# Web服务器系统中基于反馈控制的比例延迟保证

潘文平　慕德俊　戴冠中　武航星
（西北工业大学自动化学院　西安 710072）

**摘　要**　随着Web应用在商业领域的广泛使用，Web服务器系统需要在高负载下提供区分服务，以满足用户的不同需求。为实现以延迟作为评价指标的区分服务，本文在Web服务器系统的连接管理和请求处理两个层次建立了基于反馈控制的比例延迟保证模型。模型中的反馈控制器通过动态计算和调节不同类别客户占用的资源：(服务线程和数据库连接)，能保证高优先级的客户较快得到服务而不同类别客户的平均延迟比保持不变。为测试闭环系统的性能，设计了两种分别服从均匀分布和重尾分布的动态负载。仿真结果表明，即使并发客户连接的数目剧烈变化，控制器作用下的服务器系统仍然能够达到较好的比例延迟保证，可靠地为用户提供区分服务。

**关键词**　反馈控制，比例区分服务，Web服务器

## Feedback Control-based Proportional Delay Guarantees in Web Server Systems

PAN Wen-Ping　MU De-Jun　DAI Guan-Zhong　WU Hang-Xing
(College of Automation, Northwestern Polytechnical University, Xi'an 710072)

**Abstract**　As more business applications become Web enabled, Web server systems evolve to provide service differentiation to satisfy different requirements of clients under heavy load conditions. To implement service differentiation with respect to delays in the Web server system, in this paper, proportional delay guarantee models based on feedback control are constructed at connection management level and request processing level. Controllers in the models can dynamically calculate and adjust the amount of resources, the worker threads and the database connections, for different types of clients according to their proportional delay requirements. Two kinds of workloads, which follow uniform and heavy-tailed distributions respectively, are generated for simulation of the close-loop system. Experiment results indicate that, proportional delay guarantees can be achieved in the Web server system successfully even if the number of the concurrent clients is changed abruptly under different kinds of workloads.

**Keywords**　Feedback control, Proportional differentiation service, Web server

## 1　Web应用的延迟及相关工作

在HTTP协议之下，客户通过TCP连接向Web服务器或Web应用服务器发送请求，并接收服务器返回的响应。从客户的角度来看，对于Web应用，其端到端的延迟主要由以下几个方面构成：(1)连接延迟。指TCP连接从建立到被服务线程或服务进程接受的时间间隔，即已经完成的连接在Accept Queue中的排队时间。图1(2)显示了已建立的TCP连接进入Accept Queue的情形。当accept( )被调用时，此队列队首的连接出队并被分配给服务线程或服务进程。(2)处理延迟。即服务器读取客户请求，生成响应所需的时间。(3)传输延迟。主要指在客户与服务器间通过三次握手建立TCP连接(如图1[1]所示)，以及通过TCP连接传输请求和响应所需的时间。

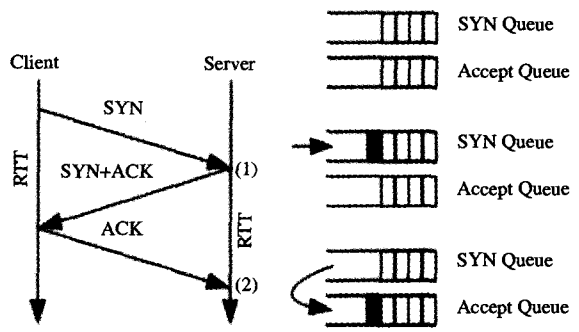


图1　三次握手与连接延迟

为在服务器上实现以延迟作为评价指标的区分服务和性能保证，近年来人们展开了大量工作。当发送请求的并发的客户数目多于Web服务器如Apache等设置的最大服务线程

**潘文平**　博士研究生，主要研究方向为Web QoS控制、网络拥塞控制；**慕德俊**　教授，博士生导师，主要研究领域为计算机网络、信息安全；**戴冠中**　教授，博士生导师，主要研究领域为自动控制、复杂系统与复杂性理论等；**武航星**　博士研究生，主要研究方向为网络拥塞控制。

