ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.19, No.4, April 2008, pp.956–966
DOI: 10.3724/SP.J.1001.2008.00956

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

En-Route Transcoding缓存的优化放置和替换*

李春洪[1,2+], 冯国富[1,2], 顾铁成[1,2], 陆桑璐[1,2], 陈道蓄[1,2]

[1](南京大学 计算机软件新技术国家重点实验室,江苏 南京 210093)

[2](南京大学 计算机科学与技术系,江苏 南京 210093)

Optimal Placement and Replacement Scheme in En-Route Transcoding Caching Systems

LI Chun-Hong[1,2+], FENG Guo-Fu[1,2], GU Tie-Cheng[1,2], LU Sang-Lu[1,2], CHEN Dao-Xu[1,2]

[1](State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China)

[2](Department of Computer Science and Technology, Nanjing University, Nanjing 210093, China)

+ Corresponding author: Phn: +86-25-58821506, Fax: +86-25-83300710, E-mail: lch@dislab.nju.edu.cn, http://cs.nju.edu.cn

Li CH, Feng GF, Gu TC, Lu SL, Chen DX. Optimal placement and replacement scheme in en-route transcoding caching systems. *Journal of Software*, 2008,19(4):956−966. http://www.jos.org.cn/1000-9825/19/956.htm

Abstract: This paper investigates cache routing and cache management problems for en-route transcoding caching systems. An active cache routing algorithm called CCRA (cost-aware cache routing algorithm) is designed, which, using a controllable probing load, can find the potential cache objects with minimal access cost. Then an analyzable model for en-route transcoding caching is established, with which the cooperative cache placement and replacement problem is formulated as an optimization problem and the optimal locations to cache the object are obtained by using a dynamic programming algorithm. Results of the simulation show that the proposed scheme outperforms existing meta placement algorithms on metric of CSR (cost save ratio).

Key words: en-route transcoding caching; cache routing; cooperative placement; cache replacement

摘 要: 对 en-route transcoding 缓存中的缓存路由和协同放置及替换问题进行了研究.提出了 CCRA(cost-aware cache routing algorithm)缓存路由算法,能以可控的探测开销来发现潜在的、具有最小访问开销的缓存对象.在此基础上,建立了 en-route transcoding 缓存的分析模型,将缓存放置和替换问题形式化为一个最优化问题,并利用一种基于动态规划的方法来求解最佳缓存放置策略.仿真结果表明,与已有的元算法放置策略相比,该协同放置和替换策略可以获得更好的 CSR 性能.

关键词: en-route transcoding 缓存;缓存路由;协同放置;缓存替换

中图法分类号: TP393 **文献标识码**: A

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、屏幕大小等资源上存在很大差异[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.通过编码转换(transcoding)对内容进行

* Supported by the National Natural Science Foundation of China under Grant Nos.60573106, 60402027, 60573131 (国家自然科学基金); the National Basic Research Program of China under Grant No.2002CB312002 (国家重点基础研究发展计划(973)); the Natural Science Foundation of Jiangsu Province of China under Grant Nos.BK2005411, BK2005208 (江苏省自然科学基金)

Received 2006-09-27; Accepted 2006-12-27

适应性裁剪,是实现普适多媒体服务访问的关键技术[2,3].编码转换是一种计算密集型操作,transcoding代理服务器很容易成为系统的瓶颈.利用多个transcoding代理服务器协同地为用户提供服务,是提高系统服务能力和可扩展性的有效途径[4−7].代理服务器被组织成一定的结构(如层次式结构),协同地进行缓存定位和路由、编码转换和对象传输等.En-Route transcoding缓存[6,7]是一种新型的transcoding代理缓存体系结构,它对传统的en-route Web缓存[8]进行了扩展,使得缓存节点同时具备缓存和transcoding服务能力.用户请求沿网络层最短路径向源服务器传递,各en-route节点拦截请求报文,并在其缓存中查找满足服务需求的缓存对象.如果存在满足需求的缓存对象,则请求不再向源服务器传递,而由该en-route节点将对象(经过必要的编码转换)沿原路径发送给用户.此外,在由transcoding代理服务器组成的应用层内容分发网络中,各transcoding代理可以视为一个应用层路由器,因此,这种应用层内容分发网络也可视为一种应用层en-route transcoding缓存系统.

En-Route Web缓存的查找、放置和替换已得到较为深入的研究[8,9].但在transcoding缓存中,同一对象的多个版本分散在各缓存节点中,版本之间存在较为复杂的转换关系.传统路由、放置和替换算法难以直接而有效地应用于en-route transcoding缓存中.本文以最小化系统的总体服务开销为目标,提出了CCRA(cost-aware cache routing algorithm)缓存路由算法.在此基础上,建立en-route transcoding缓存系统的分析模型,给出缓存放置和替换的优化模型,并利用一种基于动态规划的方法来求解缓存对象的最佳放置位置.

1 相关工作

近年来出现的en-route Web缓存是一种新型的协同缓存体系结构,它在部分路由节点部署了缓存服务(称为en-route缓存节点),拦截途径的请求报文,并检查其缓存空间:如果缓存命中,则将该对象发送给用户,请求不再向上游节点传递;否则,请求沿正常路径向源服务器传递.对TCP或HTTP协议进行适当的扩展,en-route缓存可以方便地在现有的Internet环境中部署.当对象从服务节点(en-route缓存或者源服务器)向用户传输途中的各en-route缓存节点可选择是否在其缓存中保留对象的副本.Nikolaos等人[10]利用 4 种启发式元(meta)算法来确定对象放置的节点.Tang等人[8]建立了en-route Web缓存的分析模型,推导了各种缓存放置策略下的系统收益表达,给出了基于动态规划的求解方法.但以上方法无法直接而有效地运用于en-route transcoding缓存系统.

已有的协同transcoding缓存研究[4,5]主要关注分布式环境中的缓存查找定位问题,对于缓存协同放置和替换则没有很好地研究.Li等人[6,7]研究了en-route transcoding缓存系统中的优化放置问题,给出了分析模型和基于动态规划的求解方法.但是,他们假设一个对象在每个节点最多只存在一个版本,并采用lazy型缓存路由算法.Lazy型缓存路由算法是指请求向源服务器传递过程中,遇到第一个满足服务需求的缓存对象即停止缓存查找,忽略了可能存在于其他节点的、具有更小访问开销的缓存对象.本文提出的放置策略允许对象的多个版本同时存在于一个缓存中,且采用一种active型缓存路由算法CCRA来发现具有更小访问开销的缓存对象.

