Feedback Control for Elastic Distributed Storage in a Cloud Environment

Harsh Thakor[#1], Hitesh Patel[*2]

[#1]*Department of Information Technology, Kalol Institute of Technology, Kalol*
[1]harshthakor27@gmial.com

[*2] Head of the Department, Department of Information Technology,
Kalol Institute of Technology, Kalol
[2]hiteahldit@gmial.com

Abstract— Cloud computing is an ability of a system to scale up and down (request and release resources) in response to changes in its environment and workload. Elasticity can be achieved manually or automatically. Efforts are being made to automate elasticity in order to improve system performance under dynamic workloads. In this experience in designing an feedback controller for a storage service deployed in a Cloud environment. To design our controller, we have adopted a control theoretic approach. Automation of elasticity is achieved by providing a feedback controller that automatically increases and decreases the number of nodes in order to meet service level objectives under high load and to reduce costs under low load. Every step in the building of a controller for elastic storage, including system identification and controller design, is discussed. We have evaluated our approach by using simulation. We have developed a simulation framework EStoreSim in order to simulate an elastic key-value store in a Cloud environment and be able to experiment with different controllers.

Keywords— *Cloud Enviroment, cloudservice, types of Feedback controller, Estore simulation.*

I. INTRODUCTION

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics, three delivery models, and four deployment models". NIST - Definition of Cloud Computing.[1]

On-demand self-service: A consumer with an instantaneous need at a particular timeslot can avail computing resources (such as CPU time, network storage, software use, and so forth) in an automatic (i.e. convenient, self-serve) fashion without resorting to human interactions with providers of these resources. [2]

Broad network access: These computing resources are delivered over the network (e.g. Internet) and used by various client applications with heterogeneous platforms (such as mobile phones, laptops, and PDAs) situated at a consumer's site. [2]

Rapid elasticity: For consumers, computing resources become immediate rather than persistent: there are no up-front commitment and contract as they can use them to scale up whenever they want, and release them once they finish scaling down. Moreover, resources provisioning appears to be infinite to them, the consumption can rapidly rise in order to meet peak requirement at any time. [2]

Resource pooling: A cloud service provider's computing resources are 'pooled' together in an effort to serve multiple consumers using either the multi-tenancy or the virtualization model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. The motivation for setting up such a pool-based computing paradigm lies in two important factors: economies of scale and specialization. The result of a pool based model is that physical computing resources become 'invisible' to consumers, who in general do not have control or knowledge over the location, formation, and originalities of these resources (e.g. database, CPU, etc.) . [2]

Measured Service: Although computing resources are pooled and shared by multiple consumers (i.e. multitenant) the cloud infrastructure is able to use appropriate mechanisms to measure the usage of these resources for each individual consumer through its metering capabilities. [2]

Type of basic cloud service

- **IaaS - Infrastructure as a Service-** Guest Operating Systems in the above virtualization based cloud architecture generally consists of the operating system images in the system. Developers can code in such a way that user can access the multiple virtual machines remotely. Even developers can add virtual networking devices and other required virtual wares. Such Infrastructure services can be called as **Infrastructure as a Service**. [3]
- **PaaS – Platform as a Service-** As Explained in IAAS, developers can further write such a code that they can combine the different web servers/platform (Dos etc.) and allow users to use them together in

abstract way. Such Services are known as PAAS (Platform as a service).[3]

- **SaaS - Software as a Service**-SAAS is same as normal web application but due to features of cloud computing it could have the features like elasticity, support of multiple platforms and many more. Example of service of cloud computing.[3]
- **Anything as Service (Xaas)-**

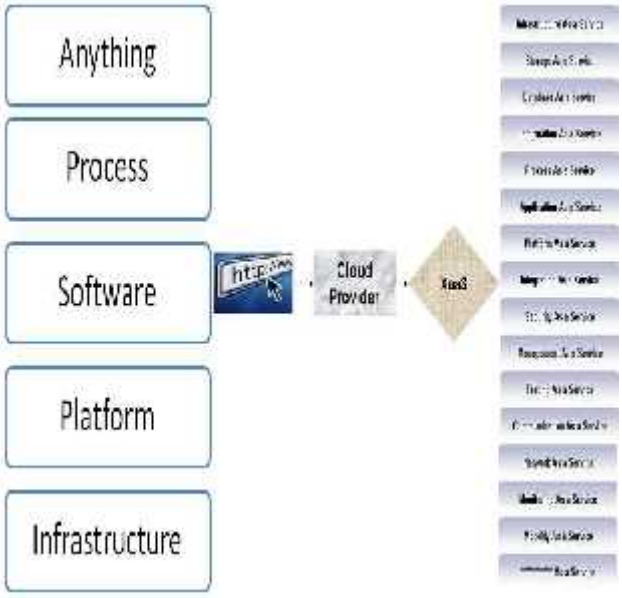


Fig.1: Anything as a Service [4]

