ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

Volume 3, Special Issue 3, March 2014

2014 International Conference on Innovations in Engineering and Technology (ICIET'14)

On 21st& 22nd March Organized by

K.L.N. College of Engineering, Madurai, Tamil Nadu, India

An Efficient Distributed Dynamic Load Balancing Algorithm for Private Cloud Environment

G.Suryadevi , D.Vijayakumar, R.SabariMuthuKumar, Dr. K .G. Srinivasagan

PG Scholar, Dept of Computer Science and Engineering, National Engineering College, Kovilpatti, Tamilnadu, India.

Assistant Professor, Dept of Computer Science and Engineering , National Engineering College, Kovilpatti, Tamilnadu, India.

Assistant Professor, Dept of Computer Science and Engineering , National Engineering College, Kovilpatti, Tamilnadu, India

Professor & Head , Dept of Computer Science and Engineering, National Engineering College, Kovilpatti, Tamilnadu, India.

Abstract— In recent years cloud computing is one of the emerging research area that encompasses virtualization, networking, storage, software and on-demand web services. Clients, distributed servers and datacenters are the fundamental components of cloud infrastructure. The number of users in cloud computing is growing in an exponential rate at every day. Large number of user requests tries to allocate the resources for many applications which lead to high load on cloud server. To reducing the heavy load on server, the virtual machines are allocated for resources based on priority. In this paper the distributed dynamic priority based algorithm is used for balancing the load on instances effectively and to improve the system consistency, minimum response time and increase the throughput. Allocating the resources on virtual machines based on priority achieves the better response time and processing time. Load balancing ensures all instances in a node in the networks do the equal amount of work at any instant of time. Priority based resource provision to improve the utilization of resources and reducing response time of cloud services.

Keywords—Cloud computing, Resource allocation, Priority based Scheduling, Load balancing.

I. INTRODUCTION

Cloud computing [1], a relatively new technology, has been gaining immense popularity over the last few years where user can rent software, hardware, infrastructure and computational recourse as per user basis. The number of cloud users has been growing exponentially and apparently scheduling of virtual machines in the cloud becomes an important issue to analyze. Users can submit their jobs into cloud for computational processing or leave their data in cloud for storage. Different users have different Quality of Service requirement. Cloud scheduler must be able to schedule the tasks such a way that cloud provider can gain maximum benefit for his service and QoS requirement of user's job is also satisfied.

The cloud has three service models. In the Software as a Service (SaaS) model, the software or the applications are hosted over the cloud and are made available to the customers based on the pay-as-per-use model. Google Apps and Salesforce are examples of this model. The Platform as a Service (PaaS) model provides a hosting environment for the client's application. Examples for PaaS model are Google App Engine and Amazon Web

Services. The Infrastructure as a Service (IaaS) model lets the client to dynamically scale up or scale down the resources like processing power, storage capacity, network bandwidth etc. Example: Amazon EC2, Eucalyptus, etc.

Cloud providers are able to attain the agreed Service Level Agreement (SLA), by scheduling resources in efficient manner and by deploying application on proper Virtual Machine as per SLA [2] objective. SLAs are offered by IaaS providers to express their commitment to delivery of a certain QoS. An SLA usually include availability and performance guarantees. Most IaaS providers focus their SLA terms on availability guarantees, specifying the minimum percentage of time the system will be available during a certain period.

Load balancing [1] problem arises in many applications and they play a special role in the operation of parallel and distributed computing systems. Load-balancing [9-11] deals with partitioning a program into smaller tasks that can be executed concurrently and mapping each of these tasks to a computational resource such a processor (e.g., in a multiprocessor system) or a computer (e.g., in a computer network). By developing strategies that can map these tasks to processors in a way that balances out the load, the total processing time will be reduced with improved processor utilization. Load balancer performance shown in Figure.1.