2　朱征宇，裴仰军，陈华月，等. 个性化服务中用户近期兴趣视图的生成. 计算机工程与设计，2005(4)：952～953
3　Deerwester S, Dumais S T, Furnas G W, et al. Indexing by latent semantic analysis. Journal of the American Society for Information Science, 1990, 41(6): 391～407
4　Salton G. Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer [ED/OL]. Addison-Wesley. http://citeseer.nj.nec.com/content/26897/0 ,1989
5　Zhang Yuejie , Zhang Tao, Chen Shijie. Research on Lucene-based English-Chinese Cross-Language Information Retrieval. Journal of Chinese Language and Computing, 15 (1): 25～32
6　Wang S, Tanaka Y. Topic-oriented query expansion for web search. In: Proceedings of the 15th International Conference on World Wide Web, Edinburgh, Scotland, May 2006
7　张敏，宋睿华，马少平. 基于语义关系查询扩展的文档重构方法. 计算机学报，2004(10)：1398
8　瞿瑞彩，谢伟松. 数值分析，天津：天津大学出版社，2001. 2～3
9　赵亮，胡乃静，张守志. 个性化推荐算法设计. 计算机研究与发展，2008(8)：989～990

或服务进程数目时，连接延迟会大幅度增加。文[3,4]等通过不同的方法，实现了 Web 服务器端基于连接延迟的区分服务。文[8]进一步提出了 percentile delay，并实现了相对 percentile delay 保证。文[7]则以延迟的另外一种形式(Slowdown)作为评价指标，实现了基于 Slowdown 的区分服务。针对静态 HTTP 请求的大小呈重尾分布的特点，文[6]在 IP 层将响应的 IP 包按照其所对应的请求的大小分为多个不同优先级的队列，运用 SRPF 调度算法，有效地控制了不同响应的发送次序，从而从整体上降低了静态请求的平均响应时间。

以上改进方法都是针对 HTTP 静态请求的，并且所有工作都是通过修改操作系统相关代码实现的。对于动态请求，当 Web 应用服务器繁忙时，其处理延迟大幅度增加。而现有的处理动态请求的 Web 应用服务器，如 Tomcat，IBM WebSphere 和 BEA WebLogic 等，都不能提供类似于 Web 服务器下的延迟区分服务。而 Web 应用延迟的前两部分，即连接延迟和请求处理延迟，完全由服务器系统决定，因此可以将延迟保证模型建立在 TCP 连接管理和 HTTP 动态请求处理两个层次上，为用户提供区分服务。

## 2 二级比例延迟保证模型

### 2.1 TCP 连接管理层的比例延迟保证模型

文[15]最早提出了比例区分服务(PDS：proportional differentiated services)。由 PDS，比例延迟保证可做如下定义：如果两类客户 A 和 B 的区分参数(differentiation parameter)分别为 $w_A$ 和 $w_B$，则服务器在繁忙时应保证其延迟比满足 $\frac{d_A}{d_B}=\frac{w_A}{w_B}$。本文首先通过修改 Apache 服务器软件相关模块，建立了连接管理层的比例延迟保证模型。

#### 2.1.1 Apache 的多道处理模块

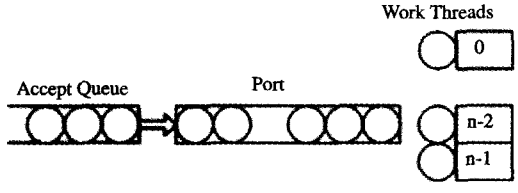


图 2 Apache MPM 模块

Apache 的连接管理模块称为多道处理模块 MPM (Multi-Processing Module)。针对不同的操作系统，Apache 采用不同的 MPM。在 Windows 平台下，Apache 2.0(本文所用源代码版本为 httpd2.0.53[14])采用的 MPM 是 WinNT MPM。WinNT MPM 可简化为一个多线程模型(默认监听线程数为一个)。如图 2 所示，一个工作者线程可对应一个客户连接，空闲的工作者线程阻塞于完成端口(IO Completion Port，即图 2 中 Port)，所有空闲的工作者线程形成空闲线程栈。监听线程通过调用 AcceptEx( )接受到一个客户连接，则将其发送到完成端口，并激活栈顶的空闲的工作者线程。工作者线程处理完该连接上的请求后，重新回到栈顶。客户连接的传递通过完成端口实现，完成端口实现了一个 FIFO 队列的功能。默认情形下，在没有空闲服务线程时，监听线程不再将从 Accept Queue 接收的连接发送到完成端口，因此完成端口中的排队长为 0，此时客户的连接延迟即其在 Accept Queue 中的排队时间。

