# Load Balancing in Cloud Based on Live Migration of Virtual Machines

Raghavendra Achar*, P. Santhi Thilagam*, Nihal Soans†, P. V. Vikyath †, Sathvik Rao† and Vijeth A. M.†
*Department of Computer Science and Engineering
National Institute of Technology Karnataka, Surathkal 575025, INDIA
†Department of Computer Science and Engineering
St. Joseph Engineering College, Mangalore 575028, INDIA

***Abstract*—Cloud computing is an upcoming trend in the field of computer science in recent years. In cloud, computing resources are provided as service in the form of virtual machine to its clients across the globe based on demand. Huge demand for cloud resources results in overutilization of servers whenever there is a heavy load. It is necessary to distribute the load across the servers in cloud by taking into consideration of allocating the right amount of resources dynamically based on the load to improve the performance of applications running in virtual machines. In this paper we present an algorithm which dynamically allocate resources based on the need and distribute the load across the servers. We conducted the experiment on Xen Cloud Platform. We use response time as a metric. The experiments conducted shows that the proposed algorithm improves the performance of applications running in virtual machines by using the feature scaling and migration.**

***Keywords*—*Virtual Machine, Live Migration, Cloud Computing.***

## I. Introduction

Cloud Computing is a new paradigm where computing resources are made available to the users in a pay per use manner [1][16]. Cloud Computing provides computing resources mainly in 3 forms namely Software as a Service (SaaS), Platform as Service (PaaS) and Infrastructure as a Service (IaaS). In SaaS, a Software application is made available to the requester. Salesforce is an example for SaaS. In PaaS, Application development platform is made available to the requester. Microsoft Azure is one of the example for PaaS. In IaaS, an Infrastructure is provided as a service to the requester. Some of the providers who offers IaaS are Amazon, Rackspace etc. Virtualization [18] is the technology behind the cloud which realize the vision of utility computing. Some of the hypervisors used in virtualization are Xen, KVM, VMware ete. Cloud provides computing resources in the form of virtual machine, which is a abstract machine runs on physical machine. The requester using the virtual machine will have the feel of working on physical machine. Cloud comprises of huge number physical servers. It is necessary to avoid hotspot for efficient resource utilization in a cloud. Overloaded servers results in performance degradation of applications and under loaded servers results in inefficient resource utilization. It is necessary to provide right amount of resource dynamically to the applications running in virtual machines in order to meet the performance. In this paper we present an algorithm for the same.

The rest of this paper is organized as follows. Section 2 reviews the related work. Sections 3 presents methodology. Section 4 describes the implementation and experimental study and section 5 presents the conclusion.

## II. Related Work

There have been some work towards resource allocation and load balancing in cloud. In paper [2] author presents an architecture and algorithm namely COMPARE_AND_BALANCE. Presented architecture uses a small cronjob which runs on each host to monitor the resources like CPU and IO load, and log their usage to the log directory on the shared storage. Algorithm runs on each physical machine and dynamically migrates virtual machines from one host to other based on resource usage. For implementation Red Hat Cluster Suite is used to create and manage cluster services. In paper [3] author presents a mechanism for load balancing of virtual machine resources based on genetic algorithm. For this authors consider historical data and current state of the system. Author use tree structure to mark the chromosome of genes and every mapping solution is considered as one tree. Here scheduling node of the system in the first level is the root node, all nodes in the second level represents physical nodes and all nodes in the third level represents virtual machines. Mechanism based on weighted least connection algorithm is presented in paper [4]. Here author consider the webservers running long-connectivity applications. Exponential smoothing forecasting method is used as prediction algorithm and it takes historical data and distinguishes them through the smoothing factor to let recent data make a greater impact on the predictive value than long-term data. In paper [5] authors presents dynamic and integrated resource scheduling algorithm for Cloud datacenters. Here author consider the factors like CPU, memory and network bandwidth. Author develop an integrated measurement for the total imbalance level of a cloud datacenter as well as the average imbalance level of each server. Cloud task scheduling policy based on Load Balancing Ant Colony Optimization (LBACO) algorithm is presented in paper [6]. The presented work balance the entire system load while trying to minimizing the makespan of a given tasks set. In Paper [7] author presents a virtual machine mapping policy based on multi-resource load balancing. Here resource consumption of the running virtual machine and the self-adaptive weighted approach is used to resolves the load balancing conflicts. Presented model adopts the centralized control architecture containing two main components namely

