维普资讯 http://www.cqvip.com

基于反馈控制的开放式实时系统自适应调度算法设计与实现*)

牛 云 戴冠中 慕德俊 梁亚琳

(西北工业大学自动化学院 西安 710072)

摘 要 对于负载不可预测且资源受限的开放式实时系统,传统"开环"调度算法不能根据系统负载情况调整调度策略,影响系统实时性能或造成资源浪费。采用双闭环反馈控制方法,改进目前开放式实时系统常用的时限驱动总带宽利用率服务器。本方法根据负载情况,动态地为不同实时性能需求的任务分配资源,保证硬实时任务满足时限要求并且提高资源利用率。实验表明,系统负载存在突发的情况下,算法既很好地控制了任务的时限错过率,又得到了较高的资源利用率。

关键词 开放式实时系统,总带宽利用率服务器,反馈控制调度,硬实时性能保证,资源利用率

Design and Implementation of Self-adaptation Scheduling Algorithm for Open Real-time System

NIU Yun DAI Guan-zhong MU De-jun LIANG Ya-lin

(College of Automation, Northwesten Polytechnical University, Xi'an 710072, China)

Abstract The traditional "open loop" scheduling algorithms perform poorly in open real time systems whose workloads vary unpredictable dynamically because the "open loop" refers to the fact that once schedules are created they are not adjusted based on the system workloads. A dual close-loop feedback scheduling algorithm is proposed to improve total bandwidth server in EDF scheduling which is widely used in open real time systems. The algorithm adjusts the resource assignation for different kinds of tasks based on feedback workloads and guarantees the deadline miss ratio of hard real time tasks with adequate resource utilization. Experiments based on VxWorks/Goahead embedded Web server system demonstrate the effectiveness of the proposed scheduling algorithm when there are large bursts of total load in the system.

Keywords Open real time systems, Total bandwidth server, Feedback control scheduling, Deadline miss ratio guarantee, Resource utilization

1 引言

随着嵌入式系统的不断深入与发展以及 Internet 与嵌入式系统的相互结合,多类型的硬实时、软实时与非实时任务共存同一系统的情况越来越广泛,这种系统称为开放式实时系统[1]。一个典型的例子就是嵌入式 Web 服务器(EWS)。Bill McCombie 在文献[2]中指出:"EWS 在硬实时、软实时系统中都可以得到应用。在软实时系统中,如自动售货机,实时性要求不严格,进程调度不是关键问题。但在硬实时系统中,如 EWS 提供工业控制现场运行的状态数据。工业控制任务具有硬实时性,而 EWS 进程没有硬实时性要求,需要与现场控制任务并行执行同时不能影响其实时性能。对开放实时系统任务进行合理调度成为关键。"

Z. Deng 等提出的开放式实时系统调度的目标[1]:保证硬实时任务满足其截止期的基础上减小弱实时任务的平均响应时间,同时系统中非相关的实时或非实时应用可独立进行实时性的验证测试。

为解决上述问题,近年来的研究大多采用基于带宽保留服务器[3]的实时调度算法。文献[4,5]提出了一种基于 CUS (Constant Utilization Server)和 TBS(Total Bandwidth Server)服务器的调度方案。它允许硬实时、软实时和非实时任务共存于一个系统,提供了应用程序之间的资源隔离支持。其结构如图 1 所示。

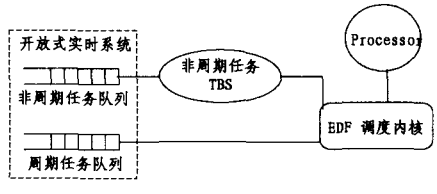

图 1 开放式实时系统任务调度模型

这些算法是基于资源预留的,需要预知任务运行参数,如最坏执行时间、最小执行周期等。然而,上述运行参数在某些开放式实时应用中很难事先确定,如 EWS 的访问请求流量具有自相似特性[6],即在秒级的时间尺度上存在突发(Burst)现象,且静态请求与动态请求的处理时间相差较大,所以 EWS 任务的执行周期和执行时间都不可能事先确定。实验说明(详见第 4 节),对于负载情况不可预测的开放式实时系统,传统的 TBS+EDF 的调度算法不能根据 CPU 负载情况动态地调节服务器调度参数,当非周期软实时任务突发增加

