Modeling and Performance management of a Virtualized Web-server

Durgesh Singh
Department of Electrical Engineering
IIT Madras, India
ee15s073@ee.iitm.ac.in

Saikrishna PS
Department of Electrical Engineering
IIT Madras, India
ee12d025@ee.iitm.ac.in

Ramkrishna Pasumarthy
Department of Electrical Engineering
IIT Madras, India
ramkrishna@ee.iitm.ac.in

ABSTRACT

This paper presents the performance management of a virtualized web-server hosting a dynamic web page. Firstly, a dynamic model of the virtualized web-server in the form of a linear state space model is developed using grey-box identification technique using input-output data. Secondly, model validations are presented to justify the efficacy of the model. Based on the developed model, a state feedback controller is designed for performance guarantees in terms of the client perceived response time under changing workload conditions.

Finally, we compare the performance of the state feedback controller with a fuzzy controller developed based on fuzzy rule base that represents the expert knowledge of the system. The real time control experiments are presented to compare the two control schemes on a virtualized server with KVM hypervisor.

CCS Concepts

•Computing methodologies → Model verification and validation;

Keywords

Virtualization, Web-server, State feedback control, Fuzzy control

1. INTRODUCTION

In server virtualization, a physical server is converted into several individual and isolated virtual servers also called virtual machines (VMs) using a hypervisor such as KVM (Kernel-based Virtual Machine) and Xen etc. Full virtualization, paravirtualization, OS (operating system) virtualization are major server virtualization approaches [9]. Some of the important advantages of virtualization are improvement in server availability, elimination of server sprawl and centralized administration.

DOI: 10.1145/1235

Over the past few decades, internet has seen a remarkable transformation from being a technical curiosity to the necessity of many people's lives indicating traffic to various web-sites. Whenever traffic of a web-site increases, there is a delay in serving requests due to overloading of servers. Recently, one of the premier Indian E-commerce company Flipkart's "Big Billion Day" sale ran into glitches when the servers were not able to handle the heavy traffic [10]. Similar instances of server overload leading to performance degradation can be found in [14].

In this paper, we present the performance management of a web-server hosted on a virtualized physical machine (VPM) with a dynamic web page. It is a multi-tier web-server with the first tier hosting an Apache web-server, a CGI (Common Gateway Interface) containing an executable code as the second tier and a database server as the third tier. For simplicity, all these tiers are installed and configured on each of the VMs of the VPM using XAMPP software [2].

Earlier work on performance management of servers include analytical models with optimal control in the work by authors of [11], self-tuning fuzzy controller in [18], (our earlier work on) state feedback control using pole placement technique for a web-server hosting a static web-page [17], hill climbing technique with analytic queuing model [12] and some others [16]. Compared to the previous approaches where either analytical modeling or queuing models were used, we present a grey-box identification technique for modeling and state feedback control for a web-server hosting a dynamic web page on a virtualized server (henceforth called the target system).

Apache [15] is the most popular open-source HTTP server for modern operating systems like Ubuntu and Windows. For a standalone Apache web-server, performance control can be achieved by varying KeepAlive(KA) and MaxClients (MC) [8]. However, for a virtualized system with hosted web-servers, the performance control is considerably difficuilt due to the increase in system complexity. The common issue one faces is that all the web-servers behave in entirely different as they share some of the common resources of the virtualized system such as network I/O and last level cache (LLC) [6].

