具有高缓存写入效率的流媒体分段缓存方法

马 杰[1),2)] 樊建平[2),3)]

[1)](中国科学院研究生院 北京 100080)

[2)](中国科学院计算技术研究所 北京 100080)

[3)](中国科学院深圳先进技术研究院 广东深圳 518067)

摘 要 流媒体代理服务器缓存是能有效降低网络传输负载的技术.长时间持续和大传输码率的两个流媒体访问特点使得流媒体代理服务器面临的负载问题十分严峻.流媒体缓存方法是流媒体代理服务器的核心组成,其引发的缓存写入操作数量对代理服务器负载有着重要的影响.文中从流媒体缓存的执行特点入手,给出了一种高网络传输减少效果和低缓存写入负载的流媒体分段缓存方法.缓存写入与访问热度相结合是该缓存方法的主要特点.在实验测试中证明了该缓存方法相比目前减少网络传输最好的 Adaptive & Lazy 缓存方法能减少 2/3 的缓存写入负载,并能获得同样的网络传输减少效果.

关键词 代理服务器;流媒体缓存;分段缓存方法;缓存写入效率

中图法分类号 TP311

A New Segmented Cache of Streaming Media with High Cache-Writing Efficiency

MA Jie[1),2)] FAN Jian-Ping[2),3)]

[1)](*Graduate University of Chinese Academy of Sciences, Beijing* 100080)

[2)](*Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100080)

[3)](*Institute of Advanced Technology, Chinese Academy of Science, Shenzhen, Guangdong* 518067)

Abstract Accessing multimedia files with streaming media technique will generate data streams with long time existed and high bandwidth consumed. Streaming media cache in the proxy server does good in reducing network traffic of streaming media. The continuous data transmission brings great challenge to the proxy sever, and it is importance to find out how to reduce cache-writing load in order to avoid the damage to the proxy service. Cache method is the core part of the streaming media cache, since it determines how proxy server works. This paper introduces a new streaming media cache method which has good effect both on improving cache-writing efficiency and reducing network traffic. Through the simulation by actual access logs, the new cache method has the same transmission saving and promote the writing efficiency as three times as the Adaptive & Lazy method.

Keywords proxy server; streaming media cache; segmented-cache method; cache-writing efficiency

1 引 言

网络传输能力和服务器服务能力是限制大规模流媒体应用的两个瓶颈,流媒体代理服务器缓存是能有效缓解这两个瓶颈的技术.流媒体访问的高传输码率和长时间持续两个特点导致代理服务器经常运行在高负载状态下.

收稿日期:2006-01-04;修改稿收到日期:2006-11-28.**马 杰**,男,1976年生,博士,研究方向为流媒体传输、流媒体服务器、网络安全、密码安全、信息系统.E-mail: tintin_7607@163.com.**樊建平**,男,1963年生,博士,博士生导师,研究领域为体系结构、系统软件、并行处理.

在流媒体缓存过程中，流媒体对象的大体积使得缓存写入操作数目显著增加.从负载角度来看，缓存写入操作在消耗代理服务器处理资源的同时也在和服务用户访问所需的缓存读取操作竞争磁盘吞吐带宽.从缓存效果来看，作为缓存内容来源的缓存写入操作是未来缓存命中服务的基础.这种两面性使得高效的缓存写入过程成为表征缓存方法实用性的重要依据.

分段缓存方法具有实现便捷和网络传输减少效果突出的特点[1-8].作为网络传输减少效果最好的分段缓存方法，Adaptive & Lazy 缓存[7-9]使用平均访问量大小决定分段大小.由于流媒体访问具有少量访问中途终止和大量访问完全浏览的趋向[10-11]，平均访问量通常会达到流媒体对象体积的一半以上.过大的分段体积在缓存更新的过程中会引发众多的缓存写入操作，进而降低 Adaptive & Lazy 缓存的实际应用能力.

本文介绍一种缓存写入高效的分段缓存方法，在保证网络传输减少效果的同时能有效减少缓存写入操作.固定分段的保存方式，根据当前访问热度决定缓存内容选取量的进入策略，考虑当前访问热度和历史访问热度的替换算法是该缓存方法的三个组成部分.在测试对比中，该缓存方法在获得了与 Adaptive & Lazy 缓存相同网络传输减少效果的同时，缓存写入效率结果为 Adaptive & Lazy 缓存结果的 3 倍以上.

本文第 2 节介绍流媒体缓存的相关研究工作；第 3 节介绍流媒体缓存的特点和请求区间的定义；第 4 节详细介绍释放操作相关缓存方法，具体分为保存方式、进入策略、替换算法 3 个部分；第 5 节介绍与已有典型缓存方法的对比测试工作；第 6 节对本文的内容进行归纳并介绍随后工作的思路.

2　相关工作

Sliding Cache[12]是最早提出的流媒体缓存方法.通过缓存一段变化的内容片段来发现访问间隔小的流媒体访问是 Sliding Cache 的主要思路.为了减少流媒体访问的初始访问延迟，前缀缓存(Prefix Cache)[13]提出在服务器内存中或代理服务器上保存流媒体对象头部内容的方法.

