations near their input data to maximize system throughput. In this technique if the data excessed then the threshold of the server goes increased and because of that hotspots had increased continuously.

Singh et. al [11] have proposed VM live migration technique.

It uses VM and data migration to migrate hot spots not only on server but also on network devices and the storage nodes. But this system was not support the green computing concept. Here introduced Extended Vector Product (EVP) which indicates the variation of resource utilization among all instances. Dhinesh et.al [12] have proposed honeybee algorithm for load balancing in cloud computing environment. In this method load equalization should be done among all virtual machine to boost up the throughput. This algorithm was updated for deciding resource allocation among all VM based on availability of resources and load on each machine.

Dong et. al [13] have proposed load balancing technique for parallel file system. In this framework information was exchanged between memory and the machine. During load balancing calculations, the heap of each server is diverse hence workload on each server was fluctuating persistently. Tanak and Bharati[14] have proposed "Load Balancing Algorithm for DHT Based Structured Peer to Peer System" , P2P system depends upon the DHT which offers abstraction for object storage and retrieval. The aim is to make sure even load distribution over nodes proportional to their capacities, and transferring virtual servers between heavily loaded nodes and lightly loaded nodes in a proximity-aware fashion for minimize the load-balancing cost. a proximity-aware load balancing scheme having the two main advantages and they are, from system viewpoint ,can reduce the bandwidth consumption for load balancing scheme dedicated to load movement.

3. SYSYTEM ARCHITECTURE

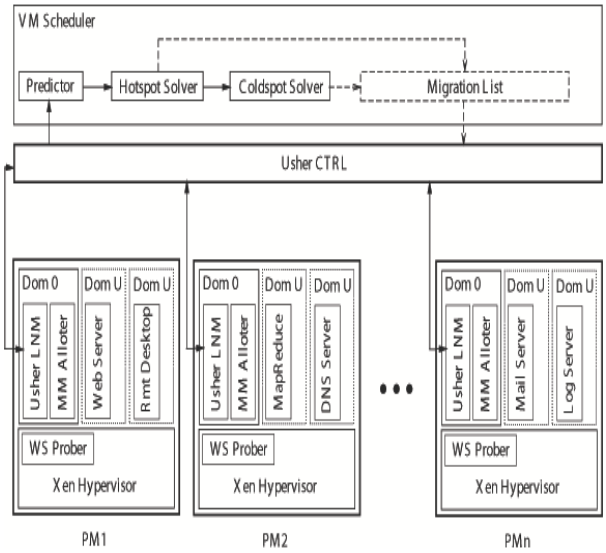

Figure 1. System architecture [1]

This system contains a set of servers used for running different application. Predictor is used to predict the load on server and future demand of VMS [1].

3.1 Skewness Algorithm

From the above architecture we introduce the concept of skewness which helps us to measure unevenness in the utilization of server. Generally, servers are classified in three types: hot spot, warm spot and cold spot. Hot spot is a small area in which there is a high temperature relative to its surrounding. Warm spot is the area in which temperature is always between hot and cold spot. Cold spot is also a small area in which there is a lowest temperature relative to its surrounding. All these spots are follows the threshold technology to classifying them.

Our algorithm evaluates the resource allocation based on demands of VMs. Here we define the server as a hot spot if utilization of resources exceeds the hot threshold. This indicates that server is overloaded and hence VMs running on it should be migrated on any other server.

Let n be the number of resources and r_i is the utilization of i-th resource. Here we define the skewness of server p where r" is the average utilization of all resources for server 'p' is represented in equation (1).

$$skewness(p) = \sum_{r \in R}^{n} (\mathrm{r} - \mathrm{ri})^2 \qquad \text{---------- (I)}$$

Skewness algorithm consists of two steps:

- ***Hotspot migration***

When server is above the hot threshold or server is overloaded at that time it is needed to migrate some running VM away from the server. This can be done by using hot spot migration. Our aim is to eliminate all hot spots if possible. For that it is needed to sort the list of all hot spots in descending order of their temperature i.e. handle the hottest one first. For each server p, sort the list of all VM according to their temperature and select the VM that can reduce the servers temperature most. In case of ties, select the VM whose removal can reduce the skewness of server. For each VM we define the destination server. If a destination server is found, we migrate the VM to that server and update the predicated load of related server. Otherwise, move on to the next VM in the list and try to find destination server for it.