#### 2.1.2 Apache 的多道处理模块的改进

为实现比例延迟保证，对原始的 MPM 模型做如下改进：根据客户类别的不同，采用多个不同的完成端口(图 3 所示为一个监听线程，两类客户的情形)，每个完成端口对应一类客户。另外增加一个控制线程 $C$，它根据上一采样周期不同类别客户的平均延迟比 $y$(如图 4)，为不同的完成端口提供不同数量的工作者线程，即调整线程比 $x$，达到不同数量的工作者线程服务于不同类别的客户。与此同时在监听线程中增加一个分类器，它按客户 IP，将特定的客户连接发送到相应的完成端口上。由于监听线程调用 AcceptEx( )前并未考虑是否存在空闲的服务线程，新到达 Accept Queue 的连接被立即发送到了完成端口，因此连接延迟从 Accept Queue 转移到了完成端口中。为防止服务器严重过载，在完成端口中的排队长度 $L$ 达到某阈值 $m$ 时，可重新调用 listen( )并将 backlog 参数设置为 0，使得操作系统将 Accept Queue 清空且不再接收新的连接，直到 $L$ 低于 $m$。

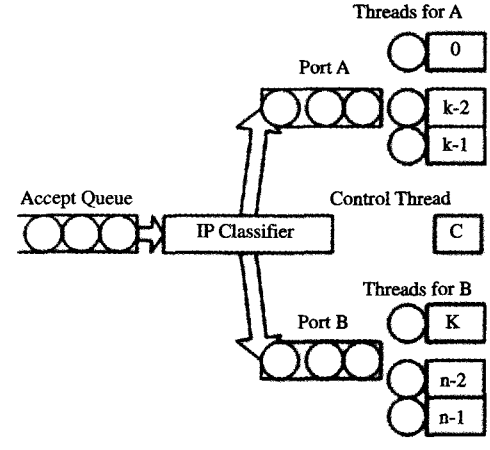


图 3 改进后的 Apache MPM 模块

改进后的 WinNT MPM 模型可以描述为一个等效的控制模型，如图 4 所示。其中控制对象为具有多个完成端口的 MPM 模块，控制系统的输入 $r$ 为 A，B 两类客户的区分参数比(即目标平均延迟比)。模型中控制线程的作用则是根据实际测量的延迟比和目标值的误差来动态调整服务于不同类别客户的工作线程的比例，从而使 Apache 服务器能够提供具有比例延迟保证的服务性能。

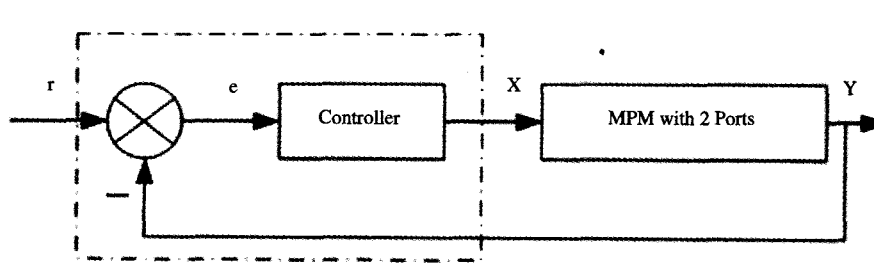


图 4 比例延迟保证控制模型(连接管理层)

### 2.2 请求处理层的比例延迟保证模型

在协议 HTTP 0.9 和 HTTP 1.0 下，每个 HTTP 请求使用一个独立的 TCP 连接，因此每个请求的反应时间都包含连接延迟。而在协议 HTTP1.1 下，同一客户的多个 HTTP 请求可以通过同一个持续连接（persistent connection）次序发送，连接延迟只会对该连接上的第一个请求的反应时间产生影响。所以有必要在请求处理阶段，为客户提供区分服务，使得不同类别的客户享受到不同的请求处理延迟。

由于多数动态请求都包含数据库操作，本文从改进数据库连接池出发，提出了在 Web 应用服务器上实现区分服务的方法，即保证服务器繁忙时，不同类别的客户请求在申请数据库连接时享受到不同的延迟。

