# 区分服务在多层 Web 应用中的实现

胡延苏[1] 戴冠中[1] 高 昂[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** 通过对三层服务器的构架分析,突破了常用的传递函数方法,建立了基于区分服务的 MIMO 系统状态空间模型,并在此基础上应用控制理论中的极点配置和状态反馈方法设计控制器,对不同优先级在不同 Web 层次上进行资源分配,实现其比例延迟保证。实验证明,原系统可近似为一组低阶线性方程,且在所设计的控制器作用下可达到良好的区分效果。

**关键词** 区分服务,多层 Web 应用,状态空间模型
**中图法分类号** TP393 **文献标识码** A

## Differentiated Services of Multi-tier Web Applications

HU Yan-su[1] DAI Guan-zhong[1] GAO Ang[1] PAN Wen-ping[2]
(College of Automation, Northwest Polytechnical University, Xi'an 710072, China)[1]
(College of Automation, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)[2]

**Abstract** Based on the analysis of three-tier Web applications, a state space model of a MIMO system supported differentiated service was proposed which breaks through the commonly transfer function ways. Then the controller was designed by pole placement and state feedback of control theory to adjust the resource quotas assigned to different classes in every tier and achieve the proportional delay guarantees. The experiments demonstrate that it is feasible to approximate the system to a group of low-level linear equations and the proposed controller can hold the relationship between different classes.

**Keywords** Differentiated services, Multi-tier Web applications, State space model

随着电子商务的迅猛发展,能够提供电子服务的底层构架变得越来越重要。其一般分为三层结构:前端 Web 服务器、应用服务器以及后端数据库。目前,如何为消费者提供 QoS(quality-of-service)区分服务,是 Web 开发商面临的一个重要问题。现有的做法一般为对系统服务能力进行离线计算并静态地分配资源权限。然而,Web 流量具有高度动态性及不确定性[1,2]。精确的资源配置虽能在特定的网络流量环境下保证站点的良好运行,但是当网络负载发生变化时,同样可能引起站点失常,如著名的"Slashdot 效应"。因此,Web 服务应能够对负载变化进行自调节,以保证其原有的期望性能指标。与单层 Web 服务器相比,多层 Web 构架的区分服务有一些自身特点[3]。

(1) 各层之间的相关性:各层间的影响通常同步发生。如空闲的上层线程或进程需阻塞等待下层服务的完成,因此上层的服务速率与下层的性能好坏密切相关。

(2) 各层资源均受限:由于硬件设施及避免网络超载,各层资源均会被设定上限。此时可能某一层未被使用,但客户请求仍排队等待,因为其所有线程或进程均在等待下层服务的完成。

(3) 提供 QoS 区分服务:单层的 Web 构架仅需通过对该层的资源配置即可完成不同用户等级间的区分服务。但在多层服务器构架中,还需考虑各层资源之间的相互配合。如增加某一等级的资源权限,会降低其他等级的服务性能;而对固定的等级,调整不同层次上的资源权限为相同的数值时,它们对该等级的输出影响也不会相同。

已有不少文献将控制理论应用至多层 Web 应用中,以实现性能保证。文献[4-6]提出了一种"非侵入式"的设计方案,即仅需在请求入口处放置相应的控制器,但把各层资源看作一个整体,没有涉及其单独的资源分配情况。这样虽无需对 Web 服务器进行修改,但却会大大降低系统性能并损害其实现区分服务的能力[7]。文献[8]在二层共享主机平台上,考虑到了每一层的资源分配,使得资源划分更为细腻,但始终没有逃离传递函数模型的局限。本文基于多层 Web 服务的区分服务提出了它的状态空间模型,并通过实验验证了其正确性。

## 1 多层 Web 应用构架

多数大型商业网站使用多层 Web 构架为浏览客户提供尽可优越的 Web 服务,如从初始的单层服务器、二层(前端服务器+数据库)到现在普遍使用的可靠性更高的三层结构(前端服务器+应用服务器+数据库),如图 1 所示。

到稿日期:2009-12-17 返修日期:2010-03-25
胡延苏(1985—),女,博士生,主要研究方向为网络 Web QoS 控制;戴冠中(1963—),男,教授,博士生导师,主要研究方向为计算机网络、信息安全;高 昂(1984—),男,博士生,主要研究方向为网络化控制;潘文平(1980—),男,博士,主要研究方向为计算机网络。

