Fuzzy Dynamic Load Balancing in Virtualized Data Centers of SaaS Cloud Provider

Md. S. Q. Zulkar Nine, Department of Electrical and Computer Engineering, North South University, Dhaka, Bangladesh

Abul Kalam Azad, Department of Electrical and Computer Engineering, North South University, Dhaka, Bangladesh

Saad Abdullah, Department of Electrical and Computer Engineering, North South University, Dhaka, Bangladesh

Rashedur M Rahman, Department of Electrical and Computer Engineering, North South University, Dhaka, Bangladesh

ABSTRACT

Cloud computing provides a robust infrastructure that can facilitate computing power as a utility service. All the virtualized services are made available to end users in a pay-as-you-go basis. Serving user requests using distributed network of Virtualized Data Centers is a challenging task as response time increases significantly without a proper load balancing strategy. As the parameters involved in generating load in the Virtualized Data Center has imprecise effect on the overall load of Virtual Machine, a fuzzy load balancing strategy is required. This paper proposes two efficient fuzzy load balancing methods - Fuzzy Active Monitoring Load Balancer (FAM-LB) and Fuzzy Throttled Load Balancer (FT-LB) for the distributed SaaS cloud provider. The authors implemented a cloud model in simulation environment and compared the results of otheir novel approach with the existing techniques. Among them FT-LB has provided better performance compared to other scheduling algorithms.

Keyword: *Cloud Computing, Dynamic Load Balancing, SaaS., Type-I Fuzzy Systems*

1. INTRODUCTION

Cloud computing is a fast growing field both in computing industry and research. Cloud has been flourished under the idea that computing can be provided to the end users as a utility service like electricity. It comes with many types and layers of computing services like –Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Many industries have flourished in this business such as Amazon EC2 (Amazon, 2015), Salesforce (2015), Microsoft Azure (2015). End users now get relief of the trouble of dealing with the complex underlying structure of computing and they can have any service they want

DOI: 10.4018/IJFSA.2015070104

in a pay-as-you-go basis (Buyya et al., 2009). Therefore, cloud providers have to deal with users who are not permanent rather temporary ones. These users request service for a short time and then pay according to that. So cloud providers, particularly SaaS, are now facing a huge amount of user requests in a short time. Thus, a good load balancing strategy is a vital factor in providing a good response time.

Load Balancing in cloud environment is a fascinating challenge as it is essential for the growth of cloud industries. SaaS Cloud providers with intense user traffic require a good load balancing strategy. We focus on load balancing strategies in this paper. Our primary interest is to investigate the existing load balancing strategies and provide more robust techniques to improve performance. Many algorithms like Round Robin Scheduler, Active Monitoring Load Balancer, Throttled Load Balancer (Wickremasinghe et al., 2010), and Min-min Scheduler (Kon et al., 2011) have been discussed in literature. Each of them has its own pros and cons. Round robin and Min-min scheduler have very simple idea of load balancing but they often fail to address the problems of unpredicted internet traffic. Active Load Balancer schedules traffic based only on the current task allocation count and ignores all other important parameters. Throttled Load Balancer basically implements a queuing strategy to address the load balancing. The inherent structure of load balancing is rather imprecise than direct. It mostly depends on the parameters of resource requirements of the user requests. So their fuzzy implication on the overall load status is required to improve the performance of the Virtual Machines. To solve these problems, we proposed Fuzzy Active Monitoring Load Balancer (FAM-LB) and Fuzzy Throttled Load Balancer (FT-LB) which provides superior performance compared to existing ones. The novel characteristics of the proposed methodology include:

1. To make the load balancing decision more accurate we have introduced the memory, bandwidth and disk-space requirements of the user requests to predict the load status of the VMs.
2. To capture the imprecise relation between the various parameters (memory, bandwidth, and disk-space) and load status a Fuzzy Inference system (FIS) is introduced.

This paper is organized as follows: the literature review related to existing Load Balancing techniques are described in Section 2 while Section 3 describes our cloud architecture used in simulation. Our proposed algorithms are detailed in Section 4. The simulation and experimental results are provided in Section 5 followed by concluding remarks in Section 6.

2. LITERATURE REVIEW

2.1. Load Balancing in Virtual Data Centers

A Virtualized Data Center with unpredictable user traffic needs a good load balancing strategy. If all the load of a Data Center is routed to a few servers while the rest sitting idle, it automatically increases the response time needed to serve a request. Uniform distribution of load among all the virtual servers can provide a better response time. Existing literature provides numerous number of load balancing techniques. These techniques are described briefly below.

2.1.1. Round Robin Load Balancer (RR-LB)

This is the most primitive type of load balancer with very simple working principle (Wickremasinghe et al., 2010). It basically passes the request to the next Virtual Machine without any prior knowledge. It starts task assigning from the first VM and continues to do so. After assigning the task to the last VM in the list, it starts task assigning from the beginning. In Algorithm 1 Current Virtual Machine, , is used to circle the request among ***n*** number of VMs.

Algorithm 1. Round robin load balancer (RR-LB)

Pre-conditions: Current Virtual Machine, cV ,
Number of Virtual Machines, n;

Post-conditions: Suitable Virtual Machine, VM_{out} .

Request Scheduling:

$cV = cV + 1;$

```
1.  If   cV  >  n
a.  Then cV = 0;
2.  else
```

a. Then $VM_{out} = cV;$

```
3.  end
```

2.2.2. Min-Min Load Balancer (MM-LB)

In Min-min Load Balancer (Kon et al., 2011), the best response time is used to schedule the job to the appropriate VM. Best response time is the lowest time that a VM has taken to serve any request so far. It also computes the current response time of all VMs. Current response time is the time that was required to serve the last user request. At step 3a of Algorithm 2, if the current response time of a VM is lower than its best response time, MM-LB immediately updates the best response time with the current response time and schedules the VM. Otherwise, it schedules the VM that has the minimum best response time.

2.2.3. Active Monitoring Load Balancer (AM-LB)

Active Monitoring Load Balancer (Wickremasinghe et al., 2010) is more robust than the RR-Load Balancer. It actively monitors the current load status of each VMs and assigns task to the least loaded VM. Algorithm 3 provides an overview of the algorithm. When the Data Center Controller (DCC) receives a request, it immediately makes a query to the Active Monitoring Load Balancer (AM-LB) to get the suitable VM (steps 1-2). Active Load Balancer scans the current task allocation count of each VM and returns the VM with minimum task allocation count (steps 3-4). Then it increases its task allocation counter by 1 (step 5). After the task is executed in the allocated VM, DCC notifies the AM-LB to decrease the current allocation count of that VM (steps 6-7).

