

Fuzzy Control for Guaranteeing Absolute Delays in Web Servers

¹Yaya wei, ¹Chuang Lin, ²Thiemo Voigt, ¹Fengyuan Ren

¹Department of Computer Science, Tsinghua University,

²Swedish Institute of Computer Science

E-mail {yywei,clin}@csnet1.cs.tsinghua.edu.cn; thiemo@sics.se

Abstract

This paper presents a fuzzy control approach that guarantees absolute delays in web servers. Previous work has proposed the use of classical PI controllers for delay guarantees. However, a disadvantage of the classical PI controller is that the system model, which is obtained by system identification, mismatches the real system and inevitably degrades the performance of the web system. In contrast with classical PI controllers, fuzzy controllers are nonlinear and therefore independent of the accurate model of the plant, i.e. the controlled system. Hence, fuzzy controllers seem to be very suitable for web servers. Our experiments show that fuzzy controllers indeed perform better than PI controllers presented in earlier papers.

Key words- QoS, Resource Scheduling, Fuzzy control

1. Introduction

As the wide spread usage of web service grows, the number of accesses to many popular web sites is ever increasing. Moreover, web servers experience an extreme variation in demand ranging from little demand to an enormous increase of requests caused by the Slashdot-effect or flash crowds. Therefore, it is not feasible to provision web servers for peak load which means that even well-equipped web servers may be overloaded. During overload not all requests can be served in a timely manner. Therefore, performance-enhancing mechanisms that provide good service to premium customers even during server overload are of major importance.

Recent research has highlighted the importance of QoS differentiation in terms of delay guarantees for web servers. In [1], a PI controller has been used to provide relative and absolute delay guarantees. But system non-linearities and modeling inaccuracies are inherent problems of PI controllers [2]. To solve this problem, in [3], a queuing-theoretic prediction was used to augment the pure feedback loop. However, since predictions from queuing theory apply only to long term averages, they do not handle transient behavior very well. All these approaches try to accommodate non-linearities, but they cannot solve the problem perfectly. In this paper we propose a new algorithm that uses a non-linear controller, namely a fuzzy controller. To the best of our knowledge, this paper presents the first attempt to use a non-linear controller to guarantee absolute delays. Our approach provides the following features:

- It is independent of the accurate model of the plant, which makes it very adequate for web systems which are inherently non-linear.
- The parameters of the proposed fuzzy controller automatically adapt to various servers whereas the parameters of the PI controller must be optimized for all different kinds of servers with different speed etc.
- The performance of our proposed fuzzy controller is superior to the classical PI controller.

The rest of the paper is organized as follows. In Section 2 the bottleneck of our system is presented. In Section 3 we describe the system with the proposed fuzzy controller. In Section 4, the design of our proposed fuzzy controller is provided. Section 5 presents experimental results. Finally, in Section 6, we summarize our conclusions.

2. Bottleneck

Our control approach targets a bottleneck that is known from the Apache server, version 1.3.x. Apache

This work is supported by the National Natural Science Foundation of China (No.60429202, 90412012 and 60373013), NSFC and RGC (No. 60218003), the National Grand Fundamental Research 973 Program of China (No.2003CB314804)

deploys child processes to handle HTTP requests and a main process that always adjusts the number and the status of child processes.

Let the connection delay denote the time interval between the arrival of a connection request and the time the connection is accepted. Let the processing delay denote the time interval between the web server starts processing a request and the time the server completes the transmission of the response to the client. Lu et al. have experimentally demonstrated that the processing delay on already established connections is much smaller than the connection delay [1]. Therefore we define “delay” as the connection delay. Then our QoS goal is to provide absolute connection delay guarantees by process assignment for differentiated classes.

3. System architecture

Our system architecture is shown in Figure 1. It includes Connection Scheduler, Monitor and Fuzzy Controller modules.

- Connection Scheduler

The Connection Scheduler listens to the well-known port and accepts every incoming connection request. The Scheduler classifies requests into different classes based, for example, on the client’s IP address. A new request is allocated to a process only if the number of assigned processes for the request’s class is less than the process counter for this class.

- Monitor

The monitor carries out the measurements of the absolute delays C_k experienced by requests of class k .

