Web QoS中的预测控制与比例延迟保证

高 昂[1] 慕德俊[1] 胡延苏[1] 潘文平[2]
(西北工业大学自动化学院 西安 710072)[1] (南京航空航天大学自动化学院 南京 210016)[2]

摘 要 控制理论已被应用于Web服务器中,以改进其QoS性能。但当Web负载剧烈变化时,已有的基于反馈的比例延迟控制的实时性往往不佳。分析了HTTP 1.1请求页面中嵌入URL的个数和嵌入文件大小的重尾特征,根据Web服务器的排队特性以及HTTP1.1持续连接的特点,通过不同优先级等待队列的长度来预测排队延迟,从而调整相应的服务线程配额,实现相对比例延迟保证。实验证明,这种方法取得了良好的效果,显著提高了系统的动态性能。

关键词 Web QoS,预测控制,比例延迟保证,重尾分布
中图法分类号 TP393 **文献标识码** A

Predictive Control and Proportional Delay Guarantee in Web QoS

GAO Ang[1] MU De-jun[1] HU Yan-su[1] PAN Wen-ping[2]
(College of Automation, Northwestern Polytechnical University, Xi'an 710072, China)[1]
(College of Automation, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)[2]

Abstract Control theory is being used for QoS in Web server, but when workload varies significantly, the real-time performance of the feedback control needs to be improved. After analysing the heavy-tailed property of the number and size of embedded URL in the required pages, based on the queuing feature of the Web server and the HTTP 1.1 persistent connection, a predictive approach was desired to achieve the relative delay guarantee by calculating the future delay of the different priorities and adjusting their quota of service threads. The experimental results demonstrate that the proposed approach achieves a better dynamic performance.

Keywords Web QoS, Predictive control, Proportional delay guarantee, Heavy-tailed distribution

将经典反馈控制应用于Web服务器,以实现比例延迟服务[1](Proportional Delay Differentiated Service, PDDS),该方法近年来在Web QoS上得到了广泛的研究[3-5]。但这仅仅是一种被动的方法(reactive approach),控制器根据输出的偏差调整控制量,使得偏差尽可能为零。如果能够在系统发生偏差之前,对控制量进行调整,则可以提高系统的实时性。

本文针对Web静态页面,根据等待队列的长度对排队时间进行预测,调整不同优先级服务线程配额,实现比例延迟保证。从实验结果看,系统的动态性能相对于先前的反馈控制[3-5]有了显著的提高。

1 预测控制

1.1 概述

图1描述了Class i到达曲线(Arrival curve)和服务曲线(Service curve)以及等待时间之间的关系[2]。其中,到达曲线是对等待队列(Waiting queue)中TCP连接请求到达率的描述,服务曲线是对Web服务器处理Class i能力的描述。图中到达曲线与服务曲线在x方向的距离$\omega(k)$表示在k时刻Class i实际的延迟(本文对延迟做如下定义:$\omega(k)$是总延迟,指页面请求的TCP连接建立到页面传输完毕,连接关闭之间的时间),$\tilde{\omega}(k)$表示k时刻对下一时刻延迟的预测,y方向上的距离$n(k)$表示等待队列的长度。

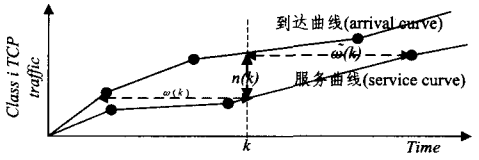


图1 排队延迟和等待队列

本文根据客户端的IP地址,将请求分为N个Class (Class i, $i=1,\cdots,N$, δ_i越小,优先级越高)。实验模型如图2所示。每类TCP连接进入对应的等待队列中,以FIFO的方式接受服务。队列观测器(Queue Monitor)观察队列长度,在任意k时刻,都可以获知该时刻等待队列i中的TCP连接个数$n_i(k)$。如果能够通过等待队列中$n_i(k)$的个数对$\tilde{\omega}(k)$做出预测,调整对应线程池中的线程配额,使式(1)成立,就可以提前对负载变化做出调整,从而保证两者之间的优先级比例。

$$\frac{\omega_{i+1}(k)}{\omega_i(k)}=\frac{\tilde{\omega}_{i+1}(k)}{\tilde{\omega}_i(k)}=\frac{\delta_{i+1}}{\delta_i},\ i=1,\cdots,N-1 \tag{1}$$

到稿日期:2009-02-25 返修日期:2009-05-13 本文受国防基础研究项目(C2720061361)资助。

高 昂(1984-),男,博士研究生,研究方向为网络QoS、网络化控制,E-mail:snailgao@gmail.com;慕德俊(1963-),男,教授,博士生导师,研究方向为计算机网络、信息安全;胡延苏(1985-),女,博士研究生,研究方向为网络化控制;潘文平(1980-),男,博士研究生,研究方向为计算机网络。