- ***Green computing***

It is recent trend towards operating system to save the energy. When the resource utilization of active server is too low, some of them can be turned off to save energy. This can be handling by green computing algorithm. When average utilization of resources on active server are below the green computing threshold at that time this algorithm is invoked. Here the challenging task is during low load reduce the number of active server without sacrificing performance of a system.

For clod spot on server p, it must to check that we can migrate all its VMs somewhere else. For that we find the destination server who accommodate. After migration of all VM on server p, check whether the server is below cold spot or not. If it is then green computing algorithm turnoff that server and save the energy. After accepting the migrated VM the resource utilization of server must be below the warm threshold. While we can save the energy it may create hot spot in future hence warm spot is design to prevent the system from this situation. We try to eliminate cold spot with lowest cost first. We select the servers which contains the least skewness. If we find the destination server for all VMs on cold spot, we record the continuous migration and released servers predicted load can be updated. Otherwise, we do not migrate any of its VMs. Eventually we save the energy by keeping the servers in idle states.

4. PROPOSED WORK

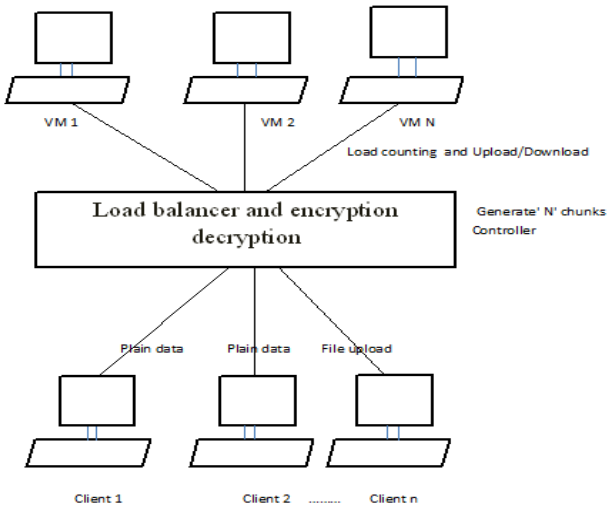

Figure 2. Proposed System architecture

Figure 2 shows proposed system architecture .Each client contains a statistical collection of files which are forwarded to job manager where sub servers are work. Job manager calculates the load on each node before commit on database. Job manager creates chunks for each server according to server load. Server load can be calculated on the basis of cpu utilization of each server as well as memory occupied by each server. Once it calculates the load with these two entities it disperses the chunks over multiple servers. Data encryption should be performed by servers with the help of DES and MD5 encryption scheme and unique key can be send to authorized users. Load rebalancing task can change file allocation among VMs by adjusting their load.

Load Re-balancing schemes depending on whether the system dynamics are important can be either static or dynamic. Static schemes do not use the system information and are less complex while dynamic schemes will bring additional costs for the system but can change as the system status changes. A dynamic scheme is used here for its flexibility. The model has a main controller and balancers to gather and analyze the information. Thus, the dynamic control has little influence on the other working nodes. The system status then provides a basis for choosing the right load balancing strategy.

The load Re-balancing model given in this article is aimed at the public cloud which has numerous nodes with distributed computing resources in many different geographic locations. Thus, this model divides the public cloud into several cloud partitions. When the environment is very large and complex, these divisions simplify the load balancing. The cloud has a main controller that chooses the suitable partitions for arriving jobs while the balancer for each cloud partition chooses the best load balancing strategy.

During load balancing each server firsts estimate weather it is overloaded or under loaded without any global knowledge. A server is light if the number of chunks is smaller than the threshold. Here the nodes are randomly selected. Specifically, each node contact to other number of randomly selected nodes and form a vector 'V'. Vector consists of entries of each node. Specifically, in this study, suggest offloading the load rebalancing task to storage nodes by having the storage nodes balance their loads spontaneously. This eliminates the dependence on central nodes. The storage nodes are structured as a network based on distributed hash tables (DHT). DHTs enable nodes to self-organize and repair while constantly offering lookup functionality in node dynamism, simplifying the system provision and management. It also helps for replica management.