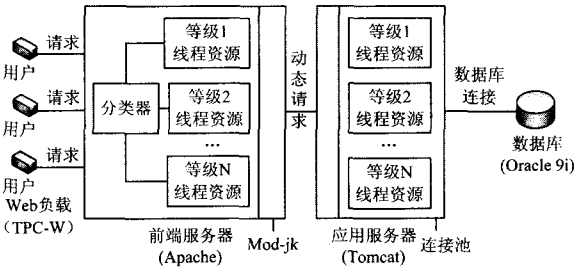


图 1　基于区分服务的多层 Web 应用构架

以三层 Web 应用构架为例说明其工作原理。前端服务器又称为 Web 服务器，作为三层结构中的表现层，功能主要包括：(1)接收网络中的用户请求，处理静态请求并把相应请求直接返回至客户端；(2)同时，把用户的动态请求发送至第二层应用服务器；(3)接收第二层服务器的相应请求，并发送至客户端。常见的 Web 服务器有 Microsoft IIS, Apache 等。

第二层应用服务器作为中间的逻辑和数据处理层，主要处理用户的动态请求。应用服务器首先接收从 Web 服务器传来的动态请求，然后在第三层数据库中查找相应的信息，最后将相应信息发送到 Web 服务器，并由 Web 服务器返回至客户端。常见的应用服务器有 IBM WebSphere, Oracle IAS, BEA WebLogic, IPlanet Application, Tomcat 等。

第三层数据库存储各种 Web 信息，可放在一台或多台主机上，提供信息收集、搜索、产品管理等服务。常见的数据库有 IBM DB2, MySQL, Oracle 等。

但原始的各种 Web 服务器以及应用服务器仅支持单队列模型，不能实现不同客户优先级的区分服务。因此，本文提出改进的基于区分服务的多层 Web 应用构架，如图 1 所示。首先在 Web 服务器处建立分类器，把客户请求根据不同的 IP 地址或请求内容分为 $N$ 个等级，并为每个等级在 Web 服务器和应用服务器处建立其相应的线程池，通过调整不同等级的线程资源分配实现不同优先级的区分服务。

## 2　控制器设计

与 SISO 模型不同，MIMO 模型不仅可表达不同 Class 资源权限之间的相互关系和影响，且对于某个 Class，可反映出多个控制输入对系统输出的不同作用。同时，MIMO 模型允许控制器同时调整位于不同层次上的不同客户等级的资源分配，因此该模型更贴切多层 Web 服务构架。在此基础上，本文设计了状态反馈控制器，以实现 Web 资源的动态分配并最终实现不同客户优先级间的 QoS 保证。

### 2.1　系统建模

设多层 Web 服务器系统中有 $M$ 层可利用资源，每层总资源为 $C_j$(如服务线程池连接池)，可规范化为 100%。$u_{i,j}$ 为第 $j$ 层服务器上 Class $i(1\leqslant i\leqslant N)$ 可获得的资源权限，即占用服务资源的百分比。由于 $\sum_{i=1}^{N}u_{i,j}=C_j(1\leqslant j\leqslant M)$，则共有 $(N-1)\times M$ 个独立变量。取系统输入变量为 $u=[u_{1,1},u_{1,2},\cdots,u_{1,M},\cdots,u_{N-1,1},u_{N-1,2},\cdots,u_{N-1,M}]^{\mathrm{T}}$。设 $t_i$ 为 Class $i$ 的端对端 QoS 参数，此处取其端对端响应时间。定义 $x_i=\frac{t_i}{\sum_{m=1}^{N}t_m}$ $(1\leqslant i\leqslant N)$，表示 Class $i$ 规范化后的 QoS 比率。易有 $\sum_{i=1}^{N}x_i=1$，故 $N-1$ 个 $x_i$ 相互独立。令 $X(k)=[x_1,x_2,\cdots,x_{N-1}]^{\mathrm{T}}$，$y_i(k)=x_i(k)$，则对基于区分服务的多层 Web 服务器系统，可利用线性 MIMO 状态空间来描述其数学模型：

$$X(k+1)=AX(k)+Bu(k) \tag{1}$$

$$Y(k)=CX(k) \tag{2}$$

式中，$A$ 为 $(N-1)\times(N-1)$ 阶的状态矩阵，$B$ 为 $(N-1)\times[(N-1)\times M]$ 阶输入矩阵，$C$ 则为 $(N-1)\times(N-1)$ 阶的输出矩阵，此处 $C$ 为单位阵 $I_{(N-1)\times(N-1)}$。