The novelty of our approach lies in the application of system identification technique which has been traditionally applied to dynamical systems such as process plants, mechanical systems etc. Due to complex nature of the target system, we seek a data-driven approach [5] for modeling the target system dynamics. In this approach, the input-output data

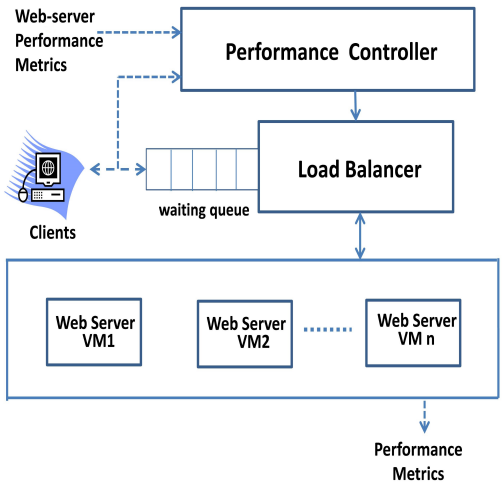

Figure 1: Performance management of web-servers

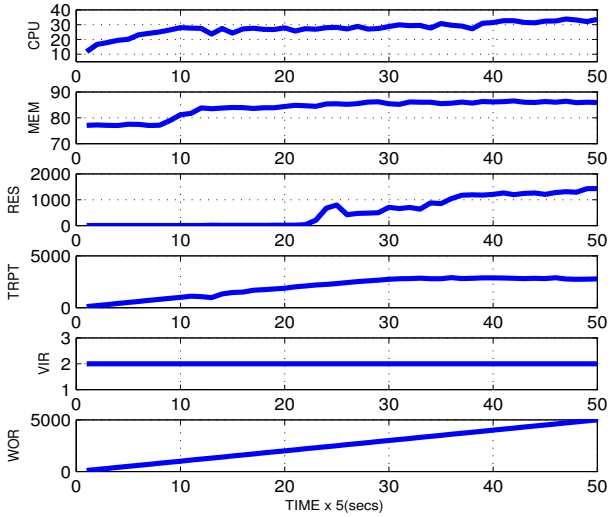

Figure 2: Response of hosted web-server with increasing workload with two running VMs

is collected from the target system by exciting it with a predefined pattern of the control input while applying HTTP (Hypertext Transfer Protocol) workload. This ensures that the validity of the model under different control input and HTTP workload patterns [8].

Each of the VM hosting a web-server with a dynamic web page simultaneously solves a set of ODE's (ordinary differential equations) embedded as a CGI script. As HTTP load increases on the web-server there is an increase in the CPU and memory utlization, the response time and also the throughput. Therefore, we choose these variables as state variables in the proposed state space model (explained later in the paper). We also chose the constraints on CPU and memory utilization for each of the VM as $\leq$ 80 % and $\leq$ 90 % respectively. The constraints are provided so that none of the incoming requests fail or get delayed due to unavailability of CPU or memory resources.

The summary of contributions of this paper are as follows: Firstly, a linear state space model of the target system is developed using grey-box identification technique, secondly, we apply state feedback control for performance control of the web-servers. And thirdly, we compare the performance of state feedback controller with Fuzzy controller which is developed based on fuzzy rule base that represents the expert knowledge of the target system behaviour.

The rest of the paper is organized as follows: Section 2 of this paper presents the modelling of the web-server system. Section 3 explains working and default control scheme of Apache web server. Section 4 and Section 5 describe the State feedback controller and fuzzy controller respectively. Comparison of both controllers are illustrated in Section 6. Section 7 consists the conclusion including future work. Throughout the paper each VM represents a single web server so these texts will be used interchangeably.

2. VIRTUALIZED WEB-SERVER SYSTEM

In this section, a grey-box model of the target system is presented. It is a linear state space model with four state variables and a control input (explained in the following subsection). The unknown parameters of the proposed model are estimated using linear regression technique [13]. The validation of identified model is performed with real time test data. Figure 1 shows the block diagram for performance management of a virtualized web-server. A load balancer distributes the load among the existing and newly added VMs hosting the web-server. The manipulation of virtual machines is done by the proposed controllers (explained later in the paper).

2.1 Experimental Testbed

Our experiments have been conducted with two server machines with Ubuntu (Linux) as the operating system, one for emulating the client (8 core, 16 GB RAM and 3.2 GHz CPU), and the other server (called the hosting server with 8 core, 32 GB RAM and 3.2GHz CPU) for hosting web-server with a dynamic web page. Each of the VMs in the hosting server consists of XAMPP software to create a multi-tier web application. XAMPP provides Apache web-server, ProFTPD server and the MySQL server.