*)基金项目:国防基础科研项目(项目编号:C2720061361)。牛 云 博士研究生,主要研究方向为嵌入式网络处理器、实时系统调度算法;戴冠中 博士生导师,主要研究领域为自动控制、信息安全;慕德俊 博士生导师;梁亚琳 硕士研究生。

维普资讯 http://www.cqvip.com

时，会影响硬实时任务的正常执行。

反馈控制理论与实时调度的结合是解决上述不可预测环境下的调度问题的有效方法之一，近年来成为国内外学者研究的热点。文献[7]综述了反馈理论在提高软件尤其是软实时系统性能方面的结构与应用。文献[8]通过对实时任务的服务分级将反馈控制技术应用于改进传统的 EDF 调度。文献[9]通过系统反馈信息调整可延期服务器的调度参数，从而获得容错的实时调度。但上述算法都是针对软实时任务调度的。对于开放式实时系统中硬实时、软实时任务并行执行的情况下，系统调度 QoS 的控制没有提出有效的解决方法。

本文利用反馈控制控制理论设计控制器，根据 CPU 实际负载情况动态调整 EDF＋TBS 的 Deadline，budget 等调度参数，使得硬实时任务的截止期错过率(miss ratio)保持为 0，并尽可能减少非周期软实时任务的反应时间。算法记为 FC_EDFTBS。

2 系统模型

为便于定量分析系统的可调度性等实时性能指标，对系统中的任务进行建模[3]。

设有任务集

$$s=Tp_1,Tp_2,\cdots,Tp_n;Tap_1,Tap_2,\cdots,Tap_m \tag{1}$$

其中，$Tp_i(1\leqslant i\leqslant n)$为周期任务，每个周期任务 Tp_i 可用(2)式所示的四元组表示：

$$Tp_i=\langle Cp_i,Pp_i,\Phi p_i,Dp_i\rangle \tag{2}$$

其中，Cp_i 为任务的最大计算时间，Pp_i 为任务周期，Φp_i 为任务释放时刻，Dp_i 为任务时限；对于大多数的周期任务 $Dp_i=Pp_i$。

$Tap_k(1\leqslant k\leqslant m)$为非周期任务，每个非周期任务可由如下的三元组表示：

$$Tap_k=\langle Cap_k,\Phi ap_k,Dap_k\rangle \tag{3}$$

其中，Cap_k 为任务作业流中的最大计算时间，Φap_k 为任务释放时刻，Dap_k 为任务时限。

我们建立 TBS 调度管理非周期任务，TBS[3] 可以看成是系统调度机制创建的特殊任务，它为调度对象提供服务。TBS 可用如下模型来描述：

$$T_s=\langle e_s,D_s\rangle \tag{4}$$

其中，e_s 为服务器的执行预算，D_s 为服务器在 EDF 下的时限。

传统的预算补充规则如下：

R1：初始状态，$e=0$，$D_s=0$。

R2：当一个执行时间为 e 的非周期作业在时刻 t 到达一个空的服务器队列时，令 $D_{s(k)}=\max(D_{s(k-1)},t)+e/U_s$，且 $e_s=e$。队列非空时，直接将作业加入队列。

R3：当服务器完成当前作业时，

(a) 若服务器有储备，即令服务器的时限为 $D_{s(k)}=D_{s(k-1)}+e/U_s$，且 $e_s=e$。

(b) 若服务器空闲，则什么也不做。

其中，U_s 称为服务器的强度(size)。传统的 U_s 由(5)式确定。

$$U_s=\max(Cap_k/Dap_k) \tag{5}$$

上述服务器的强度计算规则需要确定非周期任务的 Cap_k 和 Dap_k，这种方式为非周期任务预留了过多资源且不能动态调整。如前所述，这在某些开放式实时系统如 EWS 中是不适用的。而由 TBS 的预算补充规则可见，U_s 可以直接影响服务器在 EDF 下的优先级，U_s 越大，则分给非周期任务的 CPU 带宽越多。相对地，留给硬实时任务的 CPU 带宽就越少。如果可以根据 CPU 负载情况动态的调节 U_s，则可以调整各类任务分配的资源，从而根据负载情况区分服务硬实时、软实时任务。FC_EDFTBS 就是以此为基础对 TBS 进行改进。

3 开放式实时系统 FC_EDFTBS 结构

设计算法的目标是保证开放实时系统中的硬实时任务满足其时限要求，在此基础上尽量提高调度算法对非周期任务的 QoS。硬实时任务的调度性能用截止期错过率衡量，非周期软实时任务的调度性能用平均响应时间衡量。