2.2.4. Throttled Load Balancer (T-LB)

Throttling the requests is an efficient load distribution strategy. Throttling can be implemented using different level of thresholds (Wickremasinghe et al., 2010). In Algorithm 4, a threshold value of 2 is used. It means each VM can serve maximum two requests simultaneously. In step 1, a user request is received by the DCC and it queries the Throttled Load Balancer (T-LB) for available VMs in step 2. T-LB checks whether the VM is busy or available (step 4). A Virtual Machine is available when no task is assigned in it. So T-LB immediately schedules it to serve the user request and changes its status from Available to Running. Running means VM has one task assigned to it. If no VM with status Available is found, then it tries to find a VM with Running status. If found then it updates the VM status from Running to Busy and schedules the request in it. When T-LB finds all the VMs status Busy, it notifies the DCC by setting VM_{out} to -1. DCC immediately puts the request in the queue. When a VM finishes a task it notifies the DCC and DCC upgrades the VM status to one step higher availability (i.e. from Running to Available or from Busy to Running) as shown in steps 5 through 8. The DCC then checks the queue, $\boldsymbol{Q}$. If the queue is not empty, it dequeues a request from $\boldsymbol{Q}$ and repeats from step 3 or else it starts scheduling a new request from step 2.

Algorithm 2. Min-min load balancer (MM-LB)

Pre-requisite: Best Response time **i^th** VM, $R_t(i)$;
Current Response time of **i^th** VM, $C_t(i)$;
User Request, r ;Number of Virtual Machines, n;
Post-requisite: Suitable Virtual Machine, VM_{out} .
Request Scheduling:
1. Initialize Best response time $B_t(i) = 0$;
2. $C_t(i) = \infty$;
3. For each **VM** i
a. If $C_t(i) < B_t(i)$
i. Then $B_t(i) = C_t(i)$;
$VM_{out} = i$;
b. Else
i. Data Center scans $B_t(i)$ to get $\min(R_t(i))$
$VM_{out} = i$
4. end

Algorithm 3. Active monitoring load balancer (AM-LB)

Pre-conditions: Current Request Allocation Counter, $V_c(i)$,
User Request, r ; Number of Virtual Machines, n;
Post-condition: Suitable Virtual Machine, VM_{out} .
Request Scheduling:
1. Data Center receives, r from user-base via internet.
2. Data Center Queries the Active Monitoring Load Balancer (AM-LB).
3. AM-LB scans the $V_c(i)$, find the least loaded VM;
4. AM-LB sends VM_{out} to Data Center Controller.
5. Data Center update $V_c(VM_{out}) = V_c(VM_{out})$ **+ 1;**
After Processing De-allocation:
6. When request served,
Data Center update $V_c(VM_{out}) = V_c(VM_{out})$ **- 1;**
7. Continue From **Step 2**
8. end

3. ARCHITECTURE

The overall cloud architecture is a complex hierarchical structure. Interconnected virtualized Data Centers with high scalability and availability are crucial parts of the Cloud. Figure1 shows a simple version of the Virtualized Data Center where VMs are the basic computing blocks. Here the Userbase defines the overall behavior of the users in a particular geographic region. In our simulation, for simplicity, we combine the user requests from huge number of users in a Userbase to group them to internet cloudlets. These internet cloudlets are sent to a Data Center via internet. The main characteristic of the internet is transmission delay. We know total delay can be described as,

$$T_{total} = T_{latency} + T_{transfer} \quad (1)$$

Algorithm 4. Throttled load balancer (T-LB)

Pre-conditions: Request Queue, Q ; User Request, r ;
Virtual Machine State, $V_s(i) \epsilon \{Busy, Running, Available\}$;
Number of Virtual Machines, n;

Post-condition: Suitable Virtual Machine, VM_{out} .

Request Scheduling:
1. Data Center receives, r from user-base via internet.
2. Data Center Queries the Throttled Load Balancer (T-LB).
3. T-LB scans $V_s(i)$ to see the Virtual Machine Status of VMs.
4. For each VM i
a. If $V_s(i) = Available$
i. Then $VM_{out} = i$;
ii. $V_s(i) = Running$
b. else if $V_s(i) = Running$
i. Then $VM_{out} = i$;
ii. $V_s(i) = Busy$
c. else
i. $VM_{out} = -1$;
ii. Data Center queues the request r in Q ;

After Processing De-allocation:
5. When request served, T-LB updates V_s
6. If $V_s(VM_{out}) = Running$
a. Update $V_s(VM_{out}) = Available$
7. If $V_s(VM_{out}) = Busy$
a. Update $V_s(VM_{out}) = Running$
8. Data Center Checks the queue, Q
9. If $(Q \; != O)$
a. r= Dequeue(Q)
b. Continue from **Step 3**
10. else continue from **Step 2**
11. end

Where, $T_{latency}$ is the network latency and $T_{transfer}$ is the time taken to transfer a fixed sized data from source to destination.

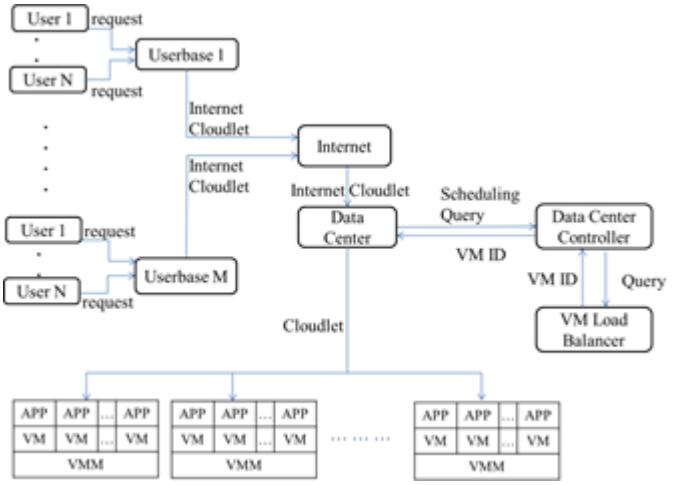
$T_{latency}$ can be modeled as a poisson distribution.

$$T_{transfer} = \frac{D}{Bw_{peruser}} \tag{2}$$

$$Bw_{peruser} = \frac{Bw_{total}}{N_r} \tag{3}$$

Where, Bw_{total} is the total available bandwidth of the Data Center, $Bw_{peruser}$ is the amount of bandwidth allocated per user and N_r is the number of request currently in transmission. The DCC is the administrative module of the Data Center. It manages the hardware resource along with VM and network resource administration within the Data Center. A specific part

Figure 1. Simple virtualized data center



of DCC is responsible for distributing an enormous amount of traffic to the VMs in a way that confirms optimal response time for the user requests. It is known as the VM Load Balancer. DCC makes query to the VM Load Balancer for the most suitable VM that can process the request with minimum response time. Actual VMs are installed over the Virtual Machine Manager (VMM) where VMMs are installed directly to the hardware. These VMMs work as an abstraction layer and control the interaction of VMs with the hardware resources. Applications are installed in the VMs.

4. PROPOSED LOAD BALANCING ALGORITHMS

To enhance the load balancing capacity of the Virtualized Data Center, we have proposed two novel models and simulated their capacity with the existing load balancing algorithms. The load status of the Virtual Machines is conceptually imprecise. The parameters like memory, bandwidth or disk-space requirements have a fuzzy implication (Zadeh, 1975) on the overall load status of the VMs.