In this system the hot spot migration depends on not only cpu utilization but also memory utilization. During the hot spot detection first selection of server is performed by the user. User can select the server randomly there is no any scheduling technique is used for it. After selection of server it predicts the load of server and checks the status of the server those are stated below.

- If load of node is equal to zero it means the server is in idle state.
- If load degree is greater than zero but less than the threshold value then it should be considered in a normal state.
- If load of degree is exceeds the threshold value then it considered in overloaded state.

During the execution of system if the threshold value is below the degree of a load then chunks directly send on the selected node but if it is exceeds the threshold value then server first find the server which contains the least load and send file on that node. Here the thresholds are set according to cpu and memory utilization of servers. Here if cpu utilization maximizes then it calculate the memory utilization if it will below the threshold then file must be send on the selected server. System calculates the load on cpu and memory basis hence the number of migrations are reduced. Load rebalancing scheme first predict the load according to cpu utilization of resources but if it exceeds the threshold at that time it compute the memory utilization of servers and then rebalance the load according to it. In the previous technique if cpu usage exceeds the threshold at that time the system migrates the VMs on the least load servers but here it rebalance the load with the help of memory instead of migration. After migrations of chunks the load should be rebalance and the load must be update.

Here we have used the amazon EC2 cloud service for creating and running the instances. It provides facility to automatic grow and shrink the memory of any instance after reconfigure it. Hence if cpu and memory will exceeds the threshold at that time we can able to reconfigure the instance and increase the memory easily. Here chunks can be allocated according to availability of memory on each server. In order to afford security to data, the data is stored in the encrypted shape in the nodes. The file that is to be uploaded in the cloud is chosen by the client. The encryption procedure is performing over the data with the key provided. The encrypted file is finished in different chunks and stored in different nodes. At the time of downloading those chunks are decrypted and merge in one file and then send to the user.

Chunk formation depends on availability of space on each server. It's not compulsory at every time to form a chunk in equal size. File is fragmented according to threshold. After sending of chunk if any server goes in hot threshold it immediately rebalance the load with memory utilization of that server and accept it. Because of this load rebalancing technique most of the time all servers are normal state. Here inputs are taken as follows.

4.1 Mathematical Module

System S = {Ts, CPUuti , WCcpu ,Tm, Um, WCm, F, A, Ds, Bw, Nlat, WCnet}

Where

Ts = Timestamp

CPUuti = CPU Utilization

WCcpu = Weight constant of CPU

Tm = Total Memory

Um = Used Memory

WCm = Weight constant of memory
F = Finish time of user request

A = Arrival time of user request

Ds = data size of single request

Bw = Band width

Nlat = Network latency

WCnet = Weight constant of network

Here the weight constant assumption parameter into scale between 0 and 1; divide to CPU, Memory and Network, so that the total is currently just 1. The CPU has greater impact on the execution of a VM comparing in memory or net. So, it has a maximum weight constant.

Inputs:
The static parameters include the number of CPU's, the CPU processing speeds, the memory size, etc. Dynamic parameters are the memory utilization ratio, the CPU utilization ratio, the network bandwidth. T is threshold.

Process:-

1. Define a load parameter set: F= {F1,F2…Fm} with each Fire presents the total number of the parameters.
2. Compute the load degree as

   Load Degree(N)= Σ αiFi

   Where i= 1…m.
3. Average cloud partition degree from the node load degree statistics as:

   Load degree avg=Σi=1..n Load Degree(Ni)
4. Three level node status are defined

   Load degree(N)=0 for **Idle**

   0<Load Degree(N[i])>=< T **Normal**

   Load_Degree(N[i]) > T **Overloaded**

Output :-

Idle or Normal Or Overloaded

In this structural engineering a brought together load balancer is utilized to part the record into pieces keeping in mind the end goal to store the information in different hubs. The information to be put away in the cloud is scrambled before capacity for more security. The encryption is finished by the key created at the customer side. At that point the scrambled information is made into lumps and put away in different hubs. At the point when the server control performs operations on information such as cancellation or updating load awkwardness issue happens. This issue can be explained by the rebalancing calculation which adjusts the heap in the cloud after the above operations performed. Like this we can perform the load rebalancing technique.

5. ALGORITHMS

Algorithm 1: Compute VM Load from data nodes

Input: ith Node input

Output: - Idle or Normal Or Overloaded in percent (%).

