INT J COMPUT COMMUN, ISSN 1841-9836
Vol.7 (2012), No. 4 (November), pp. 661-673

The Research of Differentiated Service and Load Balancing in Web Cluster

A. Gao, Q. Pan, Y. Hu

Ang Gao, Quan Pan,Yansu Hu
School of Automation
Northwest Polytechnical University
Xi'an 710072, China
E-mail: gaoang@nwpu.edu.cn,
quanpan@nwpu.edu.cn, huyansu@gmail.com

Abstract: Differentiated service, as a key solution to meet the heterogenicity of Web clients' QoS requirements, has been widely used to optimize the server utilization without over-providing resources. Based on the relative differentiated service, this paper treats the application of proportional delay as a optimal control problem, and focuses on the cluster-side architecture improvement as well as QoS controller design. A load balancing Web cluster architecture supported differentiated service is proposed and implemented. By system identification and resource optimal control, the front-end dispatcher could adjust the resource quotas assigned to different classes in every single back-end server, and Multi-class based Maximum Idle First load balancing strategy is designed to ensure a fair resource consumption among back-end nodes. As a result, the end-to-end delay is controlled and proportional delay is guaranteed. The experiments demonstrate that no matter using Round-Robin, Least Connection Scheduling or Maximum Idle First load balancing strategy, the proposed resource optimal controller could hold the relationship among different classes. Compared to Round-Robin and Least Connection First Scheduling, Maximum Idle First strategy increases the cluster throughput by 33% and reduces the average delay by 21%.
Keywords: Differentiated Service, Maximum Idle First, Load Balancing, Proportional Delay Guarantee

1 Introduction

With the dramatic explosion of online information, the Internet is undergoing a transition from a data communication infrastructure to a service intergraded utility. The increase of Web applications, Web clients and HTTP requests on the Internet makes the Web server systems often suffer from huge pressure of heavy workload. The Deployment of Web cluster system keeps increasing to meet the demand for availability, scalability and stability of the diversified performance demands of clients.

Web cluster organizes a number of Web servers as a logical entirety to enhance the storage and processing capacity. The cluster also shows a good expansibility, whose capability can be easily tuned by changing the number of back-end server, which are connected by high speed local area network. Clients' HTTP requests are well-proportioned and transparently dispatched to back-end. Server nodes work concurrently and the responsiveness as well as the reliability of Web sites are improved (see [17]).

To take full use of every server's processing resources, a major issue is how to arrange each server node appropriate requests according to its capability. However, the cluster system scale is limited by the financial cost of Web sits and IDC (Internet Digital center), and the load characteristics of Web sites is often affected by the browsing habits, geographic distribution and breaking news. It is impossible to accurately predict the peak load and prepare enough

computing resources. So it is not cost-effective to allocate excessive computing resources for a Web site to accommodate the potential peak. Even the large-scale clusters, there will still be the case of overload. Now, the Internet has become a commercial product, the growth of e-commerce is creating demand for services with financial incentives for the service providers, i.e., the economic transaction is more important than a simple browsing, and the premium users also expect better quality services. So, the other major problem Web cluster system facing is how to meet the Service Level Agreements(SLAs) with their clients without excessively over-provisioning resources (see [21]).

As an initial effort, a feedback control mechanism is designed to achieve Proportional Delay Differentiation Service and Load Balancing (PDDS-LB) in a Web cluster system. First, according to the general Web cluster system framework and the HTTP processing procedure, a load balancing Web cluster architecture is proposed. Second, considering the residue delay[1] is the main factor affecting the users' experience in a Web application, with the aid of system identification and optimal control, we design a feedback controller, which periodically re-allocates the processing resources to keep the residue delay ratio around the set point. Finally, we present Multi-class based Maximum Idle First load balancing strategy(MIF) to achieve efficient and fair resource consumption. Experiment results show that our mechanism is effective, the total throughput increases by 33%, and the average delay reduces by 23%.

2 The Overview of Cluster

2.1 The Architecture and Forwarding Technology

Figure 1 gives the framework of Web cluster.The front-end is called *Dispatcher*, which is the entrance of the cluster system. The clients' HTTP requests first reach dispatcher, then are distributed to back-end servers according to the load balancing strategy. The the dispatcher can select a back-end according to the Request-URL in HTTP Message and/or other information in entity-header fields, such as *User-Agent*, *From*, *Host*, etc. According to the layer of dispatcher, clusters can be divided into three types (see [17]): *L4/2 Cluster-L4 Switcher,L2 Forwarding*; *L4/3 Cluster-L4 Swithcer,L3 Forwarding*; *L7 Cluster- Application Layer Forwarding.*

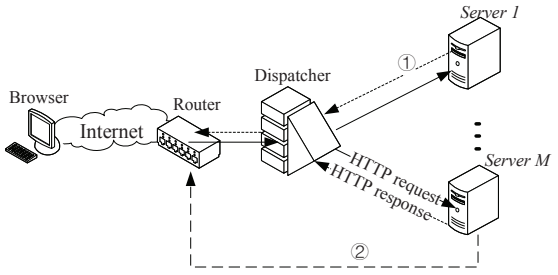


Figure 1: The Processing of HTTP Request in Web Cluster

The dispatcher should establish TCP connection with clients and back-ends concurrently. Although compared with the hardware-accelerated forwarding method used by L4/2 and L4/3 cluster, L7 cluster has the limitation of larger processing overhead. But it is still a promising implementation of Web server cluster (see [11]). It could not only combine L4/2, L4/3 forwarding technology (see [3]), but also take usage of application information to enforce the content-aware dispatcher and combine the priority scheduling with the processing/threads-based Web QoS control scheme (see [18]).

