Two Level Fuzzy Approach For Dynamic Load Balancing In The Cloud Computing

Mir Mohammad Alipour, Mohammad Reza Feizi Derakhshi
University of Tabriz
Iran
mohammad.alipour@tabrizu.ac.ir
mfeizi@tabrizu.ac.ir

ABSTRACT: *The concept of the Cloud computing has significantly changed the field of parallel and distributed computing systems today. Cloud computing enables a wide range of users to access distributed, scalable, virtualized hardware and/or software infrastructure over the Internet. Load balancing is the process of distributing the load among various nodes of a distributed system to improve resource utilization, minimum response time and maximize throughput while also avoiding a situation where some of the nodes are heavily loaded while other nodes are idle or doing very little work. Load balancing ensures that all the processor in the system or every node in the network does approximately the equal amount of work at any instant of time.*

In this paper, we propose a new two level fuzzy approach for dynamic load balancing in cloud computing. This approach characterizes the uncertainty in a distributed system by using the fuzzy logic. The processor speed and queue length of nodes are used to balance the load of the nodes in cluster level and processing power and average queue length of clusters are used to balance the load of the clusters in cloud level through fuzzy logic. In comparative study, proposed algorithm shows better average of response time and throughput than Round Robin and Randomize algorithms.

Keywords: Distributed systems, Cloud Computing, Load Balancing, Fuzzy logic.

Received: 2 October 2016, Revised 1 November 2016, Accepted 9 November 2016

1. Introduction

Cloud computing [1,2] is an emerging commercial infrastructure and internet based cost-efficient computing paradigm in which information can be accessed from a web browser by customers for their requirement. It is a modality of computing characterized by on-demand availability of resources in a dynamic and scalable fashion, where the resource is used to represent infrastructure, platforms, software, services, or storage. It promises to eliminate the need for maintaining expensive computing facilities by companies and institutes alike. Cloud computing in a generally term is defined for anything that involves delivering hosted services over the Internet. It is based on the concept of shared computational, storage, network and application resources provided by a third party.

Sharing resources result in better efficiency to cost ratio, less response time, higher throughput, more reliability, extensibility and double development and better flexibility in observation. The increasing need of computing capacity has caused the computing environment to change from centralized to distributed state. Hence, distributed systems play a major role in computing requirements.

Great interest in cloud computing has been manifested from both academia and private research centers and numerous projects from industry and academia have been proposed. In commercial contexts among the others, we highlight: amazon elastic compute cloud, IBM's blue cloud and etc. There are several scientific activities driving toward open cloud computing middleware and infrastructures such as reservoir, eucalyptus and etc [3].

In a typical distributed system setting, tasks arrive at the different nodes in a random fashion. This causes a situation of non-uniform loading across the system nodes to occur. Loading imbalance is observed by the existence of nodes that are highly loaded while others are lightly loaded or even idle. Thus, resource management or load balancing is major challenging issue in cloud computing. Load balancing is a methodology to distribute workload across multiple computers, or other resources over the network links to achieve optimal resource utilization, maximize throughput, minimum response time, and avoid overload [4].

The objective can be achieved by two ways: either by avoid sending more tasks to the overloaded nodes and keep checking the nodes' states periodically, or taking a migration decision to migrate some tasks from the overloaded nodes to the lightly loaded nodes for the sake of improving the system performance as a whole.

The load balancing problem in a distributed computing environment has been addressed extensively by many studies and researches. As a result, various algorithms have been introduced to solve the load balancing problem. All kinds of these algorithms fall into one of the following two categories: dynamic or static [5,6].

Dynamic load balancing algorithms use the information describing the state of the system in order to improve the accuracy of load balancing decision. Therefore, the dynamic algorithms are appropriate to be utilized in practical situations. However, this kind of algorithms should have the ability to collect the information about the current state of the system, to store the collected information and to analyze them. Therefore, the dynamic algorithms may cause additional overhead computations for the system. On the other side, the static algorithms don't exploit such information. In static algorithms, the decisions are taken based on a priori knowledge of the system. Thus, the static algorithms don't have the ability to deal with the dynamic changes of such environments.

The precise and fast load balancing scheme is an essential factor in increasing the efficiency and performance of cloud computing components. However, many factors lead to ambiguous information about the clusters/nodes states, i.e. lack of shared memory among independent clusters. This ambiguity leads to uncertainty in decision for load balancing. Another important aspect is that the states of the clusters/nodes can be changed rapidly. Therefore there is always some amount of uncertainty in the state information used for making a decision. Hence it is necessary that the decision making process deals with these uncertainties. Fuzzy logic is one of the methods that deal with the uncertain information [7-11].

In this paper, we utilize a two level Fuzzy Inference System based approach to model uncertainty factors to determine the nodes states and thereby the clusters states. In this paper we use two parameters as input variables for the sake of more accurate fuzzy-logic based approach. We consider a continuous stream of independent jobs which arrive to the system and are stored into a single job queue. We assume that all jobs are not preemptable, and each job is described by its ID, submission time and an estimation of its execution time. The cloud is composed of many independent clusters of heterogeneous processing machines.

The rest of the paper is organized as follows. Related works is reported in Section 2. In Section 3 the fuzzy logic are discussed. Section 4 presents the system model. Then, the proposed two level fuzzy logic approach for load balancing in cloud computing is introduced in Section 5. The experimental results is reported in Section 6. Finally a conclusion is drawn in Section 7.