Similarly as Explained in (IAAS, PAAS and SAAS), developers can write the code for any resources which could be used by users through web interface.[4]

II.CLOUD COMPUTING USE THE FEEDBACK CONTROL

Feedback control applies to cloud computing system as well .consider a cloud Computing system with desired output characteristic. For example-HTTP 2.0 Web service to run at no greater then 66% utilization. So that if any one of them fails the other to immediately absorbs the entire load. here the measured the output is CPU utilization.

HTTP 2.0 Web services, CPU utilization in cloud computing system, the measured output typically depend on the nature of the request being served, or workload.

Workload is often characterized in term of the arrival process (e.g.possion, self similar), and the distribution of service times for the resource used (e.g., CPU, memory, and database locks).[5]

Feedback control property

Stable-if for any bounded input, the output is also bounded. Stability is typically the first property considered in designing control systems since unstable systems cannot be used for mission-critical work.

Accurate:-control system is accurate if the measured output converges to reference input.

Steady-state error:-is the steady-state value of the control error.

Short settling times:-short settling are particularly important for Disturbance rejection in the presence of time varying workload Change.

Not ever shoot:-objective is to maximize throughput subject to the constraint that response time is less then 1 sec. [5]

Feedback Control characteristics/requirements- Workload settings could be on-off, predictable or unpredictable bursts.

Decision making system should not require detailed knowledge of implementation or any modification to the system implementation/source-code.

Many server-based systems have a single application environment that has to cater to multiple client classes. [5]

Types of Feedback control system

- **Feedback control system**
- **Feed forward control system**

[1] Feedback controllers

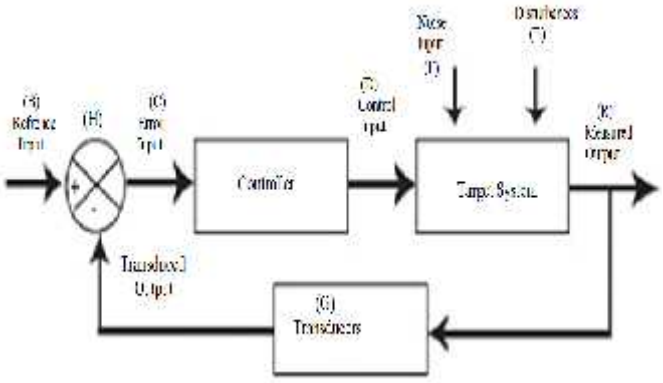


Fig: 2Block diagram of a feedback control system[5]

feedback control we typically monitor (measure) the system output that we want to regulate, which is, in our case, the 99th percentile of read operations latency over a fixed period of time (called R99p thereafter).The feedback controller calculates the error, which is the difference between the set point, which in our case is the required SLO value of R99p, and the measured system output as shown in equation.

$$e(t) = \text{SetpointSLO\_R99p} - \text{MeasuredR99p}(t)$$

For the control input, an intuitive choice would be to use the number of storage servers. In other words, to try to find how changing the number of nodes affects the R99p of the key-value store. However, there are two drawbacks for this choice of a model. First, the model does not account for the current load on the system. By load we mean the number of operations processed by the store per second (i.e., throughput). The latency is much shorter in an under loaded store than in an overloaded store. In this case, the load is treated as disturbance in the model. Controllers can be designed to reject disturbances but it might reduce the performance of the controller. Using the number of servers

(which we can control) as a control input seems to be a natural choice since we can not control the load on the system as it depends on the number of clients interacting with our web application. In the design of the feedback controller for Eastman, we propose to model the target store using the average throughput per server answer the control input. Although we can not control the total throughput on the system, we can indirectly control the average throughput of a server by adding/removing servers. Adding servers to the system reduces the average throughput per server, whereas removing servers reduces the average throughput per server. Our choice is motivated by the near linear scalability of elastic key-value stores. For example, when we double both the load and the number of servers, the R99p of the store remains roughly the same as well as the average throughput per server.[5]

