Indian Journal of Fundamental and Applied Life Sciences ISSN: 2231– 6345 (Online)
An Open Access, Online International Journal Available at www.cibtech.org/sp.ed/jls/2015/02/jls.htm
2015 Vol. 5 (S2), pp. 1978-1990/Fatemimoghadam and Azamrabiei
Review Article

FUZZY CONTROLLER FOR ELASTICITY PROBLEM IN CLOUD COMPUTING

Mohammad Reza Fatemimoghadam[1] and [*]Azamrabiei[2]
[1]*Qeshm international Branch, Islamic Azad University*
[2]*Islamic Azad University of Dolatabad*
**Author for Correspondence*

ABSTRACT
Clouds provide an illusion of an infinite amount of resources and enable elastic services and applications that are capable to scale up and down (grow and shrink by requesting and releasing resources) in response to changes in its environment, workload, and Quality of Service (QoS) requirements. Elasticity allows achieving required QoS at a minimal cost in a Cloud environment with its pay-as-you-go pricing model. In this paper, we present our experience in designing a fuzzy elastically controller for a key-value store. The goal of our research is to investigate the feasibility of the control theoretic approach to the automation of elasticity of Cloud-based key-value stores. We describe design steps necessary to build a fuzzy controller for a real system, namely Voldemort, which we use as a case study in this work. The design steps include defining touch points (sensors and actuators), system identification, and controller design. We have designed, developed, and implemented a prototype of the fuzzy elasticity controller for Voldemort. Our initial evaluation results show the feasibility of using fuzzy control to automate elasticity of distributed key value stores.

Keywords: *Cloud Computing; Elasticity; Fuzzy Control; Key-Value Store; Voldemort*

INTRODUCTION
Advantages of Cloud computing over other large scale computing alternatives. The first one is that the end-user does not need to be involved in the configuration and maintenance of the Cloud. A developer does not have to buy servers, security solutions, etc, and set them up; rather, the developer builds applications on Cloud resources (Rastogi, 2010) of a Cloud provider. The second advantage and probably the most important one, is that the end-user only pays for resources the developer requests and uses. That is why the Cloud computing approach is less expensive than its alternatives. However, this property, called "pay-as-you-go" has an important drawback. If allocated resources exceed the required amount, it will incur a waste of money. On the other hand, if they obtained resources are not adequate, the system might not meet the Service Level Objectives (SLOs), resulting in a negative impact on its performance and , as a consequence, negative impact on user experience with the system. Considering these issues, the concept of elasticity comes to the attention of many Cloud users. There is a difference between elasticity and scalability. In the context of this work, scalability means the expansion capacity of the system, which is expected to be reached in the future. This approach has a very important drawback, which is that the ultimate size of the system should be specified in advance. There is a contradiction between this concept and the "pay-as-you-go" property of Cloud computing, because in this approach the end-user should pay for the ultimate size of the system which might never be fully utilized. Another disadvantage is, as scaling up means adding more physical resources; it might be not easy to scale down by removing them. To deal with these problems, a new approach, called Elasticity, has become favorite in recent years. In this approach, the final size of the system is not predefined. But an elastic system is capable of scaling up and down (growing and shrinking by requesting and releasing resources) at runtime in response to changes in its environment, workload, and Quality of Service (QoS) requirements. In the case of increasing the load, a new instance is added to meet SLOs; whereas, if the load decreases, a number of instances are removed from the system in order to reduce the cost. In order to effectively and efficiently utilize the elasticity property of the system, the fuzzy control should be automated. In this paper, we present our experience in designing an elasticity controller for a key-value store. The goal of our research is to investigate the

feasibility of a control theoretic approach to automation of elasticity of a Cloud-based storage by performing all design steps necessary to build a fuzzy controller for a real system, namely Voldemort (Sumbaly *et al.*, 2012), used as a case study. The design steps include defining touch points (sensors and actuators), system identification, and controller design. We have designed and developed a prototype of the fuzzy controller for Voldemort. There has been several related works in the area of the automation of elasticity Cloud-based storage services such as (Lim *et al.*, 2010) and (Moulavi *et al.*, 2012). For example, in (Lim *et al.*, 2010), the SLO is specified as a requirement on the average response time. The system variable monitored and used in feedback control is the CPU utilization, because, as shown by the authors, the CPU utilization (which is relatively easy to monitor) is highly correlated with the response time. In the case of high CPU utilization, the number of active nodes is increased by adding new nodes to the system. Similarly, the number of active nodes is decreased in the case of low CPU utilization. However, in a Cloud environment the aforementioned correlation might not hold due to the variable performance of Cloud VMs (Tickoo *et al.*, 2010) and (Iyer *et al.*, 2009).
In (Bonvin *et al.*, 2010), one of the system variables is the response time, i.e., the time it takes to send a request from a client to an application, to process the request, and to return the result to the client. The round trip time depends on several metrics such as physical medium and distance between the source and the destination, the existence of interference in the system etc. These metrics do not reflect the amount of the load in the system. Thus, the round-trip time is not a good option as a system variable to be monitored and used for control. In (Bonvin *et al.*, 2010), only scalability has been considered; however, in an elastic system, resources can be removed in the case of low workload. In this paper, we consider service time as a system variable monitored and used in fuzzy control. It is the time needed for an operation to be served in the system (a distributed storage, namely Voldemort (Sumbaly *et al.*, 2012), in our case). In other words, it does not include the network round-trip time.
This time reflects changes in the workload fashion. We design, implement, and evaluate an automatic controller, which is built based on control theory considering the elasticity property in a distributed storage. We use the Voldemort distributed key-value store as a case study. In other words, the controller would be an extension to Voldemort in a way that it scales a Voldemort cluster up by allocating more nodes in the case of a high load and scales it down by removing a number of nodes in the case of a decreasing load. The goal here is that a system uses the resources in an efficient way, so that it does not waste resources in the case of a low load. On the other hand, it adds a number of nodes to meet the SLO in case of increasing workload. In this work, we define the SLO as the 99th percentile of read operation latency over 1 minute period. Moreover, the automatic controller eliminates the need for the administrator of the system to configure the system manually to leverage the main advantage of Cloud computing (as a utility), "pay-as-you-go". The rest of paper is organized as follows. In Section II we present the architecture of our fuzzy controller integrated with the Voldemort key-value store. Section III describes the system identification process followed by the controller design described in Section IV. Evaluation of our fuzzy controller is discussed in Section V.
Finally, we present conclusions and future work in Section VI.