2. Related works

Z. Chaczko et. al [12] proposed a Token Routing dynamic load balancing algorithm. The main objective of this algorithm to minimize the system cost by moving the tokens around the system. But in a scalable cloud system agents could not have the enough information of distributing the work load due to communication bottleneck. So the workload distribution among the

agents is not fixed. The drawback of the token routing algorithm can be removed with the help of heuristic approach of token based load balancing.

Z. Xu et. al [13] proposed a static Round Robin algorithm. In this approach, the processes are divided between all processors. Each process is assigned to the processor in a round robin order. The process allocation order is maintained locally independent of the allocations from remote processors. Though the work load distributions between processors are equal but the job processing time for different processes are not same. So at any point of time some nodes may be heavily loaded and others remain idle. Also, they proposed a static Randomized algorithm [13]. In this algorithm a process can be handled by a particular node n with a probability p. The process allocation order is maintained for each processor independent of allocation from remote processor. This algorithm works well in case of processes are of equal loaded. However, problem arises when loads are of different computational complexities.

M. Beltran et. al [14] presented a Central queuing algorithm, which works on the principal of dynamic distribution. Each new activity arriving at the queue manager is inserted into the queue. When request for an activity is received by the queue manager it removes the first activity from the queue and sends it to the requester. If no ready activity is present in the queue the request is buffered, until a new activity is available. But in case new activity comes to the queue while there are unanswered requests in the queue the first such request is removed from the queue and new activity is assigned to it. When a processor load falls under the threshold then the local load manager sends a request for the new activity to the central load manager. The central manager then answers the request if ready activity is found otherwise queues the request until new activity arrives. A Connection mechanism Load balancing algorithm is presented in [15]. It can also be based on least connection mechanism which is a part of dynamic scheduling algorithm. It needs to count the number of connections for each server dynamically to estimate the load. The load balancer records the connection number of each server. The number of connection increases when a new connection is dispatched to it, and decreases the number when connection finishes or timeout happens.

G. E. Gonçalves et. al [16], proposed algorithms for allocation of computing and network resources in a Distributed cloud with the objectives of balancing the load in the virtualized infrastructure and of considering constraints, such as processing power, memory, storage, and network delay. The evaluation of the algorithm shows that it is indeed adequate for link allocation across different physical networks.

S. M. Nejad et al. [17], proposed a fuzzy logic-base load balancing in centralized distributed system. The efficiency of proposed load balancing is studied in OPNET simulation environment. The lengths of sent packet and service rate in each node have been considered non- constant. The current load of the system and waiting time for the last processed task in each node are considered as the fuzzy controller inputs. Based on the weight assigned for each node, a percentage of existing tasks is assigned to that node. The result of the experiments in the states of constant and variable nodes, and constant and variable tasks, indicates that this algorithm performs much better than both static and dynamic algorithms in terms of throughput, drop rate and response time.

S. Sethi et. al [18], introduced the novel load balancing algorithm using fuzzy logic in cloud computing, in which load balancing is a core and challenging issue in Cloud Computing. The processor speed and assigned load of Virtual Machine (VM) are used to balance the load in cloud computing through fuzzy logic. It is based on Round Robin (RR) load balancing technique to obtain measurable improvements in resource utilization and availability of cloud computing environment.

S. Wang et.al [19], introduced a two-phase scheduling algorithm under a three level cloud computing network. The proposed scheduling algorithm combines OLB (Opportunistic Load Balancing) and LBMM (Load Balance Min-Min) scheduling algorithms that can utilize more better executing efficiency and maintain the load balancing of system.

3. Fuzzy Logic

Fuzzy logic is a method based on multi-valued logic which aims to formalize approximate reasoning [20]. It is used to deal with different types of uncertainty in knowledge based systems. Some of the relevant characteristics of fuzzy logic are fuzzy sets, linguistic variables and fuzzy rules. A fuzzy set is a collection of objects characterized by a membership function with a continuous grade of membership which can be ranged between zero and one [21]. A linguistic variable is a type of variable that uses words instead of numbers to represent its values (e.g. slow, medium and fast) [20]. The values used to define linguistic variables are called terms and the collection of terms is called term set.

The fuzzifier is the input interface which maps a numeric input to a fuzzy set so that it can be matched with the premises of the fuzzy rules defined in the application-specific rule base. The rule base contains a set of fuzzy if-then rules which defines the actions of the controller in terms of linguistic variables and membership functions of linguistic terms. The fuzzy inference engine applies the inference mechanism to the set of rules in the fuzzy rule base to produce a fuzzy set output. This involves matching the input fuzzy set with the premises of the rules, activation of the rules to deduce the conclusion of each rule that is fired, and combination of all activated conclusions using fuzzy set union to generate fuzzy set output. The defuzzifier is an output mapping which converts fuzzy set output to a crisp output. Based on the crisp output, the fuzzy logic controller can drive the system under control.

4. System Model

Here we define a hierarchical structure for describing a network of distributed system, several nodes in which every node may be a complex combination of multiple types of resources (CPU's, memory, disks, switches and etc) and the physical configurations of resources for each node may be heterogeneous, form a clusters and each cluster has a cluster load manager (CRLM). Different physical clusters make a cloud with a cloud load manager (CDLM). CDLM communicates with CRLMs and CRLMs communicate with nodes as shown in the Fig. 1. A Cloud has different number of heterogeneous clusters and each cluster has different number of heterogeneous nodes.

Each CRLM only communicate with its own nodes and CDLM. Also, each node only communicates with its CRLM and none of the cluster's nodes don't communicate with each other.