4.1. Fuzzy Active Monitoring Load Balancer (FAM-LB)

Our novel algorithm takes into account three resource requirements of user requests. After that it computes the resource requirements of the existing user requests in the VM. Then it uses a Type-I Fuzzy System to infer the load status of the Virtual Machines. The crisp values of memory, bandwidth and disk-space requirements are converted to fuzzy values. A generalized form for the membership function, U_M of fuzzy values of memory, *M= {VeryLow, Low, Med, High, VeryHigh}* can be described by equation (4).

$$U_M\left(M\right) = e^{-\left(M-c\right)^2 / 2\sigma^2} \tag{4}$$

Here for every fuzzy value, the variance of Gaussian distribution, σ =45 is the same. Table 1 shows the mean c for every fuzzy value. Figure 2 shows the graphical representation of the fuzzy variables. Each Virtual Machine is assigned 512 MB memory.

For bandwidth requirement, *B = {VeryLow, Low, Med, High, VeryHigh}* estimation we have used the following equations. These

Table 1. Center of gaussian membershiip function of diffenrent fuzzy values corresponds to memory

Fuzzy Values	VeryLow	Low	Med	High	VeryHigh
Center c	0	150	270	375	512

fuzzy values are depicted in Figure 3. Assigned Bandwidth of each VM is 1000 Mbps.

$$\mu_{VeryLow}(B) = -\frac{B}{200} + 1, 0 \le \mathrm{B} \le 200 \quad (5)$$

$$\mu_{LOW}(B) = \begin{cases} \frac{B}{150} - \frac{2}{3}, & 100 \le B \le 250 \\ -\frac{B}{150} + \frac{8}{3}, & 250 \le \mathrm{B} \le 400 \end{cases}$$

$$\mu_{MED}(B) = \begin{cases} \frac{B}{200} - \frac{3}{2}, & 300 \le B \le 500 \\ -\frac{B}{200} + \frac{7}{2}, & 500 \le \mathrm{B} \le 700 \end{cases}$$

$$\mu_{HIGH}(B) = \begin{cases} \frac{B}{150} - 4, & 600 \le B \le 750 \\ -\frac{B}{150} + 6, & 750 \le \mathrm{B} \le 900 \end{cases}$$

$$\mu_{VeryHigh}(B) = \frac{B}{200} - 4, \quad 800 \le \mathrm{B} \le 1000$$

To estimate the disk-space requirement, *D= {Low, Med, High}* we have used the following membership functions graphically shown in Figure 4.

$$\mu_{LOW}(d) = -\frac{d}{2000} + 1, 0 \le \mathrm{d} \le 2000 \quad (10)$$

$$\mu_{MED}(d) = \begin{cases} \frac{D}{3500} - \frac{2}{7}, & 1000 \le d \le 4500 \\ -\frac{D}{3500} + \frac{16}{7}, & 4500 \le \mathrm{d} \le 8000 \end{cases} \quad (11)$$

$$\mu_{HIGH}(d) = \frac{d}{3000} - \frac{7}{3}, \quad 1000 \le \mathrm{d} \le 4500 \quad (12)$$

The output load status can be inferred using following membership equations-

$$U_L(l) = e^{-(l-c)^2 / 2\delta^2} \quad (13)$$

Here all elements of disk-space *D = {VeryBusy, Busy, Moderate, Available, HighlyAvailable}* is a Gaussian membership function with

Figure 2. Membership function of verylow, low, med, high, veryhigh for memory

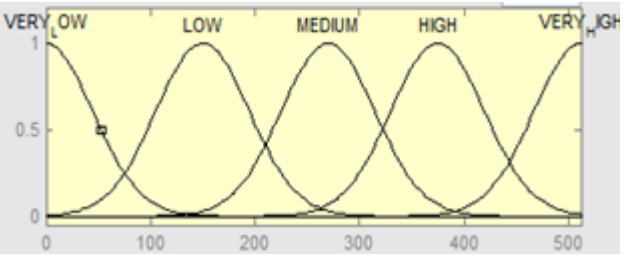


Figure 3. Membership function of verylow, low, med, high, veryhigh for bandwidth

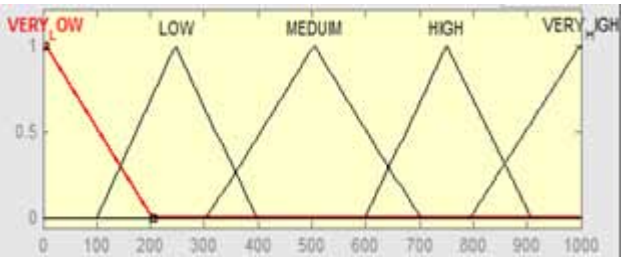
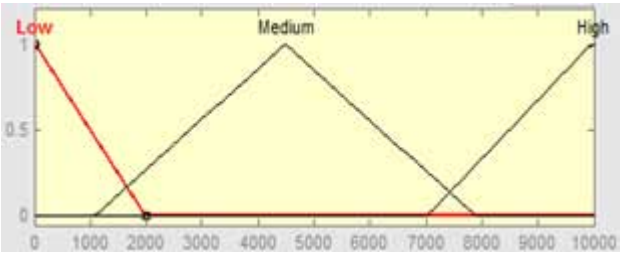


Figure 4. Membership function of low, medium, high for disc space



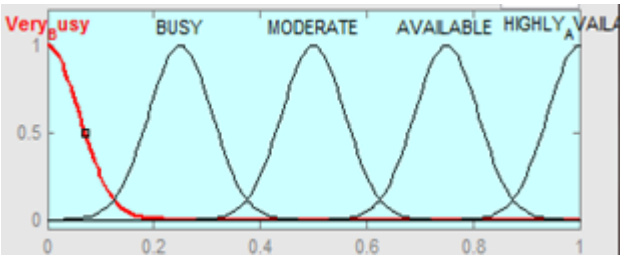
variance σ =0.06 and the mean value c for each variable is provided in Table. 2. There graphical representation is shown in Figure 5. We have assigned 10GB disk-space to each VM.

We then construct Fuzzy Inference System with 75 rules using Mamdani Min implication (Klir & Yuan, 1975; Mamdani & Assilian, 1975). For example, if memory requirement is ***veryLow***, Bandwidth requirement is ***Low,*** and disk-space requirement is ***Low*** then the virtual memory load status is ***highlyAvailable.*** All these 75 rules are combined with Else operator. The interpretation of connective Else is OR operation. The fuzzy rules is shown in Table 3. These rules are generated based on the resource requirement of the tasks that are being served by the VMs. When a task arrives in the Data Center Controller, it generates a query to FAM-LB to get a VM where the task can be scheduled (steps 1-2). FAM-LB estimates three resource requirements of the existing tasks in all VM (step 3). In step 4, it evaluates the load status of each VM using Fuzzy Inference as described above. Our novel algorithm selects the least busy VM using Fuzzy Inference system and sends it to the Data Center Controller (step 5). Data Center Controller increases the current task allocation counter of the selected VM (Step 6). As soon as the task has finished, the DCC reduce the task allocation counter as shown in step 7. After that DCC prepare itself to assign another user request (step-8).