Fuzzy Controller Framework

We have designed and implemented a controlling framework to automate elasticity in distributed key-value stores. Fuzzy control can be manual in a way that adding or releasing resources would be done by the administrator of the system. However, our framework has been designed to monitor the load in the system and allocate or release resources based on a fuzzy controller as described in this section. When the load increases, the nodes probably cannot handle the requests in appropriate time (the SLO). Therefore, the controller would detect this and handle this issue by adding a number of nodes according to its parameters. In other word, the role of the controller is deciding about the time of adding the nodes to the system and the number of nodes that are going to be added. Similarly, when the load decreases and the service time becomes less than the SLO, the nodes are less busy and the storage can handle the requests with less number of nodes. Therefore by removing some nodes, we can save more resources and reduce the cost of using resources as a result.

A. System Architecture

We mentioned that in a Cloud environment which is dynamic, the management of resources becomes very important. They should be managed in such way that they would keep their efficiency. We design and implement a controller that monitors the performance of the storage system and requests to allocate or release the nodes based on the deviation from the desired performance caused by changes in the workload.

Figure 1 shows a generic control framework.

We can see that the system includes two inputs and one output, that the first input is the value as Average service time that is obtained from all nodes through sensors, and second input is the amount of load that enters to system, that this load considering the number of requests for delete, get and put are all reported in second, and the number of requests are obtained through the same sensor that obtains service time. Output of system in controller given the input parameters are calculated and set in an interval of numbers.

Figure 2 shows the architecture of the framework which consists of the following six major components.

• **Voldemort**: A distributed key-value store that consists of a cluster of nodes.

• **YCSB**: A benchmark tool which is an open source framework that allows creating various load scenarios (Cooper *et al.*, 2010).

• **Sensor**: this part of information system about the service time in each node and the number of requests transmitted to node for each order are obtained in three minutes and transmitted to filter.

Actuator (rebalance tool): It gets a target cluster file from the elasticity controller and updates the cluster.

• **Filter**: It smoothes the service time signal that is given to the controller by avoiding spikes in output values resulting from noise.

Fuzzy controller: the fuzzy controller is a fuzzy inference system that receives service time at any interval and also the amount of load built on system from sensor, and decides upon adding or removing the number of nodes that collected based on parameters mentioned before.

Table 1 provides a summary on components of controller body.

In the following we present in more details the components of the framework.

B. Voldemort

Voldemort is an open-source, distributed, Fault-tolerant non-relational key-value hash table. It is used, in particular, in linked in for highly-scalable storage services. Voldemort supports automatic data replication and data partitioning among multiple servers, as well as data versioning to improve data integrity in the case of failures without compromising availability. Voldemort is decentralized as the voldemort nodes are independent from each other, and hence there is no single point of failure. Voldemort includes a rebalance tool that allows adding and removing nodes. For more details on the voldemort key-value store, the reader is referred to (Sumbaly *et al.*, 2012) and (Lim *et al.*, 2010).

C. Yahoo! Cloud Serving Benchmark (YCSB)

In order to generate client requests, we have used an open source benchmark tool called Yahoo! Cloud Serving Benchmark (YCSB) (Cooper *et al.*, 2010). The most important characteristic of this tool is its extensibility which means that it can be used to benchmark Cloud storage systems and also to generate new types of workload.

D. Touch Points

According to (Hellerstein *et al.*, 2004), a touch point is an interface to them an aged resource that implements the sensor and actuator behaviors for the resource. In our work, we define a touch point as an interface to Voldemort that include sensors and actuators, which measure the service time in each voldemort server, aggregate these measurements, and actuate by adding or removing Voldemort nodes.

Sensor: A sensor is a software component that is able to monitor a Voldemort server (e.g., measure the read operation latency). We use server sensors to monitor the server load in the Voldemort cluster by measuring service time the system sensor in Figure 2 use the server sensors to collect and to aggregate measurements and give the stream of aggregated values to the filter **Actuator:** In our system, an actuator is an API provided by the voldemort rebalance tool that allows adding and removing Voldemort nodes and activate data rebalancing among nodes. An actuator is used to make some changes in the storage in order to move system performance to a desired region. From our point of view, a desired system is one that uses the resources based on the load changes. This is possible by adding or removing nodes in Voldemort. If the load increases, it will add some more resources to handle the increasing load and if the load decreases, it will remove some nodes not to waste the resources and save more money. Adding and removing nodes is done during the rebalancing process. Therefore we used the rebalance tool as an actuator.

E. Filter

Sometimes values measured by the system sensor are not smooth enough because of noise. In order to reduce the influence of noise, we use a smoothing filter that can decrease the fluctuations in the measured output values. We have designed a filter component in a way that we give more weight to the previous filter output and less weight to the new filter input that is described mathematically as follows.

Filter Output = 0.9(Previous Filter Output)+0.1(New Filter Input)

F. FIS Controller

One of the most important components in controller body is F is controller that gets filtered values from filter and decides upon adding or removing new nodes based on parameters through which controller designed. The stages to recognize system and design controller are as follows: Firstly, experiments are carried out to recognize system such that the measured data to be collected by sensors. In the next stage, recognizing system must come to realize, mentioned that recognizing system means recognizing relationships between input and output controller specified in system and how input depends on input that can be considered as a tie among applications in abstract and concrete model.

