基于QoS的磁盘调度策略*)

厉 励 张宏坡 李 海 周 兵
(郑州大学 郑州 450001)

摘 要 随着视频点播、视频会议、视频监控、数字图书馆等流媒体应用的普及,流媒体服务器存储资源管理成为制约服务质量的瓶颈之一。根据多媒体服务器的性能要求,提出了一种支持QoS的磁盘调度策略。它由三个主要部分组成:探测模块、负载监测模块和自适应管理模块。探测模块,负责判断当前的资源情况能否满足服务请求;自适应模块,根据负载监测模块检测到的负载变化情况,动态调整服务周期在实时请求和尽力服务请求之间的分配。实验表明此磁盘调度策略能在保证实时请求无抖动执行的同时,明显减少了非实时请求的响应时间。

关键词 多媒体服务器,磁盘调度,实时请求,尽力服务请求

A Disk Scheduling Scheme with QoS Guaranteeing

LI Li ZHANG Hong-Po LI Hai ZHOU Bing
(Zhengzhou University, Zhengzhou 450001)

Abstract With booming of the application of streaming media such as VoD, video conference, video surveillance system, digital library etc, the management of storage resource becomes one of the bottlenecks that restrict the service quality of streaming media server. In this paper a novel disk scheduling scheme of QoS guaranteeing is proposed for media streaming servers in this paper. The scheme consists of three parts: a request detecting module which determines whether the current resource meets the demand; a workload detecting module which monitors the load change in each application; a disk period manager which uses the workload statistical results to dynamically adjust the period allocation of each application. The scheme gives priority to the Best-Effort disk requests while meeting real time request deadlines. Experiments show this scheme can reduce the mean response time of the Best-Effort requests while guaranteeing the real time requests with little jitters.

Keywords Media streaming server, Disk scheduling scheme, Real time request, Best-Effort request

1 引言

随着网络技术的飞速发展,视频会议、数字视频、数字图书馆等实时应用大量出现,多媒体文件服务器要存储大量的异类的数据(文字、图像、音视频等),它不仅要能够处理对文本、图像这样离散数据的请求,同时还要能处理对音频、视频这样连续数据的请求,由于这类多媒体应用必须在特定的时间段内传送特定数量的媒体流信息,它要求端系统和网络系统都必须支持QoS[1,2]。

多媒体应用需要传送大量的连续数据,而连续数据是与时间相关的,它们要求采用实时磁盘调度算法处理;离散数据虽没有实时要求,但是必须尽可能地让它们有合理的响应时间。因此,针对多媒体服务器中存储媒体的不同类型的要求,就必须采用合理的磁盘调度策略[3],一方面对实时请求要求保证其时间约束参数,另一方面要尽最大努力地调度非实时请求,让它们有合理的响应时间。

传统的磁盘调度算法与磁盘的物理特性紧密相关,为了优化性能,没有考虑到读写数据的实时性要求。常用操作系统采用的尽力服务策略无法保证多媒体应用的QoS要求,而实时操作系统由于对任务执行时限的严格限制,不能很好地适应应用的动态变化和取得较高的资源利用率。要支持应用程序的QoS要求[4],确定相关磁盘读写的最小延迟是非常重要的,这要求在磁盘调度中考虑实时性的要求。因此本文提出了一种支持QoS的磁盘的调度策略。

2 磁盘调度策略

多媒体服务器既要完成传统的尽力服务任务,如文字、图像下载等,还要提供有时间性能要求的软实时请求的服务,如流媒体服务。虽然多媒体服务器接受的任务复杂多样,大致上可以化分为这两类[6]:尽力服务请求和软实时服务请求。因此,我们将磁盘的读写请求对应地分为两类:实时请求和尽力服务请求。

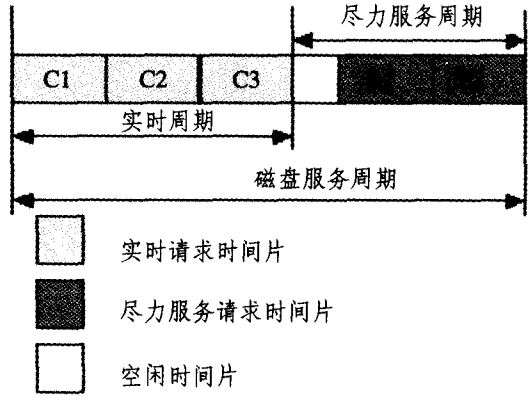


图1 磁盘服务周期的划分

*)河南省高校杰出人才创新工程和河南省骨干教师项目资助。厉 励 讲师,研究方向:图像处理。张宏坡 工程师,研究方向:计算机网络。李 海 工程师,研究方向:数字视频压缩编码。周 兵 博士,教授,研究方向:数字视频处理。