978-1-4799-2275-8/13/$31.00 ©2013 IEEE

scheduling controller and resource monitor. The scheduling controller is responsible for virtual machine lifecycle management and fulfilling allocation policy. The resource monitor collects the information about resources from physical hosts. In paper [8] author presents a load balancing scheme and a migration policy for virtual machine cluster to identify a suitable virtual machine from overloaded server to the suitable target server. Paper presents a prediction method to ensure the transient spike does not trigger needless virtual machines migration. Paper also presents a benefit estimates model in order to decide whether the migration of jobs in virtual machines under the same physical machine is benefit for the whole system. In paper [9] author presents a load balance model for the public cloud based on the cloud partitioning concept with a switch mechanism to choose different strategies for different situations. In paper [10] author presents a dynamic load balancing algorithm based on virtual machine migration. The algorithm use trigger strategy based on fractal methods. The strategy determines the timing of the virtual machine migration through forecasting. In our work we have consider both scaling and migration for efficient resource allocation.

## III. Problem Statement

In the world of Cloud Computing, dynamic resource allocation and load balancing is an interesting issue which is open for research. The success of this rising model is dependent on the effectiveness of techniques used to allocate the resources in most optimal way. Thus to achieve this, the following scenario has been taken up as the problem statement.

Let $C = \{S_1, S_2, ..., S_n\}$, where C is a cloud and $S_1, S_2, .., S_n$ are the servers. Let $S_j = \{v_{j1}, v_{j2}, ...., v_{jl}\}$ where $v_{j1}, v_{j2}, ..., v_{jl}$ are the virtual machines in the server $S_j$. Let $v_{ji} = \{v_{id}, v_{cpu}, v_{ram}\}$ where $v_{id}$ is the virtual machine Id, $v_{cpu}$ is the speed of the processor and $v_{ram}$ is the RAM size of the virtual machine. Let $CPUmax_i$ and $RAMmax_i$ be the changeable maximum CPU and RAM respectively that can be allocated to the virtual machine $v_i$. Let $R_1, R_2, R_3, ...R_m$ be the response time of the applications running in virtual machines and m is the total number of virtual machines in the cloud. The problem is to allocate the right amount of CPU $sc_i$ and RAM $rm_i$ with the help of scaling and migration to each virtual machine $v_i$ to ensure that the response time $R_i$ is within the acceptable range.

## IV. Methodology

This section describes the methodology used in order to achieve specific objectives. Presented work is based on XCP(Xen Cloud Platform) and Credit Scheduler. We brief about Xen, Credit Scheduler followed by system architecture, interactions and algorithms.

### A. Xen and Credit Scheduler

Xen [11] is a hypervisor which allows the execution of multiple virtual machines on a single physical machine. It is responsible for CPU scheduling of the all virtual machines running on the hardware. Xen not only abstracts the hardware for the virtual machines but also controls the execution of virtual machines. Domain U (DOM_U) are the virtual machines which has no direct access to physical hardware on the machine. Domain O (DOM_O) is a virtual machine running on the Xen hypervisor which has special rights to access physical I/O resources as well as interact with the other virtual machines (DOM_U). All Xen virtualization environments require DOM_O to be running before starting any other virtual machines. The credit scheduler [12] is a proportional fair share CPU scheduler, which is the default scheduler for Xen hypervisor. In credit scheduler each domain is assigned weight and cap. The values for weight can be varied from 1 to 65535 and the default value is 256. A domain with weight 512 will get double cpu usage as with weight of 256.

### B. System Architecture

Figure 1 shows the system architecture. The architecture consists of n number of servers and cloud controller. Each server is having a hypervisor like Xen to run multiple virtual machines. Requester requests for computing resources through cloud controller. The required computing resources are provided in the form of virtual machine.

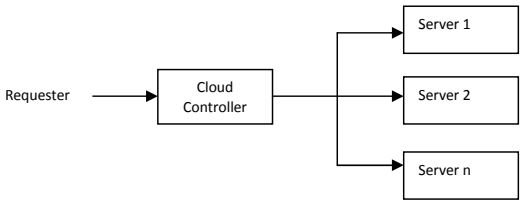


Fig. 1. System Architecture

Following scenario in Figure 2 depicts the interaction among various components when applications are running in a virtual machine. During the interaction, the DOM_O sends a Req_ResourceStatus message to the virtual machine, requesting the details of resource usage. If the resources granted to the virtual machine is underutilized (under loading), the virtual machine sends an Res_Underloaded message back to the DOM_O via the hypervisor. The DOM_O makes use of the existing scaling methods to indicate the level to which the resources must be scaled down. The information provided by the DOM_O is then sent to the hypervisor which performs the job of scaling down of the resources, the result of which is reflected in the operation of the virtual machine.