3.1 可调度性测试

文献[3]证明，独立、可抢占的周期任务和非周期任务系统中，如果周期性任务的总利用率与非周期任务的瞬间利用率之和小于或等于 1，那么按照 EDF 算法，这个系统是可调度的。在由非周期任务服务器的系统中即(6)式成立：

$$\sum_{k=1}^{n}\frac{Cp_i}{\min(Dp_i,Pp_i)}+U_s\leqslant 1 \tag{6}$$

其中：$\sum_{k=1}^{n}\frac{Cp_i}{\min(Dp_i,Pp_i)}$为硬实时周期任务利用率，$U_s$ 为服务器强度。为保证硬实时任务的可调度性，可以将(7)式作为动态调节 U_s 的上限。

$$U_s\leqslant 1-\sum_{k=1}^{n}\frac{Cp_i}{\min(Dp_i,Pp_i)} \tag{7}$$

3.2 FC_EDFTBS 调度机制

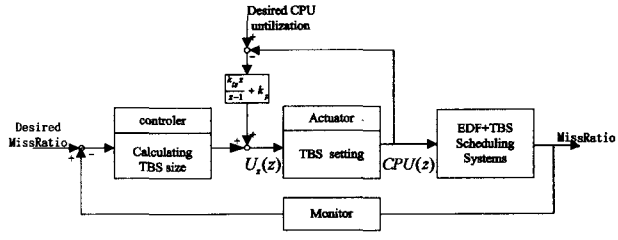

图 2 反馈调度算法结构

调度系统的结构如图 2 所示，采用双环反馈结构。输出量是系统中硬实时任务的时限错过率；输入量是期望的时限错过率；控制量是 TBS 强度 $U_s(z)$；操作量是 CPU 利用率 $CPU(z)$。主反馈保证硬实时任务的时限错过率稳定在设定值，局部反馈挪用硬实时任务的空闲利用率，尽可能提高非周期软实时任务的 QoS。

控制器算法由差分方程(8)－(11)给出：

$$U_s(k)=U_s(k-1)+K_{tp}\times E_{MR(k)}+K_p\times(E_{CPU(k)}-E_{CPU(k-1)})+K_{ts}\times E_{CPU(k)} \tag{8}$$

其中

$$E_{MR(k)}=MR_s-MR(k) \tag{9}$$

$$E_{CPU(k)}=CPU_t-CPU(k) \tag{10}$$

由式(10)得

$$E_{CPU(k)}-E_{CPU(k-1)}=CPU(k-1)-CPU(k) \tag{11}$$

调度器在 k 时刻采样系统的实际时限错过率($MR(k)$)和 CPU 利用率($CPU(k)$)。当 $MR(k)$ 大于错过率的设定值(MR_s)时，说明此时系统硬实时性能不佳，则 $K_{tp}[MR_s-MR(k)]$项为负值，使 U_s 降低，减少分给 TBS 的 CPU 带宽；当 $CPU(k)$小于方程(6)确定的 CPU 利用率阈值(CPU_t)时，说明此时系统有能力处理更多的非周期任务，则 $K_{ts}[CPU_t-CPU(k)]$项为正值，使 U_s 增大，增加分给 TBS 的 CPU 带宽。

维普资讯 http://www.cqvip.com

执行器中 TBS 预算、时限设定规则与传统 TBS 的规则基本相同，但补充的预算不是不可预测的最坏执行时间，而是根据非周期任务的不同类型（比如 EWS 中的动态、静态请求）赋予固定的平均执行时间作为预算。而服务器的能力由控制器的输出 U_s 决定。

4 实验结果

一般情况下，硬实时任务的资源消耗基本稳定[1]，实验的主要目的是检验非周期任务突发情况下开放式实时系统在本算法下保证硬实时应用性能的能力。实验基于本实验室开发的 EWS 系统。该系统进行现场实时控制的同时，远程工作站可以通过 Web 请求获得现场数据。系统硬件平台基于 s3c2510ARM 处理器，软件平台基于实时操作系统 VxWorks[10] 和 Goahead 嵌入式 Web 服务器[11]。利用 VxWorks 的任务管理 API，Hook 函数以及软件看门狗，定时器等开发中间件实现 FC_EDFTBS，并接管 Goahead 建立的 Web 请求处理任务，由 TBS 调度管理。使本算法对上层应用程序透明。系统结构如图 3 所示。