#### 2.2.1 数据库连接池的工作原理[5]

数据库连接池内维护着多个可以重复使用的数据库连接。初始化时，连接池生成一定数量的数据库连接，并将其状态全部设置为空闲。当客户请求申请连接时，如果连接池内存在空闲的连接，则连接池立即查找并返回给客户请求一个空闲的连接，同时将该连接的状态设置为繁忙；如果连接池内没有空闲的连接，并且连接池维护的总连接数未达到预设的最大值，连接池将新建一些数据库连接（通常按照默认的递增值，比如两个）并将其返回给客户请求；如果此时连接池内的总连接数达到了预设的最大值并且没有空闲的连接，连接池将在等待一定时间后，再次查找空闲连接，并将结果返回给客户请求。客户请求使用完连接，将其退还给连接池时，连接池不会将该连接关闭，而是重新将该连接的状态设置为空闲，供其它客户请求使用。数据库连接池避免了过多的数据库连接的建立与关闭操作，提高了连接的利用率，降低了动态请求的处理时间，因而 Web 应用服务器中得到了广泛的应用。

#### 2.2.2 数据库连接池的改进

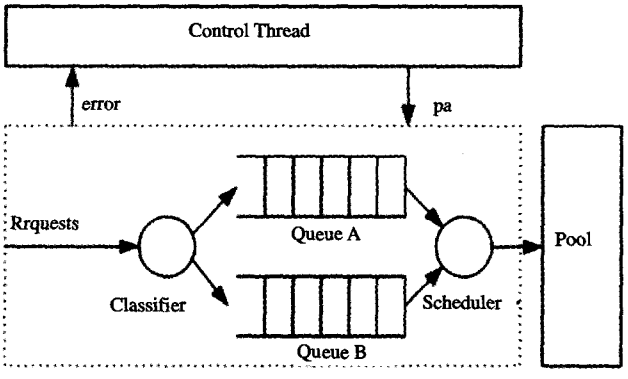


图 5 数据库连接池的改进模型

为实现比例延迟保证，对数据库连接池作如下改进。(1)增加请求分类器。分类器按客户请求的源 IP、会话 ID 或请求的 http-header 等信息将客户请求分为两类：普通请求和优先请求。每类请求形成一个 FIFO 队列，等待空闲的数据库连接。(2)建立数据库连接池控制线程。假定当前为第 $k$ 个采样时刻，控制线程将根据第 $k-1$ 个采样周期每类请求的平均延迟，调节第 $k$ 个周期中每类请求获取空闲连接的概率 pa 和 1-pa，即保证不同类别的请求获得不同比例的资源。以概率作为控制量的目的是实现细粒度的资源分配。(3)建立请求分派器。在数据库连接池出现空闲连接时，分派器将先生成一个随机数 $x$，该随机数是来自 U(0,1)分布的一个样本。比较 $x$ 与 pa 后，分派器将空闲的连接分派给相应类别的一个客户请求。如果第 $k$ 个采样周期连接池累计服务了 S 个请求，则 A 类请求占 pa，而 B 类请求占 1-pa。

## 3 系统辨识与控制器设计

近年来，控制理论在网络服务器系统性能改进中的应用引起了人们的广泛关注。文[11]将反馈控制方法运用到邮件服务器中，根据一定的利润模型，通过控制器动态调节服务器参数，提高了邮件服务器的给运营商带来的收益。文[2,7]等将控制方法运用到 Web 服务器的性能改进之中。本文则将反馈控制方法综合运用到 Web 服务器和 Web 应用服务器构成的具有负载均衡能力的服务器系统中（如图 7），为用户提供区分服务。首先在请求处理层进行系统辨识和比例延迟保证控制器设计。

假定改进后的数据库连接池，可以等效为一个线性模型。该被控模型的输出是两类客户请求的平均延迟比$\frac{d_B}{d_A}$，输入是两类客户请求队列获取下一个空闲数据库连接的概率比$\frac{p_B}{p_A}$。假定 $m$ 阶线性模型能达到较好的近似程度，那么相应的差分方程可记为：

$$Y(k)=\sum_{i=1}^{m}[a_iY(k-i)+b_iX(k-i)] \tag{1}$$

其对应的 $Z$ 域传递函数为：

$$G(z)=\frac{Y(z)}{X(z)}=\frac{\sum_{i=1}^{m}b_iz^{m-i}}{z^m-\sum_{i=1}^{m}a_iz^{m-i}} \tag{2}$$

