一种有资源适应性的 transcoding代理缓存机制

李春洪,冯国富,李文中,顾铁成,陆桑璐,陈道蓄
(南京大学计算机软件新技术国家重点实验室,江苏南京 210093)

摘 要: 在基于 transcoding代理的流媒体服务系统中,CPU和网络是两种潜在的瓶颈资源.本文提出了一种有资源适应性的 transcoding代理缓存机制,统一考虑 CPU和网络的资源需求,以提高系统的服务能力.首先推导了多版本缓存策略下网络收益和 CPU收益的计算方法.通过引入一个时变的影响因子 α(t),给出了缓存系统聚合资源收益的表达.在此基础上给出了单个对象的缓存价值函数,并设计了 RAC替换算法.实验表明 RAC具有较好的资源适应性和系统吞吐率.

关键词: 编码转换; 代理缓存; 缓存替换算法; 价值函数
中图分类号: TP393 **文献标识码**: A **文章编号**: 0372-2112 (2006) 08-1526-04

A Resource-Adaptive Transcoding Proxy Caching Mechanism

LI Chun-hong, FENG Guo-fu, LI Wen-zhong, GU Tie-cheng, LU Sang-lu, CHEN Dao-xu
(State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu 210093, China)

Abstract: In the transcoding proxy based streaming media systems, CPU and network are potential bottleneck resources. In this paper, a resource-adaptive transcoding proxy caching mechanism is proposed, which deals with network and CPU demand in an integrated fashion and aims to improve the system's capability potentially. We first explore the network gain and CPU gain of caching multiple versions at the same time. By introducing a time-vary influence factor α(t), the aggregated resource gain of the caching system is derived. Then, we derive the merit function of caching a single object under a given caching status, and design the RAC cache replacement algorithm. The experimental result shows that RAC can achieve good resource-awareness and improved system throughput.

Key words: transcoding; proxy caching; cache replacement algorithm; merit function

1 引言

在普适计算时代,越来越多的移动设备具备了 Internet访问能力,它们在网络、CPU、存储、屏幕大小等资源上有很大差异[1].在这样的异构环境中,单一服务难以满足所有用户的需求.利用编码转换(transcoding)技术对多媒体内容实施适应性裁剪,是实现普适多媒体服务访问的关键技术[2].为减少源服务器负载,通常在代理服务器部署编码转换服务[3,4].编码转换是一种计算密集型操作,transcoding代理的 CPU能力直接影响接入用户的数量.因此在普适流媒体系统中,源服务器至代理间的网络链路和代理服务器的 CPU能力是两个潜在的瓶颈[5].代理缓存是消除网络瓶颈,提高传统流媒体服务能力的有效途径[6,7].大部分已有的 transcoding代理缓存系统以最小化用户访问延迟为目标[8~10],而利用缓存来提高普适流媒体系统服务能力还缺乏很好的研究.本文提出了一种有资源适应性的缓存算法 RAC(Resource-Adaptive Caching).RAC根据当前 CPU和网络资源的紧缺程度,动态地调整缓存价值函数,表现出较强的资源适应性,提高了系统的服务能力.

文[5]和本文具有相同的研究背景,提出了三种缓存算法: FVO(Full Version Only), TVO(Transcoded Version Only)和自适应缓存算法.当网络资源充足,CPU资源受限时,FVO具有较高的请求接纳率;当网络资源受限,CPU资源相对紧缺时,TVO算法具有较好的表现.自适应缓存算法根据负载情况,在 TVO和 FVO间做出选择:如果网络阻塞的请求比例大于阈值 θ,采用 FVO算法;否则采用 TVO算法.θ是一个人工设定的阈值,但作者没有对动态计算环境中 θ的选取作进一步探讨.此外,文[5]假定低版本只能从原始版本转换获得,虽然简化了缓存决策,但提出的缓存价值函数缺乏普适性.

