ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.7, July 2010, pp.1524−1535
doi: 10.3724/SP.J.1001.2010.03600

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

分布式缓存系统中一种优化缓存部署的图算法*

李文中[1,2+], 陈道蓄[1,2], 陆桑璐[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)

Graph-Based Optimal Cache Deployment Algorithm for Distributed Caching Systems

LI Wen-Zhong[1,2+], CHEN Dao-Xu[1,2], LU Sang-Lu[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: E-mail: lwz@dislab.nju.edu.cn

Li WZ, Chen DX, Lu SL. Graph-Based optimal cache deployment algorithm for distributed caching systems. *Journal of Software*, 2010,21(7):1524−1535. http://www.jos.org.cn/1000-9825/3600.htm

Abstract: Data caching has emerged as an effective way to reduce network traffic, alleviate server load and accelerate information access. Deploying a group of distributed caching nodes to cooperate with each other in serving client requests will further improve the system performance. One of the important issues in distributed caching system is coordinating cache placement to achieve access cost minimization. In this paper, a theoretical model is introduced to analyze the access cost of placing a set of data objects in distributed caching systems. In this model, the cache placement problem is formulated as an optimization problem. A graph-based algorithm is proposed to solve the problem. The algorithm applies a modified Dijkstra's algorithm to look for the shortest path in the access cost graph, which corresponds to an optimal cache deployment for the system. The correctness of the graph-based algorithm is proved theoretically and its performance is evaluated by simulations. Experimental results show that the graph-based algorithm outperforms most existing distributed caching schemes.

Key words: distributed caching system; cache placement; graph-based algorithm

摘 要: 数据缓存技术可以有效地减少网络拥塞,减轻服务器负载,加快信息访问速度.通过部署一组地域分布的缓存节点相互协作处理用户请求,可以进一步提高系统性能.在分布式缓存系统中,一个值得关注的问题是优化缓存的放置,使访问开销最小化.首先建立了一个理论模型来分析缓存副本放置对系统访问开销的影响.基于这个模型,缓存放置问题可以形式化地描述成一个最优化问题,提出了一种图算法来解决该问题.图算法使用修改的 Dijkstra 算法在访问代价图中寻找一条最短路径,该路径对应一种最优的缓存部署.理论上证明了图算法的正确性,并使用仿真实验对其性能进行评估.实验结果表明,图算法的性能优于大部分现有的分布式缓存机制.

* Supported by the National Natural Science Foundation of China under Grant Nos.60803111, 90718031, 60721002 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2006AA01Z199 (国家高技术研究发展计划(863)); the National Basic Research Program of China under Grant No.2009CB320705 (国家重点基础研究发展计划(973)); the Jiangsu Provincial Natural Science Foundation of China under Grant No.BK2009100 (江苏省自然科学基金)

Received 2008-01-06; Revised 2008-02-20; Accepted 2009-03-05

关键词: 分布式缓存系统;缓存放置;图算法
中图法分类号: TP316 **文献标识码:** A

随着 Internet 的迅速发展,网络用户的不断增多,网络应用对于带宽和服务器性能的要求也越来越高.数据缓存技术将经常访问的数据缓存到靠近用户的网络节点上,可以有效缓解网络拥塞,减轻服务器负载,并减少网络访问延迟[1−3].为了充分发挥缓存的效率,人们提出了协同缓存的思想[4−9]:通过一组地域上分布的缓存节点相互连接,共享缓存内容,形成一个分布式缓存系统.当一个节点请求的数据不在本地缓存时,它可以从附近的缓存节点获得该数据,而不必每次都向服务器发送请求.协同缓存与单机缓存相比,可以进一步提高缓存利用率,加快信息访问的速度.

在分布式协同缓存系统中,一个值得研究的问题是如何优化缓存数据的分布.在传统的缓存机制中,节点各自独立地进行缓存决策,分别缓存本地访问最频繁的数据.这种单机决策的方法会造成缓存数据冗余,缓存系统的效率低下[10].理想的做法是,各个节点协同地进行缓存决策,充分发挥缓存效率,使系统的整体访问开销降到最低.

本文研究优化缓存部署问题,提出了一种分布式缓存系统最低访问开销的图算法.本文的主要贡献可以归纳如下:(1) 建立了一个理论模型来分析缓存副本放置对系统访问开销的影响,并将缓存放置问题形式化地描述成一个最优化问题;(2) 提出了一种求最优缓存部署的图算法,它根据用户请求和网络距离等信息构建一个代价图,并使用 Dijkstra 算法在代价图中寻求一条最短路径,该最短路径代表一种最优的缓存部署方案;(3) 从理论上证明了图算法的正确性,并使用仿真实验验证算法的有效性.实验结果表明,与其他现有的分布式缓存算法相比,图算法的性能有明显提高.

1 相关工作

在协同缓存的早期研究工作中,Malpani 等人[4]提出,通过一组代理缓存服务器之间共享缓存内容,可以获得比单机缓存更好的性能.为了实现节点之间缓存数据的共享和协同,人们提出了许多支持协同缓存的协议.ICP(Internet cache protocol)[11]协议采用层次式结构来进行缓存数据的查找和定位.CARP(cache array routing protocol)[12]协议使用哈希表来定位缓存节点,可以减少节点之间相互通信的开销.WCCP(Web cache coordination protocol)[13]协议则在路由器上可以实现对用户访问透明的重定向.Summary Cache[14]协议则使用 Bloom filter 来建立和维护其他节点的缓存目录,可以快速地定位附近的缓存数据.

CRISP(caching and replication for Internet service performance)[1]系统提出利用缓存目录来实现协同缓存.它使用一个目录服务器来维护全局的缓存数据的元信息,可以通过查询缓存目录来定位最近的缓存节点.这种集中式目录的机制效率很高,但是目录服务器也容易成为瓶颈.Zhang 等人提出了适应性缓存的思想[3],在这种系统中,Web 服务器与缓存服务器被组织成多个相互交迭的组播组,使用组播技术转发用户请求和回传数据,根据用户的访问情况自动调整缓存数据的分布,用户访问频繁的数据能够沿着分布树从源服务器向客户端靠近.通过组管理和维护策略自动调整组播组的大小,从而达到均衡负载的目的.Bhattacharjee 等人[6]则提出了自组织的网络缓存机制,它可以实现于无结构网络,具有更好的扩展性.各节点可以根据邻近节点的信息进行缓存放置和替换决策.在自组织网络中,如何优化缓存数据的分布是一个很有挑战性的问题.

