International Journal of Grid Distribution Computing
Vol.7, No.5 (2014), pp.97-110
http://dx.doi.org/10.14257/ijgdc.2014.7.5.09

A New Fuzzy Logic and GSO based Load balancing Mechanism for Public Cloud

Uma Singhal[1] and Sanjeev Jain[2]

Madhav Institute of Technology and Science
Gwalior, India
[1]umasinghal5@gmail.com, [2]dr_sanjeevjain@yahoo.com

Abstract

The recent advances in cloud computing set up well established research in Distributed computing, virtualization, web services, utility computing, have offered many benefits in scalability, cost and efficiency for cloud service users. These advantages are further expected to fulfill the demands of cloud users for cloud services efficiently. This brings the problem of fault tolerance, scalability, efficiency, high availability. Central to these issues require the dynamic and efficient load balancing techniques. In this paper, an effective load balance model is presented for public cloud in which cloud is partitioned with a switch mechanism to choose different load balancing strategy for different load. Fuzzy Logic have been extensively used in various applications such as image processing, data mining, networking, etc. due to its efficient internal architecture and its compatibility to solve various optimization problems. Fuzzy base networks have been observed to produce optimal results in various combinatorial optimization problems. Another important area which provides significant results in solving optimization is the swarm intelligence approach. GSO is observed to have provided significant optimal solution in lesser iterations. In this paper The Fuzzy logic and GSO based load balancing algorithm applied to the load balancing strategy to enhance the utilization and efficiency in the public cloud environment.

Keywords: *Cloud partition, load balancing, Fuzzy logic, GSO*

1. Introduction

Cloud computing is a new model of large-scale virtualization, distributed computing, software, network and web services. It has stimulated computing and data resides into large data centers [1] and away from desktop and manageable PCs. It has the ability to connect the power of Internet and wide area network (WAN) to make use of resources that are available remotely, hence providing cost effective solution to most of the real life requirements [2]. It gives the scalable IT resources such as applications and services, coupled with the infrastructure on which they monitor over the Internet, on pay-as-you-go basis to maintain the capacity rapidly and easily. It supports to occupy changes in demand and helps any organization from investing too much capital cost for its hardware and software need [3, 4]. Thereby, cloud computing is a structure for providing a suitable, on-demand network access to a common pool of computing resources. Cloud service model are of three types, as shown in Figure 1.

Cloud Software as a service (SaaS): The competence provided to the consumer is to make use of the provider's applications consecutively running on a cloud communications. The applications are genial to get from various client devices over a thin client interface for example a web browser. The consumer does not require to deal with the basic cloud infrastructure.

ISSN: 2005-4262 IJGDC

Cloud Platform as a Service (Paas): The service provided to the consumer is to setup its own application on the cloud network. Consumer formed or obtained applications created by means of programming languages and tools sustained by the service provider. The customer does not need to supervise or control the basic cloud structure, but has command over the deployed applications and perhaps configuration setting for cloud domain.

Cloud Infrastructure as a Service (Iaas): The facility given to the customer is to provision processing, storage space, networks, and other basic computing resources where the consumer is capable to deploy and run random software, which contains operating systems and placed applications. The consumer does not need to supervise the underlying cloud communications but has command over operating systems, placed applications, storage and perhaps limited control of select networking components.