A synthetic workload generator tool called httperf [4] is used for emulating a large number of client requests. HAproxy [3] is used as the load balancer to distribute HTTP load across the web-servers. The two servers are connected by 1 Gbps LAN (Local Area Network).

Web-server Configuration: Apache web-server is composed of workers consisting of threads or processes. Each worker has three states: busy, wait and idle. Whenever a request arrives, it looks for workers in idle state and if they are available then it is served, otherwise it waits for a worker to become idle.

The timeout parameter gives the maximum wait time of the requests. Connections fail (not served) if they are not served before the timeout. After receiving a request, worker goes to busy state and after completion, it waits for the requests to come from same client for a time known as KeepAlive. For more on Apache web-server configuration readers may refer to [15]. Figure 2 shows the open loop response of the Apache web-server with the state variables

(explained in the next subsection). It is clear from the plot that as the workload increases the memory and CPU increases. For workload upto 2200 req/sec, the response time is below $10ms$ after which it start increasing due to saturation. Similarly, the throughput is satisfactory upto 2200 req/sec. However, as the workload increases further, the difference between the incoming requests and served requests increases (incoming requests fail beyound 2200 req/sec that are to be served). This can be observed from the throughput plot, which becomes almost flat after 2200 req/sec.

2.2 Linear state space model

Model Assumptions: The proposed state space model is developed with the following assumptions:

- The incoming HTTP requests are considered as disturbance/exogenous input to the model while the number of VMs used for serving them are considered as the control inputs.
- The relevant data for solving a set of ODEs as a CGI script is stored in the MySQL server.
- Default setting of all Apache web-server are unaltered while retries, maxconn, contimeout, clitimeout, srvtimeout parameters of load balancer are tuned appropriately.
- Values of the CPU and memory utilization obtained from each web-server hosted on the VMs are averaged.

2.2.1 State Variables

The exogenous input for our model is the HTTP workload $WOR(k)$ (requests/sec) and the control input is the number of virtual machines $VIR(k)$. The state variables of the model are the average CPU utilization $CPU(k)$ in % (of all the running VMs), memory usage $MEM(k)$ in % (of all the running VMs), response time $RES(k)$ in ms and throughput $TRPT(k)$ in req/sec.
The model is expressed as

$$x(k+1) = A\ x(k) + B\ u(k), y(k) = C\ x(k) \qquad (1)$$

where $x = [CPU\ \ MEM\ \ RES\ \ TRPT]'$, $u = [u_c\ \ u_d]' = [VIR\, WOR]'$. $x(k)$ is the state vector, u_c is the control input and u_d is the exogenous input. Since the output and state variables are the same, therefore C is the identity matrix.
The target system dynamics in state space form is given by:

$$x(k+1) = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ a_{41} & a_{42} & 0 & a_{44} \end{bmatrix} x(k) + \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \\ b_{41} & b_{42} \end{bmatrix} \begin{bmatrix} u_c(k) \\ u_d(k) \end{bmatrix}$$

where all a_{lm} and b_{ln}, where $l = 1, 2, 3, 4, m = 1, 2, 3, 4, n = 1, 2$ are to be estimated.

2.2.2 System Identification

In order to cover a larger input space (control input) the number of VMs are varied in a sinusoidal manner with changing workload. The amplitude of the sine wave is selected such that it covers the range of the possible values of the control input. We chose a sampling time of 1 second for identification, validation and control experiments. The sampling time is chosen based on our exploratory experiments which conclude that for a larger sampling time the model may not be accurate, whereas smaller sampling time introduces noise in the measurement data.