[1]residue delay is consist of connecting time and processing time.

From the view of forwarding technology used by dispatcher, there are also three types of clusters: *Reverse Proxy*, *TCP Splicing* and *TCP Handoff*. The most widely-used is Reverse Proxy which is shown in Figure 1 Arrow ①. The dispatcher distributes HTTP requests to back-ends and transmits the responses back to clients. In order to avoid the redundant data copies and enhance the forwarding efficiency of reverse proxy, TCP Splicing is enforced in operation system kernel. As Arrow ② shown in Figure 1, by means of TCP Handoff protocol, the TCP connection established between dispatcher and clients is transfer to back-ends, and the response is directly sent back to the clients without relaying of dispatcher, which increases the potential throughput of the system.

TCP splicing and TCP Handoff are both proposed to enhance the forwarding efficiency. However, none of them is software-compatible because the implements require the amendments of TCP/IP protocol stack and system kernel in dispatcher and/or back-end servers. For the reasons above, our research focus on L7 cluster whose dispatcher running as Reverse Proxy.

2.2 The Related Work

The current related work mainly focus on the load balancing strategy and algorithm. While the researches of L4 cluster are just on how to uniformly spread the requests. Besides this, L7 cluster considers the contents of the request as well.

Typical L4 load balancing strategies are as follows: Round Robin (RR), i.e. executes $(i+1)$ mod n for every request and selects back-end according to the result; Weighted Round Robin (WRR), i.e. every back-end servers' processing capacity is in accordance with its weight. The larger weight the more times requests be sent; Least Connection Scheduling (LCS) and Weighted Least Connection Scheduling (WLCS), which forward the requests to the server with least active TCP connections; Source Hashing Scheduling (SHS) and Destination Hashing Scheduling (DHS) are statics mapping algorithms, the IP source address or destination address of HTTP request is hashed and mapped to a certain back-end server.

There are two common scheduling strategy used in L7, Client-aware Policy (CAP) (see [4]) and Locality Aware Request Distribution (LARD)(see [13]). CAP is actually a kind of RR scheduling supported client-side classification. Same kind of requests are spreading uniformly without considering their content. LARD is server-status based load balancing strategy, in the threshold of stability, the same URL are forwarding to the same server to achieve a higher hit rate.

However, no matter L4 or L7 cluster research does not concern the issue of differentiated service and the related study is few. [21] gives the Demand-driven Service Differentiation(DDSD) approach, which improves Web Cluster model into a multi-queueing system, such as $M/M/1/\infty$ and $M/G/1/\infty$ models proposed by [19,20]. By selecting Stretch Factor, it treats the re-allocation of cluster resources as a SLAs constrained optimization problem. However, because queueing model is based on the premise of Poisson arrival rate and exponential distribution of processing time, queueing model can not accurately describe the of HTTP traffic feature when the arrival rate and leave rate does not match(see [2, 9]). Dynamic Partitioning (DynamicPart) algorithm is proposed by Casalicchio (see [1]). By feedback control, the back-ends is dynamically servicing different kinds of requests, which transforms a best-effort Web cluster into a QoS-enhanced system. But the performance isolation is enforced at the level of server host, it can only provide coarse-grain control at host-level, which disturbs the on-line extension and the resources utilization.

Our research purpose is to provide a differentiated services to increase the resources utilization of Web cluster at the level of processing/thread. First, the resource consumption of every back-end should be coordinated and load-balanced. Then the resource management and scheduling

should meet the QoS requirement (see [5]).

3 QoS-Enhanced Web Cluster

3.1 PDDS-LB Architechure

The processing/thread-based Web QoS control is used in the researches of single Web server. The processing or thread pool is partitioned into several parts to support the performance isolation. The processing or thread number in each part is called *quota*. The HTTP requests are classified into different business flows (1...N). By adjusting the quotas of every class, the performance isolation and differentiation service are achieved (see [8,14]). The initial ideal of PDDS-LB is a unified scheduling of all back-ends' processings/threades and a fair consumption of different servers' resource to the same flows. As shown in Figure 2, PDDS-LB supports differentiated service in two layers:

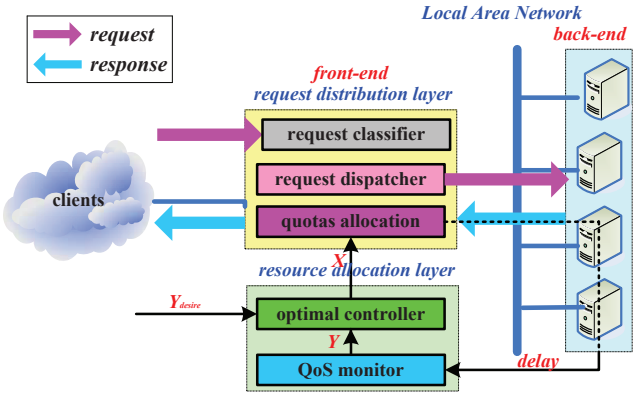


Figure 2: PDDS-LB Architecutre

In the resource allocation layer, the QoS monitor perceives the average residue delay of each class [2] $\ell_i, (1 \leq i \leq N)$ while the back-ends report their status periodically (detailed in Section 5). The optimal controller adjusts every back-ends's quota $c_{i,j}, (1 \leq j \leq M)$ assigned to different business flows.

In the request distribution layer, the requests are separated into different priorities according to the specified strategy and SLAs. The request dispatcher selects a back-end for each request by the load balancing algorithm.

The front-end dispatcher is the key element of the cluster, which establishes TCP connection with back-end and clients simultaneously. Back-end servers deploy the same contents and can service all the requests. Classification strategy back-ends used is consistent with the front-end. Different requests flows passed into the corresponding connecting queueing wait to be served in the manner of First-In-First-Out(FIFO).