[2]Feed forward controller- The controller uses the model to reason about the current status of the system and makes control decisions. If the measured throughput is far below the line, this indicates that the system is under loaded and servers (VMs) could be removed and vice versa. When a spike is detected, the feed forward controller uses the model to calculate the new average throughput per server. This is done by calculating the intersection point between the model line and the line connecting the origin with the point that corresponds to the measured throughput. The slope of the latter line is equal to the ratio of the write/read throughput of the current workload mix. Then the calculated throughput is given to the actuator, which computes the new number of servers that brings the storage service close to the desired operating point where the SLO is met with the minimal number of storage servers.

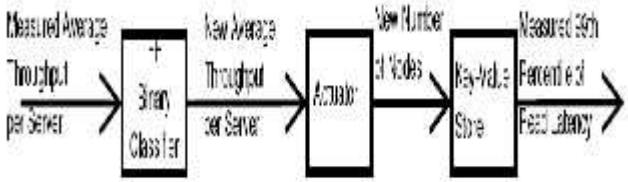


Fig: 3Feed Forward Controller

The controller uses the model to reason about the current status of the system and makes control decisions. If the measured throughput is far below the line, this indicates that the system is under loaded and servers (VMs) could be removed and vice versa. When a spike is detected, the feed forward controller uses the model to calculate the new average throughput per server. This is done by calculating the intersection point between the model line and the line connecting the origin with the point that corresponds to the measured throughput. The slope of the latter line is equal to the ratio of the write/read throughput of the current workload mix. Then the calculated throughput is given to the actuator, which computes the new number of servers that brings the storage service close to the desired operating point where the SLO is met with the minimal number of storage servers.

Feed forward controller does not measure the R99p nor does make decisions based on error but relies on the accuracy of the model to check if the current load will cause an SLO violation. This makes the feed forward controller sensitive to noise such as changes in the behavior of the VM provided by the Cloud.[5]

III. PROBLEM STATEMENT

We are all the generally use the web 2.0 application. Wikis, social networks, media sharing and rapidly growing number of users and the amount of user generated data. Problem is the challenge for storage service batter performance. Every time maintain Growing number of users and the amount of data (scalability).How to manage uneven load, user geographically scattered (low request latency, load balancing). Partial failures, very high load (high availability). Acceptable data consistency guarantees (e.g., eventual consistency).

IV. PROPOSED SYSTEM AND SIMULATION ENVIRONMENT

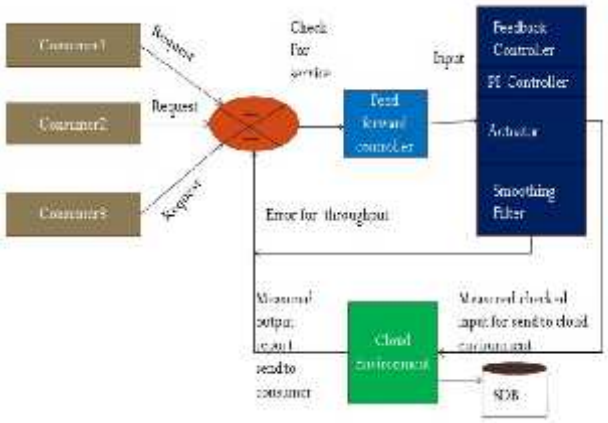


Fig: 4Controllersusing cloud computing Architecture

The performance model used to predict the performance of the system given its current state (e.g., current workload).However, due to the variable performance of Cloud VMs (compared to dedicated physical servers), it is difficult to accurately model the performance of the services running in the Cloud.

To address the challenges of controlling a noisy signal and variable performance of Cloud VMs, ElastMan consists of two main components, a feed forward controller and a feedback controller. ElastMan relies on the feed forward controller to handle rapid large changes in the workload (e.g., spikes). This enables ElastMan to smooth the noisy 99th percentile signal and use feedback controller to correct errors in the feed forward system model in order to accurately bring the 99th percentile of read operations to the desired SLO value. In other words, the feed forward control is used to

quickly bring the performance of the system near the desired value and then the feedback control is used to fine tune the performance.

Cloud Instance Component represents an entire storage instance within a cloud. The component architecture for instance is shown in Figure.

