21st International Symposium on
Mathematical Theory of Networks and Systems
July 7-11, 2014. Groningen, The Netherlands

Stability Analysis of Cloud Computing Systems under Uncertain Time delays

Saikrishna PS, Ramkrishna Pasumarthy, Kruthika H.A.

Abstract— **In this paper we discuss the notion of stability applied to computational systems with focus on Cloud computing systems. We consider the case study of a web service application hosted on a private Cloud environment. The effects of input delay on auto-scaling mechanism used for resource provisioning on a cloud is examined. The stability analysis is developed based on a dynamic model of the Cloud obtained by system identification. For the model obtained we apply LMI conditions necessary for estimating the upper bound on the time delay for stability. This upper bound is derived based on the assumption of a predefined maximum workload in connections/sec or equivalently the throughput parameter. This analysis of time delay helps in evaluating the performance of the Cloud, measured in terms of the response time of accessing the web service and the throughput. The experiments are performed on a Eucalyptus based cloud setup.**

I. Introduction

A Cloud Computing environment has emerged as an attractive model for providing IT based services like computing, storage, and network to large as well as small enterprises. The popularity of Cloud computing is due its essential characteristics like on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service [20]. This enables users, administrators and researchers to reduce costs in setting up large scale infrastructures for various purposes. The on demand scalability also referred to as auto-scaling is being used to avoid over-provisioning of resources and to prevent degradation of offered quality of service at all times. Feedback control theory has been an efficient tool in addressing such problems. We refer to [8], [9], [10], [11], [15], [16], [17] for work related to dynamic resource provisioning. For an efficient controller design, a system model representing relationship between input and output is necessary. For physical systems fairly accurate models can be derived from first principles. However, for computational systems although queuing theory allows for analysis of quantities such as utilization and latency, it may not be fine grained for control over short time scales [10]. Therefore, most of the work in literature relies on empirical modelling for model based and adaptive control. With this approach the empirical relationships between variables are derived. The models obtained via these relationship are then used for designing appropriate controllers. The disadvantage of this approach is that the models obtained are static in nature and fail to capture the dynamic behaviour of the target system.

Saikrishna PS, Ramkrishna Pasumarthy and Kruthika HA are with Department of Electrical Engineering, IIT Madras, India. ramkrishna@ee.iitm.ac.in, ee12d025@ee.iitm.ac.in, ee11s039@ee.iitm.ac.in

In this paper we present a model based auto-scaling framework for a Private Cloud based on open source Eucalyptus platform and study the effect of delays in this mechanism. In order to demonstrate the auto-scaling feature of the Cloud, we take a case study of a web service application. It is well known that the web traffic fluctuates unpredictably over time and hence to meet the quality of service (QoS) requirements of the clients is a challenging task [17], [18]. Also forecasting a web based load does not guarantee good results owing to the stochastic nature of the web traffic. A natural choice in such circumstances is to rely on a dynamic provisioning based on Cloud to meet the ever-changing demand from normal load to peak load. In [15], [16], [17] authors have implemented model based controllers for web services by identifying the systems based on the dynamic processes taking place in the web-server. We use grey-box approach to derive state space models for our cloud application. It provides a scalable approach to model systems with large number of inputs and outputs [12]. Other benefits of using state space models is that they can be easily extended. Earlier work on state-space modelling for Cloud was presented by [14] on a simulation framework called EStoresim. For our experimental setup, we propose discrete time state-space model valid around a single operating point. We then proceed for controller synthesis based on this model without delay. As we perform our experirments on a private Cloud within an internal network the delays would be small for demonstration of instability. For a realistic scenario of a Public Cloud, we simulate the input delay. This would enable us to include communication and networking delays as observed in a Public Cloud at any given time. We also assume a maximum connection rate which the Cloud set-up would experience under realistic workloads. We then analyse the stability of the auto-scaling mechanism for different values of delays. We use literature based on [1] for analysis of discrete time delay systems for conditions on delay dependent stability. The rest of the paper is organized as follows: Section II describes the overview on stability computational systems while Section III and IV presents an introduction to the Cloud and effects of time delay on Cloud computing systems. In Section V , we describe system identification technique. Later in Section VI we find the bound on time delay for stability. Finally we present the theoretical results and simulation results in sections VII and end with some concluding remarks in Section VIII.