Recognizing on building a model of system. Setting measured inputs and outputs corresponding to them using system detection method, can be helpful to build a mathematical model through which estimating real system. In this system, black box method is used, that lest to build a model regardless of system's characteristics. This is a mathematical method in which the model is proposed based on measuring input and output values through detection experiments. As system under study is a complicated system with so many parameters, this model is a better choice. The process of detecting black box generally includes the stages as follow:

-specify inputs and outputs in system to use it in model
-carry out experiment and collect inputs and outputs corresponding with each other
-preprocessing data and selecting useful part of data
-design a model based on collected data
-observe behavior of system, if model does not indicate behavior of system; it has to return to the first stage

System Identification

System identification is the process of recognizing the relation between the control input and monitored output of the system and how output depends on input. It can be considered as a link between applications in the real world and model abstractions. Identification is about building a formal model of the system. Starting from a set of measured input and corresponding output values, by using a system identification approach, it is possible to create a mathematical model, which estimates the real system. In this work, we used the black-box approach (Hellerstein *et al.*, 2004), which allows building a model without knowing properties of the system. This is a mathematical approach in which a model is proposed based on measured input and output values by means of identification experiments. We used this approach in our work because the studied system (Voldemort) is a complex system with many parameters. The black-box system identification process typically passes the following steps.

• Determining inputs and outputs of the system to be used in the model;
• Experiment and collect the input and corresponding output values;
• Preprocess data and select the useful part of data;
• Design a model based on the data which have been collected;
• Observe the system behavior. If the model does not reflect the system behavior, go to the first step.

A. Input and Output of System

Input of system means the number of nodes that supposed to be removed or added. Output of system is based on amount of requests that are built on system, where a request in which experiment conducted is the time of put service, because different loads on Voldemort include logical responses.

B. State Space Model *(SSM)*

To define Voldemort system so as to design fuzzy controller, it is required forecasting all the states that might occur based on amount of input, that this has been shown in table below. Further, in this State-space model, service time and amount of load on system have been classified in five stages including, Indeed, input and output parameters are defined in table above such that appeared in fuzzy set, mentioned that how to obtain parameters in fuzzy set relies on the type of system in which parameters implemented Table I State space model As shown, input parameters are divided into five groups(very low, low, average, high, very high), yet output is defined in verbal, e.g. in case said positive of high, that is, the number of nodes must be defined regarding their ratio. After State space of system specified, addressing service time for requests then is considered through which a parameter assumed as input of controller's service time is specified by carrying out an experiment. Hence, after data passed through filter, they will be separated based on type of request (delete-get-put) so as to investigate system's reaction. Results indicate that among three requests including delete-get-put, the request put acts logically. Given that parameter (put) at the worst mode reaches to 35 minutes, and if service time is considered T(x), there would be:

$0 <= \mu T(x) <= 35$

Table I: Components in the controlling framework

Component	Tool used / Implemented in
YCSB	Embedded benchmark tool in voldemort
Actuator	Embedded rebalance tool in voldemort
Voldemort Storage	Java
Fuzzy Controller	Matlab / java
Sensor	Java
Filter	java

Table II: Linguistic variables

service time number of request	Very low	Low	Average	High	Very high
Very low	Negative of so high	Negative of so high	Negative of high	Positive of very high	Positive of high
Low	Negative of very high	Negative of high	Constant	Positive of high	Positive of very high
Average	Negative of high	Constant	Constant	Positive of high	Positive of high
High	Positive of high	Positive of high	Constant	Positive of high	Positive of very high
Very high	Positive of high	Negative of very high	Positive of high	Positive of very high	Positive of very high

We have used the Simulink environment to design the controller. The graphical designed controller in Simulink is shown in Figure 4 .As stated earlier, this controller includes two inputs The second input (St) that is mean of service time has been shown in figure, and membership functions on this input are from type of trampf, defined in [0,35]. Output of this controller has been defined in [-3,3]. Further, a variable with the preliminary value of 2 is available that output of controller is summed with this variable, and this output is set in threshold function that is available in figure. You can see threshold function in Figure 5

A=[-3,3]
B=[0,3.33e+04]
C=[0,35]
According to inputs and outputs, the expressions below are true in this controller.

$$\forall Req\ , \forall St\ \in B, C\ :\ \exists$$

Evaluation and Experimental Results

In this section we present the evaluation of the Elasticity controller for the Voldemort key-value store. We mentioned that the second step in the system identification is data acquisition. In the next section we show how we gathered data to identify the system.

A. Setup

We discuss the experimental setup in two parts, node setup and benchmark setup. In both parts, the parameters selected empirically by running various experiments that led us to the most efficient parameters. Node Setup: Our cluster consists of 8 Voldemort nodes running on 8 machines. The node setup of our experiment is described in Table III. Benchmark Setup: For effective benchmarking, we used two machines that are capable of simulating multiple clients' requests in order to load the Voldemort nodes as much as benchmark parameters were set. Table IV, shows the setup used in our benchmark. B. Benchmark Experiment for System Identification We started with two active nodes and ran the controller.