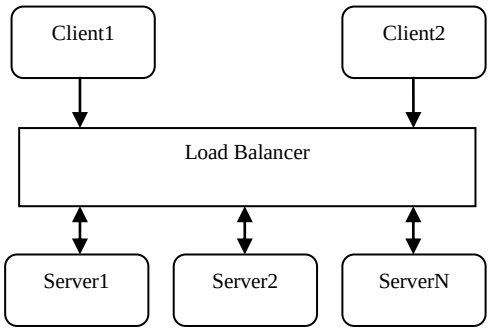

Figure. 1. Load balancer

Load balancing is a process of reassigning the total load to the individual nodes of the collective system to make resource utilization effective and to improve the response time of the job. Simultaneously removing a condition in which some of the nodes are over loaded while some others are idle. Load balancing algorithms can be classified in two different ways: static load balancing algorithms and dynamic load balancing algorithms. A static load balancing algorithm does not take into account the previous state or behavior of a node while distributing the load. On the other hand, a dynamic load balancing algorithm checks the previous state of a node while distributing the load.

II. RELATED WORK

In [3] proposed Haize concept for resource allocation. Infrastructure as a Service clouds use two resource allocation policies. An Immediate allocation policy allocates the resources if available, or else the request is rejected. Best-effort policy also allocates the requested resources if available otherwise the request is placed in a FIFO queue. Haizea uses resource leases as resource allocation abstraction and allocating Virtual Machines. It is used the immediate, best effort, advanced reservation and deadline sensitive policies. This work provides a better way to support deadline sensitive leases in Haizea while minimizing the total number of leases rejected by it.

The best effort policy is used for resource allocation in eucalyptus. It maximizes resource utilization and acceptance of requests compared to the existing algorithm of Haizea.

In [4] proposed five database allocation algorithms for distributing databases to nodes in the cloud platform. Each database node in the cloud platform often has to service multiple number of database application system. Under the resource limitations of the node, evenly distributed databases into node are an important issue needed to be addressed. The database sizes and the number of databases must be taken into account for workload balancing among database hosts. In this private cloud to consider the databases sizes and forward the request to the job scheduler.

In cloud computing the scheduling of virtual machine requests is an important issue. The requested tasks can be completed in a minimum time according to the user defined time. In [5] to evaluate a scheduling algorithm that is an efficient technique for scheduling virtual machines between servers. In this the scheduling technique results are compared. Comparing these techniques the priority based scheduling algorithm improves the resource utilization and reduce the waiting time.

Load balancing of non preemptive independent tasks on virtual machines is an important aspect of task scheduling in clouds. Some VMs are overloaded and remaining VMs are under loaded with tasks for processing, the load has to be balanced to achieve optimal machine utilization. The honey bee behavior algorithm [6] balances the priorities of tasks on the machines in a way that the amount of waiting time of the tasks in the queue is minimal. Then there is a significant improvement in an average execution time and reduction in waiting time of tasks on queue.

In priority based resource allocation the resource provisioning is done by considering the SLAs and with the help of parallel processing. Considering multiple SLA parameter and resource allocation by preemption mechanism for high priority task execution can improve the resource utilization in Cloud. The algorithm [8] considered multiple SLA parameters. That is memory, network bandwidth, and required CPU time. In this the various scheduling algorithms present for scheduling virtual machines and also proposed a dynamic priority based scheduling algorithm. The purpose of this algorithm is to calculate the priorities for incoming requests during the execution time. The objective of this algorithm is to adapt to dynamically changing progress of the incoming request.

III. PROPOSED SYSTEM ARCHITECTURE

The Resource allocation for load balancing is designed on Eucalyptus private cloud. It has the better performance for dynamic load balancing. Eucalyptus is

comprised of six components: Cloud Controller, Walrus, Cluster Controller, Storage Controller, Node Controller and an optional VMware Broker Other than the VMware Broker, each component is a stand-alone web service. Cloud Controller acts as a Front- end interface for users. It knows overall status of cloud resources and controllers. Cluster Controller acts an Administers networking resources and compute nodes.

Walrus Controller Storage for virtual machine images to use by compute nodes. Storage Controller acts as a persistent storage device which can be mounted inside active virtual machines .Compute Nodes provides the execution environment for virtual machines in the cloud. Load Balancing is performed by Cluster Controller. It acts as the front end for a cluster within a Eucalyptus cloud and communicates with the Storage Controller and Node Controller. The CC manages instance execution and Service Level Agreements per cluster.