针对上述划分，这里磁盘调度采用一种基于服务周期定时启动调度算法的策略[4]，并将服务周期划分为两部分(如图1)，一部分用于服务实时请求，一部分用于服务尽力服务类型的请求。这种划分可以保证非实时请求不会得不到服务或等待的时间过长，具体周期比例的划分要依据应用的具体环境而定。

服务周期划分的比例可以通过手工来确定，根据以往的经验值或者估计将来的负载情况，由人工确定这个比例。这种方法可以调整以天为单位的变化，而对短暂的变化却无能为力[5]。因此本文提出一种根据实际监测到的两类请求的负载情况，自适应调整磁盘服务周期的分配策略，就是要动态调整服务周期的划分比例。

3 QoS机制

当新的实时请求到来时，首先判断是否接受该请求。假设根据测试得到的第 n 个实时请求任务的磁盘请求占用率为 u_n，则在满足 $\sum_i \frac{u_i}{P_i} \leqslant U_{rt}$ 情况下接受请求，其中 P_i 为第 i 个任务的周期，U_{rt} 为实时请求占用的服务周期。

4 磁盘调度模型

本文讨论的问题是在服务器具有一个磁盘的情况下，它要为两类请求服务：尽力服务请求和实时请求。R_{be} 和 R_{rt} 分别表示为两类请求的服务周期的比例，其中 $0< R_{be}, R_{rt}<1$；$R_{be}=1-R_{rt}$。

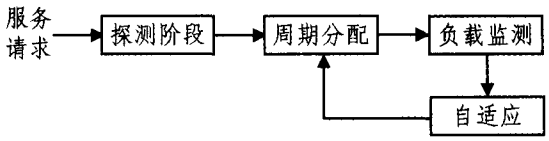


图2 调度模型

4.1 负载监测模块

负载监测模块通过一个滑动的检测窗口，来监测两类应用的负载变化情况。如图3所示。

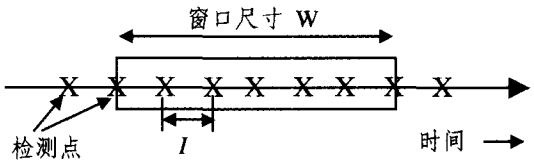


图3 监测窗口

窗口包括两个参数，窗口的尺寸 W，它记录了在此时间内有多少个以 I 为单位的时间段进入窗口；检测间隔 I 表示两次检测之间的时间间隔。那么 W/I 就表示每个窗口包含的检测次数。

从一个服务请求到达开始，监测模块跟踪记录该请求使用各种资源的情况，直到该请求得到磁盘的服务。监测模块需要对每类请求的如下参数进行监测(如图4)：请求到达的速率、请求等待时间、磁盘利用率。

(1)请求到达的速率。监测模块记录每个测量间隔 I 中到达的请求数(N_{be} 和 N_{rt})以及请求的平均尺寸(S_{be} 和 S_{rt})。

(2)请求等待时间。这里不用请求的实际等待时间，而是用请求队列的长度来表示一个请求等待的时间(Q_{be} 和 Q_{rt})，具体就是用每个检测间隔末尾的瞬间队列长度来表示。

(3)磁盘利用率。监测模块利用磁盘的利用率作为每类应用请求占用的服务周期。磁盘利用率定义为磁盘为每类请求服务的时间与检测间隔的比：

$$U_{be}=\frac{\sum_j T_{be}^j}{I}, U_{rt}=\frac{\sum_j T_{rt}^j}{I}$$

其中，T_{be}^j 表示为每个尽力服务请求服务的时间；T_{rt}^j 表示为每个实时服务请求服务的时间。

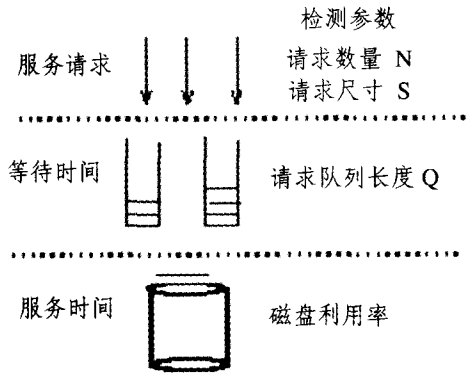


图4 监测参数

服务器负载监测模块通过检测窗口(W)的检测每类应用的负载情况，自适应管理模块根据监测到的数据，周期性地调整每类应用请求占用比例。这里涉及到一个参数 P，即，多长时间进行一次调整。监测模块的时间窗口大小 W 和 P 的关系是：

