An Efficient Adaptive Load Balancing Algorithm for Cloud Computing Under Bursty Workloads

Sally F. Issawi
Faculty of Information Technology
Islamic University of Gaza (IUG)
Gaza, Palestine
sally.issawi@gmail.com

Alaa Al Halees
Faculty of Information Technology
Islamic University of Gaza (IUG)
Gaza, Palestine
Alhalees@iugaza.edu.ps

Mohammed Radi
Faculty of Applied Science
Al-Aqsa University
Gaza, Palestine
moh_radi@alaqsau.edu.ps

***Abstract*—Cloud computing is a recent, emerging technology in the IT industry. It is an evolution of previous models such as grid computing. It enables a wide range of users to access a large sharing pool of resources over the internet. In such complex system, there is a tremendous need for an efficient load balancing scheme in order to satisfy peak user demands and provide high quality of services. One of the challenging problems that degrade the performance of a load balancing process is bursty workloads. Although there are a lot of researches proposing different load balancing algorithms, most of them neglect the problem of bursty workloads. Motivated by this problem, this paper proposes a new burstness-aware load balancing algorithm which can adapt to the variation in the request rate by adopting two load balancing algorithms: RR in burst and Random in non-burst state. Fuzzy logic is used in order to assign the received request to a balanced VM. The algorithm has been evaluated and compared with other algorithms using Cloud Analyst simulator. Results show that the proposed algorithm improves the average response time and average processing time in comparison with other algorithms.**

Keywords-cloud computing; cloud analyst; burstness; fuzzifier; load balancing algorithm

I. INTRODUCTION

Nowadays most developments in the IT industry come to meet the demands for utilizing more resources in lower costs. This technological trend has enabled the evolution of a new computing model called cloud computing, in which resources and services are available on the Internet and can be leased and released on demand [1]. Cloud computing can improve business performance by minimizing the overhead of buying, managing and controlling IT resources. The financial model applied in cloud computing is "Pay-per-Use" so the consumer only pay for his needs. The scale up in demands make load balancing a major concern in cloud computing. This is defined as a method to distribute the workload across one or more servers, network interfaces, hard drives, or other computing resources. Load balancing is used to make sure that none of the existing resources are idle while others are being utilized [2]. One of the most challenging problems that dramatically degrade the performance of a load balancing process is the burstiness in workloads. Bursty traffic refers to an uneven pattern of data transmission: sometimes very high data transmission rate while other times low [3].

Several load balancing algorithms had been proposed which focus on key elements such as processing time, response time and processing costs. However these algorithms neglect the case of bursty workloads. According to that, in this research we proposed an efficient adaptive load balancing algorithm for cloud computing under bursty workloads. The proposed algorithm adapt to the variation in the received request rate by adopting two different load balancing algorithms according to the workload state. It detects the start of the burst by calculating the average received requests and use Round Robin (RR) in the burst case, otherwise the Random algorithm is used. Those two load balancing algorithm select a suitable VM based on the knowledge provided by a fuzzifier.

The algorithm has been evaluated and compared with other algorithms using the Cloud Analyst simulator. Results show that the proposed algorithm improves the average response time and average processing time compared to other algorithms.

II. CLOUD COMPUTING

The core idea behind cloud computing is not a new one. It was actually pronounced way back in 1960 [4]. Cloud computing is a type of parallel and distributed system. It consists of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources. The services delivered to the consumer are based on service-level agreements (SLA), established through negotiation between the service provider and consumers [5]. The objective of cloud computing is to provide secure, qualitative, scalable, quick, more responsive, on demand, cost-efficient and automatically provisioned services such as computation services, storage services, networking etc. Although those services are geographically distributed all over the world, they are provided in a location independent way [4]. Cloud computing can help improve business performance while making a contribution to control the cost of delivering IT resources to any organization. It minimizes the overhead of buying, managing and controlling IT resources. The financial model applied in cloud computing is "Pay-per-Use" so the consumer only pay for his needs.