- Fuzzy Controller

The proposed fuzzy controller guarantees our desired delay of 8 seconds for the priority class (class 0) by assigning enough processes p_0 to handle requests for this priority class.

For every control loop, the delay error, which is computed by the difference between the measured absolute delay C_k and the desired absolute delay (also called delay set point) DD_k , and the error rate inputs to the controller. The controller then computes the number of processes that are allocated for each class.

4. Design of the proposed fuzzy controller

Based on the mathematical model of a system, classical PI control engineering uses its inputs to design control actions or to analyze their effectiveness. This method has been used in [1]. Fuzzy control, introduced by Mamdani for the control complex process [5], does not build a mathematical model as PI controllers do.

Instead, fuzzy controllers facilitate incorporation of human expert knowledge on server tuning.

Our proposed fuzzy controller has two inputs: the delay error (e) and the change of e (Δe). The output is the regulating processes (Δp). $p_i(k)$ denotes the number of processes assigned to class i . An integrator element converts this output (Δp) into an actual value ($p_0(k)$) which denotes the number of Apache server processes dedicated for class 0 requests during the $k+1^{\text{th}}$ period.

The actions of the fuzzy controller are guided by a rule base with linguistic IF-THEN rules, so-called fuzzy rules. For example, “IF E is NB and ΔE is NB, THEN U is NB”. The terms E , ΔE and U are input linguistic variables; NB is a linguistic value. Linguistic variables take on a “degree of truth” for each possible linguistic value. This is represented as a continuous value between 0 and 1 where 0 is false, 1 is true, and 0.5 indicates we are halfway certain. We have several ways to convert the value of numeric variables into linguistic values of linguistic variables, and do the reverse as well. For example, using a triangular membership functions, we map between the numeric variable e and the linguistic value E , as shown in Fig. 2 (other functions such as Gaussian may also be used).

The objective of our fuzzy controller is to guarantee the delay of the premium class as close as possible to a pre-specified reference value. The requests from the basic class are processed in the best effort mode.

The model of the fuzzy system, comprising the control rules and the term sets of the variables with their related fuzzy sets, was obtained through a tuning process that started from a set of initial insight consideration. Then we progressively modified the parameters of the system until it reached level of performance considered to be adequate. For e , since the value domain of C_0 is $(0, \infty)$, the minimum value of $e=(C_0-DD_0)$ is $-DD_0$. Without loss of generality, assume that e varies in the section $[-DD_0, DD_0]$ and Δe varies in the interval $[-1.25DD_0, 1.25DD_0]$ that is sufficient for observing delay variance through many experiments. The intervals are chosen based on the knowledge about the process to be controlled and sometimes through trial and error to achieve the best possible controller performance. Unfortunately, there exist no well-defined methods for finding good interval boundaries.

To obtain a finer control action nearby the set point, a nonuniform distribution of the membership functions of the two inputs (see Figs. 2 and 3) and output (see Fig. 4) has been adopted. For both E and ΔE , we have chosen five fuzzy term sets, which are negative big (NB), negative small (NS), zero (Z), positive small (PS) and positive big (PB). U is divided into four more

fuzzy term sets i.e. negative huge (NH), negative medium (NM), positive medium (PM), and positive huge (PH). We defined more term sets for U since more term sets increase flexibility and enable us to make the controller react more accurately. For U , the values which are assigned to the corners of triangular membership function termed “NH” and “PH” should be large enough. If they are too small the controller might converge too slowly.

Table 1 here describes all the 25 possible control rules. To design these rules, we add the expert heuristics about our specific system.

Table 1. Fuzzy Control rules

E ΔE	NB	NS	Z	PS	PB
NB	NH	NB	NB	NM	NS
NS	NB	NM	NS	Z	PS
Z	NS	Z	Z	Z	PS
PS	NS	Z	PS	PM	PB
PB	PS	PM	PB	PB	PH

5. Experiments

We have modified the source code of Apache 1.3.9 web server which runs on a Linux platform to implement the adaptive architecture and fuzzy control algorithm. The sampling period was set to 30 sec which is adopted from [1].