This proposed architecture consists of three modules. First module is job scheduler. The client submits their requests on server which is present in the IaaS cloud environment. When an available resource task is assigned to a cloud, first resource availability in this cloud will be checked by job scheduler. A job scheduler records the execution schedule of all resources using a slot and to check the resource availability in this cloud.

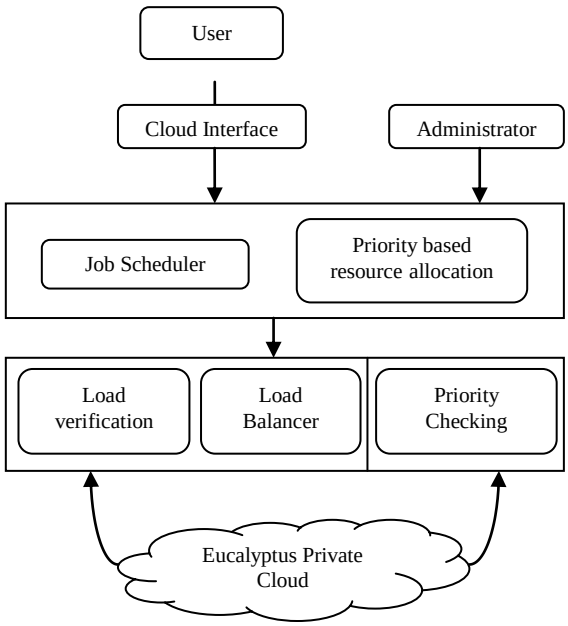

Figure. 2. Cloud Architecture

Second module is resource allocation. In this module the incoming requests are directed to the virtual machines. Third module is load balancer. Load balancer will allow incoming request to be routed into servers that host the web application.

A. Job scheduler

The client submit their requests on server which is present in the IaaS cloud environment, the requests are placed in the queue and send to the server. In scheduler [15] the requests are equally distributed to the nodes. When an available resource task is assigned to a cloud, first resource availability in this cloud will be checked by the job scheduler. A job scheduler record execution schedule of all resources using a slot. The job scheduler is used to simulate from users for virtual machines. In this module we have considered a web service which has different task and this web service application are placed in the server and made the user to access these applications from the client machine

B. Resource allocation

In resource allocation we have to select the available resources depending upon the specification of our task by using load balancing condition. When the user sends the request to server, the server identifies [13] the request and redirects to the particular virtual machine. The virtual machine processes the requested task and sends the corresponding response to the client request. If any one of virtual machine have overloaded, apply load balancing policies and we can redirect the incoming request to the other virtual machines to balance the load among cloud environment.

C. Load balancing workflow

In load balancer we have created a web application in which multiple numbers of users is allowed to access the virtual machines efficiently from the client side. Initially the different users use the application and submit the task to the server. The user's requests are placed in the task queue and it is sent to the load balancer. The load balancer checks for the available virtual resources which are connected to it and it also has the status information of every virtual machine which is connected to it. The status information represents the status of every virtual machine, such as whether it is in busy state or in available state. Based on the status of virtual machines the task can be allocated to the virtual resources. For checking the availability of virtual machine we have to update the status of the virtual machine shown in Figure. 3.

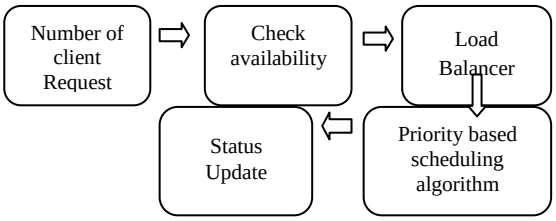

Figure. 3. Load balancing workflow

The status update is represented in the form of table to represent the status of CPU speed, memory and disk of different virtual machines. The status update is used to represent the availability of the virtual machines whether it is in busy state or in the idle state so that it can be allocated for other task if it is free. The output of this load balancing workflow is to make use of available resource efficiently. When the incoming request is overloaded then the admin set the priority to each request. Multiple numbers of requests are submitted in the server and set the priority to each request. The priority is based on capacity and load factor. For using priority the client requests are redirected to the nodes.