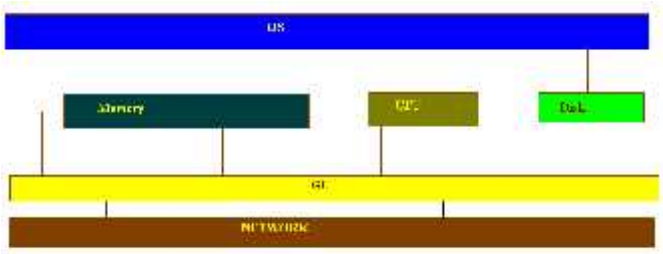


FIG: 5 CLOUD INSTANCE COMPONENTS

Cloud Provider Component represents an important unit in the implementation. It is the heart of a simulated Cloud computing infrastructure and provides vital services to manage and administer the nodes (VM instances) within the Cloud. The Cloud provider component architecture.

Feedback Controller represents the controller that can connect to the Cloud provider and retrieve information about the current nodes in the system. The main responsibility of the controller component is to manage the number of nodes currently running in the Cloud. In other words, it attempts to optimize the cost and satisfy some SLO parameters. The overall component architecture is For further information on EStoreSim please refer.

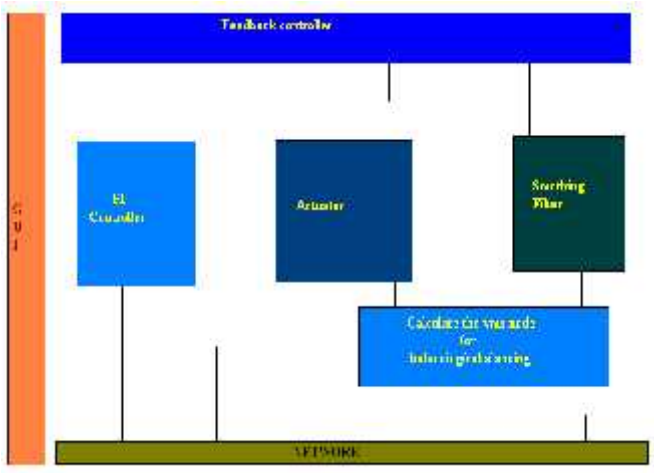


Fig: 6 Feedback controller

Feed forward /Feedback Proposed Controller Flow Chart

Our elasticity controller ElastMan combines the feed forward controller and feedback controller. The feedback and feed forward controllers complement each other. The feed forward controller relies on the feedback controller to correct errors in the feed forward model; whereas the feedback controller relies on the feed forward controller to quickly respond to spikes so that the noisy R99p signal that drives the feedback controller is smoothed.

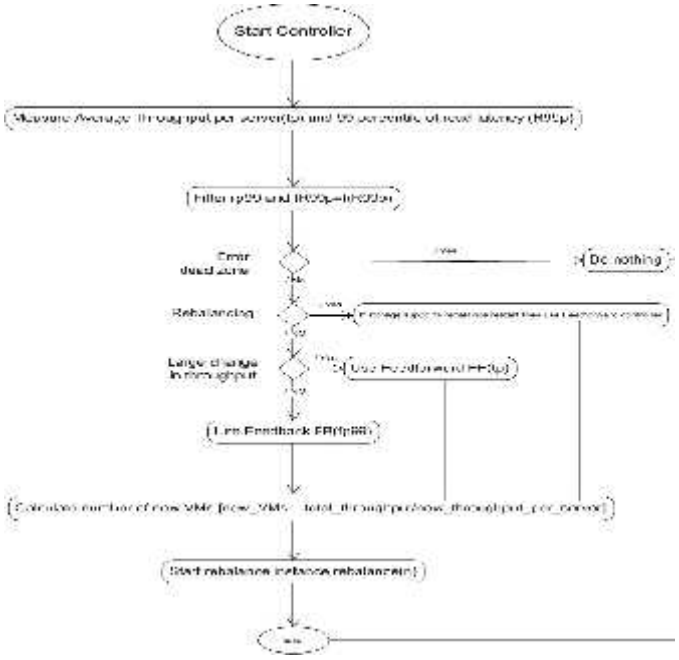


Fig: 7 Proposer Algorithm for Feedback/feed forward Controller system

The ElastMan elasticity controller starts by measuring the current 99th percentile of read operations latency (R99p) and the current average throughput (tp) per server. The R99p signal is smoothed using a smoothing filter resulting in a smoothed signal (fR99p). The controller then calculates the error ***e***. If the error is in the dead zone defined by a threshold around the desired R99p value, the controller takes no action. Otherwise, the controller compares the current **tp** with the value in the previous round. A significant change in the throughput (workload) indicates a spike. The elasticity controller then uses the feed forward controller to calculate the new average throughput per server needed to handle the current load. On the other hand, if the change in the workload is relatively small, the elasticity controller uses the feedback controller which calculates the new average throughput per server based on the current error. In both cases the actuator uses the current total throughput and the new average throughput per server to calculate the new number for servers.

