Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. VI (2011), No. 4 (December), pp. 635-646

The Research of QoS Approach in Web Servers

Y. Hu, D. Mu, A. Gao, G. Dai

Yansu Hu, Dejun Mu, Ang Gao, Guanzhong Dai
School of Automation
Northwest Polytechnical University
Xi'an 710072, China
E-mail: huyansu@gmail.com, mudejun@nwpu.edu.cn, snailgao@gmail.com, daiguanzhong@nwpu.edu.cn

Abstract: Proportional Delay Guarantee has been widely used in the Web QoS service, and the most basic methods are the feedback of control theory and the predictive control of queuing theory. While the former belonging to passive control has a long setting time and imperfect real-time, the latter can not simulate the Web server queuing system well because of the model limitations. After the experimental verification and shortages analysis of the two methods, an improved approach is proposed in this paper. Based on the queuing feature of Web server and the HTTP 1.1 persistent connection, the improved approach predicts the delay by calculating the queue length and service rate and achieves the relative delay guarantee of different classes by adjusting their quota of worker threads. The experimental results demonstrate that the approach could maintain the relative delay guarantees well even in poor network environment and performs a much better superior compared with the traditional methods.
Keywords: Web Server, Proportional Delay Guarantee, Feedback Control, Predictive Control

1 Introduction

The increasing diversity of Web applications the last decade has witnessed an increasing demand for provisioning of different levels of quality of service (QoS) to meet changing system configuration and satisfy different client requirements. Proportional delay differentiation (PDD) service aims to ensure the QoS parameters between data flow of different classes to meet the specified proportions, so the requests with higher priority will receive the quality of service which is "at least not lower than" the low priority. There are many previous studies worked on the Web QoS and two different methods are used to achieve the differentiated service: the feedback of control theory (see [1], [2], [3], [4] and [5]) and predictive control of queuing theory (see [2], [6] and [7]). Although the algorithms are different, the QoS architecture of Web server they used is non-distinctive.

The QoS of Web application is usually related to the network layer transmission and the operating system kernel. The latter that needs to replace the operating system on all the terminals is difficult to deploy, so more and more researchers tend to the differentiation service on the application layer. It ensures the PDD service by changing the original Web FIFO mechanism and the modified Apache MPM (Multi-Processing Modules) architecture is illustrated in Figure 1.

1. The single connection queue is improved to a multi-queue structure in accordance with the classified strategy. The listener monitors the network port, accepts the client TCP connections, classifies the requests based on some classified strategy, and then puts them into the appropriate waiting queue.

2. Any class can not consume the server resource unlimitedly, so the requests of different classes must be isolated. The thread per connection structure of MPM allows all the worker threads in pool to be the resource to be allocated. So we divide the pool into several sub-pools which are isolated with each other. The number of threads in each sub-pool is known as the thread quota, and the requests are serviced in the corresponding sub-pool.

3. For N kinds of classes, let $c_i(i = 1, ..., N)$ be the thread quota allocated to the class i. When load varies, the size of sub-pool is dynamically adjusted to ensure the proportion between classes constant. And is calculated by feedback control used the delay observer to get the real delay (Figure 1 ①method), or by predictive control used the queue length observer to get the request arrival rate(Figure 1 ②method).

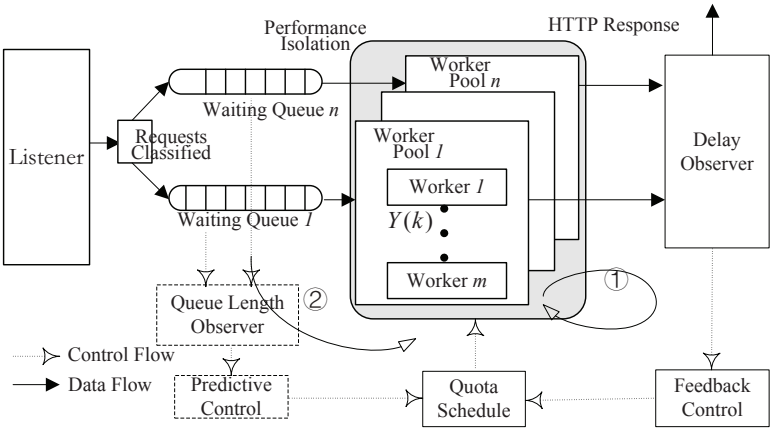


Figure 1: The Web server architecture for PDD service

But there are some imperfections and limitations in both the methods. For example, the feedback control is essentially a passive approach which works only after the deviation appeared. The delay in response is significant since the server response time is particularly slow. Although the predictive control based on queuing theory adjusts the controller output in advance, it can not get the precise plant model because of limitations itself. So an improved predictive approach is proposed to overcome their shortages and the contributions in this paper can be summarized as follows:

1) A general architecture for proportional delay service in Web server is summed up.
2) The two methods referred above are verified and compared by experiments.
3) Implement and evaluate an improved approach to get a better performance.