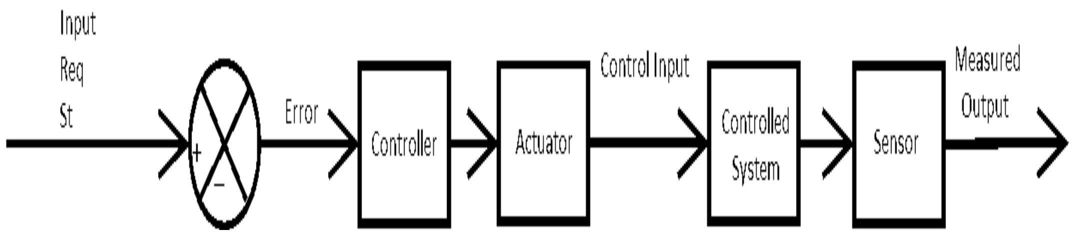

Figure 1: Generic Control Framework

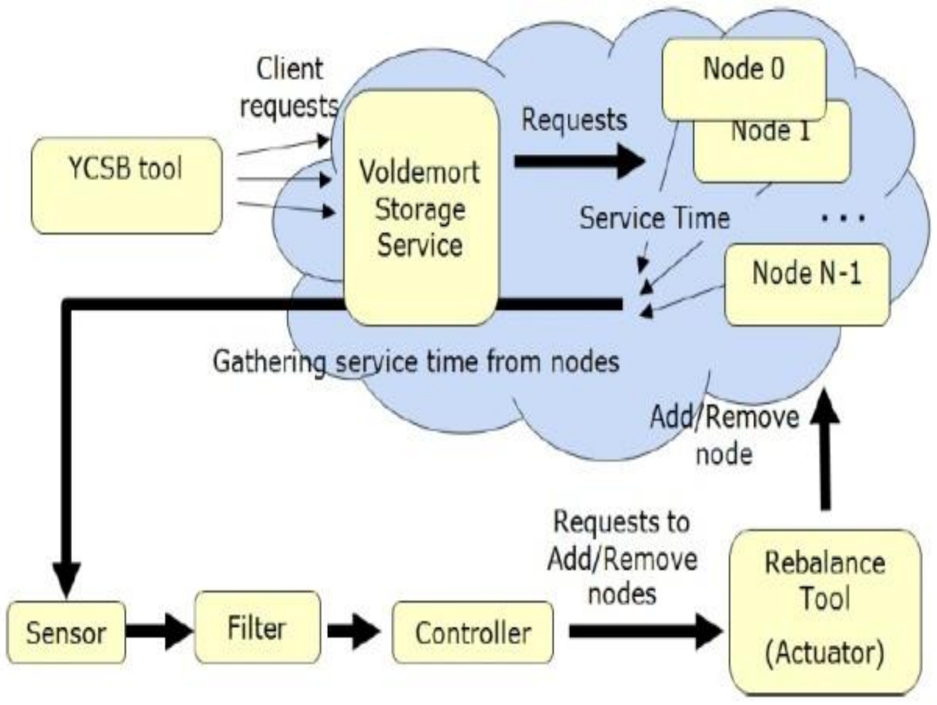
Client requests
YCSB tool
Voldemort Storage Service
Requests
Node 0
Node 1
Service Time
. . .
Node N-1
Gathering service time from nodes
Add/Remove node
Requests to Add/Remove nodes
Sensor
Filter
Controller
Rebalance Tool (Actuator)