Table 2. Center of gaussian membershiip function of diffenrent fuzzy values corresponds to load

Fuzzy Values	VeryBusy	Busy	Moderate	Available	Highly Available
Mean c	0	0.25	0.50	0.75	1

Figure 5. Membership function of verybusy, busy, moderate, available, highlyavailable for load status



4.2. Fuzzy Throttled Load Balancer (FT-LB)

As we have seen earlier that an efficient queuing strategy with a proper threshold can provide state of the art performance in load balancing. We use a Queuing threshold value of 5 for our novel algorithm. Memory, bandwidth, disk-space requirements are also considered to schedule tasks within the threshold limit. If the threshold is exceeded it starts queuing like the T-LB. At steps 1 – 3, DCC queries the FT-LB for a VM and FT-LB starts scanning the load status of each VM. In step 5, if any VM with load status Available is found, then FT-LB immediately schedules the VM. When there is no VM available and all VMs current task allocation count is lower than the queuing threshold, it makes decision using Fuzzy Inference system which is identical to the FIS of FAM-LB algorithm (step 6). And when all the VMs are running over the queuing threshold limit, FT-LB starts to queue the requests. At steps 9-10, when it finishes the job, the DCC changes the VM state to the next higher availability state. Then the DCC checks the queue for user requests. If found it enqueques the Q and start from step 2. Otherwise it starts from step 1.

5. SIMULATIONS AND RESULTS

Large scale internet application simulation is very difficult to perform as it requires large scale hardware which is hard to manage. It can be done in small test-bed using a small environment, but it can hardly accommodate the huge unpredicted internet traffic. In order to consider these we use simulator software CloudAnalyst (Wickremasinghe et al., 2010) which is a special version of the widely used virtual environment simulator CloudSim (Calheiros et al., 2011; Buyya, Ranjan & Calheiros, 2009).

5.1. Simulation Tools

CloudAnalyst is a robust cloud simulator which facilitates realistic cloud environment with large scale internet load. The main advantage of the simulator is that it is flexible enough to test and design the new ideas. It also ensures the repeatability of the experiment. It means that the simulator generates similar environment with the same parameters. It helps us to simulate different load balancing stretigies using similar envirionment so that a comparision can be made among them.

5.2. Simulation Results

To simulate a large scale web application we have used traffic load of the popular social networking site Facebook. It has over 981 million (Socialbakers, 2013) active users recorded on January, 2013. We have used the approximate distribution of Facebook users across the globe which is given below in Table 4.

For simulation we have used 1/2000 th of the scale of Facebook user distribution as shown in Table 5. In our simulation, for simplicity, we

Table 3. Rule table for fuzzy inference sysem

SL	Memory Status	Bandwidth Status	Diskspace Status	Load status
1	VeryLow	VeryLow	Low	HighlyAvailable
2	VeryLow	VeryLow	Medium	HighlyAvailable
3	VeryLow	VeryLow	High	HighlyAvailable
4	VeryLow	Low	Low	HighlyAvailable
5	VeryLow	Low	Medium	HighlyAvailable
6	VeryLow	Low	High	HighlyAvailable
7	VeryLow	Medium	Low	HighlyAvailable
8	VeryLow	Medium	Medium	Available
9	VeryLow	Medium	High	Available
10	VeryLow	High	Low	Available
11	VeryLow	High	Medium	Available
12	VeryLow	High	High	Available
13	VeryLow	VeryHigh	Low	HighlyAvailable
14	VeryLow	VeryHigh	Medium	Available
15	VeryLow	VeryHigh	High	Available
16	Low	VeryLow	Low	HighlyAvailable
17	Low	VeryLow	Medium	HighlyAvailable
18	Low	VeryLow	High	HighlyAvailable
19	Low	Low	Low	HighlyAvailable
20	Low	Low	Medium	HighlyAvailable
21	Low	Low	High	Available
22	Low	Medium	Low	Available
23	Low	Medium	Medium	Available
24	Low	Medium	High	Available
25	Low	High	Low	HighlyAvailable
26	Low	High	Medium	Available
27	Low	High	High	Available
28	Low	VeryHigh	Low	HighlyAvailable
29	Low	VeryHigh	Medium	Available
30	Low	VeryHigh	High	Available
31	Medium	VeryLow	Low	HighlyAvailable
32	Medium	VeryLow	Medium	Available
33	Medium	VeryLow	High	Available
34	Medium	Low	Low	HighlyAvailable
35	Medium	Low	Medium	Moderate
36	Medium	Low	High	Moderate
37	Medium	Medium	Low	Available
38	Medium	Medium	Medium	Moderate
39	Medium	Medium	High	Moderate
40	Medium	High	Low	Available

continued on following page

Table 3. Continued

SL	Memory Status	Bandwidth Status	Diskspace Status	Load status
41	Medium	High	Medium	Moderate
42	Medium	High	High	Busy
43	Medium	VeryHigh	Low	Available
44	Medium	VeryHigh	Medium	Moderate
45	Medium	VeryHigh	High	Busy
46	High	VeryLow	Low	HighlyAvailable
47	High	VeryLow	Medium	Available
48	High	VeryLow	High	Available
49	High	Low	Low	HighlyAvailable
50	High	Low	Medium	Available
51	High	Low	High	Available
52	High	Medium	Low	Available
53	High	Medium	Medium	Moderate
54	High	Medium	High	Busy
55	High	High	Low	Available
56	High	High	Medium	Moderate
57	High	High	High	Busy
58	High	VeryHigh	Low	Available
59	High	VeryHigh	Medium	Busy
60	High	VeryHigh	High	Busy
61	VeryHigh	VeryLow	Low	HighlyAvailable
62	VeryHigh	VeryLow	Medium	Available
63	VeryHigh	VeryLow	High	Available
64	VeryHigh	Low	Low	Available
65	VeryHigh	Low	Medium	Available
66	VeryHigh	Low	High	Available
67	VeryHigh	Medium	Low	Available
68	VeryHigh	Medium	Medium	Moderate
69	VeryHigh	Medium	High	Busy
70	VeryHigh	High	Low	Available
71	VeryHigh	High	Medium	Moderate
72	VeryHigh	High	High	Busy
73	VeryHigh	VeryHigh	Low	Available
74	VeryHigh	VeryHigh	Medium	Moderate
75	VeryHigh	VeryHigh	High	VeryBusy

Algorithm 5. Fuzzy Active Monitoring Load Balancer (FAM-LB)

Pre-Condition: Current Request Allocation Counter, $V_c(i)$, User Request, r ; Number of Virtual Machines, n;
Fuzzy Variables:

Memory Requirement, $M_v \in \{veryLow, Low, Medium, High, veryHigh\}$,

Bandwidth Requirement, $B_v \in \{veryLow, Low, Medium, High, veryHigh\}$,

Disk Space Requirement, $D_v \in \{Low, Medium, High\}$;

Virtual Machine Status, $S_v \{veryBusy, Busy, Modedate, Available, highlyAvailable\}$;

Post-Condition: Suitable Virtual Machine, VM_{out} .

