Apache Web服务器中基于LTI模型的多服务类别比例延迟控制

吕健波[1] 戴冠中[1] 潘文平[2]
(西北工业大学自动化学院控制与网络研究所 西安 710072)[1]
(南京航空航天大学自动化学院 南京 210016)[2]

摘 要 通过系统辨识建立了Apache Web服务器的线性时不变(LTI)模型,用于描述两类Web客户连接的连接延迟比与服务线程比间的关系,并针对此LTI模型,设计了一个控制器。此控制器通过动态分配服务于不同类别连接的服务线程的数量,可实现Web服务器端的比例延迟保证,并进一步将此区分服务的控制模型扩展到了面向多个连接类别,通过多个控制器实现了相邻两个客户连接类别的比例延迟保证。仿真表明,即使过载状态下并发客户连接的数目急剧变化,闭环系统中的Web服务器也能为多个客户连接类别提供比例延迟保证。

关键词 Web服务器,排队论,反馈控制,比例延迟保证
中图法分类号 TP393 **文献标识码** A

LTI Model-based Proportional Delay Control for Multiple Service Classes on Apache Web Server

LU Jian-bo[1] DAI Guan-zhong[1] PAN Wen-ping[2]
(Control & Networks Institute, College of Automation, Northwestern Polytechnical University, Xi'an 710072, China)[1]
(College of Automation Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)[2]

Abstract The linear time invariant (LTI) model for Apache Web server was identified experimentally. This model was used to describe the relationship between connection delay ratio and service thread ratio for two classes of Web client connections. Based on this model, a controller was designed to implement proportional delay guarantee by adjusting the number of service thread for different class of connections. An extension for multiple connection classes was implemented by using multiple controllers. Simulation results indicated that, Apache Web server in the closed-loop system guaranteeds the proportional delay for different connection classes quite well, even if the number of concurrent client connections changes abruptly under heavy load conditions.

Keywords Web Server, Queuing theory, Feedback control, Proportional delay guarantee

据统计,在遍布全球的Web站点中,Apache Web服务器的覆盖率接近70%[1],由Web应用引发的数据流量超过了整个Internet网络流量的60%。Aapche服务器是Internet上使用最为广泛的Web服务器,其性能成为了影响整个Internet网络性能的关键因素之一,而Apache服务器工作环境的开放性、工作负载的不确定性都加大了对其性能进行评价、调节和改进的难度。近年来,许多研究者都试图为Web服务器建立等效的数学模型,进而达到调节和改进其性能的目的。本文先分析了Apache服务器的体系结构、核心模块和几种等效模型及其在Aapche服务器性能调节与改进中的应用,然后针对Windows平台下改进后的Apache服务器,建立起了等效的LTI模型,并在此基础上设计控制器,构造闭环控制回路,达到Apache服务器上的比例延迟控制,从而提升了过载状态下Apache服务器的服务性能。

1 Apache服务器体系结构及其多处理模块

Apache服务器的体系结构[5,9]如图1所示:CORE规定了HTTP请求处理的各种规则和各个步骤,APR是Apache服务器对操作系统相关调用的封装,各个MOUDLE则对应于HTTP请求处理的各个阶段,服务器在启动时根据配置文件的设置加载并初始化相应的MOUDLE。

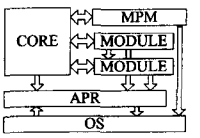

图1 Apache服务器体系结构

多处理模块MPM(MULTIPLE PROCESSING MODULE)是整个Apache的核心,并直接与操作系统相关联。针对不同的操作系统,Apache服务器采用不同的MPM。Apache服务器MPM模块的类型主要有以下几种:多进程模型、多线程模型、多进程和多线程复合模型等。线程或进程是Apache服务器的连接管理单元,通常一个线程或一个进程对应于一个客户连接,相关请求和响应通过此连接传递。多进程模型尽管因为频繁的进程切换需要保存CPU状态等信息

到稿日期:2010-01-20 返修日期:2010-04-05
吕健波(1970-),男,讲师,主要研究方向为Web QoS、GIS QoS等,E-mail:jianbolu@hotmail.com。