D. Dynamic Priority Based Algorithm

A dynamic load balancing algorithm checks the previous state of a node while distributing the load. The proposed algorithm uses dynamic priority for the requests based on which the virtual machines are scheduled. It schedules the VMs to the requests depending upon their

priority value, which varies dynamically based on their load factor. This dynamic priority concept leads to better utilization of the resources. Priority of a request is assigned depending upon its capacity and the load factor.

Algorithm: Priority (PBSA)
Input: N number of client Request
Output: Balance the Request

Algorithm check priority schedule
1. Get the available VMList
   //find the appropriate VMList from job scheduler
2. If priority1 is not assign to request
3. Priority1=max available VM
4. Else if priority1 is set
5. Turn ON priority1
6. Used VMList
7. If VM is not used VMList then
   Add VM to used VMList
8. Deploy request on new VM
9. If priority N is not assign goto step1
10. Assign VM to ClientRequests
11. If assigned the requests then return Successful
12. End for
13. Return available VMList

Priority [13] based scheduling Algorithm is intended to be used by organizations need to implement small to medium sized local clouds. This algorithm should scale to larger sized clouds because one of the main contributions of the cluster controller is load balancing compute nodes.

TABLE I. INITIAL PRIORITY VALUE

Request	Arrival Time(ms)	Execute Time(ms)	Priority (P)	Service Time(ms)
R0	0	5	3	0
R1	1	3	2	3
R2	2	8	1	8
R3	3	6	4	16

Each process is assigned as a priority. Process with highest priority is to be executed first and so on. Processes with same priority are executed on first come first serve basis. Priority can be decided based on memory, time or any other resource requirement shown in Table.1.

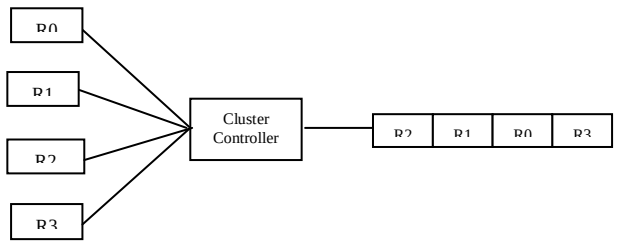

Figure. 4. Priority based scheduling

The advantage of using dynamic load balancing is that if any node fails, it will not stop the progress of the system; it will only affect the system performance. When compared to a centralized environment in distributed dynamic load balanced system, the nodes can work together. However, selecting a suitable server needs real time communication with the other nodes of the network and hence, generates multiple numbers of messages in the network. Dynamic load balancer uses the load balancing policies for keeping track of updated information.

IV. EXPERIMENTAL RESULTS

A. Experiment setup

Eucalyptus [12] has developed an integrated solution to address the challenges of cloud configuration and management. The Eucalyptus service components make it faster and easier for users. We deploy a Eucalyptus cloud across our enterprise's on-premise data center. Eucalyptus allows you to use our own collections of resources (hardware, storage, and network) using a self-service interface on an as-needed basis. The eucalyptus enterprise allows sensitive data to remain secure from external intrusion behind the enterprise firewall. We install Eucalyptus on the Linux distribution of Cent OS 6.

The job scheduler is used to simulate requests from users for virtual machine instances. We chose a schedule that closely resembles requests for cloud resources. Jobs in this experiment come as requests from users for virtual machines. There are five different sized virtual machines and users are not limited in the size or number of instances they can request. An error is returned to the user if there is not enough space to host a requested virtual machine. When virtual image is launched, the execution resulting of running system is called an instance. An instance is a virtual machine. A virtual machine is basically an operational private computer. It contains an operating system, web applications, network accessibility, and disk drives. Eucalyptus agrees you to run instances from both Linux-based images and Windows-based images.