Research on cloud computing is still at an early stage. Several new challenges keep emerging from industry

applications, and many issues need to be properly addressed. Some of the challenging research issues in cloud computing are: security and privacy, performance, resource management and scheduling. Resource management is a very important issue in cloud computing, as large numbers of users share the same resources. So in order to meet Quality of Service (QoS) standards and insure best resource utilization, proper resource management mechanisms should be used in deferent levels such as management of memory, disk space, CPU's, cores, threads, VM images, I/O devices etc. Resource provisioning can be defined as allocation and management of resources to provide desired level of services. Job scheduling is an essential process in resource provisioning where the order of the job execution is established in order to optimize performance parameters such as response time, processing time, waiting time etc. [4]. One of the most important issues in job scheduling is load balancing which is the interest of this work.

III. LOAD BALANCING

The scale up in demands make load balancing a major concern in cloud computing. It is defined as a method to distribute workload across one or more servers, network interfaces, hard drives, or other computing resources. Load balancing is used to make sure that none of the existing resources are idle while others are being utilized [2].

Basically there are 2 types of load balancing algorithm depending on their implementation method:

1) Static Algorithms

In this type, the load is divided equivalently between nodes. This algorithm depend on prior knowledge of the system, it does not consider the current state of the node and will degrade the performance of the system. This type of algorithms is referred to as round robin algorithms [2, 6].

2) Dynamic Algorithms

Dynamic algorithms make decisions based on current state of the system. No prior knowledge is needed [2]. So workloads can be distributed efficiently over nodes. Dynamic load balancing can be done in two ways: [7]

- Distributed dynamic load balancing:

In the distributed one, all nodes in the system execute the dynamic load balancing algorithm and the task of load balancing is shared among them. Its advantage is that if one or more nodes in the system fail, the system performance will be affected to some extent, but it will not cause the total load balancing process to halt.

- Non-distributed dynamic load balancing:

In the non-distributed one, the load balancing algorithm is executed by a single node of the system and the task of load balancing is dependent only on that node. A failure in this one node will cause the total load balancing process to halt.

Static (round robin) algorithms are based on a simple rule in dividing the loads among nodes but this leads to more loads conceived on servers and thus imbalanced traffic discovered as a result. However; dynamic algorithm predicated on a query that can be made frequently on servers, but sometimes prevailed traffic will prevent these queries to be answered, and correspondingly more added overhead can be distinguished on the network [6].

IV. BURSTY WORKLOAD

One of the most challenging problems that dramatically degrade the performance of load balancing process is burstiness in workloads. Bursty traffic refers to an uneven pattern of data transmission: sometime very high and other times very low [3]. Burstiness occurs in workloads in which bursts of requests aggregate together during short periods of time and create periods of peak system utilization. Figure 1 shows three different levels: strong, week, and no burstiness.

This problem is often observed in large systems including web based applications [8], grid services [9], multitier architectures [10], and large storage systems [11]. It can dramatically degrade system performance, make the system unavailable and lead to a total failure. Burstiness considered as one of the most complex problem nowadays in cloud computing, as the number of users that uses cloud services increases day by day, so load balancer must consider the performance of each instance under both bursty and non-bursty workloads for efficient resource utilization.

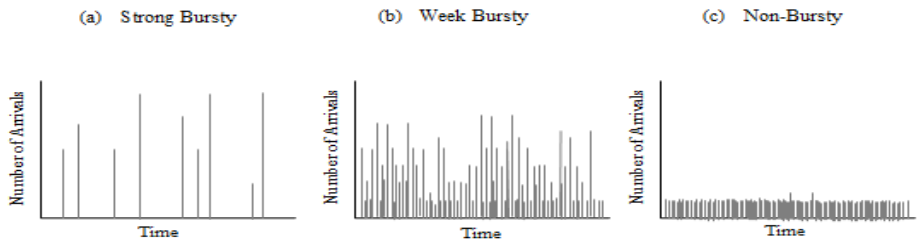

Fig. 1. Different levels of burstness

V. RELATED WORKS

Several approaches had been proposed to handle the load balancing issues in cloud computing systems. All these works aimed to improve the process of distributing the workload among cloud nodes and try to achieve optimal resource utilization, minimum data processing time, minimum average response time, and overload avoidance. However most of these