### 3.1 系统辨识

按 4.1 配置试验平台，数据库连接池中连接总数目设为 20。以请求的源 IP 作为分类标准。两台客户机分别模拟 50 个客户，利用 SURGE 向服务器发送动态请求。服务器在第 $k$ 个采样时刻，计算采样周期(15s)内 A,B 类客户(假定 A 类为高优先级)的平均延迟比 $Y(k)$，并根据伪随机二位式序列当前的值，将不同类别客户请求下一周期获取空闲数据库连接的概率比 $X(k)$ 设置为 1 或 1.5。输入的伪随机二位式序列按如下公式计算：

$$\varepsilon(k)=\varepsilon(k-p)+\varepsilon(k-q)(\mathrm{mod}\ 2) \tag{3}$$

当 $p$ 取 8、$q$ 取 5 时，得到周期为 255 的输入序列。当 $\varepsilon(k)=1$ 时，调整 $X(k)$ 为 1；当 $\varepsilon(k)=0$ 时，调整 $X(k)$ 为 1.5。试验共进行了 40min。所有试验数据均由控制线程写到服务器的日志文件。根据递推最小二乘估计[16]的思想，对于此 $m$ 阶系统，设由前 $N+m$ 次采样数据得到的参数的最小二乘估计为 $\theta_N$，则在第 $N+m+1$ 次采样后，$\theta_N$ 可以修正为

$$\theta_{N+1}=\theta_N+\frac{P_N\varphi_{N+1}}{\varphi_{N+1}^TP_N\varphi_{N+1}+1}[Y(N+m+1)-\varphi^{N+1}\theta_N] \quad (N>0) \tag{4}$$

其中：

$$P_{N+1}=P_N-\frac{P_N\varphi_{N+1}\varphi_{N+1}^TP_N}{\varphi_{N+1}^TP_N\varphi_{N+1}+1} \tag{5}$$

$$\varphi_N^T=(Y(N+m-1),\cdots,Y(N),X(N+m-1),\cdots,X(N)) \tag{6}$$

取 $\theta_0=0$，$P_0=15I$，可得到系统参数的估计值：$\hat{\theta}^T=(\hat{a}_1,\cdots,\hat{a}_m,\hat{b}_1,\cdots,\hat{b}_m)$。估计值与真值之间存在误差，可定义损失函数：

$$J(m)=\sum_{k=m+1}^{m+N}\{y(k)-\sum_{i=1}^{m}[a_iY(k-i)+b_iX(k-i)]\} \tag{7}$$

采用 F 检验法确定系统的阶数。设 $n_1$ 和 $n_2$ ($n_1<n_2$) 是模型

的两个近似阶数。构造统计量：

$$F=\frac{J(n_1)-J(n_2)}{J(n_2)}\cdot\frac{N-2n_2}{2(n_2-n_1)} \tag{8}$$

其中 $N$ 为样本数目，$2n_1$ 和 $2n_2$ 分别对应系统的阶数为 $n_1$ 和 $n_2$ 时系统参数的数目。文[13]证明：当 $N$ 充分大且 $n_2>n_1\geqslant m$ 时，$F$ 服从分布 $F(2(n_2-n_1),N-2n_2)$。假设 $1\leqslant m\leqslant 6$，按(4)(7)(8)式分别计算 $\hat{\theta}^T$，$J(m)$ 以及 $F$ 的值。取置信度 $\alpha=5\%$，则 $F_{0.05}(2,86)=3.10$。因为 $F_{0.05}(2,86)>F$，即 $m$ 取 1 和 2 时 $J(m)$ 已无显著变化，所以 $m=1$ 时，(1)式即达到了足够的近似程度。因此原系统为一阶线性系统，其中系统参数 $\hat{\theta}^T=(-0.0172,0.7463)$。

### 3.2 控制器设计

由于积分环节能减小闭环系统的稳态误差，且 PI 控制器较便于在程序中实现，因此将 PI 控制器作为设计目标。控制模型如图 6 所示。根据 Web 应用服务器实际工作状况，将闭环系统的性能指标定为：稳态误差：$e_{ss}=0$；调节时间：$t_s<300$s。