ISBN: 978-90-367-6321-9

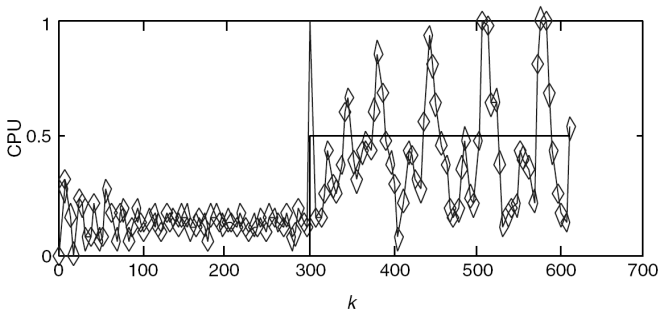

Fig. 1. Unstable Apache HTTP Server

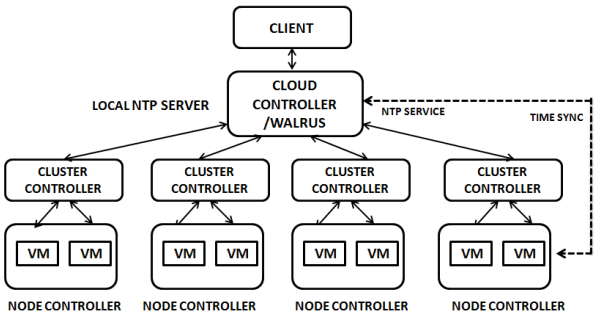

Fig. 2. Eucalyptus Cloud Architecture

II. Stability of Computational Systems

Several properties of feedback control systems can be applied to computational systems. In general a control system is stable if it produces a bounded output for a bounded input. Since a computational system does not produce an unbounded output, the instability is manifested in terms of large oscillation between extreme values of the performance parameters. See Figure 1. The performance metrics such as overshoot, settling time can be applied to these systems as well. This ensures better transient and steady state behaviour of the computational systems under time varying workloads. A computational system with a feedback control is expected to to converge, even though it may not be constant due to stochastic nature of other processes involved it [13]. These systems have operating regions where they deliver good performance and other regions where they do not. This is in-turn reflected in terms of performance metrics as mentioned earlier. The illustration of instability of an Apache HTTP server was demonstrated by [12], where controller overreacts to the scholastics in the CPU utilization metric. See Figure 1. The amplitude of oscillations are constrained by the range of the CPU utilization metric. Such a behaviour may lead to abnormal QoS (quality of service) for the end user.

III. Introduction to the Cloud :

A. Architectural Overview:

Eucalyptus [19] is an on-premise private Cloud platform, designed as a distributed system. It consists of a modular set of 5 components which are Linux based processes that interact with each other to form the Cloud layer. Figure 2 shows the architecture of Eucalyptus Cloud platform. The components that comprise the Eucalyptus architecture are:

1) Node Controller (NC)
2) Cluster Controller (CC)
3) Walrus Storage Controller (WS3)
4) Storage Controller (SC)
5) Cloud Controller (CLC)

B. Software and component Communication:

1) Each of the Eucalyptus components acts as an independent web service that exposes a Web Service Description Language (WSDL) document defining the application program interface (API) to interact with it [19]. We have installed the CLC and Walrus components on the same server so that they operate as separate web services within a single Java environment, and can use a fast-path for inter-service communication.
2) Walrus is a simple storage service for Eucalyptus Cloud, separating it from the CLC may decrease the efficiency of the messaging between the two services, and therefore we have installed walrus service togather with the CLC.
3) The CLC and Walrus communicate with Eucalyptus clients independently. End-users interact with Eucalyptus through a client interface via euca2ools (linux command line interface) or browser based Hybridfox [19].
4) To enable communication between the services CLC/Walrus/SC/CC, we need to open/enable the ports 8443 and 8773 through 8777 in the firewall settings. This allows Eucalyptus components to communicate over a private network.
5) We use static networking mode for the Cloud network, this mode allows the administrator to predefine the IP addresses for the virtual machines to be instantiated on the NC machine. For more details refer to [19].
6) The request for a Virtual machine (henceforth called as VM) can be made by Cloud user or by the administrator himself. In both the cases the request for a new VM to the NC is made by the CLC service.

C. Experimental Testbench

1) **Setup:** We have setup the Cloud with 2 physical machines, the front end machine hosts CLC/Walrus/CC/SC components and a VT (virtualization enabled) second physical machine has the NC component installed. By adding more NC components we can scaleup the capacity of the Cloud. The table in Figure 3 shows the machine configuration and roles.
2) **Time Synchronization:** A local NTP (Network Time Protocol) server service is setup on the front end machine which servers as the time server for all the Eucalyptus components.
3) **Image management:** An image defines what will run on a guest instance with the Eucalyptus Cloud.

MACHINE NAME	Machine Role	CPU (Cores)	RAM (GB)	Disk (GB)	eth0	eth1
Cloud Controller	CLC/SC/CC /WALRUS	8	16	2000	PRIVATE	PUBLIC
Node Controller	VM HOST	24	48	500	PRIVATE	PUBLIC
Client	Request Generation	8	8	500	PUBLIC	--------

Fig. 3. Machine Roles and Configuration

We have developed a customized Eucalyptus machine image from ubuntu 12.04 desktop image and uploaded it to the front end. We then installed Apache2 web-server on the image with a static web content of size of 150kB.

4) **Load Balancing:** It is a mechanism to distribute workload among different VM's as and when they are made available by auto-scaling. We have used Haproxy (High Availability Proxy) load balancer for our setup [21].