在文献[6]中，流媒体对象的内容被分成大小不等的段，这些段的长度从头部开始指数变化式地增长.结果表明指数缓存方式在网络传输减少效果方面明显优于前缀缓存方法.

Adaptive 和 Lazy 缓存方法[7-8]指出：为头部内容划分专门空间[6]、缓存头部固定内容[13]都不是最好的缓存方式.收集请求并使用平均访问量大小确定分段是 Adative 和 Lazy 缓存提出的新方式，并在文献[7-9]中被证明具有最好的网络传输减少效果.

针对网络传输能力小于最大播放码率时 VBR 流媒体对象无法正常播放的实际问题，Video Staging 方案[14]给出一种解决方法：将 VBR 流媒体对象传输码率大于网络传输能力时超出部分保存在代理服务器上，并利用这部分内容弥补传输码率超出网络传输能力时被丢弃的数据包.

利用内容提供者标记的热点内容位置，文献[15]介绍了缓存关键段的流媒体缓存方法.文献[4]中将流媒体对象分割成多个块，并针对每个块进行独立的缓存操作.为了增加 VBR 流媒体对象访问高峰时期的播放质量，文献[16]给出了缓存码率高峰前数个帧内容的方式.

在其它方面，多个代理服务器协同缓存是流媒体缓存的常见研究方向[2,17].文献[6,14]介绍了如何针对分层编码的流媒体对象进行代理服务器缓存.文献[13,18-19]给出适合流媒体缓存的替换算法.

文献[9]中将已有流媒体缓存研究进行了分类和对比，结果显示出所有的流媒体缓存方式都具有可观或突出的磁盘 I/O 操作数量.但在这些已有的研究中都没有考虑缓存写入操作的执行效率.

3　请求区间

部分缓存方式是流媒体缓存过程的特点.与完整缓存的传统互联网缓存不同，流媒体对象的缓存过程存在没有缓存内容、部分内容缓存和完整缓存三种状态.

部分内容缓存状态是流媒体缓存过程的特殊阶段(图 1)，缓存内容随着访问请求和释放操作间隔发生而间歇变化是这一状态的特征.依据这个特征，下面给出连续请求和请求区间的定义.

连续请求. 对于每个流媒体对象来说，如果在两个相邻请求之间没有发生释放操作，就称这两个请求是连续请求.而连续请求数目是指连续多少个请求没有被释放操作所中断.

请求区间. 对每个流媒体对象来说，释放操作的发生使得所有发生过的请求被分成一个个连续请求序列，而在每个连续请求序列中缓存内容保持着

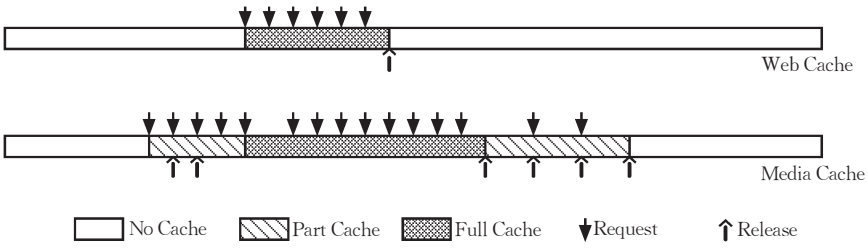


图 1 缓存内容状态

递增的趋势.这些彼此隔离的连续请求序列被定义为请求区间.

连续请求和释放操作是请求区间的决定元素,分别反映不同的访问热度变化.由于替换选取命中是释放操作发生的前提,选取访问热度低的替换选取原则使得释放操作能用来体现访问热度的下降.而最近访问是体现受欢迎度的有效依据,当请求到来时缓存系统通常都会引发内容保存的操作.根据上面连续请求的定义,连续请求之间不存在体现访问热度下降的释放操作,结合连续请求会引发缓存内容的持续增长的特性,连续请求的出现可看作是访问热度上升.

根据两个决定元素所反映的访问热度的变化趋势,请求区间和访问热度可被看作存在以下关联:(1)请求区间之间发生了释放操作,请求区间之间的访问热度具有隔离性.(2)连续请求是请求区间的组成,结合请求区间之间的访问热度隔离性,当前请求区间的连续请求数目具有体现当前访问热度大小的能力.相比传统访问热度表征方式(例如访问频率),由于考虑了释放操作表征的访问热度降低,请求区间与访问热度的关联能更准确地反映缓存执行过程的真实情况.

4 释放操作相关缓存方法

本节将详细介绍利用请求区间与访问热度关联而设计的流媒体分段缓存方法.由于利用了释放操作作为缓存设计的依据,该缓存方法被称之为释放操作相关缓存方法.

4.1 保存方式

在分段缓存方法中,逻辑块和分段是保存方式的两个基本概念.其中,逻辑块是存储的基本空间单位,包含多个逻辑块的分段是缓存执行的基本空间单位.变动和固定是分段大小的两种确定方式.