接下来通过系统辨识的方法确定矩阵 $A$、$B$ 的元素。为降低系统复杂度，假定 Class $i$ 在各层服务器上的资源权限仅对其自身有影响。又一个完成队列可表示为一阶线性系统[1]，故 Class $l$ 可表示为

$$x_l(k+1)=a_lx_l(k)+b_{l,1}u_{l,1}(k)+b_{l,2}u_{l,2}(k)+\cdots+b_{l,M}u_{l,M}(k)$$

即 $x_l(k+1)=\theta_l\phi_l(l)$

$$\theta_l=[a_l,b_{l,1},b_{l,2},\cdots,b_{l,M}]$$

$$\phi_l(k)=[x_l(k),u_{l,1}(k),u_{l,2}(k),\cdots,u_{l,M}(k)]^{\mathrm{T}}$$

采用最小二乘法进行参数估计：

$$\hat{\theta}_l(k+1)=\hat{\theta}_l(k)+\frac{e_l(k+1)\phi_l^T(k)P_l(k-1)}{\lambda+\phi_l^T(k)P_l(k-1)\phi_l(k)} \tag{3}$$

$$e_l(k+1)=y_l(k+1)-\hat{\theta}_l(k)\phi_l(k) \tag{4}$$

$$P_l^{-1}(k)=P_l^{-1}(k-1)+(1+(\lambda-1)\frac{\phi_l^T(k)P_l(k-1)\phi_l(k)}{(\phi_l^T(k)\phi_l(k))^2})\phi_l(k)\phi_l^T(k)$$

$$(0<\lambda<1) \tag{5}$$

$\hat{\theta}_l(k)$ 为参数矩阵 $\theta_l$ 的测量值，$e_l(k+1)$ 为测量误差向量，$P_l(k)$ 为协方差阵，$\lambda$ 为遗忘系数。依此类推，可得

$$\hat{\theta}(k)=[\hat{\theta}_1(k),\hat{\theta}_2(k),\cdots,\hat{\theta}_{N-1}(k)]^T=[\hat{A},\hat{B}]$$

### 2.2　状态反馈控制器

状态反馈控制器可通过调整状态反馈控制率 $u(k)=-KX(k)$ 实现对参考输入的跟踪，结构框图如图 2 所示。

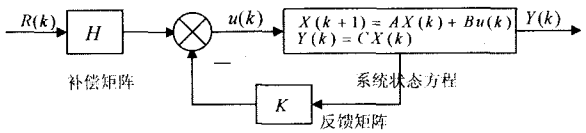


图 2　状态反馈控制框图

定义参考输入向量 $R(k)=[r_1(k),r_2(k),\cdots,r_{N-1}(k)]^{\mathrm{T}}$，故误差向量为 $E(k)=R(k)-Y(k)$。记 $E(k)=0$ 时系统状态变量的稳态值为 $X_{SS}$，相应的控制输入为 $u_{SS}$，此时

$$u(k)=-K(X(k)-X_{SS})+u_{SS} \tag{6}$$

#### 2.2.1　极点配置

假定系统性能指标为：调节时间 $k_S^*$ 个采样周期，最大超调 $M_P^*$，闭环期望主导极点为 $r=\mathrm{e}^{\pm j\theta}$。由一阶系统特性可知

$$r=\mathrm{e}^{-4/k_S^*} \tag{7}$$

$$\theta=\pi[\frac{\log(r)}{\log(M_P^*)}] \tag{8}$$

其余 $(N-3)$ 个极点应对闭环系统的影响足够小，可选取 $0.1r$，则对应的期望闭环特征传递函数多项式为

$$F(z)=(z^2-2r\cos\theta z+r^2)(z-0.1r)^{N-3} \tag{9}$$

#### 2.2.2　反馈矩阵

对于多输入系统，可根据下式确定状态反馈阵 $K$：

$$det[zI-(A+BK)]=F(z) \tag{10}$$

由于 $K$ 为$[(N-1)\times M]\times(N-1)$矩阵，有$[(N-1)\times M]\times(N-1)$个元素待定，而闭环特征多项式只有$(N-1)$个系数，故 $K$ 阵的选取并不唯一。本文将其简化为单输入系统的极点配置问题。式(1)可写为

$$X(k+1)=AX(k)+b_1u_1(k)+b_2u_2(k)+\cdots+b_{(N-1)\times M}u_{(N-1)\times M}(k)$$