2 The Comparison of Two PDD Methods

2.1 The Feedback of Control Theory

Assuming C worker threads are concurrent on the Web server. For N kinds of classes, let δ_i be the constant weighting factor of class i where the higher priority has a smaller parameter value, and d_i be the expectation of class i's average measured delay, then the proportional delay guarantees can be described as follows:

$$d_i/d_j = \delta_i/\delta_j, 1 \le i \le N, 1 \le j \le N, \quad (1)$$

$$\sum_{i=1}^{N} c_i = C \tag{2}$$

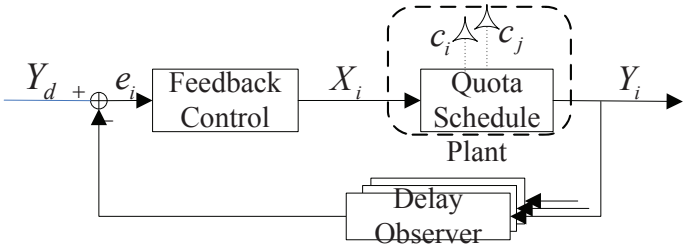


Figure 2: The feedback control model

Where c_i is the thread quota of class i. So the improved MPM is equivalent to a control model as Figure 2. The system desired output is the inherent priority parameter ratio between class i and $i+1$, which is

$$\mathbf{Y}_{\text{desire}} = [y_{1desire}, y_{2desire}, ..., y_{N-1disire}]^{\mathrm{T}}. \tag{3}$$
$$y_{idesire} = \delta_i/\delta_{i+1}, i = 1, 2, ..., N-1. \tag{4}$$

Similarly, the controller output is the thread quota ratio and $\mathbf{Y}(\mathbf{k})$ is the measured output ratio between the class i and $i+1$, which is

$$\mathbf{X}(k) = [x_1(k), x_2(k), ..., x_{N-1}(k)]^{\mathrm{T}}. \tag{5}$$
$$x_i(k) = c_i(k)/c_{i+1}(k), i = 1, 2, ..., N-1. \tag{6}$$
$$\mathbf{Y}(k) = [y_1(k), y_2(k), ..., y_{N-1}(k)]^{\mathrm{T}}. \tag{7}$$
$$y_i(k) = d_i(k)/d_{i+1}(k), i = 1, 2, ..., N-1. \tag{8}$$

Obviously, the deviation is $\mathbf{E}(k) = \mathbf{Y}(k) - \mathbf{Y}_{desire}$. On the Web server, all classes share a single host resource. So the feedback controllerďż˝ďż˝class-per-loopďż˝ďż˝adjusts $\mathbf{X}(k)$ to satisfy Equation (1) by the measured $\mathbf{E}(k)$.

The Web server is modeled as a linear system, i.e. a r-order difference equation is

$$\mathbf{Y}(k) = \sum_{l=1}^{r} [a_l \mathbf{Y}(k-l) + b_l \mathbf{X}(k-l)]. \tag{9}$$

Where a_l and b_l are the unknown parameters. The transfer function in z-domain is

$$\mathbf{G}(z) = \frac{\mathbf{Y}(z)}{\mathbf{X}(z)} = \frac{\sum_{l=1}^{r} b_l z^{r-l}}{z^r - \sum_{l=1}^{r} a_l z^{r-1}}. \tag{10}$$

The plant's mathematical model (order and parameters) is obtained by system identification which is presented by the authors in [8] in detail. Then we can design the controller based on the classical control theory. Take the PI control for example, the controller transfer function in z-domain is

$$D(z) = K_p + \frac{K_I T(z+1)}{2(z-1)}, \tag{11}$$

And the Closed-loop system transfer function is

$$G_C(z) = \frac{D(z)G(z)}{1 + D(z)G(z)}. \tag{12}$$

2.2 The Predictive Control of Queuing Theory

Although the feedback maintains the system at the balance, it takes an imperfect real-time for the lagging output. So [2, 6, 7] try to correct the deviation before the system output being affected with the help of queuing theory. In the predictive control, the queue length observer (see Figure 1②) measures the request arrival rate λ_i and service rate μ_i. Then the predictive controller reallocates the thread quota c_i for respective class.