2 模型、定义和假设

2.1 编码转换模型

设源服务器中对象的原始版本具有完整语义信息,可以转换成任意低版本.而低版本也可以进一步转换成更低的版本.版本转换关系用加权编码转换图WTG(weighted transcoding graph)表示[11],其中,各顶点代表一个对象版本,有向边表明版本间存在转换关系,边的权值代表转换开销.

定义 1(可转换版本集). 在对象i的编码转换图TG_i中,如果存在一条有向边$(u,v)\in[TG_i]$,则称版本v为版本u的可转换版本.版本v所有可转换版本的集合,称为版本v的可转换版本集,用符号$\psi_i(v)$表示.

例如,在图 1 所示的WTG图中,$\psi_i(1)=(1,2,3,4)$,$\psi_i(2)=(2,3,4)$.

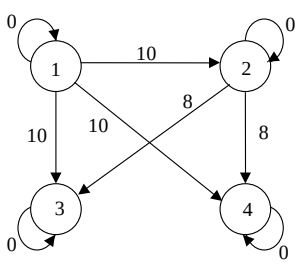

Fig.1 An example of WTG graph

图 1 WTG 图示例

Yamaoka等人[12]提出了一种编码转换开销的估算方法.假设对象

质量主要取决于图像大小($q.s$)、帧速($q.f$)和比特率($q.b$),并用向量$q=(q.s,q.f,q.b)$表示.将质量为q的对象转换为质量为q'的版本所需的计算力由两部分组成:一是解码质量为q的对象所需计算力;二是将其编码成质量为q'的版本所需计算力.文献[12]的实验结果表明,解码和编码的计算需求与所处理的像素数量成正比.那么,编码转换所需的计算能力表示为

$$r(q\to q')=r_{dec}(q)+r_{enc}(q')=\tau_d\cdot(q.s\cdot q.f)+\tau_e\cdot(q'.s\cdot q'.f) \quad (1)$$

其中,τ_d和τ_e为常量.

2.2 系统模型

基于en-route transcoding缓存的内容分发网络可以用一个图$G(V,E)$表示,其中,V为路由节点集合,E为网络链路集合.设节点集$V_C(V_C\subseteq V)$中的每个节点都具有transcoding缓存能力,称为en-route transcoding节点.节点集$V_S(V_S\subseteq V)$中,每个节点都与一个源内容服务器关联.设源内容服务器也具备编码转换服务能力.源内容服务器集维护了全部对象的完整版本,且各源服务器维护的对象集不相交.当用户请求对象o_{iC}时,请求报文沿着最短路径向对象o_i所在的源内容服务器S传递.对于那些目标同为S的请求来说,其传递路径形成了以S为根、各缓存节点为子孙节点的树形结构.沿途的缓存节点拦截用户请求,利用CCRA缓存路由算法来发现满足用户需求并具有最小服务开销的缓存对象.如果沿途所有的缓存节点中均不存在满足需求的缓存对象(即全局缓存失效),则请求最终发送至源服务器.如果需要进行版本转换,则由服务节点(en-route节点或源内容服务器)提供相应的编码转换服务.请求报文经过的每个en-route缓存节点搜集相关的参数,并附加在请求报文中.依据这些参数信息,服务节点计算出对象o_{iC}在回送路径中各en-route节点中的优化放置策略,沿途各en-route缓存节点据此来执行缓存的放置和替换操作.

3 CCRA 缓存路由算法

在基于en-route tarnscoding缓存的内容服务系统中,请求的访问开销由两部分组成:编码转换开销和网络传输开销.设A_n发起对o_{ij}的请求,其上游节点A_0中的缓存对象o_{iH}被命中为服务对象,则请求o_{ij}的服务开销$Cost(A_n,o_{ij},A_0,o_{iH})$计算如下:

$$Cost(A_n,o_{ij},A_0,o_{iH})=\alpha\cdot c(A_0,A_n,o_{ij})+(1-\alpha)\cdot w(o_{iH},o_{ij}) \quad (2)$$

其中,$c(A_0,A_n,o_{ij})$为将o_{ij}从A_0传输至A_n的网络开销,$w(o_{iH},o_{ij})$为$o_{iH}\to o_{ij}$的编码转换开销,α为影响因子,用于刻画两类开销的相对重要性.

对象o_i可能有多个不同版本分布在节点A_n至源服务器间的en-route节点中,其中满足服务需求的版本也可能不止一个.由于这些对象版本所处的位置不同,它们转换为被请求版本的开销也不同,因此访问开销也不同.文献[6,7]采用了较为简单的缓存路由算法:当请求经过第一个缓存命中的en-route节点时,即终止查找.我们称这种缓存路由算法为lazy算法.Lazy算法的优点是实现简单,具有较低的路由开销;缺点是忽略了可能存在于上游节点中的具有更低服务开销的缓存对象.图 2(a)为lazy算法示意图,假设节点中的缓存对象o_{iH_1}被命中为服务对象,则请求的访问开销为$Cost(A_n,o_{ij},A_0,o_{iH_1})$.而在图 2(b)所示的active缓存路由算法中,A_0的上游节点A_{-k}中的对象o_{iH_2}被选择为服务对象,则访问开销为$Cost(A_n,o_{ij},A_{-k},o_{iH_2})$.两种算法访问开销之差为

$$Cost(A_n,o_{ij},A_0,o_{iH_1})-Cost(A_n,o_{ij},A_{-k},o_{iH_2})=(1-\alpha)\cdot(w(o_{iH_1},o_{ij})-w(o_{iH_2},o_{ij}))-\alpha\cdot c(A_{-k},A_0,o_{ij}) \quad (3)$$

如果在A_{-k}存在具有更小转换开销的版本,且节省的转换开销大于额外的传输开销,式(3)大于 0,此时,active算法比lazy算法更有效.我们实现了一种称为CCRA的active缓存路由算法.为减小协议的控制开销,CCRA通过设置探测预算变量*Probe_budget*值来控制向上游节点探测的范围.CCRA算法的伪代码描述如下:

1. *Probe_budget*=∞;
2. *Current_node*=A_k;
3. *Lastest_feasible_node*=−1;
4. *Lastest_feasible_object*=*null*;
5. while (*Probe_budget*>0 and *Current_node*!=*Content_server*) do

```
6.      oiH=Lookup(oij,Current_node);
7.      if (oiH!=null)
8.          Probe_budget=((1−α)*w(oiH,oij))/α;
9.          Lastest_feasible_node=Current_node;
10.     endif
11.     Upper_node=Get_the_upperlevel_node(Current_node);
12.     Probe_budget−=α*c(Curret_node,Uppper_node,oij);
13.     Current_node=Upper_node;
14. endwhile
15. if (Current_node==Content_server)
16.     if (Lastest_feasible_node!=−1)
17.         return (Lastest_feasible_node,Lastest_feasible_object);
18.     else
19.         return (Content_server,oi0);
20.     endif
21. else
22.     return (Lastest_feasible_node,Lastest_feasible_object);
23. endif
```