Korupolu 等人[5]研究了协同缓存放置和替换问题,他们针对层次结构的网络拓扑,以优化节点之间的访问延时为目标,提出了一个分析模型,把最优缓存放置问题转化为一个最小代价流问题,并给出了最优解法.然而,该算法的时间复杂度较高,节点的通信开销比较大,也不适用于非结构的网络缓存系统.为了避免缓存副本产生冗余,提高缓存命中率,Ramaswamy 等人[15]提出失效期(expiration-age,简称 EA)缓存部署策略.EA 算法的基本思想是,将一组网络节点的缓存空间看成一个合成的空间,对每个缓存对象引入一个 EA 值作为决策依据,EA 值定义为数据对象最后一次在缓存中命中到它被替换出缓存所经历的时间,表征该对象在缓存空间内的存活期.当一个节点接收到一个数据对象时,是否缓存该对象由其他缓存有该对象的节点共同决定.这种方法可以减少节

点组内的缓存冗余度,同时保证缓存的数据具有尽可能长的存活期.Modulo[6]是针对自组织网络提出的一种缓存放置和替换策略,其目标是减少网络拥塞和访问延迟.它提出了缓存半径的概念,在缓存半径范围内,一个数据对象最多会被缓存一次.对于给定的缓存半径,使用一个称为 Modulo 的哈希函数将数据对象映射到缓存节点上,使得一个节点只缓存一组特定的数据,同一半径内的节点之间缓存的数据不会相同.它明显降低了缓存冗余度,从某种程度上改善了系统性能.但是,这一策略没有考虑到数据的热门程度和网络距离等因素的影响.

Che 等人[16]建立了一个分层的协同缓存模型,提出了一种 Filter 算法,把缓存层次看作低通滤子,只有访问频率达到某个阈值时才会被缓存,低于阈值的对象会被过滤到下一个缓存层次上.这种算法需要知道全局访问频率信息,在实际情况中很难应用.Laoutaris 等人[17]提出了若干层次式缓存放置算法:Prob,LCD(leave copy down)和 MCD(move copy down).他们的研究表明,LCD 算法的实现很简单,但是可以取得和 Filter 算法差不多的效果.Tang 等人[18]给出了在 en-route 缓存中优化分布缓存副本的方法.En-route 缓存使用固定路由,访问节点和数据源之间形成树状结构,在这种情况下,缓存放置问题被证明可以化为一个最优化问题,并使用动态规划来求解.Li 等人[19]研究了多版本媒体文件的最优部署问题,基于多媒体编码转换的代理缓存系统,提出了动态规划解法.Alan 等人[20]则针对在线流媒体应用,研究了代理服务器和客户端缓存的协同副本放置技术,把最优缓存部署问题转化为一个多约束线性规划问题来求解.然而,大部分研究工作以层次式或树状的网络为背景,在这种结构化的网络中,可以计算出最优缓存部署方案,但这并不适用于广域、无结构的分布式协同缓存系统.

2 分布式协同缓存系统

本文研究在广域分布的协同缓存系统中如何优化缓存副本部署.图 1 显示了一个分布式缓存系统,在这个系统中,一组广域的缓存节点相互连接,形成一个无结构的覆盖网络.节点之间相互共享缓存数据,协同处理用户请求.当一个节点产生一个数据请求时,它首先检查本地缓存是否有请求的数据,如果缓存有该数据,则直接返回给用户;否则,用户请求会被转发给其他缓存节点.用户请求可能会经过若干次转发,直至到达一个缓存有请求数据的节点,或者到达源服务器.该数据会沿着转发的路径,回传给用户.在该路径上的节点都可以根据某种原则,决定是否需要缓存该数据的副本.

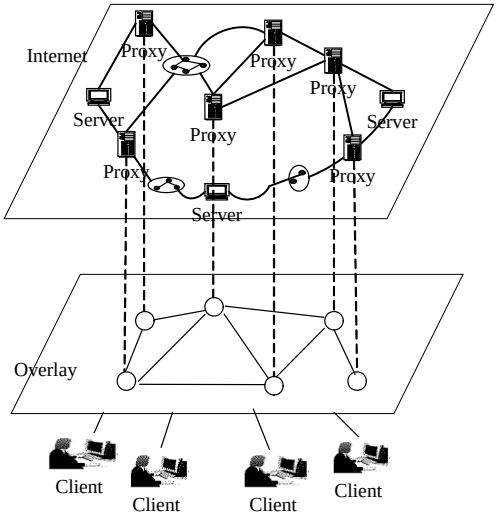

Fig.1 Distributed caching system

图 1 分布式缓存系统

为了简化分析,我们不考虑缓存的一致性问题,假设缓存的数据都是有效的.另外,我们假设系统可以定位最近的缓存副本(例如可以使用前面介绍的缓存目录或 Summary Cache 协议来实现),这样,用户的数据访问请求总是从最近的缓存副本得到响应.我们关注优化缓存副本部署问题:给定一组协同缓存节点,已知节点之间的

网络距离以及每个节点对于一组数据对象的请求率,决定如何分布缓存数据对象的副本,使得系统的总访问开销最小.

3 问题描述

在广域分布的、无结构的网络中,要实现全局最优缓存部署是很困难的.我们采用一种局部优化的方法:针对每次数据访问的路径所经过的节点来进行缓存数据的优化部署.图 2 显示了分布式协同缓存中的一条数据访问路径.节点 u 需要访问数据对象 O,由于本地没有缓存所请求的对象,所以用户请求被转发,直至到达一个缓存有该请求对象的节点 v(或源服务器).节点 v 响应用户请求,并把数据 O 沿原路回传给节点 u.假设从 u 到 v 的路径上依次经过 n 个节点,将节点 u 编号为 0,节点 v 编号为 n,路径上的其他节点依次编号为 1,2,…,n−1.我们的目标是优化该路径上的缓存副本的部署.

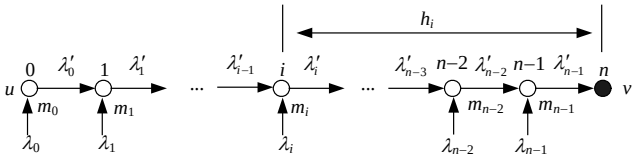

Fig.2 Data access in distributed caching system

图 2 分布式缓存系统中的数据访问

假设 λ_i' 表示在一段时间内节点 i 观察到的访问对象 O 的请求次数(称为请求到达率),这些请求可以分为两部分:(1) 来自访问路径上节点 i−1 转发的请求,其次数为 λ'_{i-1};(2) 来自访问路径之外的其他节点的请求(包括节点 i 自身产生的请求),其次数记为λ_i.很显然,有 $\lambda_i=\lambda_i'-\lambda'_{i-1}$ (对节点 0,令 $\lambda_0=\lambda_0'$).