M/M/1/∞ Queuing System

The Web server performance mainly subjects to the system bottlenecks(see [7]), so each class in Apache can be considered as a M/M/1/∞ queuing model. Define $\rho = \lambda/\mu$ be the system traffic intensity. And the residence time is the sum of queuing time (connecting time) and service time (transfer time), which is

$$l = \omega + \chi \tag{13}$$

Where ω and χ are independent from each other. According to paper [2], the mean residence time is calculated by Equation (14).

$$\bar{l} = \overline{\omega} + E[\chi] = \frac{\rho}{\mu(1-\rho)} + \frac{1}{\mu} = \frac{1}{\mu - \lambda}, (\rho < 1) \tag{14}$$

To meet the proportion relationship specified by Equation (1), we can get

$$\frac{\mu_j(k) - \lambda_j(k)}{\mu_i(k) - \lambda_i(k)} = \frac{\delta_i}{\delta_j} \tag{15}$$

While normalized the thread quota

$$s_i = c_i/C_i, 1 \le i \le N, \sum\nolimits_{i=1}^{N} s_i = 1, \tag{16}$$

Hence, for class i, the service rate can be rewritten as

$$\mu_i(k) = \mu s_i(k) \tag{17}$$

Where μ is the total service capacity. Then Equation (14) becomes $\overline{l_i} = \frac{1}{\mu s_i - \lambda_i}$. Combined with Equation (15),

$$\begin{cases} \dfrac{\mu s_2(k) - \lambda_2(k)}{\mu s_1(k) - \lambda_1(k)} = \dfrac{\delta_2}{\delta_1} \\ \dfrac{\mu s_3(k) - \lambda_3(k)}{\mu s_2(k) - \lambda_2(k)} = \dfrac{\delta_3}{\delta_2} \\ \ldots \\ \dfrac{\mu s_N(k) - \lambda_N(k)}{\mu s_{N-1}(k) - \lambda_{N-1}(k)} = \dfrac{\delta_N}{\delta_{N-1}} \\ \\ \sum_{i=1}^{N} s_i = 1 \end{cases} \tag{18}$$

Solving equations at each sampling time, we can get the thread quota for different classes $(s_1, s_2, ..., s_N)$.

M/G/1/∞ Queuing System

In M/M/1/∞ model, the service time obeys the memoryless exponential distribution. But when it is general distribution, the correlation between the current and several previous service time must be considered (see [9]). The paper [6, 10] described the "heavy tail" features of Web service time series $\{\chi_n, n \geq 1\}$, and gave a more compatible queuing model M/G/1/∞ for the Web server. According to the Pollaczek-Kinchin(PK) formula and lemma, if s_i is the normalized thread quota for class i, service time χ_i obeys the bounded pareto distribution, the mean queuing time is

$$m_{1,i} = E[\chi_i] = m_i/s_i, m_{2,i} = E[\chi_i] = m_2/{s_i}^2 \tag{19}$$

Hence the mean residence time is

$$\begin{aligned} \overline{l_i} &= \overline{\omega} + E[\chi_i] = \frac{m_{2,i}\lambda_i}{2(1-\lambda_i E[\chi])} + \frac{m_1}{s_i} = \frac{m_{2,i}\lambda_i}{2(1-\lambda_i E[\chi])} + \frac{m_1}{s_i} \\ &= \frac{m_2\lambda_i}{2s_i(s_i - m_1\lambda_i)} + \frac{m_1}{s_i} \end{aligned} \tag{20}$$

Where m_1 ,m_2 are the constants related to shape parameter of bounded pareto distribution, λ_i is the requests arrival rate which is obtained by the queue length observer. Combined Equation(20) with Equation (1),

$$\frac{s_i(s_i - m_1\lambda_i)}{s_{i+1}(s_{i+1} - m_1\lambda_{i+1})} \frac{2m_1 s_{i+1} + \lambda_{i+1}(m_2 - 2m_1^2)}{2m_1 s_i + \lambda_i(m_2 - 2m_1^2)} = \frac{\delta_{i+1}}{\delta_i} \tag{21}$$

Solve the equations similar to Equation (18) and get the results $(s_1, s_2, ..., s_N)$.

