ISSN 1000-9825, CODEN RUXUEW
*Journal of Software*, Vol.18, No.1, January 2007, pp.146−156
DOI: 10.1360/jos180146

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

# 内容传递网络处理能力受限代理放置贪婪算法*

陈益峰[1+], 何炎祥[2], 曹建农[3]

[1](武汉大学 水资源与水电工程科学国家重点实验室,湖北 武汉 430072)

[2](武汉大学 计算机学院 软件工程国家重点实验室,湖北 武汉 430072)

[3](香港理工大学 电子计算学系,香港)

## A Greedy Algorithm for Capacity-Constrained Surrogate Placement in CDNs

CHEN Yi-Feng[1+], HE Yan-Xiang[2], CAO Jian-Nong[3]

[1](State Key Laboratory of Water Resources and Hydropower Engineering Science, Wuhan University, Wuhan 430072, China)

[2](School of Computer, State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China)

[3](Department of Computing, Hong Kong Polytechnic University, Hong Kong, China)

+ Corresponding author: Phn: +86-27-68772221, Fax: +86-27-68772310, E-mail: csyfchen@whu.edu.cn

**Chen YF, He YX, Cao JN. A greedy algorithm for capacity-constrained surrogate placement in CDNs. *Journal of Software*, 2007,18(1):146−156**. http://www.jos.org.cn/1000-9825/18/146.htm

**Abstract**: A new surrogate placement strategy, CCSP (capacity-constrained surrogate placement), is proposed to enhance the performance for content distribution networks (CDNs). CCSP aims to address surrogate placement in a manner that minimizes the communication cost while ensuring at the same time the maximization of system throughput. This work differs from the existing works on the resource allocation problem in communication networks, CCSP considers load distribution and processing capacity constraints on surrogates by modeling the underlying request-routing mechanism, thus guaranteeing a CDN to have minimum network resource consumption, maximum system throughput, and better load balancing among surrogates. An efficient greedy algorithm is developed for a simplified version of the CCSP problem in tree networks. The efficiency of the proposed algorithm is systematically analyzed through the experimental simulations.

**Key words**: content distribution network; surrogate placement; load balancing; greedy algorithm

摘 要: 提出了旨在提高内容传递网络服务性能的代理放置策略 CCSP(capacity-constrained surrogate placement).CCSP 在保证最大化系统吞吐量的条件下,以最小化系统通信开销为目标,求解最优的代理放置方式.与通信网络中的资源分配问题现有求解策略不同,CCSP 通过模拟内容传递网络的请求路由机制,考虑了代理服务器的负载分布及处理能力约束,从而保证系统具有最低的资源消耗、最大的吞吐能力和良好的负载均衡.提出了高效的贪婪算法用以求解树型网络条件下的 CCSP 问题,并通过仿真实验系统地分析了算法的有效性.

* Supported by the National Natural Science Foundation of China under Grant No.90104005 (国家自然科学基金); the Natural Science Foundation of Hubei Province of China under Grant No.2003ABA047 (湖北省自然科学基金); the Science-Technology Plan of Hubei Province of China under Grant Nos.2002S4108, 2002AA102B06 (湖北省科技计划); the University Grant Council of Hong Kong under CERG Grant No.PolyU5105/05E (香港大学研究基金)

Received 2004-08-26; Accepted 2006-03-31

关键词: 内容传递网络;代理放置;负载均衡;贪婪算法
中图法分类号: TP393 文献标识码: A

当前,Internet 上的大部分网络流量是由用户对少数流行 Web 站点的访问产生的,这些站点为了在竞争中胜出,希望以较低的成本提供更好的服务.实现这一目标的有效策略是:从服务提供者(如 Akamai,Exodus,Digital Island 等)租用计算和存储资源,从而为 Web 内容提供者提供经济有效的文档存储、维护和访问方式.为此而建立的基础设施称为内容传递网络(content distribution network,简称 CDN),其基本思想是:在源服务器站点与客户之间安插代理或缓存,每个客户请求由客户附近的代理或缓存作出响应,同时保证适当的完整性和一致性[1],从而减轻源服务器的负载,避免内容在网络上的重复传输,减小客户请求的响应时间,并改善 Web 服务的可靠性、容错性和可扩展性[2].

代理放置问题和请求路由问题是对 CDN 性能有重大影响的两个基本问题.前者决定代理服务器的个数和放置位置;后者决定如何将客户请求定向到邻近的目标代理服务器.这两个问题已经独立地得到广泛研究,然而,二者决策之间的内在联系和相互影响并没有引起足够的重视.一方面,代理放置策略依赖请求路由机制为每个客户请求选择合适的目标代理,所以不同的请求路由机制导致不同的最优代理放置策略;另一方面,请求路由机制依赖动态的网络状态和服务器负载信息来分配客户请求,实现负载均衡,因而不同的代理放置策略导致同一请求路由机制获得不同的吞吐能力和负载均衡效果.现有的研究工作采用以网络为中心的观点求解代理放置问题[3−9],认为客户请求总是能够被定向到最邻近的代理服务器,所以他们在描述代理放置问题时仅考虑网络延迟.然而,最邻近代理选择机制很可能导致某些代理的负载集中,这意味着 CDN 规划应保证每台代理服务器具有强大的处理能力,以处理这种负载集中[3].问题在于,当人们出于管理或经济上的考虑需要配置一组同构的代理服务器,或者当内容提供者想从服务提供者租用特定容量的代理服务器时,代理的处理能力是预先给定的,可能难以根据需要随意升级.超载的代理服务器或者丢弃请求,或者将部分请求重定向到其他代理,前者降低系统吞吐量,后者则增加网络流量,二者均导致 CDN 性能的显著下降.换言之,现有的代理放置决策由于没有模拟 CDN 的请求路由机制,完全忽略了服务器的负载及其接入带宽限制.以网络为中心的观点存在的另一个问题是不能有效地评价 CDN 的性能,因为严重的代理负载不均衡会显著降低 CDN 的性能.

本文研究处理能力受限的代理放置问题(capacity-constrained surrogate placement,简称 CCSP),其基本思路是采用队列理论[10]模拟服务器吞吐能力,并根据流量模式和特定的请求路由机制,将超载服务器上的部分客户请求重定向到其他轻载代理服务器,解决代理之间的负载均衡,从而在系统吞吐量达到最大的条件下使网络流量最小化.尽管人们已经在文件分配[11,12]、数据对象放置[13,14]和容量受限的设施布局[15]等研究领域提出了具有与 CCSP 问题类似目标的求解技术,但这些技术因为没有同时考虑处理器的处理能力和访问重定向,不能直接用于求解代理放置问题.除非采用集中方式调度客户请求,如 IBM's Network Dispatcher[16],从而完全控制客户请求路由.但集中式的请求调度引入单一决策实体,它随着请求数量的增加会成为性能瓶颈和系统的单故障点.