Request Scheduling:

1. Data Center receives, r from user-base via internet.
2. Data Center Queries the Fuzzy Active Monitoring Load Balancer.
3. FAM-LB estimate the required M_v, B_v, D_v of request in each VM.
4. VM_{out} = **Fuzzy-Inference-System**(M_v, B_v, D_v);
5. FAM-LB sends VM_{out} to Data Center Controller.
6. Data Center update $V_c(VM_{out}) = V_c(VM_{out})$ **-+1;**

After Processing De-allocation:

7. When request served,

Data Center update, $V_c(VM_{out}) = V_c(VM_{out})$ **- 1;**

8. Continue From **Step 2**
9. end

assume that most users use Facebook on evening (7.00–9.00 pm local time) and one tenth of the users are logged on during off-peak hours. We further assume that each user makes 5 requests every five minutes. We have simulated the load that follows the poison process. For our experiment, 50 VMs are deployed in the Data Center, each with 512 MB of memory and processor capacity with speed of

100 MIPS. The FIS proposed in our novel algorithms contain 75 inference rules which are used to take the fuzzy decision.

The output surface of FIS for any two input pair is shown in Figure 6, Figure 7, and Figure 8. This output surfaces are designed in a way that they predict higher output value when inputs have higher values, (i.e. when memory, bandwidth, and disk-space requirements are high then the load status is High) and vice versa.

For simplicity the CloudSim simulator generates load with fixed resource requirements which is not ideal to analyze the performance of the load balancing algorithm. To overcome this problem we modified the CloudAnalyst version to generate load with variable resource requirements. Then we have tested our novel load balancing algorithms against the existing approaches in both situations. We have used the DC user request processing time and overall response time to compare the performances. To evaluate the performance we have recorded both DC processing time and average response time for each job. The simulator is designed in such a way that in can generate identical hourly

Algorithm 6. Fuzzy throttled load balancer (FT-LB)

PRE-REQUISITE: REQUEST QUEUE, Q ; USER REQUEST, r ;
VIRTUAL MACHINE STATE, $V_s(i) \epsilon \{Busy, Available\}$;
CURRENT REQUEST ALLOCATION COUNTER, $V_c(i)$,
NUMBER OF VIRTUAL MACHINES, n;
FUZZY VARIABLES:
MEMORY REQUIREMENT, $M_v \epsilon \begin{Bmatrix} veryLow, Low, Medium, \\ High, veryHigh \end{Bmatrix}$,
BANDWIDTH REQUIREMENT, $B_v \epsilon \begin{Bmatrix} veryLow, Low, Medium, \\ High, veryHigh \end{Bmatrix}$,
DISK SPACE REQUIREMENT, $D_v \epsilon \begin{Bmatrix} Low, Medium, \\ High \end{Bmatrix}$;
VIRTUAL MACHINE STATUS, $S_v \begin{Bmatrix} veryBusy, Busy, Modedate, \\ Available, highlyAvailable \end{Bmatrix}$;

POST-REQUISITE: SUITABLE VIRTUAL MACHINE, VM_{out} .
REQUEST SCHEDULING:
1. DATA CENTER RECEIVES, **r** FROM USER-BASE VIA INTERNET.
2. DATA CENTER QUERIES THE FUZZY THROTTLED LOAD BALANCER.
3. FT-LB SCANS $V_s(i)$ TO SEE THE LOAD STATUS OF VMS.
4. FOR EACH **VM** i
5. IF $V_s(i)$ = $Available$
I. THEN $VM_{out} = i$;
B. ELSE
I. $VM_{out} = -1$;
6. IF $VM_{out} = -1$ $and V_c(i)$ **< QUEUINGTHRESHOLD**
A. VM_{out} = **FUZZY-INFERENCE-SYSTEM**(M_v, B_v, D_v);
7. ELSE
A. DATA CENTER QUEUE THE REQUEST r IN Q ;
AFTER PROCESSING DE-ALLOCATION:
8. WHEN REQUEST SERVED, FT-LB UPDATES V_s
9. IF $V_s(VM_{out}) = Busy$
A. UPDATE $V_s(VM_{out}) = Available$
10. DATA CENTER CHECKS THE QUEUE, Q
11. IF $(Q \; != O)$
A. R=ENQUEUE(**Q**)
B. CONTINUE FROM **STEP 3**
12. ELSE CONTINUE FROM **STEP 2**
13. END

load for each algorithm. The hourly load is depicted in Figure 9.

In case of fixed load scenario, the hourly reports of average processing time of DC are depicted in Figure 10, Figure 11, Figure 12, Figure 13, Figure 14, and Figure 15. It can be seen that the processing time increases in proportion to the hourly loading of DC as depicted in Figure 9. In Figure 11, the MM-LB processing time increases proportionally with the increase

Table 4. Simulated data

Userbase	Region	Time Zone	Peak Hours (Local time)	Peak Hours (GMT)	Simultaneous Online Users During Peak Hrs	Simultaneous Online Users During Off-peak Hrs
UB1	**0 – N. America**	**GMT – 6.00**	**7.00–9.00 pm**	**13:00-15:00**	**121,000**	**12,100**
UB2	**1 – S. America**	**GMT – 4.00**	**7.00–9.00 pm**	**15:00-17:00**	**71,500**	**7,150**
UB3	**2 – Europe**	**GMT + 1.00**	**7.00–9.00 pm**	**20:00-22:00**	**126,000**	**12,600**
UB4	**3 – Asia**	**GMT + 6.00**	**7.00–9.00 pm**	**01:00-03:00**	**139,500**	**13,950**
UB5	**4 – Africa**	**GMT + 2.00**	**7.00–9.00 pm**	**21:00-23:00**	**25,500**	**2,550**
UB6	**5 – Ocenia**	**GMT + 10.00**	**7.00–9.00 pm**	**09:00-11:00**	**7,000**	**700**

Table 5. Region-wise facebook user (Socialbakers, 2013).