2.3 Experimental Results

The test-bed consists of a Web sever and two client machines, each with a 3.0GHz Pentium 4 professor and 521MB RAM connected with 100 Mbps Ethernet. The Web server is Apache 2(Httpd-ver2.0.53) running on Windows NT and the total number of server processes is configured to 100. The two client machines run Liunx-2.6.27 with SURGE (see [11]) (ver 1.00a) as the workload generator and each operates 120 concurrent UE. Requests are classified according to their source IP address. All the experiments are under HTTP1.1 pipeline and the number of maximum concurrent clients in SURGE is 1. Set $\delta_1/\delta_1 = 1/2$,which means the delay of high class is half of the low one. The sampling period is set to 10 sec, and all the experiments last 3000 sec. 300 valid data is measured and the controller works after 750 sec.

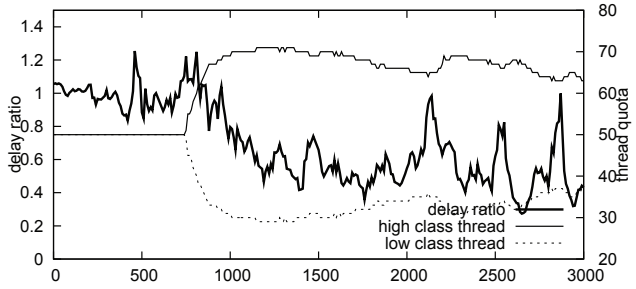
The results under PI controller and predictive controller M/G/1/∞ are respectively shown in Figure 3. Obviously, before the controllers works there is no differentiated service in different classes. But after 750 sec, the thread quota for high class increases until the expectation is satisfied. And all the methods achieve the proportional delay guarantees well.

In order to compare the features of the two methods, the mean value of the measured delay ratio l_1/l_2 and the variance relative to $\delta_1/\delta_2 = 0.5$ are defined as follows:

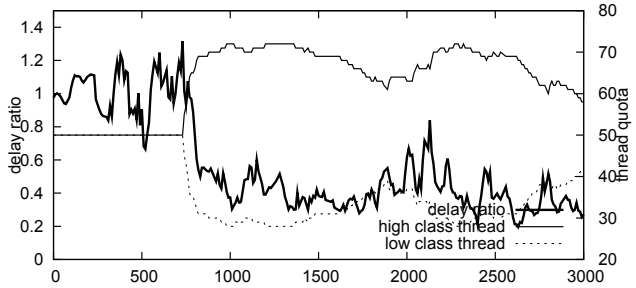
$$E_e = \frac{\sum_{k=75}^{300} l_1(k)/l_2(k)}{300 - 75} \tag{22}$$

$$D_e = \sum_{k=75}^{300} [l_1(k)/l_2(k) - \delta_1(k)/\delta_2(k)]^2 \tag{23}$$

Compute Equation (22) and (23), we get $E_e = 0.5198$,$D_e = 4.56$ under the PI controller,while $E_e = 0.4136$,$D_e = 5.5376$ under the M/G/1/∞ predictive controller.Now we can see



(a) PI control



(b) M/G/1/∞ queuing predictive control

Figure 3: The change of delay ratio and thread quota under different controllers

1. As a passive control, the PI controller works until the deviation appeared. Although the output is stabilized in the vicinity of the expectation value, there is a long setting time $t_s \approx 300s$.

2. In contrast, the latter with a much better setting time $t_s \approx 40s$ only maintains the delay proportion at 0.4. This is because the queuing theory based on PK formula asks for the statistics balance which means the requests arrival rate is equal to service rate. So the model limitation leads to the ineffaceable deviation.

3 An Improved Predictive Control

From Section 2, the queuing theory is invalid when the Web loads bursts and fluctuates dramatically. Meanwhile, for the Web queuing process, the number of requests in the front of request j can be measured. On this issue, an improved predictive control is proposed in this paper to achieve the relative delay guarantees.