根据层次式缓存[5,9,17,18]的请求路由机制,我们将 CDN 抽象为一棵以源服务器为根结点的树,并将置于其上的代理组织成层次结构.这种拓扑结构可以显著降低客户重定向的管理成本,简化代理协作及负载平衡的方案设计.在这种拓扑结构中,每个代理有一个父代理和一个直接子孙代理集合.每个客户请求沿着客户到源服务器之间的唯一路径向源服务器转发.如果在转发路径中遇到一个尚未超载的代理服务器,且该代理服务器持有客户所请求的数据,则该代理代表源服务器直接为客户服务.如果在转发路径中没有遇到这样的代理服务器,则该请求继续沿代理层次结构向源服务器转发,直至遇到一个可用的代理服务器为止.通过在代理层次结构中复制内容,CDN 可以消除数据在网络路径上的重复传输,从而减小网络流量.但代理层次结构上的多个数据副本引入了代理上的复制内容与源服务器上的内容之间的副本一致性维护问题.我们采用应用层组播维护副本的一致性.在这种一致性维护方法中,源服务器发出的更新消息以组播方式沿代理层次结构逐层向下传递[9,19].这种一致性维护方法的优点是可以设计灵活、简单的算法:它既可以采用推的技术实现,也可以采用拉的技术实现.

图 1 给出了一个树型 CDN 拓扑.一组代理服务器(包括源服务器)放置在结点集{1,6,10,15,20}中.这些代理

服务器彼此协作,为客户请求提供服务并维护副本的一致性.如在图 1 中,结点 11 发出的请求沿结点 11 到结点 1 的唯一路径向源服务器转发.正常情况下,放置在结点 10 的代理将截取并代表源服务器直接响应该请求.然而,如果该代理超载,则该请求将继续沿树结构向上转发,直至遇到另一个可用的代理服务器,如结点 6 为该请求提供服务;而更新消息则首先从结点 1(即源服务器)向其直接子孙代理结点 6 和结点 20 传播,然后继续从结点 6 沿代理层次结构向下传播到结点 10 和结点 15.需要指出的是,当前的商业 CDN 通常采用客户 DNS 服务器选择机制及轮次、概率轮次或随机等选择算法实现客户请求重定向,且未必严格采用树型结构组织代理服务器.然而,考虑到当前 CDN 相对静态,缺乏可扩展性,因此采用 P2P 机制构建协作型 CDN 成为解决 CDN 可扩展性的重要技术手段[20].在 P2P CDN 环境以及移动计算环境下,动态 CDN 拓扑即可组织为上述树型结构,这使本文的研究具有一定的现实意义和应用前景.

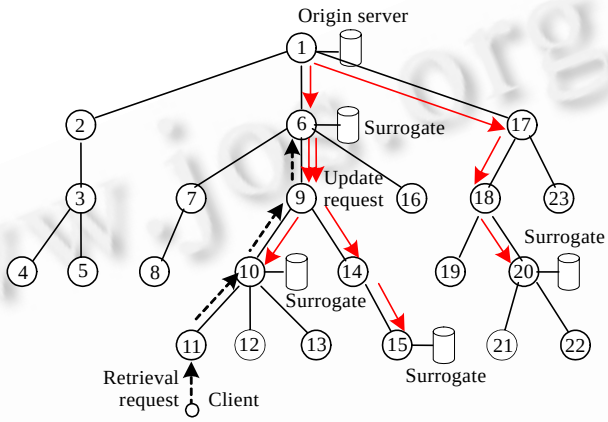


Fig.1 Request-Routing and consistency maintenance under transparent data replication

图 1 树型 CDN 的请求路由及一致性维护

## 1 系统模型

### 1.1 服务器模型

考虑到一个计算机系统的性能通常受到某个瓶颈部件(如访问服务器的资源带宽)的限制,本文利用 M/G/1/K*PS 队列模型[10]模拟 CDN 服务器.在 M/G/1/K*PS 队列模型中,假设 HTTP 请求(包括检索请求和更新请求)的到达过程服从到达率为$(\lambda+\mu)$(即读请求为$\lambda$,写请求为$\mu$)的泊松过程,而服务时间需求服从均值为 $\bar{x}$ 的一般分布.服务规则为处理器共享,即队列中的作业在用完规定的时间片后便被挂起,直至队列中的其他每个作业以轮次(round-robin)方式获得等量的服务为止.系统中可被同时处理的请求总数限制为 $K$,即如果系统中的作业总数达到了预定值 $K$,则新的到达将被阻塞.以 $P^{b*}$表示阻塞概率,则

$$P^{b*}=\frac{(1-\rho)\rho^{K}}{1-\rho^{K+1}} \tag{1}$$

式中,$\rho$为输入的通信流量,$\rho=(\lambda+\mu)\bar{x}$.

因此,系统中被阻塞的请求率为$(\lambda+\mu)P^{b*}$,系统吞吐量为 $H=(\lambda+\mu)(1-P^{b*})$.CDN 通过代理服务器之间的协作,将已路由到某代理服务器的过载请求(即$(\lambda+\mu)P^{b*}$)重定向到其他轻载的服务器,从而提高整个系统的吞吐能力.考虑到更新请求总是要求本地服务,只有检索请求可以被转发到其他代理服务器,且 CDN 服务器通常以检索请求为主,因此,通过定义 $P^{b}=(1+\mu/\lambda)P^{b*}$,则阻塞的请求率可以表示为$\lambda P^{b}$.下文中,我们总是使用 $P^{b}$ 而非 $P^{b*}$.CDN 服务器的处理能力由 $\bar{x}$ 和 $K$ 两个参数描述,我们假设它们是已知的.实际上,如果针对特定到达率的平均响应时间可以通过测量加以估计,则 $\bar{x}$ 和 $K$ 的估计量就可以通过最大化平均响应时间观测值的似然函数得到[10].

### 1.2 树型网络下的问题描述

我们把网络拓扑模拟为一棵树 $T_r(V,E)$,其中:$V$ 为结点集;$E\subseteq V\times V$ 为边集;$r\in V$ 为源服务器所在的根结点.树中的每个结点表示一个自治系统(AS),每条边对应于连接两个自治系统的物理链路.文献[3]讨论了在 AS 级上描述代理放置问题的一些优点.另外一个优点是,各 AS 级上的通信模式是相对稳定的,尽管客户群每天都在变化[5].静态代理放置正是因为这个优点才显现出其研究意义和应用价值.