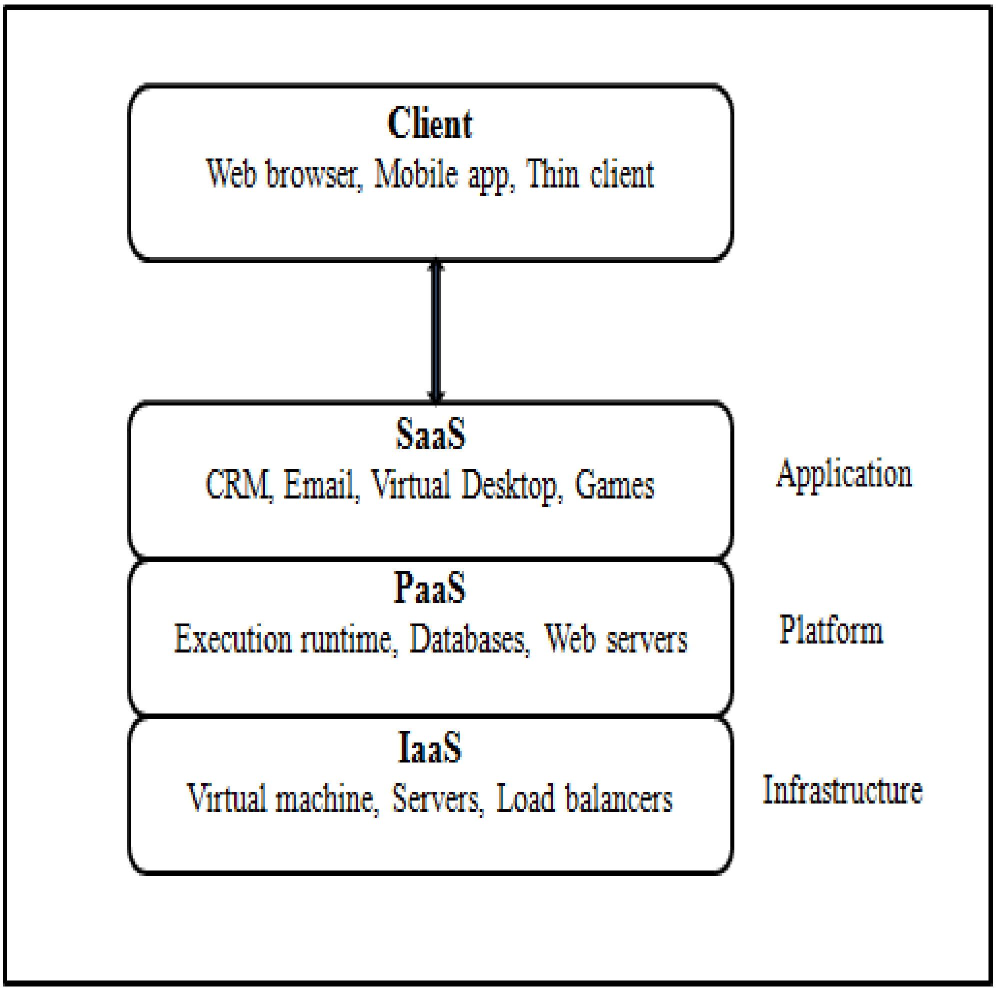
Client
Web browser, Mobile app, Thin client
SaaS
CRM, Email, Virtual Desktop, Games
Application
PaaS
Execution runtime, Databases, Web servers
Platform
IaaS
Virtual machine, Servers, Load balancers
Infrastructure

Figure 1. Cloud Service Model

2. Related Work

Y. Zhao et al. [5] explained the problem that happens in intra-cloud load balancing with substantial hosts by adaptive survive migration of virtual machines. A load balancing model is suggested by the author to decrease the time for virtual machines migration by common storage, to maintain load with servers based on their processor or IO usage, etc. and to preserve virtual machines' zero-downtime relocation in the process. This algorithm promises that the migration of VMs is at all times from high cost physical hosts to low cost host but supposing that each physical host has sufficient memory which is a weak assumption.

A. Bhadani et al. [6] presented a Central Load Balancing Policy for Virtual Machines (CLBVM) that balances the load evenly in a distributed cloud computing environment. This algorithm enhances the performance of the system but not considering fault-tolerant of the system.

Z. Zhang et al. [7] discussed about a load balancing system based on ant colony and complex network theory (ACCLB) in an open cloud computing group. It uses small-world and scale-free features of a composite network to attain improved load balancing.

X. Liu et al. [8] introduced a lock-free multiprocessing load balancing solution that avoids the use of shared memory in difference to further multiprocessing. Load balancing solutions which is required to share memory and lock to manage a user session. It is accomplished by altering Linux kernel.

J. Hu et al. [9] discussed about a scheduling approach on load balancing of VM resources that make use of historical data and present state of the system. This approach achieves the better load balancing and condensed dynamic migration by means of a genetic algorithm. It supports in regulating load-imbalance problem and high migration cost therefore achieves better resource utilization.

3. Proposed Cloud Partitioning for the Public Cloud

There are numerous cloud computing categories. This work mainly focuses on a public cloud. A public cloud is established on the typical cloud computing model, and its services provided by service provider [11]. A public cloud will comprises of several nodes and the nodes are in different physical locations. Cloud is partitioned to manage this large cloud. A cloud consist of several cloud partition with each partition having its own load balancer and there is a main controller which manage all these partition.