3.1 Design

The relationship of the requests arrival rate, service rate and the queue length of class i are demonstrated in Figure 4(see [12]).The arrival curve $Arrvial_i(k)$ presents the rate of TCP connections, and the service curve $Service_i(k)$ presents the capacity of Web server for class i.

$$\frac{\Delta Arrival_i(k)}{\Delta k} = \lambda_i(k) \tag{24}$$

$$\frac{\Delta Service_i(k)}{\Delta k} = \mu_i(k) \tag{25}$$

When class$i$ is overload, the arrival rate is always greater than the Web service rate,i.e. $\lambda_i(k) > \mu_i(k)$.So the vertical distance between the two curves is the actual length $n_i(k)$ of waiting queue i at the k^{th} sampling time, which is $n_i(k) = Arrival_i(k) - Service_i(k)$. From the view of the tail, the waiting time $w_{i,n_i-1}$is the sum of the processing time of the top $(n_i - 1)$ requests. Considered the processing time χ_{i,n_i}itself, the mean residence time is

$$l_{i,n_i} = w_{i,n_i-1} + \chi_{i,n_i} = \sum_{j=1}^{n_i} \chi_{i,j} \tag{26}$$

If the mean residence time $l_i(k+1)$ can be predicted as $\widetilde{l_i}(k)$ based on the queue length $n_i(k)$, we can adjust the thread quota to satisfy Equation (1).

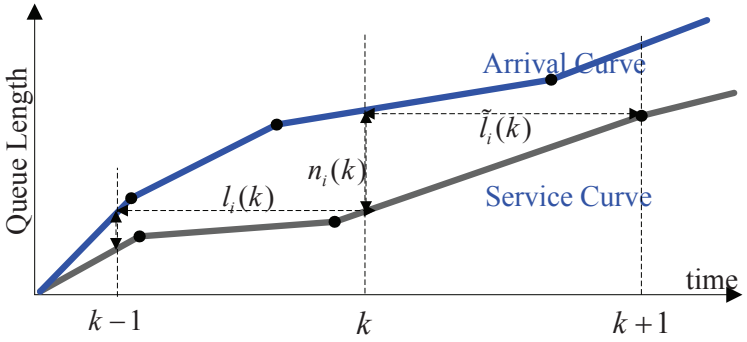


Figure 4: The arrival curve and service curve

3.2 Dealy Prediction

The Web load is actually the superposition of lots of ON/OFF process(see [11]). Just like Figure 5, Web pages are transferred during Object, and each page is composed of several embedded files which are transferred in a single TCP persistence connection under HTTP 1.1. The intervals between every embedded URL are Active OFF time corresponding to the processing time spent by browse parsing Web page. The Inactive OFF time is a longer pause corresponding to clients' "thinking procedure". Therefore, the request of a Web page consists a sequence of HTTP requests with self-similarity, which is

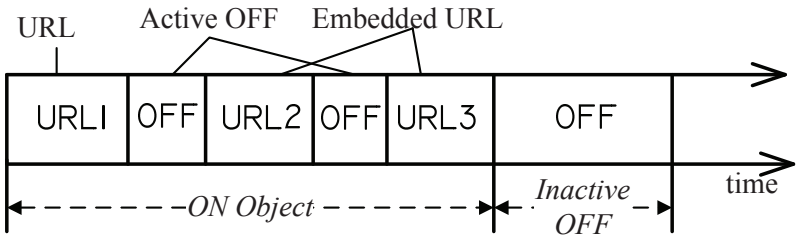


Figure 5: HTTP1.1 ON/OFF model

$$s_{i,n} = \sum_{j=1}^{y_{i,n}} x_{i,j} n = 1, 2, ...n_i(k) \tag{27}$$

Where $s_{i,n}$ means the size of the n^{th} Web page in waiting queue i, $x_{i,j}$ is the size of embedded files, and $y_{i,n}$ is the number of its embedded files. The previous research indicates that the heavy-tailed distribution is more accuracy for Web page (see [13] and [14]),and the probability mass function is:

$$f(x) = \frac{\alpha k^{\alpha}}{1-(k/p)^{\alpha}} x^{-\alpha-1}, k < x < p \tag{28}$$