During the rebalance operation, which is needed in order to add or remove servers, both controllers are disabled as proposed by Lim at all. The feedback controller is disabled because the rebalance operation adds a significant amount of load on the system that causes increase in R99p. This can mislead the feedback controller causing it to wrongly add more resources. However if the storage system supports multiple rebalance instances or modifying the currently running rebalance instance, the feed forward controller can

still be used. This is because the feed forward controller relies on the measured throughput of read/write operations (and it does not count rebalance operations) thus it will not be affected by the extra load added by the rebalancing operation.

Because the actuator can only add complete servers in discreet units, it will not be able to fully satisfy the controller actuation requests which are continuous values. For example, to satisfy the new average throughput per server, requested by the elasticity controller, the actuator might calculate that 1.5 servers are needed to be added (or removed). The actuator solves this situation by rounding the calculated value up or down to get a discrete value. This might result in oscillation, where the controller continuously adds and removes one node. Oscillations typically happen when the size of the storage cluster is small, as adding or removing a server have bigger effect on the total capacity of the storage service. Oscillations can be avoided by using the proportional thresholding technique as proposed by Limited. The basic idea is to adjust the lower threshold of the dead zone, depending on the storage cluster size, to avoid removing a server that will result in SLO violation and thus will request the server to be added back again causing oscillation.

V. RESULTS AND ANALYSIS

The instance configurations for these experiments are as follows:

- CPU frequency: 4 GHz;
- Memory: 8 GB;
- Bandwidth: 1 MB/s;

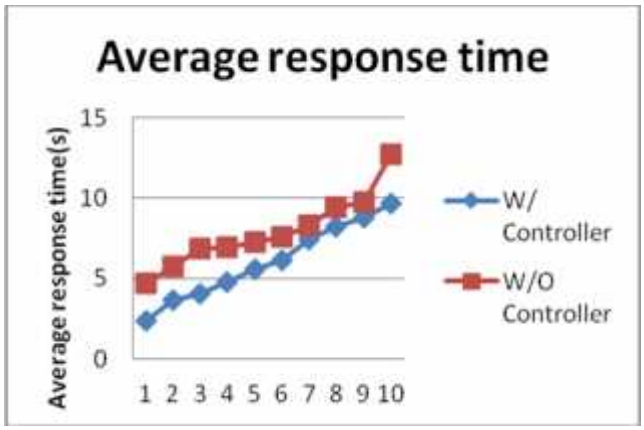
Number of simultaneous downloads: 70

Results-I for SLO Increasing Work Load:

In this series we conducted two experiments: one with controller and another without controller. In the results and figures presented below, they are denoted by **w/controller** and **w/o controller**, respectively. Each experiment starts with three warmed up instances. By a warmed up instance we mean that in this instance each data block is requested at least once thus it resides in the memory of this instance.

Workload that is used for this experiment is of two levels: normal and high. Under the normal load the time interval between consecutive requests is selected from a uniform random distribution in the range [10, 15] seconds that corresponds to an average request rate of 4.8 requests per minute. Under the high load the time interval between consecutive requests is selected from a uniform random distribution in the range [1, 5] seconds that corresponds to an average request rate of 20 requests per minute. The experiment starts with normal load and after 500 seconds the workload increases to the high level. This is shown in Fig.

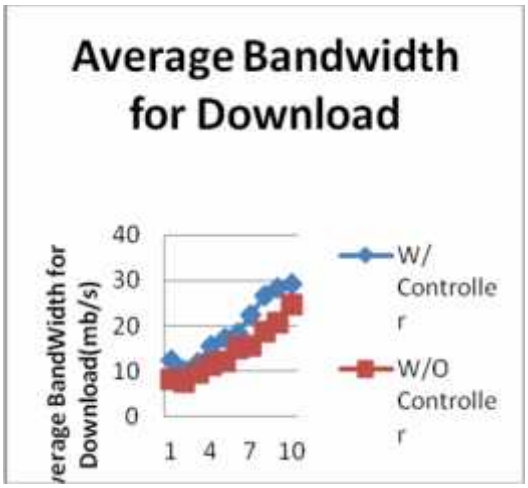
Sensing of instance output is done every 25 seconds. In the case of controller, actuation is performed every 100 seconds. Thus there are 4 sets of measured data at each actuation time that the controller should consider. In order to calculate values of the system output, the controller computes averages of data sets. The duration of each experiment is 200 seconds with warm up of 100 seconds. SLO requirements are as follows:



Time (s)

Fig: 8 Average response time(s)

Depicts the Average Response Time for the 1 to 10 experiments Response time we mean the time that it takes for an instance to respond to a request that download is started and not the actual download time. As it is seen from the diagram, the average response time for the experiment with the controller is generally lower than the experiment without controller.



Time(s)

Fig: 9 Average Bandwidth for Download

Average bandwidth per download. If an instance has a bandwidth of 4 Mb/s and has two current downloads running, the bandwidth per download is 2 Mb/s. As can be seen from the 1 to 10 experiment with controller shows significantly higher bandwidth per download. This is mainly because the instances receive high number of requests and bandwidth is divided among less requests also. This will end up having higher bandwidth available on each instance.

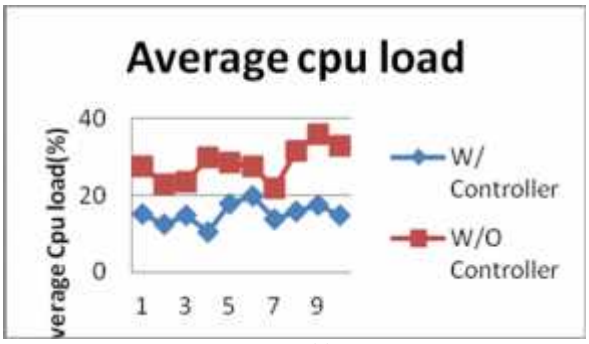


Fig: 10 Average CPU Load (%)

Depicts the Average CPU Load for the aforementioned 1 to 10 experiments. The Average CPU Load is the average of all nodes' CPU Loads at each time the sensing is performed. As one can see in Figure 5.1 CPU loads for the experiment with the controller is generally lower than the same experiment without the controller. This is due to the controller that launches new instances under high workloads causing a huge drop in average CPU Load.

SLO Violation final result

SLO mines the Service level organization is basically all the service maintain controller. SLO Violation means the number of service to check that controller and without controller. And how to check SLO violation so this equation see below it.

$$\text{SLO Violations} = \frac{100\% \times \text{Number of SLO Violations}}{\text{Total Number of SLO Check}}$$

Table SLO VIOLATIONS

SLO violation	W/Controller	W/O Controller
CPU Load	51%	72%
Response time	4.52	7.0473
Bandwidth	7(MB/s)	4(MB/s)

Table: SLO VIOLATIONS

Fig: 11 SLO VIOLATIONS

VI. CONCLUSION AND FUTURE WORK

Cloud environment and described the steps in designing it including system identification and controller design. The controller allows the system to automatically scale the amount of resources while meeting performance SLO, in order to reduce SLO violations and the total cost for the provided service. We also introduced our open source simulation framework (EStoreSim) for Cloud systems that allows to experiment with different controllers and workloads. We have conducted two series of experiments using EStoreSim. Experiments have shown the feasibility of our approach to automate elasticity control of a key-value store in a Cloud using feedback control. We believe that this approach can be used to automate elasticity of other Cloud based services.

REFERENCE

[1]NIST, Definition of Cloud Computing, Draft version http://csrc.nist.gov/groups/SNS/cloud-computing/index.html.
[2] http://en.wikipedia.org/wiki/Cloud_computing.
[3] "Cloud Security Issues", Balachandra ReddyKandukuri, Ramakrishna Paturi V, Dr. Atanu Rakshit (2009)
[4]NIST Definition of Cloud Computing, Draft version http://csrc.nist.gov/groups/SNS/cloud-computing/index.htm.
[5] Feedback control of computing system,Joeph L Heslersten.

Book:

[1] EC2LAB- SAAS USING AMAZON ELASTIC CLOUD COMPUTE, Manisha Gaikwad

DISSERTATIONS THESIS:

[1] Peter Sobeslavsky, **"**Elasticity in Cloud Computing", Joseph Fourier University, 2011