假设源服务器持有 $N$ 个对象,每个对象 $i$ 的大小为 $s_i(1\le i\le N)$.对于任意对象 $i$,每个结点 $v$ 关联一个非负的检索请求率 $\lambda_{v,i}$.我们还假设所有的数据更新请求由源服务器发起.对于对象 $i$,源服务器将更新消息以非负的请求率 $\mu_i$ 组播到所有的代理服务器上.$E$ 中的每条链路 $(u,v)$ 关联一个距离量度 $d(u,v)$,它可以解释为带宽,跳数或链路成本等.假设 $\pi_{x,y}$ 是结点 $x$ 和 $y$ 之间的一条路径,则与路径 $\pi_{x,y}$ 关联的距离可以表示为 $d(x,y)=\sum_{(u,v)\in\pi_{x,y}}d(u,v)$.我们用 $f(s_i,d(u,v))$ 表示对象 $i$ 通过链路 $(u,v)$ 或路径 $\pi_{u,v}$ 的数据传输成本,它度量了在该链路或路径上将对象 $i$ 从结点 $u$ 传输到 $v$ 的资源占用情况.

设有 $M$ 个代理将被放置在一个域集 $P(P\subseteq V,r\in P,|P|=M)$ 上.对于任意结点 $v\in T_r$,我们定义 $v$ 结点的父代理为包含在 $P\backslash\{v\}$ 中的最小祖先代理服务器,即从 $v$ 向树根 $r$ 行进时遇到的在 $P\backslash\{v\}$ 中的第 1 个结点,用 $C(v,P)$ 表示.此外,对于任意 $v$,定义 $v$ 的直接子孙代理集 $D(v)$ 如下:

$$D(v)=\begin{cases}\{u:u\in P\wedge u\in T_v\wedge C(u,P)=C(v,P)\}, & v\notin P\\ \{u:u\in P\wedge C(u,P)=v\}, & v\in P\end{cases}.$$

这样,在给定流量模式的条件下,我们可以得到特定代理放置方案的通信成本和系统吞吐量.首先考虑在网络中不放置代理服务器的情况,此时,所有客户请求都被定向到源服务器.总的数据传输成本由式(2)给出:

$$Cost^t(T_r,\{r\})=\sum_{i=1}^{N}\sum_{v\in T_r}\lambda_{v,i}f(s_i,d(v,r)) \tag{2}$$

式中,$\lambda_{v,i}$ 表示从结点 $v$ 发出的对对象 $i$ 的访问速率.

当在网络中放置一组代理服务器 $P$ 后,如果用 $Cost(T_r,P)$ 表示数据传输成本的减小值,则总的数据传输成本可以表示为 $Cost^t(T_r,P)=Cost^t(T_r,\{r\})-Cost(T_r,P)$.

$$Cost(T_r,P)=\sum_{i=1}^{N}\sum_{v\in P\backslash\{r\}}((1-P_v^b)\lambda_{v,i}^t f(s_i,d(v,r))-\mu_i f(s_i,d(v,C(v,P)))) \tag{3}$$

式(3)中的第 1 项表示代理放置后总的检索开销的减少量,第 2 项表示总的更新成本的增加量.根据放置在结点 $v$ 上的代理的吞吐量,定向到 $v$ 的部分客户请求将直接由该代理提供服务,这些请求不再向源服务器转发,因而使检索成本减小;另外,放置在结点 $v$ 上的代理引入了从其父结点传递更新消息的一致性维护开销.$\lambda_{v,i}^t$ 表示到达结点 $v$ 的对对象 $i$ 的总的检索请求,而项 $(1-P_v^b)\lambda_{v,i}^t$ 则表示代理 $v$ 能够直接在本地处理的总的请求:

$$\lambda_{v,i}^t=\lambda_{v,i}+\sum\nolimits_{u\in B_v}P_u^b\lambda_{u,i}^t \tag{4}$$

式中:$B_v$ 表示结点 $v$ 的子女结点集合;$P_v^b$ 表示结点 $v$ 的阻塞概率.这里,我们扩展阻塞概率的概念:如果 $v$ 是一个代理结点(即 $v\in P$),则 $P_v^b$ 由队列模型计算得到;否则,令 $P_v^b$ 等于 1,这意味着对于那些没有放置代理服务器的结点,到达的所有请求都将被转发到它们的父结点.定义 $\lambda_v^t=\sum_{i=1}^{N}\lambda_{v,i}^t$, $\mu^t=\sum_{i=1}^{N}\mu_i$,它们用来计算 $P_v^b$.

在给定的请求路由机制下,只有当源服务器超载时,客户请求才会被抛弃,因为当代理服务器达到其吞吐能力的极限时,其所在域的路由器会将阻塞的请求向根结点转发.CDN 中总的阻塞请求为

$$Block(T_r,P)=P_r^b\lambda_r^t \tag{5}$$

式中,$\lambda_r^t$ 表示在放置一组代理 $P$ 后定向到 $r$ 的总的检索请求率.显然,如果 CDN 中总的阻塞请求 $Block(T_r,P)$ 达到最小,则系统的吞吐量就达到最大.

在给定的树型拓扑和流量模式下,$Cost^t(T_r,\{r\})$ 是固定不变的,因此,最小化总的数据传输成本 $Cost^t(T_r,P)$ 等价于最大化数据传输成本的减小量 $Cost(T_r,P)$.从而,目标可以修改为寻找一个最优的代理服务器放置方案,使得在整个系统中被阻塞的请求达到最小(即 $\mathrm{Min}_P(Block(T_r,P))$ 的条件下,数据传输成本的减小达到最大(即

$\mathrm{Max}_P Cost(T_r,P)$).我们定义目标函数如下:

$$Obj(T_r,P)=\mathop{\mathrm{Max}}_{P\subseteq V,|P|=M,r\in P}(Cost(T_r,P)-\gamma Block(T_r,P)) \tag{6}$$

式中,$\gamma$是为了在通信流量的减小和代理服务器间的负载均衡之间达到合理折衷而引入的惩罚系数.通过优化目标函数(6),就可以在最大化系统吞吐量的条件下使系统总的通信成本达到最小.

这样,树型拓扑中处理能力受限的代理服务器放置问题(CCSP 问题)可以描述为:给定树型拓扑 $T_r(V,E)$,流量模式和代理服务器容量规划策略,寻找一组规模为 $M$ 的代理服务器 $P(P\subseteq V,r\in P,|P|=M)$,使得式(6)得到满足.