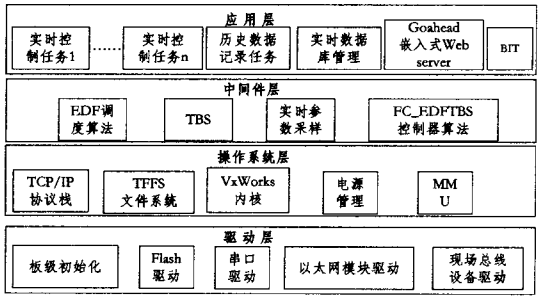

图 3 FC_EDFTBS 实验平台软件结构

在 EWS 系统上建立一组独立的硬实时周期任务，用来进行工业现场实时监控或维护实时数据库。当实时作业错过其时限时，立即删除该作业。Goahead 建立非周期任务响应 Web 请求。系统中硬实时任务的 CPU 占用率约为 85%。FC_EDFTBS 调度上述混合任务集，调度的目标是保证硬实时任务的错过率为“0”的基础上，最小化 Web 请求的响应时间。FC_EDFTBS 控制器参数由工程中常用的“先比例后积分”的实验试凑法确定。取 $K_{tp}=0.12$，$K_{ti}=0.025$，$K_p=0.2$，采样周期为 10ms，方程(10)的 CPU 利用率阈值(CPU_t)为 0.95。当方程(8)的计算结果为负值时，说明此时 CPU 超载，将 U_s 置为“0”，禁止新的 Web 请求进入系统。待 U_s 大于 0 时，由 TBS 恢复接收。同时，U_s 的值不超过(7)式的可调度上限。

为研究 FC_EDFTBS 在系统负载剧烈变化下的调度性能，在如图 4 所示的 Web 请求突发情形下，比较 FC_EDFTBS、不同 TBS 规模的传统 TBS+EDF 算法的调度效果。

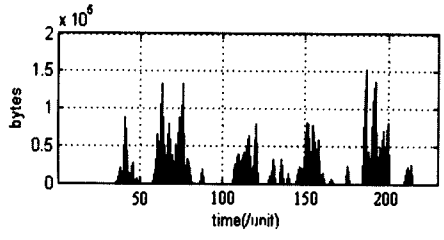

图 4 Web 请求流量

图 5 显示了 TBS 的规模为 0.01 时的调度情况。由于预留给非周期任务的 CPU 带宽很小，虽然可以保证硬实时任务性能，却导致 CPU 利用率过低，使 Web 请求的响应延迟大幅增加（如图 8 所示）。相反地，图 6 显示了 TBS 的规模为 0.15 时的调度情况，虽然可以减小 Web 请求的响应延迟（如图 8 所示），但是由于不能动态调整服务器规模，在系统负载突发变化时，会造成硬实时任务频繁错过时限，任务实例因此被调度器删除（CPU 利用率抖动由此造成），使系统调度失败。图 7 显示了 FC_EDFTBS 算法的调度效果，算法根据负载变化及时限错过率调整 TBS 的规模，使负载突发增加时，保证 CPU 的利用率始终保持在设定值 0.95 左右，获得了满足系统要求的实时任务时限错过率和较小的 Web 请求平均响应时间（如图 8 所示）。

表 1 汇总了上述实验的 CPU 利用率（UTIL）、硬实时任务时限错过率（MRA）、任务相对吞吐量（若干个采样周期内，按时完成的任务数与进入系统任务数的比值，记为 THOR）和 Web 平均请求延迟（GRD）四个性能指标参数。EDFTBS0.15 受硬实时任务的完成数影响，EDFTBS0.01 受弱实时任务完成数的影响，THOR 值比 FC_EDFTBS 低。