而效率有所降低，但一个进程的异常不会影响到其它的进程，因此具有高可靠性。由于特定进程下的多线程都处于该进程的地址空间内，因此多线程模型效率较高，但这种模型的MPM模块必须保证绝对线程安全才能够构建健壮的Apache服务器系统。多进程和多线程混合模型则具有健壮性和可扩展性两种优点，并且可以通过进程和用户组之间的映射为在单一服务器上部署多个虚拟主机提供可能。

对Apache服务器建模，可以针对Apache服务器整体，也可以只针对其核心，即MPM模块。本文在客户连接管理层实现区分服务，因此针对Apache服务器的MPM模块建模。

2 Apache服务器建模

在充分了解Apache服务器体系结构的前提下，建立Apache服务器的数学模型具有多方面的作用。首先，服务器运行在一个开放的环境中，特定数学模型的建立有助于准确地预测服务器的工作状况并由此作出必要的调整以合理地利用服务器端的系统资源。其次，Apache服务器的某些系统参数设置不当会严重影响其服务性能，建立合适的系统模型，通过动态的参数设置，有助于提高服务器的服务性能并达到合理的资源利用率。另外，基于服务器的Web QoS机制也引起了人们的研究兴趣，相关数学模型的识别和应用有助于实现服务器端的特定的区分服务和性能保证。

理解Apache服务器的工作方式是建立Apache服务器的数学模型的基础。以Windows平台下的Apache服务器为例，Apache服务器通过维护一个线程池同时服务于多个客户，在HTTP/1.1下，每个客户通过持续连接(persistent connection)访问服务器的过程可以作以下简单描述：客户端浏览器通过三次握手与服务器建立一个TCP连接，服务器将此连接分配给一个空闲的服务线程，客户通过此连接向服务器发送请求，服务线程解析此请求并返回响应，此后服务线程维持此TCP连接以等待新的请求的到来。如果服务线程在一定时间内未收到新的请求，服务器将主动关闭此连接。同一用户能够与服务器建立的并行连接的数量上限则由各浏览器设定。

同一时段内访问服务器的客户数量和每个客户每次接受服务的时间的随机性，以及服务器先到先服务的特点，为建立基于排队论的Apache服务器的数学模型提供了条件。

另外，Apache服务器多个可调的参数(如最大并行用户数MaxClients，活动连接维持时间KeepAliveTimeOut等)与服务器服务性能(如请求完成率等)及操作系统的工作状况(如系统资源利用率、吞吐量等)形成的类似输入输出的关系为建立基于控制理论的Apache服务器的数学模型提供了条件。此外，通过对连接队列、请求队列、服务进程或服务线程等的综合动态管理，实现相应的性能保证也能促进对Apache服务器控制模型的研究，以建立合适的Apache服务器的模型。以下分别从排队论和控制理论两个角度讨论Apache服务器建模问题。

2.1 基于排队论的服务器模型

根据Apache服务器的工作特点，文献[7]将Aapche服务器等效为一个M/G/1/K * PS排队系统(见图2)，即排队系统的输入(即请求的输入率)为泊松分布(输入率为λ)，服务时间为一般分布，服务台(CPU)数量为1台，由多个并行接受服务的客户分时占用，系统容量为K。根据此排队模型，由排队论的相关理论，可以将系统的阻塞率、吞吐量、平均逗留时间分别记为：

阻塞率(即系统饱和的概率，其中$\rho=\lambda\bar{X}$为服务率)：

$$P_b=P[N=K]=\frac{(1-\rho)\rho^K}{1-\rho^{K+1}}; \tag{1}$$

吞吐量(以请求为单位)：

$$H=\lambda(1-P_b) \tag{2}$$

平均逗留时间(即平均响应时间)：

$$T=\frac{E[N]}{H}=\frac{\rho^{K+1}(K\rho-K-1)+\rho}{\lambda(1-\rho^K)(1-\rho)} \tag{3}$$

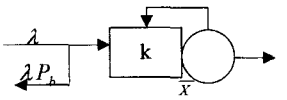

图2 Apache服务器等效M/G/1/K * PS排队模型

在给定输入率λ的情况下，根据大量的实测数据，按照最大似然估计法，可以求得此排队系统的两个参数$\bar{X}$和K。仿真显示，K值与Apache服务器的系统参数MaxClients一致，因此假定的排队模型是有效的。由此在已知MaxClients参数的情况下，根据以上的排队系统模型，就能很好地预测系统的阻塞率、吞吐量、逗留时间等，这就使得服务器管理人员能够根据服务器所在操作系统的系统配置，合理地进行相关参数的设置，有效利用系统资源，改善服务器服务性能。