## 2 贪婪算法

通过细致研究,我们发现上述问题无法通过类似于文献[9]给出的动态规划方法来求解,因为代理服务器 $v$ 的阻塞概率只有在以 $v$ 为根的子树中的代理放置方案完全已知的条件下才能导出,而无法用多阶段决策的方式加以计算.因此,我们提出一个贪婪算法求解该问题,如算法 1 所示.

贪婪算法的计算过程如下:首先令 $P=\{r\}$,即将源服务器放置在树根结点 $r$ 上;然后迭代算法且每步只放置一个代理,直到 $M$ 个代理全部放完为止.在每个迭代步中,对于$\forall v\in V\backslash P$,计算试图将 $v$ 加入 $P$ 的目标函数增量;目标增量最大的结点将被选择并加入 $P$ 中.候选结点 $v$ 的目标增量包含 3 部分:(1) 结点 $v$ 本身的贡献;(2) 对 $v$ 的祖先代理数据检索成本减小量的修改,因为在结点 $v$ 放置代理后,定向到这些代理的请求及其阻塞概率将发生变化;(3) 对结点 $v$ 的直接子孙代理结点数据更新成本增加量的修改,因为这些代理的父代理将变成 $v$ 而不再是 $C(v,P)$.利用这种增量计算格式,贪婪算法可以高效运行.

现在我们来推导该算法的增量计算格式.设已有一个代理集 $P(P\subset V,|P|<M,r\in P)$置于网络上,并设有一个候选结点 $v(v\in V\backslash P)$欲加入到 $P$ 中.定义 $v$ 的有序祖先结点集 $A(v),v\notin A(v)$,即 $A(v)$中的元素依次为从 $v$ 向树根结点行进过程中遇到的结点.显然,$A(v)$中的第一个元素为 $v$ 的父结点;且对于 $A(v)$中的任意两个连续元素 $u$ 和 $w$,$w$ 是 $u$ 的父结点.当候选结点 $v$ 加入 $P$ 之后,其成本函数增量$\Delta Cost(T_r,P\cup\{v\})$及目标增量$\Delta Obj(T_r,P\cup\{v\})$的计算格式如下:

Step 1. 计算结点 $v$ 本身的贡献.

$$\Delta Cost(T_r,P\cup\{v\})=\sum_{i=1}^{N}((1-P_v^b)\lambda_{v,i}^t f(s_i,d(v,r))-\mu_i f(s_i,d(v,C(v,P)))).$$

Step 2. 修改 $v$ 的祖先代理数据检索成本减小量.

令 $\Delta\lambda=-(1-P_v^b)\lambda_v^t,\Delta\lambda_i=-(1-P_v^b)\lambda_{v,i}^t$.现依次从 $A(v)$中取出一个结点 $u$,直至取完为止(其中,$A(v)$中的最后一个元素必为树根结点 $r$,对其只计算 $\lambda_r^{t,new}$ 和 $P_r^{b,new}$,而不作其他计算).注意,带上标 $new$ 的变量对应于 $v$ 已加入 $P$ 的情况.令 $\lambda_u^{t,new}=\lambda_u^t+\Delta\lambda,\lambda_{u,i}^{t,new}=\lambda_{u,i}^t+\Delta\lambda_i$.如果 $u\in P$,则进一步根据 $\lambda_u^{t,new}$ 计算 $P_u^{b,new}$,并且计算:

$$\Delta\lambda\leftarrow\Delta\lambda-(1-P_u^{b,new})\lambda_u^{t,new}+(1-P_u^b)\lambda_u^t,$$

$$\Delta\lambda_i\leftarrow\Delta\lambda_i-(1-P_u^{b,new})\lambda_{u,i}^{t,new}+(1-P_u^b)\lambda_{u,i}^t,$$

$$\Delta Cost(T_r,P\cup\{v\})\leftarrow\Delta Cost(T_r,P\cup\{v\})+\sum_{i=1}^{N}(((1-P_u^{b,new})\lambda_{u,i}^{t,new}-(1-P_u^b)\lambda_{u,i}^t)f(s_i,d(u,r))).$$

Step 3. 修改结点 $v$ 的直接子孙代理结点数据更新成本增加量.

设在 $v$ 加入 $P$ 之前$|D(v)|=t$,则

$$\Delta Cost(T_r,P\cup\{v\})\leftarrow\Delta Cost(T_r,P\cup\{v\})+t\sum_{i=1}^{N}\mu_i f(s_i,d(v,C(v,P))).$$

这是因为,当在结点 $v$ 放置一个代理时,$v$ 将取代 $C(v,P)$沿代理层次结构向 $D(v)$发出所有更新活动.这样,就消除了更新消息在路径$\pi_{v,C(v,P)}$上的重复传输,从而减小更新成本.对于图 1 给出的例子,假定我们通过运行上述贪婪算法,已经在结点 1 放置源服务器并在结点 10 和结点 15 放置代理服务器.显然,这两个代理的更新消息是直接从结点 1 传播下来的.现在,假定我们打算在结点 6 放置一个代理.由于 $D(6)=\{10,15\}$,更新消息将从结点 6 而不是从结点 1 传播下来.这样,就避免了更新消息在$\pi_{1,6}$上的重复传输.

Step 4. 计算 $Block(T_r,P\cup\{v\})$和$\Delta Obj(T_r,P\cup\{v\})$.

$$Block(T_r, P\cup\{v\}) = P_r^{b,new}\lambda_r^{t,new},$$

$$\Delta Obj(T_r,P\cup\{v\})=\Delta Cost(T_r,P\cup\{v\})-\gamma(Block(T_r,P\cup\{v\})-Block(T_r,P)).$$

易证,上述贪婪算法的复杂性为 $O(ML)$,其中,$L$ 是 $T_r$ 的路径长度,定义为 $T_r$ 上各结点的祖先结点数之和.

算法 **1**. 求解 CCSP 问题的贪婪算法.