Region	Users
North America	242 million
South America	143 million
Europe	252 million
Asia	279 million
Africa	51 million
Ocenia	14 million

Figure 6. Output surface for bandwidth and memory

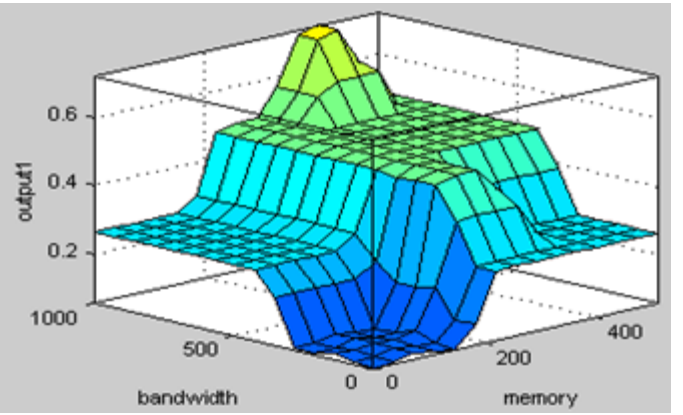
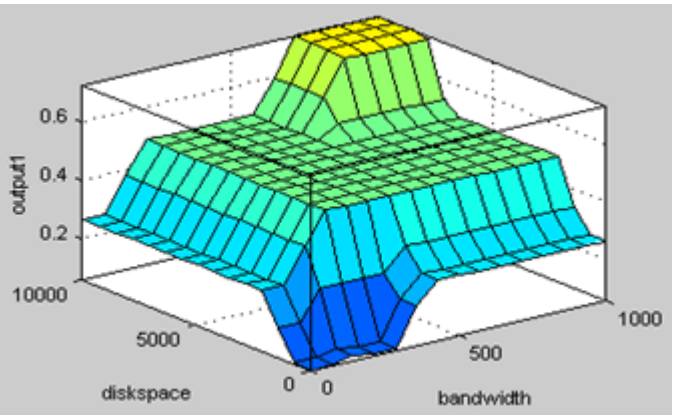


Figure 7. Output surface for diskspace and bandwidth



of load and average processing time. Compared to that, RR-LB can handle user requests more efficiently. The processing time needed for AM-LB and T-LB are merely same throughout the off-peak hours. A summary of both processing time and response time is provided in Table 6 and 7. It is seen that the MM-LB produces the worst average request processing time (1450.70 ms) compared to rest of the algorithms. Both of our novel algorithms provide superior performance than existing ones. With fuzzy decision making scheme, our novel algorithms provide more robust load balancing in both peak and off peak hours and reduce processing time as well. Their average response time is recorded as 12.47 ms and 6.57 ms respectively. FT-LB outperforms all other load balancers. Overall response time is the time needed to serve a user after the request is made. It includes internet routing time needed for both sending and receiving the request. The overall response time of each load balancer is shown in Table 7. As expected, MM-LB produces the worst response time and our novel algorithms have reduced the response time to 304.78 ms and 303.32 ms. However, the performance increase is not very significant due to the fixed load scenario. Fixed load scenario is highly unrealistic for load balancing. As all tasks demand same amount of resources, it favors the techniques that do not consider resource requirements in decision making.

To evaluate those algorithms in more challenging scenario we simulated the tasks with variable resource requirements. All the tasks come with their own resource requirements in real world scenario. To model the variable resource requirement we have used Gaussian distribution for bandwidth, disk space and memory. So values in the suitable range of the distribution are generated to tag them with resource requirements. The hourly average processing time for AM-LB, T-LB and our proposed two algorithms, FAM-LB and FT-LB are recorded in Figure 16 to Figure 19. In this scenario, the average processing time for both existing algorithms go higher and our proposed novel algorithms can successfully reduce the processing time. A comprehensive summary of both hourly processing time and overall response time is provided in Table 8 and Table 9. It can be observed that in this challenging scenario, MM-LB produces the highest processing time of 1289.65 ms and RR-LB produces moderately high response time of 17.89 ms. However, our novel algorithms – FAM-LB and FT-LB provide superior performance with response time of 7.23 ms and 4.21 ms. It is clear that FT-LB produces a result which is 41% faster than its closest competitor, FAM-LB. The other performance metric is overall response time

Figure 8. Output surface for diskspace and memory

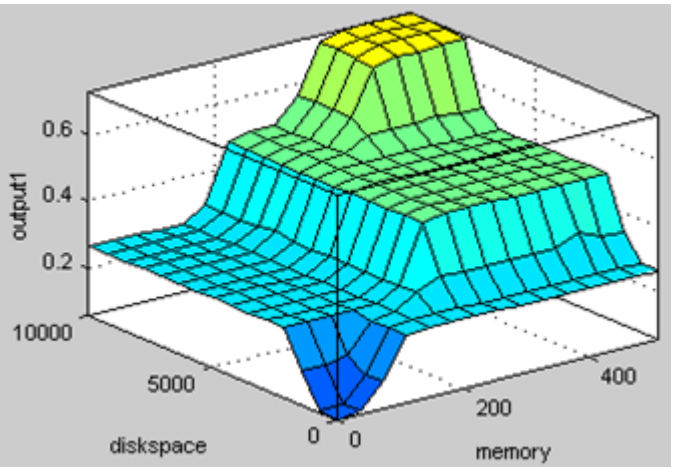


Figure 9. Data center hourly loading for all load balancers

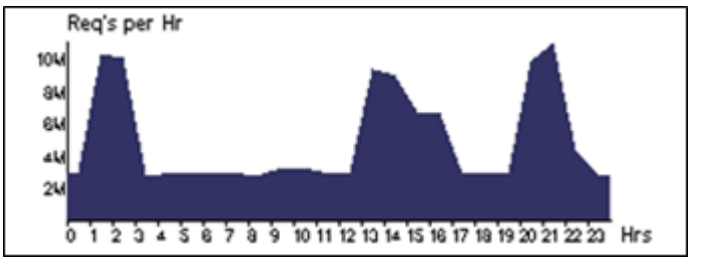
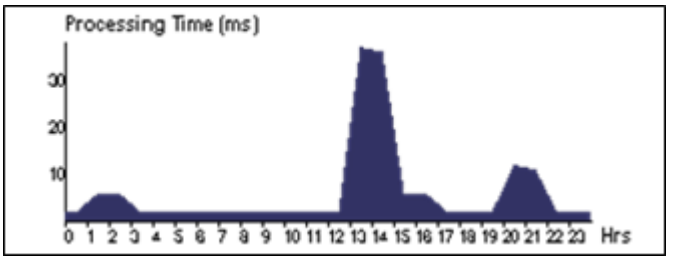


Figure 10. Data center hourly average processing times for RR-LB (fixed load)



as recorded in Table 9. It shows that FAM-LB and FT-LB generate faster response time than other existing algorithms. Between them, FT-LB produces 27.22% improved performance.