IV. Time Delay in Cloud Computing Systems

The time delay in feedback control is inevitable with the extensive use of networked control systems [2], [3] and [4]. Most of the control systems operate in the presence of delays due to data acquisition, decision making and execution of a control philosophy. In process control delay is due to mass and energy flow. The effect of delays become more pronounced in distributed and interconnected systems. On a broader perspective we focus on delays in networked control systems which are due to the various links in the communication medium. Here the delays appear in parallel computation and computer networking. Cloud computing architectures use a network of computational nodes called VM's to achieve performance levels that are not attainable by a single VM. By a mechanism called auto-scaling in the Cloud the resources are made available to the user on demand. Here the time delay due for VM availability is often overlooked. Therefore auto-scaling would be more effective if the VM provisioning is done on time. The delays in a cloud environment are mainly due to the following:

1) Networking and communication delays.
2) VM start-up time or booting time delays.

The first type of delays are of the order of ms to few $seconds$. Whereas the second type are of the order of minutes, in this case the VM is yet to be instantiated for use. Start-up time delay is inevitable in case of unavailability of a released VM in the resource pool of the Cloud. These can have serious implications on the nature of application deployed on cloud. In our case we study the impact of type (1) delays on stability of web-service application hosted on cloud. Earlier works by [5],[6] have addressed this issue and have made a performance study of different Cloud providers with respect to different types of VM's. But no solution has been proposed to solve this problem for time critical applications. Hence we have proposed a control theoretic approach to analyse the stability of Cloud computing system. Figure 4 shows the web application setup on the Cloud.

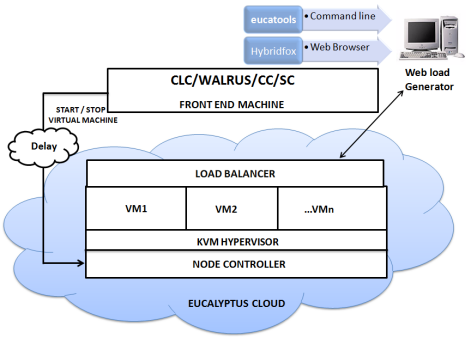

Fig. 4. Cloud Application Setup

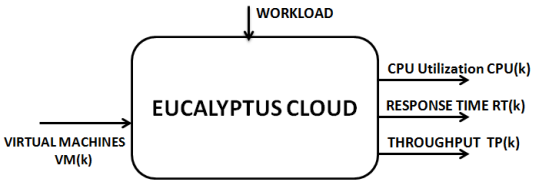

Fig. 5. A Schematic of the Input Output model of the Webserver on Cloud

V. System Identification

A. State Variables

We aim to develop linear state space models valid around a single operating point based on the experimental data. The state variables considered are CPU utilization of the node controller in percentage, web-server response time and throughput in $seconds$ and MB/sec respectively. The input to the model is the number of VM's and outputs are same as the state variables. The schematic of the same is shown in Figure 5. The following relations describe the evolution of the states of the system:

1) **CPU Utilization:** The CPU utilization of the node controller machine running the instances is represented as $CPU(k) = x_1(k)$. It depends on the number of VM's and the past CPU value. This is modelled as
$x_1(k+1) = a_{11}x_1(k) + b_1u(k), \quad u(k) = VM(k)$
for certain constants a_{11}, b_1, which are to be identified and will be identified later in the paper.
2) **Response time:** The time taken to respond to a request from the workload generator represented as $RT(k) = x_2(k)$. It depends on the past value of the CPU, the response time and the number of VMs.
$x_2(k+1) = a_{21}x_1(k) + a_{22}x_2(k) + b_2u(k)$
3) **Throughput:** The average rate of successful request delivery over the internal network between the client and the node controller, represented as $TP(k) = x_3(k)$. It depends on the past CPU, throughput and the number of VM's.
$x_3(k+1) = a_{31}x_1(k) + a_{33}x_3k) + b_3u(k)$

Since we operate at single operating points we define offset values for outputs and inputs to the model as:
$y_1(k) = x_1(k) - \overline{x_1}$
$y_2(k) = x_2(k) - \overline{x_2}$

$y_3(k) = x_3(k) - \overline{x_3}$ and
$u(k) = u_1(k) - \overline{u_1}$
The linear state space dynamics can now be formulated as a matrix equation as under:

$$x(k+1) = Ax(k) + Bu(k)$$
$$y(k) = Cx(k) + Du(k) \quad (1)$$

where, $x(k) \in \mathbb{R}^3$, $A \in \mathbb{R}^{3\times3}$, $B \in \mathbb{R}^3, u \in \mathbb{R}$, $C = \mathbf{I}_{3\times3}, D = 0$ and $\overline{(\cdot)}$ represents the value at the operating point.

B. Measurement and Actuation:

We acquire the state variable CPU Utilization from the /proc/stat folder in linux root file system. Since CPU utilization has stochastic noise, we use a 3 point moving average filter for smoothing the real time data. For the measurement of response time and throughput, we rely on our custom made workload generator program described in the next section. Actuation or starting a new VM is done based on the commands executed from the controller modelled in Simulink. These commands are finally executed by the bash script (in linux) to start/stop a VM. Refer to Figure 11 in section VII for measurement and control scheme.