TABLE II. LOAD CONFIGURATION

Instance Type	Virtual CPU	Disk Size(GB)	Memory (MB)
m1.small	1	5	256
m1.medium	1	10	512
m1.large	2	10	512
m1.xlarge	2	10	1024
m3.xlarge	4	15	2048
m3.2xlarge	4	30	4096

The virtual machine type defines the available resources such as number of CPUs, memory size, and disk capacity. Eucalyptus has pre-defined VM types. We can change the quantity of resources associated with each of the five VM types, but cannot change the name of the VM

types or the number of VM types available. If you modify the sizes they must be well-ordered. This means the CPUs, memory size and disk capacity of the next VM type must be greater than or equal to the size of the preceding VM type.

B. Results

The proposed dynamic priority based algorithm implemented like graphical representation. Java language is used for implementing load balancing on servers. Assuming the application is deployed in eucalyptus private cloud. The deployment procedures are described in the following table.

TABLE II. EUCA DEPLOYMENT DETAILS

First, create the security group, and authorize ports for SSH (22), ping (ICMP), and LoadBalancer admin UI (444).

```
#euca-create-group      loadbalancer      -d
"Security Group for Loadbalancer"
GROUP   loadbalancer   Security   Group   for
Loadbalancer

# euca-authorize -P tcp -p 22 -s 0.0.0.0/0
loadbalancerGROUP loadbalancer
PERMISSION loadbalancer ALLOWS tcp 22 22
FROM CIDR 0.0.0.0/0

# euca-authorize -P tcp -p 444 -s
0.0.0.0/0 loadbalancer GROUP loadbalancer
PERMISSION loadbalancer ALLOWS tcp 444 444
FROM CIDR 0.0.0.0/0

# euca-authorize -P icmp -t -1:-1 -s
0.0.0.0/0 loadbalancer GROUP loadbalancer
PERMISSION loadbalancer ALLOWS icmp -1 -1
FROM CIDR 0.0.0.0/0

# euca-describe-group loadbalancer GROUP
345590850920 loadbalancer Security Group
for Loadbalancer
PERMISSION 345590850920 loadbalancer
ALLOWS tcp 22 22 FROM CIDR 0.0.0.0/0
PERMISSION 345590850920 loadbalancer
ALLOWS tcp 444 444 FROM CIDR 0.0.0.0/0
PERMISSION 345590850920 loadbalancer
ALLOWS icmp -1 -1 FROM CIDR 0.0.0.0/0

#euca_describe_images
Eki-FA8F3EA5   Eri-IE6544D0   Emi-40C93CC2

#euca-run-instances-kerna eki- FA8F3EA5-
ramdisk
Eri- IE6544D0
```

Launch under the load balancer group.

```
# euca-run-instances -k admin emi-
826F3A01 -g loadbalancer
```

Run "euca-describe-instances" to see when the instance gets into a running state:

```
# euca-describe-instances i-BF33466F
RESERVATION r-7D23418A 345590850920
loadbalancer

INSTANCE i-BF33466F emi-826F3A01 euca-192-
168-55-105.wu-tang.euca-hasp.eucalyptus-
systems.com euca-10-106-45-14.wu-
tang.internal running admin0

m1.small 2012-11-28T17:32:44.184Z enter-
the-wu eki-B420361A eri-31D33872

monitoring-disable euca-192-168-55-
105.wu-tang.euca-hasp.eucalyptus-
systems.com euca-10-106-45-14.wu-
```

Launch instances on load balancer group and run the instances. Once, the instance is up, access the Load Balancer Admin UI shown in Figure.5.

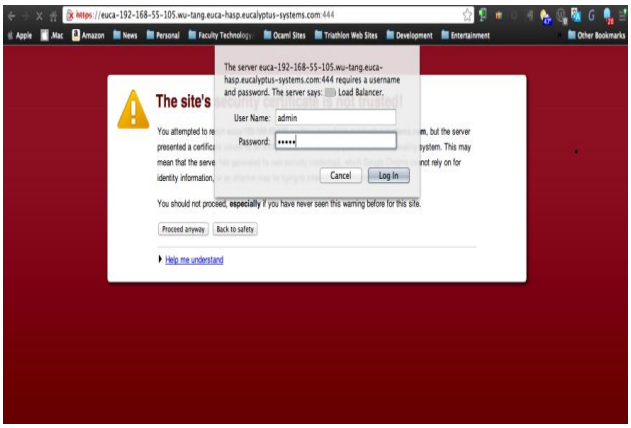

Figure.5. Load Balancer Admin UI Login

In private cloud environment VM server details are monitored and the service usage of client is updated on server queue shown in Figure.6.