Figure 2: Framework Architecture

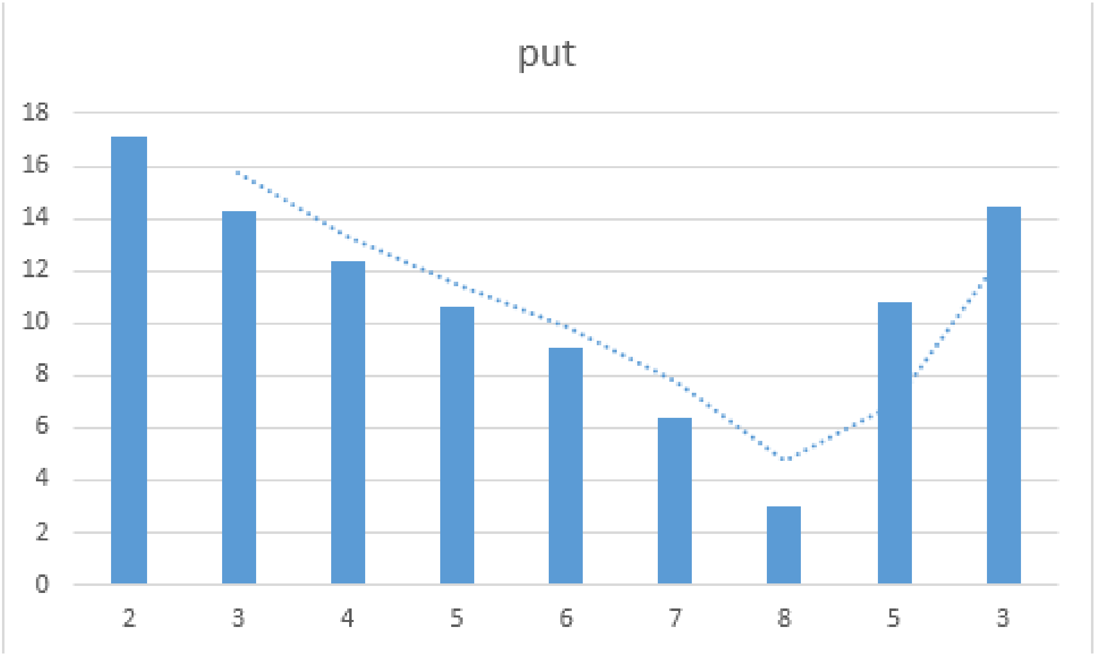

Figure 3: (Put) Service time on 8 node

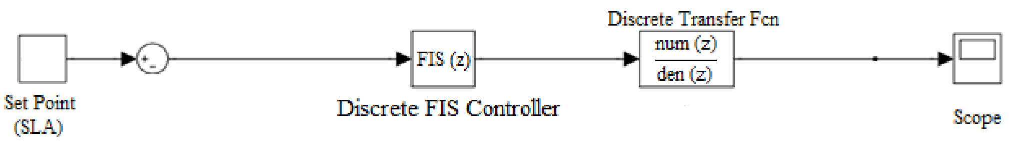

Figure 4: Graphical design of the FIS controller using Simulink

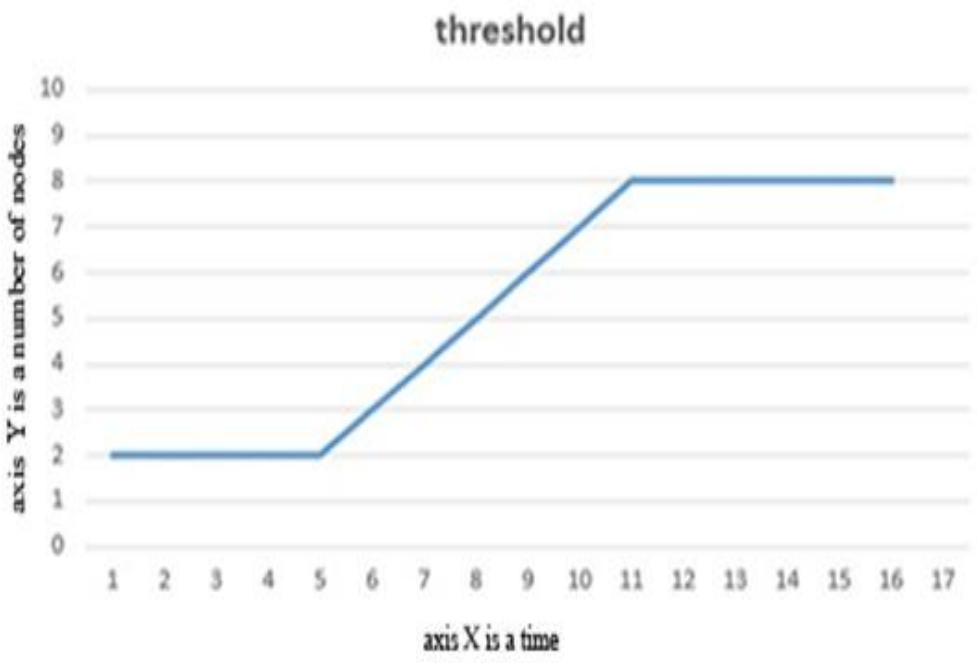

Figure 5: Threshold function for 8 node

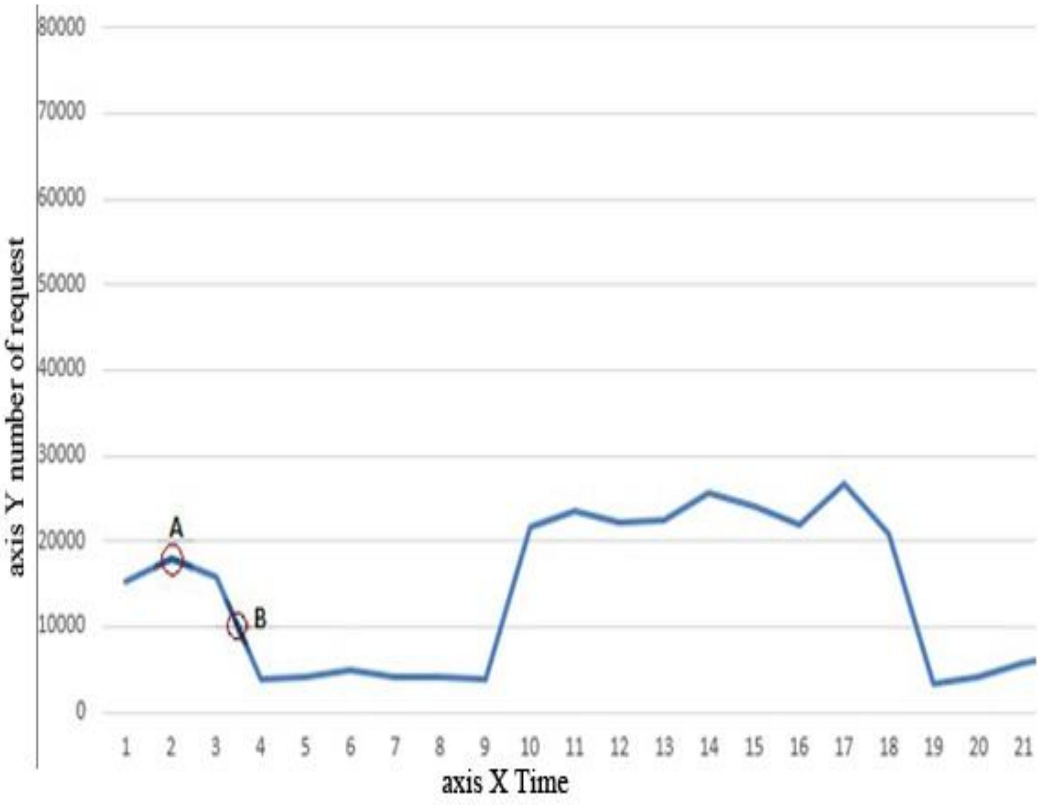

Figure 6: Request import

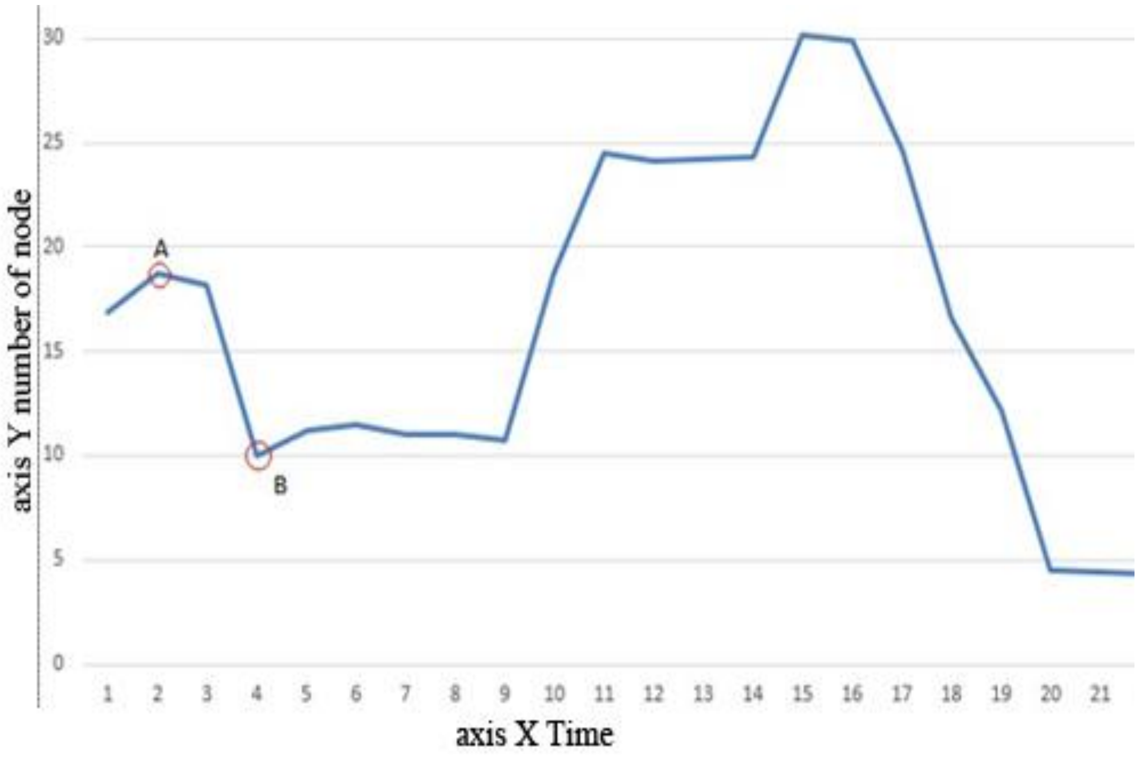

Figure 7: Service time import