考虑以下 4 个方面,保存方式采用较大体积的固定分段:(1)逻辑块体积较小时变动分段方式会加重缓存内容的分散程度,也就增加了内容读取的时间;(2)变动分段很容易造成每次分配空间体积过大,在访问密集时降低缓存内容随请求而更新的敏感度;(3)固定分段的体积过小会增加缓存判断、空间分配、I/O 操作启动、进程切换等操作的执行频率;(4)较大体积的分段可以包含数量更多的图像帧,从而有效降低流媒体编码方式对缓存过程的影响[6-9].

和文献[6-9]等研究采用的方式相同,保存方式的较大体积固定分段主要是从头部开始、按照设定的缓存分段大小对流媒体对象进行分段.

4.2 进入策略

通过给请求设定缓存写入限制量的方式来达到缓存内容选取与当前访问热度相结合是进入策略的主要思路.缓存写入限制量是指一个请求所能带来的最大缓存内容增长幅度.利用请求区间连续请求数目和当前访问热度的关联,缓存写入限制量大小具有随着连续请求数目增多而指数增长的属性.

缓存写入限制量指数变化的基础是连续请求为 1 时的取值,这个取值称之为初始值.在进入策略中,缓存写入限制量被赋予初始值的请求有以下两类:

(1)流媒体对象的头次请求.热点对象的头次请求之后存在一个访问密集时期是互联网访问的特有现象[10-11].根据这种访问现象,赋予头次请求大的初始值能有利于提升缓存命中效果.但由于热点对象的比例通常较小,过大的取值也会带来过多的无效缓存写入.从这两方面出发,头次请求的初始值大小设定为$\frac{N}{m}$,其中 N 为流媒体对象的体积,m 的取值决定了在几个连续请求的作用下能完整缓存流媒体对象的内容.

(2)紧接释放操作的请求.访问热度逐步下降是日志分析研究归纳的另一个互联网访问现象[10-11].连续请求逐渐减少而连续释放次数逐渐增多是访问热度逐步下降的体现,在访问热度逐渐下降的过程中连续释放操作发生后访问热度依然会相当可观.根据访问热度过低会引发完全释放的缓存特点,释放

操作发生后存在剩余缓存内容可被认为存在剩余访问热度，而释放之前缓存内容长度 $N_{\max}$ 与连续释放次数 H_{current} 的比值可用来表征访问热度的剩余值. 根据这些设定这类请求的初始值在没有剩余缓存内容时为 1，在有剩余缓存内容时为 $\frac{N_{\max}}{H_{\text{current}}}$.

另外，并发请求和大量空间空闲是缓存选取需要考虑的两种特殊情况. 当不同访问间的时间间隔小于缓存内容的时间长度时多个请求会同时访问缓存内容. 在这种并发情况下将后面请求的写入内容限制量转嫁到最前请求上可以提升后面请求的缓存服务量. 空闲缓存空间的服务几率始终为零，在缓存空间存在空闲时暂时取消缓存写入限制量的限制会更有利于提高缓存服务几率.

根据上面的思路，进入策略具体如下：

1. 缓存操作从 $N_{\text{current}}+1$ 个分段开始，N_{current} 为已缓存内容的分段数.

2. 每个请求到来时会被赋予一个缓存写入限制量 L，L 取值如下：

a) 为流媒体对象的第一个请求时，L 等于 $\frac{N}{m}$；

b) 请求之前刚发生过释放操作且目前存在缓存内容时，L 等于 $\frac{N_{\max}}{H_{\text{current}}}$；

c) 请求之前刚发生过释放操作且目前没有存在缓存内容时，L 等于 1.

d) 接到前一个请求时，$L=k\times L'$，k 为大于 1 的整数，L' 为上一个请求的缓存写入限制量.

3. 在有 n 个并发请求的情况下，第一个请求的缓存写入限制量将变为 $L_{\text{sum}}=\sum_{1}^{n}L_x$.

4. 缓存空间的申请为每次一个分段.

5. 在无法获得空闲空间或者缓存写入量超过 L 时缓存操作终止.

6. 缓存操作没引发替换操作之前不会判断缓存写入量是否超过 L.

和指数缓存利用距离头部位置、Lazy 缓存利用平均访问量而确定缓存写入量的两种方式相比，由于访问热度体现了缓存内容被再次访问的几率，上述围绕热点内容执行缓存写入的进入策略更有利于提升缓存写入效率.

4.3　替换算法

再次访问几率是影响替换选取结果的重要因素，请求区间与访问热度的关联对于替换选取有着明显的指导作用. 首先，当前请求区间的连续请求数目 R_{current} 可体现当前访问热度，访问热度高则再次访问几率就相对要大. 其次，过去的每个连续请求序列都表示了当时的访问热度，请求总数目 R_{sum} 和请求区间数目 A_{num} 的比值 $\frac{R_{\text{sum}}}{A_{\text{sum}}}$ 也就能体现过去的平均访问热度. 平均访问热度大的内容通常比平均访问热度低的内容具有更高的再次访问几率.

除上述两方面之外，下面几个因素对释放选取也有指导意义：(1) 最近请求到来时间 T_{recent} 越近的缓存内容通常再次访问几率就越高；(2) 为了平均存储资源的使用和减少完全释放发生，缓存内容长度 N_{current} 大的内容应该具有一定程度上的释放优先级；(3) 减缓缓存内容的连续释放速度能避免偶然释放带来的缓存内容损失，这需要连续释放次数 H_{current} 高的缓存内容具有高一些的保存优先级.