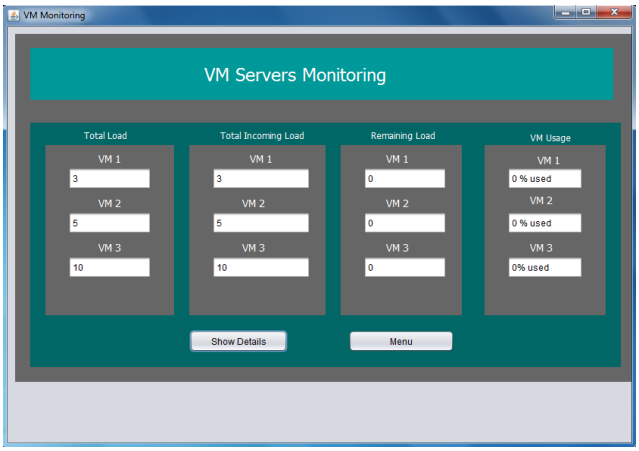

Figure.6.VM server status updation

The memory usage of VM server is shown in Figure.7.Using load baancer admin UI the memory usage isdescribed. In this x-axis get the resource utilization and y-axis get the number of virtua mchines.

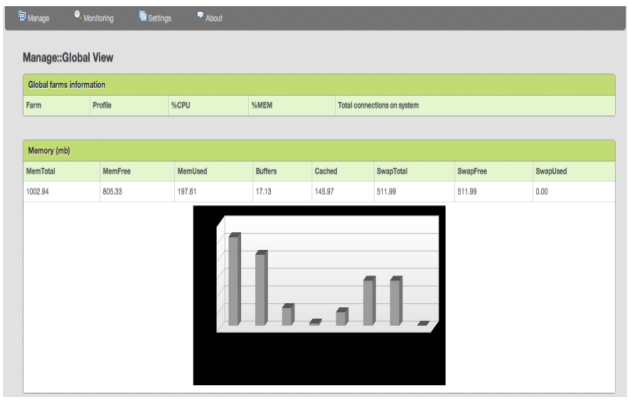

Figure.7. Utilization Report

The proposed dynamic priority based algorithm makes a better utilization of the available resources in cloud is shown in the Figure.8and Figure.9.