Heartbeat is original used to make a high amiability of cluster infrastructure (front-end and back-ends)(see [15,16]). In PDDS-LB, it is referenced as a daemon to provide the communication between client infrastructure. At every heartbeat time, the status reporter sends the status messages to front dispatcher and the quotas of business flows are adjusted according to the optimal controller's output.

[2] residue delay is the sum of connecting time w and processing time χ.

3.2 PDDS-LB Software Component

Considered our previous work (see [8,9]), we implement and deploy PDDS-LB in the Apache plantform. the back-end server is modified into the multi-processing queue architecture. On the dispatcher, *mod_ proxy* (see in [6]) is activated as reverse proxy, and *mod_ proxy_ balancer* is used to translate the request URL into absolute URL in back-end server.

1. *mod_ cluster* is modified to achieve communication between dispatcher and back-end servers. At every heartbeat time, back-end servers initiatively report their status. Meanwhile, *mod_ clsuter* in dispatcher implements the function of resource allocation layer. By optimal control, the refreshed quotas of classes are sent back to back-end servers.

2. *mod_ proxy_ balancer* is modified to enforce the functions of request distribution layer. *mod_ cluster* calculates the busyness of every back-end server according to the their status and URLs are redirected to a appropriate server. Then Apache httpd obtains responses from it.

4 Cluster Resource Management

4.1 System Model Identification

Supposed the cluster consists of M back-end servers, serving N classes of business flows. Every back-end runs $C_j (j = 1, \ldots, M)$ service threads:

$$C_j = \sum_{i=1}^{N} c_{i,j}, \varsigma_i = \sum_{j=1}^{M} c_{i,j}, 1 \le i \le N, 1 \le j \le M, \tag{1}$$

where $c_{i,j}$ is the number of threads assigned to class i in j^{th} back-end, i.e. the quota that server j assigns to class i. ς_i is the number of threads assigned to class i in the whole cluster. At every sampling time, the optimal controller adjusts $c_{i,j}$ to hold Equation (2):

$$\frac{\ell_i}{\ell_l} = \frac{\delta_i}{\delta_l}, 1 \le i \le N, 1 \le l \le N, \tag{2}$$

where δ_i is the class i's priority assigned by SLA. The smaller δ_i, the higher its priority. As shown in Figure 2,the controlled object is threads of the whole cluster. For the constrain of Equation (1), the input has $I = N \times M$ independent variables. At the k^{th} sampling time, the input is:

$$\mathbf{X}(k) = [c_{1,1}(k), c_{1,2}(k), \ldots, c_{1,M}(k), \ldots, c_{N-1,1}(k), c_{N-1,2}(k), ..., c_{N-1,M}(k)]^{\mathrm{T}}.$$

Define $y_i(k)$, $y_{i_{desire}}$ are the *normalized residue delay* and its expected value:

$$y_i(k) = \frac{\ell_i(k)}{\sum_{l=1}^{N} \ell_l(k)}, y_{i_{desire}} = \frac{\delta_i}{\sum_{l=1}^{N} \delta_l}, 1 \le i \le N. \tag{3}$$

Because $\sum_{i=1}^{N} y_i(k) = 1$ and $\sum_{i=1}^{N} y_{idesire} = 1$, the output has $O = N - 1$ independent variables. So let

$$\begin{aligned} \mathbf{Y}(k) &= [y_1(k), y_2(k), \ldots, y_{N-1}(k)]^{\mathrm{T}}, \\ \mathbf{Y}_{desire} &= [y_{1_{desire}}, y_{2_{desire}}, \ldots, y_{N-1_{desire}}]^{\mathrm{T}}, \end{aligned}$$

where $\mathbf{Y}(k)$ is the measurement output. According to the derivation $\mathbf{E}(k)$ between $\mathbf{Y}(k)$ and $\mathbf{Y}_{desire}$, the optimal controller adjusts the threads quotas of every classes to guarantee the relative relationship constant.

Strictly speaking, we require a discrete and nonlinear model for PDDS-LB. However, such a nonlinear model is not amenable to the straight forward theoretical design and analysis (see [12]). SO the following linear model is used to approximate the system. Supposing r-order could be precise enough, the corresponding difference equation is:

$$\mathbf{Y}(k) = \sum_{j=1}^{r} [a_j \mathbf{Y}(k-j) + b_j \mathbf{X}(k-j)], \tag{4}$$

and the Z-domain transformation is:

$$\mathbf{A}(z^{-1})\mathbf{Y}(k) = \mathbf{B}(z^{-1})\mathbf{X}(k) + \varepsilon(k), \tag{5}$$

where $\varepsilon(k) = [\varepsilon_1(k), \varepsilon_2(k), ..., \varepsilon_{N-1}(k)]^T$ is O-order not related white noise sequence with mean zero.

$$\begin{aligned} \mathbf{A}(z^{-1}) &= \mathbf{I} - \mathbf{A}_1 z^{-1} - ... - \mathbf{A}_r z^{-r}, \mathbf{A}_i \in \mathbf{R}^{O\times O}, 0 < i \le r. \\ \mathbf{B}(z^{-1}) &= \mathbf{B}_1 z^{-1} + ... + \mathbf{B}_r z^{-r}, \mathbf{B}_j \in \mathbf{R}^{O\times I}, 0 < j \le r. \end{aligned}$$

Because of observation error and system noise, define $\boldsymbol{\varepsilon}(k) = [\varepsilon_1(k), \varepsilon_2(k), ..., \varepsilon_{n-1}(k)]^{\mathrm{T}}$ be O-order white noise sequence. Equation (5) can be rewritten as:

$$\mathbf{Y}(k+1) = \boldsymbol{\theta}\boldsymbol{\Phi}(k) + \boldsymbol{\varepsilon}(k+1), \tag{6}$$

where $\boldsymbol{\theta} = [\mathbf{B}_1 ..., \mathbf{B}_r, \mathbf{A}_1, ...\mathbf{A}_r], k \ge r-1$, $\boldsymbol{\Phi}(k) = [\mathbf{X}^{\mathrm{T}}(k), ..., \mathbf{X}^{\mathrm{T}}(k-r+1), \mathbf{Y}^{\mathrm{T}}(k), ..., \mathbf{Y}^{\mathrm{T}}(k-r+1)]^{\mathrm{r}}$, $\boldsymbol{\theta} \in R^{O\times[O\times 2\times r]}$.