我们使用数据传输所经过的跳数来衡量访问对象的开销.如图 2 所示,假设用 h_i 来表示节点 i 到距其最近的一个缓存有所请求对象的节点的跳数,节点 i 访问对象 O 的开销可以表示为 $C(i,O)=\lambda_i'h_i$.由于经过节点 i−1 的访问请求必然经过节点 i,在整个访问路径上,所有节点访问对象 O 的总开销可以表示为

$$Cost=\sum_{i=0}^{n-1}(\lambda_i'-\lambda'_{i-1})h_i=\sum_{i=0}^{n-1}\lambda_i h_i.$$

当对象 O 从节点 n 回传给节点 0 时,沿途上的每个节点都可以决定是否缓存对象 O 的副本.缓存部署问题研究如何合理地将缓存副本放置在路径的节点上,使得总体访问开销最小.被选中放置缓存副本的一组节点的集合称为一个缓存部署.下面先给出它的形式化定义.

定义 1(缓存部署). 在分布式缓存系统中,假设对象 O 被缓存在节点 0 到节点 n 的路径上的 K 个中间节点:$d_1,d_2,\ldots,d_K$($0\leqslant K\leqslant n$,$0\leqslant d_1\leqslant d_2\leqslant\ldots\leqslant d_K<n$),集合 $D_K=\{d_1,d_2,\ldots,d_K\}$ 是路径上节点集合{0,1,…,n−1}的一个子集,则称 D_K 为一个缓存部署.

由于节点的缓存空间是有限的,当对象 O 被放置到节点 i 上时,一个或多个缓存对象会被替换出去,以腾出空间来容纳新的对象.访问这些对象的开销就会增加.我们将由于缓存替换而增加的访问开销称为替换开销[21].用 m_i 来表示节点 i 放置对象 O 的替换开销,假设被替换出去的对象集合为 R,则 m_i 可以计算如下:

$$m_i=\sum_{O_j\in R}C(i,O_j)\ .$$

下面研究访问开销的通用的计算方法.假设 x,y 是从节点 0 到节点 n 的路径上的两个节点,$0\leqslant x\leqslant y\leqslant n-1$,则从 x 到 y 的路径上的节点为 $x,x+1,\ldots,y$.我们考虑这一组节点的访问开销,分别研究如图 3 和图 4 所示的两种数据访问情况.

(1) 如图 3 所示,缓存副本放置在节点 y,所有从 x 到 y 上的节点的访问请求都由 y 来响应.用函数$\varphi(x,y)$表示这种情形下的总访问开销,可有如下计算:

$$\varphi(x,y)=\begin{cases}0, & x\geqslant y\\ \sum_{i=x}^{y-1}\lambda_i\cdot(y-i), & 0\leqslant x<y\end{cases} \tag{1}$$

(2) 如图 4 所示,缓存副本放置在节点 x 和 y,由于请求的对象总是从最近的缓存副本处取回,因此节点 $x,x+1,\ldots,\left\lfloor\frac{x+y}{2}\right\rfloor$ 的请求由节点 x 响应,节点 $\left\lfloor\frac{x+y}{2}\right\rfloor+1,\ldots,y$ 的请求由节点 y 来响应.用函数 $\phi(x,y)$ 表示这种情形下的总访问开销,可有如下计算:

$$\phi(x,y)=\begin{cases}0, & x\geqslant y-1\\ \sum_{i=x+1}^{\left\lfloor\frac{x+y}{2}\right\rfloor}\lambda_i\cdot(i-x)+\sum_{j=\left\lfloor\frac{x+y}{2}\right\rfloor+1}^{y-1}\lambda_i\cdot(y-j), & 0\leqslant x<y-1\end{cases} \tag{2}$$

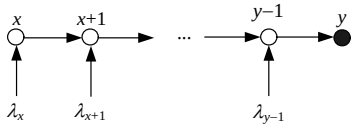

Fig.3 Access cost with object caching in y

图 3 数据对象缓存在 y 节点时的访问开销

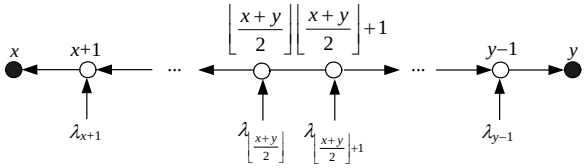

Fig.4 Access cost with object caching in both x and y

图 4 数据对象同时缓存在 x 和 y 节点时的访问开销

对于更一般的情况,假设 $D_K=\{d_1,d_2,\ldots,d_K\}(0\leqslant d_1\leqslant d_2\leqslant\ldots\leqslant d_K<n)$是一个缓存部署,它将访问路径上的节点划分为 $K+1$ 个子集合$\{0,\ldots,d_1\},\{d_1,\ldots,d_2\},\ldots,\{d_K,\ldots,n\}$.对于每个子集,其节点的数据访问满足上述两种情况之一.因此,对于缓存部署 D_K,系统的总访问开销等于各子集的访问开销与替换开销之和,可以表示为

$$\varphi(0,d_1)+\sum_{i=1}^{K}\phi(d_i,d_{i+1})+\sum_{i\in D_K}m_i \tag{3}$$

其中,设 $d_{K+1}=n$.上式就是我们要优化的目标函数.我们首先给出分布式缓存部署问题的形式化定义.

定义 2(缓存部署问题). 给定一组非负实数$\lambda_0,\lambda_1,\ldots,\lambda_{n-1},m_0,m_1,\ldots,m_{n-1}$,假设 $D_K=\{d_1,d_2,\ldots,d_K\}(0\leqslant d_1\leqslant d_2\leqslant\ldots\leqslant d_K<n)$是集合$\{0,1,\ldots,n-1\}$的一个缓存部署,定义目标函数 $C(D_K)$如下:

$$C(D_K)=\varphi(0,d_1)+\sum_{i=1}^{K}\phi(d_i,d_{i+1})+\sum_{i\in D_K}m_i \tag{4}$$

缓存部署问题就是寻找一个集合 D_K,使得目标函数 $C(D_K)$的值最小.

4 求最优缓存部署的图算法

本节介绍一种求解最优缓存部署的图算法.我们构造一个代价图来表示系统访问开销.代价图是一个有向的带权图,在代价图中,从源节点到目标节点的每条访问路径表示一种缓存部署方案,路径的权重之和等于该缓存部署的访问开销.我们使用 Dijkstra 最短路径算法来求代价图的一条最短路径,该路径确定了一种最优的缓存部署方案.最后,我们来证明这种图算法的正确性.