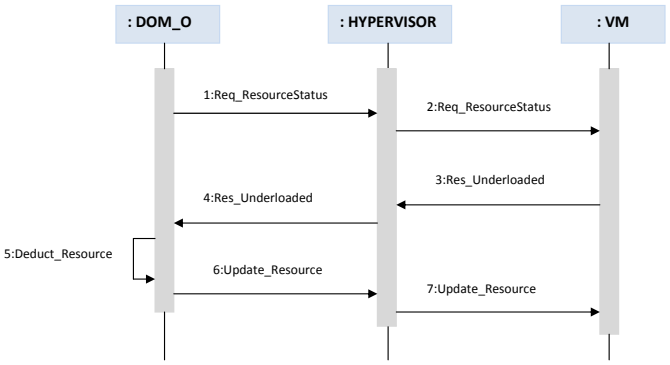


Fig. 2. Sequence of activities to scale down the resources

Below in Figure 3 depicts the scenario where the virtual machine is in need of resources. The DOM_O sends a Req_ResourceStatus message to the virtual machine through the hypervisor requesting the details of resource utilization. The virtual machine sends an Res_Overloaded message back to the DOM_O stating that resources are overloaded. The DOM_O searches for resources that are not used or wasted by other virtual machine. When resources are found, the resources are allocated to the virtual machine as per requirements (scaling up). In case if there is no enough resources that can be provided in the current server, the virtual machine needs to be migrated to a suitable server. There are two ways in which virtual machine can be migrated. They are Live migration [13][14][15] and Non-Live Migration. Live Migration refers to the process of moving a running virtual machine between different physical machines with minimum down time. Memory, storage, and network connectivity of the virtual machine are transferred from the original host machine to the destination. Here we perform Live Migration. The DOM_O identify the suitable target server and migrate the virtual machine to the target server. The server to which the virtual machine is migrated is decided based on the selection criteria provided in the algorithm. Based on this, hypervisor initiates the migration of the virtual machine to the suitable target server.

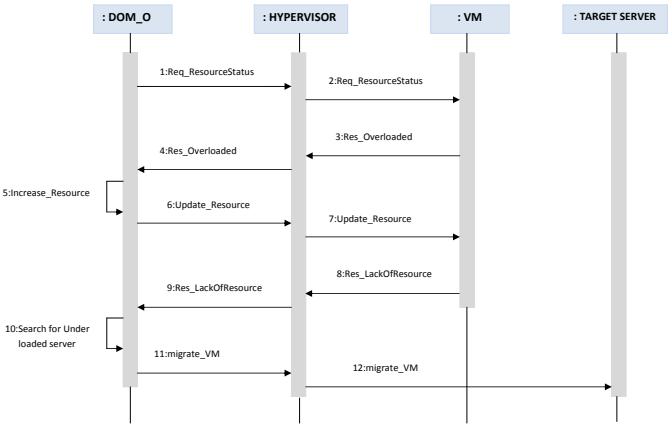


Fig. 3. Sequence of activities to migrate VM to suitable target server

### C. Resource Allocation and Load Balancing Algorithm

This section presents Resource Allocation and Load Balancing Algorithm. Resource allocation is the process of allocating the required amount computing resources such as CPU, memory and storage to the virtual machine automatically. The algorithm periodically checks the CPU and RAM Utilization. If the Applications running in the virtual machine is in need of additional resources, initially it is provided through scaling. If enough resources are not available, then virtual machine is migrated to the suitable server where there is enough resource.