*Probe_budget*值初始设置为∞(第 1 行);请求到达第一个命中节点时,*Probe_budget*被更新为$(1-\alpha)\cdot(w(o_{iH},o_{ij})/\alpha$($o_{iH}$为命中对象,第 8 行).从*Probe_budget*扣除上游节点至当前节点间传输对象o_{ij}的网络开销(第 12 行).如果*Probe_budget*大于 0,则探测继续;否则探测结束,返回命中节点和对象ID(第 16~20 行).

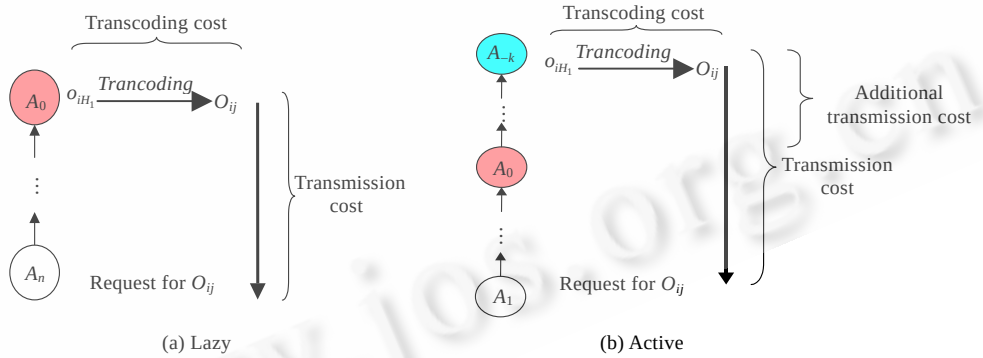

Fig.2 Lazy caching routing algorithm vs. active caching routing algorithm

图 2 Lazy 缓存路由算法与 active 缓存路由算法

4 缓存的优化放置与替换

设在A_n的用户请求o_{iC},根据CCRA算法,A_0中的缓存对象o_{iH}被选择为服务对象.A_0执行编码转换服务(如果$o_{iH}\neq o_{iC}$),并沿相反路径向A_n传送o_{iC}.在传输过程中,有选择地在部分节点上放置对象副本.之后,部分对o_{iC}的请求可在更接近用户的位置获得服务,节省了访问开销(称为访问开销节省量).但另一方面,由于缓存空间有限,节点必须逐出部分缓存对象以放置o_{iC},导致对被替换对象访问开销的增加(称为访问开销损失).因此,缓存放置和替换决策的关键在于计算各种放置策略下访问开销节省量和损失量,选择对降低访问开销贡献(即访问开销收益)最大的放置策略,使系统的总体服务开销达到最小.我们首先对版本间的转换关系作以下假设:

假设 1. 对象的低版本只能从它的完整版本转换获得.

由假设 1 可知,$\forall o_{ij}, \psi(o_{ij})=\{o_{ij}\}\cup\{o_{i0}\}$.

4.1 问题的形式化描述

首先讨论在中间节点v放置对象o_{iC}的访问开销节省量和损失量求解.在v放置o_{iC}后,不仅请求o_{iC}的请求可以降低访问开销,而且请求低版本$o_{ij}(o_{ij}\in\psi(o_{iC}))$的用户也可能获益.

定义 2(缺失损失). 如果缓存在节点v的对象o_{iC}被移除,则节点v因请求对象o_{ij}而增加的访问开销,称为对象o_{iC}在节点v因请求o_{ij}的缺失损失,并记为$m_v(o_{iC}|o_{ij})$.

分两种情况讨论$m_v(o_{iC}|o_{ij})$的求解.

情况 1. $o_{ij}\notin\psi(o_{iC})$,即o_{ij}不能由o_{iC}转换生成.显然,o_{iC}的移除与否,请求开销并无变化.因此,$m_v(o_{iC}|o_{ij})=0$.

情况 2. $o_{ij}\in\psi(o_{iC})$,即用户请求的对象可以从o_{iC}转换获得.设在v移除o_{iC}后,根据CCRA算法,v^+(v^+可能是v本身)中的缓存对象o_{iH}被选择为服务对象.因此,移除o_{iC}后,在v请求o_{ij}增加的访问开销为$\sigma=\alpha\cdot C(v,v^+,o_{ij})+(1-\alpha)\cdot(w(o_{iH},o_{ij})-w(o_{iC},o_{ij}))$.如果$\sigma<0$,说明虽然$o_{iC}$可以转换为$o_{ij}$来满足服务请求.但根据CCRA算法,$o_{iC}$不会被选为服务对象.因此,$o_{iC}$在$v$因请求$o_{ij}$的缺失损失为

$$m_v(o_{iC}\mid o_{ij})=\begin{cases}\sigma, & \sigma>0\\ 0, & \sigma\le 0\end{cases} \tag{4}$$

设在节点v,对象o_i各版本的访问频率分别为$f_v(o_{i0}),f_v(o_{i1}),\ldots,f_v(o_{i(m-1)})$,则节点$v$缓存对象$o_{iC}$,访问开销的节省量为$\sum_{o_{ij}\in\Psi(o_{iC})}(f_v(o_{ij})\cdot m_v(o_{iC}\mid o_{ij}))$.

如果缓存空间已满,v必须逐出部分缓存对象来放置对象o_{iC}.设当前节点v的缓存对象集合为H_v.从节点v逐出缓存对象$o_x(o_x\in H_v)$,访问开销损失为$\sum_{o_y\in\Psi(o_x)}(f_v(o_y)\cdot m_v(o_x\mid o_y))$.显然,逐出的对象应能腾出足够的空间来容纳$o_{iC}$,同时,还应保证访问开销损失最小.我们采用一种启发式贪婪算法来选择逐出的缓存对象.首先为各缓存对象计算规范化访问开销损失值NCL(normalized cost loss,即腾出单位存储空间带来的访问开销损失量).设对象o_x的大小为$s(o_x)$,则o_x的NCL值为$\frac{1}{s(o_x)}\cdot\sum_{o_y\in\Psi(o_x)}(f_v(o_y)\cdot m_v(o_x\mid o_y))$.按照NCL值的大小对缓存中的对象排序,并选择具有最小NCL值的对象作为逐出对象,直至腾出足够的缓存空间为止.