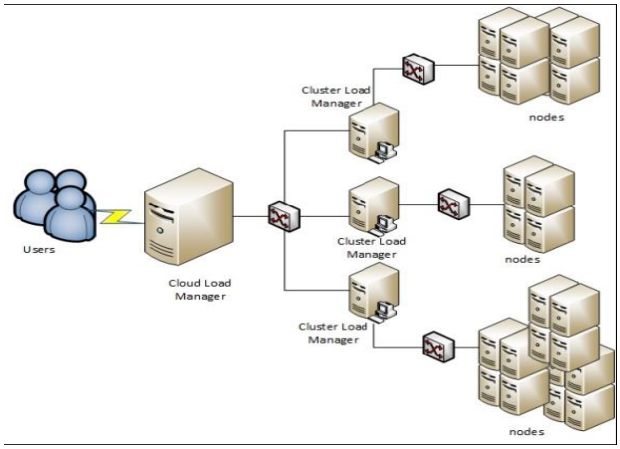


Figure 1. Hierarchical structure of distributed system in Cloud computing

5. Two level fuzzy approach for load balancing

In this section we describe the proposed two level fuzzy approach for load balancing in cloud in order to achieve better response time and throughput. We implement this load balancing technique based on Fuzzy logic. Where the fuzzy logic is natural like language through which one can formulate their problem.

The advantages of fuzzy logic are easy to understand, flexible, tolerant of imprecise data and can model nonlinear functions of arbitrary complexity, and is used to approximate functions and can be used to model any continuous function or system [22,23]. Fuzzy inference is the process of formulating the mapping from a given input to an output using fuzzy logic and the mapping provides a basis from which decisions can be made, or patterns recognized. The load balancing in cloud is done in two level,

level 1: selecting appropriate cluster of the cloud to send the job based on cluster power and average queue length of the cluster using fuzzy logic.

level 2: selecting appropriate node of cluster to assign the job based on CPU speed and queue length of the node using fuzzy logic.

Therefore, Cloud maintains the important information about each cluster like cluster power and average queue length of the cluster, which receives from clusters periodically. And clusters maintain the important information about each node of the cluster like CPU power and queue length of the node, which receives from nodes periodically.

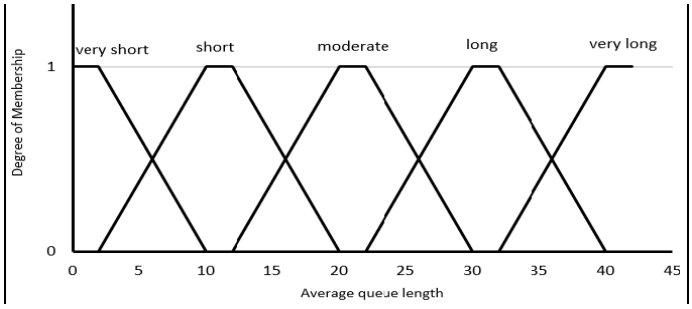
As soon as cloud receives a job to allocate, CDLM evaluates the states of the clusters belonging to its cloud, then identifies the least loaded cluster and sends the job to it. This process is done for cluster level load balancing, therefore, a newly arrived job to the cloud will be sent to the lightly loaded cluster. Then CRLM of the recipient cluster evaluates the states of the nodes belonging to its cluster, then assigns the job to the least loaded node of it.

This process is performed for node level load balancing, therefore, a newly arrived job to the cluster will be assigned to the lightly loaded node. Assuming that, the job allocated to each cluster/node will be processed in that cluster/node and will not be relocated.

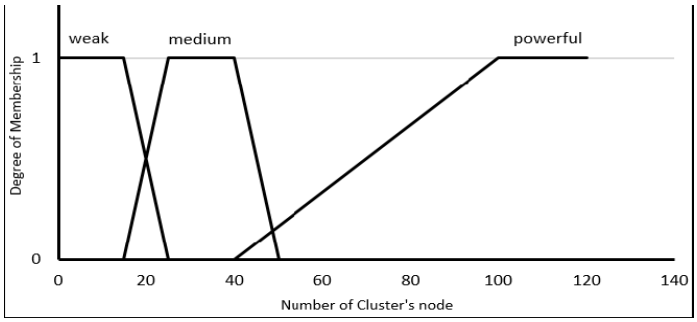
Current state of the cloud, clusters and nodes, is updated without any request. Then, based on the data received from the nodes, the fuzzy CRLM evaluates the workload of the nodes and based on the data received from CRLM, the fuzzy CDLM calculates the workload of each cluster. Then, the CDLM forwards the arriving jobs from user to the appropriate cluster. Then, the fuzzy CRLM, based on the inferred states of different nodes, ranks the nodes based on their workload. Then, the CRLM assign the arriving jobs from CDLM to the appropriate node.

5.1 Fuzzy Inference for Job Allocation in CDLM (level 1)

Fuzzy CDLM needs data about each cluster to determine the state of that cluster include cluster power and average queue length to cloud level (level 1) local balancing. Cluster power of each cluster is fixed and once a cloud is created, preserved. And the cloud collects periodically data about each cluster's average queue length which is the average of the cluster's nodes queue length. This information is then passed to the cloud Fuzzy Inference System (FIS) that performs the needed analysis and takes a decision about the clusters state.



(a) Average-queue-length



(b) Cluster-power

Figure 2. Two inputs membership functions of fuzz CDLM

Fuzzy CDLM algorithm for job allocation adopts node average queue length and cluster power as the input variables for fuzzy sets and define a set of membership function. The two membership functions, average-queue-length membership functions and cluster-power membership functions for the inputs that were used for the fuzzy based algorithm are shown below in Fig. 2.

The x-axis corresponds to the cluster's attributes and the y-axis gives the degree of membership (0 indicates no membership, 1 indicates total membership). For the average-queue-length membership function, the length of queue has five member functions: very-short, short, moderate, long and very-long.