Compute Load (VM id) : weight Degree Inputs: The static parameter comprise the number of CPU's, the CPU dispensation speeds, the reminiscence size, etc. active parameters are the memory consumption ratio, the CPU exploitation ratio, the network bandwidth.

Procedure:-

1. Characterize a load limit set: F= {F1,F2…Fm} with each Fire present the total number of the consideration.
2. Calculate the load capacity as weight Degree(N)= ∑ αiFi Where i= 1…m.
3. Ordinary cloud partition degree from the node consignment degree statistics as:

   Load amount avg=∑i=1..n
4. LoadingDegree(Ni)

   Three height node position are defined

   Load_degree(N)=0 for inactive,

   Load_degree(N) <= T for normal and

   Load_degree(N)>T for overfull.

Algorithm 2: VM object Migration

Input: Input File from user f, memory size f is M, selected server s.

Output: File store with specific server

1. User select file randomly and select the server.
2. System takes each server memory load from assign server sm.
3. if (M>sm)

   Find i to n from available server
4. Select i th server which having memory >M and min load degree.
5. Create file chunks and encrypt the data.
6. Store the data on ith server.
7. End procedure

Algorithm 3: DES with MD5.

Basically MD5 is hashing function, its used for generating a hash of given string instead of SHA, but DES is encryption techniques. When we use DES with MD5 and PBE's, so PBE is work like secure hash function of MD5 data and DES encrypt all the data using crypto technique within a single key for both direction. So finally conclude, we can provide better security instead of RSA, Elgamal encryption scheme, ECC using combination of these three functions using the single key which will eliminate the additional resource dependency and time complexity as well.

Below is the procedure of DES with MD5 encryption using

PBE's function.

Key generation mode

- Key= {byte [], random, k}
- Data encryption mode
- Cipher= {plain data, key}
- Data decryption mode
- Plain data= {cipher, key}

This part has been done using set theory. The given algorithms complexity has measure using polynomial time.

Output: specific data node with load ratio and file transaction success or fail.

6. RESULT AND DISCUSSION

The system is work with multiple virtual machines. One we deploy the system on EC2 environment it will work fine. For the proposed system performance evaluation, we calculate matrices for accuracy. We implement the system on java 3-tier MVC architecture framework with INTEL 2.5 GHz i3 processor and 4 GB RAM. The system is work with multiple virtual machines. One we deploy the system on EC2 environment with four virtual machines it will work fine. The estimated results for the system based on below tables. The below table show how re-balancing algorithm works with chunk creation. On the basis of below table we first given 90 KB file as input to the system, and how workload has distributed into the different servers based on CPU load.

Here X=data size, Y= CPU load

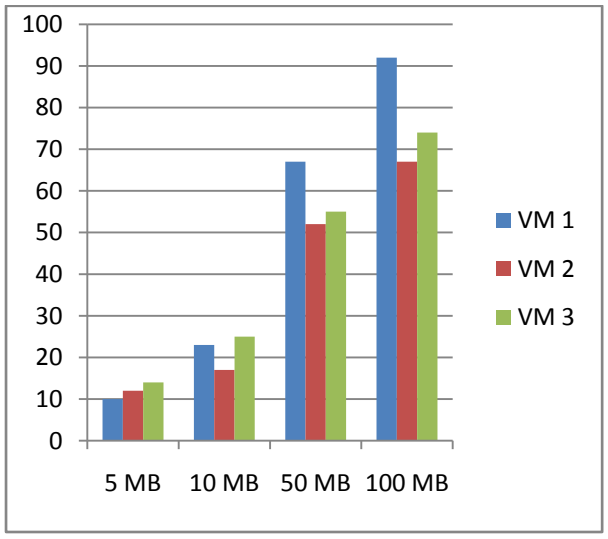

Figure 3. Data Uploading performance

Here first we upload the 5 MB files by client that fragmented in to chunks and send it among three virtual machines. After increasing the size of file the load on CPU will get increased here file distributed in to chunk and distributed according to availability of usage in each VM. After uploading the 100 MB file the VM1 exceeded and reached above 90 percent and we can say that the server in hot threshold. Here we consider the threshold ranges are given in below table.