**Algorithm 1** Resource Allocation and Load Balancing

```
Amt ← Resource in Percentage
T ← Time in second
CPU ← CPU Utilization
RAM ← RAM Utilization
while VM is running do
  if CPU > UpperThreshold1 and CPU < UpperThreshold2
  then
    Scale Resource up by Amt
    Wait for duration T
    if CPU > UpperThreshold1 and CPU < UpperThresh-
    old2 then
      Scale Resource up by Amt
      Wait for duration T
      if CPU > UpperThreshold1 and < UpperThreshold2
      then
        MigrateToServer( )
      end if
    end if
  else if CPU > UpperThreshold2 then
    Scale Resource up by 2*Amt
    if CPU > UpperThreshold2 then
      MigrateToServer( )
    end if
  else if CPU < LowerThreshold then
    Scale Resource down by Amt
    Wait for duration T
    if CPU < LowerThreshold then
      Scale Resource down by Amt
      Wait for duration T
      if CPU < LowerThreshold then
        MigrateToServer
      end if
    end if
  end if
  if RAM > UpperThreshold1 and RAM < UpperThresh-
  old2 then
    Scale Resources up by Amt
    Wait for duration T
    if RAM > UpperThreshold1 and RAM < Up-
    perThreshold2 then
      Scale Resource up by Amt
      Wait for duration T
      if RAM > UpperThreshold1 and < UpperThresh-
      old2 then
        MigrateToServer( )
      end if
    end if
  else if RAM > UpperThreshold2 then
    Scale Resource up by 2*Amt
    if RAM > UpperThreshold2 then
      MigrateToServer( )
    end if
  end if
end while
```

In this algorithm, parameter Amt refers to amount of additional resource added to the virtual machine for every duration T, if it needs. The variable UpperThreshold1, UpperThreshold2

and LowerThreshold indicates the threshold value for resource utilization. We set UpperThreshold1 as 70%, UpperThreshold2 as 95% and LowerThreshold as 10%. For every iteration current CPU utilization is compared with UpperThreshold1 and UpperThreshold2. If its value is between these two threshold, additional resource of amount Amt is added to the virtual machine by scaling. After duration T, if its value is between UpperThreshold1 and UpperThreshold2, again resource is scaled up by Amt. After duration T, if utilization is still between LowerThreshold1 to LowerThreshold2, virtual machine is migrated from current server to the suitable target server. We consider the servers with average utilization as the target server. Similarly algorithm proceeds for RAM utilization. For our experiment we consider the value of Amt as 20 and value of T as 20.

### D. Algorithm for Scaling

This section presents scaling algorithm for virtual machine. The algorithm takes 3 input parameters namely Resource Name, Scale Type and Amt. Resource Name refers to the name of the resource to be scaled. Here Resource Name may be either CPU or RAM. The second parameter is the Scale Type. This parameter takes the value either UP or DOWN. The value UP indicates scaling up of resource and DOWN indicate scaling down of resource. The third parameter Amt is the amount of resources to be scaled up or down.

**Algorithm 1** Scaling of Resource

```
Amt ← Resource in Percentage
if Resource Name = CPU and Scale Type = UP then
   Cap ← Current CPU cap value of VM
   if Cap < 100 then
      Scale up the CPU by Amt
   end if
else if Resource Name = CPU and Scale Type = DOWN
then
   Cap ← Current CPU cap value of VM
   if  Cap < 20 then
      Scale down the CPU by Amt
   end if
else if Resource Name = RAM then
   SMin ← Static min value of RAM
   SMax ← Static max value of RAM
   DMin ← Dynamic min value of RAM
   DMax ← Dynamic max value of RAM
   if  Scale Type = DOWN and DMax < SMin then
      Scale down RAM by Amt
   else if  Scale Type = UP and DMax < SMin then
      Scale up RAM by Amt
   end if
end if
```

The algorithm initially compare Resource Name. If this value is CPU, then it verify the value of Scale Type. If its value is UP, it indicate the scaling up of CPU resource. This is done with the help of cap value of credit scheduler. Similarly Resource Type DOWN indicate the scaling down of CPU resource. The amount of resources to be scaled up or down is given in the parameter Amt. If the Resource Type is RAM, then based on the value of Scale Type (UP or DOWN) RAM size is increased or decreased by Amt. This is done by changing dynamic min or dynamic max value of the RAM using Xen API.

### E. Illustration

The below graph shows the CPU utilization in a virtual machine. During the time interval of 5 to 10 minutes, the CPU utilization is stagnant, i.e. it peaks between the defined thresholds (5% and 15%), hence scaling down is performed. For the next 5 minutes, the utilization of the virtual machine is optimal. From the time interval of 20 to 25 minutes, the utilization peaks within the two defined limits for scaling up (70% and 95%). So resource is scaled up by 20% as shown in the graph by an upwardly pointing arrow. For the next 5 minutes, the virtual machine routinely operates without anomalies. The utilization of the virtual machine peaks between the defined thresholds (70% and 95%) between interval 35 to 45 minutes. At the $47^{th}$ minute, the resources are scaled up by 20%. Since the CPU utilization continues to peak even after scaling up the available resources, it becomes imminent to migrate the virtual machine to a suitable target server.