根据上述几个方面，释放选取函数可表示如下：

$$u(x)=\frac{\left(\frac{R_{\text{sum}}}{A_{\text{sum}}}+R_{\text{current}}+H_{\text{current}}\right)}{(T_{\text{current}}-T_r)\times N_{\text{current}}}.$$

为了避免低选取函数值的内容替换高选取函数值的内容，释放选取结果还将与写入对象再进行比较. 若释放选取结果小于写入对象则进行执行释放工作，否则拒绝缓存写入.

释放缓存内容尾部的 $2^{H_{\text{current}}}$ 个分段是释放操作的执行方式，其中 H_{current} 为当前的连续释放次数. 根据连续释放次数而指数递增释放量的释放方式主要考虑以下两个方面：

(1) 偶然发生的释放操作不会损失大体积的热点对象缓存内容，进而在访问集中时期可有效避免偶然因素引发大体积的缓存内容交换.

(2) 对象访问热度长期处于最低值和当前缓存写入频繁是导致连续释放次数增多的两个因素，在这两种情况下释放工作都应该加快释放的步伐.

和已有流媒体替换算法相比，上述替换算法有两个新颖之处. 首先，在替换选取方面延长了高连续请求数目的缓存内容存放时间. 其次，在内容释放方面避免了偶然释放操作引发的缓存内容交换幅度. 结合进入策略中根据连续请求而指数变化缓存内容量的措施，释放操作相关的缓存方法在写入和释放两方面都考虑请求区间和访问热度的关联.

5　实验验证和结果分析

实验内容和实验环境：

模拟测试工作是通过 LittleDuck 模拟器[20] 所完成的. 相比 MiddleSim[2]，由于具有事件驱动、缓存方式独立、层次化结构的 3 个特征，LittleDuck 模

拟器在结果准确的基础上模拟速度更加快速,并且能便捷地加入各种缓存方式和搭建不同的缓存场景[20].

在模拟测试中,指数分段缓存和 Adaptive & Lazy 缓存被用来作为对比参照.其中,Adaptive & Lazy 缓存[7-8]是目前网络传输减少效果最好的缓存方法[9].指数分段缓存[6]根据头部距离而指数分段的方式与本文缓存方法的缓存内容选取方式具有一定的相似性.其他相近缓存方法在文献[6-8]中已被证明在网络传输减少效果方面明显小于指数分段缓存和 Adaptive & Lazy 缓存.

参照文献[6-8],指数分段缓存的实现预留了$\frac{1}{10}$的缓存空间来存放头部内容.Adaptive & Lazy 缓存选取函数的指数因子组合设定为($p_1=1, p_2=1, p_3=1$).而释放操作相关缓存方法中,缓存递增指数 k 设定为 2、与头次请求初始值相关的 m 设定为 3、固定分段的大小为 $500k$.

缓存效果评价使用了传输减少率、请求命中次数和字节利用率 3 个指标.其中,反映网络传输减少的传输减少率为缓存带来的网络传输减少量和用户总访问量的比值.反映访问延迟缩短的请求命中次数为请求到达代理服务器时缓存内容存在的次数.反映缓存写入效率的字节利用率为网络传输减少量和缓存写入内容量的比值.

实验输入:

测试中使用了 4 个日志.Campus[2]与 hpc[11]分别来源于美国校园网上的流媒体服务器和互联网上 HP 公司的流媒体服务器,具备时间跨度长、访问频繁度低的特性.Short 与 report 是分别通过 medisyn 生成器[21]与 gimso 生成器[22]获得的访问日志,都具有时间跨度小、访问频率高的属性.4 个日志都具有请求与流媒体对象的关联符合 zipf 分布($f(x)=\frac{1}{\Omega x^{\alpha}}$, $x\in(1,n)$)、请求到来时间遵循 pdf 分布($f(x)=k\times(1-x)^{-\frac{1}{\lambda}}$)的访问规律(表 1).

表 1 测试使用的日志

日志名称	流媒体对象数目	日志持续时间	请求数目/万	α	λ	k
hpc	1434	30 个月	32.0000	—	—	—
campus	141	6 个月	0.1632	—	—	—
short	1000	4 天	10.0000	1.50	—	—
report	500	7 天	1.5000	0.73	1	0.001

实验结果:

由于以下两个原因,实验分析中主要讨论空间比例为 50%以下的测试结果:(1)在缓存应用中,缓存空间的体积大小通常会明显小于可访问内容的总量,在大体积比例下的测试结果不具有实际对比意义;(2)在大体积比例下各种缓存方式都接近完整的缓存方式,测试结果会十分接近.