3.1 Job Assignment Strategy

Algorithm for allocating the jobs to cloud partition as shown in Figure 2

```
Step 1: jobs appear at the main controller.
Step 2: selecting the cloud partition.
Step 3: if cloud partition is idle or normal state then
Step 4: jobs reach at the cloud partition balancer.
Step 5: allocating the jobs to particular nodes based on the strategy.
Step 6: process finishes.
```

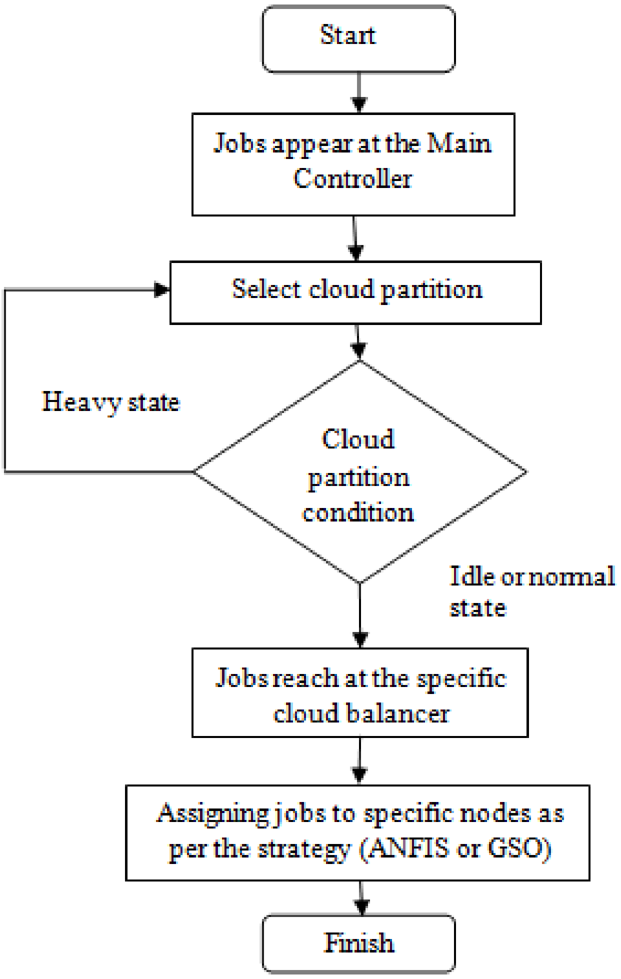
Start
Jobs appear at the Main Controller
Select cloud partition
Heavy state
Cloud partition condition
Idle or normal state
Jobs reach at the specific cloud balancer
Assigning jobs to specific nodes as per the strategy (ANFIS or GSO)
Finish

Figure 2. Proposed Job Assignment Strategy

4. Proposed Load Balancing Strategy based on Different States

In cloud, Load Balancing is a mechanism to assign workload over one or more servers, hard drives, network boundary or other computing resources. Representative datacenter implementations relies on massive, significant computing hardware and network communications, which are subject to the common risks associated with any physical device, comprising hardware failure, power interruptions and resource limits in case of high demand.

High-quality of load balance will increase the performance of the entire cloud. Though, there is no general procedure that can work in all possible different conditions. There are several method have been employed to solve existing problem.

Each specific method has its merit in a specific area but not in all circumstances. Hence, proposed model combines various methods and interchanges between appropriate load balance method as per system status. Here, the idle status uses an fuzzy logic while the normal status uses a global swarm optimization based load balancing strategy.

4.1 Load Balancing using Fuzzy Logic

When the status of cloud partition is idle, several computing resources are free and comparatively few jobs are arriving. In these conditions, this cloud partition is capable to process jobs as fast as possible so an effortless load balancing method can be used.

Zadeh [12] proposed a fuzzy set theory in which the set boundaries were not exactly defined, but in actual boundaries were gradational. Such a set is indicated by continuum of grades of membership function which allocates to each object a membership grade ranging from zero to one [12]. A new load balancing algorithm based on Fuzzy Logic in Virtualized environment of cloud computing is implemented to achieve better processing and response time. The load balancing algorithm is implemented before it outstretch the processing servers the job is programmed based on various input parameters like assigned load of Virtual Machine (VM) and processor speed. It contains the information in each Virtual machine (VM) and numbers of request currently assigned to VM of the system. Therefore, It recognize the least loaded machine, when a user request come to process its job then it identified the first least loaded machine and process user request but in case of more than one least loaded machine available, In that case, we tried to implement the new Fuzzy logic based load balancing technique, where the fuzzy logic is very natural like human language by which we can formulate the load balancing problem.