4.1 代价图的构造

我们使用一个有向带权图来描述缓存部署问题,这样的图称为代价图.假设访问路径上的节点集合为$\{0,1,\ldots,n-1\}$,用户请求在节点 n 上得到响应.对应的代价图的顶点有 $n+1$ 行、$n+1$ 列.顶点的排列如图 5 所示.第 0 行有 $n+1$ 个节点,从左到右分别记为 $V_{(0,0)},V_{(0,1)},\ldots,V_{(0,n)}$,第 $i(1\leqslant i\leqslant n)$行有 $n+2-i$ 个顶点,分别记为 $V_{(i,i-1)},V_{(i,i)},\ldots,V_{(i,n)}$.相应地,第 $j(0\leqslant j\leqslant n-1)$列有 $j+2$ 个顶点,第 n 列有 $n+1$ 个顶点.

对于每一行,从左到右每两个相邻顶点之间有一条水平有向边.对于每一列,每个顶点到该列的最后一个顶

点都有一条垂直有向边(第 n 列除外).为了加以区分,在图 5 中,水平边用实线表示,垂直边用虚线表示.我们给每条边赋以一个权值,赋权的方法如下:

(1) 第 0 行的水平边($V_{(0,j)}$,$V_{(0,j+1)}$),权重为

$$W(V_{(0,j)},V_{(0,j+1)})=\sum_{k=0}^{j}\lambda_k,\quad 0\leqslant j\leqslant n-1 \tag{5}$$

(2) 第 i 行的水平边($V_{(i,j)}$,$V_{(i,j+1)}$),权重为

$$W(V_{(0,j)},V_{(0,j+1)})=\begin{cases}0, & i=j+1\\ \sum\limits_{k=\left\lfloor\frac{i+j-1}{2}\right\rfloor+1}^{j}\lambda_k, & \text{otherwise},\end{cases}\quad 1\leqslant i\leqslant n,i-1\leqslant j\leqslant n-1 \tag{6}$$

(3) 第 j 列的所有垂直边,其权重都赋为 m_j,即

$$W(V_{(i,j)},V_{(j+1,j)})=m_j,\ 0\leqslant i\leqslant j,0\leqslant j\leqslant n-1 \tag{7}$$

在代价图中,我们定义点 $V_{(0,0)}$为源节点,第 n 列的节点集合为目标节点,即

$$SOURCE=\{V_{(0,0)}\},DESTINATION=\{V_{(0,n)},V_{(1,n)},\ldots,V_{(n,n)}\}.$$

从源节点到任一目标节点的路径称为访问路径.寻找最优缓存部署的关键是求一条最短的访问路径.

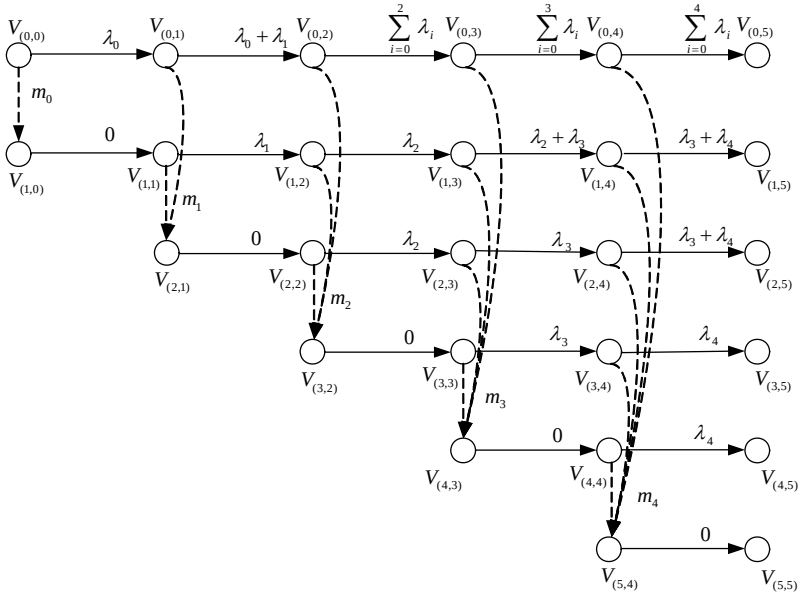

Fig.5 An access cost graph with n=5

图 5 一个访问代价图示例,n=5

4.2 求最优缓存部署的图算法

基于上述构造的代价图,可以求出最优缓存部署.其基本思想是,使用 Dijkstra 算法求出代价图中的一条最短访问路径,该最短路径就对应一种最优缓存部署方案.具体过程描述如下.

(1) 使用第 4.1 节中介绍的方法构造一个代价图,并给每条边赋以相应的权重;

(2) 使用 Dijkstra 算法,求代价图中源节点到目标节点集的所有访问路径的长度,然后求出最短的访问路径;

(3) 将该最短路径映射为一种缓存部署方案,假设第(2)步中求出的最短访问路径经过 k 条垂直边:

$$(V_{(0,x_1)},V_{(x_1+1,x_1)}),(V_{(x_1+1,x_2)},V_{(x_2+1,x_2)}),\ldots,(V_{(x_{k-1}+1,x_k)},V_{(x_k+1,x_k)}),$$

则对应的缓存部署为 $D=\{x_1,x_2,\ldots,x_k\}$,将 D 作为最优缓存部署输出.

4.3 图算法的正确性

本节证明图算法的正确性.为了证明图算法输出的是一个最优的缓存部署,我们首先证明以下引理:

引理 1. 在代价图中,每一条访问路径对应于一个缓存部署;反之,每个缓存部署对应一条访问路径.

证明:如果访问路径不经过任何垂直边,在代价图中只有一条路径 $V_{(0,0)}\to V_{(0,1)}\to\ldots\to V_{(0,n)}$,则其对应的缓存部署为空集.

如果一条访问路径经过 k 条垂直边 $(V_{(0,x_1)},V_{(x_1+1,x_1)}),(V_{(x_1+1,x_2)},V_{(x_2+1,x_2)}),\ldots,(V_{(x_{k-1}+1,x_k)},V_{(x_k+1,x_k)})$,则其对应的缓存部署为 $D=\{x_1,x_2,\ldots,x_k\}$.