approaches neglected the effect of burstiness on the load balancing process. Sethi, et al in [12] designed a new load balancing technique using fuzzy logic based on a Round Robin (RR) algorithm to obtain measurable improvements in resource utilization and availability of cloud-computing environment. The proposed technique uses a fuzzifier to perform the fuzzification process that converts two types of inputs which are the processor speed and the assigned load of the Virtual Machine (VM), and one output which is the balanced load to create an inference system. The Fuzzy based Round Robin (FRR) load balancer compared to the conventional Round Robin (RR) load balancer minimizes the data center processing time and overall response time. The problem with Round Robin algorithms in general however, is that they are not able to handle bursty workloads. Even with the proposed enhancement on RR by using fuzzy logic, burstiness is not considered.

In [13], a fuzzy logic load balance algorithm focused on a public cloud was proposed. The main idea of the algorithms was to partition the Cloud to several cloud partitions with each partition having its own load balancer, and a main controller to manage all these partition. Results showed enhancements in resource utilization and availability in the cloud computing environment. The drawback of this approach is the difficulty of testing the technique in a real environment to make sure that it has achieved good results. In [14], a smart burstiness-aware algorithm (ARA) to balance bursty workloads across all computing sites, and thus to improve overall system performance, was proposed. The presented algorithm predicts the beginning and the end of workload bursts and automatically on-the-fly shift between two schemes: "greedy" (i.e., always select the best site) which has better response time under the case of no burstiness and "random" (i.e., randomly select one) which has better response time under burstiness. Both simulation and real experimental results show that this algorithm improves the performance of the cloud system under both bursty and non-bursty workloads. Although this algorithm gives good results, it does not consider an important factor in load balancing, which is the current utilization of available resources.

In [7], a dynamic load balancing model that considers utilizing resources under burstiness cases was proposed. The suggested architecture consists of four parts: Cloud controller server, Node controller server, Agents, and Virtual machines. All requests first go to the cloud controller server and then they are transferred to the load balancer. Finally a virtual instance is selected by the load balancer based on the information supplied by the monitoring agent about CPU usage, memory and storage space usage. The researchers claimed that this algorithm should ensure the optimum utilization of cloud resources, faster response time, and cut the economic cost for an organization. However they did not do any experiments or evaluations for their work.

In [15], load balancing under bursty environment for Cloud Computing was also investigated. A dynamic load balancing algorithm which maintains the state of all virtual machine (VM) resources was proposed. The algorithm. based on CPU, memory and storage space utilization, selects the less utilized VM resource to handle the request. A monitoring agent was used to continuously monitor CPU usage, memory and storage space usage, and the current and the expected load for each virtual machine. Based on this information a Pheromone (or probability) was assigned for every VM. When a request arrives to the datacenter, the load balancer transfers the request to the VM which has the least Pheromone. Authors mentioned that their algorithm improved the performance but they did not provide any comparisons with other load balancing algorithms and they did not share any experiments results.

In [16], an approach to overcome the un-utilized resource provisioning and the power consumption problems under bursty and fractal behavior workload was proposed. It consists of two phases for resource utilization provisioning, called "predictive and reactive provisioning". Firstly the forecasting module predicts the work load for the next control horizon, and then the controller estimates the number of necessary resources, such as processing cores, for the predictable part of the incoming load. In order to avoid the consequences of forecasting errors, the system allocates extra resources that can be used to serve unpredictable loads. This allocation is made based on the history of the system operation. The proposed approach improves the resource utilization and the power consumption. On the other hand, if some prediction error happens beyond the estimation of the extra resources, it would be subject to delay in getting the resource till the system allocates available resources.

VI. ADAPTIVE LOAD BALANCING ALGORITHM

The request rates received by the datacenter are not constant all the time. Sometimes large number of requests aggregated in a small period of time creating a burst. This affect the performance of the load balancing algorithm as it increase the processing time and the repose time of the datacenter. The performance of several load balancing algorithms differs according to the users' requests rate. For example some algorithms work efficiently under low workload while their performance is degraded under high workload and vice versa. To overcome burst problem and benefit from different load balancing algorithms advantages we propose a new load balancing algorithm called Adaptive algorithm.