1: set $P=\{r\}$, $Cost(T_r,P)=0$, $\lambda_{v,i}^t=\sum_{u\in T_v}\lambda_{u,i}$, $\lambda_v^t=\sum_{i=1}^N\lambda_{v,i}^t$, for $\forall v\in T_r,\forall i(1\le i\le N)$; compute $Block(T_r,P)$, $Obj(T_r,P)$;

2: **while** ($|P|\le M$) {

3: **for** $\forall v\in V\backslash P$, compute $\Delta Cost(T_r,P\cup\{v\})$, $Block(T_r,P\cup\{v\})$ and $\Delta Obj(T_r,P\cup\{v\})$;

4: find $v\in V\backslash P$ such that $\Delta Obj(T_r,P\cup\{v\})$ is maximized;
$P\leftarrow P\cup\{v\}$, $Cost(T_r,P)\leftarrow Cost(T_r,P)+\Delta Cost(T_r,P)$, $Obj(T_r,P)\leftarrow Obj(T_r,P)+\Delta Obj(T_r,P)$;

5: **for** $\forall u\in A(v)$, update $\lambda_u^t$, $\lambda_{u,i}^t$ in order;

6: **for** $\forall u\in D(v)$, $C(u,P)\leftarrow v$;}

## 3 性能评价

我们通过仿真实验评价本文提出的代理放置算法的性能.此外,我们还要研究惩罚系数、通信量大小和服务器处理能力对代理放置决策的影响.为了比较算法的性能,我们以随机算法和动态规划算法(dynamic programming,简称 DP)作为比较基准.随机算法随机地从 $V$ 中选择 $M$ 个代理 $P(r\in P)$,完全忽略输入信息.为了改善性能,我们将算法运行 100 次,并选择最好的解作为随机算法的最终结果.DP 算法忽略了代理服务器的吞吐量,给出具有最小网络通信成本的最优代理放置策略.其实现是将各候选代理结点的 $\bar{x}$ 设为 0,这意味着各代理具有无限强大的处理能力,每个请求能在瞬间得到服务.因此,我们称 DP 算法的解为忽略吞吐量的最优放置策略.在得到忽略吞吐量的最优放置策略后,通过将各代理的 $\bar{x}$ 重设为正常值,我们就可以获得该放置策略在代理服务器处理能力受限情况下的性能.

如目标函数(6)所示:对于 CCSP,我们不仅关心通信流量的减小量,而且关心整个系统的吞吐量.因此,我们用两个规范化的性能指标评价放置算法的性能,即由放置 $M$ 个代理的忽略吞吐量的最优解规范化的网络成本减小量和由仅放置源服务器在树根结点的结果规范化的阻塞请求率.一个解的网络成本减小量越大,阻塞请求率越小,则该解就越优越.

与文献[9]的做法类似,我们使用模拟生成的树型拓扑和流量模式评价算法性能.我们认为,尽管在放置实验中没有使用 Web 日志记录,但我们的仿真实验仍然具有良好的代表性,其原因如下:(1) 通过比较文献[3,5,7,8]中使用模拟流量模式和 Web 日志记录数据的仿真实验结果,我们发现这些实验结果具有相同的规律;(2) 即使在使用 Web 日志的实验[3,5,7,8]中,其网络拓扑或者是高度简化的,或者是模拟生成的,表明这些实验环境仍然不能完全符合真实的网络状况;(3) 流量模式的研究表明[21,22],访问流行度和对象大小的分布严格服从某种规律,这意味着这些分布特征可以被准确仿真且其仿真结果具有良好的代表性.

### 3.1 仿真实验环境

树型拓扑通过输入一对参数即结点总数(treeSize)和树结点的最大度数(treeDegree),以宽度优先的方式生成.不失一般性,树的每条边关联一个在(0,1)之间随机分布的距离,因为所有的边距都可以成比例地映射到该区间.树中的每个结点 $v$ 代表一个 AS 域,并关联一个从该域中发送的检索请求率 $\lambda_v$.树根结点 $r$ 还关联一个均匀分布在(minWtRate,maxWtRate)之间的更新请求率 $\mu$.源服务器持有 $N$ 个 Web 对象,对象的访问流行度服从 Zipf-like 分布[21,22],其中检索请求的分布参数为 $\theta_r$,而更新请求的参数为 $\theta_w$,即如果一个 Web 对象的访问流行度的排序为 $i$,则其访问概率与 $i^{-\theta}$ 成正比.我们假设 Web 对象检索请求和更新请求具有不同的流行度.

考虑两种流量模型:随机模型和一致模型.在随机模型中,首先为每个结点 $v$ 均匀地从(minRdRate, maxRdRate)生成 $\lambda_v$,对对象 $i$ 的总的访问由 $\lambda^i=(\sum_{v\in V}\lambda_v)i^{-\theta_r}/\sum_{i=1}^N i^{-\theta_r}$ 计算.结点 $v$ 对对象 $i$ 的访问 $\lambda_{v,i}$ 在 $V$ 上均

匀地生成,且满足 $\sum_{v\in V}\lambda_{v,i}=\lambda^i$.最后,修改$\lambda_v$ 为 $\lambda_v=\sum_{i=1}^{N}\lambda_{v,i}$.在一致模型中,为便于比较,保持$\lambda_v$ 的值与随机模型相同,但令$\lambda_{v,i}$为 $\lambda_{v,i}=\lambda_v i^{-\theta_r}/\sum_{i=1}^{N} i^{-\theta_r}$.其中,随机模型是默认模型.

每个对象 $i$ 赋以一个大小 $s_i$,已有研究表明,其分布为重尾(heavy-tailed)分布[22],即存在相对少数极大的孤立点坐落于分布的尾端,但其对总的通信量有很大的贡献.我们假设重尾对象大小分布服从双指数 Pareto 分布(doubly exponential Pareto distribution)[22],其累积分布函数为

$$F(s)=1-(s_0/s)^{\beta},\ \beta>0, s\geq s_0>0 \tag{7}$$

式中:$\beta$称为尾指数(tail index);$s_0$ 为在重尾分布中随机对象大小的最小可能值.服从重尾分布的对象大小的抽样相当容易,这里不再赘述.为简单起见,我们用链路或路径的距离与在其上传输的数据量的乘积函数表示数据传输成本,即令 $f(s_i,d(u,v))=s_i\times d(u,v)$.

此外,树中的每个结点 $v$ 还关联另外两个值,即 $\bar{x}$ 和 $K$,其意义在于:如果 $v$ 被选中作为候选代理服务器,则 $v$ 的处理能力恰好用这两个参数来表征.$\bar{x}$ 和 $K$ 分别均匀分布在(minSvTime,maxSvTime)之间和(minJobLimit, maxJobLimit)之间.我们用随机策略和一致策略来模拟 CDN 服务器的处理能力.在一致策略中,所有的 CDN 服务器都是同构的,即具有相同的处理能力;而在随机策略中,CDN 服务器是异构的.默认策略是随机策略.仿真实验采用的其他默认参数设置详见表 1.