另外文献[12]将多进程的Apache服务器等效为一个G/G/M排队系统，其中M为服务进程数，系统的输入以会话为单位。在此基础上，分析了不同连接管理策略(即HTTP1.0下的非持续连接，HTTP1.1下的多个并行持续连接以及由浏览器支持的及早关闭持续连接)对客户端请求的反应时间和响应时间的影响。仿真表明，排队系统的理论结果与数值仿真结果非常一致。

但是，服务器系统的排队模型通常假设输入过程为泊松过程，即任务到达间隔时间独立同分布，均服从负指数分布。在实际的Internet网络中，HTTP请求序列的间隔被证明服从重尾分布[11]，且重尾指数在区间(1,2)。

2.2 基于控制理论的服务器模型

将控制方法应用于软件系统，实现软件系统上的区分服务和性能保证也引起了人们的研究兴趣。文献[10]针对Lotus Notes邮件服务器，通过系统辨识确定了其受控模型，设计了一个控制器，用以最大化服务器的吞吐量并最小化客户端的反应时间，取得了良好的效果。确定Apache服务器的等效控制模型需要解决以下的问题：

所选的系统输入输出量是否可测的；

系统是线性的还是非线性的；

系统是线性时不变还是线性时变系统；

系统的阶次为多高才可达到必要的精度；

系统参数的估计方法如何选择。

为控制器设计以及程序实现的方便，通常先假定被辨识的系统为线性时不变的系统，再使用通过辨识得到的模型验证假设的成立。以下分别从SISO和MIMO模型两方面加以分析。

2.2.1 单输入单输出(SISO)模型

Web流量的增长，使得网络的QoS机制不足以为客户提

供有效的性能保证，如何利用操作系统的内在机制或通过改进 Web 服务器软件，实现基于操作系统或基于 Web 服务器软件的区分服务和性能保证引起了人们的研究兴趣。基于操作系统改进的方法需要深入了解操作系统的内核机制以作出相应修改[2]，并同时修改服务器软件的相关调用部分，所以难度较大，而基于服务器软件改进的方法则简便易行。为了向不同的客户提供不同的连接延迟，文献[3,4]将修改后的 Apache 服务器的 MPM 模块等效为一个单输入单输出的控制模型，其输入为不同类别客户的线程比，输出为不同类别客户的连接延迟比，在此基础上实现了一个闭环控制系统。在控制器的作用下，Apache 系统能根据上一采样周期内不同类别客户的延迟比，调节服务于不同类别客户的线程的数量，从而实现相对延迟保证。本文则在此基础上实现了 Windows 平台上的比例延迟保证并将此区分服务的控制模型扩展到了面向多个服务类别。

2.2.2 多输入多输出(MIMO)模型

Web 流量的突发性、Web 应用的多样性等诸多因素，使得 Apache 服务器参数设置成为了影响 Apache 服务器服务性能的关键因素之一。而现有的各种商用服务器中，许多服务器参数是在其配置文件中设定的，不能根据服务器的工作状态动态改变。文献[6]从优化 Apache 服务器端操作系统的 CPU 利用率和内存利用率出发，提出了 Apache 服务器的多输入多输出控制模型。该模型的输入是 Apache 服务器的两个参数：MaxClients 和 KeepAliveTimeOut。在给定的 CPU 利用率和内存利用率的参考值下，通过控制器作用，Apache 服务器根据操作系统的实际状态，动态调节两个参数的大小，达到指定的资源利用率。假定一阶系统即为理想的目标系统。在系统辨识阶段，系统的输入 MaxClients 和 KeepAliveTimeOut 采取离散正弦波的形式，采样时间设定为 5 秒，根据试验得到的数据，利用最小二乘估计法进行参数估计，得到被控对象的传递函数为：

$$\begin{bmatrix} cpu_{k+1} \\ mem_{k+1} \end{bmatrix} = \begin{bmatrix} 0.53 & -0.109 \\ -0.0256 & 0.630 \end{bmatrix} \begin{bmatrix} cpu_k \\ mem_k \end{bmatrix} + 10^{-4} \begin{bmatrix} -84.5 & 4.39 \\ -2.48 & 2.81 \end{bmatrix} \begin{bmatrix} KA_k \\ MC_k \end{bmatrix} \quad (4)$$