若输入 $u_i(k)$可使系统

$$X(k+1)=AX(k)+b_iu_i(k) \tag{11}$$

完全可控，则取状态反馈律为

$$u_i(k)=K_ix(k) \tag{12}$$

式中，$K_i=[k_{i,1},k_{i,2},\cdots,k_{i,N-1}]$，利用式(10)即可求得。若不存在使得系统式(11)完全可控的 $u_i(k)$，构造$\hat{u}_i(k)=\alpha u_S(k)$满足条件，其中 $u_S(k)$是标量，$\alpha$ 是 $r$ 维常值向量。此时取状态反馈律为 $u_S(k)=K_Sx(k)$，$K_S$ 求解方式如上，$K_i=\alpha K_S$，则系统反馈阵 $K=[K_1,K_2,\cdots K_{N-1}]^T$。

## 3 闭环系统仿真

### 3.1 Test-bed

Test-bed 由 5 台运行 Windows 平台的 100Mbps Ethernet 的 PC 组成，其中 1 台作为 Web 服务器，服务器软件为 Apache2.0.53，并发 100 个服务线程；1 台运行 Tomcat 6.0.13 作为第二层的应用服务器，其连接池允许的最大连接数设置为 50 个；1 台运行 MySQL 作为后端数据库，其连接方式如图 1 所示：Apache 通过 mod_jk 模块遵从 AJP13 协议把客户的动态请求依次发送至 Tomcat，在标准的 JSP/Servlet 容器中，可以将动态请求映射到 JSP 页面或者 Servlet 的 Java 类，动态请求利用数据库连接池的连接，完成客户与数据库的交互。这样，系统可调度资源为 Web 服务器上的线程个数和应用服务器上连接池中的连接个数。

选用 TPC-W 基准测试该多层服务器系统。TPC-W 是 TPC 委员会(Transaction Processing Council)2000 年发布的一个电子商务应用基准，但并不提供源码。因此，我们修改由 Wisconsin-Madison 大学 PHARM 实验室发布的 TPC-W JAVA 应用程序(ver1.5)[10]，以与该实验平台兼容，并在 2 台 PC 上运行该程序作为负载发生器。客户请求基于会话(session)给出，负载强度由并发的模拟浏览器个数决定。其中，每个会话中请求间隔即客户端思考时间服从均值为 0.7s 的指数分布。数据库由 10 个表格组成，包含 10000 个条款及 288000 个客户。

### 3.2 系统辨识

在上述实验平台上，根据本文 2.1 节算法进行系统辨识。Apache 和 Tomcat 根据伪随机序列当前的值，调整下一周期不同优先级客户端资源配额，即线程个数和连接个数，故 $M=2$。客户端由 2 台 PC 模拟 2 类客户，即 $N=2$。取 $t_1:t_2=1:3$，$X=1/4$。输入的伪随机序列按式(11)产生：

$$\varepsilon(k)=\varepsilon(k-p)+\varepsilon(k-q)(\mathrm{mod}\ 4) \tag{13}$$

取 $p=7$，$q=13$，$\varepsilon(k)$与 $u_{1,j}(k+1)$具体对应关系如表 1 所列。

表 1 $\varepsilon(k)$与 $u_{1,j}(k+1)$对应关系

| $\varepsilon(k)$ | $u_{1,j}(k+1)$ |
|---|---|
| 0 | 0.25 |
| 1 | 0.40 |
| 2 | 0.60 |
| 3 | 0.75 |

取采样周期 $T=8$s，实验共进行 1200s，获得 120 组有效数据，对应系统参数的最小二乘辨识为

$$x_1(k+1)=a_1x_1(k)+b_{1,1}u_{1,1}(k)+b_{1,2}u_{1,2}(k)$$

$$a_1=-0.1028,b_{1,1}=0.4371,b_{1,2}=0.5619$$

验证结果如图 3 所示。可见辨识值较好地近似于实际的测量值，所以用该状态空间模型描述区分模型是合适的。取调节时间 $k_S^*=20$，则闭环期望主导极点 $r=e^{-4/k_S^*}=0.8187$。由式(10)、式(11)可得 $K=[K_1,K_2]^T=[2.11,1.64]^T$。