**Table 1** Default settings for simulation parameters

表 **1** 仿真实验参数的默认设置

| Parameter | Default setting | Parameter | Default setting | Parameter | Default setting |
|---|---|---|---|---|---|
| treeSize | 600 | $N$ | 1000 | treeDegree | 6 |
| minWtRate | 1 | $\gamma$ | 10 | minSvTime | 0.000 1 |
| maxWtRate | 80 | $\beta$ | 1.2 | maxSvTime | 0.01 |
| minRdRate | 1 | $s_0$ | 4 | minJobLimit | 50 |
| maxRdRate | 80 | $\theta_r$, $\theta_w$ | 0.8, 0.4 | maxJobLimit | 300 |

### 3.2 仿真实验结果

#### 3.2.1 贪婪算法的性能

通过将本文提出的贪婪算法与忽略代理吞吐能力的 DP 算法和随机算法进行比较,研究代理放置算法的性能.我们分别在 200,600 和 1 000 结点的树型网络上评价算法的性能,所有其他参数均设为默认值.在仿真实验中,我们逐步增大代理的放置数量,直至在网络上放置 $M$=0.3×treeSize 个代理为止.图 2 给出了 600 结点网络的仿真实验结果(其他网络的仿真实验结果类似).注意,在图 2 中,“optimal”对应于 $\bar{x}$ 设为 0 时 DP 算法导出的网络成本减小量的上限值,而“DP”则对应于在“optimal”放置策略中将各代理的 $\bar{x}$ 重设为正常值后得到的性能指标.

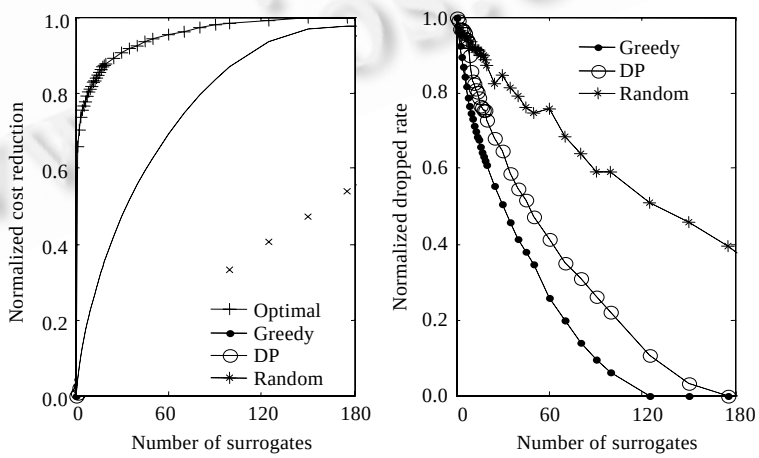


Fig.2 Normalized traffic reduction and blocked request rate

图 2 规范化的网络流量减小量及阻塞请求率

从图2可以得出如下几个规律:首先,贪婪算法远优于忽略吞吐量的“DP”放置策略,这表明,在代理放置决策中考虑代理服务器处理能力约束是必要的.相应于 200,600 和 1 000 结点网络的仿真实验结果,贪婪算法给出的网络成本减小量最优值分别比 DP 解优 3.08,6.93,8.85 倍,平均比 DP 解优 1.50,2.03,2.92 倍.当在网络上放置更多代理时,网络成本减小量的差别逐渐减小;另一方面,随着代理数量的增加,DP 算法和贪婪算法在阻塞请求率上的性能差别日益显著.这是因为:当代理放置数量很少时,两种放置方案都能充分利用各代理的处理能力,因而二者的阻塞请求率大致相同;当代理放置数量增加时,某些代理不再超载,DP 算法由于没有将系统吞吐量作为额外的优化目标,一般给出具有更多阻塞请求的放置方案;其次,即使我们以 100 次运行的最佳解作为随机算法的终解,随机算法的性能始终表现最差,尤其是在系统放置较多代理的情况下.相应于 200,600 和 1 000 结点网络的实验结果,贪婪算法在网络成本减小量上平均比随机算法优 1.95,3.35,4.04 倍;第三,就网络成本减小量而言,贪婪算法与“optimal”上限值之间存在较大差距.这种性能差距并不是源于贪婪算法性能不佳,而是由于当代理的处理能力受限后,大部分的请求无法在本地得到处理而必须转发给源服务器,从而增加了网络成本.随后可见,这种性能差距可以通过配置更强大的代理服务器或配置更多的代理而减小.

3.2.2 罚系数对代理放置的影响

在第 3.2.1 节中,我们通过将罚系数$\gamma$设为默认值研究了贪婪算法的性能.本节研究罚系数对代理放置决策的影响.我们在600结点的树型网络上评价罚系数的影响,实验中,令$\gamma$在0~500之间变化,而把其他参数均设为默认值.实验结果如图3所示.

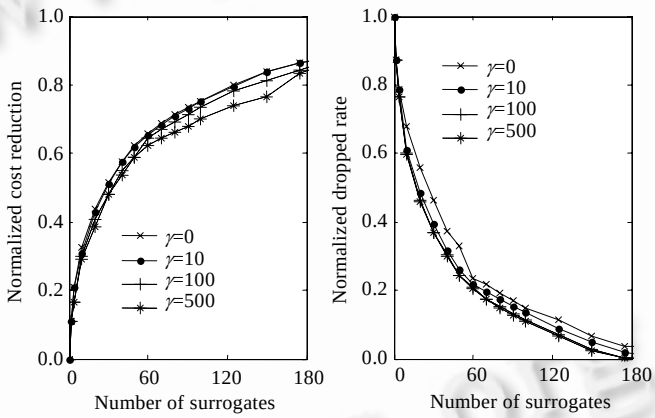


Fig.3 Impact of the penalty coefficient $\gamma$

图3 罚系数$\gamma$对代理放置的影响

图3表明:数值较大的$\gamma$将导致阻塞请求的减小,但会使网络通信量略微增加.然而,这种影响并不显著,当$\gamma$从0变化到500时,$\gamma$对网络成本减小量的影响在最坏情况下不超过20%,平均为7.3%,而其对阻塞请求的影响一般也在26%范围之内.使用大的$\gamma$值可使阻塞请求率更快地随代理数量的增加下降到0.但为了获得这种边际增量,网络成本将稍有增加.一般而言,贪婪算法对罚系数并不十分敏感,其原因是:为了在最大化网络成本减小量和最小化阻塞请求率之间找到合理折衷,贪婪算法首先适当放宽阻塞请求上的约束条件,选择远离源服务器的结点放置代理,以显著减小网络成本.大的$\gamma$值促使算法选择能够最大化系统吞吐量而又尽可能远离源服务器的结点放置代理,因而,$\gamma$的影响并没有我们预想的那么显著.这个性质很重要,因为它间接证明了贪婪算法具有逼近最优解的能力.从图中还可以看出,默认的$\gamma$值是一个合适的选择.