分别采用极点配置方法和 LQR 方法，设计控制器。仿真结果表明，在控制器的作用下，闭环系统能使操作系统的 CPU 利用率和内存利用率在 Apache 服务器轻载或重载的状况下都能迅速达到相应的参考值。因此由系统辨识得到的等效控制模型达到了足够的精度，而且控制器是有效的。

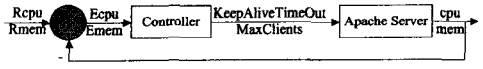

图 3 改进后 Apache 服务器的等效 MIMO 控制模型

2.3 相关问题

与基于排队论的方法相比，用反馈控制方法对 Apache 服务器建模有以下优点：首先，控制模型通常不需要关注 Web 负载的分布特点，在负载强度出现波动比如输入过程不满足泊松分布时，控制模型同样有效，而此时经典的排队模型则并不适合。其次，控制模型可以同时关注多个性能指标，比如通过建立 MIMO 模型可以发现多个系统输入和输出间的相互影响，这是排队模型难以实现的。

但是通过建立服务器的控制模型实现服务器的性能改进存在以下问题：首先，模型的精度有待进一步提高。系统辨识确定的模型的精度直接影响到控制器的设计，也就影响到了闭环系统的性能。其次，需要最小化离散的系统变量给闭环系统带来了影响以及控制环节给服务器系统增加了额外负载压力。比如 Apache 服务器的服务线程数量是离散的。最后，需要选择合理的采样时间。比如采样频率过高，控制线程因频繁地使用系统资源如 CPU 时间等也会影响到闭环系统的性能，所以必须考虑到采样时间的影响。

3 基于 LTI 模型的比例延迟保证

3.1 比例延迟保证的定义

根据文献[3,4]，在请求连接的客户数目多于服务器设定的服务进程或服务线程数的情况下，将客户程序与服务器程序建立连接的时刻与服务器程序将此连接分配给空闲的工作线程的时刻分别记为 t_c 和 t_d，则该客户的连接延迟时间可以记为 $t_{delay}=t_d-t_c$。对于不同优先级的客户，根据客户的应用需求不同，其连接延迟时间也应有所不同，比例延迟保证功能为不同优先级的客户提供不同的延迟比，以保证优先级高的客户尽快得到服务器的响应。如果 A 类客户的延迟时间是 t 秒，而 A，B 两类客户的延迟比例是 1∶2，则 B 类客户的延迟时间分别是 $2t$ 秒。

3.2 Apache 服务器 MPM 模块的改进及其等效 LTI 模型

在 Windows 平台下，Apache 服务器的 MPM 模块采用单完成端口形式。空闲的工作者线程形成空闲栈，监听线程将接受到的客户连接置入完成端口，激活栈顶的工作者线程，工作者线程在处理完该客户连接断开后重新回到栈顶。完成端口的作用类似于一个 FIFO 队列。改进后的 MPM 模块采用双队列的形式，并增加分类器和控制线程，如图 4 所示。分类器根据客户连接的源 IP 将不同类别的连接置入不同的队列，而控制线程则根据上一个采样周期内不同类别客户的延迟比，调节服务于不同类别客户的线程的数量。其等效控制模型如图 5 所示，r 为指定的延迟比例，x 和 y 分别为两类客户的服务线程比和实际延迟比。

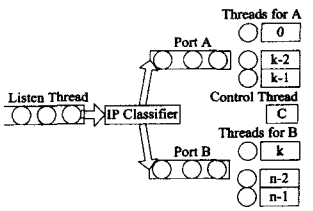

图 4 改进后的 Apache MPM(WinNT)模型

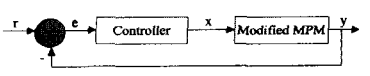

图 5 改进后 Apache 服务器的等效 SISO 控制模型

3.3 系统识别与控制器设计

假设系统可以等效为一个线性模型，且二阶线性模型可以达到较高的精度。相应的差分方程为：

$$Y(k)=aY(k-1)+bY(k-2)+cX(k-1)+dX(k-2) \quad (5)$$

系统辨识中，服务器端配置为：处理器 pentium 4 2.8 GHz，内存 512MB，操作系统 Windos XP；客户机的配置为：处理器 pentium 4 2.8GHz，内存 512MB，操作系统 RedHat 9。服务器端使用改进的 Apache 服务器。客户机使用请求发生