2 问题描述

2.1 Transcoding模型

原始版本具有完整语义信息,可以被转换成任何低版

收稿日期: 2005-08-02;修回日期: 2006-05-02
基金项目:国家 973重点基础研究发展规划(No. 2002CB312002);国家自然科学基金(No. 60402027, 60573106);国家 863高技术研究发展计划(No. 2004AA112090)

本.而低版本也可进一步转换成更低的版本,且 CPU 开销往往小于直接从原始版本转换的开销.由于 transcoding是一种有损操作,版本间的转换关系通常是不可逆的.版本转换关系可用加权编码转换图 WTG (Weighted Transcoding Graph)表示[10],如图 1在 WTG图 G_i 中,顶点 $v \in V[G_i]$代表对象的一个版本,有向边 $(u, v) \in E(G_i)$表明版本 u可以转换到 v,权值 $w_i(u, v)$代表转换开销.在 G_i 中,如果存在一条有向边 $(u, v) \in E[G_i]$,则称 u为 v的可用版本.版本 v的所有可用版本的集合,称为 v的可用版本集,用符号 $\Phi_i(v)$表示.例如在图 1中 $\Phi_i(4) = \{1, 2, 4\}$.

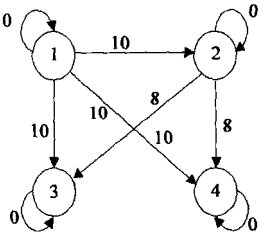

图 1　WTG图示例[10]

2.2　服务模型

一个普适流媒体系统由源服务器、transcoding代理和用户组成.源服务器存放了全部对象的原始版本. Transcoding代理对媒体流执行编码转换,并提供缓存服务. Transcoding代理接受用户的请求,确定满足需求的版本,并搜索缓存空间.如果缓存中存在用户请求的版本(称为版本命中),用户的请求被接纳.如果缓存中没有用户请求的版本,但存在可用版本(称为可用命中),则选择转换开销最小的可用版本,并预留所需的 CPU资源.如果预留成功,请求被接纳,否则请求被阻塞(称为 CPU阻塞).如果缓存失效,则向源服务器请求对象的原始版本. Transcoding代理确定所需的传输带宽及原始版本转换到所需版本的 CPU资源,并进行资源预留.如果预留成功,请求被接纳,否则请求被阻塞(因网络资源不足导致的阻塞称为网络阻塞;因 CPU资源不足导致的阻塞称为 CPU阻塞).

2.3　问题的形式化

我们首先给出如表 1所示的符号说明.

表 1　符号说明

符号	描述
B	从内容服务器到 transcoding代理之间瓶颈链路的带宽
C	代理服务器的 CPU能力
$O = \{o_1, \cdots, o_N\}$	系统中所有媒体对象的集合,其中 N为媒体对象的总数
$o_{i,x}$	o_i的版本 x
b_i	o_i的原始版本的编码率
$\lambda_{i,x}$	$o_{i,x}$的平均访问频率
$s_{i,x}$	$o_{i,x}$的大小
G_i	对象 i的编码转换图
w_i	对应于 G_i的权值矩阵,描述版本间转换的 CPU开销
S_{cache}	缓存空间的大小
H(O)	某时刻缓存中的对象集合

2.3.1　网络收益

设当前缓存状态为 H(O),用户请求 $o_{i,x}$.如果缓存中存在 $o_{i,x}$或可用版本,则不必从源服务器请求 o_i的原始版本.因此节省的网络带宽为 b_i;如果缓存失效,网络收益为 0.因此在缓存状态 H(O)下,由于用户请求 $o_{i,x}$,系统节省的网络资源为:

$$g^{net}(o_{i,x} \mid H(O)) = \begin{cases} 0, & \text{if } H(O) \cap \Phi_i(x) = \varnothing \\ b_i, & \text{else} \end{cases}$$

缓存的网络收益为:

$$G^{net}(H(O)) = \sum_{i=1}^{N} \sum_{x \in V[G_i]} \lambda_{i,x} \cdot g^{net}(o_{i,x} \mid H(O)) \quad (1)$$

2.3.2　CPU收益

如果缓存中存在 $o_{i,x}$,则不需要任何 transcoding处理,就可直接满足用户的请求.相比于从源服务器请求完整版本并转换为 $o_{i,x}$,节省的 CPU资源为 $w_i(1, x)$;如果存在 $o_{i,x}$的可转换版本,可以由这些可转换版本经 transcoding获得.系统选择具有最小转换开销的可转换版本 y进行转换,耗费的 CPU资源为 $w_i(y, x)$.因此,节省的 CPU资源为 $(w_i(1, x) - w_i(y, x))$.在缓存状态 H(O)下,由于用户请求 $o_{i,x}$节省的 CPU资源为:

$$g^{cpu}(o_{i,x} \mid H(O)) = \begin{cases} 0, & \text{if } H(O) \cap \Phi_i(x) = \varnothing \\ w_i(1, x) - \min\limits_{y \in H(o_i) \cap \Phi_i(x)} w_i(y, x), & \text{else} \end{cases}$$