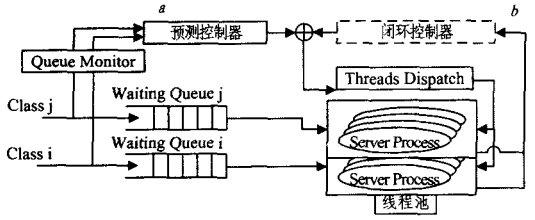


图 2　实验模型结构

1.2　页面大小

文献[6]将 Web 负载描述为大量 ON/OFF 源的叠加，从而产生具有自相似特性的 HTTP 请求序列，如图 3 所示。在每个 ON 过程中，客户端请求页面文件，每个页面文件可能包含多个嵌入的其它元素（如图片和语音文件）。对于支持 HTTP1.1 的客户端，这些嵌入文件（Embedded URL）通常是在一个持续连接内以 Pipeline 方式完成的。Inactive OFF 过程相当于客户端的思考时间，此时请求页面传输完毕，TCP 连接关闭。HTTP 请求序列实际上组成了整个页面，所以有

$$s_i^n=\sum_{j=1}^{y_i^n}x_i^j,n=1,2,\cdots,n_i(k) \tag{2}$$

其中，s_i^n 表示等待队列 i 中第 n 个请求页面的大小，等于该页面中的嵌入文件 x_i^j 的总和；y_i^n 表示第 n 个页面中的嵌入请求的个数。大量的研究表明[7,8]，Web 上传输的文件大小 x 具有重尾特性，其概率密度函数为

$$f(x)=\frac{\alpha k^\alpha}{1-(k/p)^\alpha}x^{-\alpha-1},k<x<p \tag{3}$$

其中，k 和 p 分别表示最小和最大的文件；$\alpha(0<\alpha<2)$ 为形状参数，决定分布函数拖尾的严重程度，通常考虑 $1<\alpha<2$ 的情况，此时随机变量的均值有限且方差为无穷大，符合网络通讯的实际情况。嵌入请求的个数 y_i^n 服从 Pareto 分布[6]，y_i^n 与 x_i^j 独立，x_i^j 相互独立，所以请求页面大小的数学期望为

$$E[s_i]=E[y_i]E[x_i],i=1,2,\cdots,N \tag{4}$$

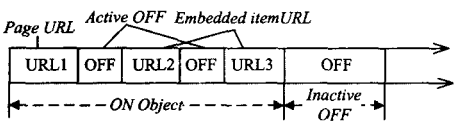


图 3　ON/OFF 模型

1.3　延迟估计

在 k 时刻，若等待队列中每个 TCP 连接之上承载的 HTTP 请求的页面的大小表示为 $\{s_i^1,s_i^2,\cdots,s_i^{n_i(k)}\}$，则要传输的页面总大小可以为

$$\Delta_i(k)=\sum_{n=1}^{n_i(k)}s_i^n=\sum_{n=1}^{n_i(k)}\sum_{j=1}^{y_i^n}x_i^j \tag{5}$$

从队尾看来，预测的总延迟 $\tilde{w}_i(k)$ 等于 $m_i(k)$ 个线程并发处理 $\Delta_i(k)$ 大小文件所耗费的处理时间。文献[9]指出，静态网页的处理时间和页面大小成正比，因此单个页面的处理延迟可以写成

$$T_i(s,m_i)=b(s/64)+cm_i \tag{6}$$

其中，b 表示传输系数，c 表示系统线程切换和同步的开销系数，m_i 表示该类请求的线程配额。此时，系统可以并发处理 m_i 个 i 类请求，b,c 的参数可以由线性回归获得，则预测的总延迟可以表示为

$$W_i(m_i(k),n_i(k))=(\sum_{n=1}^{n_i(k)}T_i(s_i^n,m_i(k)))/m_i(k) \tag{7}$$

$$\tilde{\omega}_i(k)=E[W_i(m_i(k),n_i(k))]=\frac{n_i(k)}{64}(\frac{bE[s_i]}{m_i(k)}+64c) \tag{8}$$

1.4　线程分配

将式(8)代入式(1)，有

$$\frac{\tilde{\omega}_{i+1}(k)}{\tilde{\omega}_i(k)}=\frac{\delta_{i+1}}{\delta_i}=\frac{n_{i+1}(k)m_i(k)}{n_i(k)m_{i+1}(k)}(\frac{bE[s_{i+1}]+64cm_{i+1}(k)}{bE[s_i]+64cm_i(k)}) \tag{9}$$