Where k and p are the size of minimum and maximal file,$\alpha(0 < \alpha < 2)$ is the sharp parameter which determine the variability of distribution. Generally set $1 < \alpha < 2$to correspond to the actual communications. The number of embedded files $y_{i,n}$ follows Pareto distribution (see [11]).$y_{i,n}$ and $x_{i,j}$,$x_{i,j}$ itself are all independent from each other. The mathematical expectation of page sizes$_i$is

$$E[s_i] = E[y_i]E[x_i], i = 1, 2, ...N \tag{29}$$

Let $\{s_{i,1}, s_{i,2}, ..., s_{i,n_i(k)}\}$be the size of HTTP requests on a TCP connection at the k^{th}sample. Then the total size of the page in waiting queue i to be transferred is

$$\Delta_i(k) = \sum\nolimits_{n=1}^{n_i(k)} s_{i,n} = \sum\nolimits_{n=1}^{n_i(k)} \sum\nolimits_{j=1}^{y_{i,n}} x_{i,j} \tag{30}$$

This means that the forecasting delay is the time spent by $c_i(k)$ threads concurrently processing the files of size $\Delta_i(k)$.

The paper [15] illustrates that the processing time of a static Web page is approximately linear to the size of request files. Combined the condition that the unit of the operation system is 64KB, the total size of the page to be transferred is

$$\chi(s, c_i) = b \times (s/64) + d \times c_i \tag{31}$$

Where b is the transmission coefficient, d is the overhead of threads switching and synchronization. They can be obtained by linear regression. So the total delay predicted is:

$$l_i(n_i(k), c_i(k)) = \sum\nolimits_{n=1}^{n_i(k)} \chi_i(s_{i,n}, c_i(k))/c_i(k) \tag{32}$$

$$\widetilde{l_i}(k) = E[l_i(n_i(k), c_i(k))] = \frac{n_i(k)}{64}(\frac{bE[s_i]}{n_i(k)} + 64d) \tag{33}$$

3.3 Threads Dispatch

Combined Equation (33) with Equation (1):

$$\frac{\widetilde{l_i}(k)}{\widetilde{l_{i+1}}(k)} = \frac{\delta_i}{\delta_{i+1}} = \frac{n_i(k)c_{i+1}(k)}{n_{i+1}(k)c_i(k)} \frac{bE[s_i] + 64dc_i(k)}{bE[s_{i+1}] + 64dc_{i+1}(k)} \tag{34}$$

Regardless the priority, Web pages stored in server is identical. So we have $E[x_i] = E[x_{i+1}]$. If $Z_i(k) = n_i(k)/n_{i+1}(k)$,$e = bE[s_i]$,$f = 64d$,Equation (34) can be rewritten as

$$\frac{\delta_i}{\delta_{i+1}} = Z_i(k) \frac{fc_i(k)c_{i+1}(k) + ec_{i+1}(k)}{fc_i(k)c_{i+1}(k) + ec_i(k)} \tag{35}$$

Where $Z_i(k)$ is obtained by the queue length observer, and Equation (35) can be written into the form of $c_{i+1} = F_i(c_i(k), Z_i(k))$. Take class 1 be the reference ($\delta_1 = 1$).Considered the Equation (2), the number of threads for each class is solved.

3.4 Parameter Regression

As described in Section 3.2, the parameters b and $d$in Equation (31) are calculated by the Least Square algorithm. The sampling period is set to 10 sec and the parameter regression experiments totally obtain 61 group valid data. Figure 6 presents the pages size, the predict delay and the measured delay when the thread quota varies. The identification results are $b = 9.7$,$d = 55.7$ under 0.05 significant level.

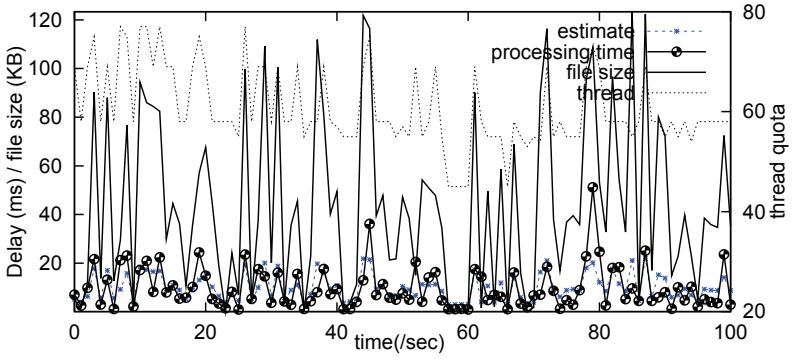


Figure 6: Parameter regression

3.5 Experimental Results

The test-bed has been shown in the section 3.3. Request rate of class1 varies from 6/s to 15/s every 300 seconds and class 2 keeps 12/s. The sampling time is $T = 10s$ and all the experiments last 2000s. The reference value is 0.5, i.e. $\delta_1/\delta_1 = 0.5$. The thread quota is equivalent for each class in the initial state. The results under different controllers are shown in Figure 7. Each sub-picture has two insets. The upper is the request arrival rate of two classes and their delay proportion. The lower presents the change of their thread quota. Compared the controllers, the results are in accordance with what we analyzed before:

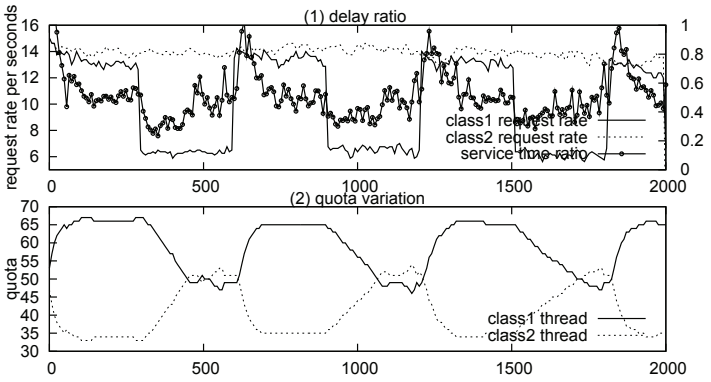
1. Whatever using feedback, $M/G/1/\infty$ predictive control or improved predictive control, they all achieve PDD in Web server. The experiments also did at sampling time $T = 8s$ and $T = 15s$ which are similar. So In order to save space, we only give the best effect figure at $T = 10s$.

2. Define the variance Ξ be the stability evaluation index ,which is

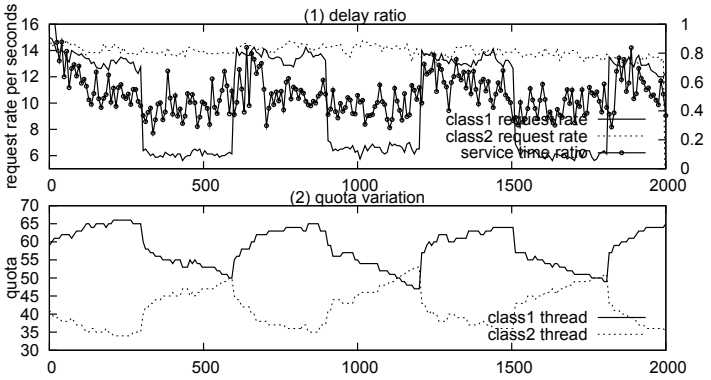
$$\Xi = \sqrt{\sum_{k=1}^{n} (y_i(k) - y_{1d})^2/n} \tag{36}$$

   Compute it under the different methods respectively, we find that the variance of proportional delay to 0.5 in improved predictive controller and $M/G/1/\infty$ predictive controller is 40.18% and 57.4% of which in PI controller. This means the predictive controller is more stable when the load changes.

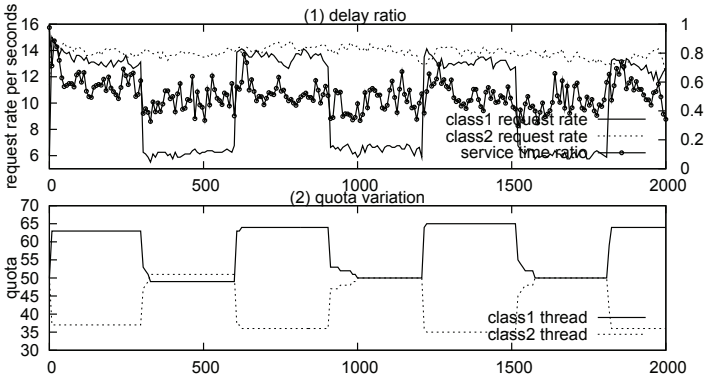
3. Compared the setting time in Figure 7a,7b and 7c. For the feedback, the setting time is nearly 100s at rising edge and 150s at the falling edge. The $M/G/1/\infty$ predictive controller is almost the same situation but a little shorter. But those of the improved predictive controller are less than 1s at both edges, which presents a much better dynamic



(a) PI control



(b) M/G/1/∞ queuing predictive control



(c) Improved predictive control

Figure 7: The change of delay ratio and thread quota under different controllers

performance. There are two reasons for this. One is the load changes influence waiting queue length first, and then this leads to the delay changes observed which exists delay. The other is the refreshed controller output works in the next sample time, while both the predictive controllers adjust the quota early.