表 2 传输减少率

(a)

空间比例/%	传输减少率/%					
	campus			hpc		
	释放操作相关缓存	A&L 缓存	指数缓存	释放操作相关缓存	A&L 缓存	指数缓存
5	22.41	21.21	0.93	83.02	78.40	6.95
10	49.38	48.57	7.38	94.59	91.37	45.19
15	67.86	57.57	63.37	95.22	94.63	70.27
20	79.40	64.27	72.23	96.79	95.83	95.40
25	85.07	69.43	80.74	97.46	96.8	98.23
30	88.16	70.88	86.40	97.64	97.21	98.36
35	89.34	72.81	88.70	97.92	97.60	98.41
40	90.06	75.74	89.25	98.00	97.79	98.42
45	90.20	78.62	90.16	97.93	97.89	98.41
50	90.87	82.09	90.72	97.85	97.97	98.43

(b)

空间比例/%	传输减少率/%					
	short			report		
	释放操作相关缓存	A&L 缓存	指数缓存	释放操作相关缓存	A&L 缓存	指数缓存
5	75.22	69.64	19.81	16.24	11.66	3.12
10	89.74	89.49	27.02	50.89	47.64	8.10
15	92.57	92.53	55.99	63.61	60.10	14.88
20	93.78	93.87	69.16	69.95	66.37	24.30
25	94.42	94.66	85.02	74.07	71.85	35.08
30	95.00	95.06	92.45	76.95	74.96	50.61
35	95.29	95.24	94.23	79.03	77.73	74.55
40	95.45	95.42	94.94	80.76	80.35	80.39
45	95.55	95.56	95.41	82.33	82.66	82.42
50	95.61	95.64	95.68	83.74	84.69	84.05

表 2 中给出传输减少率的实验结果，其中：(1) 释放操作相关缓存方法的传输减少率普遍大于 Adaptive & Lazy 缓存 3%以上. (2) 指数缓存明显低于另外两种方法. 例如，在 campus 日志中缓存空间为 5%、10%时，指数缓存传输减少率低于另两种方法 20%、40%以上.

图 2 的字节利用率结果表示：(1) 释放操作相关缓存方法的结果十分稳定地处于高水平，而指数缓存的结果波动变化. (2) Adaptive & Lazy 缓存方法的结果明显小于另外两种方法. 例如，在 hpc 日志的 5%、10%体积比例下，释放操作相关缓存方法的字节利用率分别为 Adaptive & Lazy 缓存结果的 3.94 倍和 4.33 倍.

图 3 请求命中次数展现出指数缓存最高、释放操作相关缓存其次、Adaptive & Lazy 缓存最低的规律. 不同缓存方法请求命中次数随着缓存空间增大而呈现逐渐接近的现象是图 3 另一个特点.