For the Cluster-power membership functions, number of cluster's node are define the three member functions: weak, medium, and powerful.

The output (load of the cluster) which is called 'cluster-load' and has seven membership functions (as shown in Fig. 3): very-very-low (VVL), very-low (VL), low (L), moderate (M), high (H), very-high (VH) and very-very-high (VVH).

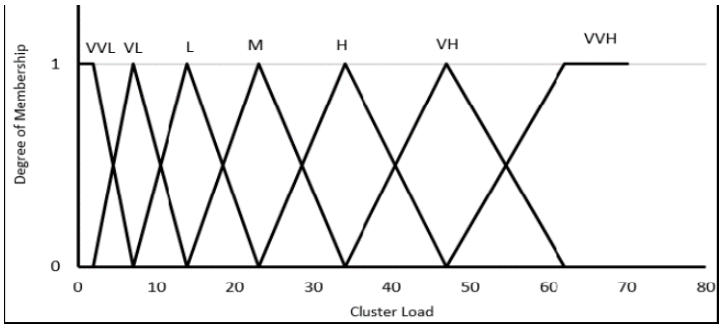


Figure 3. Fuzzy functions for the load of cluster (output)

Using the rules as shown in Fig. 4, we determine which output(s) are chosen and the degree of membership to these output functions determines the value of choice happen to any node.

Fuzzy output is converted to a final action values through the defuzzifier. For this part the COA (Center of Area) method is applied as described in Equation (1).

$$f(x) = \frac{\sum_{l=1}^{15} \bar{y}^{l} \left(\prod_{i=1}^{2} \mu_{A_i^l}(x_i)\right)}{\sum_{l=1}^{15} \left(\prod_{i=1}^{2} \mu_{A_i^l}(x_i)\right)} \quad (1)$$

Where $\mu_{Ai}{}^{l}(x_i)$ is membership degree of the ith variable of lth inference rule, and y^{-l} is a center value of lth inference rule output. The node with the minimum output value is chosen as the best cluster which receives the next incoming job in the cloud.

5.2 Fuzzy Inference for Job Allocation in CRLM (level 2)

Fuzzy CRLM needs data about each node to determine the state of that node include CPU speed and queue length to cluster level (level 2) local balancing. CPU speed of each node is fixed and once a cluster is created, preserved. Also, cluster collects periodically data about each node's queue length from the table contains the information about current jobs of node. This information is then passed to the cluster FIS that performs the needed analysis and takes a decision about the node's state.

Fuzzy CRLM algorithm for job allocation adopts node queue-length and CPU-power as the input variables for fuzzy sets and define a set of membership function. The two membership functions, queue-length membership functions and CPU-power membership functions for the inputs that were used for the fuzzy based algorithm are shown below in Fig. 5. The x-axis corresponds to the node's attributes and the y-axis gives the degree of membership. For the queue-length membership function the length of queue has five member functions: very-short, short, moderate, long and very-long.

Rule 1: IF Cluster-power is weak AND queue-length is very-short Then cluster-load is very-very-low.

Rule 2: IF Cluster-power is weak AND average-queue-length is short Then cluster -load is moderate.

Rule 3: IF Cluster-power is weak AND average-queue-length is moderate Then cluster -load is high.

Rule 4: IF Cluster-power is weak AND average-queue-length is long Then cluster -load is very-high.

Rule 5: IF Cluster-power is weak AND average-queue-length is very-long Then cluster -load is very-very-high.

Rule 6: IF Cluster-power is medium AND average-queue-length is very-short Then cluster -load is very-very-low.

Rule 7: IF Cluster-power is medium AND average-queue-length is short Then cluster -load is low.

Rule 8: IF Cluster-power is medium AND average-queue-length is moderate Then cluster -load is moderate.

Rule 9: IF Cluster-power is medium AND average-queue-length is long Then cluster -load is high.

Rule 10: IF Cluster-power is medium AND average-queue-length is very-long Then cluster -load is very-very-high.

Rule 11: IF Cluster-power is powerful AND average-queue-length is very-short Then cluster -load is very-very-low.

Rule 12: IF Cluster-power is powerful AND average-queue-length is short Then cluster -load is very-low.

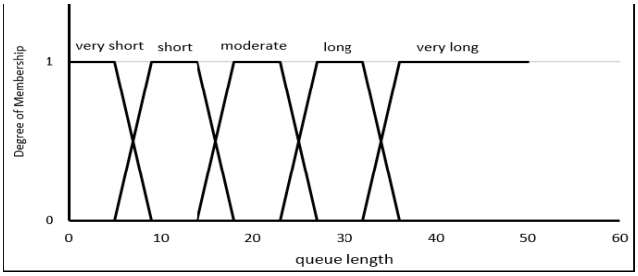
Rule 13: IF Cluster-power is powerful AND average-queue-length is moderate Then cluster -load is low.

Rule 14: IF Cluster-power is powerful AND average-queue-length is long Then cluster -load is moderate.

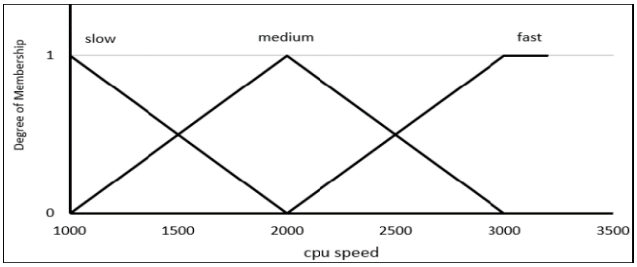
Rule 15: IF Cluster-power is powerful AND queue-length is very- long Then cluster -load is very-very-high.