器 SURGE1.0[11] 作为测试的工具。SURGE 使用 UE(User Equivalent)线程模拟实际的 Web 客户，每个 UE 是一个 ON/OFF 过程。多个并发的 UE 同时向 Web 服务器发送请求，可引发具有自相似特征的网络流量。并发 UE 的数目决定了服务器端的负载强度，并发 UE 数量的变化即可引起服务器端所承受的负载强度的变化。首先，利用 SURGE1.0 在 Apache 的文件目录下生成 2000 个被请求文件。然后在 Windows XP 下运行改进的 Apache 服务器，最大并行连接数设置为 100。接着另外两台客户机在 Linux 下启动 SURGE1.0。两台客户机分别模拟 A，B 两类客户(各 90 个)运行 2400 秒，从服务器上请求文件。选取周期为 255 的伪随机二位式序列[8]：

$$\varepsilon(k)=\varepsilon(k-n)+\varepsilon(k+k_0)(\mathrm{mod}\ 2) \tag{6}$$

式中，n 取 8，k_0 取 5。当 $\varepsilon(k)=1$ 和 0 时，分别将不同类别的工作者线程比例调整为 2∶1 和 1∶1。记录每一采样时刻的线程比 X 和延迟比 Y。采样周期为 15 秒。

$$\begin{bmatrix} Y(3) \\ Y(4) \\ \vdots \\ Y(k) \end{bmatrix} = \begin{bmatrix} Y(2) & Y(1) & X(2) & X(1) \\ Y(3) & Y(2) & X(3) & X(2) \\ & \vdots & & \ddots \\ Y(k-1) & Y(k-2) & X(k-1) & X(k-2) \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \tag{7}$$

由最小二乘估计法[8]：若 $E(A)=B\beta$ 且 $Var(A)=\sigma^2 I$，则 $\beta_{LS}=(B'B)^{-1}B'A$。由此推得系统参数为：

(0.7360，−0.611，0.9309，−0.1071)

根据服务器的实际工作情况，将闭环系统的性能指标确定为：稳态误差 $e_s=0$；调节时间 $t_s<360$sec。利用 MATLAB 下的根轨迹工具，根据闭环系统的性能指标，将闭环控制系统的极点选择在(0.687，0.386±i0.613)，可得到相应 PI 控制器的参数 $K_P=0.1566$，$K_I=0.0191$。

3.4 程序具体实现

按图 4，闭环控制系统对源程序(本文所用源代码版本为 httpd-2.0.53)的改进主要有：

(1)相关数据结构

```
typedef struct CompContext
{ …
  SOCKET accept_socket;
  struct sockaddr * sa_server;
      struct sockaddr * sa_client;
  apr_time_t time_from;//当前客户连接建立时间
      apr_time_t time_to; //当前客户连接被分配时间
} COMP_CONTEXT;//改进的客户连接相关信息
struct QueueTime//新增结构，记录每类客户单位采样周期内的延迟
{
  apr_time_t time_all; //本类客户总延迟时间
  int connection_counter;//本类客户总连接数
};
  iThreadStatus[n]//用于记录 n 个线程分别服务客户的类别。
```

(2)监听线程

获取到一个客户连接(AcceptEx())；

根据连接的相关信息填充一个 COMP_CONTEXT 结构类型的变量 context；

根据连接的源 IP 判断客户的类别；

由分类器将此 context 发送到相应的完成端口(PostQueuedCompletionStatus())。

(3)工作线程(假定为第 i 号线程)

确定本线程 i 当前服务的客户类别 iThreadStatus[i]；

在相应完成端口等待客户连接信息(GetQueuedCompletionStatus())；

从相应完成端口获取到一个客户连接的相关信息；

计算该客户延迟，本类客户数目增加一个，本类客户总延迟时间相应增加；

进入 Apache 其它模块，处理连接上的请求，直到超时或客户断开。

(4)控制线程部分

计算上一采样周期内不同类别客户的平均延迟比 Y，计算控制器输出 X；

根据 X 调整 iThreadStatus[n]中各项的值；

各类客户的总延迟时间和客户连接数重设为 0；

本线程睡眠 15 秒(sleep())。

3.5 闭环系统仿真

3.5.1 两服务类别比例延迟保证

按照表 1，初始时刻两类客户数量相等，在 1155 秒时低优先级的服务类别 B 的客户增加一倍。预设的延迟比为 0.5，即高优先级服务类别的延迟为低优先级的一半。如图 6 所示，在客户数量激增的情况下，Apache 服务器在控制器的作用下，达到新的平衡，实现了面向两服务类别的比例延迟保证。这与文献[3]中在 Linux 平台下获得的结论一致。因此 Apache 服务器的等效二阶 LTI 模型达到了足够的精度，根据此模型设计的控制器是有效的。