All experiments were conducted on a test-bed of PCs. All PCs ran Linux-2.4.18. The servers and the clients were connected by a 10Mbps Ethernet hub. The clients generate web traffic using the Surge workload generator [4].

5.1 Latency guarantee at Overload

Each machine had a 600MHz Celeron processor and 256 MB RAM. The experimental setup is as follows.

- Server: The total number of processes was configured to 128. It is a constant number in our experiment. The timeout of connection of HTTP/1.1 was set to 15 sec, the default value. The number of processes was initially set to 40 for class 0 and 88 for class 1.
- Client: We used three client machines in this experiment. Two client machines generate requests for class 0 and one client machine for class 1. In the first 1000 sec, one client machine simulated 90 class 0 clients and one client machine simulated 180 class 1 clients. At 1000 sec, a third client machine was started simulating 30 class 0 clients.

The results of the experiments are shown in Fig. 5 and 6. Without any controller (see Fig. 5), the delay of class 0 increased from 21 sec (at time 900 sec) to 35

sec (at time 1080 sec) after the third client machine was turned on. Obviously, the server was overloaded and the delay set point of class 0 (8 sec) cannot be guaranteed.

For comparison, the classical PI controller is also implemented in our experiment. The parameters of the PI controller for our specific plant are designed by the Root Locus method, which is used in [1]. The digital form of our PI control function is

$$p_i(k) = p_i(k-1) - 4.6*(e_j(k) - 0.3*e_j(k-1)) \quad (6)$$

Fig. 6 show the PI controller exhibits more oscillations than the proposed fuzzy controller.

5.2. Adaptation to various plants

To investigate the robustness of the proposed fuzzy controller, we placed our web server on another two machines. One has only little higher speed while the other has much higher speed than the machine which has been identified exactly in the previous section. These scenarios demonstrate how e.g. a system upgrade to a faster system would affect the two types of controllers.

The configuration of the original machine was called server 0. To evaluate the performance of the two controllers, two other machines (called server 1 and server 2) with different speeds, ran the same modified Apache software with both PI and fuzzy controllers. Server 1 had a 800MHz Pentium III processor and 256 MB RAM, which is only slightly faster than server 0. Server 2 had a 1.66GHz AMD Athlon processor and 480MB RAM, i.e. it is much faster than server 0. The workload is set up as follows. The class 0 has 90 users and the class 1 has 180 users. To avoid the starting phase, the controller was activated at 180 sec. The experiment results are shown in Fig.7 and 8.

Fig. 7 shows that for server 1, the response time converges to the set point with the PI controller although the oscillation is a little larger than in server 0 in some sampling points. The proposed fuzzy controller performs well in server 1. We can conclude that PI controller still works for small machine changes (but not very well) while fuzzy controllers work very well with small machine changes.

For server 2, it can be seen from Fig. 8 that the delays in server 0 and server 2 differ significantly. The PI controller in server 2 even is not able to converge to the set point. For the fuzzy controller, the performance of server 2 and server 0 are very similar. Thus, the fuzzy controller of server 0 can also be suitable for server 2. In summary, in our experiments the PI controller does not work with very different server, while the fuzzy controller still works even for servers with very different characteristics.

In conclusion, this experiment suggests that whereas the parameters of the PI controller must be optimized for all different kinds of servers with different speed etc, our proposed fuzzy controller automatically adapts to various servers.

6. Conclusion

In this paper, we have proposed a novel fuzzy controller for web server delay control. The key characteristic of the algorithm is that it is independent of the model of the plant and provides a non-linear control action, which eliminates the control error caused by model inaccuracies. For classical PI controllers, the linear model of a nonlinear system is not accurate, and the linearization inevitably causes errors that degrade the performance of the system. In addition, our proposed fuzzy controller can adapt to various web servers with different speed. Hence, our approach does not require system identification on a per-system basis as long as the set point is the same.