Recursive least square (RLS) estimate algorithm is used to calculate parameter matrix $\boldsymbol{\theta}$, and the white noise-similarity of pseudo-random sequence is used as impulse to fully stimulate the system . In the system identification experiment, the cluster is composed of 4 back-end server, each of which runs 100 threads serving two classes of business flows. i.e. $N = 2$, $M = 4$,$C_j = 100, j = 1, \ldots, 4$, $\boldsymbol{Y}_{desire} = [y_{1_{desire}}] = [1/3]$. In order to fully stimulate the system, at every sampling time, the quotas assigned to every classes $\boldsymbol{X}(k+1) = [c_{1,1}(k+1), c_{1,2}(k+1), ..., c_{1,4}(k+1)]^{\mathrm{T}}$ are adjusted according to the current value of pseudo-random sequence. The pseudo-random sequence is generated as Equation (7). Set $p = 7$, $q = 12$, the relationship between $\epsilon(k)$ and $\mathbf{X}$ is shown in Table 1.

$$\epsilon(k) = \epsilon(k-p) + \epsilon(k-q) (\mathrm{mod} 4) \tag{7}$$

Table 1: Relation of $\epsilon(k)$ and $\mathbf{X}$ when $p = 7$, $q = 12$

$\epsilon(k)$	$c_{1,j}(k+1)$	$c_{2,j}(k+1)$
0	25	75
1	40	60
2	60	40
3	75	25

The system identification experiment lasts 5000 seconds with the sampling time $T = 30s$ and we got 150 sets of effective data. Supposing $\hat{\boldsymbol{\theta}}_q$ is the estimation of $\boldsymbol{\theta}$ from the former q ($q \ge r-1$)

sampled data. After the $q+1^{th}$ sampling time, $\hat{\boldsymbol{\theta}}_q$ can be revised as:

$$\hat{\boldsymbol{\theta}}_{q+1} = \hat{\boldsymbol{\theta}}_q + \frac{[\mathbf{Y}(k+1) - \hat{\boldsymbol{\theta}}_q \boldsymbol{\Phi}(k)]\boldsymbol{\Phi}^{\mathrm{T}}(k)\mathbf{P}_q}{\lambda + \boldsymbol{\Phi}^{\mathrm{T}}(k)\mathbf{P}_q\boldsymbol{\Phi}(k)}, \tag{8}$$

where

$$\mathbf{P}_{q+1}^{-1} = \mathbf{P}_q^{-1} + [1 + (\lambda - 1)\frac{\boldsymbol{\Phi}^{\mathrm{T}}(k)\mathbf{P}_q\boldsymbol{\Phi}(k)}{(\boldsymbol{\Phi}^{\mathrm{T}}(k)\boldsymbol{\Phi}(k))^2}]\boldsymbol{\Phi}(k)\boldsymbol{\Phi}^{\mathrm{T}}(k),$$

$\mathbf{P}_q$ is covariance matrix and λ is forgetting factor. $\boldsymbol{\Phi}$, $\mathbf{Y}$ are measured by QoS monitor. By selecting an appropriate $\hat{\boldsymbol{\theta}}_0$ and $\mathbf{P}_0$, we could get the estimation of parameter matrix. The criteria for $\hat{\boldsymbol{\theta}}_0$ and $\mathbf{P}_0$ is

$$\begin{cases} \hat{\boldsymbol{\theta}}_0 = \boldsymbol{\iota}, \boldsymbol{\iota} \text{ is sufficiently small real vector} \\ \mathbf{P}_0 = \alpha^2\mathbf{I}, \alpha \text{ is sufficiently large real number} \end{cases} \tag{9}$$

The loss function is defined as Equation (10) to describe the variance between identified residue delay proportion and its measured value.

$$j(m) = \sum_{k=m}^{M+m-1} ||\mathbf{Y}(k+1) - \hat{\boldsymbol{\theta}}_q\boldsymbol{\Phi}(k)||^2, \tag{10}$$

where $||\bullet||$ is vector norm, and R is sample size.

The system order r is desided by F-test. Supposed m_1 and m_2 are adjacent orders of system, statistics variable H is constructed as follows:

$$H(m_1, m_2) = \frac{j(m_1) - j(m_2)}{j(m_2)}\frac{M - 2m_2}{2(m_2 - m_1)}. \tag{11}$$

If M is large enough and $m_2 > m_1$, $H(m_1, m_2)$ obeys F-distribution.

$$H \sim F(2(m_2 - m_1), M - 2m_2).$$

The following pseudo-code used for $j(m)$ will be generated:

1. **Begin**
2. Set m be the maximum possibel order, i.e. $m = m_{MAX}$.
3. $q = 0$, $k = m - 1$, chose an appropriate $\hat{\boldsymbol{\theta}}_0$ and $\mathbf{P}_0$.
4. $\boldsymbol{\Phi}(k)$ is constructed according to the former $[k - m + 1, k]$ sample data.
5. $\hat{\boldsymbol{\theta}}_{q+1}$ and $\mathbf{P}_{q+1}$ is calculated as Equation (8).
6. $k = k + 1$, $q = q + 1$.
7. IF $k \leq M$, goto *4*; Else $\hat{\boldsymbol{\theta}}_{M-m+1}$ is the RLS estimation of this m-order system.
8. $j(m)$ is calculated as Equation (10).
9. $m = m - 1$.
10. IF $m \geq 1$, goto *3*.
11. **End**.