Figure 4. Rule base of the fuzzy inference system for CDLM

For the CPU-speed membership functions the processing speed of CPU's are define the three member functions: slow, medium, and fast.



(a) Queue-length



(b) CPU-speed (Ghz)

Figure 5. Two inputs membership functions of fuzz CRLM

The output (load of the node) which is called 'node-load' and has five membership functions: very-low, low, moderate, high and very-high (as shown in Fig. 6).

Using the rules as shown in Fig. 7, we determine which output(s) are chosen and the degree of membership to these output functions determines the value of choice happen to any node.

Fuzzy output is converted to a final action values through the defuzzifier. For this part the COA (Center of Area) method is applied as described in Equation (1).

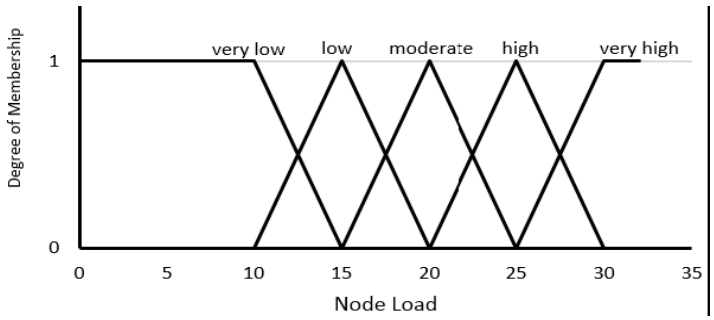


Figure 6. Fuzzy functions for the load of node (output)

Rule 1: IF CPU-speed is low AND queue-length is very-short Then node-load is very-low.
Rule 2: IF CPU-speed is low AND queue-length is short Then node-load is low.
Rule 3: IF CPU-speed is low AND queue-length is moderate Then node-load is high.
Rule 4: IF CPU-speed is low AND queue-length is long Then node-load is very-high.
Rule 5: IF CPU-speed is low AND queue-length is very-long Then node-load is very-high.
Rule 6: IF CPU-speed is medium AND queue-length is very-short Then node-load is very-low.
Rule 7: IF CPU-speed is medium AND queue-length is short Then node-load is low.
Rule 8: IF CPU-speed is medium AND queue-length is moderate Then node-load is moderate.
Rule 9: IF CPU-speed is medium AND queue-length is long Then node-load is very-high.
Rule 10:IF CPU-speed is medium AND queue-length is very-long Then node-load is very-high.
Rule 11: IF CPU-speed is fast AND queue-length is very-short Then node-load is very-low.
Rule 12: IF CPU-speed is fast AND queue-length is short Then node-load is very-low.
Rule 13: IF CPU-speed is fast AND queue-length is moderate Then node-load is moderate.
Rule 14: IF CPU-speed is fast AND queue-length is long Then node-load is high.
Rule 15: IF CPU-speed is fast AND queue-length is very- long Then node-load is very-high.

Figure 7. knowledge base of the fuzzy inference system for CRLM

6. Experimental Results

In the simulation, we considered that a cloud composed of five independent clusters of heterogeneous nodes. Also, we have assumed static number of clusters and nodes, respectively in the cloud and cluster as shown in Table 1.

In the simulations, about 5000 jobs was generated and used. Jobs are generated continuously and independently and are stored into a cloud's single job queue. All the generated jobs are not removable, and each jobs is described by three attributes: the job ID, job's generation time and estimation of its execution time, which has a normal distribution, N(60 ; 30).

We compare the efficiency of the proposed two level fuzzy algorithm (Fuzzy) with two algorithms of Randomize and Round Robin. Compares are done based on two criteria of average response time and throughput.

Fig. 8 shows the assignment of jobs among the different nodes of cluster 3. Round Robin algorithm distributes workload among the nodes equally without considering their CPU speed and Similarly Randomize algorithm distributes workload among the nodes pure randomly. But it is obvious form the Figure that the workload was not distributed in a fair manner by Fuzzy algorithm where some nodes have workload more than other. For instance, the nodes 1 and 3 have the heaviest workload, whereas the node 2 has much less workload. This is because, Fuzzy algorithm considers CPU speed and consequently queue length of nodes to distribute jobs among them.

Cluster No.	Node's Number	CPU Speed (GHz) of the Node
1	1	6000
1	2	3000
1	3	6000
1	4	5400
1	5	4600
2	1	7500
2	2	6000
3	1	1800
3	2	2500
3	3	1500
3	4	1000
3	5	2000
3	6	3000
3	7	2000
3	8	2000
3	9	2500
3	10	3000
4	1	7000
4	2	1000
4	3	2500
4	4	4000
4	5	4600
4	6	5500
4	7	5500
5	1	4500
5	2	1500
5	3	3000
5	4	3000

Table 1. Cloud's configuration

Fig. 9 shows the assignment of the jobs among the different nodes of each cluster in the cloud. In each cluster, Round Robin algorithm distributes workload among the nodes of a cluster equally and Similarly Randomize algorithm distributes workload among the nodes of a cluster pure randomly. For instance, using Round Robin algorithm, each node of cluster 1, receives 1/5 workload of cluster 1, because cluster 1 has 5 nodes and each node of cluster 2, receives 1/2 workload of cluster 2, because cluster 2 has 2 nodes and etc. But Fuzzy algorithm doesn't distribute the workload equally among the nodes of a cluster, where some nodes have workload more than other.