The fuzzification process is carried out by fuzzifier that transforms two types of input data like assigned load and processor speed of Virtual Machine (VM) and one output as balanced load which are required in the inference system. By evaluating the load and processor speed in virtual machine in our proposed work like two input parameters to produce the better value to equalize the load in cloud environment, fuzzy logic is used. These parameters are taken for inputs to the fuzzifier, which are needed to estimate the balanced load as output.

To affiliate the outputs of the inferential rules [13], low-high inference method is employed. A number of IF-THEN rules are determined by making use of the rule-based fuzzy logic to get the output response with given input conditions, here the rule is comprised from a set of semantic control rules and the supporting control objectives in the system.

1. If (processor_speed is low) and (assigned_load is least) then (balanced_load is medium)
2. If (processor_speed is low) and (assigned_load is medium) then (balanced_load is low)
3. If (processor_speed is low) and (assigned_load is high) then (balanced_load is low)
4. If (processor_speed is Medium) and (assigned_load is least) then (balanced_load is high)
5. If (processor_speed is Medium) and (assigned_load is medium) then (balanced_load is medium)
6. If (processor_speed is Medium) and (assigned_load is high) then (balanced_load is low)
7. If (processor_speed is high) and (assigned_load is least) then (balanced_load is high)
8. If (processor_speed is high) and (assigned_load is medium) then (balanced_load is medium)
9. If (processor_speed is high) and (assigned_load is high) then (balanced_load is medium)

10. If (processor_speed is very_high) and (assigned_load is least) then (balanced_load is high)
11. If (processor_speed is very_high) and (assigned_load is medium) then (balanced_load is high)
12. If (processor_speed is very_high) and (assigned_load is high) then (balanced_load is medium)

As shown above, there are 12 potential logical output response conclusions in our proposed work. The Defuzzification is the method of changing fuzzy output set into a single value and the smallest of minimum (SOM) procedure is employed for the defuzzification.

The total sum of a fuzzy set comprises a range of output values that are defuzzified in order to decode a single output value. Defuzzifier embraces the accumulated semantic values from the latent fuzzy control action and produces a non-fuzzy control output, which enacts the balanced load associated to load conditions.

The defuzzification process is used to evaluate the membership function for the accumulated output. The algorithm-1 is defined to manage the load in Virtual machine of cloud computing as follows:

```
Begin
            request_to_resource()
      P1
      If (resource
     free) Begin
          Estimate
        connection_strength()
        Select fuzzy_rulebase()
        Return
     resource End
   Else
  Begin
       If (Anymore resource found)
             Select_next_resou
        rce() Go to P1
       Else
         Exit
  End
End
```

The proposed Algorithm-1 begins when a user makes a request for connecting to resource. It checks for availability of resource. If the resource is found then it calculates the connection strength. Then this connection is chooses by which resource is accessed as per load and processor speed of virtual machine using fuzzy logic.

4.2 Load Balancing using GSO (Glowworm Swarm Optimization)

When the status of cloud partition is normal, tasks arrives with faster rate compare to idle state and the condition becomes more complex, thus a novel strategy is deployed for load balancing. Each user desired his job in the shortest time; as a result the public cloud requires a strategy that can finish the job of all user with adequate response.

In this optimization algorithm, each glowworm is distributed in the objective function definition space [14]. Also each glowworm transfer its own luciferin value and have the respective scope called local-decision range. As the glow searches in the local-decision range for the neighbor set, in the neighbor set, glow attracted to the neighbor with brightest glow. That is glow selects neighbor whose luciferin value greater than its own, and the flight direction will change each time different will change with change in selected neighbor.

Each glowworm i encodes the object function value $\mathrm{J}(\mathrm{x_i}(\mathrm{t}))$ at its current location $\mathrm{x_i}(\mathrm{t})$ into a luciferin value $\mathrm{l_i}$ and advertises the same within its neighborhood. The neighbor's set ($\mathrm{N_i}(\mathrm{t})$) of glowworm i comprises of those glowworms that have comparatively a higher luciferin value and that are situated within a dynamic decision range and their movements are updated by equation (8) at each iteration.