Then $\mathbf{J} = [j(1), j(2), ..., j(m_{MAX})]^{\mathrm{T}}$ is obtained. Given the degree of confidence $\alpha = 5\%$, where is $F_{0.05}(2, 144) \approx 3.05$. Because $H(2, 3) = 2.29 < F_{0.05}(2, 144)$, which means there is no

significant reduction of the loss function when system order changes from 2 to 3. So the PDDS-LB model can be modeled as a second-order linear time-invariant system. The corresponding RLS estimation of parameter matrix $\boldsymbol{\theta}$ is:

$$\begin{aligned}
\hat{\boldsymbol{\theta}} &= [\hat{\boldsymbol{B}}_0,\hat{\boldsymbol{B}}_1,\hat{\boldsymbol{A}}_1,\hat{\boldsymbol{A}}_2], \\
\hat{\boldsymbol{B}}_1 &= [b_{1,1}, b_{1,2}, b_{1,3}, b_{1,4}] = [-0.0004, -0.0005, -0.0004, -0.0004] \\
\hat{\boldsymbol{B}}_2 &= [b_{2,1}, b_{2,2}, b_{2,3}, b_{2,4}] = [0.0049, 0.0060, 0.0056, 0.0050] \\
\hat{\boldsymbol{A}}_1 &= a_1 = 0.4528 \\
\hat{\boldsymbol{A}}_2 &= a_2 = 0.0922
\end{aligned}$$

Figure 3 is the compare of identified value vs. actual measurement of $\boldsymbol{Y}(k)$. The identified value of $\boldsymbol{Y}(k)$ is closed to the measured value, so 2-order linear MIMO model is appropriate to describe PDD-LB.

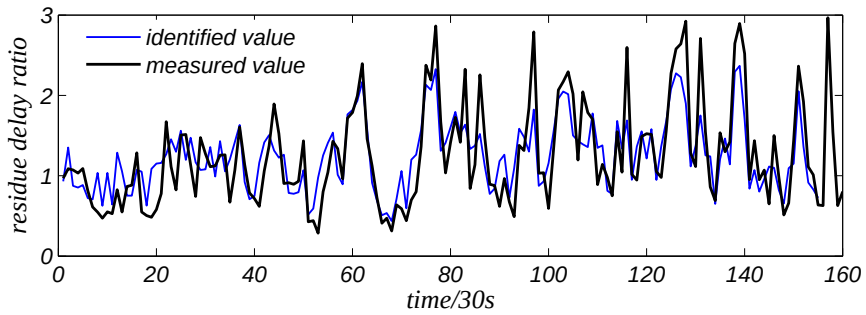


Figure 3: The identified value of $\boldsymbol{Y}(k)$ is closed to actual measurement

4.2 Optimal Controller Design

The resource reallocation of process/thread can be treated as an optimization problem which minimizes the deviation between $y_i(k)$ and $y_{i_{desire}}$, while penalizes large changes in the control variables to save the overhead (see in [12]). So we construct the following quadratic cost function:

$$\mathfrak{L} = \mathfrak{E}\{||\boldsymbol{W}[\boldsymbol{Y}(k+1) - \boldsymbol{Y}_{desire}]||^2 + ||\boldsymbol{Q}[\boldsymbol{X}(k) - \boldsymbol{X}(k-1)]||^2\}, \tag{12}$$

where $\boldsymbol{W}$ is $O \times O$ positive-definite weighting matrix and $\boldsymbol{Q}$ is $I \times I$ positive-definite penalty matrix. Diagonal elements of $\boldsymbol{W}$ represent the priorities of the corresponding classes, the smaller $w_{i,i}$, the more discriminated against the class i . In our application, $\boldsymbol{W}$, $\boldsymbol{Q}$ are diagonal matrix and unit matrix. The derivative of $\mathfrak{L}$ about $\boldsymbol{X}$ is zero when at its minimum, so there is(the derivation of equation see in [10, 12]):

$$\begin{aligned}
\frac{\partial \mathfrak{L}}{\partial \boldsymbol{X}(k)} &= 2(\boldsymbol{W}\hat{\boldsymbol{B}}_0)^{\mathrm{T}}\boldsymbol{W}[\hat{\boldsymbol{\theta}}\tilde{\boldsymbol{\Phi}}(k) - \boldsymbol{Y}_{desire}] + 2(\boldsymbol{W}\hat{\boldsymbol{B}}_0)^{\mathrm{T}}\boldsymbol{W}\hat{\boldsymbol{B}}_0\boldsymbol{X}(k) \\
&+ 2\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{X}(k) - 2\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{X}(k-1) = 0,
\end{aligned} \tag{13}$$

By solving the Equation (13), we can obtain the following optimal control law at the k^{th} sampling time

$$\begin{aligned}
\boldsymbol{X}^*(k) &= [(\boldsymbol{W}\hat{\boldsymbol{B}}_0)^{\mathrm{T}}\boldsymbol{W}\hat{\boldsymbol{B}}_0 + \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q}]^{-1}\{(\boldsymbol{W}\hat{\boldsymbol{B}}_0)^{\mathrm{T}}\boldsymbol{W}[\boldsymbol{Y}_{desire} - \hat{\boldsymbol{\theta}}\tilde{\boldsymbol{\Phi}}(k)] \\
&+ \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{X}(k-1)\}.
\end{aligned} \tag{14}$$

5 Maximum Idle First

The resource optimal control is used to calculate the quotas ς_i assigned to class i, while the load balancing strategy is to assure a fair assumption of threads, avoiding some back-end starvation or overload. Most researches of load balancing are on the promise of Possion arrival distribution and the exponential service time distribution, such as RR and LCS. However, the tendency that Web pages is on a going to contains more and more dynamic objects and database operation makes processing time hard to meet the exponential distribution, and all of these strategies can not support differentiated service. In order to precisely estimate the load status of each classes in back-end servers and dispatch new coming requests equally, we propose a load balancing strategy called Maximum Idle First (MIF).

