Computer Engineering and Intelligent Systems
ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)
Vol.6, No.4, 2015

Utilizing Divisible Load Scheduling Theorem in Round Robin Algorithm for Load Balancing In Cloud Environment

Dr. Thomas Yeboah [1], Prof. Odabi I. Odabi[2], Mr. Christopher Ayaaba A. Abilimi[3]

1. Christian Service University, Department of Computer Science, P.O.Box 3110,Ghana.
2. WellSpring University, Benin-City, Nigeria
3. Christian Service University, Department of Computer Science,P.O.Box 3110,Ghana

tyeboah@csuc.edu.gh

ABSTRACT

Cloud Computing is a newly paradigm in computing that promises a shift from an organization required to invest heavily for limited IT resources that are internally managed, to a model where the organization can buy or rent resources that are managed by a cloud provider, and pay peruse. With the fast growing of cloud computing one of the areas that is paramount to cloud computing service providers is the establishment of an effective load balancing algorithm that assigns tasks to best Virtual Machines(VM) in such a way that it provides satisfactory performance to both, cloud users and providers. Among these load balancing algorithms in cloud environment Round Robin (RR) algorithm is one of them.

In this paper firstly analysis of various Round Robin load balancing algorithms is done. Secondly, anew Virtual Machines (VM) load balancing algorithm has been proposed and implemented; i.e. 'Divisible Weighted Round Robin(DWRR) Load Balancing Algorithm'. This proposed load balancing algorithm utilizes the Divisible Load Scheduling Theorem in the Round Robin load balancing algorithm.

In order to evaluate the performance of this proposed algorithm (DWRR) the researcher used a simulator called CloudSim tool to conduct a test on the performances between the proposed algorithm (DWRR) and the types of Round Robin algorithms. After a thoroughly comparison between these algorithms, the results showed that DWRR outperforms the various types of Round Robin(Weighted Round Robin and Round Robin with server affinity)algorithms in terms of execution time (makespan) with the least complexity.

Keywords: Scheduling Algorithm, performance, cloud computing, load balancing algorithm, Divisible Load scheduling Theory.

1. INTRODUCTION

Cloud Computing can be considered as the delivery of computing as a service rather than a product, whereby shared resources, software, and information are provided to computers and other devices as a utility (like the electricity grid) over a network (typically the Internet). This is basically "on-demand" service. It means whenever we need for some applications or some software, we demand for it and we immediately get it. We have to pay only that we use. This is the main principle behind cloud computing.

Cloud Computing has basically two parts, the First part is **Client Side** and the second part is Server **Side**. The Client Side requests to the Servers Side and the Server Side responds to the Clients. The request from the client firstly goes to the Master Processor of the Server Side. The Master Processor are attached to many Slave Processors, the master processor sends that request to any one of the Slave Processor which have free space. The main concern to these issues depend on the establishment of an effective load balancing and efficient scheduling algorithm than can be implemented in cloud environment to ensure that all Virtual Machines are busy in their assigned job and none of the Virtual Machines gets Idle. Load balancing therefore is the process of distributing the load among various virtual machines of a cloud computing system to improve both resource utilization and job response time while also avoiding a situation where some of the virtual machines are heavily loaded while other virtual machines are idle or doing very little work. Load balancing ensures that all the virtual machines in the cloud system or every node in the network does approximately the equal amount of work at any instant of time.

In view of this the main objective of this research is to develop an effective load balancing and scheduling algorithm by utilizing Divisible Load Scheduling Theorem (DLST) in the Round Robin Algorithm in order to improve the different performance parameters like throughput, latency etc. for the clouds of different sizes.

2. RELATED WORKS

Efficient job scheduling to datacenters in cloud computing environment is done with the main aim of achieving load balancing. Due to novelty of cloud computing a few research activities have been conducted in terms of finding an allocation of jobs to datacenters in cloud environment.

2.1 Round Robin and Its Variants

Round Robin algorithm is one of the most popular and simple algorithms used to implement load balancing in cloud environment. Due to its simplicity and popularity in nature; there has been lot of research carried out to improve performance of this algorithm and therefore there are so many variants of this algorithm.