Adaptive algorithm is a load balancing algorithm used by the datacenter to distribute the received tasks efficiently over the virtual machine under bursty workload by swapping between two policies depending on the requests rates. It consists of three main components as follows:

1- Burst detector.
2- Load Balancing Algorithms.
3- Fuzzifier.

When the datacenter receives a request, the burst detector determines the workload state (Normal or Burst). Depending on the burst detector decision, the datacenter will select the appropriate load balancing policy for that state. After that, the selected load balancing algorithm will assign the received task to a suitable VM depending on the information supplied by the fuzzifier. When the VM complete its assigned task, it informs

the data center. The main steps of the adaptive algorithm are shown in Figure 2.

A. Burst Detector

The burst detector is responsible for detecting the variation in the workload, and determining whether the state of the workload is burst or not, using a specific threshold. When a request arrives, the burst detector checks the rate of the requests in the last 15 minutes and if it exceeds the threshold it indicates that the status is burst. Depending on experiments, we found that 15 minutes is a suitable time interval. Depending on the detector decision, the datacenter will select the proper load balancing policy.

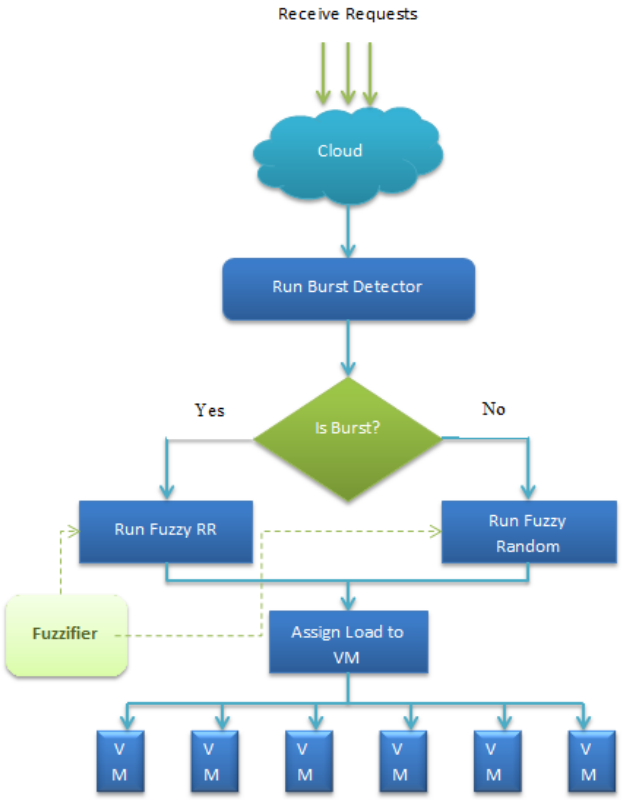

Fig. 2. Adaptive load balancing algorithm flow chart

B. Load Balancing Algorithm

The proposed approach uses two load balancing algorithms, one efficient in normal cases and another efficient in burst cases. According to experiments done on three load balancing algorithms (RR, ESCE and Random), Random policy performs the best in low workload and Round Robin performs best in high workload.

1) Random Policy:

When the burst detector decides that the workload state is normal, the random policy will be applied. The fuzzifier supply the random policy with a candidate list of balanced VMs in the data center, then the policy will select one of these VMs randomly and assign the received task to it.

2) Round Robin Policy:

Round Robin will be used when workload state is burst. The same as random, the fuzzifier will provide a candidate list of the most balanced VMs for Round Robin policy. Then Round Robin will use this list to allocate VMs in a cycle manner.

C. Fuzzifier

The main function of the fuzzifier is to enhance the decision of the load balancing algorithm by providing a list of the most balanced VMs in the data center and deliver it to the load balancer to allocate one VM from this list. The fuzzifier is consisted of a Fuzzy Inference System (FIS) to simulate the way of human decision making by using fuzzy control rules and linguistic parameters. In our wok, the FIS uses two inputs which are processor speed and the load in VM, and balanced load as the output. Twelve IF-THEN rules are employed as shown in Figure 3. For the FIS, an open source Java library called jFuzzyLogic [17] was used. This library offers a fully functional and complete implementation of a fuzzy inference system, providing a programming interface and Eclipse plugin to easily write and test code for fuzzy control applications [18, 19].