下面描述A_0和A_n之间的对象放置问题.图 3 所示为请求o_{iC}的并获得服务的快照.

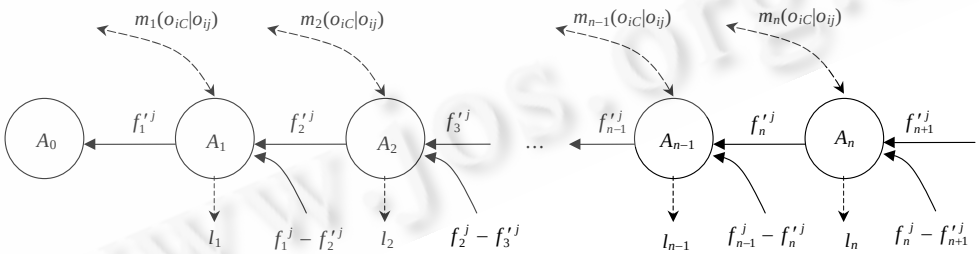

Fig.3 The analysis model for en-route transcoding caching scheme

图 3 En-Route transcoding 缓存的分析模型

令A_n为发出请求的节点,A_0中的缓存对象o_{iH}为服务对象.A_0将o_{iH}转换为o_{iC}(如果$o_{iH}\neq o_{iC}$),并将其沿路径$A_1,A_2,\ldots,A_{n-1}$向节点A_n传送.令f_k^j为节点A_k接收到的对o_{ij}的请求率,$f_k'^j$为穿越A_k的对o_{ij}的请求率.f_k^j主要来自两个方面:(1) 穿过A_{k+1}的请求率$f_{k+1}'^j$;(2) 来自以A_k为根的其他分支的请求率$f_k^j-f_{k+1}'^j$.设已知o_{iC}在各中间节点的缺失损失为$m_k(o_{iC}|o_{ij})$,其中,$j=0,1,\ldots,m-1,k=1,2,\ldots,n$.设在节点集$\{A_{v_1},A_{v_2},\ldots,A_{v_r}\}$上放置$o_{iC}$,其中,$1\le v_1,v_2,\ldots,v_r\le n,r\le n$.分两种情况来分析访问开销收益的求解,最后给出统一的表达.

情况 1. $o_{iC}=o_{i0}$,即o_{iC}为完整版本,它可以转换成其他任意版本.因此,对o_i任意版本的请求都可能从o_{iC}的放

置中获益.我们分别对两类请求的访问开销进行讨论:(1) 请求o_{i0}的访问开销节省量;(b) 请求非完整版本的访问开销节省量.

首先讨论请求o_{i0}的访问开销节省量.对o_{i0}的请求最终只可能在节点A_0得到满足.因此,A_k接收的对o_{i0}的请求将穿越$A_k,A_{k-1},\ldots,A_1$,直至A_0.在图 3 中, $f_k^0 - f_k'^0$ (k=1,2,…,n)成立.若在A_k放置o_{i0},则A_k将拦截所有对o_{i0}的请求.因此,在 $A_{v_1},A_{v_2},...,A_{v_r}$ 放置o_{i0}后, $A_{v_1},A_{v_2},...,A_{v_r}$ 接收到的对o_{i0}的请求率分别为

$$(f_{v_1}^0 - f_{v_2}^0),(f_{v_2}^0 - f_{v_3}^0),...,(f_{v_{r-1}}^0 - f_{v_r}^0), f_{v_r}^0 .$$

因此,请求o_{i0}的访问开销收益为

$$\sum_{x=1}^{r}((f_{v_x}^0 - f_{v_{x+1}}^0)\cdot m_{v_x}(o_{i0}\,|\,o_{i0})) \tag{5}$$

其中, $f_{v_{r+1}}^0$ 设置为 0.

对于非完整版本的请求相对更为复杂.因为某些中间节点可能已经缓存了o_{ij}($o_{ij}\neq o_{i0}$),对o_{ij}的请求可能会被这些中间节点拦截.因此在图 3 中, $f_k'^j = f_k^j$不一定成立.此外,如果$m_k(o_{i0}|o_{ij})$=0,那么即使在A_k放置o_{i0},它也不会成为服务节点,即A_k不会拦截对o_{ij}的请求.在 $A_{v_1},A_{v_2},...,A_{v_r}$ 放置o_{i0}后, A_{v_x} ($1\le x<r$)对o_{ij}的请求率变化与两个因素相关:(1) A_{v_x} 和 $A_{v_{x+1}}$ 之间是否缓存了o_{ij};(2) $m_{v_{x+1}}(o_{i0},|\,o_{ij})$ 的取值.为此,引入变量$Ind(v_x,v_{x+1},o_{ij})$和$Inf(v_x,o_{i0},o_{ij})$,分别指示v_x和v_{x+1}之间是否存在o_{ij}和 $m_{v_x}(o_{i0},|\,o_{ij})$ 的取值情况:

$$Ind(v_x,v_{x+1},o_{ij}) = \begin{cases} 0, & \exists k(v_x \le k \le v_{x+1}), o_{ij} \in H_{A_k} \\ 1, & \text{else} \end{cases} \tag{6}$$

$$Inf(v_x,o_{i0},o_{ij}) = \begin{cases} 0, & m_{v_x}(o_{i0}\,|\,o_{ij}) = 0 \\ 1, & m_{v_x}(o_{i0}\,|\,o_{ij}) > 0 \end{cases} \tag{7}$$

因此,在节点 $A_{v_1},A_{v_2},...,A_{v_r}$ 上放置o_{i0},请求o_{ij}($o_{ij}\neq o_{i0}$)的访问开销节省量为

$$\sum_{x=1}^{r}((f_{v_x}^j - Ind(v_x,v_{x+1},o_{ij})\cdot Inf(v_{x+1},o_{i0},o_{ij})\cdot f_{v_{x+1}}^j)\cdot m_{A_{v_x}}(o_{i0}\,|\,o_{ij})) \tag{8}$$

其中, $f_{v_{r+1}}^j$ 和$Inf(v_{r+1},o_{i0},o_{ij})$设置为 0.