Singh etal [10] proposed a modified round robin (MRR), which is superior than simple Round Robin (RR) and has less waiting response time, usually less pre-emption and context switching thereby reducing the overhead and saving of memory space. In Modified Round Robin (MRR) algorithm, Time Slice (TS) is calculated based on (range× total no of process (N)) divided by (priority (pr) × no of process in queue (p)). Range is calculated by (maximum burst time +minimum burst time) divided by 2. This algorithm improved a scheduling up to some extent but not much. In paper [7] a Priority Based Round Robin was introduced which uses a calculated intelligent time slice. In their research the calculated intelligent time slice allocates a different Time Quantum (TQ) to the individual process according to the priority given to each process. Their results show that PBDRR performs better than MRR algorithm in terms of reducing the number of context switches, average waiting time and average turnaround time. Subasish et. al [2] also proposed a load balancing based algorithm on Time Slice Priority Based Round Robin (TSPBRR). In their algorithm they initially took Time Quantum (TQ) as half of the first request of the Burst Time (1/2BT). It is observed in their research that this algorithm gives considerable improvement over that of MRR but still needs more improvement in response time. A load balancing framework, Weighted Round Robin (WRR) algorithm that allocates request to various virtual machines in a round robin fashion based on the weight without considering the current load on each virtual machine was also proposed in [9]. In their proposed algorithm if a request is allocated to a busy VM it can lead to wait few request wait for a long time that can increase response time. Komal et. at [4] introduced a round robin with server affinity algorithm. This algorithm uses two data structures: Hash map – This stores the entry of the last VM and VM state list- It stores the allocation status of each VM. In their algorithm when a request is made it first look into the Hash map and if the VM is available, there is no need to run the Round Robin I this case; therefore can save a significant amount of time.

2.2 Divisible Load Scheduling Theory

In [6] Divisible load scheduling involves the study of optimal distribution of arbitrarily divisible loads among the number of processors and links in a system with the aim to minimize the total computation time of processing entire workload. The time required for completing a task with one processor is very high [6]. Due to this the task is divided into sub-tasks that can be processed sequentially or parallel.

In a Divisible Load Scheduling Theory the total load is of the arbitrarily divisible kind that can be partitioned into fractions of loads to be assigned to all the master and slave computers in the cloud.

In this case each master computer first assigns a load share to be measured to each of the corresponding N slave computers and then receives the measured data from each slave. Each slave then begins to measure its share of the load once the measurement instructions from the respective master have been completely received by each slave.

3. PROPOSED ALGORITHM - DWRR

3.1 System Model

The Divisible Weighted Round Robin with server Affinity is an improvement over the Weighted Round Robin VM load balancing Algorithm and Round Robin with server affinity algorithm. It is observed that Weighted Round Robin Algorithm does not save the state of the previous allocation of a VM to a request from given Userbase while the same state is saved in the proposed algorithm. More so the time required for completing a task with one processor is very high [6] as compare to multiple processors. This is a limitation to both Weighted Round Robin and Round Robin with server Affinity.

The Divisible Weighted Round Robin Algorithm is implemented by dividing the task into arbitrary sub-tasks. Each master computer has several slave computers. Out of these some are executed sequentially and some are executed parallel. So the total time period for completing the task decreases and hence the performance increases. In this algorithm each master computer is assigned a weight depending on the processing power as discussed in Weighted Round Robin (WRR) algorithm. To these VMs of different processing powers; the tasks are assigned or allocated to the most powerful VM and then to the lowest and so on according to its weight and its availability of each masters and the previous allocation of a VM to a request from given Userbase is also saved as also discussed in Round Robin with server affinity.

3.2 The Algorithm

The designed algorithm used is as follows:

1. Divide the task into arbitrary sub-tasks for each master computer.
2. Create VM's for each sub-task according to their processing power.
3. Initialize all the VM's allocation status to AVAILABLE and that of Hash map with no entries.
4. When a request is made it goes into the VM load balancer of each sub-task to identify a VM with least load for all the sub-tasks if the Hashmap contains entry for userbase.
5. Then the load is assigned to the VM with higher weight for all the sub-tasks.
6. Loaded id is then sent to the various data centers of each sub-task.
7. Execute Request on VM for each sub-task.
8. Deallocate VM and set VM_State to AVAILABLE;
9. The Weighted VM Load Balancer updates the allocation table by decreasing the allocation count for the VM by one.
10. Repeat 5