C. Customized workload generator:

We have developed a program in bash script (linux) using well known HTTP load generator called httperf [23] to load the VM's on the node controller. The program developed takes into consideration the following limits for client from which this program is run. For the client to sustain the generated load, it is important to keep these limits so as to avoid assuming client performance limits as server performance limits. These are sizes of:

1) TCP port space
2) Number of open file descriptors
3) Socket buffer memory

For more information refer to [23].
The workload generator continuously acquires different parameters like response time, reply rate, standard deviation and errors while loading the VM's.

D. Software Component Tuning

1) Linux kernel parameter tuning: sysctl is a program used to modify kernel settings on the linux operating system. This allows one to optimize specifically the way your kernel is handing things, specifically, networking. We load the optimal parameters to /etc/sysctl.conf file for the parameters to remain even after reboot. The command sysctl -p is used to load the parameters to the kernel.
2) Apache web-server tuning: We tune the max-clients and keep-alive parameters of the apache web-server such that the maximum throughput is achieved [24].
3) Workload generator tuning: The workload generator can generate any arbitrary workload based on a set of tuning parameters.

E. Parameter Estimation:

The following are the important considerations for parameter estimation for the state space models for Cloud described in section A. For more information refer to [12], [22].

1) Operating region corresponds to Node Controller CPU Utilizations 4.6-34.6% which corresponds to VM utilization from 5-62%. We have selected the operating regions based on the load capacity of the workload generator and the Node controller machine. This is decided by factors described in the previous sections C and D.
2) For estimating the parameters we excite the system with triangular workload from the load generator. This means that the HTTP request rates are raised from low to high value and vice versa. Measurement of that state variables is performed as mentioned in subsection B. The load generator program is tuned to excite the target system to cover the operating region mentioned earlier. Tuning is done such that after every 5 seconds a VM is started, and this process is continued till we reach the maximum number of VMs. Similarly we decrease the number of VMs when workload decreases after every time 5 seconds till we reach the minimum. We then collect the data for estimation using MATLAB. The following values for the system matrices and the corresponding input matrices were obtained around the operating point:
For Node Controller CPU utilizations between 4.6 - 34.6%:
$$A = \begin{bmatrix} 0.8448 & 0 & 0 \\ 0.0044 & 0.9861 & 0 \\ -104.87 & 0 & 0.8678 \end{bmatrix}, B = \begin{bmatrix} 0.6687 \\ -0.0149 \\ 837.263 \end{bmatrix} \quad (2)$$
3) Figure 9 represents the input and Figures 6, 7 and 8 represent the experimental values of the CPU Utilization of node controller, the response time and throughput of the web service hosted on VM's. It can be observed that increase in the number of VM's has an effect of increase in throughput and decrease in response time. These plots are similar to a the observations made by previous works [17], [7], [8] and [10] to name a few. The response time as observed is in milli-seconds and decreases exponentially with the addition of VM's. As compared to previous works mentioned above the variation is small because of the fact that the workload generator is in the same network accessing web-server with static web pages. The Throughput parameter increases with the number of VM's. These modelling results can be extended to any multi-tier and/or dynamic web page application hosted on Cloud.
4) The validation of the state space model was performed with the experimental data. The model output fits to the validation data to 83.5% for CPU , 63.4% for response time and 93% for throughput data. Hence this is a fairly

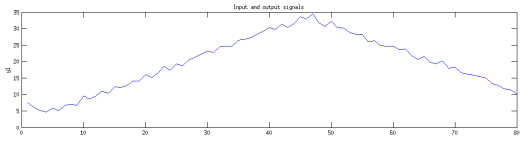

Fig. 6. CPU utilization (output)

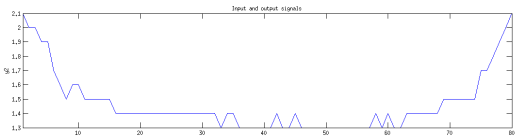

Fig. 7. Response time (output)

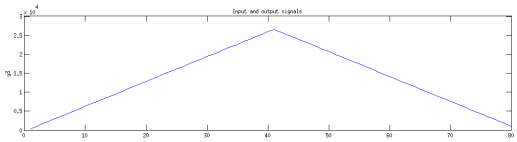

Fig. 8. Throughput (output)

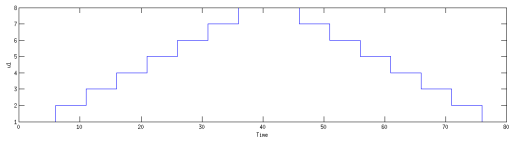

Fig. 9. Virtual Machines Input

accurate model except for the response time parameter due to its non-linear relationship with the input.

.