Local-decision range update:

$$\mathrm{r_d^i}(\mathrm{t}+1) = \min\left\{\mathrm{r_s}, \max\left\{\begin{matrix} 0, \mathrm{r_d^i}(\mathrm{t}) \\ +\beta(\mathrm{n_t} - |\mathrm{N_i}(\mathrm{t})|\end{matrix}\right\}\right\}; \qquad (8)$$

and $\mathrm{r_d^i}(\mathrm{t}+1)$ is the glowworm i's local-decision range at the $\mathrm{t}+1$ iteration, $\mathrm{r_s}$ is the sensor range, $\mathrm{n_t}$ is the neighbourhood threshold, the parameter β generates the rate of change of the neighborhood range. Local-decision range consist of the following number of glow:

$$\mathrm{N_i}(\mathrm{t}) = \{\mathrm{j}: \|\mathrm{x_j}(\mathrm{t}) - \mathrm{x_i}(\mathrm{t})\| < \mathrm{r_d^i}; \mathrm{l_i}(\mathrm{t}) < \mathrm{l_j}(\mathrm{t})\}; \qquad (9)$$

and, $\mathrm{x_j}(\mathrm{t})$ is the glowworm i's position at the t iteration, $\mathrm{l_j}(\mathrm{t})$ is the glowworm i's luciferin at the t iteration.; the set of neighbours of glowworm i comprises of those glowworms that have a comparatively higher luciferin value and that are situated within a dynamic decision range whose range $\mathrm{r_d^i}$ is defined above by a circular sensor range $\mathrm{r_s}(0 < \mathrm{r_d^i} < \mathrm{r_s})$.Each glowworm as given in equation (10), i elects a neighbor j with a probability $\mathrm{p_{ij}}(\mathrm{t})$ and process toward it as:

Probability distribution used to select a neighbor:

$$\mathrm{p_{ij}}(\mathrm{t}) = \frac{\mathrm{l_j}(\mathrm{t}) - \mathrm{l_i}(\mathrm{t})}{\sum_{\mathrm{k} \in \mathrm{N_i}(\mathrm{t})} \mathrm{l_k}(\mathrm{t}) - \mathrm{l_i}(\mathrm{t})} \qquad (10)$$

Movement update:

$$\mathrm{x_i}(\mathrm{t}+1) = \mathrm{x_i}(\mathrm{t}) + \mathrm{s}\left(\frac{\mathrm{x_j}(\mathrm{t}) - \mathrm{x_i}(\mathrm{t})}{\|\mathrm{x_j}(\mathrm{t}) - \mathrm{x_i}(\mathrm{t})\|}\right); \qquad (11)$$

Luciferin-update:

$$l_i(t) = (1 - \rho)l_i(t - 1) + \gamma J\big(x_i(t)\big); \qquad (12)$$

and $l_i(t)$ is a luciferin value of glowworm i at each t iteration, $\rho \in (0,1)$ leads to the reflection of the accumulative goodness of the path . This path is followed by the glowworms in their ongoing luciferin values, the parameter γ only ascends the function fitness values, $J(x_i(t))$ is the value of test function.

In this optimization algorithm, each glowworm is distributed in the objective function definition space [44]. These glowworms transfer own luciferin values and have the respective scope called local-decision range. As the glow searches in the local-decision range for the neighbor set, in the neighbor set, glow attracted to the neighbor with brightest glow. That is glow selects neighbor whose luciferin value greater than its own, and the flight direction will change each time with change in selected neighbor.

The below algorithm presents the core logic of GSO based cloud scheduler. In our proposed algorithm, all hosts associated to a Cloud are referred a swarm and each host in the Cloud is a glowworm in this swarm. Following the analogy from the classical GSO, the position of each host in the swarm can be formed by its load. This definition helps to explore in the local search space and try to reduce the load. Every time a user demands a VM, it is initialized in a random host (getInitialHost ()) and each host in the search space gets a position as per its load through the calculateTotalLoad(hostId) method. Load refers to the total CPU utilization within a host. The neighborhood of that glowworm is also acquired through getNeighbors (hostId, neighborSize). Each one of the neighbors in the neighborhood is selected randomly as this delivers the best results. The size of the particle neighborhood is a parameter assigned by the user.