4. Finally, compared how they adjust the thread quota when load changes in Figure 7b and 7c. Because of the limitation of PK formula, it takes a long time to adjust the quota to the steady state in Figure 7b. But the improved predictive controller can complete the

procedure in just "one step".

4 Conclusion and Future Work

The paper first presents a general Web QoS architecture, and then analyzed the defects of two traditional differentiated methods by experimental results. Finally, based on the queue length observer and parameter regression, improved predictive controller is proposed and produces a much better performance in real-time and stability.

But there are still some problems. For example, if the parameters b and d obtained online, it will be more accurate to the Web model. Meanwhile, the paper only talked about the static requests and is not compatible to the dynamic loads. In the future, we should also break the local area network, and take the network congestion into account.

Bibliography

[1] J. Wei, C. Xu, eQoS:Provisioning of client-perceived end-to-end qos guarantees in web servers. *IEEE Transactions on Computers*, Vol.55, No.12, pp.1543-1556,2006.

[2] Y. Lu, T. Abdelzaher, C. Lu, L. Sha, X. Liu, Feedback control with queuing theoretic prediction for relative delay guarantees in web servers, *The 9th IEEE Real-Time and Embedded Technology and Applications Symposium*, Washington, DC., USA,pp.208-217,2003.

[3] K.H. Chan, X. Chu, Design of a Cluster-Based Web Server with Proportional Connection Delay Guarantee, *The IEEE International conference on Communications*, Beijing,China, pp.5692-5696,2008.

[4] W. Pan, D. Mu, H. Wu, Q. Sun, Proportional Delay Differentiation Service in Web Application Servers: A Feedback Control Approach, *International Journal of Intelligent Information Technology Application*, Vol.1, No.1,pp.37-42,2008.

[5] Y. Lu, T.F. Abdelzaher, A.Saxena, Design, implementation, and evaluation of differentiated caching services, *IEEE Transactions on Parallel and Distributed Systems*, Vol.15, No.5, pp.440-452,2004.

[6] J. Wei, X. Zhou, C. Xu, Robust processing rate allocation for proportional slowdown differentiation on Internet servers, *IEEE Transactions on Computers*, Vol.54, No.8,pp.964-977,2005.

[7] L.Sha, X. Liu, U.Y. Lu, T. Abdelzaher, Queueing model based network server performance control, *The Proceedings Real-Time Systems Symposium*, Austin, USA,pp.81-90,2002.

[8] C. Lu, T. Abdelzaher, J. Stankovic, S. Son, A Feedback Control Approach for Guaranteeing Relative Delays in Web Servers, *The 7th Real-Time Technology and Applications Symposium*, Taipei, Taiwan,pp.51-62,2001.

[9] Y. Tang, X. Tang, *The queuing theory foundation and analysis*, Science Press. Beijing. 2005.

[10] X. Zhou, J. Wei, C.Z. Xu, Processing rate allocation for proportional slowdown differentiation on Internet servers. *The 18th International Parallel and Distributed Processing Symposium*, Santa Fe, USA, pp.1247-1256,2004.

[11] P. Barford, M. Crovella, Generating Representative Web Workloads for Network and Server Performance Evaluation, *Proceedings of the 1998 Joint International Conference on Measurement and Modeling of Computer Systems*, Madison, USA,pp.151-160,1998.

[12] J. Liebeherr, N. Christin, JoBS: Joint buffer management and scheduling for differentiated servers, *Proceedings of the 9th International Workshop on Quality of Service*, Karlsruhe, German,pp.404-418,2001.

[13] M. Harchol-Balter, Task assignment with unknown duration, *Journal of the ACM*, Vol.49, No.2, pp.260-288,2002.

[14] M. Arlitt, T. Jin, A workload characterization study of the 1998 world cup web site, *IEEE network*, Vol.14,No.3, pp.30-37,2000.

[15] TF Abdelzaher, N. Bhatti, Web server QoS management by adaptive content delivery, *Proceedings of the 7th International Workshop on Quality of Service*, London,UK, pp.216-225,1999.

[16] A. Gao, D. Mu, Y. Hu, W. Pan, Proportional Delay Guarantee in Web QoS Based on Predictive Control, *Proceedings of iCISE 2009*, Nanjing,China,pp.173-176,2009.