无论客户端优先级的高低如何，请求的服务器端文件在平均意义上是没有差别的，即 $E[x_i]=E[x_{i+1}]$。若令 $Z_i(k)=\frac{n_{i+1}(k)}{n_i(k)}$，$e=bE[s_i]$，$f=64c$，则式(9)可以写成

$$\frac{\delta_{i+1}}{\delta_i}=Z_i(k)(\frac{fm_{i+1}(k)m_i(k)+em_i(k)}{fm_{i+1}(k)m_i(k)+em_{i+1}(k)}) \tag{10}$$

其中，$Z_i(k)$ 是 k 时刻的观测量。式(10)可以写成 $m_{i+1}(k)=F_i(m_i(k),Z_i(k))$ 的形式。这里选择 Class 1 为参考，对应的 $\delta_1=1$，在第 k 个采样周期，依次求出 $m_i(k),i=2,\cdots,N$，代入 $\sum_{i=1}^{N}m_i(k)=M$，这里 M 为线程池中工作线程总和，从而求得满足式(1)的 $m_i(k)$。

2　实验验证

2.1　Test-bed

Test-bed 由 4 台运行于 100Mbps Ethernet 的 PC 组成，所有 PC 均配置 Pentium 4 3.00GHz，512MB 内存。其中 Web Server 为运行于 Windows NT 上的 Apache 2（Httpd - ver. 2.0.53），最大支持并发线程数设置为 100。另外 3 台 PC 为 Linux 系统（kernel- 2.6.27），运行 SURGE（ver. 1.00a）作为模拟的 Web 负载，其中两台配置为较高优先级 IP，另外一台为较低优先级 IP。所有实验均采用 HTTP1.1 with Pipeline，SURGE 采用默认设置，单个 Client 并发请求的 TCP 连接设为 1。

2.2　线性参数拟合

如 1.3 节描述，需要通过线性回归来确定式(6)中参数 b,c 的值。本文通过最小二乘法在 0.05 的显著性水平下对 b,c 的值做线性回归，得 $b=9.7,c=73.7$，且该检验的显著性水平 R-qr=68%，即包含了所有 $T_i(s,m_i)$ 中 68% 的数据；F 检验值约等于 42，表示回归模型可以解释的变差比残差大很多。验证结果如图 4 所示，可见式(6)的预测值较好地近似于实际的测量值，因此用该回归模型来解释 $T_i(s,m_i)$ 是合理的。

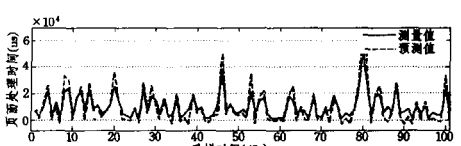


图 4　参数拟合验证

2.3　页面大小估计

实验中数据的统计结果如图 5 所示，分别表示嵌入请求个数图 5(a)、嵌入文件大小图 5(b)以及实际请求的页面图 5(c)的概率密度分布。可以看出，绝大多数嵌入对象不大于 10kbyte，并且页面大小的分布也具有重尾特性。实验中，用观测的样本均值来近似页面大小的数学期望，有 $e=bE[s_i]=62000$，$f=64c=5000$。

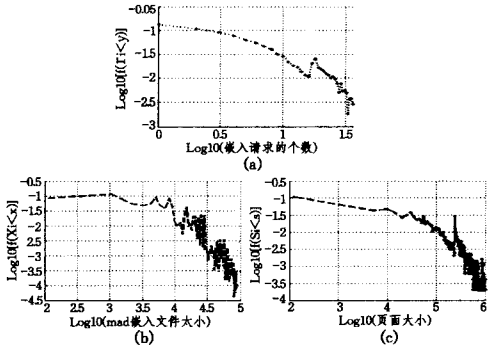
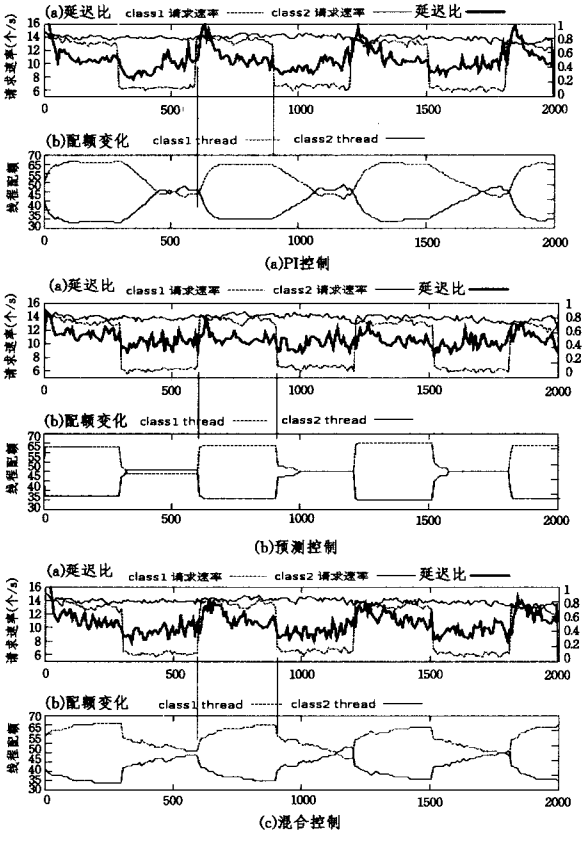