3.3 The flow chart of the Algorithm

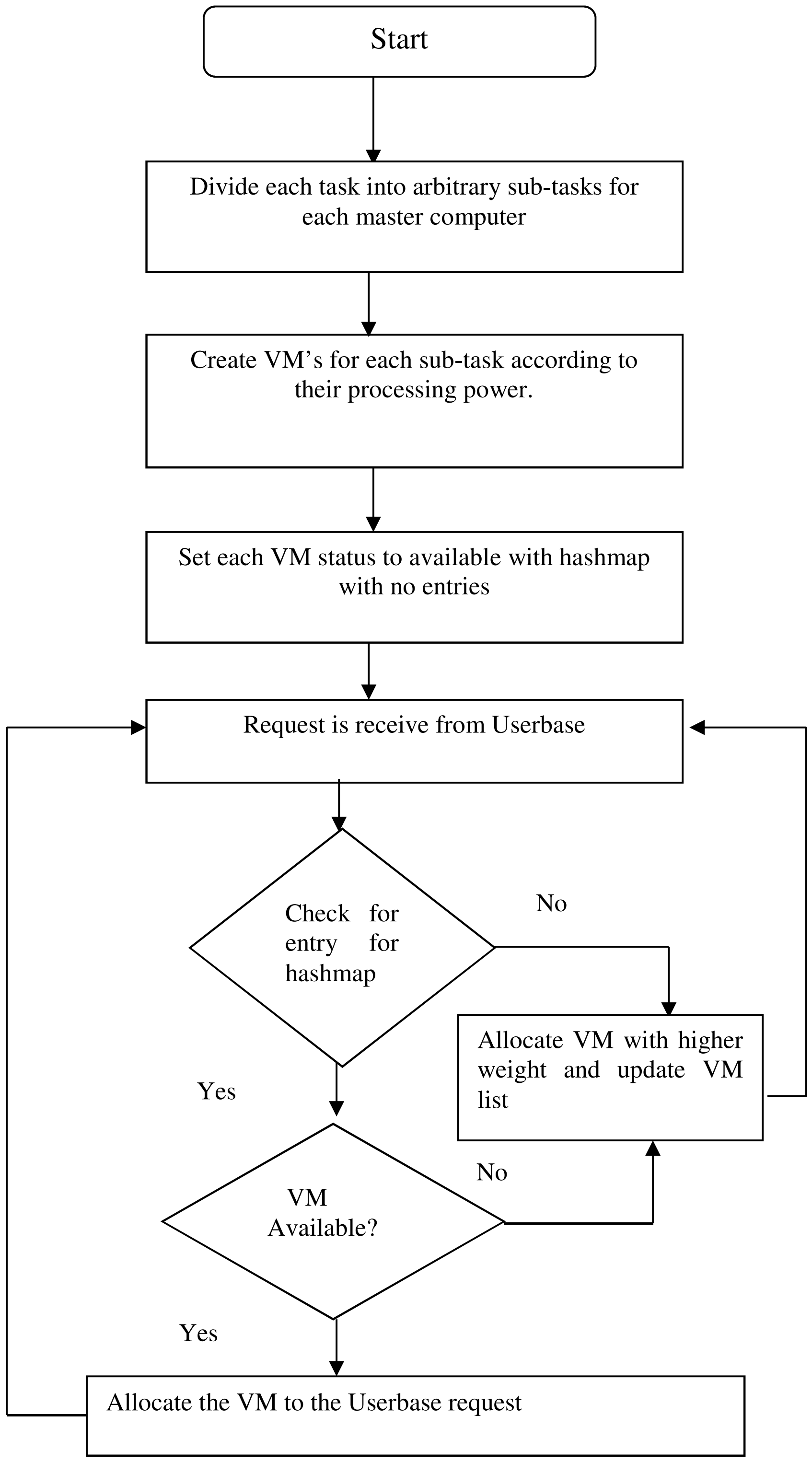
Start
Divide each task into arbitrary sub-tasks for each master computer
Create VM's for each sub-task according to their processing power.
Set each VM status to available with hashmap with no entries
Request is receive from Userbase
Check for entry for hashmap
No
Allocate VM with higher weight and update VM list
Yes
No
VM Available?
Yes
Allocate the VM to the Userbase request

Fig 1-1 : Flowchart of proposed algorithm

3.4 Measurement of Reporting Time

At the beginning when time t=0, all the slaves are in their idle state. Each slave in the cloud environment will start receiving instructions from their corresponding master computer when t = t1 as can be seen in fig 1-2 .