3.2.3 通信流量大小对代理放置的影响

本节研究通信量大小对代理放置决策的影响.实验在一个 600 结点的树型网络上进行,通过逐步增大各结点的检索请求率和更新请求率,评价通信流量对代理放置决策的影响.令检索请求率和更新请求率分别从[1,50],[1,80],[10,120]和[30,150]抽样,而将其他参数设为相应的默认值.由于同一网络中的通信流量大小变化

时,网络成本和阻塞请求率也会随之变化.因此为便于比较,我们对性能指标进行规范化处理,使得在各种实验条件下贪婪算法得到的在网络中放置 180 个代理的网络成本减小量总是规范化为 1,以及在网络中仅放置一个代理的阻塞请求率总是规范化为 1.图 4 给出了贪婪算法的计算结果.

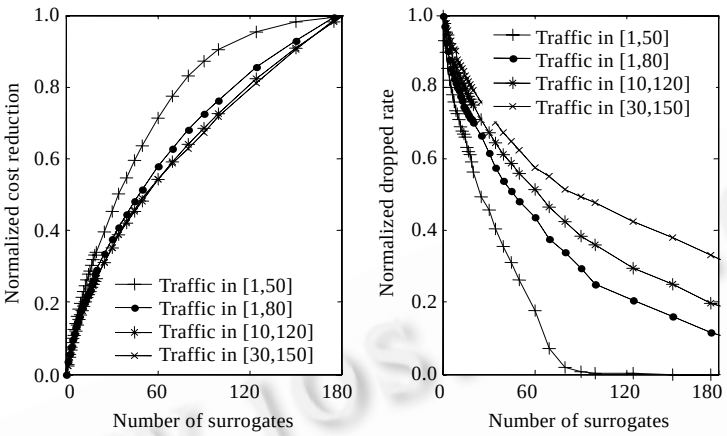


Fig.4 Impact of traffic volume on surrogate placement

图 4 通信流量大小对代理放置的影响

图 4 表明,当网络流量相对较小时,网络传输成本和阻塞的请求均会随代理的增加而迅速减小,这是因为对于小的通信流量,增加一个代理可以吸收相当数量的请求而显著改善系统性能.当网络流量增加时,为了获得相同的规范化性能,必须配置更多的代理服务器.此外,我们还通过实验比较了随机流量模型和一致模型对性能的影响,但二者的差别很小.

3.2.4 服务器处理能力对代理放置的影响

本节研究服务器处理能力对代理放置决策的影响.实验仍然使用 600 个结点的树型网络.实验中,我们令各候选代理服务器的 $\bar{x}$ 值在 0.01~0 之间变化,而将 $K$ 和其他参数设为默认值.主要实验结果如图 5 所示,图中我们用“MST”表示 $\bar{x}$ .

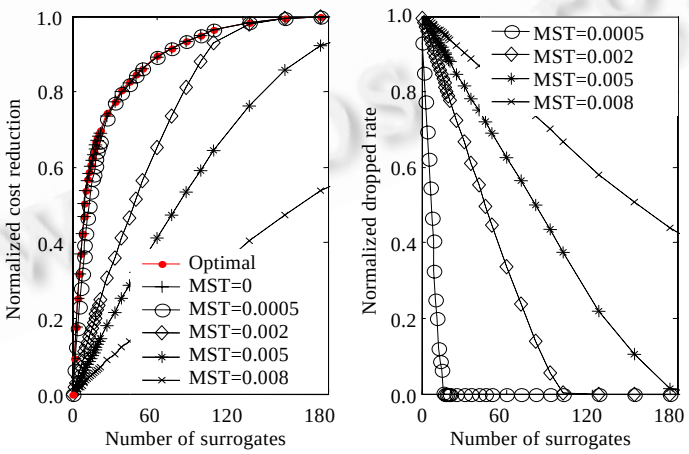


Fig.5 Impact of server power on surrogate placement

图 5 服务器处理能力对代理放置的影响

在图 5 中,“optimal”曲线上有一个明显的弯肘,这意味着性能增量随代理数量的增加而迅速减小.为数不多的代理(树规模的 5%~10%)吸收绝大部分流量减小量(77.3%~89.1%).接下来我们研究在各候选代理的处理能

力可认为无限强大(即 $\bar{x}$ 设为 0)的极端条件下贪婪算法的性能.此时,系统不阻塞任何客户请求,问题的优化目标简化为最大化通信成本减小量.从图 5 可见,贪婪算法的性能十分逼近最优解.在最坏情况下,贪婪算法的网络成本减小量仅比最优解差 1.19%,而在平均情况下仅差 0.33%.图 5 还表明:当服务器的处理能力规划得很强大时(即令 $\bar{x}$ 取很小值,如图 5 中 MST=0.0005 的情况),随着代理服务器的增加,阻塞的请求急剧递减至 0,且网络流量的减小量越来越逼近最优解.因此,当计划在网络上配置非常强大的 CDN 服务器时,以网络为中心的观点求解代理放置问题是合适的.当服务器处理能力下降时,各代理所能处理的请求数量有限,大量的请求必须转发给其父代理,此时,网络成本和系统吞吐量均得不到优化,除非配置更多的代理弥补这种性能损失.可以看出,网络成本减小量的相应曲线比“optimal”曲线平缓得多.这意味着在网络上配置极少量的代理(如 5~10 个)就可以显著减小网络流量(31.7%~53.7%)的好处不复存在.此外,通过比较同构和异构服务器规划策略,其性能差别很小,原因是二者在系统中具有相同的总处理能力.

## 4 结束语

本文研究树型 CDN 中考虑服务器处理能力或其负载分布的代理放置问题,目标是要找出最优的代理放置策略,使得在最大化 CDN 吞吐能力的条件下,使得网络通信流量达到最小.为了解决该问题,我们提出了一种高效的贪婪算法,并在仿真实验中与随机解和仅考虑数据传输成本的动态规划算法的最优解进行了比较.此外,我们还研究了罚系数、通信流量大小和服务器处理能力对代理放置决策的影响.