如果 $P<W$，那么就有一些检测点被重复使用，因为这些检测点还没有移出检测的窗口。

如果 $P>W$，那么就会有一些检测点永远也不会被考虑进去。因此，为避免上述问题，选择 $P=W$。

另外，自适应管理模块对服务周期的调整要遵循一定约束条件，这些约束规则由管理员来制定，如规定每类应用请求的最大和最小占用比例$[R_{be,min}, R_{be,max}]$，$[R_{rt,min}, R_{rt,max}]$。这样就可以避免对每类应用请求分配过大或过小服务周期。

4.2 自适应模块

自适应模块根据负载监测模块检测到的参数，动态调整磁盘服务周期在不同应用请求之间的分配。这里，可以按磁盘的利用率来估算分配比例，但在服务器过载的情况下，由于磁盘的利用率为百分之百，所以就不能再按磁盘的利用率来估算了，因此，在这种情况下，应按照两类应用请求到达的速率进行估算。

(1)根据磁盘利用率估计周期划分比例

自适应模块根据负载监测模块获得磁盘利用率，来估算周期划分比例。由于每类应用请求对性能的要求不同，因此要采取不同的估算策略。对于尽力服务请求，由于它对响应时间要求较低，采用磁盘占用的平均值来估计，对于 t 时刻的估计值表示为 $\mathrm{Median}^t(U_{be})$。对于实时服务请求，由于它有响应时间的要求，因此应当分配较高比例的磁盘占用，t 时刻的占用比例表示为 $\mathrm{Perc}^t(U_{rt})$。对于两类应用请求所占用的百分比，可以采用静态或动态的分配策略。如果采用动态分配，可以根据负载变化来调整分配的比例，变化大，分配的比例就高。分配比例可按如下公式计算：

$\mathrm{Perc}(U_{rt}) = C_0+\log(C_v)$；$C_v$ 是一个变化系数。

C_0 为分配比例的初始值；$C_v = \sigma(U_{rt})/E(U_{rt})$；$E$ 为磁盘利用率的平均差，σ 为磁盘利用率的标准差。

计算出磁盘利用率后，自适应模块采用指数平滑的方法计算周期划分比例。

$$Median^{t+1}(U_{hr}) = \alpha * Median^{t}(U_{hr}) + (1-\alpha) * Median^{t}(U_{hr}) \quad (1)$$

$$Perc^{t+1}(U_{rt}) = \alpha * Perc^{t}(U_{rt}) + (1-\alpha) * Perc^{t}(U_{rt}) \quad (2)$$

其中，α 是指数平滑参数，$0 \leqslant \alpha \leqslant 1$，$\alpha$ 越大，说明当前变化对周期分配的影响越大。

(2)根据请求到达的速率来估计周期划分比例

采用磁盘利用率可以很好地表示各类应用请求所需要的周期，但是在过载的时候，磁盘是饱和的，其利用率是 100%，所以不能反映出各类应用请求实际所需要的。因此，在短暂过载的情况下，自适应模块采用请求到达速率来来估计每类应用请求所需要的周期。通常。到达速率高的请求应该分配高比例的服务周期。但是，在过载的时候已经无法满足实际请求的需要，自适应模块的主要任务是保证服务器在过载时采取调整动作的稳定性，并反映出两类请求对服务周期要求的相对差异。

自适应模块首先计算出请求到达的数量以及要求的尺寸大小，然后按下面两式分别计算出两类应用请求所需要的服务周期：

$$R'_{hr} = Median^{t}(N_{hr}) * (t_{seek} + t_{rot} + Median^{t}(S_{rt}) / t_{xfr}) \quad (3)$$

$$R'_{rt} = Perc^{t}(N_{rt}) * (t_{seek} + t_{rot} + Perc^{t}(S_{rt}) / t_{xfr}) \quad (4)$$