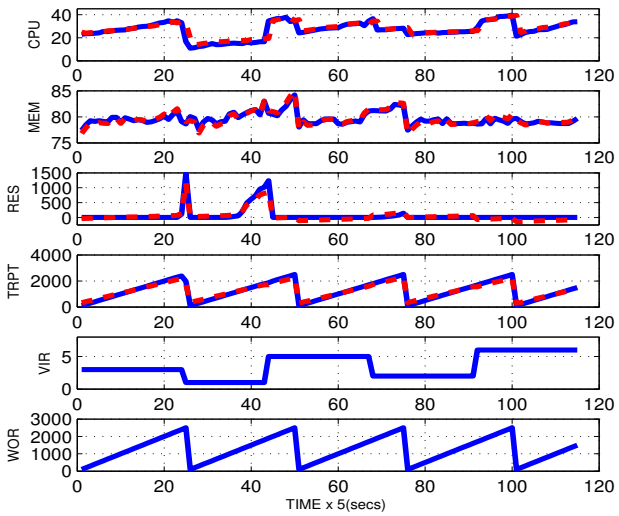

Figure 3: Validation plot: The real time data (blue or solid lines) obtained from the experimental setup is compared with the model prediction (red or dashed lines)

The state space model after parameter estimation using linear regression technique is given by:

$$A = \begin{bmatrix} 0.7309 & 0.0457 & 0 & 0 \\ -0.0003 & 0.989 & 0 & 0 \\ 2.635 & 1.152 & 0.676 & 0 \\ 0.305 & 0.121 & 0 & 0.958 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.5912 & 0.0006 \\ 0.0887 & 0.0005 \\ -37.18 & 0.0383 \\ 2.34 & 0.0437 \end{bmatrix} \qquad (2)$$

2.2.3 Model Validation

Figure 3 shows the validation plot i.e. experimental data vs model predicted data. The HTTP workload is changed in traingular manner while the VMs are changed arbitarily to observe change in state variables. It can be observed that the validation plots of the CPU, memory, the response time and the throughput closely follow the actual plots from the target system. Even though the response time behaviour is stochastic in nature, the model prediction shows good tracking with respect to the actual data.

3. STATE FEEDBACK CONTROL

In state feedback control, pole locations are selected according to required dynamic response. The objective of control is to ensure QoS (Quality of Service) i.e to keep the response time below a certain value (50 ms in our case).

In order to implement to apply state feedback control the target system represented by (1) must be controllable. The controllability matrix is given by

$$\mathcal{C} = \begin{bmatrix} B & AB & \ldots & A^{n-1}B \end{bmatrix} \qquad (3)$$

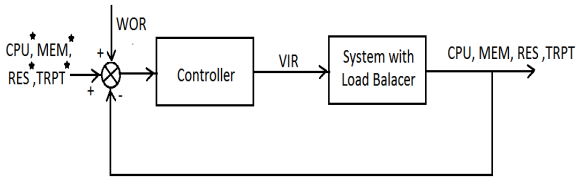

Figure 4: Block Diagram of feedback controller

where n is the number of states in the system.

A linear time-invariant system is controllable if and only if $\mathcal{C}$ is invertible [7]. The controllability matrix of the target system (2) is invertible, hence it is controllable.

3.1 Evaluation of state feedback gain

Figure 4 shows the block diagram of the closed loop system. We design a state feedback controller for web-servers for acheiving desired performance. The desired value of response time is 10 ms. The key requirement is that controller should be able to achieve this objective with changing workload. The state feedback gain is obtained by using pole placement technique for which transient specifications are chosen as: settling time of 30 sec and overshoot less than 5%. Therefore, the dominant poles according to desired specifications are $0.87\pm$ 0.12 i. The other poles are chosen as 0.7 and 0.6. The control law $u = -Kx$ provides the number of VMs to serve the requests.

The control interval is choosen to be 10 seconds, which means the number of VMs serving the requests are varied every 10 seconds according to control law.