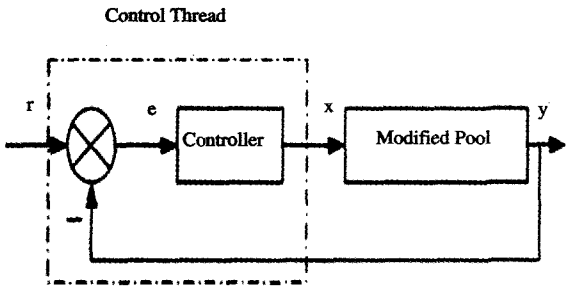


图 6 比例延迟保证控制模型(请求处理层)

PI 控制器的 Z 域传递函数为

$$D(z)=K_P+\frac{K_IT(z+1)}{2(z-1)}=\frac{(2K_P+K_IT)z+K_IT-2K_P}{2(z-1)} \tag{9}$$

由(2)、(7)式，闭环系统的传递函数为

$$G_C(z)=\frac{D(z)G(z)}{1+D(z)G(z)} \tag{10}$$

利用 MATLAB[17] 极点配置工具，将闭环极点选取(0.767，−0.237)在则控制器相应参数值为：$K_P=0.10$；$K_I=0.0213$。采用增样式控制算法，则有

$$\Delta x(k)=x(k)-x(k-1)$$

$$=K_P\{(1+\frac{TK_I}{K_P})e(k)-e(k-1)\} \tag{11}$$

故：$\Delta x(k)=0.42e(k)-0.1e(k-1)$ (12)

易证闭环系统达到了预期的性能指标。

由于在连接管理和请求调度两个层次上均采用了双队列和资源池的模型，因此以上设计的 $PI$ 控制器也可用于实现连接管理层闭环系统的比例延迟保证。

## 4 闭环系统仿真

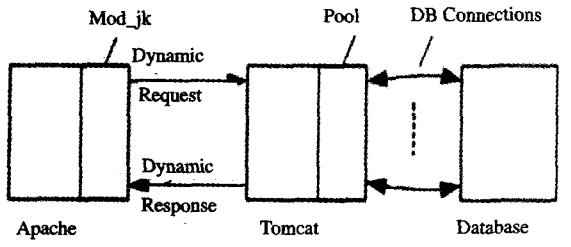


图 7 服务器系统结构

### 4.1 Test-bed

在 Test-bed 中，所有 PC 机均配置有 pentium4 2.8GHz 处理器，512 M 内存。改进的 SURGE[9] 作为测试工具，运行在 Redhat 9 上，向 Web 服务器发送动态请求。Web 服务器为 Apache(版本为 httpd2.0.53)，静态请求可由 Apache 直接处理。在 Apache 内，Mod_jk 模块将动态请求按一定方式(如加权方式)分派给后端的多个应用服务器，以实现负载均衡。Web 应用服务器为 Tomcat(源代码版本为 5.5.17)。由于本文主要目的不是 Web 负载均衡问题，故在试验中只使用一台 Web 应用服务器。后端数据库为 Oracle 9i。

### 4.2 动态负载的分布

对于 HTTP 静态请求，可以根据请求所对应文件的大小估算请求的处理时间。而动态请求通常并不与静态文件关联，而且需要利用较多的系统资源，因此难以统计其处理时间的分布。本文使用两种属于不同分布的动态负载，测试控制器作用下闭环系统的性能，两种负载大小的均值都为 350ms。第一种负载的大小服从均匀分布，即假定每个请求的处理时间为来自 U(0,700)的一个样本；第二种负载的大小服从重尾分布，假定有界帕累丝分布(bounded-perato，其累积分布函数如式(13))为实际的重尾分布模型。另外由于双峰分布(bi-modal，其密度函数如式(14))在 $B\gg E(X)>A$ 且 $\alpha\approx 1$ 时，具有明显的重尾特征，并且能很好地近似其它类型的重尾分布如有界帕累丝分布等[12]，因此本文在实际程序中使用双峰分布生成第三种负载，并取 $A=50$，$B=6050$，$\alpha=0.95$，以保证负载的均值为 350ms。在程序设计时，使用服务线程对象的函数成员 sleep( )模拟实际的请求处理时间。