表 1 系统仿真配置 1

客户机	服务类别	客户数目	运行时段/秒
1	A 类	90	0～1700
2	B 类	90	1155～1700
3	B 类	90	0～1700

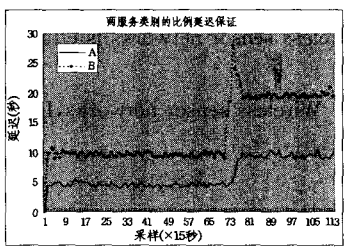

图 6 两个服务类别的比例延迟保证

3.5.2 多服务类别比例延迟保证

假设在 Web 服务器系统上客户连接可以划分为 n 个服务类别，则可将此 n 个类别划分为 $n-1$ 个重叠的组，其中第 j ($1\leqslant j\leqslant n-1$)组 G_j 包含第 j 类和 $j+1$ 类客户请求以及一个独立的控制器 C_j。针对组 G_j，可建立仅包含两类客户连接的控制模型，通过建立闭环控制回路，即可实现第 j 类和 $j+1$ 类客户连接的比例延迟保证。假设在第 k 个采样时刻，控制器 C_j 的输出为 $x_j(k+1)$，则通过求解以下方程组，即可得到在第 $k+1$ 个采样周期中，各类客户连接在下一采样周期中获取服务线程的比例 $P_j(k+1)(1\leqslant j\leqslant n)$。

$$\begin{cases} \sum_{j=1}^{n} P_j(k+1)=1 \\ \dfrac{P_j(k+1)}{P_{j+1}(k+1)}=x_j(k+1) \quad (1\leqslant j\leqslant n-1) \end{cases} \tag{8}$$

按照表 2，初始时刻 3 个服务类别客户数量分别为 50 个，在 300～750 秒时，优先级为中级的服务类别 B 的客户数

(下转第 52 页)

and Reflections, 2003: 287

[15] Robertson G G, Mackinlay J D, Card S K. Cone trees: animated 3D visualizations of hierarchical information[C]// ACM. New York. NY, USA, 1991

[16] Yee K P, Fisher D, Dhamija R, et al. Animated exploration of dynamic graphs with radial layout[C]// Citeseer. 2001

[17] Johnson B, Shneiderman B. Tree-maps: A space-filling approach to the visualization of hierarchical information structures[C]// VIS'91 Proceedings of the 2nd Conference on Visualization'91. Los Alamitos, CA, USA: IEEE Computer Society Press, 1991

[18] Zhao S, Mcguffin M J, Chignell M H. Elastic hierarchies: Combining treemaps and node-link diagrams[C]// Citeseer. 2005

[19] Balzer M, Deussen O. Hierarchy Based 3D Visualization of Large Software Structures [C]// Proceedings of the Conference on Visualization '04. IEEE Computer Society, 2004

[20] Nguyen Q V, Huang M L. EncCon: an approach to constructing interactive visualization of large hierarchical data[J]. Information Visualization, 2005, 4(1): 1-21

[21] Kennedy J. Exploring Multiple Trees Through DAG Representations[J]. IEEE Transactions on Visualization and Computer Graphics, 2007, 13(6): 1294-1301

[22] Graham M, Kennedy J, Downey L. Visual comparison and exploration of natural history collections [C]// Proceedings of the Working Conference on Advanced Visual Interfaces. ACM, 2006

[23] Sigman E, Wittenburg K. Visual Focusing and Transition Techniques in a Treeviewer for Web Information Access[C]// IEEE Symposium on Visual Languages(VL'97). 1997

[24] Furnas G W, Zacks J. Multitrees: enriching and reusing hierarchical structure[C]// ACM. New York. NY, USA, 1994

[25] Eades P, Wormald N C. Edge crossings in drawings of bipartite graphs[J]. Algorithmica, 1994, 11(4): 379-403