VI. Time Delay in the Cloud

A. Cloud Model with Delay

We model the our Cloud application with delay as a state feedback control scheme. A dynamic state feedback controller as shown in Figure 11, tracks the reference and this is similar to a PI controller for SISO systems. The state vector of the model is augmented with control error $e(k) = r(k) - y(k)$, where $r(k)$ is the reference and $y(k)$ is the output. We use integrated control error which describes the error accumulated over time denoted by $x_I(k)$. It is computed using the difference equation $x_I(k+1) = x_I(k) + e(k)$. The augmented state vector is $\begin{bmatrix}(\mathbf{x(k)}\\ x_I(k))\end{bmatrix}$. We then obtain the closed loop model :

$$\begin{bmatrix}(\mathbf{x(k+1)}\\ x_I(k+1))\end{bmatrix} = (\begin{bmatrix}\mathbf{A} & 0\\ -\mathbf{C} & 1\end{bmatrix}\begin{bmatrix}\mathbf{x(k)}\\ x_I(k))\end{bmatrix} + \mathcal{B}u(k) + \begin{bmatrix}0\\1\end{bmatrix} r \quad (3)$$

where $\mathcal{B}$ is given by

$$\mathcal{B} = \begin{bmatrix}\mathbf{B}\\0\end{bmatrix}\begin{bmatrix}-(\mathbf{K}_P & K_I)\end{bmatrix})$$

With input delay we have,

$$\begin{bmatrix}(\mathbf{x(k+1)}\\ x_I(k+1)\end{bmatrix} = (\begin{bmatrix}\mathbf{A} & 0\\ -\mathbf{C} & 1\end{bmatrix}\begin{bmatrix}\mathbf{x(k)}\\ x_I(k))\end{bmatrix} + \mathcal{B}u(k-h_k) + \begin{bmatrix}0\\1\end{bmatrix} r \quad (4)$$

After substituting for $u(k)$ and new input matrix $\mathcal{B}$ we have

$$\begin{bmatrix}(\mathbf{x(k+1)}\\ x_I(k+1))\end{bmatrix} = (\begin{bmatrix}\mathbf{A} & 0\\ -\mathbf{C} & 1\end{bmatrix} + \begin{bmatrix}\mathbf{B}\\0\end{bmatrix}\begin{bmatrix}-(\mathbf{K}_P & K_I)\end{bmatrix})\begin{bmatrix}\mathbf{x(k)}\\ x_I(k))\end{bmatrix} + \begin{bmatrix}0\\1\end{bmatrix} r \quad (5)$$

With input delay the above equation can be written as :

$$\begin{bmatrix}(\mathbf{x(k+1)}\\ x_I(k+1))\end{bmatrix} = \begin{bmatrix}\mathbf{A} & 0\\ -\mathbf{C} & 1\end{bmatrix}\begin{bmatrix}\mathbf{x(k)}\\ x_I(k))\end{bmatrix} + \begin{bmatrix}\mathbf{B}\\0\end{bmatrix}\begin{bmatrix}-(\mathbf{K}_P & K_I)\end{bmatrix})\begin{bmatrix}\mathbf{x}(k-h_k)\\ x_I(k-h_k))\end{bmatrix} + \begin{bmatrix}0\\1\end{bmatrix} r \quad (6)$$

The cloud model with delay can be written in the standard form for time delay systems as:
$\mathbf{x}(k+1) = A\mathbf{x}(k) + A_1\mathbf{x}(k-h_k)$
where the matrices A and A_1 are given by

$$A = \begin{bmatrix}\mathbf{A} & 0\\ -\mathbf{C} & 1\end{bmatrix} and \quad A_1 = \begin{bmatrix}\mathbf{B}\\0\end{bmatrix}\begin{bmatrix}-\mathbf{K}_P & K_I\end{bmatrix} \quad (7)$$

We consider throughput variable for static state feedback and for reference tracking with $r(k) = 20000$ to illustrate the effect of delay on stability. Therefore, **C**= $\begin{bmatrix}0 & 0 & 1\end{bmatrix}$. Considering the static state feedback, we place pole of the throughput variable at 0.8, this would yield a value of:
$\mathbf{K}_P = \begin{bmatrix}-0.0670 & 0 & 0\end{bmatrix}$
Similarly, we use Integrator constant $K_I = 0.00002$ for a good convergence with zero delay. It is observed in simulations that higher values of K_I leads to instability without delay. For simulation, the initial conditions for the Cloud model is takes as:
$X0 = \begin{bmatrix}8.5; 1.5; 6917\end{bmatrix}$
This selection is based on the input-output data used for identification. We also choose the initial condition for the delay block used for simulation to be zero for simplicity.