其中 t_{seek} 为平均寻道时间，t_{rot} 为平均旋转等待时间，t_{xfr} 为磁盘传输数据的速率。(3)、(4)等式中的第一项反映了磁盘请求的数量，第二项反映了磁盘为每个请求的服务时间。

(3)计算周期划分比例

自适应模块首先初始化，由管理员设置的服务周期最初的划分比例(R^0_{hr} 和 R^0_{rt})，在每个周期 P，自适应模块估算每类应用请求周期占用，并按如下算法对周期划分做出调整：

• Case1：$Median^{t}(U_{hr}) < R^0_{hr}$ AND $Perc^{t}(U_{rt}) < R^0_{rt}$。即两类请求的实际需要都小于初始设置的值，所以不做任何调整。

• Case2：$Median^{t}(U_{hr}) \geqslant R^0_{hr}$ AND $Perc(U_{rt}) < R^0_{rt}$。即尽力服务请求的需要大于实际设置的值，而实时服务请求的需要小于实际设置的值，自适应模块就增加尽力服务请求的占用比例，减少实时服务请求的占用比例。

$$R^{t+1}_{hr} = Median^{t}(U_{hr}) \quad (5)$$

$$R^{t+1}_{rt} = 1 - R^{t+1}_{hr} \quad (6)$$

其中，R^{t+1}_{hr} 为尽力服务请求新的(下一时刻)占用比例，R^{t+1}_{rt} 为实时服务请求新的占用比例。

• Case3：$Median^{t}(U_{hr}) < R^0_{hr}$ AND $Perc^{t}(U_{rt}) \geqslant R^0_{rt}$。即尽力服务请求的需要小于实际设置的值，而实时服务请求的需要大于实际设置的值，自适应模块就增加实时服务请求的占用，减少尽力服务请求的占用比例。

$$R^{t+1}_{rt} = Perc^{t}(U_{rt}) \quad (7)$$

$$R^{t+1}_{hr} = 1 - R^{t+1}_{rt} \quad (8)$$

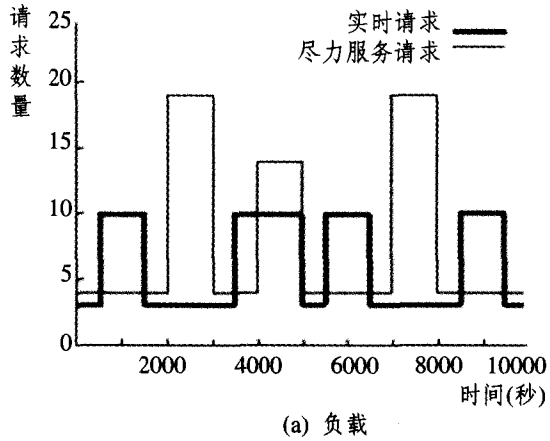
• Case4：$Median^{t}(U_{hr}) \geqslant R^0_{hr}$ AND $Perc^{t}(U_{rt}) \geqslant R^0_{rt}$ 或者 $Q_{hr} \geqslant Q$ AND $Q_{rt} \geqslant Q$。其中 Q 是设置的一个很大的门限值。此时服务器工作在过载状态下，自适应模块根据请求到达的速率，估算周期划分比例(R_{rt} 和 R_{hr})，然后按下式计算出新分配比例：

$$R^{t+1}_{hr} = \frac{R'_{hr}}{(R'_{hr} + R'_{rt})} \quad (9)$$

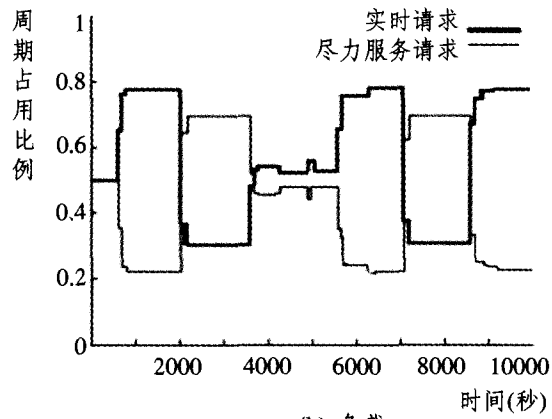
$$R^{t+1}_{rt} = \frac{R'_{rt}}{(R'_{hr} + R'_{rt})} \quad (10)$$