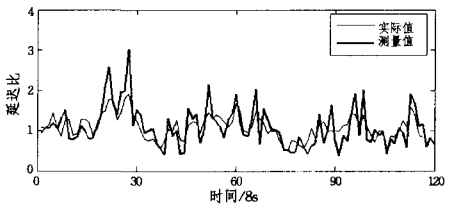


图 3 系统辨识值与实际测量值比较

### 3.3 实验结果分析

为了验证该实验平台下多层 Web 服务器区分服务模型的正确性以及对应的状态反馈控制器的有效性，本文设计了两组实验。第一组实验结果如图 4 所示，高低优先级客户均运行 120 个浏览器请求，前 50 个采样周期控制器关闭，此时系统无区分服务可言；400s 之后，在控制器的作用下，各后端服务器分配给不同优先级的资源配额随之调整，对应的实际延迟之比可较好地稳定在 1/3 附近。其中，图 4 上图为高低优先级每个采样周期内的平均延迟以及相应的延迟比；图 4 下图为其请求到达率。可见，此时高低优先级的请求到达率基本一致。第二组实验用以测试负载突变时的系统延迟保证效果。以图 5 为例，控制器关闭直至第 30 个采样周期，高低优先级客户同样启动 120 个浏览器，在第 80 个采样周期时，低优先级客户再次启动 60 个浏览器请求，直至实验结束。图 6 与图 5 类似，只是第 80 个采样周期时，高优先级客户再次启动 60 个浏览器请求。由图可以看出，负载发生变化时会引起延迟比的波动，但在控制器的作用下仍可迅速收敛至目标值。

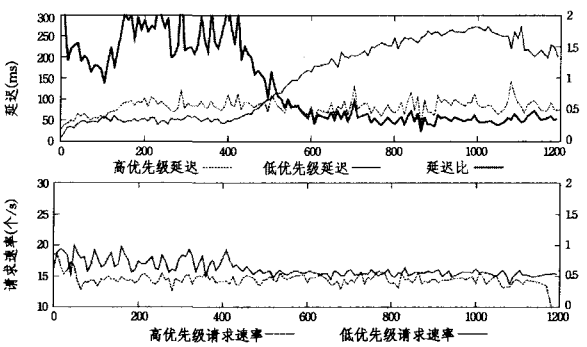


图 4 高低优先级请求到达率恒定时的比例延迟保证

从上述实验结果可以看出，在多层 Web 服务器各层资源有限的情况下，即使负载变化剧烈(如低优先级请求增多)，本文给出的区分模型以及状态反馈控制器可较好地实现不同客户优先级的区分服务，维持其延迟比恒定，且调节时间满足指定的性能指标。对比两组实验结果可明显看出，负载发生变化时延迟比的输出波动相对较大，分析其可能的原因有二：(1)系统辨识结果不够精确，可能在模型简化的过程中造成了关键信息的丢失，没有完全反映出系统特性，且离线辨识往往

(下转第 110 页)

channel simulator for wireless communications: case study with DSRC and UWB channels [J]. IEEE Transactions on Instrumentation and Measurement, 2009, 58(8): 2755-2766

[6] Garfinkel S L, Juels A, Pappu R. RFID privacy: an overview of problems and proposed solutions [J]. Security & Privacy Magazine, 2005, 3(3): 34-43

[7] Juels A, Rivest R L, et al. The blocker tag: selective blocking of RFID tags for consumer privacy [C]// 10th Conf. on Computer and communications security. Washington: ACM Press, 2003: 103-111

[8] Karjoth G, Moskowitz P A. Disabling RFID tags with visible confirmation: clipped tags are silenced [C]// ACM Workshop on Privacy in Electronic Society. Alexandria: ACM Press, 2005: 27-30

[9] 邓森磊，马建峰，周利华. RFID匿名认证协议的设计 [J]. 通信学报，2009，30(7)：20-26

[10] IEEE Std 802.11i™ Amendment 6: Medium Access Control (MAC) Security Enhancements [S]

[11] Goldreich O. Foundations of Cryptography [M]. Cambridge: Cambridge University Press, 2001

[12] Bolotnyy L, Robins G. Physically unclonable function-based security and privacy in RFID systems [C]// 5th Conf. on Pervasive Computing and Communications. Washington: IEEE Press, 2007: 211-220

[13] Shannon C. Communication Theory of Secrecy Systems [J]. Bell System Technical Journal, 1949, 28(4): 656-715

(上接第 91 页)