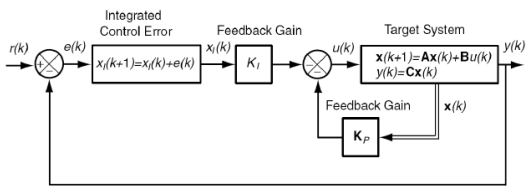

Fig. 10. Dynamic State Feedback controller

B. Stability Analysis with Time Delay

To evaluate the stability conditions with time delay, we refer to the LMI conditions given by [1]. Consider the discrete time system :

$$x(k+1) = (Ax(k) + A_1 x(k-h_k)) \\ x(\theta) = \Psi(k), \theta \in \{-h_M, -h_{M+1}, ..., 0\} \quad (8)$$

where, $x(k) \in \mathbb{R}^n$ is the state vector at instant k, h_k is a positive number representing the time delay of the system that satisfies the following $0 < h_k \leq h_M$, where h_M is a positive integer representing the maximum time delay of the system.

Theorem 1: The System given by (8) is asymptotically stable with $h_k = h_M$, if there exist real symmetric matrices $P > 0$, $Q > 0$ and $Z > 0$ satisfying the following LMI:

$$\Phi = \begin{bmatrix} \Gamma & (A+A_1)^T P A_1 & -A^T P A_1 & h_M (A-I)^T Z \\ * & -Q & -B^T P B & h_M B^T Z \\ * & * & -Z & 0 \\ * & * & * & -Z \end{bmatrix} < 0 \quad (9)$$

$$where, \Gamma = -P - P^T + Q + A^T P (A + A_1) + (A+B)^T P A \quad (10)$$

We obtain the matrices A and A_1 as shown below:

$$A = \begin{bmatrix} 0.8448 & 0 & 0 & 0 \\ 0.0044 & 0.9861 & 0 & 0 \\ -104.8700 & 0 & 0.8678 & 0 \\ 0 & 0 & -1.0000 & 1.0000 \end{bmatrix} \quad (11)$$

$$A_1 = \begin{bmatrix} -0.0448 & 0 & 0 & 0.0000 \\ 0.0010 & 0 & 0 & -0.0000 \\ -56.0966 & 0 & 0 & 0.0167 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (12)$$

VII. Simulation Results

Based on the Theorem 1 explained in section VII B, we use YALMIP toolbox in MATLAB to obtain a bound on delay for aymptotic stability to be 8 seconds.

We then implement the dynamic state feeback loop for the system shown in Figure 10 in Simulink. The results of simulations carried out on the model are shown in Figures 12, 13, 14 and 15. It can be observed from Figure 15 that the number of virtual machines requirement without delay is 3 with a quick convergence to the target value. For 5 second delay, the virtual machine requirement goes up to 4 and finally converges to 3. Similarly for delay of 12 seconds, the VM requirement goes upto 5 and converges to 3. The VM requirement for delay of 15 seconds repeatedly fluctuates from 0 to 6 without convergence thereby demonstrating instability. The effect of variation in the input VMs due to delay is reflected in the state variables as shown in Figures 12, 13, and 14. The oscillations in the state variables increase with delay,

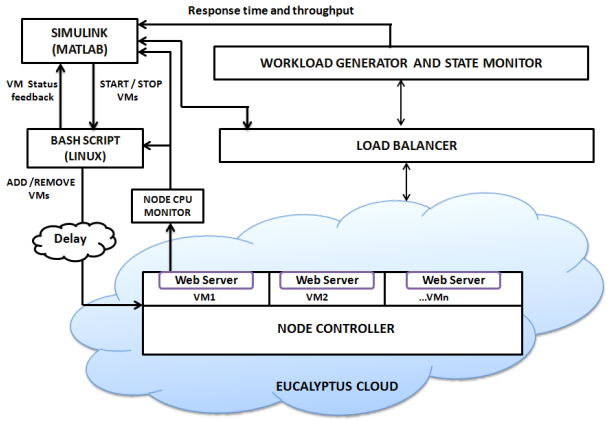

Fig. 11. Control Architecture

and beyond 15 seconds the controller performance degrades further. This is well in agreement with the theoretical bound calculated to be 8 seconds calculated based on Theorem 1. It can be seen that for a Cloud computing system, the conditions obtained on delay give a fairly good idea of the system stability. Moreover, with respect to the performance of the hosted application this bound on the time delay serves as a guideline for complete performance degradation.

A. Controller Architecture in Linux:

We propose to implement control architecture as shown in Figure 11 in in continuation with the present work .It represents a practical implementation of the dynamic state feedback controller as shown in Figure 11. We can measure the CPU utilization from the idle CPU time from linux using a code written in bash script (in Linux). The response time and throughput can be measured by the workload generator and state monitor. All the states above are interfaced to MATLAB by file I/O. This architecture would enable us to validate the results of controller design with time delay.

VIII. Conclusions and Future Work

In this paper we have built an experimental test-bed for a Eucalyptus Cloud to demonstrate effect of input delays on the stability. The hosted application performance was analysed under time delay based on the state space model valid around a single operating point. The LMI conditions provide sufficient conditions for the delay dependent stability which are consistent with simulations on the dynamic model. We then proposed a controller architecture for validation in real time. This would help in validation of controller design for time delay systems.