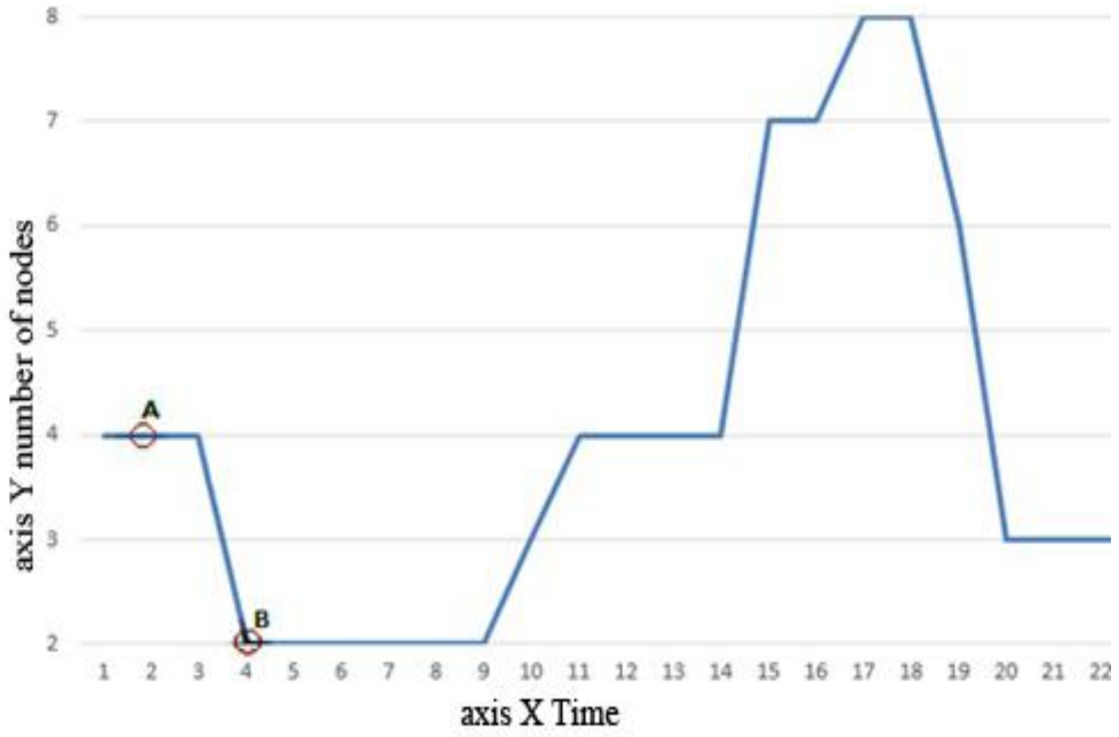

Figure 8: Export Controller

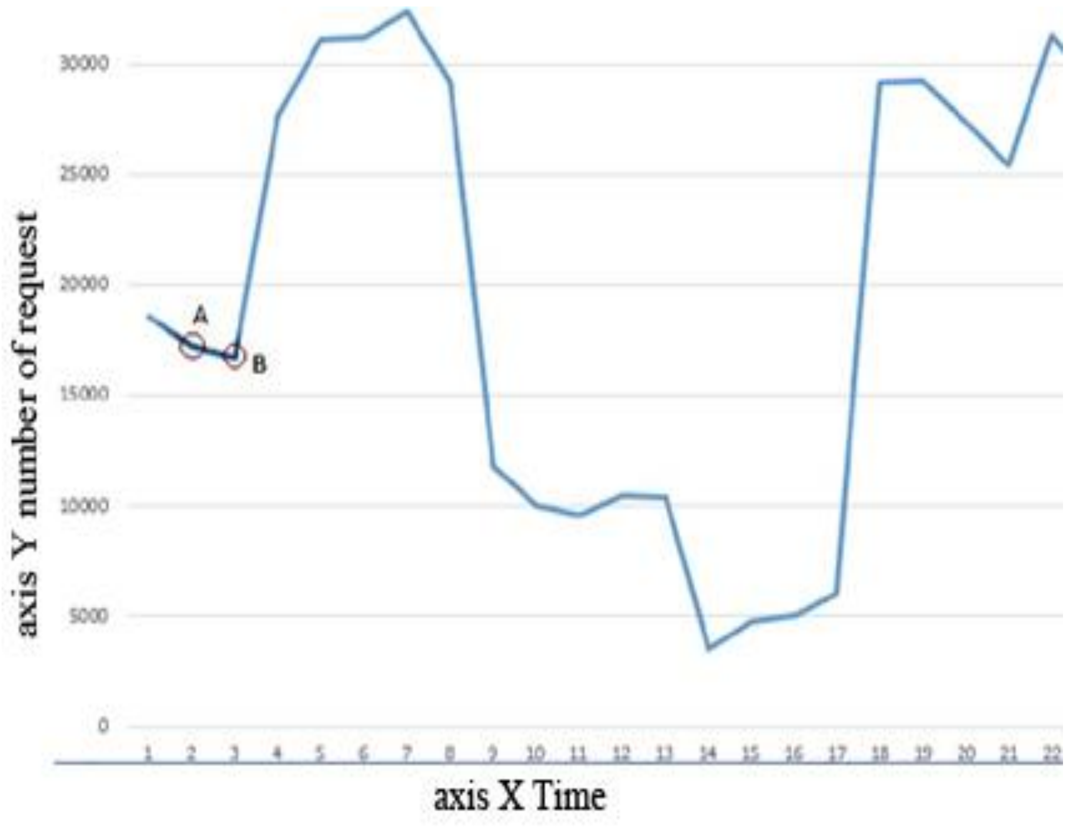

Figure 9: Request import

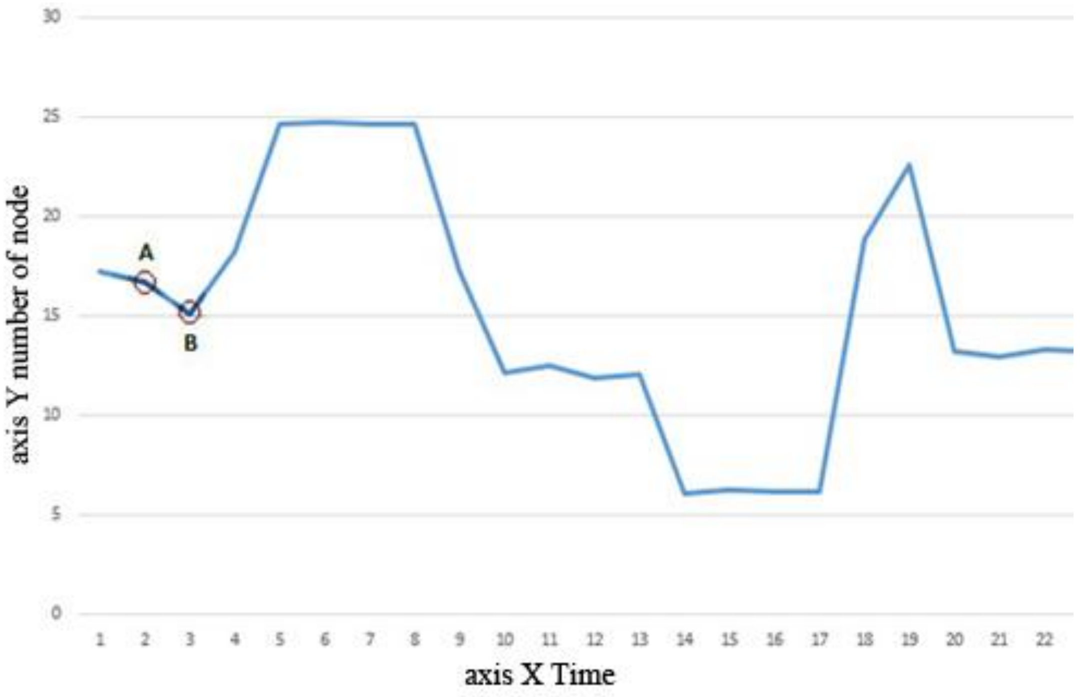

Figure 10: Service time import

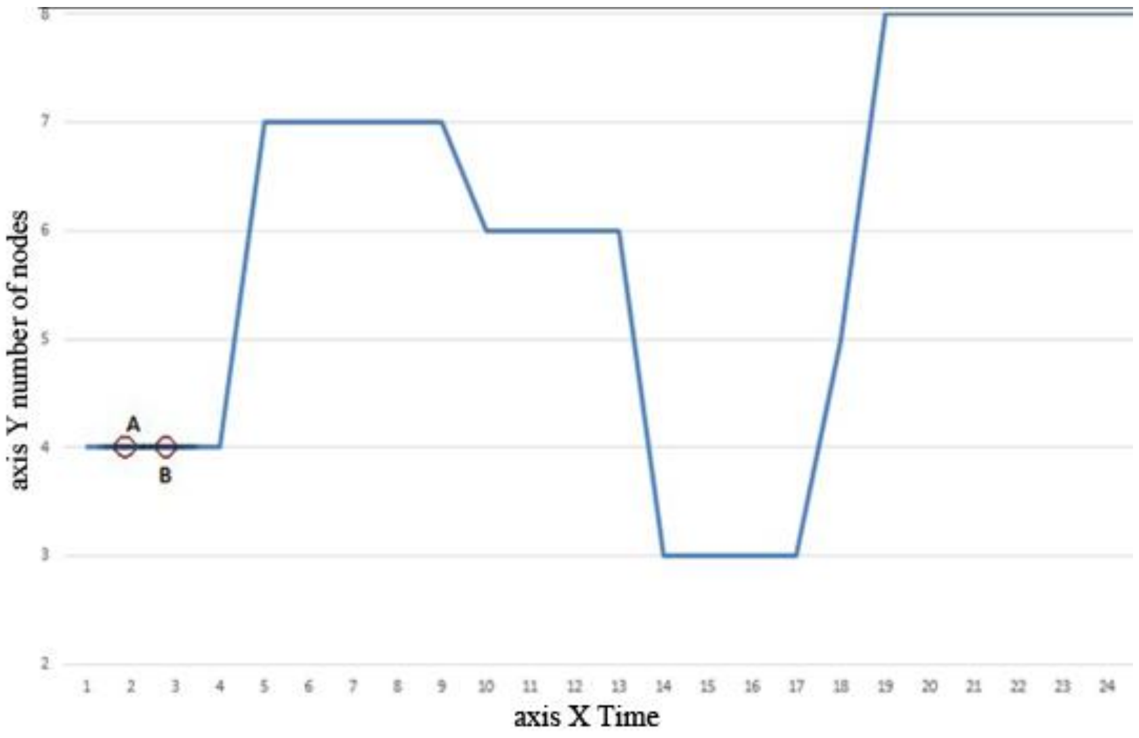

Figure 11: Export Controller

Then, a delay for 30 minutes in each load, balance in nodes 4-7 is added. Thereby, with the same delay, node 7 is removed and system works out with node 6, and by delay for 30 minutes and decreasing load on system, number of nodes decreases and system acts with three nodes, and then this experiment is repeated, and load increases on system. The results of design and experiments are shown in figure, where load on system changing from low to high emerges in next output. Note that axis of (Xs) indicates biopsy and axis of (Ys) indicates number of nodes. As shown in figures, by increasing load on system, service time increases.