仿真实验结果表明:贪婪算法具有逼近最优解的能力,次优的 CCSP 放置策略的性能通常比忽略服务器吞吐量的最优代理放置策略优 1.5~3.0 倍,从而揭示了在代理放置决策中考虑代理服务器之间的负载分布或服务器性能瓶颈的必要性,尤其是当 CDN 服务器处理能力因某种原因受到限制的时候更是如此.此外,本文提出的贪婪算法比随机算法优出很多(一般优 2.0~4.0 倍).随着通信流量的增加或服务器处理能力的降低,为了在 CDN 中获得高性能或吞吐能力,必须配置更多的代理服务器.如果各个域中的通信流量大致成比例地增加,则采用增量或分期配置的代理放置方案(如本文给出的贪婪算法)是合适的.

总之,本文的主要工作在于:(1) 给出了在代理放置决策中考虑服务器处理能力的代理放置问题的形式化描述;(2) 提出了既减小网络通信流量又同时提高系统吞吐量的代理放置策略;(3) 给出了评价 CCSP 代理放置算法的仿真实验,其中不但考虑了 Web 对象的流行度,而且考虑了对象大小的重尾分布.

尽管我们在 CCSP 领域取得了一些有意义的研究成果,然而,如何将这项工作推广到一般网络拓扑仍然是一项艰巨的任务.在一般网络条件下,CDN 复杂的请求路由机制使得难以为该问题建立准确的数学模型.此外,本文给出的形式化框架在 P2P CDN 环境及 Ad Hoc 网络环境下具有更大的吸引力,因为 P2P CDN 可采用树型结构组织动态拓扑,而 Ad Hoc 网络中的代理服务器显然受到处理能力的约束.

**References**:

[1] Day M, Cain B, Tomlinson G, Rzewski P. A model for content internetworking (CDI). RFC 3466, Network Working Group, 2003.

[2] Lazar I, Terrill W. Exploring content delivery networking. IEEE IT Professional, 2001,3(4):47–49.

[3] Cronin E, Jamin S, Jin C, Kurc AR, Raz D, Shavitt Y. Constrained mirror placement on the Internet. IEEE Journal on Selected Areas in Communications, 2002,20(7):1369–1381.

[4] Jia X, Li D, Hu X, Du D. Placement of read-write web proxies in the Internet. In: Lanus M, ed. Proc. of the IEEE ICDCS 2001. Los Alamitos: IEEE Press, 2001. 687–690.

[5] Krishnan P, Raz D, Shavitt Y. The cache location problem. IEEE/ACM Trans. on Networking, 2000,8(5):568–582.

[6] Li B, Golin MJ, Italiano GF, Deng X, Sohraby K. On the optimal placement of web proxies in the Internet. In: Choudhury AK, Shroff N, eds. Proc. of the IEEE INFOCOM’99. Los Alamitos: IEEE Press, 1999. 1282–1290.

[7] Li Y, Liu MT. Optimization of performance gain in content distribution networks with server replicas. In: Helal S, Oie Y, Chang C, Murai J, eds. Proc. of the 2003 Symp. Applications and the Internet. Los Alamitos: IEEE Press, 2003. 182–189.

[8] Qiu L, Padmanabhan VN, Voelker GM. On the placement of web server replicas. In: Bauer F, Cavendish D, eds. Proc. of the IEEE INFOCOM 2001. Los Alamitos: IEEE Press, 2001. 1587−1596.

[9] Xu J, Li B, Lee DL. Placement problems for transparent data replication proxy services. IEEE Journal on Selected Areas in Communications, 2002,20(7):1383−1398.

[10] Cao J, Andersson M, Nyberg C, Kihl M. Web server performance modeling using an M/G/1/K*PS queue. In: Lorenz P, ed. Proc. of the 10th Int'l Conf. Telecommunications. Los Alamitos: IEEE Press, 2003. 1501−1506.

[11] Dowdy LW, Foster DV. Comparative models of the file assignment problem. ACM Computer Surveys, 1982,14(2):287−313.

[12] Kurose JF, Simha R. A microeconomic approach to optimal resource allocation in distributed computer systems. IEEE Trans. on Computers, 1989,38(5):705−717.

[13] Rabinovich M, Rabinovich I, Rajaraman R, Aggarwal A. A dynamic replication and migration protocol for an Internet hosting service. In: Gouda MG, ed. Proc. of the IEEE ICDCS'99. Los Alamitos: IEEE Press, 1999. 101−113.

[14] Venkataramani A, Weidmann P, Dahlin M. Bandwidth constrained placement in a WAN. In: Kshemkalyani A, Shavit N, eds. Proc. of the 20th Annual ACM Symp. Principles of Distributed Computing. New York: ACM Press, 2001. 134−143.

[15] Chudak FA, Shmoys DB. Improved approximation algorithms for a capacitated facility location problem. In: Tarjan RE, Warnow T, eds. Proc. of the 10th Annual ACM-SIAM Symp. Discrete Algorithms. New York: ACM Press, 1999. 875−876.

[16] Hunt GDH, Goldszmidt GS, King RP, Mukherjee R. Network dispatcher: A connection router for scalable Internet services. Computer Networks and ISDN Systems, 1998,30(1-7):347−357.

[17] Heddaya A, Mirdad S. WebWave: Globally load balanced fully distributed caching of hot published documents. In: Tripathi S, ed. Proc. of the IEEE ICDCS'97. Los Alamitos: IEEE Press, 1997. 160−168.

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

[19] Wolfson O, Milo A. The multicast policy and its relationship to replicated data placement. ACM Trans. on Database Systems, 1991,16(1):181−205.

[20] Schiely M, Renfer L, Felber P. Self-Organization in cooperative content distribution networks. In: Shvartsman A, ed. Proc of the IEEE NCA 2005. Los Alamitos: IEEE Press, 2005. 109−118.

[21] Breslau L, Cao P, Fan L, Phillips G, Shenker S. Web caching and zipf-like distributions: Evidence and implications. In: Choudhury AK, Shroff N, eds. Proc. of the IEEE INFOCOM'99. Los Alamitos: IEEE Press, 1999. 126−134.

[22] Mahanti A, Williamson C, Eager D. Traffic analysis of a Web proxy caching hierarchy. IEEE Network, 2000,14(3):16−23.

陈益峰(1974－),男,福建漳平人,博士,讲师,主要研究领域为分布并行计算,岩土工程,水电工程.

何炎祥(1952－),男,教授,博士生导师,CCF高级会员,主要研究领域为分布并行处理,数据开采,多 Agent 系统,网格计算,软件工程.

曹建农(1960－),男,教授,博士生导师,CCF高级会员,主要研究领域为分布并行计算,网络互联,移动及无线计算,容错,分布式软件体系结构.