反之,对于任何一个非空缓存部署 $D=\{x_1,x_2,\ldots,x_k\}$,可以构建一条访问路径,使它从源节点出发,经过 k 条垂直边 $(V_{(0,x_1)},V_{(x_1+1,x_1)}),(V_{(x_1+1,x_2)},V_{(x_2+1,x_2)}),\ldots,(V_{(x_{k-1}+1,x_k)},V_{(x_k+1,x_k)})$, 最后到达一个目标节点.根据代价图的构造方法,由图 5 可见,这样的访问路径是可以唯一确定的:

$$V_{(0,0)}\to\ldots\to V_{(0,x_1)}\to V_{(x_1+1,x_1)}\to\ldots\to V_{(x_1+1,x_2)}\to V_{(x_2+1,x_2)}\to\ldots\to V_{(x_k+1,x_k)}\to\ldots\to V_{(x_k+1,n)}.$$

如果 D 是空集,则对应一条不经过任何垂直边的访问路径 $V_{(0,0)}\to V_{(0,1)}\to\ldots\to V_{(0,n)}$.

因此,缓存部署与代价图的访问路径是一一对应的.证毕. □

引理 2. 代价图中的每条水平边的权重,满足如下关系:

(1) 对第 0 行的水平边,有 $W(V_{(0,j)},V_{(0,j+1)})=\varphi(0,j+1)-\varphi(0,j)$ $(0\leqslant j\leqslant n-1)$;

(2) 对第 i $(0\leqslant i\leqslant n)$行的水平边,有 $W(V_{(0,j)},V_{(0,j+1)})=\phi(i-1,j+1)-\phi(i-1,j)$ $(i-1\leqslant j\leqslant n-1)$.

证明:

(1) 当 $j=0$ 时,显然 $\varphi(0,1)-\varphi(0,0)=\lambda_1$;

当 $0<j\leqslant n-1$ 时,根据φ函数的定义(公式(1)),有

$$\begin{aligned}\varphi(0,j+1)-\varphi(0,j)&=\sum_{i=0}^{j}\lambda_i\cdot(j+1-i)-\sum_{i=0}^{j-1}\lambda_i\cdot(j-i)\\&=\left(\sum_{i=0}^{j-1}\lambda_i\cdot(j+1-i)+\lambda_j\right)-\sum_{i=0}^{j-1}\lambda_i\cdot(j-i)\\&=\sum_{i=0}^{j-1}\lambda_i\cdot((j+1-i)-(j-i))+\lambda_j\\&=\sum_{i=0}^{j}\lambda_i.\end{aligned}$$

对比方程(5),有 $W(V_{(0,j)},V_{(0,j+1)})=\varphi(0,j+1)-\varphi(0,j)$.

(2) 当 $i-1=j$ 时,显然 $\phi(j,j+1)-\phi(j,j)=0-0=0$.

当 $i-1<j\leqslant n-1$ 时,根据ϕ函数的定义(公式(2)),有

$$\begin{aligned}\phi(i-1,j+1)-\phi(i-1,j)&=\left(\sum_{k=i}^{\left\lfloor\frac{i+j}{2}\right\rfloor}\lambda_k(k-i+1)+\sum_{k=\left\lfloor\frac{i+j}{2}\right\rfloor+1}^{j}\lambda_k(j+1-k)\right)-\left(\sum_{k=i}^{\left\lfloor\frac{i+j-1}{2}\right\rfloor}\lambda_k(k-i+1)+\sum_{k=\left\lfloor\frac{i+j-1}{2}\right\rfloor+1}^{j-1}\lambda_k(j-k)\right)\\&=\left(\sum_{k=i}^{\left\lfloor\frac{i+j}{2}\right\rfloor}\lambda_k(k-i+1)-\sum_{k=i}^{\left\lfloor\frac{i+j-1}{2}\right\rfloor}\lambda_k(k-i+1)\right)+\left(\sum_{k=\left\lfloor\frac{i+j}{2}\right\rfloor+1}^{j}\lambda_k(j+1-k)-\sum_{k=\left\lfloor\frac{i+j-1}{2}\right\rfloor+1}^{j-1}\lambda_k(j-k)\right).\end{aligned}$$

令 $t=\left\lfloor\frac{i+j-1}{2}\right\rfloor$,若 $i+j$ 为奇数,有 $\left\lfloor\frac{i+j}{2}\right\rfloor=\left\lfloor\frac{i+j-1}{2}\right\rfloor=t$, 则

$$\phi(i-1,j+1)-\phi(i-1,j)=\sum_{k=t+1}^{j}\lambda_k(j+1-k)-\sum_{k=t+1}^{j-1}\lambda_k(j-k)=\sum_{k=t+1}^{j}\lambda_k.$$

若 $i+j$ 为偶数,有 $\left\lfloor\frac{i+j}{2}\right\rfloor=\left\lfloor\frac{i+j-1}{2}\right\rfloor+1=t+1$.

$$\phi(i-1,j+1)-\phi(i-1,j)=\lambda_{t+1}(t-i)+\left(\sum_{k=t+2}^{j}\lambda_k(j+1-k)-\sum_{k=t+1}^{j-1}\lambda_k(j-k)\right)$$

$$=\lambda_{t+1}(t-i)-\lambda_{t+1}(j-t-1)+\sum_{k=t+2}^{j}\lambda_k$$

$$=\lambda_{t+1}+\sum_{k=t+2}^{j}\lambda_k=\sum_{k=t+1}^{j}\lambda_k.$$

对比方程(6),有 $W(V_{(0,j)},V_{(0,j+1)})=\phi(i-1,j+1)-\phi(i-1,j)$ 成立.证毕. □

引理 3. 在代价图中,每条访问路径上的权重之和等于对应的缓存部署的访问开销.