Table1. Parameters for experiment

Thresholds	Value
Hot	Above 0.9
Cold	Below 0.2
Warm	between 0.2 to 0.9

at this point the load rebalancing works. It calculates the CPU and memory utilization of server if it exceeds the cpu usage it calculates the memory and if it contains a enough memory to accept the chunk then it accept it without any migration. During this the decision time will be calculate according to the time required to send the file over a Virtual machines. It is shown below.

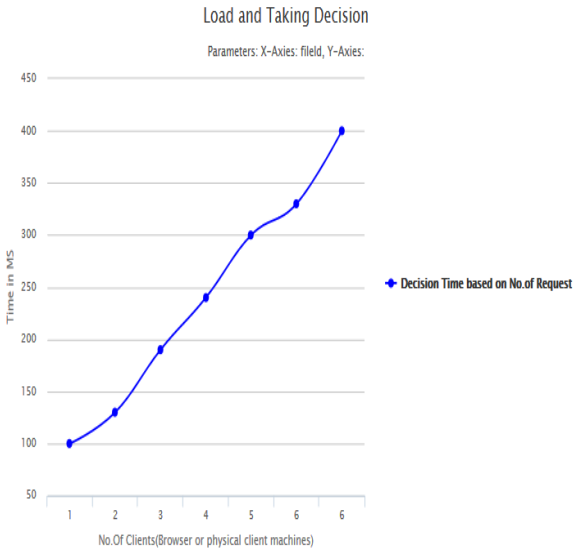

Figure 4. Decision time

Here the time is measured in milliseconds. Once the utilization is reached at the highest point that is in hot threshold then system detect the hot spot here in figure 5 we have shown the number of hot spots detected based on CPU and memory. Here when hot spots are detected according to CPU utilization it requires the no. of migrations within sometime after uploading hence the hot spots detected earlier. But when we calculate both CPU and memory utilization the hot spots are detection is minimized and hence the hotspots are detected but it resolves easily i.e. the chunks are send according to memory of server. If we consider both the parameters then the migrations will get reduced and files reached at the destination. Here we compare both the results against hot spot. First hot spot detected on VM1 then VM1 is migrated on VM3.after sending the next file the load on VM2 will get increased and it detects the hotspot hence it then migrated and load is balanced. According to this technique load balancing is performed. This hotspot detection is shown in below figure.

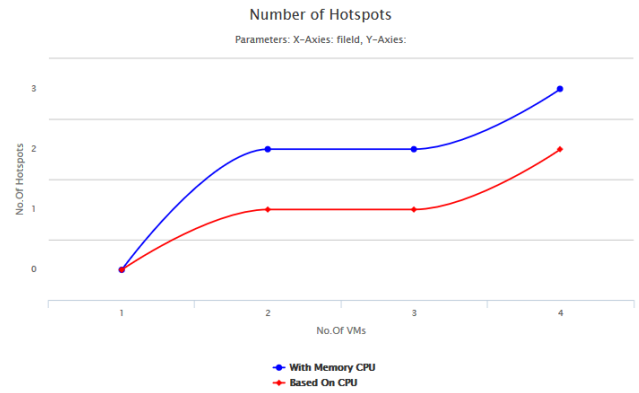

Figure 5. Average number of hotspots

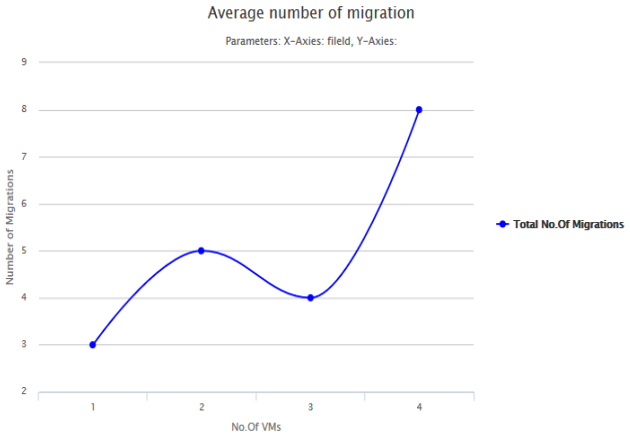

Figure 6. Average number of migrations

After detection of hotspot migration should be takes place. It is shown in above figure. Here the chunks are distributed over the servers based on its capacity. When VM1 get heated i.e. when it exceeds its CPU and memory utilization it detects the hotspot and migration begins. During the migration server find the other server which has a capacity to accommodate the VM1 and then it migrate it on another server. After that regular uploading is started. Again some time elapsed hotspot is detected on VM2 again it resolves and files are uploaded. Here migrations are reduced as compared to previous result. Hence this technique is more effective than previous.