缓存的 CPU收益为:

$$G^{cpu}(H(O)) = \sum_{i=1}^{N} \sum_{x \in V[G_i]} \lambda_{i,x} \cdot g^{cpu}(o_{i,x} \mid H(O)) \quad (2)$$

2.3.3　聚合资源收益

由于 CPU和网络是两种不同类型的资源,我们用各资源收益与该资源的最大可用量的比值来进行统一表示.式(1)和式(2)分别改写如下:

$$G^{net}(H(O)) = \frac{1}{B} \cdot \sum_{i=1}^{N} \sum_{x \in V[G_i]} \lambda_{i,x} \cdot g^{net}(o_{i,x} \mid H(O)) \quad (3)$$

$$G^{cpu}(H(O)) = \frac{1}{C} \cdot \sum_{i=1}^{N} \sum_{x \in V[G_i]} g^{cpu}(o_{i,x} \mid H(O)) \quad (4)$$

缓存的聚合资源收益统一描述为:

$$G(H(O), \alpha(t)) = \alpha(t) \cdot G^{net}(H(O)) + (1 - \alpha(t)) \cdot G^{cpu}(H(O))$$

其中,$\alpha(t)$ $(0 \le \alpha(t) \le 1)$是一个时变的因子,反映网络收益的相对重要性. $\alpha(t)$越接近 1,聚合资源收益中网络资源收益的影响越大;$\alpha(t)$越接近 0, CPU资源收益的影响越大.

由于缓存空间受限,应确定缓存哪些对象的哪些版本,使得缓存的聚合收益最大化.因此缓存问题是一个最优化问题,即确定各对象的各版本在缓存中的分布 H(O),使 $G(H(O), \alpha(t))$最大化,并满足约束条件: $\sum_{\forall o_{m,n} \in H(O)} S_{m,n} \le S_{cache}$.

3　缓存替换算法

本文设计了一种启发式的缓存算法 RAC (Resource A-

daptive Caching algorithm),基本思想是在缓存中尽可能保留对 $G(H(O),\alpha(t))$贡献较大的对象,使 $H(O)$逐渐趋于最优.

3.1 价值函数

在缓存状态 $H(O)$下,如果将一个新对象 $o_{i,x}$($o_{i,x}\notin H(O)$)加入缓存,那么缓存收益增量为 $G(H(O)+\{o_{i,x}\},\alpha(t))-G(H(O),\alpha(t))$.该缓存收益增量即为在缓存状态 $H(O)$下,$o_{i,x}$的缓存价值.考虑对象大小的影响,$o_{i,x}$的缓存价值定义为:

$$U(o_{i,x}\mid(H(O),\alpha(t)))=\frac{G(H(O)+\{o_{i,x}\},\alpha(t))-G(H(O),\alpha(t))}{s_{i,x}}$$

3.2 $\alpha(t)$的计算

在 Internet环境中,$\alpha(t)$值应根据资源的可用情况动态求得.我们设计了一种基于观测的自适应 α控制机制.如图 2所示,α参数控制器、缓存系统和观测器,形成了控制回路.观测器测量当前时刻的 CPU 阻塞率 $p_{cpu}(t)$和网络阻塞率 $p_{net}(t)$,反馈给 α参数控制器.α参数控制器计算新的 α值:$\alpha(t)=\frac{p_{net}(t)}{p_{cpu}(t)+p_{net}(t)}$.实现中,$p_{cpu}(t)$和 $p_{net}(t)$是通过观测 $(t-\Delta t,t)$时间段内 CPU 阻塞的请求和网络阻塞的请求所占分量来估计的.

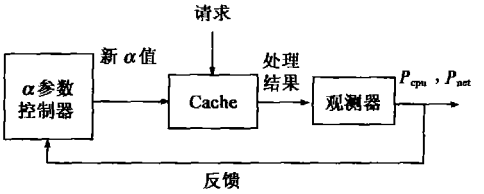

图 2 α的控制回路

3.3 算法描述

RAC替换算法的执行过程见算法 1:将 $o_{i,x}$加入缓存队列;计算队列中各对象的缓存价值,并将具有最小缓存价值的对象从缓存队列中移除;如果缓存队列中所有对象的大小总和大于缓存空间的大小,则重新计算对象缓存价值,并从队列移除缓存价值最小的对象;重复以上过程,直至缓存队列中所有对象的大小总和不大于缓存空间的大小.算法开销主要集中在对象缓存价值的排序上,故复杂度为 O(NlogN),其中 N为缓存中的对象数量.