Experimental Results: In Figure 5, the performance of the state feedback control under varying workload is shown. We observe that the CPU and the memory are bounded above within desired limits. The set value of response time is taken to be 10 ms for the state feedback controller. It can be observed that the response time stays well below the set value most of the time under changing workload.

4. FUZZY CONTROLLER

A fuzzy controller unlike traditional controllers is based on fuzzy logic/rule base. A fuzzy logic is a mathematical logic that attempts to solve problems by assigning values to an imprecise spectrum of data in order to arrive at the most accurate conclusion possible [18]. The benifits of fuzzy logic are its simplicity and its flexibility.

In the case of web-servers as the workload varies unpredictably, the response varies in a stochastic manner. If no control is applied to the system (see Figure 2 for open loop response), the performance degrades as the workload increases. Whenever workload increases abruptly, it saturates the server, during which some requests are served while the remaining requests are rejected. As the fuzzy controller is based on the expert knowledge of the target system such as the CPU and memory utilization bounds in addition to the response time (being controlled), it can serve as a better choice in comparision to the state feedback control. However, as stated earlier, it requires through knowledge of the

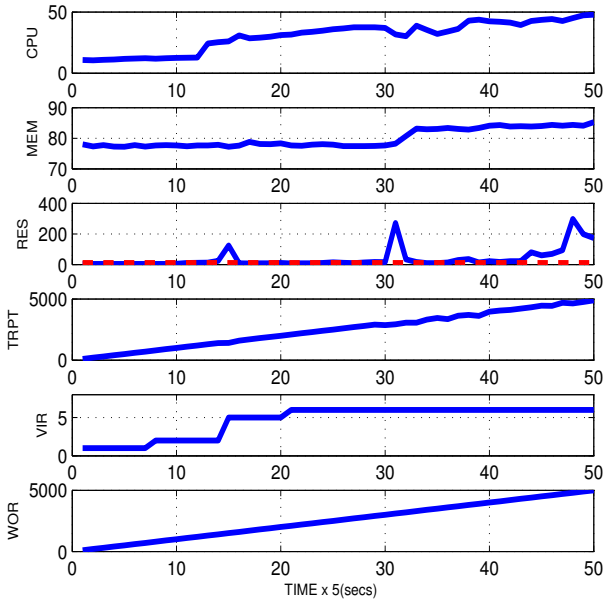

Figure 5: Performance of feedback controller. The red line(dashed) are the prediction of model and the blue lines(solid) are experimental data.

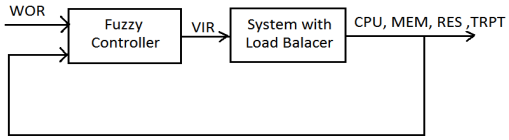

Figure 6: Block Diagram of Fuzzy Controller

target system.

In our experimental test case, the fuzzy controller is used to adjust the number of VMs according to fuzzy logic so that the desired performance is met. The fuzzy logic is based on a set of membership functions to capture the overall target system dynamics.

4.1 Design of Fuzzy controller

The inputs to the fuzzy controller are CPU utilization, memory utilization, the response time and the workload. The values of all metrics obtained at sampling instances are time averaged over control interval and are fed to fuzzy controller. The output of controller is the number of VMs given by the fuzzy rule base. One can also take input of controller as difference between the desired value and output values at sample instances and average it over control interval. As fuzzy is based on qualitative knowledge, both types of inputs are equivalent.

We use MATLAB [1] to evaluate the *if-then* rules for fuzzy controller. The block diagram of fuzzy control of system is shown in Figure 6. The *or* logic is implemented across different input variables. For example, "*if* CPU is ρ_1 *or* MEM is ρ_2 *or* WOR is ρ_3 *or* RES is ρ_4 ,*then* the number of VMs is ρ_5", where ρ_i are numerical values. A membership function