7. CONCLUSION AND FUTURE SCOPE

The load balancing is a very important task in a Cloud Computing environment to achieve maximum utilization of resources. We discussed various load balancing schemes such as existing system and proposed, each having some pros and cons. On one hand existing load balancing scheme provide easiest simulation and monitoring of environment, but difficult to model the heterogeneous nature of clouds. On the other hand, dynamic load re-balancing algorithms are difficult to simulate but are best suited in a heterogeneous environment of cloud. Load should be balanced over various hubs to enhance framework execution; response time and crypto system security mechanism are also applied over the rebalancing task.

For the future enhancement we can focus on multi cloud load balancing, such concept of load balancing is a little different between different research areas by different researchers. There is no description for load balancing in future internet. For the additional feature enhancement we also focus memory, bandwidth, and network virtualization using Xen hypervisor. Basically Virtualization technology can be used either on its own or in the cloud. Our software is readily used in either situation. In order to facilitate its use in the cloud, we include the Xen Project API (XAPI). This is the power behind cloud solutions like Xen Server and the (now deprecated) XCP.

8. REFERENCES

[1] Zhen Xiao, *Senior Member, IEEE*, Weijia Song, and Qi Chen "*dynamic resource allocation using virtual machine for cloud computing environment*", in ieee transaction on parallel and distributed systems year 2013.

[2] L. Siegele, "Let it rise: A special report on corporate IT," in *The Economist*, Oct. 2008.

[3] M.Armbrust *et al.*, "Above the clouds: A berkeley view of cloud computing," University of California, Berkeley, Tech. Rep., Feb 2009.

[4] P.Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, *"Xen and the art of virtualization,"* in *Proc. of the ACM Symposium on Operating Systems Principles (SOSP'03)*, Oct. 2003.

[5] "Amazon elastic compute cloud (Amazon EC2), http://aws.amazon.com/ec2/."

[6] TPC-W: Transaction processing performance council, http://www.tpc.org/tpcw/."

[7] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, "Managing energy and server resources in hosting centers," in *Proc. of the ACM Symposium on Operating System Principles (SOSP'01)*, Oct. 2001.

[8] C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici, "A scalable application placement controller for enterprise data centers," in *Proc. of the International World Wide Web Conference (WWW'07)*, May 2007.

[9] G. Chen, H. Wenbo, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao, "Energy-aware server provisioning and load dispatching for connection-intensive internet services," in *Proc. of the USENIX Symposium on Networked Systems Design and Implementation (NSDI'08)*, Apr. 2008.

[10] M.Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, and I. Stoica, "Improving MapReduce performance in heterogeneous environments," in *Proc. of the Symposium on Operating Systems Design and Implementation (OSDI'08)*, 2008.

[11] A.Singh, M. Korupolu, and D. Mohapatra, "Server-storage virtualization: integration and load balancing in data centers," in *Proc. of the ACM/IEEE conference on Supercomputing*, 2008.

[12] DhineshBabu L.D, P. Venkat, *"Honey bee behavior inspired load balancing o aKrishnaf tasks incloud computing environments"*, Applied Soft Computing 13 (2013) 22922303.

[13] Bin Dong, Xiuqiao Li, QimengWu, Limin Xiao, Li Ruan, *A dynamic and adaptive load balancingstrategy for parallel file system with large-scale I/O servers*, J. Parallel Distribution Computing.

[14] ChahitaTanak, Rajesh Bharati "Load Balancing Algorithm for DHT Based Structured Peer to

PeerSystem"International Journal of Emerging Technology and Advanced Engineering (ISSN 2250-2459, ISO9001:2008 Certified Journal, Volume 3, Issue 1, January 2013).

[15] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proc. of the Symposium on Networked Systems Design and Implementation (NSDI'05)*, May 2005.

[16] M. Nelson, B.-H. Lim, and G. Hutchins, "Fast transparent migration for virtual machines," in *Proc. of the USENIX Annual Technical Conference*, 2005.