VII. EXPERMENTS AND RESULTS

In order to test and evaluate the performance of our new proposed algorithm, the CloudAnalyst simulator was employed. CloudAnalyst is a tool developed at the University of Melbourne. It is a graphical simulation tool based on Cloudsim for modeling and analyzing the behavior of a cloud computing environment, which supports visual modeling and simulation of large-scale applications that are deployed on Cloud Infrastructures [20-22]. Experiments had been done to test the efficiency of the proposed scheme. Two metrics were measured in order to evaluate the performance: Response Time and Processing Time. The results were compared with three of the most popular load balancing algorithms: RR, ESCE, and Random.

A. Configurations

In order to build the simulation environment, two main components had to be configured: User Base (UB) and Data Center (DC). For the three experiments the following configurations was used.

1) User Base

The number of UBs used is 6. All UBs are in the same region with DC to ignore the transmission delay. Number of Requests per user per hour for every UB is 12 and the Data Size per Request is 100 Byte. Table I illustrates the User Bases characteristics.

2) Data Center

One Data Center was used in the experiments. The Data Center had 5 Physical Hosts with different Processor Speeds. Data Center configurations are shown in detail in Table II and Table III. The simulation time was set to one day. Experiments had been done with three different Instruction Lengths (250, 500 and 1000 Bytes).

TABLE I. USER BASE CONFIGRATIONS

Name	Peak Hours Start (GMT)	Peak Hours End (GMT)	Avg. Peak Users	Avg. Off-peak Users
UB1	1	2	100000	10000
UB2	4	5	200000	20000
UB3	6	7	700000	70000
UB4	9	10	400000	40000
UB5	13	14	500000	50000
UB6	20	21	800000	80000

TABLE II. DATA CENTER MAIN CONFIGRATIONS

Data Center	#VMs	Image Size	Memory	BW
DC1	50	10000	1024	1000

TABLE III. DATA CENTER HOSTS CONFIGRATIONS

ID	Memory (Mb)	Storage (Mb)	Available BW	Num. of Processors	Processor Speed	VM Policy
0	204800	100000000	1000000	4	2000	TIME_SHARED
1	204800	100000000	1000000	5	5000	TIME_SHARED
2	204800	100000000	1000000	2	9000	TIME_SHARED
3	204800	100000000	1000000	2	1000	TIME_SHARED
4	204800	100000000	1000000	2	15000	TIME_SHARED

B. Results

The Data Center hourly loading during the simulation time is shown in Figure 3. The experiment results showed that the adaptive algorithm recorded the best response and processing time compared to RR, ESCE, and Random algorithms. As shown in Figures 4-6, when the Instruction Length is 250 Bytes, the adaptive algorithm has better response time than RR (which is better than ESCE and Random) with a difference of 2 ms. This difference is remarkably increased when the instruction size is increased to 500 Bytes and 1000 Bytes (7ms).

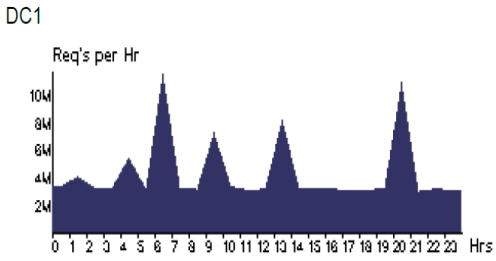

Fig. 3. Data Center Hourly Loading.