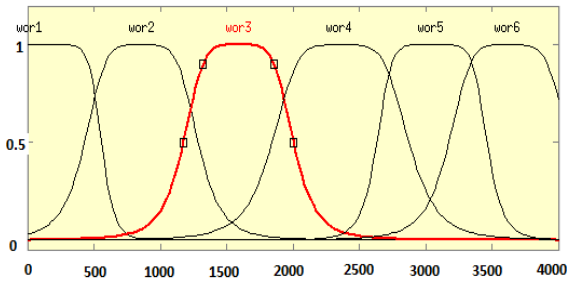

Figure 7: Membership function for workload

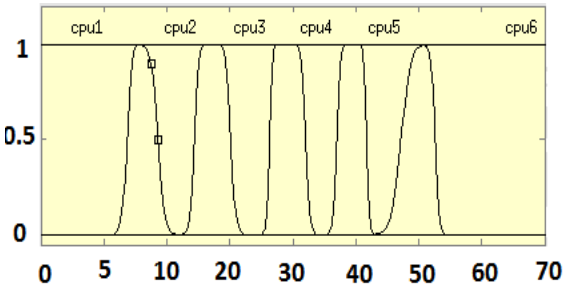

Figure 8: Membership function for CPU

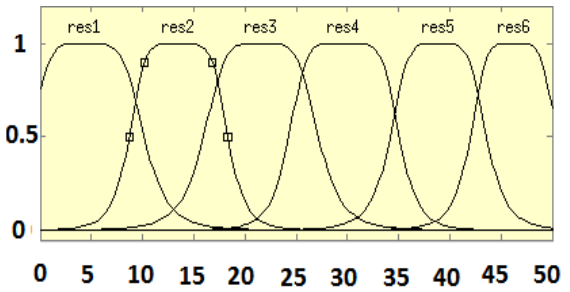

Figure 9: Membership function for Response time

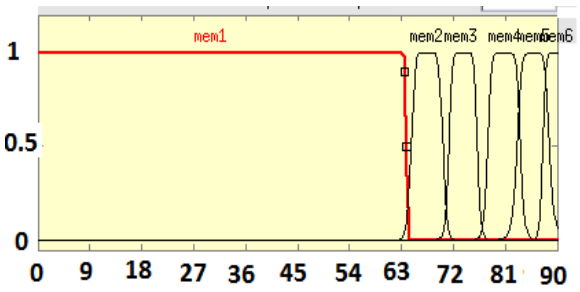

Figure 10: Membership function for Memory

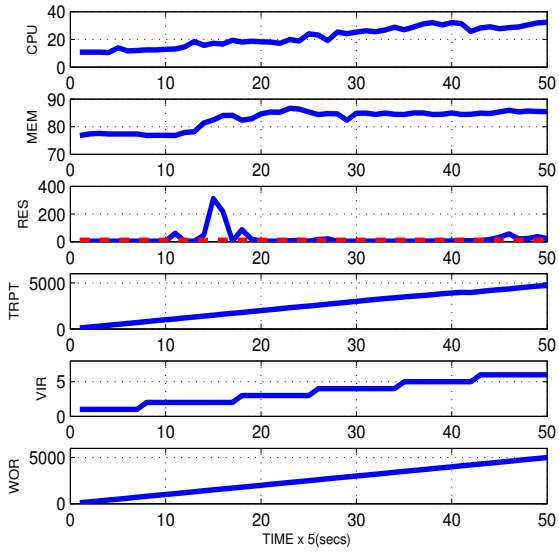

Figure 11: Performance of fuzzy controller

associated with a given fuzzy set maps an input value to its appropriate membership value. For the fuzzy controller, we have used generalized bell membership function, however traingular or trapezoidal or other membership functions can also be used. Generalized bell membership function is specified by three parameters which determine its shape. It has one more parameter than the Gaussian membership function, therefore, one more degree of freedom to adjust the steepness at the crossover points. The advantage of using this membership function is it is smooth and nonzero at all points while it has one disadvantage that it lacks asymmetry. As each VMs have an overhead memory usage, so the membership functions for the memory are distributed accordingly.