$$F(x)=\frac{1-(m/x)^\gamma}{1-(m/M)^\gamma}\text{，其中 } M\gg m \tag{13}$$

$$f(x)=\alpha\delta(x-A)+(1-\alpha)\delta(x-B)$$

$$\text{其中}\quad \delta(x)=\begin{cases}1,x=0\\0,\text{其它}\end{cases} \tag{14}$$

### 4.3 试验配置

在 TCP 连接管理层和请求处理层，参考值均设定为 0.5，即高优先级客户的平均延迟为低优先级客户的一半。数据库连接数设定为 15。服务器服务线程数目设定为 150。所有试验均进行 1800s。试验中共使用四台客户机模拟两类客户，如表 1。其中第三、四台客户机在 600s 开始运行，使得两类客户客户数分别增加 100 个。控制器在客户机启动 10s 后开始运行，以保证客户机都已启动完毕。所有试验结果均由控制线程写到日志文件。

所有试验开始时，在连接管理层，每类客户均分配 75 个工作线程；在请求处理层，pa 设置为 0.5，即分派器分别按 50%的概率将空闲的数据库连接分别分派给两类客户请求。

表 1 比例延迟保证试验客户机设置

| 客户机 | 优先级 | 客户数目 | 启动时刻/sec |
|---|---|---|---|
| 1 | 高(A) | 50 | 0 |
| 2 | 低(B) | 50 | 0 |
| 3 | 高(A) | 100 | 600 |
| 4 | 低(B) | 100 | 600 |

### 4.4 试验结果及其分析

#### 4.4.1 连接管理层的比例延迟保证

如图 8，9 所示，在连接管理层，前 600s 中由于每类工作线程数均多于该类并发的客户连接数，故客户的连接延迟为

0.600s时，并发客户连接数急剧增加，且每客户连接数均多于该类工作线程数，故两类客户的平均连接延迟有显著增加，但在控制器的作用下，其比值较好地保持在1∶2。

4.4.2 请求处理层的比例延迟保证

如图8,9所示，在请求处理层，由于数据库连接池中的数据库连接总数始终小于并发的客户请求数，故两类客户请求的平均延迟始终存在，但在控制器的作用下，其比值较好的保持在1∶2。