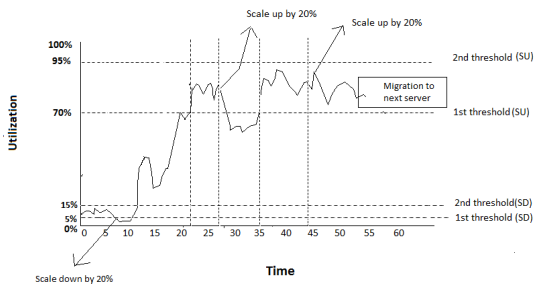


Fig. 4. CPU Utilization vs Time

## V. Implementation

Inorder to verify our algorithm we conducted a experiment using 3 servers having configuration Intel i3 2.93 Ghz processor and 2 GB RAM as shown in Figure 5. In all three servers we have installed the Xen Cloud Platform(XCP 1.6.1)[20] which contains a hypervisor to run multiple virtual machines. NFS is created on another server having configuration Intel Core 2 Duo CPU T6500 @ 2.10GHz processor, 4 GB RAM and 500 GB Secondary storage installed with ubuntu 13.04 operating system. The set up also consists of a XenCenter enabled system to monitor the virtual machines. We have implemented algorithms in shell script using Xen API.

To evaluate our algorithm, we have created two virtual machines in the first host, one virtual machine in second host and third server without any virtual machine. We run httperf [19] to find the response time of each virtual machine. We use stress load generator to generate load dynamically. The load to be generated is varied to the required amount by using Stress. We have evaluated the performance of the virtual machines against various loads in the form of CPU and RAM. We have used the response time of the virtual machines as metric in deciding the rationale behind migration of the virtual machines to relevant servers as well as scaling of the CPU resources. The response time of the applications running in virtual machine

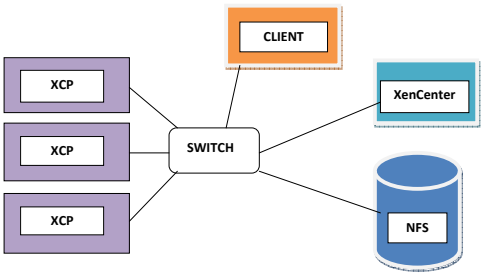


Fig. 5. Experimental set up of cloud architecture

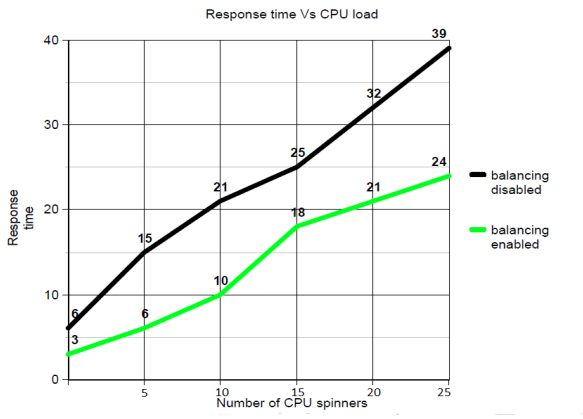


Fig. 6. Response time Vs CPU Load

with and without proposed algorithm is shown in graph above. On implementation of the algorithm, we have found out that the performance of the virtual machines are dependent on the amount of resources that have been allocated in the particular instance.

## VI. Conclusion

In this paper we have presented a resource allocation algorithm to improve the performance of the applications running in virtual machine in terms of response time and distribute the load across the servers. We conducted a experiment on Xen Cloud Platform. We have used load generating tool stress to generate the load on virtual machines. We used httperf to measure the response time of the applications running in each virtual machine. We have implemented algorithms in shell script using Xen API. Based on the experiments conducted, we have observed that the proposed algorithm, by using the features of scaling and migration has considerably improved the performance of the applications running in virtual machine in terms of response time.

## References

[1] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J. and Brandic, I. (2009). "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility." Future Generation Computer Systems, pp. 599-616.

[2] Yi, Z. and Wenlong, H. (2009). "Adaptive Distributed Load Balancing Algorithm based on Live Migration of Virtual Machines in Cloud." Proceedings of Fifth International Joint Conference on INC, IMS and IDC, pp. 170-175, IEEE.