The status of j^{th} back-end server are N sequence of two-tuples $< n_{i,j}, \tau_{i,j} >$ďżˇďżˇ$i = 1, \ldots, N$, which includes the connection queue length n_{ij} and the number of idle threads in the server pool τ_{ij}. The residue delay ℓ_i is proportional to the queue length $n_{i,j}$ and inverse proportional to the number of service threads, i.e. $\tau_{i,j}$ (see in [8, 14]). For the clients supported HTTP 1.1 , a Web session is always a sequence of HTTP requests on a consistent TCP connection as a manner of Pipeline, so there is

$$\ell_{i,j} = \alpha n_{i,j} \mathfrak{E}[\rho_{i,j}]/\tau_{i,j}, \tag{15}$$

$$\boldsymbol{idle}_{i,j} \propto 1/\ell_{i,j}. \tag{16}$$

Let $\boldsymbol{idle}_{i,j}$ be the idle degree of service for class i in server j, which is inverse proportional to the residue dalay. α is a planform dependence translation parameter. $\mathfrak{E}[\rho_{i,j}]$ is the mathematic expecting of Web session size. (see in [7]),there is :

$$\mathfrak{E}[\rho_{i,j}] = \mathfrak{E}[v_{i,j}]\mathfrak{E}[u_{i,j}], 1 \le i \le N, 1 \le j \le M \tag{17}$$

$v_{i,j}$,$u_{i,j}$ are the size and the number of the embedded requests from a Web session. Because every back-end servers deploys the same content and the self-similarity of the requests, $\mathfrak{E}[\rho_{i,j}]$ is a constant value. When a request belonged to class i arrivals at the front-end dispatcher, *mod_ proxy_ banlancer* calculates each back-end's $\boldsymbol{idle}_{ij}$, then redirects this request to the server, which has the maximum idle degree of service.

6 Experiment Evaluation

6.1 Configuration of Experiment

The test-bed is developed to evaluate the PDDS-LB mechanism, which consists of 9 computers connected together via 1Gbps Ethernet. There are 4 back-ends running Apache-2.0.53, which each has 100 concurrent threads, and 4 Linux machines simulate 120×4 different clients using SURGE-1.00a. Considering the front-end may be a new bottleneck, we configure the Apache-2.2.63 on the front-end with 1000 concurrent threads, which is large enough. In our experiment, HTTP requests are classified into 2 business flows based on the clients' IP address,i.e. class 1 and class 2. Set the expected delay ratio is $\ell_1/\ell 2 = 1/2$, $\boldsymbol{Y}_{desire} = [1/3]$, i.e. the class 1's residue delay should be the half of the class 2's.

Define the relative variance $\Psi(Y)$ be a control performance metric. A smaller $\Psi(Y)$ indicates a better stability that controller can keep $\boldsymbol{Y}(k)$ at $\boldsymbol{Y}_{desire}$.

$$\Psi(Y) = \frac{\sqrt{\sum_{k=1}^{I} ||\boldsymbol{Y}(k) - \boldsymbol{Y}_{desire}||^2 / I}}{\boldsymbol{Y}_{desire}}, \tag{18}$$

6.2 Result and Analysis

We design two sets of contrast experiments to evaluate the PDDS-LB cluster's ability of differentiation,throughput and dalely under different load balancing strategy.

Firstly, We evaluate the optimal controller's ability of differentiation under different load balancing strategy and sampling time. As shown in Figure 4(a), 4(b) and 4(c), each of them are the results of $T = 20s$ and $T = 30$ when using RR, LCS and MIF algorithm. The optimal controller launches at 800 seconds, and the measured residue delay ratio of class 1 and class2 is gradually settled around the expected value. According to the Equation (18), $\Psi(Y)$ are calculated and shown in Figure 4(d). Compared Figure 4(a), 4(b) and 4(c), the following conclusion can be summarized:

1. No matter using which load balancing algorithms or which sampling time, our optimal controller can provide proportional delay services. This proves the correctness and feasibility of PDDS-LB cluster model.

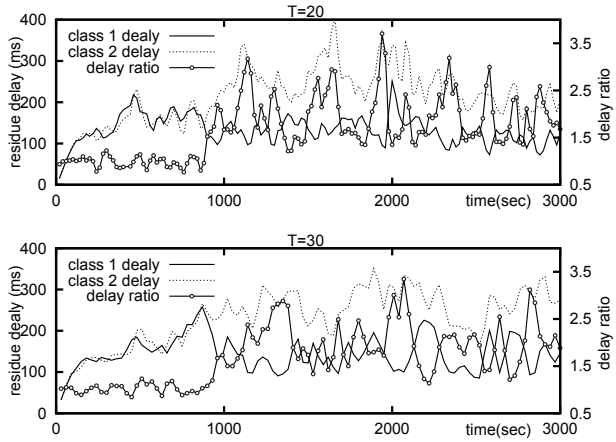
2. ℓ_i^{-800} and ℓ_i^{+800} are used respectively to compare average residue delay before and after 800 sec, there is:
$$\ell_2^{+800} - \ell_2^{-800} \geq \ell_1^{-800} - \ell_1^{+800}, \tag{19}$$
It means that the increase of class 2's delay is more than the decrease of class 1's. Nevertheless, it is worth making for the reasons in Section 1.