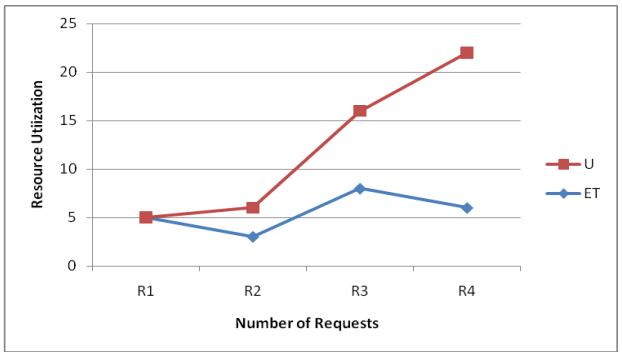

Figure. 8. Average Execution Time of requests

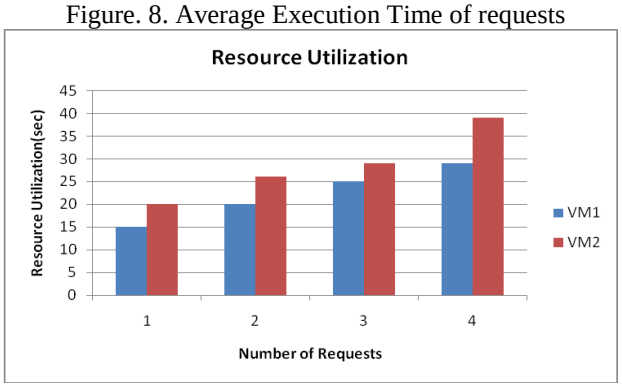

Figure. 9. Resource Utilization

V. Conclusion

A new algorithm for the scheduling of virtual machines in Eucalyptus platform has proposed. This scheduling algorithm can be extended to suit other cloud platforms also. Here, all the incoming requests from the clients have been automatically redirected to virtual machines based on priority and also the new virtual machines are created and redirected when the requests are overloaded. The simulation result shows the better utilization of virtual machines and reduces the execution time.

References

[1] Tharam Dillon and Chen Wu and Elizabeth Chang,"Cloud Computing: Issues and Challenges" *24th IEEE International Conference on Advanced Information Networking and Applications* 2010.

[2] Lejeune.J, Arantes.L, Sopena.J," Service Level Agreement for Distributed Mutual Exclusion in Cloud Computing" *12th IEEE/ACM International Conference on May 2012,pp 180-187.*

[3] Amit Nathani, Sanjay Chaudhry, Gaurav Somani (2012),'Policy based Resource Allocation in IaaS Cloud', *Future Generation Computer Systems 28, 94-103.*

[4] Yu-lung Lo and Min-Shan Lai (2013),'The Load Balancing of Database Allocation in the Cloud ', *International Multi Conference of Engineers and Computer Scientists*, vol.1.

[5] Amin Mehranzadeh and Seyyed Mohsen Hashemi (2013),'A Novel- Scheduling Algorithm for Cloud Computing based on Fuzzy Logic', *International Journal of Applied Information Systems (IJAIS)*, vol.5, pp 28-31.

[6] Dhinesh Babu L.D. P. Venkata Krishna (2013), '*Honey bee behavior inspired Load Balancing of tasks in cloud computing environments*', ELSEVIER, vol.13, pp 2292-2303.

[7] Anandharaj.V, Ms.A.Geetha (2013),'A Dynamic and Efficient Resource Management by using Heuristics Algorithm', *International Journal of Engineering Sciences & Research*, pp 1247-1250.

[8] CS.Pawar, RB.Wagh(2012).*'Priority Based Dynamic Resource Allocation in Cloud computing' Cloud and Services Computing (ISCOS)*, IEEExplore.ieee.org.

[9] Gaochao Xu,Junjie Pang, and Xiaodong Fu, Tsinghua "Load balancing model based on Cloud partitioning for the public cloud", *science and technology* 2013,vol.18, pp34-39.

[10] Chun-Cheng Lin,Hui-Hsin Chin, and Der-Jiunn Deng, IEEE members, "*Dynamic Multiservice Load balancing in Coud –Based Mutimedia System*", IEEE Systems Journal,2013.

[11] HungChang, Hsueh-Yi Chung,Haiying Shenand Yu-Chang Chao , " *Load Rebalancing for Distributed File System in clouds*", IEEE Transactions on Parallel And Distributed Systems.2013, vol.24, pp951-961.

[12] Jeffrey M. Galloway, Karl L. Smith, Susan S. Vrbsky "Power Aware Load Balancing for Cloud Computing", *Proceedings of the World Congress on Engineering and Computer Science* 2011 Vol I.

[13] E. Iniya Nehru, S. Sujatha, P. Seethalakshmi and N. Sridharan, "Priority Based Load Balancing in Cloud for Data Intensive Applications" *IOSR Journal of Engineering* 2012. vol.2, pp 65-71.

[14] Muhammad Abdulla Adnan, Ryo Sugihara and Rajesh Gupta, "*Energy Efficient Geographical Load Balancing via Dynamic Deferral of Workload*", University of California San Diego, CA, USA; Amazon.com, 2012.

[15] Claudia Rosas,Anna Sikora,ET.AL, " *Improving Performance On Data-Intensive Applications Using A Load Balancing Methodology Based On Divisible Load Theory* " , Springer Science Business Media,12 June 2012.