The processing time results show a similar trend with the response time results. As presented in Figures 4-6, the adaptive algorithm has the best processing time compared to the others. The improvement in processing time became clearer when the Instruction Length was increased

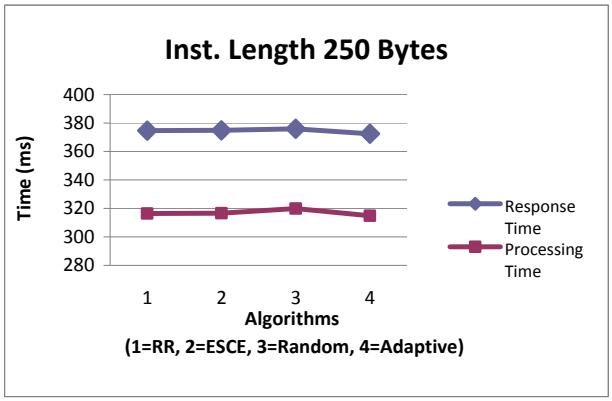

Fig. 4. Experment results when inst. length is 250 Bytess

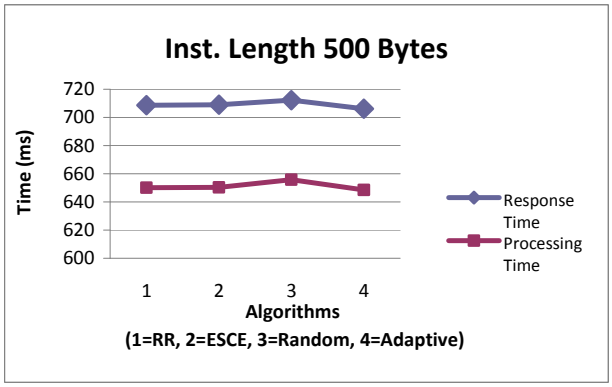

Fig. 5. Experment results when inst. length is 500 Bytes

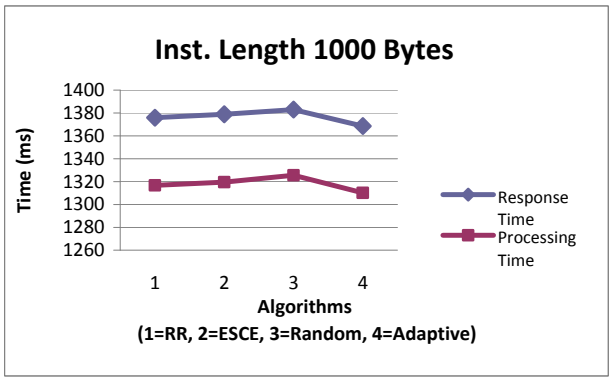

Fig. 6. Experiment results when inst. length is 1000 Bytes

VIII. CONCLUSION

Cloud computing nowadays has become a quite popular model by offering a variety of resources shared over the

Internet. In such a complex system, the need for an efficient load balancing scheme is essential in order to satisfy peak user demands and provide high quality of services. One of the challenging problems that degrade the performance of load balancing process is bursty workloads. In this paper, we proposed a load balancing algorithm called Adaptive Algorithm. The proposed algorithm is based mainly on swapping between two different algorithms (RR and Random) according to the workload status. Selecting VM to handle the received request is based on knowledge about CPU speed and the current load of the VM provided by a fuzzifier. Experiments were conducted using the CloudAnalyst simulator. Results showed that the adaptive algorithm decreased the response and the processing time and thus improved the performance of the cloud system.

REFERENCES

[1] Q. Zhang, L. Cheng, R. Boutaba, "Cloud computing: state-of-the-art and research challenges", Journal of Internet Services and Applications, Vol. 1, No. 1, pp. 7-18, 2010

[2] R. Mishra, A. Jaiswal, "Ant colony Optimization: A Solution of Load Balancing in Cloud," International Journal of Web & Semantic Technology, Vol. 3, No. 2, pp. 33-50, 2012

[3] J. Dong, Network Dictionary, Javvin Technologies Inc, 2007

[4] M. Sajid, Z. Raza, "Cloud Computing: Issues & Challenges," International Conference on Cloud, Big Data and Trust 2013, RGPV, India, November 13-15, 2013

[5] J. Uma, V. Ramasamy, P. Vivekanandan, "Load Balancing Algorithms in Cloud Computing Environment - A Methodical Comparison", International Journal of Engineering Research and Technology, Vol. 3, No. 2, pp. 272-275, 2014