It can also be observed from Figure 15 that beyond 12 seconds, the number of virtual machines requirement fluctuates from 0 to 6 without any convergence. This can drastically increase the cost of VM resources due to the fact that the pricing of VMs is based on hourly usage for any commercial Cloud. The results obtained by simulation also indicate that

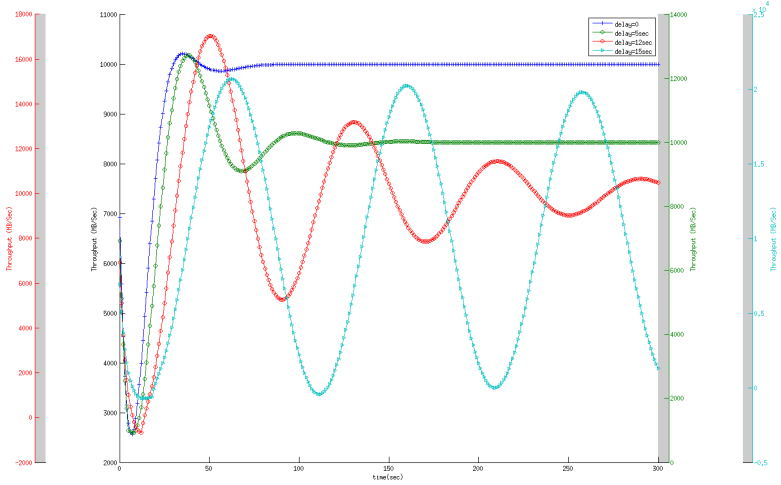

Fig. 12. Throughput of web-server on Cloud with delay

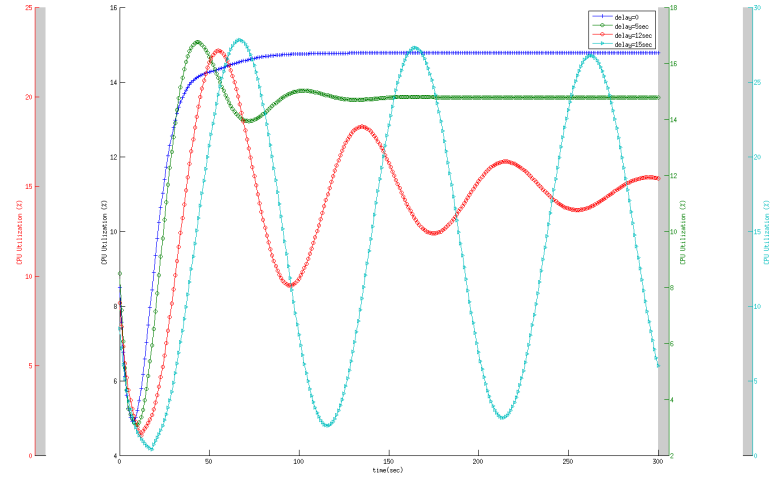

Fig. 13. CPU Utilization of web-server on Cloud with delay

during instability the throughput which is equivalent to connections/sec accepted by the web-service fluctuates rapidly and hence the response time experienced by the clients would vary from minimum to maximum values, thereby effecting the quality of service. Further it is evident from Figures 12 and 14 that for delay below 12 seconds there is performance degradation compared to the case of zero delay. Therefore, even though stability is guaranteed for the Cloud system, we observe a deterioration in performance. We thus need to modify the controller design to guarantee not only stability but also performance with reasonable cost of resources. We plan this for our future work. We also plan to work on modelling based on several operating points to get better models. This will lead to a switched discrete time delay system with a switching controller for better estimate of performance.

REFERENCES

[1] S.B. Stojanovic , L.J. Dragutin and Debeljkovic. Delay Dependent Stability Analysis for Discrete Time Systems with Time Varying State Delay, *CI and CEQ*, Vol. 17, No.4 pp. 497-503, 2011.

[2] D.S. Kim, Y.S. Lee, W.H. Kwon and H.S. Park. Maximum allowable delay bounds of networked control systems, *Control Engineering Practice*, Vol. 11, pp. 1301-1313, 2003.

[3] P. Antsaklis, J. Baillieul. Control and Communication Challenges in Networked Real-Time Systems *IEEE Trans. Aut. Control, special issue on networked control systems*,Vol. 49, pp. 1421-1597, 2004.

[4] M.Y. Chow. Guest Editorial on The Special Section on Distributed Network-Based Control Systems and Applications, *IEEE Transactions on Industrial Electronics*, Vol. 51, pp. 1126-1279, 2004.