证明:在引理 1 中,我们已经证明访问路径和缓存部署是一一对应的.假设 $D_k=\{x_1,x_2,\ldots,x_k\}$是一个缓存部署,它对应的访问路径经过 k 条垂直边 $(V_{(0,x_1)},V_{(x_1+1,x_1)}),(V_{(x_1+1,x_2)},V_{(x_2+1,x_2)}),\ldots,(V_{(x_{k-1}+1,x_k)},V_{(x_k+1,x_k)})$, 这些垂直边的权重之和为 $\sum_{i=1}^{k}m_{x_i}$. 该访问路径还经过 k+1 条水平边,这些水平边及其权重之和可以用引理 2 来计算:

(1) $V_{(0,0)}\to V_{(0,1)}\to\ldots\to V_{(0,x_1)}$, 权重和为

$$\sum_{j=0}^{x_1-1}W(V_{(0,j)},V_{(0,j+1)})=\sum_{j=0}^{x_1-1}(\varphi(0,j+1)-\varphi(0,j))=\varphi(0,x_1).$$

(2) $V_{(x_1+1,x_1)}\to V_{(x_1+1,x_1+1)}\to\ldots\to V_{(x_1+1,x_2)}$, 权重和为

$$\sum_{j=x_1}^{x_2-1}W(V_{(x_1+1,j)},V_{(x_1+1,j+1)})=\sum_{j=x_1}^{x_2-1}(\phi(x_1,j+1)-\phi(x_1,j))=\phi(x_1,x_2).$$

…

(k+1) $V_{(x_k+1,x_k)}\to V_{(x_k+1,x_k+1)}\to\ldots\to V_{(x_k+1,n)}$,权重和为

$$\sum_{j=x_k}^{n-1}W(V_{(x_k+1,j)},V_{(x_k+1,j+1)})=\sum_{j=x_k}^{n-1}(\phi(x_k,j+1)-\phi(x_k,j))=\phi(x_k,n).$$

综上,访问路径的权重总和为

$$\varphi(0,x_1)+\sum_{i=1}^{k}\phi(x_i,x_{i+1})+\sum_{i=1}^{k}m_{x_i}.$$

其中,$x_{k+1}=n$.与方程(4)比较,它刚好等于 $C(D_K)$,即权重之和等于对应的缓存部署的访问开销.证毕. □

定理 1. 图算法给出了缓存部署问题的一个最优解.

证明:根据引理 1,缓存部署和访问路径是一一对应的,引理 3 则说明访问路径的权重之和等于对应的缓存部署的访问开销.在图算法中,使用 Dijkstra 算法求出一条最短访问路径,它对应的就是访问开销最小的缓存部署.因此,图算法给出了缓存部署问题的一个最优解.证毕. □

5 性能评估

5.1 仿真环境

我们通过仿真实验来评估缓存部署算法的性能.目前,大部分研究使用图来刻画网络拓扑[22−25],其中应用最为广泛的是 Waxman 的随机图模型[24].我们使用 GT-ITM[26]网络仿真工具来生成符合 Waxman 模型的随机图,以表示一个分布式缓存系统的覆盖网络,仿真 1 000 个缓存节点.Waxman 模型的γ和β参数分别设为 0.2 和 0.5.

在分布式缓存系统中,每个节点按照一定的模式产生数据访问请求.Breslau 等人[27]的研究结果表明,用户对 Web 页面的访问服从 Zipf-Like 分布,即大部分用户请求集中在小部分的“热点”数据上.假设 N 为数据对象的总数,p_i 为第 i 个热门对象被请求的概率,有 $p_i=\frac{G}{i^{\alpha}}$,其中,$G=\left(\sum_{k=1}^{N}\frac{1}{k^{\alpha}}\right)^{-1}$,$\alpha$是表征访问集中程度的参数.类似文献[16,18,21]中的方法,仿真用户请求到达服从泊松过程,请求到达率λ设为 5.用户对数据的访问符合 Zip-Like 分

布,并默认地将α参数设为 0.9.

我们使用缓存命中率和平均访问延迟这两项性能指标来评估缓存部署算法.缓存命中率等于访问请求在分布式缓存系统中命中的次数除以访问请求的总数,它是一个被广泛用于评估缓存性能的指标.平均访问延迟等于用户取回一个数据所经过的平均跳数,它是我们算法优化的目标.

5.2 性能分析

本节对图算法的性能进行分析.为了进行对比,我们实现了最常用的单机缓存决策的 LRU(least recently used)算法以及相关工作中介绍的 MODULO 算法[6]和 EA(expiration age)算法[15].我们主要考察缓存的大小、缓存节点的数量以及用户访问模式对各种缓存部署算法的影响.

5.2.1 缓存大小的影响

影响缓存系统性能的最重要的因素是缓存的大小.在实验中,假设每个节点的缓存能力是相同的,系统所有节点的缓存空间的总和称为综合缓存空间.令源服务器中数据集的大小总和为 S,我们将综合缓存空间设为 S 的百分比,令其在 30%~80%之间发生变化,考察在不同缓存大小情况下算法的性能.

图 6 显示了系统缓存命中率随缓存大小的变化情况.从图中可见,所有缓存策略的缓存命中率都随着缓存空间的增大而增大.这是因为:一方面,随着缓存空间的增大,单个节点可容纳的数据对象越来越多,本地命中率会有所提高;另一方面,对协同缓存来说,缓存空间的增大,表明可共享的缓存数据会增多,用户访问在其他缓存节点命中的概率也会增大.例如,图中的 LRU 算法的缓存命中率从 0.56 提高到 0.71,增加了 27%.根据 4 种算法的曲线比较来看,LRU 算法的命中率最低;MODULO 和 EA 算法比较接近,都高于 LRU 算法;图算法(GRAPH)的命中率远高于其他 3 种算法,与 LRU 算法相比,图算法的命中率提高了 5%~10%.

图 7 比较了系统访问延迟随缓存大小的变化情况.可见,平均访问延迟随着缓存空间的增大而有所降低.LRU 算法的访问延迟最高;EA 和 MODULO 算法次之;图算法的访问延时最低,比其他算法降低了接近 10%.随着缓存空间的进一步增大,图算法的效果将更加明显.

LRU 算法性能较差的原因在于它是一种单机缓存决策算法,节点间没有协同,缓存冗余度大;而且只考虑数据的最近访问时间,忽略了其他影响性能的因素.图算法是一种协同的缓存部署算法,它考虑了用户请求到达率、网络距离等因素,降低了系统访问开销,在仿真实验中表现出良好的性能.

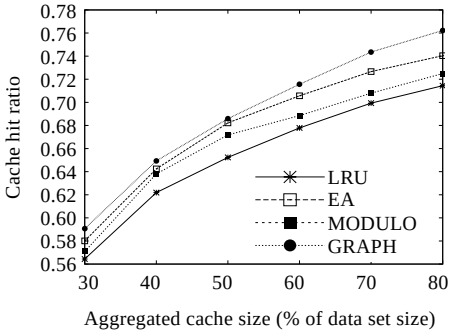

Fig.6 Cache hit ratio under different cache size

图 6 不同缓存大小下缓存命中率的比较

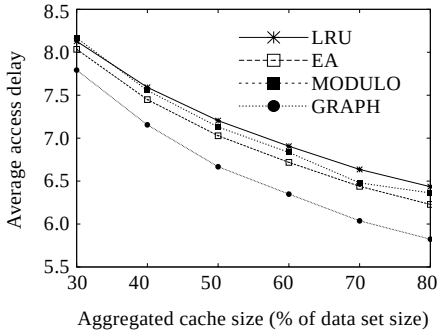

Fig.7 Average access delay under different cache size

图 7 不同缓存大小下平均访问延迟的比较

5.2.2 缓存节点数量的影响

缓存节点的数量代表了系统的规模,当数据服务器的数量和位置固定时,缓存节点越多,共享的缓存空间越大,用户请求在缓存中命中的机会也会越大.我们通过改变 GT-ITM 的仿真参数来生成不同规模的覆盖网络,令网络节点的数量从 20 变化到 1 000,研究缓存部署算法在不同网络规模下的性能.