3. In equilibrium state, i.e. $\boldsymbol{Y}(k) = \boldsymbol{Y}_{desire}$, there are
$$\ell_{1,MIF}^{+800} < \ell_{1,RR}^{+800} \approx \ell_{1,LCS}^{+800}, \tag{20}$$
$$\ell_{2,MIF}^{+800} < \ell_{2,RR}^{+800} \approx \ell_{2,LCS}^{+800}, \tag{21}$$
where $\ell_{i,RR}^{+800}$, $\ell_{i,LCS}^{+800}$, $\ell_{i,MIF}^{+800}$ are the average residue delay of class i in stable sate under RR, LCS and MIF strategies, Equation (20)and (21) proves that MIF has less delay in the stable state.

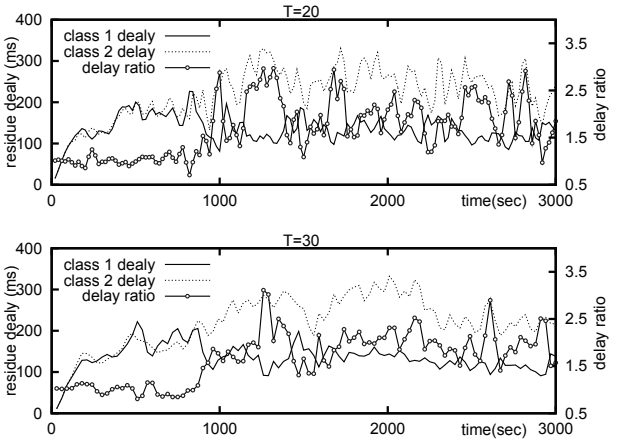
4. As shown in figure 4(d), when sampling time $T = 20s$, the delay ratio jitters severely. The possible reason is the delay jitter caused by large files, and which are more apparent in small sampling time.

Then, a further comparation is enforced to evaluate the different balancing strategy on the system throughput and residue delay. When $T = 20$, as the increase of the client's TCP connection requests arrival rate, the throughput and residue time also enhance, while the two classes of business still keep differentiate relationship. The Figure 5(a) and 5(b) are the histogram of throughput and the residue delay with the width of 20 TCP connections/sec. As can be seen, when the TCP connection arrive rate is 25/sec the system is saturated. Both in LCS and RR, the throughput of class 1 and 2 are 80 requests/sec and 40 requests/sec respectively, while in MIF, the throughput of class 1 can increase to 120 requests/sec. At the same time, in LCS and RR, the average delay of class 1 and 2 are 100ms and 280ms, while the average delay of class 2 reduce to 200ms in MIF.

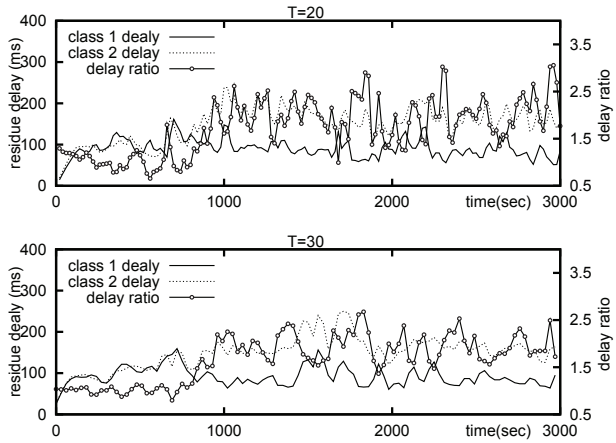
This is not only because MIF consider the impact of the connection queue length and idle threads on the residue delay, but also due to the fine-grained state feedback. The status of back-ends contain details of every class , we can design a better dispatch strategy. Although LCS and RR can dispatch requests to all back-ends evenly, since lack of the fine-grained state feedback, they cannot maximize the resource utilization. For example, if the thread quotas of class 1 will be exhausted, or its connection queue will overflow on back-end j, while the resources



(a) RR



(b) LCS



(c) MIF

$\Psi(Y)$	RR	LCS	MIF
20s	0.3843	0.3810	0.31925
30s	0.3831	0.3759	0.28000

(d) $\Psi(Y)$

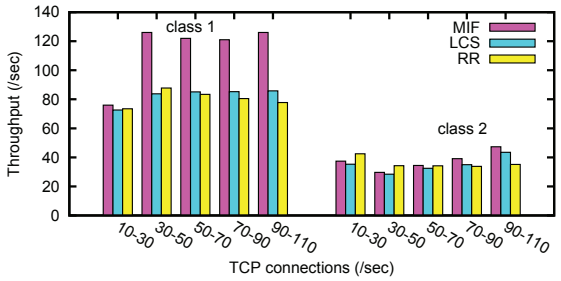
Figure 4: Differentiation effects under Different load balancing strategy and sampling time

of other classes are idle, LCS and RR tend to redirect requests to this back-end. Now, there is a new request, which is exactly belongs to class 1, resource shortage on back-end j will further deteriorate, however, other back-ends may be in idle status.