综合以上分析,在节点 $A_{v_1},A_{v_2},...,A_{v_r}$ 上放置o_{i0},总访问开销节省量为

$$\begin{aligned}&\sum_{x=1}^{r}((f_{v_x}^0 - f_{v_{x+1}}^0)\cdot m_{A_{v_x}}(o_{i0}\,|\,o_{ij})) + \sum_{o_{ij}\in\Psi(o_{iC})-\{o_{iC}\}}\sum_{x=1}^{r}((f_{v_x}^j - Ind(v_x,v_{x+1},o_{ij})\cdot Inf(v_{x+1},o_{i0},o_{ij})\cdot f_{v_{x+1}}^j)\cdot m_{A_{v_x}}(o_{i0}|\,o_{ij})) = \\ &\sum_{x=1}^{r}((f_{v_x}^0 - f_{v_{x+1}}^0)\cdot m_{A_{v_x}}(o_{i0}\,|\,o_{ij}) + \sum_{o_{ij}\in\Psi(o_{iC})-\{o_{iC}\}}(f_{v_x}^j - Ind(v_x,v_{x+1},o_{ij})\cdot Inf(v_{x+1},o_{i0},o_{ij})\cdot f_{v_{x+1}}^j)\cdot m_{A_{v_x}}(o_{i0}|\,o_{ij}))\end{aligned} \tag{9}$$

考虑到逐出对象带来的访问开销损失,在节点 $A_{v_1},A_{v_2},...,A_{v_r}$ 上放置对象o_{i0},系统的访问开销收益为

$$\sum_{x=1}^{r}((f_{v_x}^0 - f_{v_{x+1}}^0)\cdot m_{A_{v_x}}(o_{i0}\,|\,o_{ij}) + \sum_{o_{ij}\in\Psi(o_{iC})-\{o_{iC}\}}(f_{v_x}^j - Ind(v_x,v_{x+1},o_{ij})\cdot Inf(v_{x+1},o_{i0},o_{ij})\cdot f_{v_{x+1}}^j)\cdot m_{A_{v_x}}(o_{i0}|\,o_{ij}) - l_{v_x}) \tag{10}$$

其中, $f_{v_{r+1}}^j$, $f_{v_{r+1}}^0$ 和$Inf(v_{r+1},o_{i0},o_{ij})$设置为 0.

情况 2. $o_{iC}\neq o_{i0}$,即o_{iC}为非完整版本.根据假设 1,只有对o_{iC}的请求才能从o_{iC}的放置中获益.易得,在节点 $A_{v_1},A_{v_2},...,A_{v_r}$ 上放置对象o_{iC}后,请求o_{iC}带来的访问开销节省量为 $\sum_{x=1}^{r}((f_{v_x}^j - f_{v_{x+1}}^j)\cdot m_{A_{v_x}}(o_{iC}\,|\,o_{iC}))$,其中, $f_{v_{x+1}}^j$ 设置为 0.因此,在节点 $A_{v_1},A_{v_2},...,A_{v_r}$ 上放置对象o_{iC}($o_{iC}\neq o_{i0}$),系统的总收益为

$$\sum_{x=1}^{r}((f_{v_x}^j - f_{v_{x+1}}^j)\cdot m_{A_{v_x}}(o_{iC}\,|\,o_{iC}) - l_{v_x}) \tag{11}$$

综合式(10)和式(11),在节点 $A_{v_1},A_{v_2},...,A_{v_r}$ 上放置对象o_{iC},系统的总收益可以统一表达为

$$\sum_{x=1}^{r}((f_{v_x}^C - f_{v_{x+1}}^C)\cdot m_{A_{v_x}}(o_{iC}\,|\,o_{iC}) + \sum_{o_{ik}\in\Psi(o_{iC})-\{o_{iC}\}}(f_{v_x}^k - Ind(v_x,v_{x+1},o_{ik})\cdot Inf(v_{x+1},o_{i0},o_{ik})\cdot f_{v_{x+1}}^k)\cdot m_{A_{v_x}}(o_{i0}|\,o_{ik}) - l_{v_x}) \tag{12}$$

其中, $f_{v_{r+1}}^{k}$, $f_{v_{r+1}}^{C}$ 和$Inf(v_{r+1},o_{iC},o_{ik})$设置为 0.

本文的目标是确定中间节点集的一个子集$\{A_{v_1},A_{v_2},...,A_{v_r}\}$以放置$o_{iC}$,使得式(12)取得最大,进而使系统的总体访问开销达到最小.

4.2 基于动态规划的求解

首先给出缓存对象放置问题的通用定义.

定义 3(K-优化问题). $\forall k(1\le k\le n),o_{ij}\in\psi(o_{iC})$,给定 f_k^j, $m_k(o_{iC}|o_{ij})$和l_k,并令 $f_{n+1}^j=0$.令整数K满足 $0\le K\le n$, $v_1,v_2,\ldots,v_r$为r个整数,且满足 $1\le v_1<v_2<\ldots<v\le K_r$.定义目标函数$\Delta cost(K:v_1,v_2,\ldots,v_r)$为

$$\Delta cost(K:v_1,v_2,...,v_r)=\sum_{x=1}^{r}((f_{v_x}^{C}-f_{v_{x+1}}^{C})\cdot m_{A_{v_x}}(o_{iC}\mid o_{iC})+\sum_{o_{ik}\in\Psi(o_{iC})-\{o_{iC}\}}(f_{v_x}^{k}-Ind(v_x,v_{x+1},o_{ik})\cdot Inf(v_{x+1},o_{i0},o_{ik})\cdot f_{v_{x+1}}^{k})\cdot m_{A_{v_x}}(o_{i0}\mid o_{ik})-l_{v_x}),$$

其中,

$$\begin{cases}Ind(v_r,v_{r+1},o_{ik})=Ind(v_r,K,o_{ik})\\ Inf(v_{r+1},o_{iC},o_{ik})=\begin{cases}Inf(K+1,o_{iC},o_{ik}), & K<n\\ 0, & K=n\end{cases}\\ f_{v_{r+1}}^{j}=f_{K+1}^{j}\end{cases}.$$

如果r=0,定义$\Delta cost(K:\varnothing)$=0.如何确定$r$和$v_1,v_2,\ldots,v_r$,使得$\Delta cost(K:v_1,v_2,\ldots,v_r)$取得最大,这个问题称为$K$-优化问题.

定理 1. 令 $1\le k\le n,r>0$.设$v_1,v_2,\ldots,v_r$是K-优化问题的一个最优解, $u_1,u_2,...,u_{r'}$ 为(v_r-1)-优化问题的一个最优解,那么, $u_1,u_2,...,u_{r'},v_r$ 也是K-优化问题的一个最优解(证明略).