图 8 显示了网络规模对缓存命中率的影响.可见,几种算法的命中率都随着节点数量的增加而有所增大.例

如,当协同缓存节点数量从 20 增加到 1 000 时,LRU 算法的命中率从 0.28 提高到 0.72.但是,性能的提升并不是无限的,由图 8 可见,当节点数从 50 增加到 100 时,GRAPH 算法的命中率从 0.42 增加到 0.50,提高了 20%.而当节点数从 500 增加到 1 000 时,命中率从 0.73 增加到 0.78,只提高了 7%.可见,随着网络规模的进一步扩大,性能的增长会越来越缓慢,直到达到一个相对稳定的状态.从几种算法的性能比较来看,LRU 算法的命中率最低;MODULO 和 EA 算法比 LRU 算法高一些,但是不太明显;图算法的命中率明显高于其他算法.图 9 显示了平均访问延迟的变化.类似地,图算法具有最低访问延迟,比 LRU 算法降低了大约 10%.

然而,随着缓存节点数量的增加,用于协同缓存管理和缓存定位的通信开销也会相应增大,这部分开销并没有体现在我们的仿真实验中.

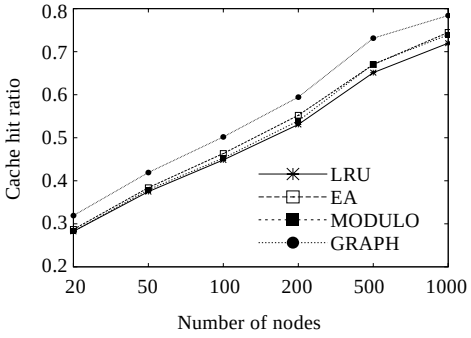

Fig.8 Cache hit ratio under different network size

图 8 不同网络规模下缓存命中率的比较

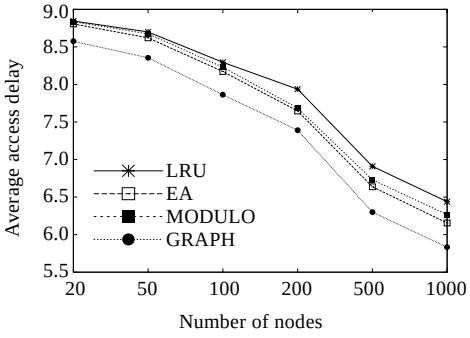

Fig.9 Average access delay under different network size

图 9 不同网络规模下平均访问延迟的比较

5.2.3 用户访问模式的影响

用户访问模式同样会影响缓存系统的性能.研究结果[27−29]表明,网络访问存在“热点”,大部分的用户访问集中在小部分的热门数据上,可以使用 Zipf-Like 分布来刻画这种情况.前面提到,Zipf-Like 分布的α参数表征用户访问的集中程度,α越趋近于 0,用户访问越趋近于均匀分布;α越趋近于 1,用户访问越集中于少部分数据上.可以通过改变α的值来模拟产生不同访问兴趣的用户请求,以便研究几种缓存部署算法的性能.

图 10 和图 11 显示了当α参数从 0.90 变化到 0.65 时,缓存系统性能的变化.当α的值较大时,用户的访问比较集中,缓存策略比较容易辨别出经常被访问的数据,因此系统性能较好.随着α参数逐渐变小,访问热点越趋分散,缓存命中率相应降低,访问延时也逐渐增加.从不同的缓存算法对比来看,图算法在不同的用户访问模式下仍然表现出最优的性能.

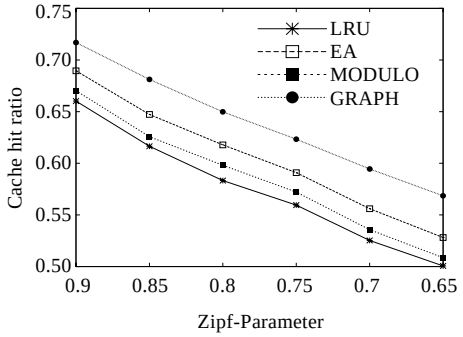

Fig.10 Cache hit ratio under different access pattern

图 10 不同用户访问模式下缓存命中率的比较

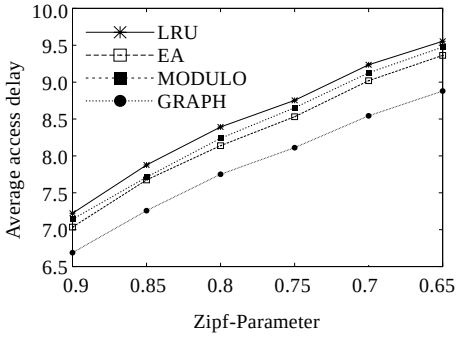

Fig.11 Average access delay under different access pattern

图 11 不同用户访问模式下平均访问延迟的比较

6 总 结

网络缓存技术通过将常用的数据缓存到靠近用户的节点上,可以减轻服务器和网络负载,减少用户访问延迟,目前已被广泛应用.本文研究在分布式缓存系统中如何优化缓存副本的放置,使系统访问总开销最小.首先,我们建立了一个理论模型描述缓存副本放置对系统访问开销的影响,并将缓存放置问题化归为一个最优化问题.然后,我们提出了一种求最优缓存部署的图算法,通过构建代价图并使用 Dijkstra 算法来求出一个最优解.从理论上我们证明了图算法的正确性,并使用仿真实验来评估算法的性能.实验结果表明,图算法的性能优于现有的分布式缓存算法.

References:

[1] Gadde S, Rabinovich M, Chase J. Reduce, reuse, recycle: A approach to building large internet caches. In: Proc. of the 6th Workshop on Hot Topics in Operating Systems (HotOS'97). Washington: IEEE Computer Society, 1997. 93.