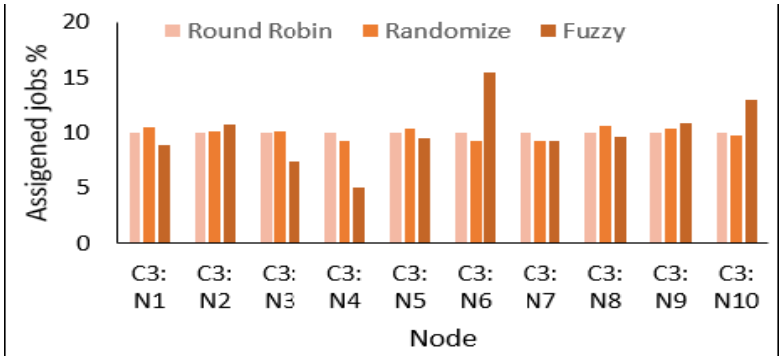


Figure 8. Distribution of jobs among the nodes of cluster 3

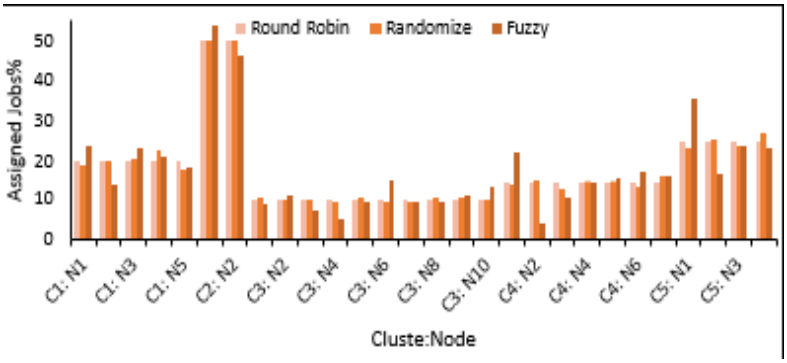


Figure 9. Distribution of jobs among the different nodes of each cluster in the cloud

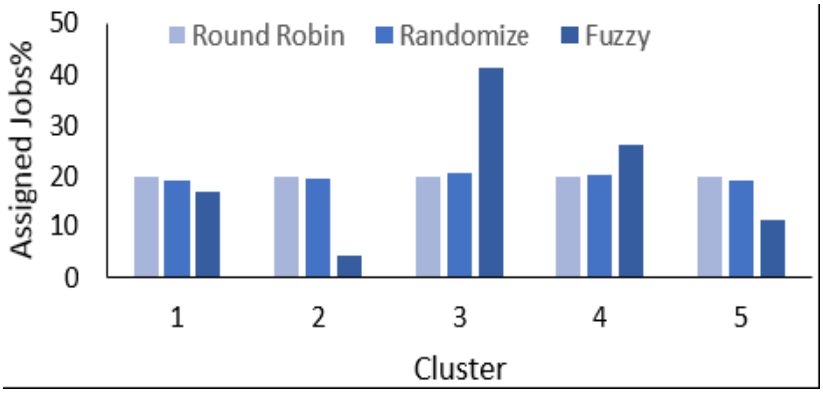
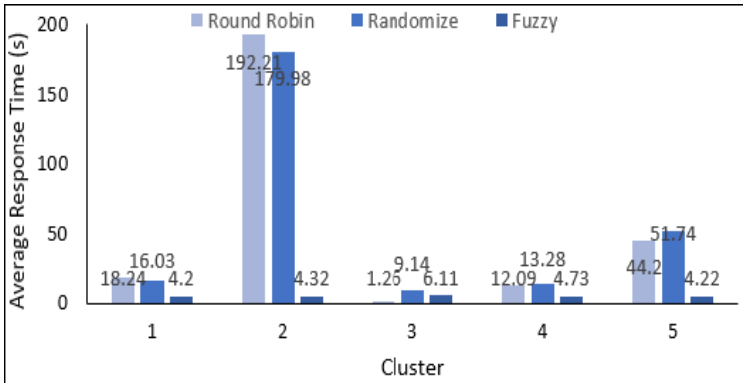


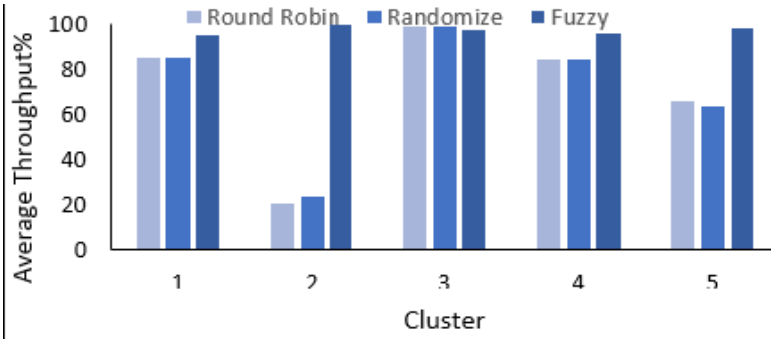
Figure 10. Distribution of jobs among the clusters of the cloud

Fig. 10 shows the assignment of the jobs among the different clusters of the cloud. Round Robin algorithm distributes workload among the 5 clusters equally without considering their processing power and Similarly Randomize algorithm distributes workload among the cluster pure randomly. But the Fuzzy algorithm distributes the workload among the clusters unequaled, where some clusters have workload more than other. For instance, the cluster 3 has the heaviest workload, whereas the cluster 2 has much less workload. This is because, Fuzzy algorithm considers cluster power and consequently average queue length of cluster to distribute jobs among them.