The plot of membership functions for the different metrics are shown in the figures 7, 8, 9 and 10. The output of a fuzzy controller mainly depends on the distribution of membership functions, so care should be taken while distributing them. As described earlier, the main concern is to get desired performance from the target system using sufficient number of VMs. This is achieved by assigning different weights in the *if-then* rules. Weights are assigned because our prime concern is/are one or two metrics not the all metrics,so higher weight for a metric implies more priority than others. Hence in our case RES is provided highest weight then $TRPT$ after that CPU and MEM. The fuzzy controller has the control interval of 10 seconds. This control interval is chosen because it is required that the desired properties are achieved even for random change in workload. If it is assured that workload change is not abrupt then control interval can be increased to higher value.

Experimental Results: In Figure 11, the implemention of fuzzy control with increasing workload is shown. The set value of response time is chosen to be 10 ms. We observe that as the workload increases the numbers of VMs changes based on the fuzzy rule base to achieve the desired performance. In addition, the CPU and the memory utilizations

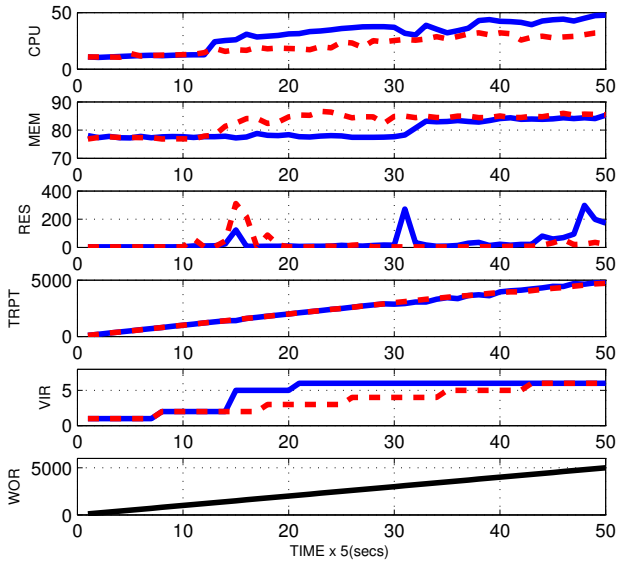

Figure 12: Comparison of State feedback and fuzzy controllers with increasing workload

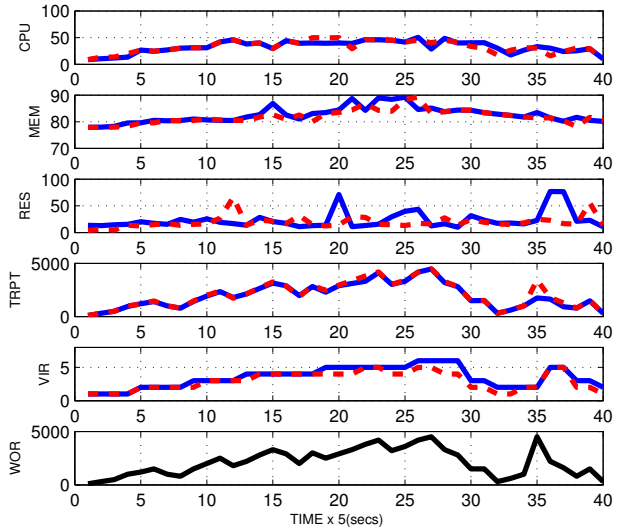

Figure 13: Comparison of State feedback and fuzzy controllers with random workload

are within the desired limits even for higher workload. It can also be observed that the response time is well below the set value most of the time under increasing workload.