[5] Ming Mao and Marty Humphrey. A Performance Study on the VM Startup Time in the Cloud. *IEEE CLOUD* 2012: 423-430

[6] Ming Mao and Marty Humphrey. Scaling and Scheduling to Maximize Application Performance within Budget Constraints in Cloud Workflows. *IPDPS* 2013: 67-78

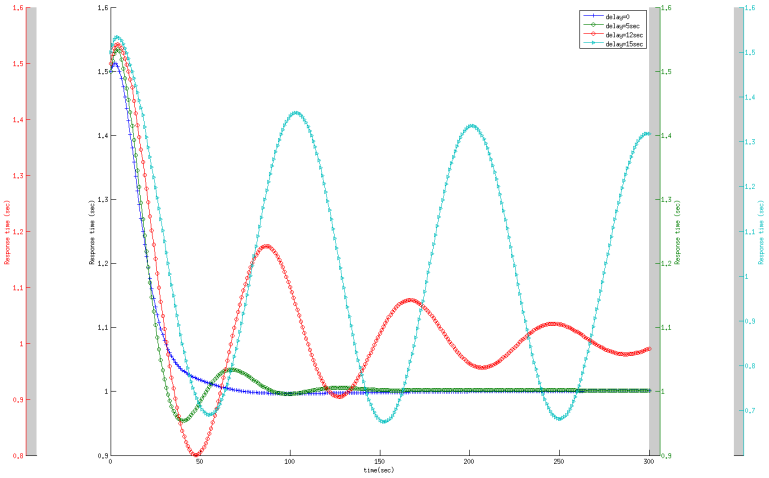

Fig. 14. Response time of web-server on Cloud with delay

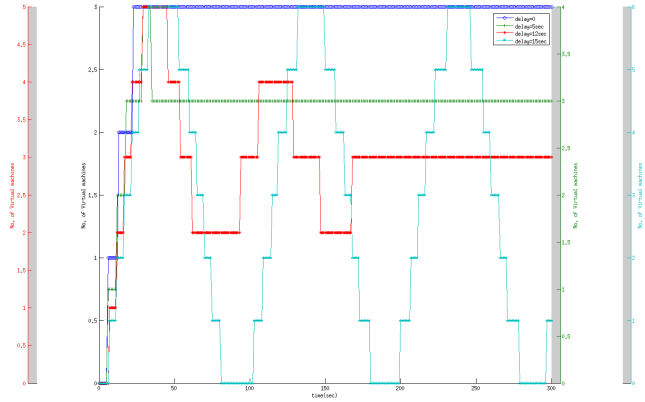

Fig. 15. Variation in the number of Virtual Machines with delay

[7] Banga Druschel. Measuring the Capacity of a Web Server, *USENIX Symposium on Internet Technologies and Systems*

[8] B. Urgaonkar, P. Shenoy, A. Chandra, and P. Goyal. Dynamic provisioning of multi-tier internet applications. *ICAC*, 2005.

[9] H. Lim, S. Babu, J. Chase, and S. Parekh. Automated Control in Cloud Computing: Challenges and Opportunities. *1st Workshop on Automated Control for Datacenters and Clouds*, June 2009.

[10] P. Padala, K. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, and K. Salem. Adaptive Control of Virtualized Resources in Utility Computing Environments. *EuroSys*, 2007.

[11] M. Assuncao , Evaluating the Cost-Benefit of Using Cloud Computing to Extend the Capacity of Clusters. *18th ACM International Symposium on High performance Distributed Computing (HPDC 2009)*, pp. 141-150.

[12] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury. *Feedback Control of Computing Systems*. John Wiley and Sons, 2004.

[13] Y.Diao and Thomas J. Watson. A control theory foundation for self-managing computing systems, *IEEE Journal on Selected Areas in Communications*, 23 , Issue 12.

[14] M. Amir Moulavi, Ahmad AL-Shishtawy, and Vladimir Vlassov. State Space Feedback Control for Elastic Distributed Storage in a Cloud,*(ICAS 2012), March 25-30, 2012. St. Maarten, Netherlands Antilles, pp. 18-27*

[15] D. Lonescu, B. Solomon, M. Litoiu, and G. Iszlai. Observability and controllability of autonomic computing systems for composed web services, In 6th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI 2011),*2011.*

[16] B. Solomon, D. Ionescu, M. Litoiu and G. Iszlai. Autonomic Control of Composed Cloud Based Services, In Proceedings of the 2010 International Workshop on Software Engineering for Adaptive and Self-Managing Systems, *2010.*

[17] T. Abdelzaher, K.J Shin and N. Bhatti. Performance Guarantees for Web Server End-Systems: A Control Theoretical Approach. IEEE Transactions on Parallel and Distributed Systems, *Vol. 13, No. 1, Jan 2002*

[18] Sinung Suakanto, Suhono H Supangkat and Roberd Saragih. Performance measurement of Cloud Computing services International Journal of Cloud Computing:Services and Architecture, *vol.2, No.2, April 2012*

[19] http://www.eucalyptus.com/eucalyptus-cloud

[20] http://csrc.nist.gov/publications/nistpubs/

[21] http://haproxy.1wt.eu/

[22] L. Ljung. System Identification: Theory for the User. 2nd ed. *Prentice Hall, Upper Saddle River, NJ, 1999.*

[23] http://www.hpl.hp.com/research/linux/httperf/

[24] http://httpd.apache.org/docs/2.2/misc/perf-tuning.html

[25] http://users.isy.liu.se/johanl/yalmip/