会造成控制应用的实时性不佳；(2)大的延迟比波动会带来较大比率的线程或连接数调整，增加系统开销，从而对请求延迟造成影响。

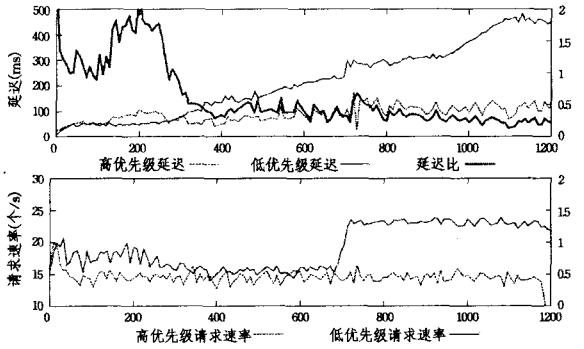


图 5 低优先级请求变化时的比例延迟保证

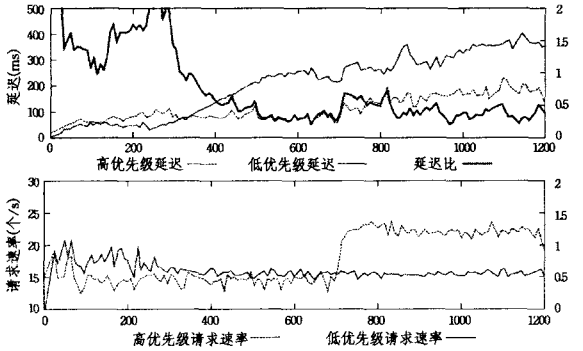


图 6 高优先级请求变化时的比例延迟保证

**结束语** 本文研究了基于多层 Web 应用的区分服务，提出并实现了一种支持比例延迟保证的多层 Web 应用的多输入多输出模型。通过系统辨识建立系统状态空间模型，并在此基础上完成了状态反馈控制器的设计。最后，在 Windows 平台下，修改了 Apache 源码以及 Tomcat 的 Servlet，并分别采用 TPC-W 基准进行了实验，验证了系统模型的有效性，实现了比例延迟保证。但同时实验结果也暴露了很多的不足。下一步工作应该更加深入研究多层 Web 应用，并寻找更贴近该系统的数学模型及其控制器的设计。

## 参 考 文 献

[1] Mogul J C. Operating systems support for busy internet servers [C]// Fifth Workshop on Hot Topics in Operating Systems (HotOS-V). Orcas Island, WA, 1995

[2] Barford P, Crovella M. Generating representative web workloads for network and server performance evaluation[C]// Measurement and Modeling of Computer systems. 1998: 151-160

[3] Diao Y, Hellerstein J L, Parekh S, et al. Modeling differentiated dervices of multi-tier Web applications[C]// 14th Annual Meeting of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS). 2006

[4] Kamra A, Misra V, Nahum E M. Yaksha: A self-tuning controller for managing the performance of 3-tiered web sites[C]// Proceedings of the Twelfth International Workshop on Quality of Service (IWQoS 2004). Montreal, Canada, 2004

[5] Liu Xue, Heo Jin, Sha Lui, et al. Adaptive Control of Multi-tiered Web Application Using Queueing Predictor[C]// Proceedings of the 10th IEEE/IFIP Network Operations and Management Symposium (NOMS 2006). Vancouver, Canada, 2006

[6] Karlsson M, Zhu Xiaoyun, Karamanolis C. An adaptive optimal controller for non-intrusive performance differentiation in computing services[C]// International Conference on Control and Automation (ICCA 05). 2005

[7] Diao Yixin, Hellerstein J L, Parekh S, et al. Controlling Quality of Service in Multi-tier Web Applications[C]// Proceedings of the 26th IEEE International Conference on Distributed Computing Systems(ICDCS 06). 2006

[8] Liu Xue, Zhu Xiaoyun, Padala P, et al. Optimal Multivariate Control for Differentiated Services on a Shared Hosting Platform [C]// Proceedings of the 46th IEEE Conference on Decision and Control (CDC'07). New Orleans, LA, 2007

[9] Hellerstein J L, Diao Yixin, Parekh S. Feedback control of computing systems[M]. Wiley-IEEE Press

[10] Bezenek T, Cain H, Dickson R, et al. Characterizing a Java Implementationn of TPC-W[C]// 3rd Workshop On Computer Architecture Evaluation Using Commercial Workloads (CAECW). Toulouse, France, January 2000