6. CONCLUSION AND FUTURE WORK

This paper has proposed two Fuzzy Dynamic Load Balancing algorithms and compares the performance with the existing load balancing algorithms. Both of our proposed algorithms

Figure 11. Data center hourly average processing times for MM-LB (fixed load)

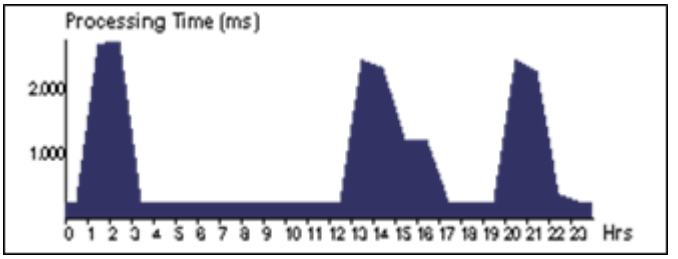


Figure 12. Data center hourly average processing times for AM-LB (fixed load)

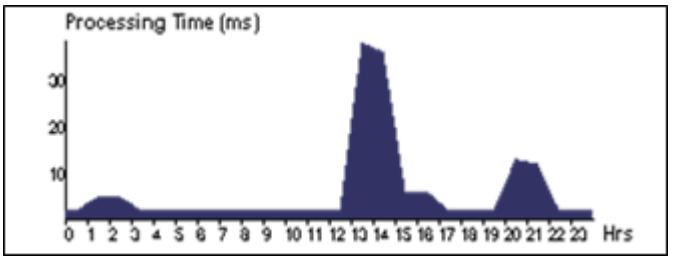


Figure 13. Data center hourly average processing times for T-LB (fixed load)

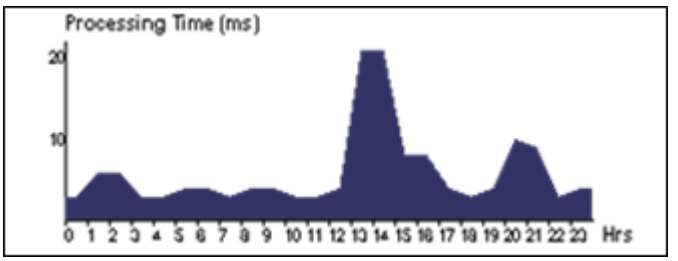


Figure 14. Data center hourly average processing times for FAM-LB (fixed load)

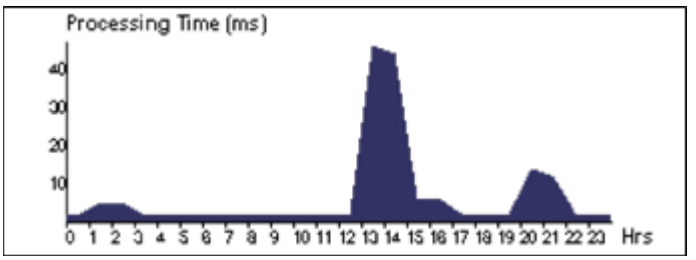


Figure 15. Data center hourly average processing times for FT-LB (fixed laod)

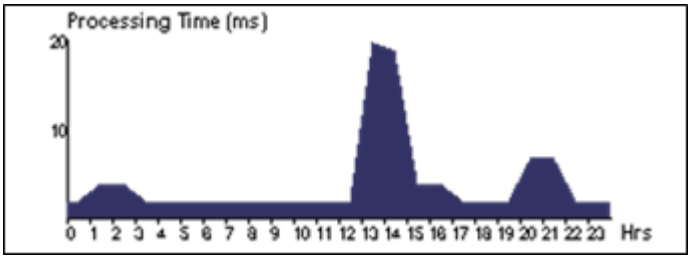


Table 6. Data center processing time (equal size load)

Algorithm	Avg (ms)	Min (ms)	Max (ms)
RR-LB	10.58	0.08	67.44
MM-LB	1450.70	25.00	6128.44
AM-LB	10.49	0.08	67.70
T-LB	8.54	0.08	134.90
FAM-LB	12.45	0.08	77.27
FT-LB	6.57	0.08	59.95

Table 7. Overall response time summary(equal size load)

Algorithm	Avg (ms)	Min (ms)	Max (ms)
RR-LB	307.23	40.39	674.80
MM-LB	1745.59	176.35	6463.57
AM-LB	307.17	40.39	674.80
T-LB	305.22	40.39	715.87
FAM-LB	304.78	40.32	674.80
FT-LB	303.32	40.39	674.80

Figure 16. Data center hourly average processing times for AM-LB (variable load)

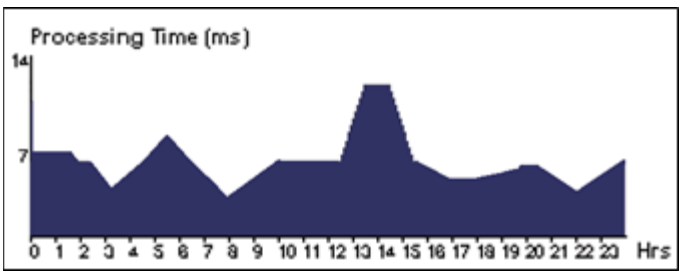


Table 8. Data center processing time (variable size load)

Algorithm	Avg (ms)	Min (ms)	Max (ms)
RR-LB	17.89	1.70	99.75
MM-LB	1289.65	12.67	2397.54
AM-LB	13.52	0.23	54.82
T-LB	7.84	0.74	64.21
FAM-LB	7.23	0.14	69.32
FT-LB	4.21	0.34	62.57

Table 9. Overall response time summary (variable size load)

Algorithm	Avg (ms)	Min (ms)	Max (ms)
RR-LB	327.42	31.98	527.85
MM-LB	1367.84	132.74	3125.85
AM-LB	315.49	24.97	494.96
T-LB	308.22	35.74	439.92
FAM-LB	271.84	37.72	512.60
FT-LB	197.84	23.97	389.21

Figure 17. Data center hourly average processing times for T-LB (variable load)

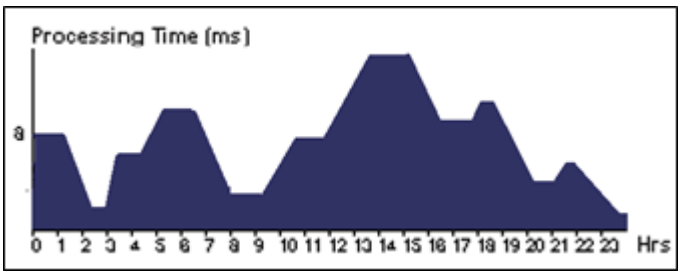


Figure 18. Data center hourly average processing times for FAM-LB (variable load)

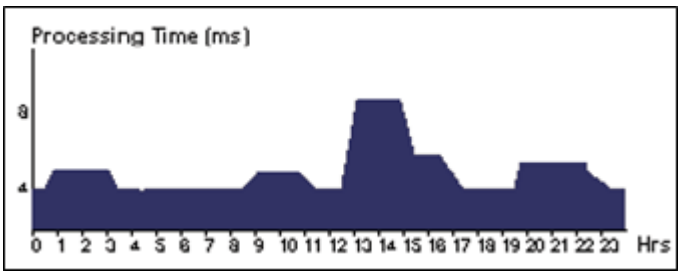


Figure 19. Data center hourly average processing times for FT-LB (variable load)