图 5　嵌入请求个数、大小和页面大小的重尾特性

2.4　结果与分析

实验共进行 2000s，采样时间为 8s，其中 Class 1 的请求率在 6～15/s 之间以 300s 为周期变化，Class 2 的请求率保持 15/s 恒定，$\delta_1/\delta_2=0.5$。图 6(a)为系统在一般 PI 控制器作用下的输出结果。图 6(b)为系统在本文提出的预测控制器作用下的输出结果，实际延迟比在 b 点(如图 2 所示)测得，预测控制器的延迟比相对 δ_1/δ_2 的方差只有 PI 控制器的40.18%，可见后者在负载变化的情况下较好地保持了系统的性能。从图中也可以反映出系统的超调明显减小。



混合控制(c)将预测控制器与 PI 控制器的输出以 1∶1 加权作为新的线程配额。从实际的效果看，相对 0.5 的方差介于(a)、(b)之间。

图 6　实验结果

再者，从图 6(a)和图 6(b)中线程配额变化与负载的关系来看，在单纯的 PI 控制下，当负载出现阶跃变化时，系统需要经历较长的调节时间才能重新达到平衡(线程配额稳定)，在上升沿，调节时间约为 100s，在下降沿，约为 150s；而在预测控制下，小于 1s 的时间即可重新达到稳定，这极大地提高了系统的动态性能。

产生这种结果的原因有两点。第一，负载变化会直接影响等待队列的长度，而等待队列变化产生的延迟要经过一段时间才能在 b(如图 2 所示)点被观测到；第二，更新过的控制量同样需要经过一段时间，才能在 b 点反映出来。而预测控制在 a(如图 2 所示)点就调整各个优先级的线程配额，为了实现准确预测，必须对负载参数进行精确的估计。图 6(c)将预测控制和反馈控制结合(混合控制)，一方面消除预测控制的偏差，一方面提高系统的实时性。尽管如此，由于上述两点原因加之反馈控制器的参数难以最优化，预测控制相对于混合控制方法还是具有相当的优越性。

结束语　本文的预测方法实现了比例延迟保证，即使是在负载剧烈变化的情况下，也得到了令人满意的结果。在 Web 负载不可预知的情况下，这种方法较经典的反馈控制具有更好的动态性能。但是由于需要参数拟合，还不能做到完全自配置和自优化，如何对参数做在线识别和在线优化，将成为下一步工作的重点。

参 考 文 献

[1] Dovrolis C, Stiliadis D, Ramanathan P. Proportional differentiated services: delay differentiation and packet scheduling[C]// Proceedings of the ACM SIGCOMM. 1999

[2] Liebeherr J, Christin N. JoBS: Joint Buffer Management and Scheduling for Differentiated Services[C]// Lecture Notes In Computer Science, Proceedings of the 9th International Workshop on Quality of Service. 2002

[3] Pan W, Mu D, Wu H, et al. Proportional Delay Differentiation Service in Web Application Servers: A Feedback Control Approach[C]// ISECS International Colloquium on Computing, Communication, Control and Management. 2008, 1: 600-604

[4] Lu C, Abdelzaher T, Stankovic J, et al. A Feedback Control Approach for Guaranteeing Relative Delays in Web Servers[C]// Real-Time Technology and Applications Symposium. Proceedings Seventh IEEE. 2001: 51-62

[5] Zhoua X, Caib Y, Chowa E. An integrated approach with feedback control for robust Web QoS design[J]. Computer Communications, 2006, 29(16): 3158-3169

[6] Barford P, Crovella M. Generating Representative Web Workloads for Network and Server Performance Evaluation[C]// Joint International Conferenceon Measurement and Modeling of Computer Systems archive. Proceedings of the 1998 ACM SIGMETRICS. 1998: 151-160

[7] Harchol-Balter M. Task Assignment with Unknown Duration [J]. J. ACM, 2002, 49(2): 260-288

[8] Arlitt M, Jin T. Workload Characterization Study of the 1 9 9 8 World Cup Website[J]. Network, IEEE, 2000: 30-37

[9] Abdelzaher T F, Bhatti N. Web Server QoS Management by Adaptive Content Delivery[C]// Quality of Service, 1999. IWQoS'99. 1999 Seventh International Workshop. 1999: 216-225