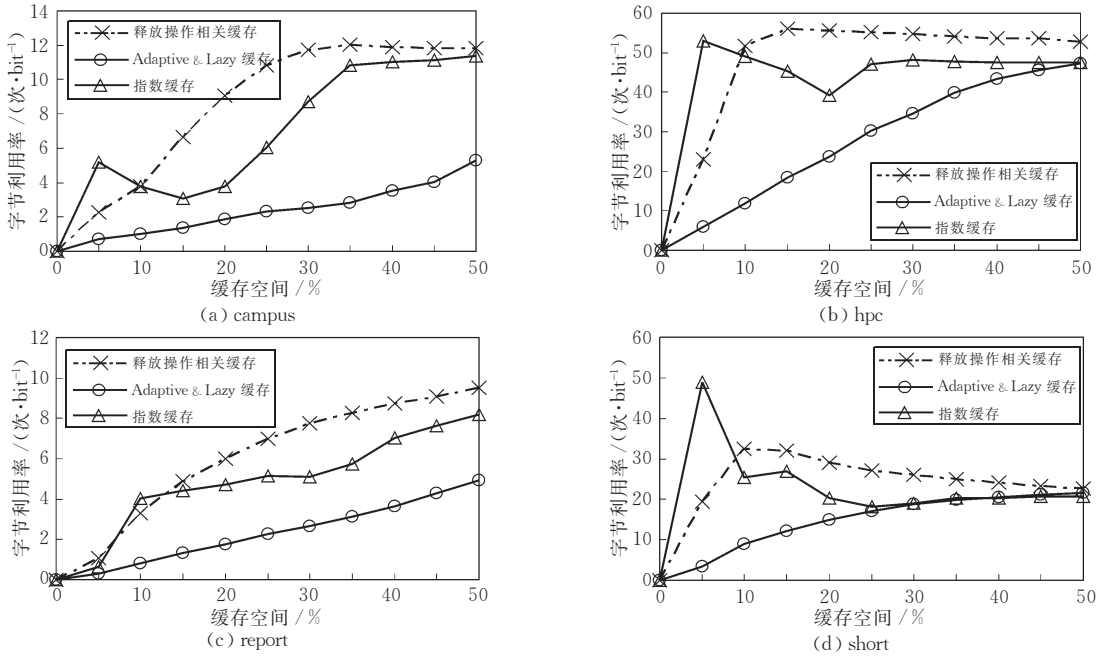


图 2　字节利用率

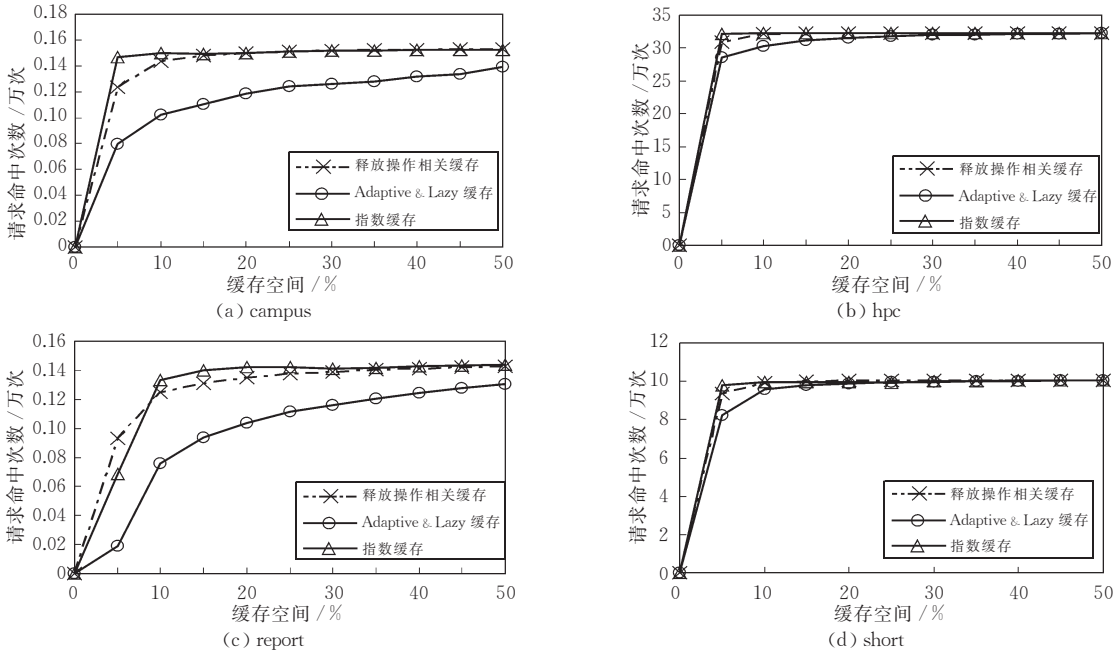


图 3　请求命中次数

实验结论和分析：

上面的实验结果可总结为以下 3 个结论.(1)释放操作相关缓存方法在网络传输节省和缓存写入效率两方面都取得了最好的效果;(2)指数缓存在网络传输减少方面远低于其它缓存方式;(3) Adaptive & Lazy 缓存在缓存写入效率方面远低于其他缓存方式.

表 3 hpc 日志的缓存写入量

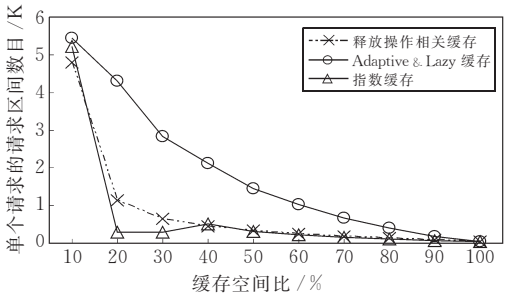
空间比例/%	缓存写入量/Byte		
	释放操作相关缓存	Adaptive & Lazy 缓存	指数缓存
5	79937606500	2.97503E+11	2916876250
10	40732483000	1.70547E+11	20506240250
15	37774754000	1.14755E+11	34417710500
20	38640551750	89894149500	54161216250
25	39317777750	70970038750	46427252250
30	39665432000	62360016250	45418728250
35	40290836500	54555745500	45857200250
40	40606918500	50196470250	45941872250
45	40531551000	47835116250	45970976250
50	41160256250	45979621500	45980007750

结论 2 的产生原因在文献[7-9]中已经进行了说明,主要是和指数缓存独特的头部内容预留空间方式密切相关.头部内容预留空间的方式使得可缓存内容的流媒体对象最大数目受到限制,尤其是在小比例空间状态下明显阻碍缓存内容的替换更新.文献[7-8]的实验结果也证明小比例空间状态下指数缓存保存流媒体对象数目较低的现象存在.另外,表 3 的结果表明在小体积空间比例状态下指数缓存的缓存写入量非常低,在说明头部预留空间阻碍缓存内容写入的同时也解释了指数缓存字节利用率初期特殊变化的现象.

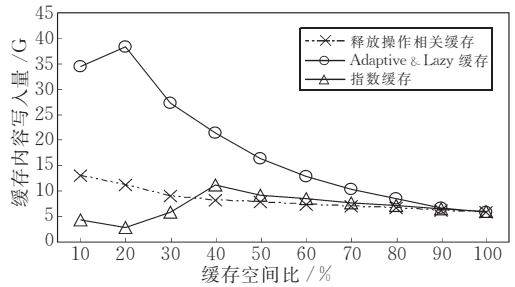
Adaptive & Lazy 缓存字节利用率极低(结论 3)的原因为平均访问量使得 Adaptive & Lazy 缓存分段体积过大.日志研究[21]发现流媒体访问存在小比例请求在刚访问不久就停止而大比例请求完整浏览流媒体对象内容的现象(例如在 hpc 日志中分别为 30%和 70%).这种现象会造成平均访问量的大小过大,也就导致 Adaptive & Lazy 缓存的分段体积过大,从而明显增加缓存写入量并显著影响字节利用率的结果：

(1)表 3 给出 hpc 日志模拟过程中的缓存写入量.可发现 Adaptive & Lazy 缓存的写入量明显大于另外两种方法,甚至完全抵消了在传输减少率方面对于指数缓存的优势.