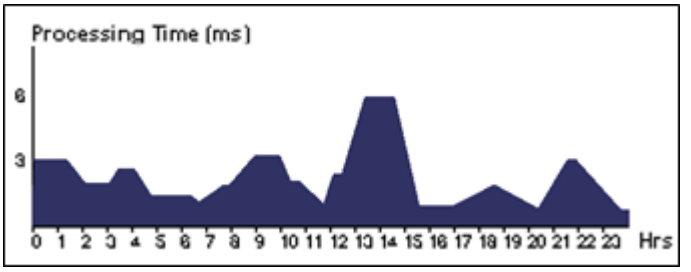
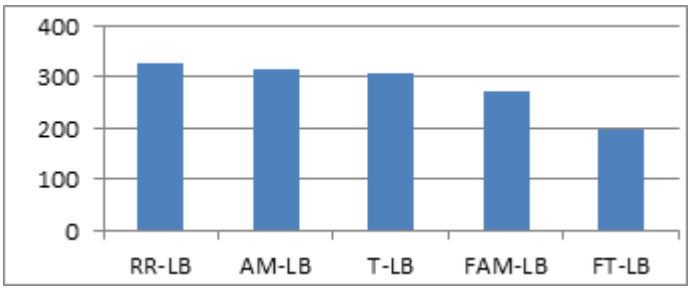


Figure 20. Comparison of overall response time among different load balancing algorithms



provide superior performance over the existing approaches. Introducing fuzzy decision making based on memory, bandwidth, and disk-space requirements of the requests have enabled them to provide superior performance than others. Improvements that are made over the existing methods are summarized below.

6.1. Improvements Over RR-LB:

Both of our proposed models make load balancing decision based on resource requirement which is absent in RR-LB as it just passes the request to the next VM. Compared to RR-LB, the response time and DC processing time of our novel approach FT-LB decreases by a factor of 4 and 1.5 respectively.

6.2. Improvements Over AM-LB

In AM-LB, it tries to find the least loaded VM based on the number of tasks assigned on each VM which is very unrealistic in case of tasks with variable resource requirements. Fuzzy inference based load estimation enables proposed FT-LB to produce approximately 3 times faster processing time than the AM-LB. The overall response time is also decreased by a factor of 1.5.

6.3. Improvements over T-LB

T-LB uses a threshold to schedule tasks and does not consider the dynamic state of the load in Data Center. Experimental results show that by introducing current resource load status in both algorithms provides superior performance. DC processing time is decreased by a factor of 1.8 where the overall response time is decreased by 36% compared to the processing time and response time of T-LB.

The main shortcoming of this paper is that we have not considered and investigated the concept of load migration and VM migration in our work. The migration strategies can affect the performance. However, a good migration policy can improve the performance even more.

For future work, the decision strategy can be extended to a broker that would route the user requests to different Data Centers in a distributed environment. The effect of load migration in load balancing can be investigated to improve the performance of the Cloud environment.

REFERENCES

Amazon, *Amazon Elastic Compute Cloud (Amazon EC2)*. Retrieved January 6, 2015, from http://aws.amazon.com/ec2/

Buyya, R., Ranjan, R., & Calheiros, R. (2009). 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 Conference*, Leipzig, Germany, (pp. 1-11). doi:10.1109/HPCSIM.2009.5192685

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. *Future Generation Computer Systems*, *25*(6), 599–616. doi:10.1016/j.future.2008.12.001

Calheiros, R., Ranjan, R., Beloglazov, A., Rose, C. A. F. D., & Buyya, R. (2011). CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. [SPE]. *Software, Practice & Experience*, *41*(1), 23–50. doi:10.1002/spe.995

Klir, G. J., & Yuan, B. (1995). *Fuzzy and Fuzzy Logic: Theory and Applications* (1st ed.). USA: Prentice Hall.

Kon, X., Lin, C., Jiang, Y., Yan, W., & Chu, X. (2011). Efficient Dynamic Task Scheduling in Virtualized Data Centers with Fuzzy Prediction. *Journal of Network and Computer Applications*, *34*(4), 1068–1077. doi:10.1016/j.jnca.2010.06.001

Mamdani, E. H., & Assilian, S. (1975). An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, *7*(1), 1–13. doi:10.1016/S0020-7373(75)80002-2

Salesforce. Retrieved January 6, 2015, from http://www.salesforce.com Microsoft Azure. Retrieved January 6, 2015 from http://www.windowsazure.com. Socialbakers. Retrieved June 6, 2013 from http://www.socialbakers.com/countries/continents/

Wickremasinghe, B., Calheiros, R., & Buyya, R. (2010). CloudAnalyst: A CloudSim-based Visual Modeller for Analysing Cloud Computing Environments and Applications, In *Proceedings of the 24th International Conference on Advanced Information Networking and Applications (AINA)*, Perth, Australia (pp.446-452). doi:10.1109/AINA.2010.32

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, *8*(3), 338–353. doi:10.1016/S0019-9958(65)90241-X

Abul Kalam Azad has graduated from Islamic University, Kushtia, Bangladesh with a BS in Computer Science and Engineering. He is currently doing Masters in Computer Science and Engineering in North South University, Dhaka, Bangladesh. He has worked on Cloud Computing, Grid Computing, Speech Processing, Image Processing, Fuzzy Logic and Neural Networks. His research interests include Cloud Computing, Intelligent Systems, Bioinformatics and Data Mining. At present he is working as a Software Engineer in Computer Ease Limited, Dhaka, Bangladesh.

Md. S. Q. Zulkar Nine is currently working as a Research Assistant (RA) of the Electrical and Computer Engineering (ECE) department of North South University. He is also pursuing his MS in Computer Science and Engineering from same university. He completed his BS in Computer Science and Engineering from MIST, Bangladesh in 2009. He has worked to investigate the area of supplier selection and has established a new approach using fuzzy theory that outperforms existing state of art approaches. His current research interest includes various load balancing, load migration, and server consolidation techniques in cloud computing. He is also interested in various approaches in machine learning, data mining and fuzzy systems. Many of his works have published in well reported journals and conference proceedings.

Rashedur M. Rahman is working as an Associate Professor in Electrical and Computer Engineering (ECE) department in North South University, Dhaka, Bangladesh. He received his Ph.D. in Computer Science from University of Calgary, Canada and Masters from University of Manitoba, Canada in 2007 and 2003 respectively. He has authored more than 60 peer reviewed journals and conference proceedings in the area of parallel, distributed, grid and cloud computing, knowledge and data engineering. His current research interest is in data mining particularly on financial, medical and educational data, data replication on grid, cloud load characterization, optimization of cloud resource placements and computational finance. He has been serving in the editorial board of a number of journals in the knowledge and data engineering filed. He also served as reviewer of couple of journals published by Elsevier, Springer and Wiley. He also serves as an organizing committee member of different international conferences organized by IEEE and ACM.

Saad Abdullah received his degree in Computer Science and Engineering from North South University in 2011. He graduated with the distinction of Summa Cum Laude in recognition of being the highest grade point average holder in his program. He is currently pursuing his Masters degree in Computer Science and Engineering from the Department of ECE in North South University. His areas of interest in research are machine learning, fuzzy systems and data mining. He has served as a Laboratory Instructor at the Dept of ECE.