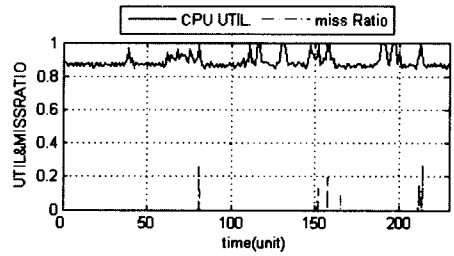

图 5 EDFTBS0.01 下 CPU 利用率和时限错过率

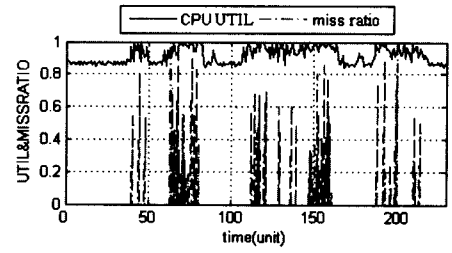

图 6 EDFTBS0.15 下 CPU 利用率和时限错过率

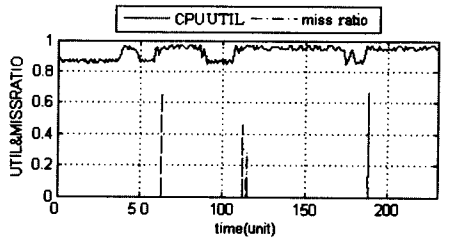

图 7 FC_EDFTBS 下 CPU 利用率和时限错过率

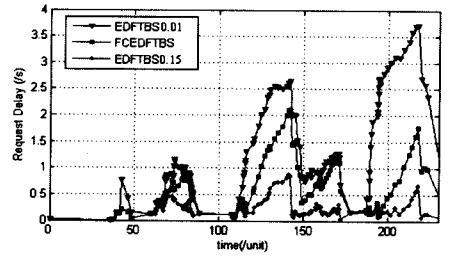

图 8 Web 响应延迟比较

（下转第 118 页）

维普资讯 http://www.cqvip.com

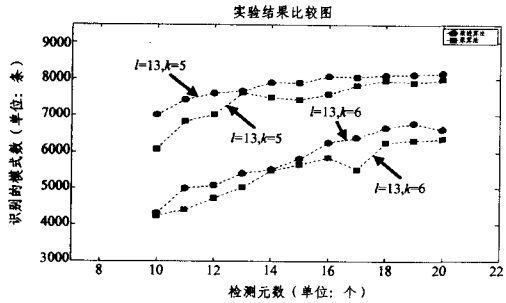

图 2 $k\in[0,l/2]$时原算法和改进算法覆盖空间的对比

实验结果如图 1、图 2 所示。实验结果表明，在模式长度 l 固定的情况下，无论匹配阈值 k 如何变化，改进算法能够识别的模式数量均高于原算法，即改进算法的检测能力优于原算法，特别是当 $k\in[0,l/2]$时，效果更加明显。图中算法改进前后识别的模式数量相同的点，说明存在原算法产生的检测元互不匹配的情况。

结束语 基于免疫原理的网络入侵检测技术中，产生有效检测元的否定选择算法采用 k 连续位匹配规则，使得有效检测元集 R 中的检测元存在相互匹配的可能，导致系统在资源有限的情况下对网络入侵的检测能力无法达到最大化。本文对否定选择算法进行了改进，提高了检测元集的整体覆盖能力和系统对网络入侵的检测能力，具有较好的实用价值。

参 考 文 献

[1] Forrest S, Parelson A, Allen L, et al. Self-nonself Discrimination in A Computer[C]// Proceeding of the 1994 IEEE Symposium on Research in Security and Privacy. Los Alamos, CA, IEEE Computer Society Press, 1994

[2] Forrest S, Hofmeyr S A, Somayaji A. Computer Immunology [J]. Communication of the ACM, 1997, 40(10): 88-96

[3] Haeseleer, Forrest S, Helman P. An Immunological Approach to Change Detection: Algorithms, Analysis and Implication // Proceedings of the 1996. IEEE Symposium on Computer Security and Privacy. IEEE Computer Society Press, Los Alanmitos, CA, 1996: 110-119

[4] Hofmeyr S A. An Immulogical Model of Distributed Detection and Its Application to Computer Security [D]. Ph. D Thesis. University of New Mexico

[5] Kim J, Bentley P. The Human Immune System and Network Intrusion Detection [C]// EUFIT99

[6] Kim J, Bentley P. Evaluation of Negative Selection in An Artificial Immune System for Network Intrusion Detection. GECCO, 2001

[7] Dasgupta D, Attoh-Okine N. Immunity-based Systems: A Survey// Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics. Orlando, Florida, Oct. 1997: 363-374

（上接第 61 页）

表 1 不同调度算法的性能比较

	UTIL	MRA	THOR	GRD
FC_EDFTBS	0.937	0.003	0.996	637ms
EDFTBS0.15	0.919	0.180	0.816	277ms
EDFTBS0.01	0.887	0.001	0.898	1188ms