(2)图 4 给出 report 日志测试中单个请求的请求区间数目变化曲线和缓存写入总量的变化曲线.从两个曲线变化基本一致(10%以后)说明缓存写入量的增多会明显增加头部内容的释放几率,并导致单个请求的增多,也就产生更多的无效缓存写入,进而进一步降低字节利用率.



(a) 单个请求的请求区间数目



(b) 缓存内容写入量

图 4 report 日志的缓存写入量与单个请求的关联

根据文献[21-22]和其他日志分析的归纳,热点内容的访问频率具有初期密集、随后下降、最后持续零星分布的变化规律.根据这种变化规律,热点流媒体对象的访问可被划分为热度集中、热度下降期、热度冷却期 3 个时期,而非热点的访问情况此处可被简单看作等同于热度冷却时期.利用连续请求数目和连续释放次数来控制缓存写入和缓存释放的措施有效保证在不同访问时期内释放操作相关缓存方法都能最大化发现访问热度,并用来提高缓存写入效率,也就获得最好的效果(结论 1)：

(1)热度集中期.释放几率低和连续请求多是这个时期的主要特性.释放操作相关缓存方法中,替换选取考虑连续请求数目的方式在增大连续请求多的缓存内容存放几率的同时,也使得偶然释放操作发生而损失缓存内容的情况被减少,也就保证了网络传输减少效果和限制了重复的缓存写入操作.

(2)热度下降期.释放几率逐渐增高和连续请求数目逐渐减少是该时期的特性.在该时期,缓存写入考虑遗留访问热度的进入策略使得释放操作相关缓存方法能对访问热度的持续性进行最大化的服务挖掘.另外,连续释放次数的增多、剩余缓存内容的减少以及连续请求数目的减少都使得连续请求引发

的缓存写入量和访问热度同样在逐渐减少，也就降低了低服务效率的缓存写入操作的发生.

(3)热度冷却期.此时期访问热度已在最低点，每次释放过程都是完全释放而连续请求数目始终较低.连续请求少和完整释放决定了此时期缓存写入量始终处于最小值，进而与网络传输减少效果最小保持一致.

三个访问时期的分析都说明释放操作相关缓存方法中缓存内容写入与访问热度的状况具有直接的关联，这种直接的关联在保证网络传输减少效果的同时也确定了缓存写入高效的特性.

对于请求命中次数结果的解释如下：流媒体对象头部内容的缓存状况影响着请求命中发生几率，每一时刻缓存的头部内容数量可体现命中几率的大小.头部内容写入次数和头部内容存放时间是命中次数的两个决定因素.由于测试输入相同，对不同缓存方法来说头部内容写入次数一致.由于缓存写入量对头部内容存放时间有负面的效果，参照图 4 和表 3 的缓存写入量关系可以解释指数缓存最高、释放操作相关缓存其次、Adaptive & Lazy 缓存最低的请求命中次数结果.

6　总结和未来的工作

与其它的流媒体缓存方法不同，释放操作相关缓存方法使用连续请求数目来反映访问热度，并用来决定每次缓存内容写入量和影响替换选取结果.实验测试说明，这种设计方式相比其他缓存方法在保证网络传输减少的同时，缓存写入效率提升明显.缓存写入效率的提升说明释放操作相关缓存方法拥有更强的实际应用能力.

将释放操作相关缓存方法与实际工作模式进一步结合是下一步工作的主要内容.流媒体缓存存在实际系统搭建困难的问题，更贴近实际的模拟环境将有力促进它的研究.LittleDuck 目前在模块化、分布式协同工作等方面还不完善.在随后工作中我们将通过对 LittleDuck 的改进，来完成缓存方法结合分布式结构、缓存方法结合内存-磁盘模式、缓存方法考虑负载空闲时段等方面的研究.

参　考　文　献

[1] Rejaie R, Handley M, Yu H, Estrin D. Proxy caching mechanism for multimedia playback streams in the internet// Proceedings of the 4th International Web Caching Workshop (WCW'99). San Diego, CA, 1999

[2] Acharya S, Smith B C. Middleman: A video caching proxy server//Proceedings of the 10th ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video(NOSSDAV'00). Chapel Hill, North Carolina, 2000

[3] Bommaiah E, Guo K, Hofmann M, Paul S. Design and implementation of a caching system for streaming media over the internet. IEEE Real Time Technology and Applications Symposium, May 2000, 111-121

[4] Ma W H, Du H C. Reducing bandwidth requirement for delivering video over wide area networks with proxy server. International Conferences on Multimedia and Expo., 2000, 4(4): 539-550

[5] Rejaie R, Yu H, Handely M, Estrin D. Multimedia proxy caching mechanism for quality adaptive streaming applications in the internet//Proceedings of the IEEE INFOCOM '00. Tel-Aviv, Israel, 2000: 980-989