算法 1RAC替换算法

```
Algorithm RAC(o_{i,x}, H(O), α(t))
1  H(O) ← H(O) + {o_{i,x}}
2  while S(H(O)) > S_cache
3     do for each object o_{m,n} in H(O)
4        do calculate caching merit
           U(o_{m,n} | (H(O) - {o_{m,n}}, α(t)))
5        find the object o_{m,n} with the lowest caching value
6        H(O) ← H(O) - {o_{m,n}}
```

4 性能评价

4.1 仿真模型

源服务器存放了 1000个对象,长度符合 $\mu=1800s$,$\sigma=180s$的正态分布,流行度满足 $\theta=0.75$的 Zipf分布.请求序列用 $\lambda=0.2$的 Poisson模型产生.用户被分为 5类,分布为:0.15,0.20,0.30,0.2和 0.15,对应的版本大小分别为原始版本的 100%,80%,60%,40%和 25%.各对象有相同的 WTG图,权值矩阵为:

$$W=\begin{bmatrix}0.0 & 0.01 & 0.01 & 0.01 & 0.01\\ \infty & 0.0 & 0.008 & 0.008 & 0.008\\ \infty & \infty & 0.0 & 0.006 & 0.006\\ \infty & \infty & \infty & 0.0 & \infty\\ \infty & \infty & \infty & \infty & 0.0\end{bmatrix}$$

w 的元素代表版本间转换的 CPU 开销占标准服务器 CPU 能力的比例.

令所有对象的原始版本的总大小为 $S(S=\sum_{i=1}^{1000}s_{i,x})$,相对缓存大小描述了缓存空间与 S的比值.设在无缓存情况下,系统接纳全部用户请求所需的 CPU 和带宽分别为 C^{Max}和 B^{Max}.CPU 系数描述代理的 CPU 能力和 C^{Max}的比值,网络系数描述实际带宽和 B^{Max}的比值.我们分析不同参数下 RAC、LRU 和 LFU 算法的性能.

4.2 结果分析

4.2.1 缓存大小的影响

CPU 系数和网络系数设定为 0.3,不同缓存大小下的实验结果如图 3所示.随着缓存空间增加,各算法的总体阻塞率都呈下降趋势,而 RAC的总体阻塞率均小于 LRU 和 LFU.由图 3(b),RAC的版本命中率远高于 LRU 和 LFU.说明在 RAC中更多的请求直接在缓存中得到满足,而不必消耗额外的网络和 CPU 资源.因此 RAC节省了更

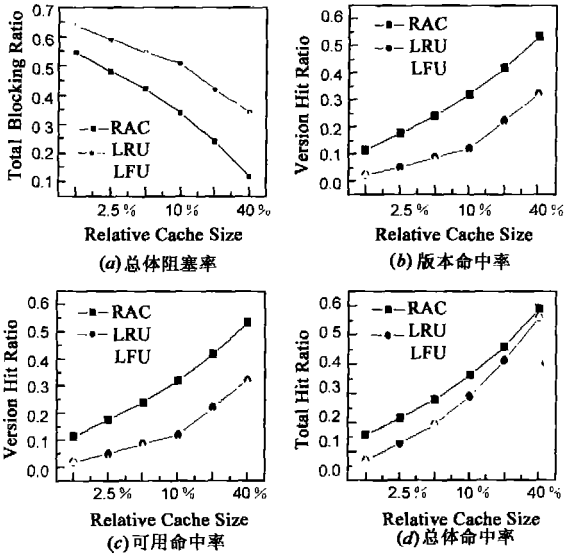

图 3 不同缓存大小下的性能

多的瓶颈资源.这是 RAC具有较高接纳率的主要原因.观察图 3(c)和 3(d),我们发现 LRU 和 LFU 的总体命中率低于 RAC,且其中可用命中率占很大的比例.说明在 LRU 和 LFU 中,用户请求消耗的 CPU 资源较多,CPU 是主要的瓶颈资源.