定义 4. 令 $0\le k\le n$,定义OPT_K是从K-优化问题的最优解放置方案$\{v_1,v_2,\ldots,v_{r-1},v_r\}$所对应的$\Delta cost(K:v_1,v_2,\ldots,v_{r-1},v_r)$值,$L_K$为最优解中的最大编号.如果最优解为空集,则定义$L_K=-1$.

显然有,$OPT_0=0$,$L_0=-1$.根据定理 1,如果$L_K>0$,

$$OPT_K=OPT_{(L_k-1)}+(f_{L_K}^{C}-f_{L_{K+1}}^{C})\cdot m_{L_K}(o_{iC}\mid o_{iC})+\sum_{o_{ik}\in\Psi(o_{iC})-\{o_{iC}\}}(f_{L_K}^{k}-Ind(L_K,K,o_{ik})\cdot Inf(K+1,o_{iC},o_{ik})\cdot f_{K+1}^{k})\cdot m_{L_K}(o_{iC}\mid o_{ik})-l_{L_K}.$$

检查L_K所有可能的取值,并选取使OPT_K最大的L_K值.特别地,可以建立如下状态转换函数:

$$\begin{cases}OPT_0=0\\ OPT_K=\max\{0,OPT_{j-1}+(f_j^{C}-f_{K+1}^{C})\cdot m_j(o_{iC}\mid o_{iC})+\sum_{o_{ik}\in\Psi(o_{iC})-\{o_{iC}\}}(f_j^{k}-Ind(j,K,o_{ik})\cdot Inf(K+1,o_{iC},o_{ik})\cdot f_{K+1}^{k})\cdot m_j(o_{iC}\mid o_{ik})-l_j,(j=1,2,...,K)\},\forall K(1\le K\le n)\end{cases}\tag{13}$$

$$\begin{cases}L_0=-1\\ L_K=\begin{cases}v, & \text{如果}v\text{满足}OPT_K=OPT_{v-1}+(f_v^{C}-f_{K+1}^{C})\cdot m_v(o_{iC}\mid o_{iC})+\sum_{o_{ij}\in\Psi(o_{iC})-\{o_{iC}\}}(f_v^{j}-Ind(v,K,o_{ij})\cdot Inf(K+1,o_{iC},o_{ij})\cdot f_{K+1}^{j})\cdot m_v(o_{iC}\mid o_{ij})-l_v\\ -1, & \text{如果}OPT_K=0\end{cases}\end{cases}\tag{14}$$

因此,对象o_{iC}在节点集$\{A_1,A_2,\ldots,A_n\}$的放置问题即为n-优化问题,并可以利用动态规划方法进行求解.在计算出OPT_k和$L_k(0\le k\le n)$之后,从$v_r=L_n$开始,对于 $1\le i<r$,迭代地设置 $v_i=L_{v_{i+1}-1}$.最终获得的r和 $\{A_{v_1},A_{v_2},...,A_{v_r}\}$ 即为最优的版本放置策略.

4.3 协议设计

协同 En-Route transcoding 缓存协议由参数估算与信息收集、缓存放置决策、对象放置与参数更新 3 个阶段组成.

4.3.1 参数估算与信息收集

各缓存节点为每个缓存对象维护了一组参数信息,包括对象版本、请求率、对应于各种请求的缺失损失等.各节点记录到达的请求信息,包括所请求的对象和版本、到达时间等.根据这些历史记录,采用“滑动窗口法”[13]来估算对象版本的请求率.当请求报文经过中间节点A_k时,A_k搜集所需参数信息,并将它们附加在请求报文中(见表 1).其中,$d(A_k,A_{k+1})$为A_k的下游节点A_{k+1}至A_k的传输延迟,如果$A_k=A_n$,则设置$d(A_k,A_{k+1})=0$.$BeCached_k(o_{ij})$参数用于指示对象o_{ij}是否存在于节点A_k的缓存中.如果$o_{ij}\in H_{v_k}$,则$BeCached_k(o_{ij})=1$;否则$BeCached_k(o_{ij})=0$.

Table 1 Information piggybacked by a request when passing through A_k

表 1 经过节点A_k时捎带的信息

$f_k(o_{iC})$	l_k	$d(A_k,A_{k+1})$	$\forall o_{ij}\in\psi(o_{iC})-\{o_{iC}\}$		
			$f_k(o_{ij})$	$BeCached_k(o_{ij})$	$m_k(o_{ij}\|o_{ij})$

4.3.2 缓存放置决策

根据请求报文“捎带”的参数信息,A_0计算$m_k(o_{iC}|o_{ij})$,$Inf(v_k,o_{iC},o_{ij})$,$Ind(v_k,v_{k+1},o_{ij})$,$\forall o_{ij}\in\psi(o_{iC})$,$1\le k\le n$,并利用动态规划算法求解$o_{iC}$在$\{A_1,A_2,\ldots,A_n\}$中的最佳放置策略.

4.3.3 对象放置与参数更新

A_0执行$o_{iH}\to o_{iC}$的编码转换,并将o_{iC}沿$A_1\to A_2\to\ldots\to A_n$传递响应报文.报文包括$o_{iC}$和放置决策信息.当响应报文经过$A_k$,且需要在$A_k$放置$o_{iC}$时,$A_k$保留$o_{iC}$的副本(如果缓存空间不足,则执行缓存替换操作).为计算各放置节点中o_{iC}的缺失损失,响应报文分别维护了相应的变量.在A_0中这些变量被赋予不同的初值;在传输过程中,各中间节点根据一定的规则对各变量进行更新.

(1) $m(o_{iC}|o_{iC})$的更新.变量m的初始值设置为$o_{iH}\to o_{iC}$的编码转换开销.在途中的每个中间节点,m值被增加上一节点至当前节点传输o_{iC}的网络开销.如果当前节点为放置节点,m值用于更新在该节点对象o_{iC}因请求o_{iC}的缺失损失,同时,m值被重置为 0.

(2) $m(o_{iC}|o_{ij})$的更新.由于在部分中间节点中可能已经缓存了对象$o_{ij}(o_{ij}\in\psi(o_{iC})-\{o_{iC}\})$,$m(o_{iC}|o_{ij})$的更新相对更为复杂.变量$m'$的初始值被设定为$o_{iH}\to o_{iC}$的编码转换开销和$m_0(o_{ij}|o_{ij})$的最小值.途中的每个中间节点,$m'$值被增加上一节点至当前节点传输$o_{ij}$的网络开销.如果当前节点已经缓存了$o_{ij}$,$m'$被重置为 0;放置节点比较$m'$的取值和$o_{iC}\to o_{ij}$的编码转换开销的大小:如果$m'>(1-\alpha)\cdot w(o_{iC},o_{ij})$,则$m(o_{iC}|o_{ij})$设定为$m'-(1-\alpha)\cdot w(o_{iC},o_{ij})$,并将$m'$重置为 0;否则,$m(o_{iC}|o_{ij})$设置为 0.