[6] Wu K, Yu P S, Wolf J L. Segment-based proxy caching of multimedia streams//Proceedings of the 10th International World Wide Web Conference (WWW'2001). Hong Kong, 2001: 36-44

[7] Chen S, Shen B, Wee S, Zhang X. Adaptive and lazy segmentation based proxy caching for streaming media delivery//Proceedings of the 13th ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'03). Monterey, CA, 2003: 22-31

[8] Chen S, Wang H, Zhang X, Shen B, Wee S. Segment-based proxy caching for Internet streaming media delivery. IEEE Multimedia, 2005, 12(3): 59-67

[9] Liu J, Xu J. Proxy caching for media streaming over the internet. IEEE Communications Magazine, 2004, 42(8): 88-94

[10] Chesire M, Wolman A, Voelker G, Levy H. Measurement and analysis of a streaming media workload//Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems (USITS 2001). San Francisco, CA, 2001: 1-12

[11] Cherkasova L, Gupta M. Characterizing locality, evolution, and life span of accesses in enterprise media server workloads//Proceedings of the NOSSDAV'02. Miami, FL, 2002: 33-42

[12] Tewari R, Vin H, Dan A, Sitaram D. Resource-based caching for web servers//Proceedings of the ACM/SPIE Multimedia Computing and Networking 1998 (MMCN'98). San Jose, 1998: 191-204

[13] Sen S, Rexford K, Towsley D. Proxy prefix caching for multimedia streams//Proceedings of the INFOCOM'99. New York, 1999, 3: 1310-1319

[14] Zhang Z L, Wang Y, Du D H C, Su D. Video staging: A proxy-server based approach to end-to-end video delivery over wide-area networks. IEEE Transactions on Networking, 2000, 8(4): 429-442

[15] Fahmi H, Latif M, Sedigh-Ali S, Ghafoor A, Liu P, Hsu L. Proxy servers for scalable interactive video support. IEEE Computer, 2001, 43(9): 54-60

[16] Miao Z, Ortega A. Scalable proxy caching of video under storage constraints. IEEE Journal on Selected Areas in Communications, 2002, 20(7): 1315-1327

[17] Chae Y, Guo K, Buddhikot M, Suri S, Zegura E. Silo, rainbow, and caching Token: Schemes for scalable fault tolerant stream caching. IEEE Journal on Selected Areas in Communications, Special Issue on Internet Proxy Services, 2002, 20: 1328-1344

[18] Reisslein M, Hartanto F, Ross K W. Interactive video Streaming with Proxy Servers. Information Sciences, 2002, 140(1-2): 3-31

[19] Almeida J M, Eager D L, Vernon M K. A hybrid caching strategy for streaming media files//Proceedings of Multimedia Computing and Networking (MMCN'01). San Jose, CA, 2001: 200-212

[20] Ma Jie, Fan Jian-Ping. LittleDuck: Streaming cache simulator. Computer Engineering, 2006, 32(14): 208-210(in Chinese)
(马杰,樊建平. LittleDuck 流媒体缓存模拟器. 计算机工程, 2006, 2(14): 208-210)

[21] Tang Wen-Ting, Fu Yun, Cherkasova Ludmila, Vahdat Amin. Medisyn: A synthetic streaming media service workload generator//Proceedings of NOSSDAV'03. Monterey, CA, 2003: 12-21

[22] Jin S, Bestavros A. GISMO: A generator of internet streaming media objects and workloads. ACM SIGMETRICS Performance Evaluation Review, 2001, 29(3): 2-10

MA Jie, born in 1976, Ph. D.. His research interests include streaming media, streaming server, network security, cryptography, information system.

FAN Jian-Ping, born in 1963, Ph. D., professor. His research interests include computer architecture, system software, parallel processing.

Background

This research work is based on the first author's doctor thesis which aimed at enhancing the usability of the streaming media proxy cache.

In many research, it has been proved that streaming media proxy cache can improve the quality of the streaming media application such as VOD. Reducing the network traffic, reducing the server workload, reducing the network delay and improving the usability are the four function of the streaming media proxy cache.

Disk I/O created by the read action is the primary workload of the media server. In the cache proxy server, the write action is needed to store the popular contents. The write action of the disk has more delay and is more exclusive than the read action, so that the proxy server of the streaming media must consider the influence of the disk write action.

The cache method of stream proxy server is the origin of the disk write action. The existed research on the streaming media cache have mentioned several approaches such as reducing traffic, reducing network delay, supporting layer media encode, supporting the user operation of dragging. As us knew, no research concerned about the write action of cache.

In this paper a new streaming media cache method is bought forward which has good effect on both improving cache-writing efficiency and reducing network traffic. Based on this method, the service of proxy cache server for user can be more stable than that based on other methods which ignore the cache writing workload.

How to simulating the streaming media cache system and how to evaluating the user accesses are the two handicaps for the research of streaming media cache. The authors' research is begun with a large amount of the evaluating tests through MiddleSim, which is the simulator of a cooperating cache system research named MiddleMan. After the analysis of those testing result, the authors design the new cache method and developed LittleDuck .

The paper is major in the cache method and not cover the details of some other research approach such as multicast, batch, patching, scheduling algorithm, prefetch.