In each iteration of the algorithm, a glowworm moves through its neighbors searching for less loaded hosts. The velocity or brightness of each glowworm is defined as the load difference between original host and its other neighbor's hosts. If any of the neighboring hosts is least loaded than the original host, then the VM is migrated to the neighbor host with a greater velocity. Taking into account that the glowworms move through hosts of their neighborhood in search of a host with the lower load, the algorithm reaches a local optimum quickly. Thus, each glowworm makes a move to one of its neighbors, which has the least load among all. If all neighbors of original host are busier than it, the VM is not migrated from the current host. Finally, the glowworm provides its associated VM to the host with the minimum load among their neighbors and finishes its task.

In the context of load balancing for cloud computing GSO algorithm check the status of the server simultaneously if it is free. For example a user wants to download a file size of 50 MB. It checks by iteration if user gets entered in server, it gets the message as achieve target.

```
Procedure:  GSOallocationPolicy (vm, hostList )

Begin

    i =0 , networkMessages =0 , velocity=
    -1 glowworm = newGlowworm (vm,
    hostList ) initialHostId = particle.
    getInitialHost( )

    currentPostionLoad=   glowworm.calculateToataLoad( initialHostId)
    neighbours =  glowworm. getNeighbours ( initialHostId ,
    neighbourSize )

    While ( i < neighbours.size( ) ) do

        neighbourId = neighbours.get ( i )

        destPositionLoad = glowworm.calculateTotalLoad (
        neighbourId )  networkMessages++

        if ( destPositionLoad = = 0)
            currentPositionLoad =
            destPositionload destHostId =
            neighbours.get ( i )

            i= neighbours.size( )

         end if

        if ( currentPositionLoad – destpositionLoad  >
            velocity ) velocity = currentPositionLoad -
            destPositionLoad  currentPositionLoad  =
            destPositionLoad

            destHostId = neighbours.get ( i )

        end if

        i++

    end  while

    allocatedHost = hostList . get( destHostId )

    if ( !allocatedhost . allocateVM
        (vm) ) GSOallocationPolicy
        (vm, hostList )

End
```

5. Experimental Setup and Parameter

The experiment is implemented with Datacenter, virtual machine, cloudlets (tasks) and host under the CloudSim simulation platform [15]. The resource situation is shown in table 1. The workload of task is from 10000 MI (Million Instruction) to 90000 MI is to be computed and type of manager for both datacenter and virtual machine is Time_shared.

In cloud simulator, the Cloudsim3.0, that facilitates modeling and simulation of Cloud computing systems and application surroundings. It provides both behavior and system modeling of Cloud system components such as host, data centers, clients, virtual machines (VMs) and resource provisioning policies. By adopting the load balancing parameters of virtual machine and data center, we have implemented on cloudsim-3.0, a cloud computing simulator and estimate the performance of proposed work using job assigning to the cloud partition. Figure 4, 5, and 6 represents the graphical representation of the load balancing degree, time and cost of proposed load balancing model in each cloud partition respectively and the proposed approaches such as Fuzzy logic and GSO based load balancing algorithm have shown the better results than the existing technique Round Robin.

Table 1. Simulation Parameters

Type	Parameters	Value
Datacenter	Number of Datacenter	10
	Number of Host	2-6
	Type of Manager	Time_shared
Virtual Machine (VM)	Total number of VMs	50
	MIPS of PE (processing element)	250-600 (MIPS)
	Number of PE per VM	1-3
	RAM(VM memory)	256-3056 (MB)
	Bandwidth	700-1500 bit
	Type of Manager	Time_shared
Task	Total number of Task	100-500
	Length of Task	10000-90000 MI
	Number of PEs requirement	1-2