5. COMPARISON OF STATE FEEDBACK AND FUZZY CONTROLLER

The implementation of the state feedback controller requires system dynamics (model) whereas for fuzzy control knowledge of system dynamics may not be required.
Figure 12 and 13 shows the comparison of fuzzy controller and state feedback controller under two different workload patterns. In Figure 12 the workload in monotonically increasing, whereas in the Figure 13, the workload is random in nature. The red coloured lines in these plots show the response due to fuzzy controller, whereas the blue coloured plots show the response due to state feedback controller.
Both the controllers provide desired performance. By observing plots, fuzzy control appears slightly better in term of utilization of CPU while the state feedback controller appears slighty better in term of usage of memory. For response time as well as for throughput both of the controller perform satisfactorily. It can be observed that fuzzy controller is good at determining the optimum number of virtual machines while providing better performance than the state feedback controller under changing workload conditions.

6. CONCLUSIONS

In this paper, we firstly develop a linear state space model of the target system using grey-box identification technique followed by model validation, secondly, we apply state feedback control based on the obtained model. A fuzzy controller is then designed based on the knowledge of the target system using fuzzy rule base. Finally, we compare the performance of state feedback controller with fuzzy controller.

In future work, we would like to extend our approach to performance control of virtualized servers over distributed network and locations, such as the cloud.

References

[1] http://in.mathworks.com/products/matlab/.

[2] https://www.apachefriends.org/index.html.

[3] http://www.haproxy.org/.

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

[5] A. Bachnas, R. Toth, A. Mesbah, and J. Ludlage. Perspectives of data-driven lpv modeling of high-purity distillation columns. In *Control Conference (ECC), 2013 European*, pages 3776–3783. IEEE, 2013.

[6] S. Blagodurov, S. Zhuravlev, and A. Fedorova. Contention-aware scheduling on multicore systems. *ACM Transactions on Computer Systems (TOCS)*, 28(4):8, 2010.

[7] J. M. Davis, I. A. Gravagne, B. J. Jackson, I. Marks, and J. Robert. Controllability, observability, realizability, and stability of dynamic linear systems. *arXiv preprint arXiv:0901.3764*, 2009.

[8] Y. Diao, N. Gandhi, J. L. Hellerstein, S. Parekh, and D. M. Tilbury. Using mimo feedback control to enforce policies for interrelated metrics with application to the apache web server. In *Network Operations and Management Symposium, 2002. NOMS 2002. 2002 IEEE/IFIP*, pages 219–234. IEEE, 2002.

[9] R. Dittner and D. Rule Jr. *The Best Damn Server Virtualization Book Period: Including Vmware, Xen, and Microsoft Virtual Server*. Syngress, 2011.

[10] IndianExpress. http://indianexpress.com/article/technology/technology-others/bigbillionday-gets-flipkart-millions-of-unhappy-customers/.

[11] L. Malrait, S. Bouchenak, and N. Marchand. Experience with conser: A system for server control through fluid modeling. *Computers, IEEE Transactions on*, 60(7):951–963, 2011.

[12] D. A. Menascé, D. Barbará, and R. Dodge. Preserving qos of e-commerce sites through self-tuning: a performance model approach. In *Proceedings of the 3rd ACM conference on Electronic Commerce*, pages 224–234. ACM, 2001.

[13] D. C. Montgomery, E. A. Peck, and G. G. Vining. *Introduction to linear regression analysis*. John Wiley & Sons, 2015.

[14] PollEverywhere. http://www.polleverywhere.com/blog/the-coca-cola-super-poll-blackout/.

[15] A. H. S. Project. https://httpd.apache.org.

[16] A. Robertson, B. Wittenmark, M. Kihl, et al. Analysis and design of admission control in web-server systems. In *American Control Conference, 2003. Proceedings of the 2003*, volume 1, pages 254–259. IEEE, 2003.

[17] P. Saikrishna, A. Chandrasekar, and R. Pasumarthy. Performance guarantees via pole placement for a web-server hosted on a private cloud. In *Control Conference (ECC), 2015 European*, pages 854–859. IEEE, 2015.

[18] J. Wei and C.-Z. Xu. A self-tuning fuzzy control approach for end-to-end qos guarantees in web servers. In *Quality of Service–IWQoS 2005*, pages 123–135. Springer, 2005.