Fig. 11.a demonstrates the average response time of the different clusters. As it can be seen, fuzzy algorithm performs better than the other algorithms. In other words, fuzzy algorithm has less average response time than the other algorithms. Because Fuzzy algorithm considers processing power of the cluster and distributes the workload among them unequaled. This strategy prevents heavy loading, weak clusters and light loading, powerful clusters and consequently queue length of clusters are balanced in this manner and results the less response time. However, Round Robin and Randomize algorithms dispatch workload between clusters equally or near equally, without considering their processing power and queue length, which make weak clusters are heavily loaded and powerful clusters are lightly loaded. This type of job distribution, results imbalanced processing queue in the clusters and nodes and directly increases response time of the nodes and clusters.



(a) Average Response Time



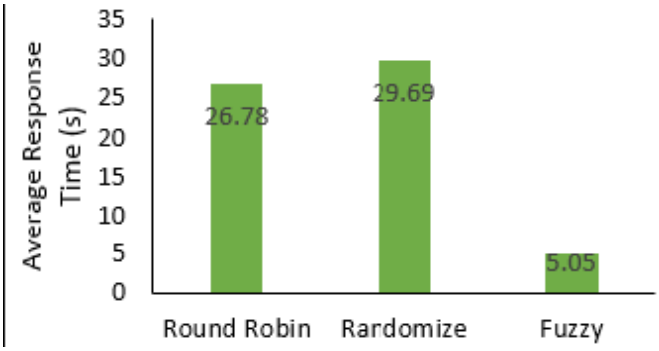
b) Average Throughput

Figure 11. Clusters level Achievements

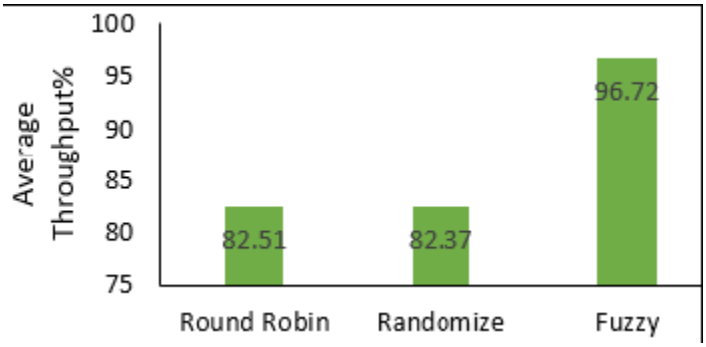
Fig. 11.b demonstrates the average throughput of the different clusters. The results indicate that, fuzzy algorithm performs better than the other two algorithms and has high throughput than the other algorithms. Considers processing power and average queue length of the clusters to distributing the workload among them make the fuzzy algorithm outperforms other two algorithm as argued for response time.

Balance load in nodes level and consequently in cluster level results in less average response time and high average throughput

in cloud level as shown in Fig. 12. These results indicate that fuzzy algorithm has more efficiency compared to Round Robin and Randomize algorithms.



(a) Average Response Time



(b) Average Throughput

Figure 12. Cloud level achievements

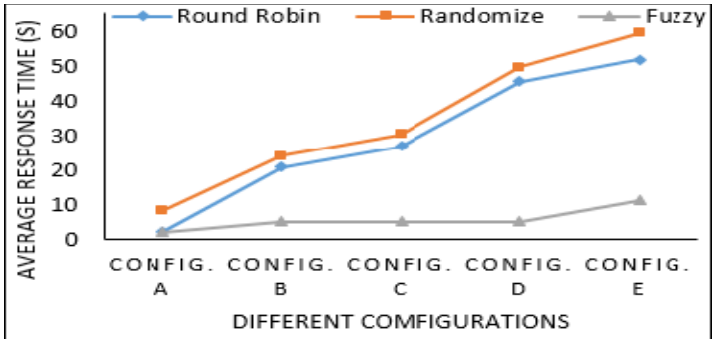
Finally, we used five different configurations for the cloud as shown in TABLE 2. Where, the first column indicates the configuration no, the second column shows the clusters number in the cloud, the third column shows the number of nodes in each cluster of the cloud respectively, the columns 'Mean CPU speed' and 'CPU speed deviation' show the parameters of normal distribution in which, CPU speed of the nodes in cloud are considered, N(Mean CPU speed; CPU speed deviation).

Config.	Clusters	Nodes	Mean CPU speed	CPU speed deviation
Config. 1	1	28	4000	0
Config. 2	1	28	4000	3000
Config. 3	2	12,16	4000	3000
Config. 4	3	5,20,3	4000	3000
Config. 5	5	5,18,2,1,2	4000	3000

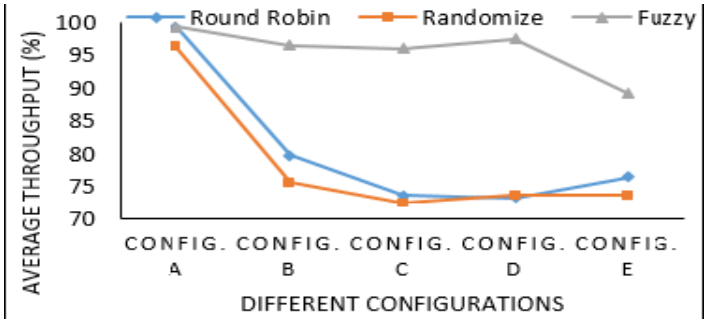
Table 2. Five different configurations of the cloud

Fig. 13 shows as cloud's degree of variance increases and its configuration goes complicated, Round Robin and Randomize algorithms can't adapt themselves with this situation and workload are distributed inefficiently, which makes increasing average

response time and decreasing average throughput of them. However, Fuzzy algorithm adapts itself with different configurations and balances workload among clusters and nodes more efficiently and shows about consistent results despite variant cloud configurations.