[26] Barth W, Junger M, Mutzel P. Simple and efficient bilayer cross counting[J]. Lecture Notes in Computer Science, 2002, 2528: 130-141

[27] Fruchterman T, Reingold E M. Graph drawing by force-directed placement[J]. Software-Practice and Experience, 1991, 21(11): 1129-1164

[28] Card S K, Mackinlay J D, Shneiderman B. Information visualization: Using vision to think[M]. San Francisco: Morgan Kauffman Publisher, 1999

(上接第 29 页)

目增加一倍，3 个类别的延迟比预设为 1∶2∶4。另外将系统服务线程的总数设置为 120 个。根据式(8)，为向 3 个服务类别提供比例延迟保证，需要使用两个控制器，构建两个控制回路。第一个控制器作用于 A、B 两个相邻类别；第二个控制器作用于 B、C 两个相邻类别。在每个采样时刻可以根据式(8)计算 A、B、C 3 个服务类获取服务线程的比例。由仿真结果(见图 7)可知，在两个控制器的作用下，Apache 服务器较好地为 3 个服务类别提供了比例延迟保证，达到了预期的目标。

表 2　系统仿真配置 2

客户机	服务类别	客户数目	运行时段/秒
1	A 类(高)	50	0～1200
2	B 类(中)	50	0～1200
3	C 类(低)	50	0～1200
4	B 类(中)	100	300～750

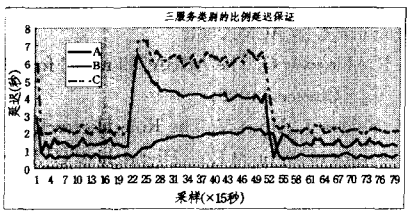

图 7　3 个服务类别的比例延迟保证

结束语　本文首先分析了 Apache 服务器的几种等效模型及其在服务器性能调节与改进方面的应用。队列模型适合于对服务器进行分析和对其各种指标做出预测，而控制模型则能实现服务器端参数的动态设置，并提供特定的区分服务和性能保证等。接着本文在 Window 平台上通过系统辨识和控制器设计，实现了闭环系统中 Apache 服务器上的面向多个服务类别的比例延迟保证。仿真表明系统辨识获得的 LTI 模型以及设计的反馈控制器都是有效的。如何选择合适的系统指标，建立更加精确的等效模型，实现服务器集群的性能改进，是下一步研究的重点。另外如何将队列的预测机制与控制方法相结合，实现基于会话或请求的准入控制，也是需要解决的问题。

参 考 文 献

[1] The netcraft web server survey [OL]. http://www.netcraft.com

[2] Voigt V, Tewari R, Freimuth D, et al. Kernel Mechanisms for Service Differentiation in Overloaded Web Servers[C]// Usenix Annual Techanical Conference. 2001

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

[4] Lu Chengyang, Abdelzaher T F, Stankovic J A, et al. A Feedback Control Approach for Guaranteeing relative Delays in Web Servers[C]// Proceedings of IEEE Real-Time Technology and Applications Symposium. TaiPei, Taiwan, June 2001

[5] Bloom R B. Apache Server 2. 0-The Complete Reference[M]. America: McGraw-Hill, Inc, 2002

[6] Gandhi N, Parekh S, Tilbury D, et al. Using MIMO feedback control to enforce policies for interrelated metrics with application to the Apache web server[C]// Proc. of the Network Operations and Management Symposium. 2002

[7] Cao Jianhua, Andersson M, Nyberg C, et al. Web Server Performance Modeling Using an M/G/1/K * PS Queue[C]// 10th International Conference on Telecommunication. Papeete, Tahati, 2003

[8] 郭齐胜，等. 系统建模原理与方法[M]. 长沙：国防科技大学出版社，2002

[9] The Apache Software Foundation[OL]. http://www.apache.org

[10] Gandhi N, Parekh S, Hellerstein J, et al. Feedback Control of a Lotus Notes Server: Modeling and Control Design[C]// American Control Conference. 2001

[11] Barford P, Crovella M. Generating Representative Web Workloads for Network and Server Performance Evaluation [J]. Measurement and Modeling of Computer Systems, 1998: 151-160

[12] Liu Zhen, Nicolas N, Villanueva J. Traffic Model and Performance Evaluation of Web Servers[J]. Performance Evaluation, 2001: 77-100