4.4.3 负载分布对控制器的影响

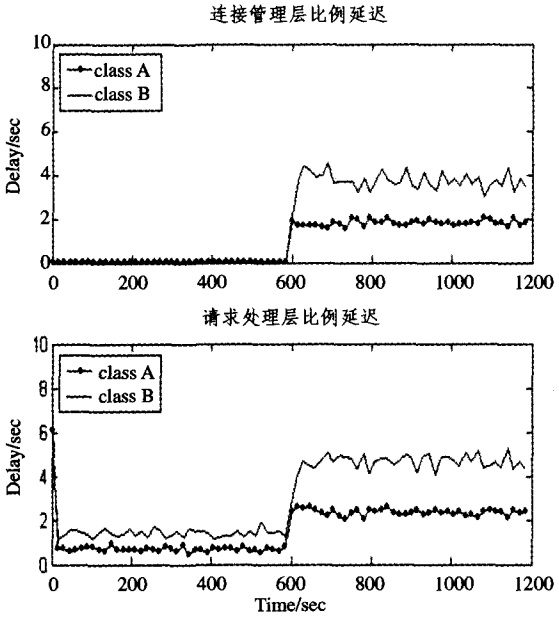


图8 均匀负载下的比例延迟保证

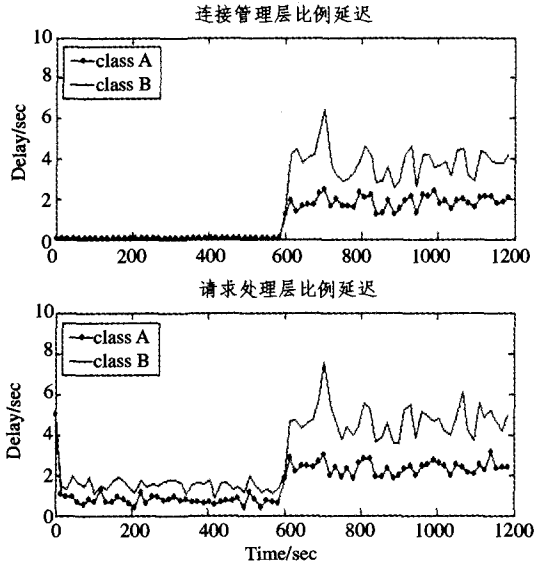


图9 重尾负载下的比例延迟保证

由图8,9可知，对于服从不同分布的动态负载，闭环系统在控制器的作用下，都能较快达到新的平衡，实现比例延迟保证。但是与均匀负载相比，重尾负载由于大小样本差别较大(如式(12)；$B:A=121$)，因此两类客户的平均延迟波动较大。对于服从均匀分布的动态负载，闭环系统仍然存在一定的稳态误差，主要有两方面原因：第一是请求间隔的波动。SURGE测试工具使用并发的ON/OFF过程引发具有自相似特点的网络流量[10]，其中OFF部分主要用于模拟用户思考时间(Think Time)，它也服从重尾分布，因此测试负载也存在波动。第二是资源调节的粒度。在连接管理层，控制器调节的是数目有限的工作线程。在请求处理层，分派器根据一定概率，分派的是单位采样周期内的空闲数据库连接，而其数目同样是离散的，有限的。

**结束语** 本文首先分析了Web应用延迟的构成，然后分别在TCP连接管理层和HTTP请求处理层建立了基于反馈控制的比例延迟保证模型，并针对动态请求，设计了多种测试负载。由于本文的改进方法完全在服务器软件中实现，因此便于应用到Apache和Tomcat之外的其它服务器中。试验结果表明，在连接管理层，控制器作用下的Apache服务器，能保证不同类别的客户连接享受到不同的延迟；在请求处理层，Web应用服务器所在的闭环控制系统则能保证不同类别的客户请求申请数据库连接时平均延迟比保持不变。因此整个服务器系统能够较好地为客户提供区分服务。

## 参 考 文 献


1 Banga G, Druschel P. Measuring the Capacity of a Web Server. In: Proceedings of the USENIX Symposium on Internet Technologies and Systems. Monterey, California, December 1997

2 Abdelzaher T F, Stankovic J A, Lu Chengyang, et al. Feedback Performance Control in Software Services. IEEE Control Systems, 2003, 23(3)

3 Lu Chengyang, Abdelzaher T F, Stankovic J A, et al. A Feedback Control Approach for Guaranteeing Relative Delays in Web Servers. In: Proceedings of IEEE Real-time Technology and Applications Symposium, TaiPei, Taiwan, June 2001

4 Chandra A, Pradhan P, Tewari R, et al. An Observation-based Approach Towards Self-managing Web Servers. Computer Communications, 2005. 1～15

5 lea D. Concurrent Programming in Java: Design Principles and Patterns Second Edition. Addison Wesley Longman Inc, 2000

6 Harchol-balter M, Schroeder B, Bansal N, et al. Size-based Scheduling to Improve Web Performance. ACM Transactions on Computer Systems, 2003, 21(2): 207～233

7 Wei Jianbin, Xu Cheng-Zhong. Design and Implementation of a Feedback Controller for Slowdown Differentiation on Internet Servers. WWW 2005. Chiba, Japan

8 Kanodia V, Knightly E W. Ensuring Lantency Targets in Multiclass Web Servers. IEEE Transaction on Parallel and Distributed Systems, 2002, 13 (10)

9 Barford P, Crovella M E. Generating Representative Web Workloads for Network and Server Performance Evaluation. Measurement and Modeling of Computer Systems, 1998. 151～160

10 Crovella M E, Bestavros A. Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes. IEEE/ACM Transactions on Networking, 1997, 5(6): 835～846

11 Diao Yixin, Hellerstein J L, Parekh S S. Using fuzzy control to maximize profits in service level management. IBM Systems Journal, 2002, 41(3)

12 Psounis K, Molinero-Fernández P, Prabhakar B, et al. Systems with multiple servers under heavy-tailed workloads [R]. Performance Evaluation, 2005, 62(7): 456～474

13 Astrom K J, Eykhoff P E. System Identification-A Survey. Automation , 1971, 7: 126～162

14 The Apache Software Foundation. http://www. apache . org

15 Dovrolis C, Stiliadis D, Ramanathan P. Proportional Differentiated Services: Delay Differentiation and Packet Scheduling. In: Proceedings of the ACM SIGCOMM 1999

16 王秀峰，卢桂章. 系统建模与辨识. 北京：电子工业出版社，2004

17 张志涌，徐彦，等. MATLAB教程. 北京：北京航空航天出版社，2001