

一种应用于分布式缓存系统中的缓存部署算法

王必尧^{1,2}, 王劲林^{1,2}, 吴刚¹, 刘学²

¹(中国科学技术大学 自动化系, 合肥 230027)

²(中国科学院 声学研究所 国家网络新媒体工程技术研究中心, 北京 100190)

E-mail: wangby@dsp.ac.cn

摘要: 缓存技术能有效的节省网络带宽, 减少用户的访问延迟. 在分布式缓存系统中, 一个值得研究的问题是如何根据用户的请求动态的进行缓存部署, 使得系统的收益最大. 描述了缓存部署问题并建立了优化模型, 在此基础上提出一种新的协作缓存部署算法, 该算法利用对象的热度、网络距离, 以及系统中各节点接收的请求和系统缓存分布信息, 依次对请求路径上的节点进行缓存部署决策, 同时该算法将计算分布到请求路径的各个节点上进行. 仿真结果表明, 该算法具有比 LRU 和 Graph 算法更高的缓存命中率和更低的访问延迟.

关键词: 分布式缓存系统; 缓存部署; 协同缓存; 访问延迟

中图分类号: TP393

文献标识码: A

文章编号: 1000-1220(2012)08-1645-05

Cache Deployment Algorithm in Distributed Caching System

WANG Bi-yao^{1,2}, WANG Jin-lin^{1,2}, WU Gang¹, LIU Xue²

¹(Department of Automation, University of Science and Technology of China, Hefei 230027, China)

²(National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: Cache techniques can be used to save network bandwidth, reduce user access latency. One important problem in distributed caching system is how to deploy cache dynamically, according to the user's request, to maximize the system's total benefit. This paper first describes the cache depoly problem, and builds the optimization model for this problem, then proposes a new cooperated cache deployment algorithm, the algorithm utilizes the object popularity, network distance, the request information that every node receive and system cache distribution information, every node that on the request path make a decision on cache deployment in sequence, meanwhile, this algorithm distributes the calculation to the nodes on the request path. As the simulation result shows, this new algorithm has higher hit ratios and lower access latency than LRU and Graph Algorithm.

Key words: distributed caching system; cache deployment; cooperated cache; access latency

1 引言

近几年, Internet 的迅速发展给网络服务系统带来了许多新的挑战, 互联网应用如网络视频, 图片分享等对于响应速度和带宽的要求越来越高. 研究者们提出了不同的方法来加快请求响应速度和减少带宽消耗. 缓存是一种有效的技术用来减轻服务器负载, 减少用户的访问延迟, 节省网络带宽. 由于单节点缓存可能使缓存节点成为性能瓶颈或造成单点故障, 人们发展了协同缓存的技术, 在协同缓存系统中, 各缓存节点相互连接, 共同为用户的请求提供服务. 与单节点缓存相比, 协同缓存可以减少请求处理时间, 提高缓存命中率, 并能提供更强的容错性.

在分布式协同缓存系统中, 一个值得研究的问题是如何根据用户的请求动态的进行缓存部署. 在非协同缓存机制中, 各缓存节点根据本节点统计的访问信息, 采用本地最近访问 (LRU) 或本地最频繁访问 (LFU) 等单节点决策算法进行缓

存决策, 这些决策的方法没有利用系统中其他节点统计的访问信息以及系统当前的缓存分布信息, 造成缓存对象分布不均, 缓存系统的性能不高^[1], 而各个节点在共享访问和缓存信息的基础上协同的进行决策, 能提高缓存系统的整体性能^[2, 3].

本文研究了分布式缓存系统中的缓存部署问题, 提出了一种新的协同缓存部署算法, 将计算分布到请求路径的各个节点上进行, 并通过仿真验证了算法的有效性.

2 协同缓存技术研究现状

协同缓存最早在 Harvest^[4] 项目中被提出, 之后研究者提出了 Internet Cache Protocol (ICP)^[5], Cache Array Routing Protocol (CARP)^[6], Summary Cache^[7] 多种协议来支持节点间的协同缓存. 协同缓存分为层次型协同缓存和分布式协同缓存^[3, 8], 在层次式或树状的结构化网络中, 可以用动态规划等算法计算出最优缓存部署方案^[6], 但这些算法不适用一般

收稿日期: 2011-04-07 基金项目: 国家“八六三”高技术研究发展计划项目(2008AA01A317)资助. 作者简介: 王必尧, 男, 1984 年生, 博士研究生, 研究方向为网络传播与控制; 王劲林, 男, 1964 年生, 研究员, 博士生导师, 研究方向为 IP 网络技术和网络流媒体应用技术、宽带网络体系结构及新业务等; 吴刚, 男, 1964 年生, 教授, 博士生导师, 研究方向为网络新媒体、新能源汽车控制系统、现代农业; 刘学, 男, 1979 年生, 助理研究员, 研究方向为网络与新媒体技术.