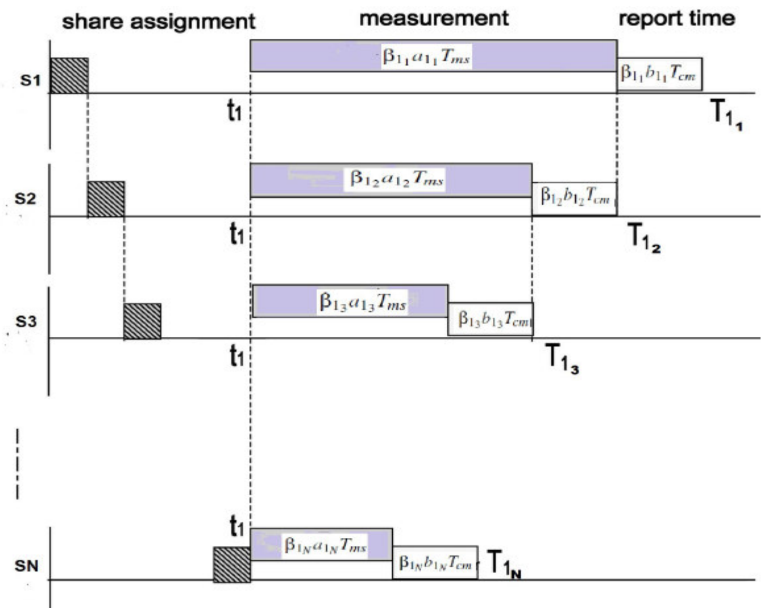
share assignment
measurement
report time
S1
S2
S3
SN
t1
$\beta_{1_1} a_{1_1} T_{ms}$
$\beta_{1_1} b_{1_1} T_{cm}$
T_{1_1}
$\beta_{1_2} a_{1_2} T_{ms}$
$\beta_{1_2} b_{1_2} T_{cm}$
T_{1_2}
$\beta_{1_3} a_{1_3} T_{ms}$
$\beta_{1_3} b_{1_3} T_{cm}$
T_{1_3}
$\beta_{1_N} a_{1_N} T_{ms}$
$\beta_{1_N} b_{1_N} T_{cm}$
T_{1_N}

Fig 1-2: Timing diagram of master computer and N-Slaves (adopted from [8])

From fig 1-2, the following parameters can be obtained:

- The total time elapses between time t=0 and reporting time of each slave of a master computer be $\mathbf{T_{ki}}$,
- The portion of load that is assigned to a slave **i** by master computer **k** be $\boldsymbol{\beta_{ki}}$,
- A constant that is inversely proportional to the measuring speed of slave **i** in the cloud be $\mathbf{a_{ki}}$**.**
- A constant that is inversely proportional to the communication speed of link i in the cloud be $\mathbf{b_{ki}}$.
- Measurement intensity constant be $\mathbf{T_{ms}}$. This is the time it takes the ith slave to measure the entire load when $\mathbf{a_{ki}} = 1$.
- Communication intensity constant be T_{cm}. This is the time it takes to transmit all of the measurement load over a link when $\mathbf{b_{ki}} = 1$.

  Some of the above used parameters and notations are taken from [8].Therefore from fig 1-2, the value of $\mathbf{T_{ki}}$ for the first master computer with N slaves can be calculated as:

$$\mathbf{T_{1N}} = t1 + \boldsymbol{\beta_{1N}}\, \mathbf{a_{1N}}\, \mathbf{T_{ms}} + \boldsymbol{\beta_{1N}}\, \mathbf{b_{1N}}\, \mathbf{T_{cm}}$$

In this case the total measurement time for the entire master computer s would be assumed a normalize unit load. This means that each master slave would handle load of (1/K). So the it means for the first master computer

$\beta_{11} + \beta_{12} + \ldots + \beta_1 n\text{-}1 + \beta_{1n} = 1/k$

It can be deduced from the timing diagram in figure 1-2 that β11a11Tms = β 12a12Tms+ β 12b12Tcm

β12a12Tms = β 13a13Tms+ β 13 β 13Tcm

.

.

.