4.3 实验结果

服务请求分为尽力服务请求和实时视频服务请求，其中 $W=P=100$ 秒；$I=1$ 秒；$\alpha=0.75$；估计实时请求服务周期占用比例为 90%。



(a) 负载



(b) 负载

图 5 实验结果

图 5(a)为请求到达的分布图，(b)为两类请求服务周期占用比例。从图中可以看出，自适应模块可以根据每类请求的负载情况，按比例动态调整服务周期的分配。请求分布涵盖了上述算法的四种情况，在 500 秒时，实时请求突然增加，自适应模块调整周期的分配，实时请求占 80%，尽力服务请求占 20%；2000 秒时，尽力服务请求突然增加，自适应模块做出对应调整，降低实时请求的周期占用，增加尽力服务请求的服务周期；4000 秒时，两类请求数量都很高，此时处于过载状态，自适应模块为保持服务器的稳定性，对周期分配比例不做大的调整，使两类请求占用比例维持在一个相对平滑的状态。

5 算法性能分析

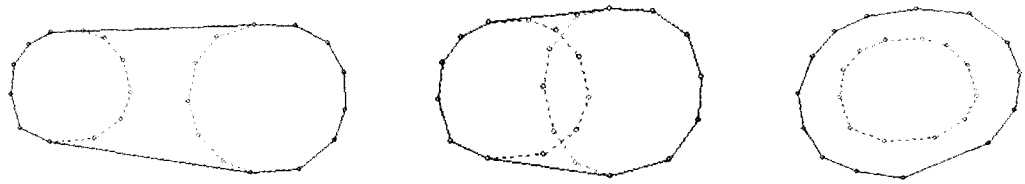
本算法应用在数字化远程监控系统中。该系统采用 C/S 结构，由监控服务器和远程客户端组成。监控服务器负责采集多路音视频数据、编码存储、为远程客户端提供远程回放等

(下转第 221 页)

5 复杂度分析及实验结果

在本文的凸壳算法中，对于顶点分别为 m 和 n 的两个凸多边形，寻找 8 个极值点的时间复杂度分别为 $O(m)$ 和 $O(n)$，由于对凸多边形进行分段处理，每段上凸壳顶点的计算只与两个同类的单调段有关，与其他段无关，每个点处理一次，因而对两个凸多边形的所有顶点进行处理的时间复杂度为 $O(m+n)$，对临时凸壳上的点进行回朔处理的时间复杂度为 $O(m+n)$，所以整个算法最坏时的时间复杂度为 $O(m+n)$，本文算法的时间复杂度低。

对本文的算法进行大量的实验，都能正确计算出两个凸多边形的凸壳。对于多个凸多边形的凸壳算法，可计算两个凸多边形的凸壳，再与另外的凸多边形进行计算。图 4 为采用本文算法生成的凸壳(虚线为凸多边形，实线为本文算法求出的凸壳)，图 4(a)为不相交的二个凸多边形的凸壳，图 4(b)为相交的二个凸多边形的凸壳，图 4(c)为两个凸多边形包含情况的凸壳。



(a) 二个凸多边形不相交　(b) 二个凸多边形相交　(c) 二个凸多边形包含

图 4　凸多边形的凸壳

结论　通过极值确定了凸壳的四个段，凸壳的每段都是单调的，根据这一性质计算两个凸多边形的凸壳算法简单可行；对每个段分别进行计算，减小了点集的规模，提高了运算速度，另一方面利用了凸多边形顶点本身有序的特点，不需进行排序，从上几个方面进行处理降低了计算复杂度；理论分析和实验表明本文算法的性能好，具有较强的实用性。下一步将应用凸壳的性质对平面点集的凸壳问题进行研究。

参 考 文 献

1 Chand D R, Kapur S S. An algorithm for convex polytopes [J]. JACM, 1970, 17(1): 78～86
2 Jarvis R A. On the identification of the convex hull of a finite set of points in the plane [J]. Information Processing Letters, 1973, 2(1): 18～21
3 Graham R L. An efficient algorithm for determine the convex hull of a finite linear set [J]. Information Processing Letters, 1972, 1(1): 132～133
4 Preparata F P, Hong S J. Convex hulls of finite sets of points in two and three dimensions [J]. CACM, 1977, 20(2): 87～93
5 周培德. 寻求平面上线段集的凸壳的算法[J]. 工程图学学报, 2003, 23(2): 116～119
6 崔国华，洪帆，余祥宣. 确定平面点集凸包的一类最优算法[J]. 计算机学报, 1997, 20(4): 330～334
7 金文华，何涛，刘晓平，等. 基于有序简单多边形的平面点集凸包快速求取算法[J]. 计算机学报, 1998, 21(6): 533～539
8 Chan T. Optimal output-sensitive convex hull algorithms in two and three dimensions [J]. Discrete Comput Geom, 1996, 16(3): 361～368
9 Brönnimann H, Iacono J, Katajainen J, et al. Space-efficient planar convex hull algorithms [J]. Theoretical Computer Science, 2004, 321(1): 25～40
10 Yao A C. A lower bound to finding convex hulls [J]. J ACM, 1981, 28(4): 780～787