(a) Average Response Time



(b) Average Throughput

Figure 13. Cloud level achievements using different configurations

7. Conclusion

In the cloud computing, load balancing is quite a challenging task in terms of jobs allocation to clusters and nodes. In this paper, we presented a two level fuzzy approach for dynamic load balancing in the cloud computing. Fuzzy logic systems can deal with the uncertainties around the input variables and their relationships and make absolute outputs.

The Processing power and average queue length of the cluster is considered as the fuzzy controller inputs in level 1 and the CPU speed and queue length of the node is considered as the fuzzy controller inputs in level 2. Obtained results show that proposed Load Balancing algorithm achieves satisfactory and consistently load balancing than Round Robin and Randomize algorithm and gives better response time and throughput.

References

[1] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., Brandic, I. Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility, Future Generation Computer Systems, Volume 25, Number 6, Pages: 599-616, ISSN: 0167-739X, Elsevier Science, Amsterdam, The Netherlands, June 2009.

[2] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. F., Buyya, R. CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms, Software: Practice and Experience, 41 (1) 23–50, January 2011.

[3] Parhizkar, B., Naser, A., Abdulhussein, A., Joshi, J. H. A (2013). Common Factors Analysis on cloud computing models, 10 (2) *IJCSI International Journal of Computer Science Issues,* 10 (2) No 1, March 2013.

[4] Sun, C. L. (2000). A fuzzy approach to load balancing in a distributed object computing network, *In:* Proc. Of 6th Int IEEE Conf. HPDC.

[5] Sharma, S., Singh, S., Sharma, M. (2008). Performance Analysis of Load Balancing Algorithms, World Academy of Science, Engineering and Technology, vol. 38

[6] Khanli1, L. M. Didevar, B. A New Hybrid Load Balancing Algorithm in Grid Computing Systems, *Journal of Computer Science* 2(5) October, 2011.

[7] Zadeh, L. A. (2002). From Computing with numbers to computing with words-from manipulation of measurements to manipulation of perceptions, *Int. J. Appl. Math. Computer Sci*., 12 (3) 307-324

[8] Nguyen, H. T., Prasad, N. R., Walker, C. L., Walker, E. A. (2002). A First Course in Fuzzy and Neural Control, CRC Press.

[9] Cheung, L., Kwok, Y. (2004). On load balancing approaches for distributed object computing systems, *The Journal of Supercomputing,* vol. 27, p. 149-175

[10] Lu, K., Subrata, R., Zomaya, A.Y. (2006). An Efficient Load Balancing Algorithm for Heterogeneous Grid Systems Considering Desirability of Grid Sites, *In*: Proc. 25th IEEE Int'l Performance Computing and Comm. Conf. (IPCCC '06)

[11] Y. Li, Yuhang Yang and R. Zhu, A Hybrid Load balancing Strategy of Sequential Tasks for Computational Grids, *In*: IEEE International Conference on Networking and Digital Society, p.112-117, 2009.

[12] Chaczko, Z., Mahadevan, V. , Aslanzadeh, S., Mcdermid, C. (2011). Availability and Load Balancing in Cloud Computing, International Conference on Computer and Software Modeling IPCSIT V.14 IACSIT Press, Singapore.

[13] Xu, Z., Huang, R. (2009). Performance Study of Load Balancing Algorithms in Distributed Web Server Systems, CS213 Parallel and Distributed Processing Project Report.

[14] Beltran, M., Guzman, A., Bosque, J.L. (2011). Dealing with heterogenity in clusters, *In*: Proceeding of the Fifth International Symposium on Parallel and Distributed Computing, ISPDC.

[15] Warstein, P., Situ, H., Huang, Z. (2010). Load balancing in a cluster computer, *In*: Proceeding of the seventh International Conference on Parallel and Distributed Computing, Applications and Technologies, IEEE.

[16] Gonçalves, G. E., Endo, P. T., Palhares, A. A., Santos, M. A., Kelner, J., Sadok, D. (2013). On the load balancing of virtual networks in distributed clouds. *In*: Proceedings of the 28th Annual ACM Symposium on Applied Computing – SAC '13, 625.

[17] Moosavi Nejad, S. R., Mortazavi, S.S., Vahdat, B. V. (2011). Fuzzy based Design and tuning of distributed systems load balancing controller, *In*: The 5th Symposium on Advances in Science and Technology (SASTech),

[18] Sethi, S., Sahu, A., Kumar, S. Efficient load Balancing in Cloud Computing using Fuzzy Logic, 2 (7) 65–71, *IOSR Journal of Engineering (IOSRJEN)* (7) PP 65-71, July 2012.

[19] Wang, S., Yan k Liao W Wang S. (2010). Towards a Load Balancing in a Three-level Cloud Computing Network, *In*: Computer Science and Information Technology (ICCSIT), 3rd IEEE International Conference on (V. 1)

[20] Zadeh, L. A. (1994). The role of fuzzy logic in modeling, identification and control, Modeling, *Identification and Control* (MIC), V 15, p.191-203.

[21] Zadeh, L. A. (1965). Fuzzy sets, Information and Control, V 8, p. 338-353.

[22] Wong, Y. F., Wong, W. C. (2002). A fuzzy-decision-based routing protocol for mobile ad hoc networks, *In*: *10th IEEE International Conference on Network,* p 317–322

[23] Raju, G. V. S., Hernandez, G., Zou, Q. (2000). Quality of service routing in ad hoc networks, *IEEE Wireless Communications and Networking Conference,* Vol. 1, p. 263 –265