β1n-1a1n-1Tms=β1na1nTms+ β1n β 1nTcm

A general expression for the above set of equations can be written as

$$\beta 1_i = \prod_{j=2}^{i} s1_j \beta 1_1$$

Again,

$$\beta 1_1 + \sum_{i=2}^{N} \prod_{j=2}^{i} s1_j \beta 1_1 = 1/K$$

From the above **β1i**can be written as

$$\beta 1_i = \frac{\prod_{j=2}^{i} s1_j}{K(1 + \sum_{i=2}^{N} \prod_{j=2}^{i} s1_j)}$$

Where i=1, 2, 3…, N

Hence for a master computer r the general equation for the portion of load assigned to each slave can be written as

$$T_{f_r} = t_1 + \frac{(a_{r_1} T_{ms} + b_{r_1} T_{cm})}{K(1 + \sum_{i=2}^{N} \prod_{j=2}^{i} s_{r_j})}$$

Therefore the minimum reporting time for homogeneous network can be given as

$$T_{f_1} = t_1 + \frac{(a_1 T_{ms} + b_1 T_{cm})(1 - s_1)}{K(1 - s_1^N)}.$$

4. SIMULATION ENVIRONMENT AND RESULT COMPARISON.

4.1 Simulation Environment

To analyze and compare the proposed algorithm(Divisible Weighted Round Robin) with the two algorithms (Weighted Round Robin and Round Robin with a sever affinity) the researcher used Cloudsim tool for the simulations. The cloud environment set up generated was having following configurations.

Parameters	**capacity**
VM memory	512
VM image size	10,000
Data center – Architecture	X86
Data center – OS	Windows Xp
Data center – Number of Machines	5
Data center – Processor speed	100MIPS
Data center – Number of processors per machine	5
Data center – Available BW per Machine	10,000
Data center – Memory per Machine	2GB

Table 1-1: cloud environment setup specifications

4.2 Comparison of Results

Fig1-3, Fig1-4 and Fig1-5give the results of the response time and Data Centre processing time for Weighted Round Robin (WRR), Round Robin with a Server Affinity (RR with Server Affinity) and proposed algorithm (Divisible Weighted Round Robin (DWRR).

Overall Response Time Summary			
Time	Avg(ms)	Min(ms)	Max(ms)
Overall ResponseTime	664.76	33.29	42207.93
Data Center Processing Time	257.38	1.36	41698.42

Fig 1-3: Weighted Round Robin Results

Overall Response Time Summary			
Time	Avg(ms)	Min(ms)	Max(ms)
Overall ResponseTime	610.51	46.10	30306.31
Data Center Processing Time	289.71	2.76	36920.16

Fig 1-4: Round Robin with Server Affinity Results

Overall Response Time Summary			
Time	Avg(ms)	Min(ms)	Max(ms)
Overall ResponseTime	576.03	46.10	30306.31
Data Center Processing Time	256.18	1.82	30078.29

Fig 1-5: Divisible Weighted Round Robin Results (Proposed Algorithm)

Comparison of overall response time and Data Centre processing time is shown in the graph shown in fig 1-6.