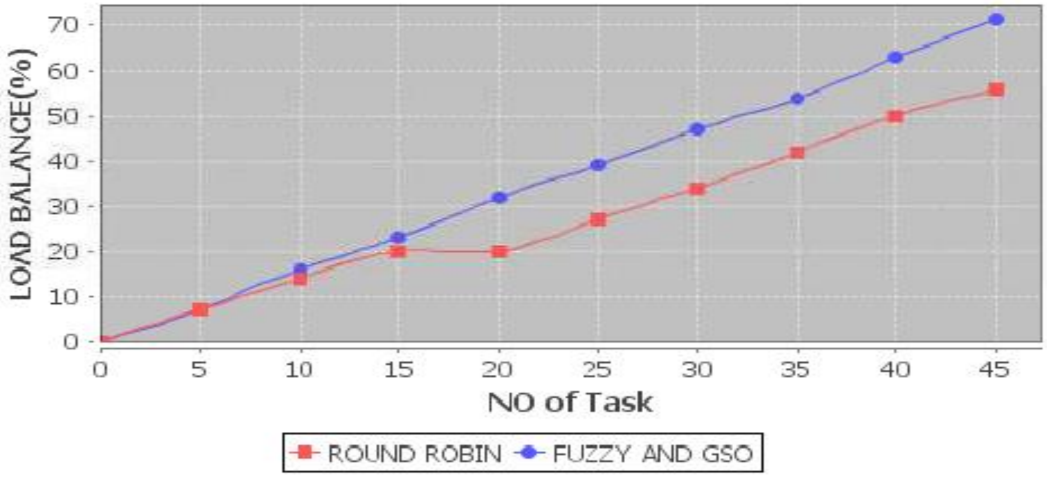

Figure 4. Load Balancing in Cloud Partition

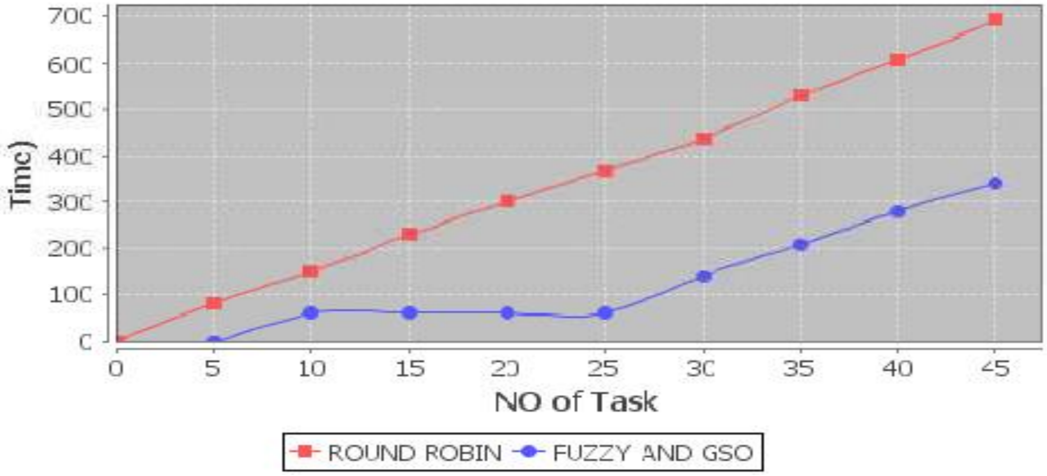

Figure 5. Time of Execution

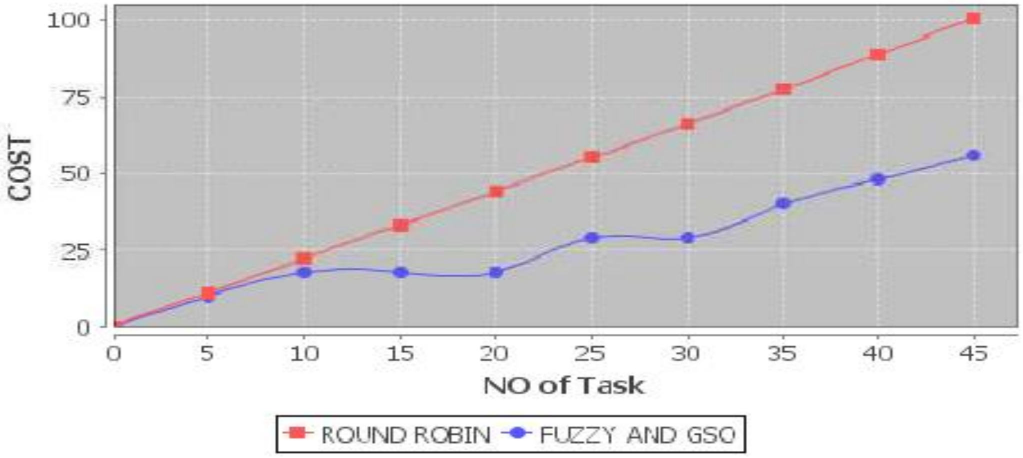

Figure 6. Cost of Execution

6. Conclusion

Load balancing in cloud computing is one of the major issue. It is used to manage the dynamic workload equally over all the nodes so that high resource utilization and availability of resources can be achieved. Existing techniques have many drawbacks, therefore an adequate load balancing technique is required that can enhance the cloud computing performance through distributing all the workload over the nodes equally with higher resource utilization. In this paper, proposed the new time efficient and cost effective load balancing algorithm using Fuzzy logic and GSO based load balancing algorithm are used to achieve significant enhancements in resource utilization and availability in cloud-computing environment.