C. First Experiment: Low Loading

The results of first experiment have been indicated in figure. It should be noted that sensor was Setup for 30 minutes and then setup at point A of controller, thereby loading reduces after 1 hour at point B, where reducing loading leads to decreasing service time, whereby some nodes are removed by controller mainly to avoid waste of resources. In following, loading is reduced and increased for several times and controller's reaction will be presented in figures (6)(7)(8).

Table III: Node setup for data acquisition

Machine Setup	
Parameter	**Value**
Processor	2
CPU Cores	2
model name	Core 2Duo E6400 /2.1GHz
Cache size (MB)	1
Memory (MB)	1856
Node Setup	
Parameter	Value
Voldemort Version	voldemort-0.96 &1.3.1
Database Server	local
Socket Timeout (ms)	18000
Routing Timeout (ms)	
Bdb cache size	128m
JVM SIZE (Min and Max)	
Replication Factor	2
Required Writes	2
Required Reads	2
Key Serializer's Type	String
Value Serializer's Type	String

D. Second Experiment: High Loading

The results of first experiment have been indicated in figure. It should be noted that sensor was Setup for 30 minutes and then setup at point A of controller, and then load on system at point B was increased after half an hour; thereby nodes were increased after increasing load on controller will be presented in figures (9)(10)(11).

Table IV: Benchmark setup

Machine Setup	
Parameter	**Value**
Processor	16
CPU Cores	8
Model Name	Core 2Duo E6400 /2.1GHz
Cache Size (MB)	6
Memory (MB)	16
Benchmark Setup	
Parameter	Value
Number of records inserted in warm-up	4500
Write Percentage (%)	2
Read (%)	95
Showing Result Interval (Sec)	40
Throughput (Ops/Sec)	2500
Sampling Time (min)	3

CONCLUSION

Conclusion and Works for Future

Currently, could computing provider suggest pay-as-you-go for have an access to their services and resources to customers. To access value of SLOs and apply this pricing model, cloud-based software must be provided. This study has addressed designing, implementing and evaluating Fis fuzzy controller to automate Elasticity of saving distributed major amounts that is called Voldemort. In designing controller and developing a Fis fuzzy controller, different problems including contact areas (sensors and operators), system detection and implementing controller must be taken into account. Service time has been considered as a variable of system and number of request was chosen for observation and control. Despite other methods, service time method includes time for coming and going that might cause noise in control system. An instrument made of Voldemort to build balance has been used as operator. Fis fuzzy controller integrated with Voldemort classifiers is evaluated. Evaluation indicates that developed Voldemort by means of controller has had elasticity against different loading, and had lower costs compared to methods that are based on setting fixed resources. Results of investigation indicate that controller is effective in reducing service time. In other words, Voldemort by means of controller can pave the way to access SLO, yet resources are applied effectively. We intend to pay attention in details on system requirements so as to provide Elasticity in system and settle problems that might be build in automation of elasticity by means of designing controller with feedback. Further, given the type of controller, this controller is considered to be implemented in Fuzzy Neural Network.

ACKNOWLEDGEMENT

It should be noted that some part in implementation of this project has been performed within Zahedan university in Iran; further, I take this opportunity to express my profound gratitude and deep regards to Mr Hojatollah Taheri for his comments and suggestions that came effective in this study.

REFERENCES

Bonvin N, Papaioannou TG and Aberer K (2010). A self-organized, fault-tolerant and scalable replication scheme for cloud storage. *Proceedings of the 1st ACM Symposium on Cloud Computing*, ser. SoCC '10 New York, NY, USA: ACM, 2010, Available: http://doi.acm.org/10.114 5/1807128.1807162 205-216.

Cooper BF, Silberstein A, Tam E, Ramakrishnan R and Sears R (2010). Benchmarking cloud serving systems with YCSB. *Proceedings of the 1st ACM Symposium on Cloud Computing*, ser. SoCC '10 New York, NY, USA: ACM, Available: http://doi.acm.org/10.1145/1807128.1807152 143–154.

Hellerstein JL, Diao Y, Parekh S and Tilbury DM (2004). *Feedback Control of Computing Systems* (John Wiley & Sons).

Iyer R, Illikkal R, Tickoo O, Zhao L, Apparao P and Newell D (2009). VM3: Measuring, modeling and managing VM shared resources. *Computer Networks* **53**(17) 2873–2887, Available: http://dx.doi.org/10.1016/j.comnet.2009.04.015Available:http://doi.acm.org/10.1145/1710115.1710126.

Lim HC, Babu S and Chase JS (2010). Automated control for elastic storage. *Proceedings of the 7th International Conference on Autonomic Computing*, ser. ICAC ’10, New York, NY, USA: ACM, Available: http://doi.acm.org/10.1145/1809049.1809051 1–10.

Moulavi MA, Al-Shishtawy A and Vlassov V (2012). State-space feedback control for elastic distributed storage in a cloud environment. *The Eighth International Conference on Autonomic and Autonomous Systems ICAS 2012,* St. Maarten, Netherlands Antilles 18–27.

Rastogi A (2010). A model based approach to implement cloud computing in e-governance. *International Journal of Computer Applications* **9**(7) 15–18, Available: http://www.doaj.org/doaj?func=abstract&id=658965.

Sumbaly R, Kreps J, Gao L, Feinberg A, Soman C and Shah S (2012). Serving large-scale batch computed data with project voldemort. *The 10th USENIX Conference on File and Storage Technologies (FAST’12)*.

Tickoo O, Iyer R, Illikkal R and Newell D (2010). Modeling virtual machine performance: challenges and approaches. *Sigmetrics Performance Evaluation Review* **37**(3) 55–60.