以上操作重复执行,直至完成最后一个放置节点的参数更新为止.

5 性能评价

5.1 实验设置

为了模拟内容分发系统的网络环境,我们用Brite[14]拓扑生成工具生成了节点数为 1 000 的网络拓扑,节点间的平均传输延迟设置为 0.4s.随机选择 200 个节点作为en-route缓存节点,5 个节点作为源内容服务器节点.每个服务器维护了1 000个完整的对象版本,对象流行度满足θ=0.75的Zipf分布,对象大小满足μ=350MB, δ=50MB的正态分布.各节点的请求到达序列用λ=0.2 的Poisson分布模型产生,所请求的对象在各服务器中均匀分布.根据设备能力,用户被分为 5 种类型,所占比例分别为 0.15,0.2,0.3,0.2 和 0.15.设满足各类用户播放需求的版本的大小分别为原始版本大小的 100%,80%,60%,40%和 25%.我们根据式(1)来模拟版本间的转换开销,并假设解码开销和编码开销分别与原版本和新版本的大小成正比,且有$\tau_e=5\cdot\tau_d$.在实验中,τ_d设定为 0.1.令系统中各内容对象的完整版本的大小总和为S,我们用“相对缓存大小”来描述总的缓存空间大小与S的比值,各en-route缓存节点的缓存空间大小由$\mu=1\%\cdot S$,$\sigma=0.1\cdot\mu$的正态分布模型产生.除了本文提出的优化放置策略OPT(optimal的缩写)外,还实现了LCE(leave copy every),LCD(leave copy down),PROB(probability-based的缩写)等 3 种元算法,它们分别与LRU(least recently used)和LFU(least frequently used)缓存替换算法进行组合可以形成 6 种缓存放置与替换策

略.在模拟实验中,我们以开销节省率CSR(cost saving ratio)为主要的性能指标.实验中,我们记录每次请求的实际访问开销和没有缓存情况下(即从源服务器获得服务)的访问开销.设所有请求的实际访问开销总和为$Access_cost_{withcache}$,没有缓存情况下的访问开销总和为$Access_cost_{withoutcache}$,CSR定义为

$$CSR=1-Access\_cost_{withcache}/Access\_cost_{withoutcache}.$$

5.2 结果分析

5.2.1 缓存大小的影响

首先考察缓存大小对系统性能的影响.对象流行度参数设定为 0.8,影响因子设定为 0.5,改变相对缓存大小参数值(从 1%~10%),分别运行不同的缓存算法,取 10 次运行的平均值进行分析.图 4 所示为各缓存算法在不同缓存大小下的 CSR 取值情况.可以看出,随着缓存空间的增加,所有缓存算法的 CSR 值都呈增大的趋势.而且在各缓存大小情况下,OPT 算法的 CSR 性能都优于其他算法.在其他缓存算法中,LCD 组合(LCD-LRU 和 LCD-LFU)的 CSR 性能最好,PROB 组合次之,而 LCE 组合的 CSR 性能表现最差.这是因为,在 OPT 算法中,各相关节点以最小化未来请求的访问开销为目标,协同地进行缓存对象的放置和替换.而元算法采用简单的放置策略,没有综合考虑在节点放置对象的获益和替换出对象带来的损失情况.图 5 和图 6 展示了不同缓存大小下各算法的版本命中率和可用命中率情况.可以看出,各算法的版本命中率和可用命中率都随着节点缓存大小的增加而有所提高.在相同缓存大小情况下,OPT 算法的版本命中率均明显高于其他算法,可用命中率则略高于 LCD-LRU 算法.可见,OPT 算法中更多的请求都能在缓存节点中得到服务,这也进一步说明了 OPT 具有较高 CSR 性能的原因.

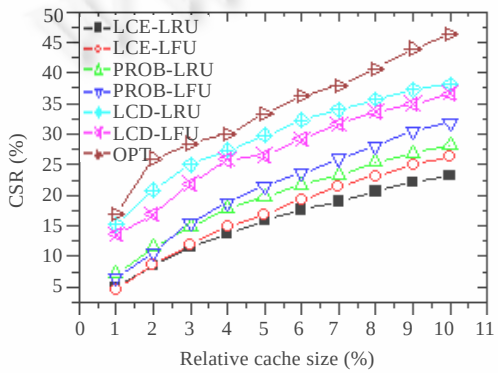

Fig.4 CSR vs. relative cache size

图 4 CSR vs.相对缓存大小

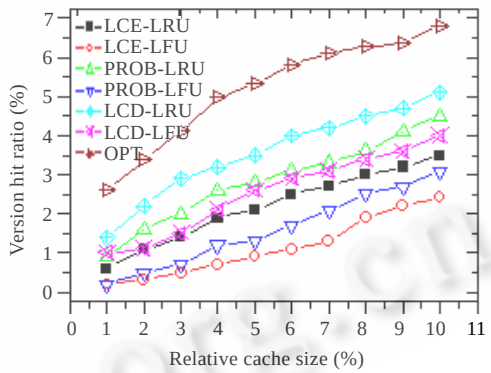

Fig.5 Version hit ratio vs. relative cache size

图 5 版本命中率 vs.相对缓存大小

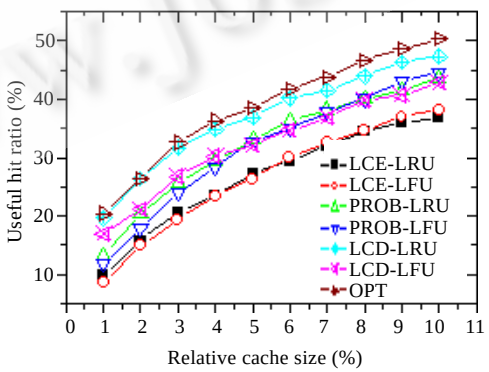

Fig.6 Useful hit ratio vs. relative cache size

图 6 可用命中率 vs.相对缓存大小

5.2.2 流行度的影响

本节考察不同对象流行度分布下缓存系统的性能.我们将影响因子设定为 0.5,相对缓存大小为 3%.在实验中,我们改变对象的流行度分布参数β的取值(从 0.7~1.2),分别运行不同的缓存算法.同样,我们取 10 次运行结果的平均值进行分析.图 7 展示了不同对象流行度分布下,各缓存算法 CSR 的取值情况.当β取值增加时,更多的请求集中在更少的对象上,各缓存算法的效率也相应地提高,表现出 CSR 值的增加.但在所有流行度分布情况下,OPT 算法都表现出最好的 CSR 性能.这是因为,在 OPT 算法中以降低总体访问开销为目的进行了优化设计,因而表现出更好的缓存效率.