References

[1] Nidhi Jain Kansal, Inderveer Chana, published ," Cloud Load balancing techniques: A step towards green computing", IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 1, January 2012.

[2] B. P. Rimal, E. Choi, and I. Lumb, "A Taxonomy, Survey, and Issues of Cloud Computing Ecosystems, Cloud Computing: Principles, Systems and Applications", Computer Communications and Networks, Chapter 2 , pages 21-46, DOI 10.1007/978- 1-84996-241-42, Springer – V erlagLondonLimited, 2010.

[3] R. W. Lucky, "Cloud computing", IEEE Journal of Spectrum, Vol. 46, No. 5, May 2009, pages 27- 45.

[4] M. D. Dikaiakos, G. Pallis, D. Katsa, P. Mehra, and A. Vakali, "Cloud Computing: Distributed Internet Computing for IT and Scienti_c Research", IEEE Journal of Internet Computing, Vol. 13, No. 5, September/October 2009, pages 10-13.

[5] Y. Zhao, and W. Huang, "Adaptive Distributed Load Balancing Algorithm based on Live Migration of Virtual Machines in Cloud", Proceedings of 5th IEEE International Joint Conference on INC, IMS and IDC, Seoul, Republic of Korea, August 2009, pages 170-175.

[6] A. Bhadani, and S. Chaudhary, "Performance evaluation of web servers using central load balancing policy over virtual machines on cloud", Proceedings of the Third Annual ACM Bangalore Conference (COMPUTE), January 2010.

[7] H. Mehta, P. Kanungo, and M. Chandwani, "Decentralized content aware load balancing algorithm for distributed computing environments", Proceedings of the International Conference Workshop on Emerging Trends in Technology (ICWET), February 2011, pages 370-375.

[8] Xi. Liu, Lei. Pan, Chong-Jun. Wang, and Jun-Yuan. Xie, "A Lock-Free Solution for Load Balancing in Multi-Core Environment", 3rd IEEE International Workshop on Intelligent Systems and Applications (ISA), 2011, pages 1-4.

[9] J. Hu, J. Gu, G. Sun, and T. Zhao, "A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing Environment", Third International Symposium on Parallel Architectures, Algorithms and Programming (PAAP), 2010, pages 89-96.

[10] V. Nae, R. Prodan, and T. Fahringer, "Cost-Ef_cient Hosting and Load Balancing of Massively Multiplayer Online Games", Proceedings of the 11th IEEE/ACM International Conference on Grid Computing (Grid), IEEE Computer Society, October 2010, pages 9-17.

[11] A.Rouse, Public cloud, http://searchcloudcomputing.techtarget.com/definition/public-cloud, 2012.

[12] L.A. Zadeh, Fuzzy sets, Inf. Control 8 (1965), 338–353.

[13] Pedrycz W and Gomide F (2011), "An introduction to fuzzy sets: analysis and design", complex adaptive systems. MIT Press, 1998.

[14] Jiakun LIU, Yongquan ZHOU†, Kai HUANG, Zhe OUYANG, Yingjiu WANG "A Glowworm Swarm Optimization Algorithm Based on Definite Updating Search Domains" Journal of Computational Information Systems 7: 10 (2011) 3698-3705.

[15] Buyya, R., Ranjan, R., Calheiros, R.N., "Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities" in Proceedings of the 7th High Performance Computing and Simulation (HPCS 2009) Conference, Leipzig, Germany, DOI: 10.1109/HPCSIM.2009.5192685, 2009.

Authors

Uma Singhal, has received the B.E. degree in Computer Science and Engineering, from MPCT, Gwalior, India in 2011. Currently she is pursuing M.Tech in Computer Science and Engineering from MITS, Gwalior and it will be completed in 2014. Her research include Cloud Computing and Adhoc Network.

Dr. Sanjeev Jain, is Director, MITS Gwalior, India. He has 27 years of teaching experience, including 7 year administrative experience as Director of MITS a Grant in aid Autonomous Engineering College of Madhya Pradesh. His teaching and research include Image Processing and Data Mining, Computer Network.