References

- [1] C. Lu, T. F. Abdelzaher, John A. Stankovic and Sang H. Son. "A Feedback Control Architecture and Design Methodology for Service Delay Guarantees in Web Servers". University of Virginia *Tech Report CS-2001-06*.
- [2] M. Andersson, M. Kihl and A. Robertsson. "Modelling and Design of Admission Control Mechanisms for Web Servers using Non-linear Control Theory". *ITCom*, September 2003.
- [3] Y. Lu, T. Abdelzaher, J. Stankovic, and S. Son. "Feedback control with queueing-theoretic prediction for relative delay guarantees in web servers". In *proceedings of the 9th IEEE Real-Time and Embedded Technology and Applications Symposium*, Washington, DC, May 2003.
- [4] P. Barford and M. E. Crovella. "Generating representative web workloads for network and server performance evaluation". *ACM SIGMETRICS'98*, Madison WI, 1998.
- [5] E. H. Mamdani. "Application of fuzzy algorithm for

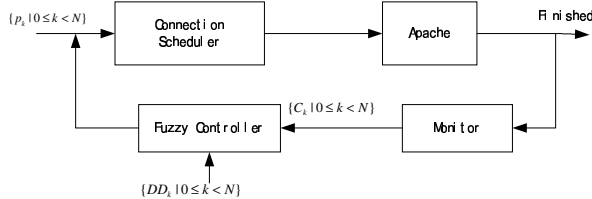


Figure 1. Architecture of the System

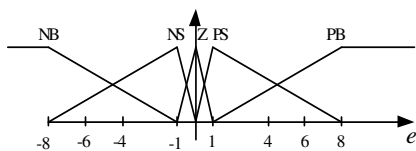


Figure 2. Membership function of E

control of simple dynamic plant", *Proc. IEEE* 121 (1974) 12.

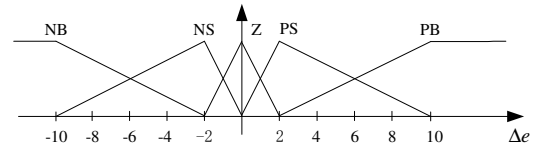


Figure 3 Membership function of ΔE

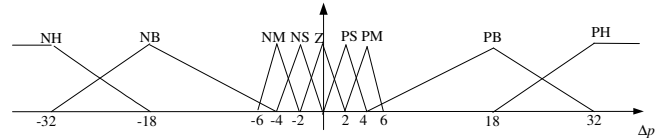


Figure 4 Membership function of U

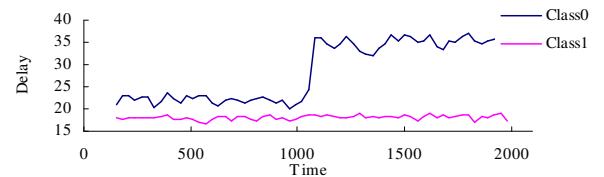


Figure 5. Delay for Class 0 and Class 1 in Open-Loop

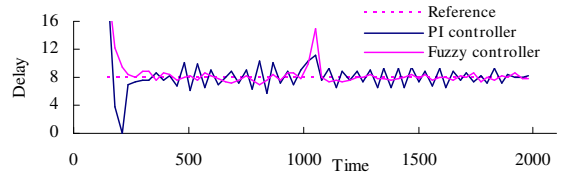


Figure 6. Delay of PI controller and fuzzy controller for Class 0

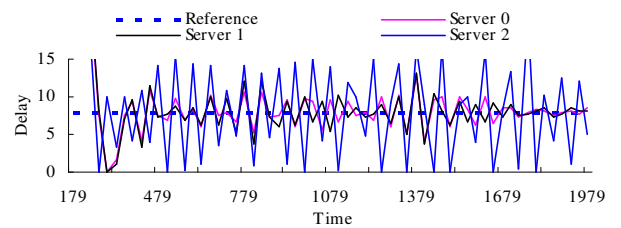


Figure 7. Delay for Class 0 with PI controller

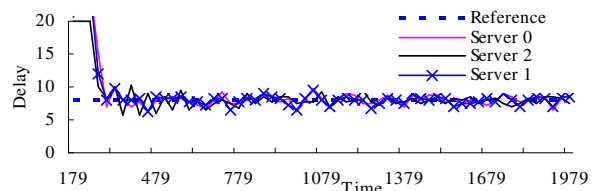


Figure 8. Delay for Class 0 with fuzzy controller