(上接第 120 页)

功能，其中采集编码都用软件方法实现。由于监控服务器既要负责多路音视频的存储，又要提供远程监控数据的检索功能，因此对磁盘的访问很频繁，磁盘调度问题就成为监控服务器的性能瓶颈。

实验 1　没有调用本算法。监控服务器同时采集 2 路音视频数据，实时监控画面基本流畅，远程数据检索基本达到性能要求；监控服务器同时采集 4 路音视频数据，实时监控画面延迟明显，偶尔有跳帧现象，远程数据检索反映很慢，还出现检索不成功的情况。

实验 2　采用本算法。监控服务器同时采集 2 路音视频数据，实时监控画面流畅，远程数据检索响应很快；监控服务器同时采集 4 路音视频数据，实时监控画面基本流畅，没有出现跳帧现象，远程数据检索响应基本正常，没有出现检索不成功的情况。

通过实验可以看出，本算法的应用，基本解决了在以软件方法实现的监控系统中，由于存储大量数据而造成的实时显示画面延迟大，远程数据检索响应慢的问题；同时也提高了监控服务器提供服务的路数。

结论　本文提出了一种流媒体服务器磁盘调度策略，给出了算法主要模块探测模块、负载监测模块和自适应管理模块的具体实现。该策略应用到基于软件压缩的多路视频监控系统中，视频服务器同时提供实时视频远程服务和历史视频的远程查询服务功能，在发送给客户端视频数据的同时，还要保存服务端采集到的音视频数据，这就会产生大量的数据流，因此，造成对磁盘的频繁访问。而且视频服务器还要接受一些控制命令，如果不及时响应，会造成监控数据的丢失。通过应用本调度算法，很好地解决了这个问题，使得访问音视频流数据的实时请求不会错过截止时间，也不会造成控制命令等尽力服务请求长时间得不到服务。自适应管理策略的应用，增加了视频服务器的易管理性，也增强了它的稳定性。

参 考 文 献

1 Aurrecoechea C, Campbell A T, Hauw L. A survey of QoS architecture [J]. Multimedia System, 1998, (6): 138～151
2 丁峰，邓勇，沈钧毅. 一种在分布式系统中支持 QoS 的方法研究[J]. 小型微型计算机系统, 2000, 22(1)
3 Nerjes G, Muth P, Paterakis M, et al. Incremental scheduling of mixed workloads in multimedia information servers[J]. Multimedia Tools and Applications, 2000, 11: 9～33
4 石柯. 支持 QoS 的操作系统框架的研究[J]. 计算机工程与应用, 2002(7): 110～112
5 Alvarez G, Keeton K, Merchant A, et al. Storage Systems Management[C]. Tutorial presented at ACM Sigmetrics 2000, Santa Clara, CA, June 2000
6 Pradhan P, Tewari R, Sahu S, et al. An Observation－based Approach Towards self－managing Web Servers[C]. In: Proc. of ACM/IEEE IntlWorkshop on Quality of Service (IWQoS), Miami Beach, FL, May 2002
7 Chesire M, Wolman A, Voelker G, et al. Measurement and Analysis of a Streaming Workload[C]. In: Proc. of the USENIX Symposium on Internet Technology and Systems (USEITS), San Francisco, CA, March 2001
8 Revel D, McNamee D, Pu C, et al. Feedback Based Dynamic Proportion Allocation for Disk I/O[J]: [Technical Report CSE-99-001]. OGI CSE, January 1999
9 Hennessy J. The Future of Systems Research[J]. IEEE Computer, August 1999. 27～33
10 Sundaram V, Shenoy P. A Practical Learning-Based Approach for Dynamic Storage Bandwidth Allocation[J]. In: Proc. of ACM/IEEE Intl Workshop on Quality of Service (IWQoS), 2003, LNCS 2707. 479～497