[6] Z. Chaczko, V. Mahadevan, S. Aslanzadeh, C. Mcdermid, "Availability and Load Balancing in Cloud Computing", 2011 International Conference on Computer and Software Modeling, Singapore, September 16, 2011

[7] M. Rutvik,, P. Yask, T. Harshal, "Architecture For Distributing Load Dynamically In Cloud Using Server Performance Analysis Under Bursty Workloads", International Journal of Engineering Research and Technology , Vol. 1, No. 9, pp. 1-4, 2012

[8] M. L. Chin, C. E. Tan, M. I. Bandan, "Efficient DNS based Load Balancing for Bursty Web Application Traffic", Vol. 1, No. 1, pp. 1-5, 2012

[9] H. Li, M. Muskulus, "Analysis and Modeling of Job Arrivals in a Production Grid", ACM SIGMETRICS Performance Evaluation Review, Vol. 34, No. 4, pp. 59-70, 2007

[10] N. Mi, Q. Zhang, A. Riska, E. Smirni, E. Riedel, "Performance impacts of autocorrelated flows in multi-tiered systems", Performance Evaluation, Vol. 64, No. 9-12, pp. 1082–1101, 2007

[11] A. Riska, E. Riedel, "Long-range dependence at the disk drive level", Third International Conference on the Quantitative Evaluation of Systems, QEST 06, pp. 41-50, USA, September 11-14, 2006

[12] S. Sethi, S. Anupama, S. K. Jena, "Efficient load Balancing in Cloud Computing using Fuzzy Logic", IOSR Journal of Engineering, Vol. 2, No. 7, pp. 65-71, 2012

[13] U. Singhal, S. Jain, "A New Fuzzy Logic and GSO based Load balancing Mechanism for Public Cloud," International Journal of Grid Distribution Computing, Vol. 7, No. 5, pp. 97-110, 2014

[14] J. Tai, J. Zhang, J. Li, W. Meleis, N. Mi, "ARA: Adaptive Resource Allocation for Cloud Computing Environments under Bursty Workloads", 2011 IEEE International Performance Computing and Communications Conference, pp. 1-8, USA, November 17-19, 2011

[15] N. D. Naik, A. R. Patel, "Load Balancing Under Bursty Environment For Cloud Computing", International Journal of Engineering Research and Technology, Vol. 2, No. 6, pp. 17-26, 2013

[16] M. Ghorbani, Y. Wang, Y. Xue, M. Pedram, P. Bogdan, "Prediction and Control of Bursty Cloud Workloads: A Fractal Framework", 2014 International Conference on Hardware/Software Codesign and System Synthesis, pp. 1-9. New Delhi, October 12-17, 2014

[17] jFuzzyLogic, http://jfuzzylogic.sourceforge.net/html/index.html

[18] P. Cingolani, J. Alcalá-Fdez, "jFuzzyLogic: a Java Library to Design Fuzzy Logic Controllers According to the Standard for Fuzzy Control Programming", International Journal of Computational Intelligence Systems, Vol. 6, Suppl. 1, pp. 61–75, 2013

[19] P. Cingolani, J. Alcalá-Fdez, "jFuzzyLogic: a robust and flexible Fuzzy-Logic inference system language implementation", 2012 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Australia, June 10-15, 2012

[20] S. R. Pakize, S. M. Khademi, A. Gandomi, "Comparison Of CloudSim, CloudAnalyst And CloudReports Simulator in Cloud Computing", International Journal of Computer Science And Network Solutions, Vol. 2, No. 5, pp. 19-27, 2014

[21] B. Wickremasinghe, R. N. Calheiros, R. Buyya, "CloudAnalyst: A CloudSim-based Visual Modeller for Analysing Cloud Computing Environments and Applications", 2010 24th IEEE International Conference on Advanced Information Networking and Applications (AINA), pp. 446-452, Australia, April 20-23, 2010

[22] B. Wickremasinghe, "CloudAnalyst: A CloudSim-based Tool for Modelling and Analysis of Large Scale Cloud Computing Environments", 433-659 Distributed Computing Project, Csse Dept., University Of Melbourne, Australia, 2009