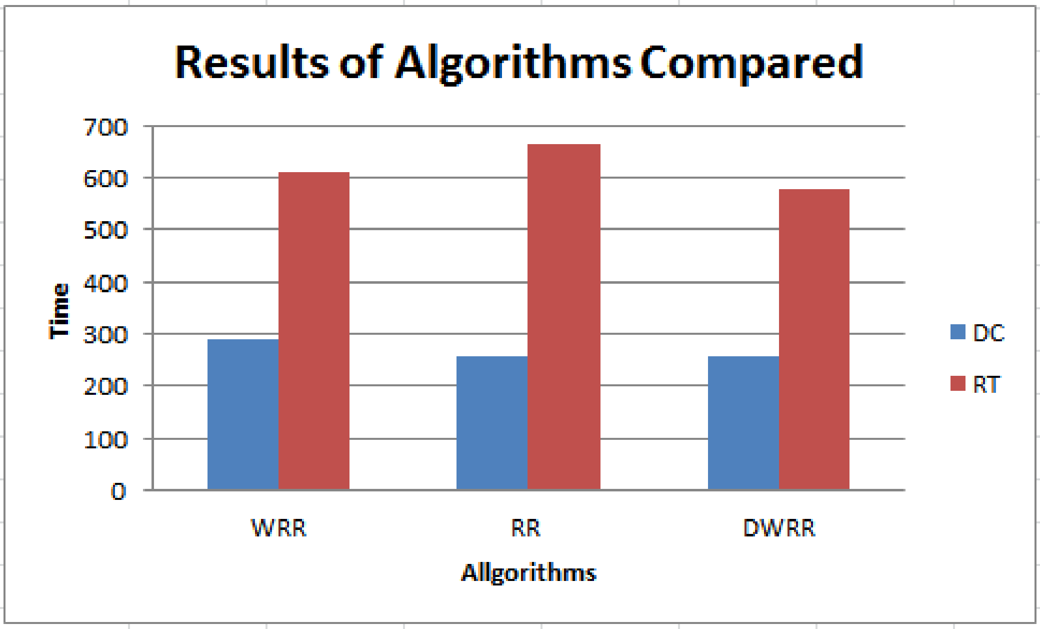

Fig 1-6: Results of Algorithms Compared

5. CONCLUSION.

It is observed that the two algorithms, Weighted Round Robin Algorithm and Round Robin algorithm used to implement the proposed algorithm do not save the state of the previous allocation of a VM to a request from given Userbase which the proposed algorithm does.

Again, these two algorithms do not keep in mind the variations in request size and the effect of number of rounds of the algorithm implementation so as to achieve fairness which also the proposed algorithm does.

Finally it also noted that the results showed by DWRR outperform the various types of Round Robin (Weighted Round Robin and Round Robin with server affinity) algorithms in terms of execution time (makespan) with the least complexity.

The major contribution of the proposed work in the field of cloud computing is that the researcher is able to introduce a new algorithm which is an improved algorithm of Weighted Round Robin and Round Robin with server affinity in cloud computing load balancing environment that reduces the response time and processing time.

6. REFERENCES.

1. Abbass H.A., “A Single Queen Single Worker Honey-Bees Approach to3-SAT,” The Genetic and Evolutionary Computation ConferenceGECCO2001, San Francisco, USA, 2001.
2. Bitam S., BatoucheM., TalbiE-G., “A survey on bee colony algorithms,” 24th IEEE International Parallel and Distributed Processing Symposium, NIDISC Workshop, Atlanta, Georgia, USA, pp. 1-8, 2010.
3. Kun L, Gaochao X, Guangyu Z, Yushuang D, “Cloud Task scheduling based on Load Balancing Ant Colony Optimization” 2011 Sixth Annual China Grid Conference.
4. Komal Mahajan, Ansuyia Makroo, Deepak Dahiya, “Round Robin with Server Affinity: A VM Load Balancing Algorithm for Cloud Based Infrastructure”, http://dx.doi.org/10.3745/JIPS.2013.9.3.379, J Inf Process Syst, Vol.9, No.3, September 2013
5. Martin R., David L, Taleb-BendiabA. “A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing” 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops.
6. Meng X, Lizhen C, Haiyang W, Yanbing B, “A Multiple QoS Constrained Scheduling Strategy of Multiple Workflows for Cloud Computing” 2009 IEEE International Symposium on Parallel and Distributed Processing with Applications.
7. Mequanint M, Thomas G.R, ”Wireless Sensor Networks: Scheduling for Measurement and Data Reporting”, August 31, 2005
8. Rodrigo N. C, Rajiv R, Anton B, César A. F, and Rajkumar B “CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms”
9. Subasish M Subhadarshini M, Smruti R, “Analysis of Different Variants in Round Robin Algorithms for Load Balancing in Cloud Computing”, International Journal of Computer Applications (0975 – 8887),Volume 69– No.22, May 2013.
10. Singh etal, “An Optimized Round Robin Scheduling Algorithm for CPU Schedu Ling”, International Journal of Computer and Electrical Engineering, Vol. 2, No. 7, 2010, pp. 2383-2385
11. Zehua Z. and Xuejie Z. “A Load Balancing Mechanism Based on Ant Colony and Complex Network Theory in Open Cloud Computing Federation”, International Conference on Industrial Mechatronics and Automation, pp-240-243,2010.
12. Zhenzhong Z, Haiyan W , Limin X,Li R, “A Statistical based Resource Allocation Scheme in Cloud” 2011 International Conference on Cloud and Service Computing.