[3] Jinhua, H., Jianhua, G., Guofei, S. and Tianhai, Z. (2010). "A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing Environment." Proceedings of International Symposium on Parallel Architectures, Algorithms and Programming, pp. 89-96, IEEE.

[4] Xiaona, R., Rongheng, L. and Hua, Z. (2011). "A Dynamic Load Balancing Strategy For cloud computing platform based on exponential smoothing forecast." Proceedings of International Conference on Cloud Computing and Intelligence Systems, pp. 220-224, IEEE.

[5] Wenhong, T., Yong, Z., Yuanliang, Z., Minxian, X. and Chen, J. (2011). "A Dynamic And Integrated Loadbalancing Scheduling Algorithm For Cloud Datacenters." Proceedings of International Conference on Cloud Computing and Intelligence Systems, pp. 311-315, IEEE.

[6] Kun, L., Gaochao, X., Guangyu, Z., Yushuang, D. and Dan, W. (2011). "Cloud Task scheduling based on Load Balancing Ant Colony Optimization." Proceedings of Sixth Annual ChinaGrid Conference, pp. 3-9, IEEE.

[7] Junjie, N., Yuanqiang, H., Zhongzhi, L., Juncheng, Z. and Depei, Q. (2011). "Virtual Machine Mapping Policy Based on Load Balancing in Private Cloud Environment." Proceedings of International Conference on Cloud and Service Computing, pp. 292-295, IEEE.

[8] Rui, W., Wei, L., Xuejie, Z. (2011)."Design and Implementation of an Efficient Load-Balancing Method for Virtual Machine Cluster Based on Cloud Service." Proceedings of International Conference on Wireless, Mobile and Multimedia Networks, pp. 321-324, IEEE.

[9] Gaochao, X., Junjie P. and Xiaodong, F.(2013)." Load Balancing Model Based on Cloud Partitioning for the Public Cloud." Proceedings of Tsinghua Science and Technology, pp. 34-39, IEEE

[10] Haozheng, R., Yihua L.,Chao Y.(2012) "The Load Balancing Algorithm in Cloud Computing Environment." International Conference on Computer Science and Network Technology, pp. 925-928, IEEE.

[11] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I. and Warfield, A. (2003)." Xen and the art of virtualization." Proceedings of the Symposium on Operating Systems Principles (SOSP), pp. 164-177, ACM.

[12] Cherkasova, L., Gupta, D. and Vahdat, A.(2007)." Comparison of the Three CPU Schedulers in Xen."' SIGMETRICS Perform. Eval. Rev., pp. 42-51, ACM.

[13] Clark, C., Fraser, K., Hand, S., Hansen, J. G., Jul, E., Limpach, C., Pratt, I. and Warfield, A. (2005). " Live migration of virtual machines."' Proceedings of the 2nd Symposium on Networked Systems Design and Implementation, USENIX Association, pp. 273-286. ACM.

[14] Hirofuchi, T., Nakada, H. and Ogawa, H. (2009)."A live storage migration mechanism over wan and its performance evaluation." Proceedings of the 3rd international workshop on Virtualization technologies in distributed computing , pp. 67-74, ACM.

[15] Hirofuchi, T., Ogawa, H. and Nakada, H., Itoh, S. and Sekiguchi, S. (2009)."A Live Storage Migration Mechanism over WAN for Relocatable Virtual Machine Services on Clouds." Proceedings of the 9th International Symposium on Cluster Computing and the Grid, pp. 460-465, IEEE.

[16] Beloglazov, A. and Buyya, R. (2010) "Energy Efficient Resource Management in Virtualized Cloud Data Centers." Proceedings of the 10th International Symposium on Cluster Computing and the Grid (CCGRID ), pp. 826-831, IEEE.

[17] Wen, H., Hai-ying, Z., Chuang, L. and Yang, Y. (2011). "Effective load balancing for cloud-based multimedia system." Proceeding of International conference on Electronic and Mechanical Engineering and Information Technology (EMEIT), pp. 165-168, IEEE.

[18] Xiaoming, G., Mike, L., Yu, M. and Marlon, P.(2009)."Supporting Cloud Computing with the Virtual Block Store System." Proceedings of Fifth IEEE International Conference on e-Science, pp. 208-215, IEEE.

[19] HP Labs. Available At: Httperf. http://www.hpl.hp.com/research/linux/httperf/ Viewed :March 2013.

[20] XCP Download, Available At: http://www.xen.org/download/xcp/ index.html, Viewed : February 2013