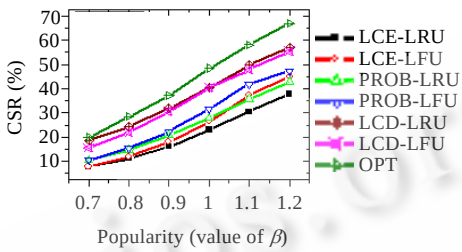

Fig.7 CSR vs. popularity distribution

图 7 CSR vs.流行度分布

5.2.3 影响因子α的影响

在式(2)中,影响因子α用来表达编码转换开销和网络传输开销在访问开销中所占的比重.影响因子α是一个可配置的参数,在系统运行中,可根据系统资源状态或者价格设置合适的α值.为了考察影响因子的变化对系统性能的影响,我们设定对象流行度分布参数为β=0.8,相对缓存大小为 3%,在实验中,改变影响因子α的取值(从 0.1~0.9),运行模拟程序.图 8 所示为不同α取值下各缓存算法的 CSR 取值情况(10 次运行结果的平均值).从图中可以看出,随着α值的变化,LCE 算法的 CSR 性能出现了较大的波动,而其他算法的 CSR 性能表现较为稳定.在不同的影响因子配置下,OPT 算法都表现出优于其他算法的 CSR 性能.可见,OPT 算法的性能对影响因子α的取值不敏感.

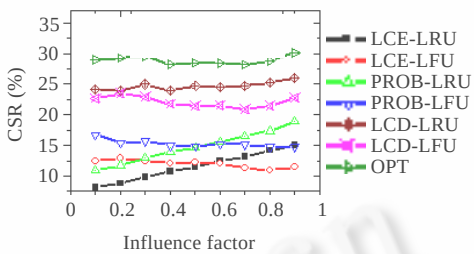

Fig.8 CSR vs. influence factor

图 8 CSR vs.影响因子

6 结 论

在普适内容分发服务中,利用多个节点协同地进行内容对象的编码转换和缓存放置,是提高系统效率和可扩展性的有效途径.但是,对象之间的转换关系给对象的协同放置和替换带来了新的问题.本文以最小化系统服务开销为目标,提出了一种 active 缓存路由算法 CCRA.在此基础上,建立了 en-route transcoding 缓存的分析模型,提出缓存放置问题的优化模型,并给出了基于动态规划的计算方法和协议设计.模拟实验和分析表明,本文提出的协同放置和替换策略在不同的应用环境和资源条件下均表现出较好的性能.

References:

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

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

[3] Chu JH, Yu SL, Lu ZH. Research on transcoding technology. ACTA Electronica Sinica, 2004,32(10):1678−1683 (in Chinese with English abstract).

[4] Canali C, Cardellini V, Colajanni M, Lancellotti R, Yu PS. Cooperative TransCaching: A system of distributed proxy servers for Web content adaptation. In: Poster Proc. of the 12th Int'l World Wide Web Conf. (WWW 2003). New York: ACM Press, 2003. 321−323. http://www.informatik.uni-trier.de/~ley/db/conf/www/www2003p.html

[5] Canali C, Cardellini V, Colajanni M, Lancellotti R, Yu PS. Cooperative architectures and algorithms for discovery and transcoding of multi-version content. In: Douglas F, Davison BD, eds. Proc. of the 8th Int'l Workshop on Web Content Caching and Distribution. Norwell: Kluwer Academic Publishers, 2004. 205−221.

[6] Li KQ, Shen H, Chin F, Zheng S. Optimal methods for coordinated en-route Web caching for tree networks. ACM Trans. on Internet Technology (TOIT), 2005,5(3):480−507.

[7] Li KQ, Shen H, Chin FYL, Zhang WS. Multimedia object placement for transparent data replication. IEEE Trans. on Parallel and Distributed Systems, 2007,18(2):212−224.

[8] Tang X, Chanson ST. Coordinated en-route Web caching. IEEE Trans. on Computers, 2002,51(6):595−607.

[9] Jiang AX, Bruck J. Optimal content placement for en-route Web caching. In: Proc. of the 2nd IEEE Int'l Symp. on Network Computing and Applications. Washington: IEEE Computer Society, 2003. 9−16. http://portal.acm.org/citation.cfm?id=824470.825330&coll=portal&dl=ACM#

[10] Laoutaris N, Syntila S, Stavrakakis I. Meta algorithms for hierarchical Web caches. In: Hassanein H, ed. Proc. of the IEEE Int'l Performance Computing and Communications Conf. (IPCCC 2004). Piscataway: IEEE, 2004. 445−452.

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

[12] Yamaoka S. Resource-Aware service composition for video multicast to heterogeneous mobile users. In: Balke W-T, Nahrstedt K, eds. Proc. of the 1st ACM Int'l Workshop on Multimedia Service Composition (MSC 2005). New York: ACM Press, 2005. 37−46.

[13] Breslau L, Jamin S, Shenker S. Comments on the performance of measurement-based admission control algorithms. In: Sidi M, ed. Proc. of the IEEE INFOCOM 2000. Piscataway: IEEE Communication Society, 2000. 1233−1242.

[14] Medina A, Matta I, Byers J. On the origin of power laws in Internet topologies. ACM Computer Communications Review, 2000, 30(2):18−28.

附中文参考文献:

[3] 褚晶辉,俞斯乐,鲁照华.视频转换编码及其实现技术的研究.电子学报,2004,32(10):1678−1683.

李春洪(1972－),男,江苏丹阳人,博士,主要研究领域为分布式系统,普适计算.

陆桑璐(1970－),女,博士,教授,博士生导师,CCF 高级会员,主要研究领域为分布式与并行系统,高性能计算.

冯国富(1977－),男,博士,主要研究领域为分布与并行系统,对等计算.

陈道蓄(1947－),男,教授,博士生导师,CCF 高级会员,主要研究领域为分布式系统,Internet 计算, CSCW.

顾铁成(1965－),男,副教授,主要研究领域为分布与并行系统.