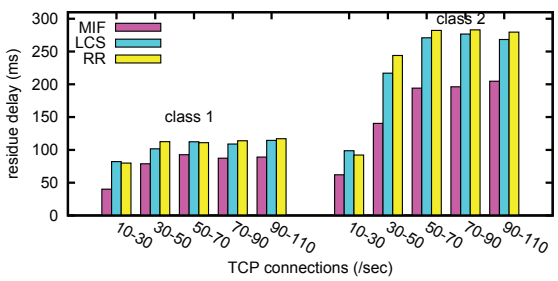
7 Conclusion

In this paper, we design a feedback control mechanism to achieve Proportional Delay Differentiation Service and Load Balancing in a Web Cluster System. By recursive least square estimation and F-test, PDDS-LB is model as a second-order liner time-invariant system. We construct the cost function and obtain the optimal law by derivation of cost function. Experimental evaluations have shown that our mechanism achieves PDDS, while the low priority class is not over-sacrificed. With the aid of MIF, we maximize the resource utilization of the cluster system.

However, our experiments are just in the condition of ideal transport layer, in the real network, link status is complex and changeable ,the factors that influence QoS interweave each other, such as bandwidth, cache, I/O etc. As part of out ongoing work, we are exploring a integrated resource management to adapt to more complex environment.



(a) Throughput



(b) Delay

Figure 5: Compared with LCS and RR, the average delay reduces 21% and the total throughput increases 33% in MIF.

Bibliography

[1] M. Andreolini, E. Casalicchio, M. Colajanni, and M. Mambelli. A cluster-based web system providing differentiated and guaranteed services. *Cluster Computing*, 7(1):7–19, 2004.

[2] G. Ang, M. Dejun, H. Yansu, and P. Wenping. Proportional Delay Guarantee in Web QoS Based on Predictive Control. In *Information Science and Engineering (ICISE), 2009 1st International Conference on*, pages 1789–1792. IEEE, 2009.

[3] G. Apostolopoulos, D. Aubespin, V. Peris, P. Pradhan, and D. Saha. Design, Implementation, and Performance of a Content-Based Switch. In *IEEE INFOCOM*, volume 3, pages 1117–1126. Citeseer, 2000.

[4] E. Casalicchio and M. Colajanni. A client-aware dispatching algorithm for web clusters providing multiple services. In *Proceedings of the 10th international conference on World Wide Web*, page 544. ACM, 2001.

[5] L. Cherkasova and M. Karlsson. Scalable web server cluster design with workload-aware request distribution strategy WARD. In *wecwis*, page 0212. Published by the IEEE Computer Society, 2001.

[6] JBoss Community. http://www.jboss.org/mod_cluster.

[7] A. Gao, D. Mu, and Y. Hu. A QoS Control Approach in Differentiated Web Caching Service. *Journal of Networks*, 6(1):62–70, 2011.

[8] A. Gao, D. Mu, Y. Hu, and W. Pan. Proportional Delay Guarantee in Web QoS Based on Predictive Control. *1st International Conference on Information Science and Engineering,ICISE2009*, 1:1789–1792, 2009.

[9] A. Gao, H. Zhou, Y. Hu, D. Mu, and W. Hu. Proportional Delay Differentiation Service and Load Balancing in Web Cluster Systems. In *INFOCOM IEEE Conference on Computer Communications Workshops, 2010*, pages 1–2. IEEE, 2010.

[10] M. Karlsson, X. Zhu, and C. Karamanolis. An adaptive optimal controller for non-intrusive performance differentiation in computing services. In *Control and Automation, 2005. ICCA'05. International Conference on*, volume 2, pages 709–714. IEEE.

[11] A.F. Liu. *Research on the Web Server Cluster Technologies*. PhD thesis, Central South Univercity, 2005.

[12] X. Liu, X. Zhu, P. Padala, Z. Wang, and S. Singhal. Optimal multivariate control for differentiated services on a shared hosting platform. In *Proc. of the 46th IEEE Conf. on Decision and Control*. Citeseer, 2007.

[13] V.S. Pai, M. Aron, G. Banga, M. Svendsen, P. Druschel, W. Zwaenepoel, and E. Nahum. Locality-aware request distribution in cluster-based network servers. In *Proceedings of the eighth international conference on Architectural support for programming languages and operating systems*, pages 205–216. ACM, 1998.

[14] W. Pan, D. Mu, H. Wu, and Q. Sun. Proportional Delay Differentiation Service in Web Application Servers: A Feedback Control Approach. *International Journal of Intelligent Information Technology Application*, 1(1):37–42, 2008.

[15] A. Robertson. Linux-HA heartbeat system design. In *Proceedings of the 4th annual Linux Showcase & Conference-Volume 4*, pages 20–20. USENIX Association, 2000.

[16] A. Robertson. The evolution of the Linux-HA project. In *UKUUG LISA/Winter Conference High-Availability and Reliability*, 2004.

[17] T. Schroeder, S. Goddard, and B. Ramamurthy. Scalable Web server clustering technologies. *IEEE network*, 14(3):38–45, 2000.

[18] Z.G. Shan and C. Ling. Performance Evaluation of QoS-Aware Load Balancing of Web-server Clusters. *Journal of System Simulation in chinese*, 17:184–189, 2005.

[19] X. Wu, M. Li, and J. Wu. Enhanced Demand-driven Service Differentiation Algorithm in Web Clusters. In *IEEE International Conference on e-Business Engineering, 2006. ICEBE'06*, pages 386–391, 2006.

[20] Z. Xing, P.L. Yan, and C.C. Guo. Proportional Stretch Factor Differentiated Service in Heterogeneous Web Server Cluster. *Computer Science in chinese*, 33(010):61–65, 2006.

[21] H. Zhu, H. Tang, and T. Yang. Demand-driven service differentiation in cluster-based network servers. In *IEEE INFOCOM*, volume 2, pages 679–688. Citeseer, 2001.