5 进一步的研究

文献[12,13]认为，将反馈控制理论应用到计算系统 QoS 保障，关键需要解决 QoS 需求在反馈控制系统中的映射、控制器/执行器的设计、控制环中目标系统（被控对象）的建模等方面的问题。

文献[8,9]和我们的实验显示 CPU 的利用率对任务的时限错过率有显著影响。我们利用改进的 TBS 作为执行器，其规模 U_s 作为控制量，控制 CPU 利用率，从而保证系统的时限错过率满足要求，完成实时 QoS 在控制系统的映射。控制目标建模是确定控制器参数的关键，也是在计算系统 QoS 保障中应用控制方法的难点[12]。目前国内外对特定计算系统精确建模并不多见。文献[13,14]利用离线系统辨识方法对通用 Web 服务器 QoS 建模。文献[15]对排队论进行改进，将请求的时限引入排队模型，提出实时排队论，并利用它对 EDF 调度算法进行建模。在今后的研究中，我们将以上述文献为基础，针对开放式实时系统的特点，建立目标系统精确模型，以进一步提高反馈调度器的性能。

结束语 本文针对负载存在突发变化的开放式硬实时系统（例如适用于工业控制的 EWS），应用反馈控制方法设计负载自适应的 FC_EDFTBS。给出了调度算法的反馈控制结构以及控制器的差分方程。通过反馈控制将硬实时系统的关键性能指标：时限错过率与调度器直接关联，从而明显减弱系统负载动态的变化对实时性能的影响。最后，我们基于 VxWorks+Goahead 嵌入式 Web 服务器系统，开发中间件实现 FC_EDFTBS 算法，通过实验对比传统调度算法的性能。实验证明 FC_EDFTBS 在负载突变时，可以获得满意的时限错过率和较高的资源利用率。

参 考 文 献

[1] Deng Z, Liu J W S. Scheduling real-time application in open environment[J]// Proceedings of the 18th IEEE Real-Time Systems Symposium. Los Alamitos, CA: IEEE Computer Society Press, 1997: 308-319

[2] McCombie B. Embedded Web servers now and in the future[J]. Real-Time Magazine, 1998(1): 82-83

[3] Jane W, Liu S. Real-Time Systems[M]. Beijing: Higher Education Press arrangement with the original publisher, Pearson Education, Inc., 2002: 195-218

[4] Abeni L, Buttazzo G. Intergrating multimedia applications in hard real-time systems // Proc. 19th IEEE Real-time SystemsSymposium. Madrid, Spain

[5] Cervin A, Eker J. Control-scheduling Codesign of Real-time Systems: The Control Server Approach// Proc. Journal of embedded computing, 2004

[6] Crovella M E, Bestavros A. Self-similarity in World Wide Web T-raffic: Evidence and Possible Causes[J]. IEEE/ACM Transaction on Networking, 1997, 5(6): 835-846

[7] Abdelzaher T F, Stankovic J A, Lu Chenyang. Feedback Performance Control in Software Services[J]. IEEE Control Systems Magazine, June 2003

[8] Lu C, Stankvoic J A. Design and Evaluation of a Feedback Control EDF Scheduling Algorithm // IEEE Real-Time Systems Symposium. Phoenix, AZ, Dec. 1999

[9] Lin Suzhen, Manimaran G. A FeedBack-based Adaptive Algorithm for Combined Scheduling with Fault-Tolerance in Real-Time Systems// Proc. Conference on High Performance Computing (HiPC). Bangalore, India, Dec. 2004: 101-110

[10] VxWorks-Programmer's Guide 5.5, Wind River Systems[EB], INC, 2002

[11] Goahead Software Foundation[EB]. http://www.goahead.com

[12] Hellerstein J L, Diao Yixin, Parekh S, et al. Feedback Control of Computing Systems [M]. Wiley-IEEE, John Wiley & Sons, 2004: 31-56, 293-334

[13] Zhang Ronghua, Lu Chenyang, Abdelzaher T F. ControlWare: A Middleware Architecture for Feedback Control of Software Performance // The 22nd International Conference on Distributed Computing Systems(ICDCS'02). 2002

[14] Lu Chenyang, Abdelzaher T F, Stankovic J A. A Feedback Control Approach for Guaranteeing Relative Delays in Web Servers // IEEE Real-time Technology and Applications Symposium. June 2001

[15] Lehoczky J P. Real-Time Queueing Theory[J]// Proceedings of the 17th IEEE Real-Time Systems Symposium. 1996: 186-195