4.2.2　算法的资源调节能力

我们对系统运行中阻塞率的变化情况进行了分析.图 4(a)～(c)为在 CPU 和网络系数分别为 0.3,相对缓存为 40%时,各算法的阻塞率随时间变化的情况.图 4(a)中,0～t1阶段,CPU 阻塞率急剧上升,网络阻塞率呈明显的下降趋势.这是因为该阶段请求版本命中的概率较小,大量请求因 CPU 资源不足被阻塞,同时造成网络负载的下降.在 t1～t2,版本命中率逐渐提高,CPU 负载得以缓解,网络负载相应提高.t2 时刻系统达到稳定.反观图 4(b)和 4(c),t1 时刻后系统即趋于稳定,CPU 阻塞率维持在较高水平,网络阻塞率却趋于 0.可见相比于 LRU 和 LFU,RAC 具有较强的资源调节能力.

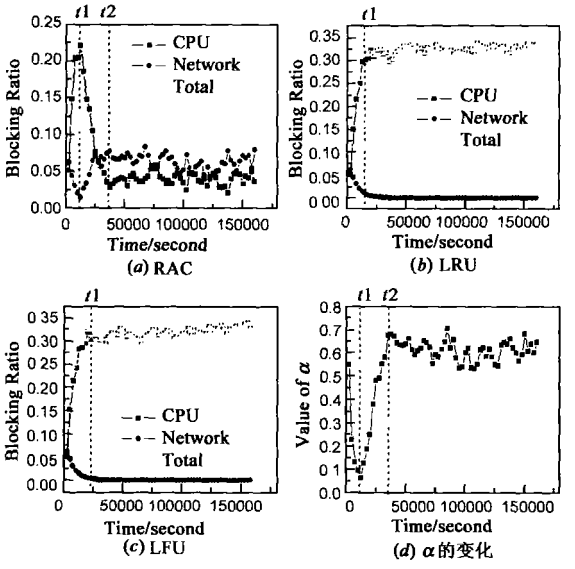

图 4　资源调节能力比较

5　结论

在基于 transcoding代理的流媒体服务系统中,CPU 和网络是两类潜在的瓶颈资源.本文以提高系统的总体服务能力为目标,提出了一种具有资源适应性的代理缓存算法 RAC.RAC可根据当前的负载情况,适时地调整缓存价值函数,使得对缓解相对紧缺的资源贡献大的对象有更高的被缓存概率.实验表明 RAC表现出较强资源的适应性,可有效地提高系统吞吐率.

参考文献:

[1] M Satyanarayanan. Pervasive computing: vision and challenges[J]. IEEE Personal Communications, 2001, 8(4): 10 - 17.

[2] W Y Lum, CM Lau. A context-aware decision engine for content adaptation[J]. IEEE Pervasive Computing, 2002, 1(3): 41 - 49.

[3] S Chandra, et al. Application-level differentiated multimedia web services using quality aware transcoding[J]. IEEE Journal on Selected Areas in Communications, 2000, 18(12): 2544 - 2565.

[4] N Björk, C Christopoulos. Video transcoding for universal multimedia access[A]. Proceedings on ACM Multimedia 2000 Workshops[C]. New York: ACM Press, 2000. 75 - 79.

[5] X Tang, et al. Streaming media caching algorithms for transcoding proxies[A]. Proceedings of the 31st International Conference on Parallel Processing[C]. Vancouver: IEEE Computer Society, 2002. 287 - 295.

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

[7] Wang J. A survey of web caching schemes for the internet[J]. Acm Sigcomm Computer Communication Review, 1999, 29(5): 36 - 46.

[8] B Shen, et al. Caching strategies in transcoding-enabled proxy systems for streaming media distribution networks[J]. IEEE Transactions on Multimedia, 2004, 6(2): 375 - 386.

[9] A Trivedi, et al. PTC: proxies that transcode and cache in heterogeneous web client environments[J]. World Wide Web, 2004, 7(1): 7 - 28.

[10] C Chang, M Chen. On exploring aggregate effect for efficient cache replacement in transcoding proxies[J]. IEEE Transactions on Parallel and Distributed Systems, 2003, 14(6): 611 - 624.

作者简介:

李春洪　男,1972年 4月出生于江苏丹阳.现为南京大学计算机科学与技术系软件与理论专业博士研究生.主要研究方向为分布式计算与并行处理.E-mail: lch@dislab.nju.edu.cn

冯国富　男,1977年 3月出生于山东沂水.现为南京大学计算机科学与技术系软件与理论专业博士研究生.主要研究方向为分布式计算与并行处理.E-mail: fgfmail@dislab.nju.edu.cn