[2] Raunak MS. A survey of cooperative caching. Technical Report, 1999. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.27.7178

[3] Zhang L, Floyd S, Jacobson V. Adaptive Web caching. In: Proc. of the 1997 NLANR Web Cache Workshop. Boulder, 1997.

[4] Malpani R, Lorch J, Berger D. Making World Wide Web caching servers cooperate. In: Proc. of the 4th Int'l World Wide Web Conf. (WWW'95). Boston: World Wide Web Consortium, 1995.

[5] Korupolu MR, Dahlin M. Coordinated placement and replacement for large-scale distributed caches. In: Proc. of the 1999 IEEE Workshop on Internet Applications (WIAPP'99). San Jose: IEEE Computer Society, 1999. 62.

[6] Bhattacharjee S, Calvert KL, Zegura EW. Self-Organizing wide-area network caches. In: Proc. of the 17th Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM'98). San Francisco: IEEE Computer Society, 1998. 600−608.

[7] Huang SN, Xi JQ. Distributed data cache systems. Journal of Software, 2001,12(7):1094−1100 (in Chinese with English abstract). http://www.jos.org.cn/ch/reader/view_abstract.aspx?flag=1&file_no=20010718&journal_id=jos

[8] Ling B, Wang XY, Zhou AY, Ng WS. A collaborative Web caching system based on peer-to-peer architecture. Chinese Journal of Computers, 2005,28(2):170−178 (in Chinese with English abstract).

[9] Li WZ, Gu TC, Li CH, Lu SL, Chen DX. Gcaching—A grid-based cooperative caching system. Journal of Computer Research and Development, 2004,41(12):2211−2217 (in Chinese with English abstract).

[10] Laoutaris N, Syntila S, Stavrakakis I. Meta algorithms for hierarchical Web caches. In: Proc. of the Int'l Conf. on Performance, Computing, and Communications (PCCC 2004). Phoenix: IEEE Computer Society, 2004. 445−452.

[11] Wessels D, Claffy K. Internet cache protocol. 1997. http://ds.internic.net/rfc/rfc2186.txt

[12] Valloppillil V, Ross KW. Cache array routing protocol v1.0. Internet draft (draft-vinod-carp-v1-03.txt). 1998.

[13] Cieslak M, Forster D. Web cache coordination protocol v1.0. Internet draft (draft-ietf-wrec-web-pro-00.txt). 1998.

[14] Fan L, Cao P, Almeida J, Broder AZ. Summary cache: A scalable wide-area web cache sharing protocol. IEEE/ACM Trans. on Networking, 1998,28(4):254−265.

[15] Ramaswamyand L, Liu L. A new document placement scheme for cooperative caching on the Internet. In: Proc. of the 22nd Int'l Conf. on Distributed Computing Systems (ICDCS 2002). Vienna: IEEE Computer Society, 2002. 95−103.

[16] Che H, Wang Z, Tung Y. Analysis and design of hierarchical Web caching systems. In: Proc. of the 20th Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM 2001). Anchorage: IEEE Computer Society, 2001. 1416−1424.

[17] Laoutaris N, Che H, Stavrakakis I. The LCD interconnection of LRU caches and its analysis. Performance Evaluation, 2006,63(7):609−634. [doi: 10.1016/j.peva.2005.05.003]

[18] Tang X, Chanson ST. Coordinated management of cascaded caches for efficient content distribution. In: Proc. of the 19th Int'l Conf. on Data Engineering (ICDE 2003). Bangalore: IEEE Computer Society, 2003. 37−48.

[19] Li K, Shen H, Chin FYL, Zhang W. Multimedia object placement for transparent data replication. IEEE Trans. on Parallel and Distributed Systems, 2007,18(2):212−224. [doi: 10.1109/TPDS.2007.29]

[20] Ip ATS, Liu JC, Lui JCS. COPACC: An architecture of cooperative proxy-client caching system for on-demand media streaming. IEEE Trans. on Parallel and Distributed Systems, 2007,18(1):70−83. [doi: 10.1109/TPDS.2007.253282]

[21] Tang X, Chanson ST. Coordinated en-route Web caching. IEEE Trans. on Computers, 2002,51(6):595−607. [doi: 10.1109/TC.2002.1009146]

[22] Zegura EW, Calvert KL, Bhattacharjee S. How to model a internetwork. In: Proc. of the 15th Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM'96). San Francisco: IEEE Computer Society, 1996. 594−602.

[23] Sidhu D, Fu T, Abdallah S, Nair R, Coltun R. Open shortest path first (OSPF) routing protocol simulation. ACM SIGCOMM Computer Communication Review, 1993,23(4):53−62. [doi: 10.1145/167954.166243]

[24] Waxman BM. Routing of multipoint connections. IEEE Journal on Selected Areas in Communications, 1998,6(9):1617−1622. [doi: 10.1109/49.12889]

[25] Verma DC, Gopal PM. Routing reserved bandwith multipoint connections. ACM SIGCOMM Computer Communication Review, 1993,23(4):96−105. [doi: 10.1145/167954.166247]

[26] Zegura E, Calvert K. Georgia tech Internet topology models. 1996. http://www.cc.gatech.edu/projects/gtitm

[27] Breslau L, Cao P, Fan L, Phillips G, Shenker S. Web caching and zipf-like distributions: Evidence and implications. In: Proc. of the 18th Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM'99). New York: IEEE Computer Society, 1999. 126−134.

[28] Arlitt MF, Williamson CL. Internet Web servers: Workload characterization and performance implications. IEEE/ACM Trans. on Networking, 1997,5(5):631−645. [doi: 10.1109/90.649565]

[29] Mahanti A, Williamson C, Eager D. Traffic analysis of a Web proxy caching hierarchy. IEEE Network, 2000,14(3):16−23. [doi: 10.1109/65.844496]

附中文参考文献:

[7] 黄世能,奚建清.分布数据缓存体系.软件学报,2001,12(7):1094−1100. http://www.jos.org.cn/ch/reader/view_abstract.aspx?flag=1&file_no=20010718&journal_id=jos

[8] 凌波,王晓宇,周傲英.一种基于 Peer-to-Peer 技术的 Web 缓存共享系统研究.计算机学报,2005,28(2):170−178.

[9] 李文中,顾铁成,李春洪,陆桑璐,陈道蓄.Gcaching——一种网格协同缓存系统.计算机研究与发展,2004,41(12):2211−2217.

李文中(1979－),男,广西平南人,博士,讲师,CCF 会员,主要研究领域为 P2P 计算,分布式数据管理,无线网络.

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并行计算.

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计算与并行处理,无线传感器网络,普适计算.