的、无结构的分布式协同缓存系统。

Tewari R 等人^[3]提出了大规模分布式缓存设计的4个基本原则:

- 1) 最小化节点转发的跳数和访问数据;
- 2) 缓存未命中时响应也不慢下来;
- 3) 节点间共享缓存;
- 4) 将缓存保存在离用户近的节点上。

作者针对这4个原则提出了相应的策略,将数据和元数据消息进行分开传输,使用地点提示来定位要转发的缓存节点,并使用推送算法使得缓存的更新以及地点提示信息有效。

Ramaswamy L 等人^[9]提出基于失效期缓存部署策略,根据统计信息不同,失效期定义为对象被替换出去的时间与最后一次缓存命中时间的差值或者对象被替换出去的时间与对象进入缓存的时间的差值除以命中次数。当一个节点收到一个对象时,通过与其他节点交换失效期信息来决定是否缓存该对象。

Bhattacharjee S 等人^[10]研究了自组织的广域网络缓存部署策略,它提出了缓存半径的概念,缓存半径通过节点间的跳数来定义,回复消息在节点间传送时,跳数值逐渐增加,收到的回复消息中跳数值是缓存半径的整数倍的节点缓存对象。该策略没有考虑到对象的热门程度的影响。

李文中等人^[2]提出了一种局部优化的缓存部署策略,使用修改的 Dijkstra 算法在构建的代价图中求出一个最优的缓存部署。该算法考虑的是单条请求路径上的访问开销,没有考虑系统中其他节点的访问开销。

总的来说,上述算法中没有充分利用系统中的信息,因此均有改进的余地,本文提出了一种新的协作缓存部署算法,该算法根据对象的热度、网络距离,以及系统中各节点的请求和缓存信息,节点依次进行缓存部署决策,该算法能有效提高系统的缓存命中率和降低访问延迟,并将计算分布到请求路径上的各个节点上,避免将计算集中到某一个节点上,使得节点的负载过大。

3 模型建立和分析

本文研究在广域、无结构的分布式协同缓存系统中如何根据用户请求动态进行缓存副本部署,图1显示了一个无结

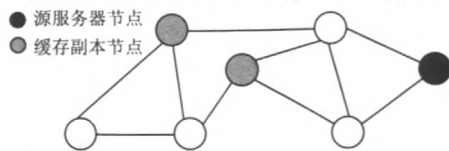


图1 分布式缓存系统

Fig.1 Distributed caching system

构的分布式缓存系统的示意图,在这个系统中,缓存节点和源服务器节点都可以接受并响应用户请求,各节点之间可以相互转发用户请求。当一个节点接收到一个对象访问请求时,它首先查询本地缓存空间中是否存在该对象,如果查询命中则直接将该对象返回给用户;否则,该节点将用户请求转发给其他缓存节点或源服务器,每个节点都采取相同的处理措施,直

至请求被转发到一个缓存有该对象的节点或者源服务器节点。反向沿着请求转发的路径,对象被发送给各个节点以及用户,该转发路径上的缓存节点可以根据事先指定的策略来决定是否缓存该对象。

假定各缓存节点上的缓存的对象是有效的,并且节点可以定位最近的缓存副本,用户的访问请求总是从距离最近的缓存副本得到响应,各缓存节点都能获得当前缓存分布。

本文用无向联通图 $G = (V, E)$ 来表示该缓存系统,其中 V 表示缓存节点的集合, E 表示缓存节点连接的边的集合,对每个 $(u, v) \in E$, 都有一个非负的距离值与之对应,对任意的两个缓存节点 u, v , 它们之间的距离 $d(u, v)$ 为图中 u 和 v 之间的最短路径长度,该距离可以是节点之间的跳数、节点之间的延迟,或者是其他形式定义的值,假定任意两个节点之间的距离是已知的。用 λ_i 表示一段时间内节点 i 记录的用户对访问对象 O 的请求次数(称为请求到达率), $NS(i)$ 表示距离节点 i 最近的缓存该对象的节点,则该节点用户请求对象 O 的代价为:

$$Cost(i, O) = \lambda_i d(i, NS(i))$$

假设请求从 u 转发到 v 依次经过 n 个节点,将请求源节点 u 编号为 0,目的地节点 v 编号为 n ,沿着请求转发方向将其他节点依次编号为 $1, \dots, n-1$,记请求路径上的节点为 p_0, p_1, \dots, p_n ,我们的目标是优化该转发路径上缓存副本的部署,缓存部署可以定义如下:

定义 1.^[2] 在分布式缓存系统中,假设对象 O 被缓存在节点 p_0 到节点 p_n 的路径上的 k 个中间节点: d_1, d_2, \dots, d_k , 集合 $D_k = \{d_1, d_2, \dots, d_k\}$ 是路径上节点集合 $P_{N-1} = \{p_0, p_1, \dots, p_{n-1}\}$ 的一个子集,则称 D_k 为路径 P_{N-1} 上的一个缓存部署。

在请求返回时,如果节点 p_i 缓存该对象,则对节点 j , 当 $d(j, NS(j)) > d(j, p_i)$ 时,节点 j 将访问对象 O 的转发目标定为节点 p_i ,记这些节点的集合为 V_i ,则节点 p_i 缓存对象 O 带来的缓存收益为:

$$b_i = \sum_{j \in V_i} \lambda_j (d(j, NS(j)) - d(j, p_i)) \quad (1)$$

当节点 p_i 缓存对象 O 时,如果 p_i 的缓存空间已满,一个或多个缓存对象需要从缓存空间删除,则 p_i 需要从其他节点访问这些对象,导致访问开销增加,称增加的访问开销为替换开销^[11]。假设从缓存空间删除的对象集合为 R_i ,对任意对象 $O_j \in R_i$,节点 p_i 访问 O_j 的目标转发节点记为 $MS(p_i, j)$,用 m_i 表示节点 p_i 缓存对象 O 的替换开销,则 m_i 的计算表达式如下:

$$m_i = \sum_{j \in R_i} \lambda_j d(p_i, MS(p_i, j)) \quad (2)$$

记在缓存部署 D_k 下,将访问对象 O 的转发目标改为 D_k 中的节点的集合为 V_{D_k} ,对 $j \in V_{D_k}$,记 j 的转发目标为 $RS_{D_k}(j)$,缓存部署问题定义如下:

定义 2. 给定一组非负实数 $\lambda_0, \lambda_1, \dots, \lambda_{n-1}, m_0, m_1, \dots, m_{n-1}$ 假设 $D_k = \{d_1, d_2, \dots, d_k\}$ ($0 \leq d_1 \leq d_2 \leq \dots \leq d_k < n$) 是节点集合 $P_{N-1} = \{p_0, p_1, \dots, p_{n-1}\}$ 的一个缓存部署,定义目标函数 $B(D_k)$ 如下:

$$B(D_k) = \sum_{j \in V_{D_k}} \lambda_j (d(j, NS(j)) - d(j, RS_{D_k}(j))) - \sum_{i \in D_k} m_i \quad (3)$$

缓存部署问题就是寻找一个集合 D_k , 使得目标函数的值最大。

4 顺序增加 (Sequential Add) 的缓存部署算法

Wu J 等人^[12]证明了定义 2 中的缓存部署问题是 NP 完全问题, 其提出的 Greedy Add 算法每次选择使得目标函数最大化的节点缓存副本, 直至目标函数不能增大为止, 但是该算法的全部计算是集中在缓存该副本的节点上, 当缓存规模增大时, 算法的计算时间较长, 不能满足文献[3]中提出的缓存未命中时响应也不慢下来的原则, 本文提出一种新的缓存部署算法, 顺序增加 (Sequential Add) 法, 沿着请求路径, 各节点依次决定是否缓存该副本, 将计算分布到请求路径上的各个节点上, 考虑式(3)的近似目标函数:

$$B'(X) = \sum_{i=0}^{n-1} x_i (b_i - m_i)$$

其中, $X = \{x_0, x_1, \dots, x_{n-1}\}$, $x_i \in \{0, 1\}$ 。

可以将缓存部署问题简化为找到一组 x_i 值使得 $B'(X)$ 最大, x_i 取值为 1 的节点即为缓存部署的节点。给定一个缓存分布, 对每个节点 m_i 为常量, 对整个访问路径至少可以计算出一个 b_i 值, 根据这个特点, 可以先计算其中一个节点的缓存收益和替换代价, 决定该节点是否缓存该副本, 并更新各节点请求对象 O 的最短距离, 然后其他节点根据该节点的缓存决策来决定本节点的缓存策略。本文选取的第一个计算的节点是请求路径上的第一个节点, 因为第一个节点是最后一个接收到返回对象的, 因此有最多的时间来进行计算, 沿着请求转发的路径, 各节点都可以根据前面节点的缓存决策来决定本节点的缓存策略, 由于可以利用前面节点的最短距离信息, 这些节点的计算量相对较少。记系统节点 i 请求对象 O 的最短距离数组 SD_i , 则更新后的最短距离数组为 SD_{i+1} , 以下根据对象的请求和回复两个阶段对算法进行描述:

第一阶段: 当节点 p_i 在接收到来自对象 O 的请求消息后, 按照以下步骤执行:

1) 如果本地没有缓存对象 O , 将请求转发到节点 p_{i+1} ; 否则, 转到第二阶段;

2) 如果 $i=0$, 则获取系统中其他节点 j 请求对象 O 的目标节点 $NS(j)$ 以及对象 O 的请求到达率 λ_j , 计算 $d(j, NS(j))$, 得到最短距离数组 SD_0 ; 否则, 请求来自其他节点, 节点 p_i 接收其他节点发送的距离数组 SD_i ;

3) 对 $\forall j \in V, j \neq p_i$, 如果 $SD_i[j] > d(j, p_i)$, 将 j 加入 V_i , 根据式(1)和式(2)计算 b_i 和 m_i ;

4) 如果 $b_i > m_i$, 则令 $x_i = 1$, 根据下式更新距离数组:

$$SD_i[p_i] = 0;$$

$$SD_i[j] = d(j, p_i); \forall j \in V_i$$

将更新后得到的距离数组 SD_{i+1} 发给节点 p_{i+1} , 第一阶段结束; 否则转到 5);

5) 令 $x_i = 0$, 将 SD_i 发送到节点 p_{i+1} , 第一阶段结束;

第二阶段: 当节点 p_i 收到对对象 O 的回复消息后, 按照以下步骤执行:

1) 如果 $i=0$, 直接返回请求对象 O 给用户, 更新统计信息, 算法结束; 否则返回对象给 p_{i-1} ;

2) 如果 $i < 0$, 并且第一阶段决策 $x_i = 1$, 则缓存对象 O , 否则不缓存对象 O , 算法结束。

本算法克服了 Graph 算法^[2]只是利用局部信息来优化的缺陷, 并且节点间的距离不局限在节点间的跳数, 整个计算过程分布到请求路径的各个节点上, 只有接收请求的节点的计算复杂度比较大, 请求路径上的其他节点可以根据前面节点的放置决策来决定本节点的策略, 同时, 请求转发和缓存部署计算分开进行, 并且计算量最大的节点有最多的时间计算缓存部署。

5 仿真和结果分析

5.1 仿真环境

本文使用 GT-ITM^[13]网络仿真工具来生成随机的拓朴图, 表示一个无结构的分布式缓存系统的覆盖网络。采用的是 Waxman 模型^[14], 模型中的 γ 参数设为 0.2, β 参数设为 0.5, 缓存节点个数默认设为 200。

假定用户访问请求依速率为 λ 的泊松过程到达每个节点, 本实验中, 请求到达率 λ 设为 5。用户对对象的访问服从 Zipf-Like^[15]分布, 该分布中, 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 $p_i = G/i^\alpha$, 其中, $G = (\sum_{i=1}^N 1/i^\alpha)^{-1}$, α 是表征访问集中程度的参数。请求对象的个数默认设为 100, α 参数设为 0.9, 每个节点的请求数为 10000。

为了便于比较, 本文将节点间的距离设为节点间的跳数, 并使用缓存命中率和平均访问延迟这两个参数来评估缓存部署算法的性能, 其中, 缓存命中率定义如下:

$$HR = \frac{\text{访问请求命中的次数}}{\text{访问请求的总次数}}$$

平均访问延迟定义如下:

$$\text{AvgDelay} = \frac{\text{访问请求取回一个对象的总跳数}}{\text{访问请求的总次数}}$$

平均访问延迟是我们算法优化的目标。对每组数据我们仿真 10 次, 计算平均值。

5.2 性能分析

本文采用 Java 对算法进行仿真, 我们实现了单节点缓存策略的 Least Recently Used (LRU) 算法, 文献[2]中介绍的图 (Graph) 算法, 以及本文提出的顺序增加 (SA) 算法, 本文主要考虑缓存空间的大小、缓存节点的数量以及用户访问模式对三种缓存部署算法性能的影响。

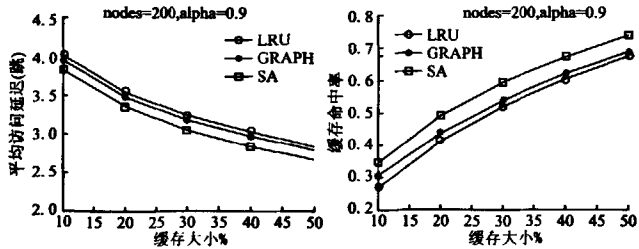
5.2.1 缓存空间大小的影响

缓存空间的大小是影响缓存系统性能的主要因素, 记源服务器中所有对象的大小总和为 S , 本实验中, 假设每个节点的缓存能力相同, 并且缓存空间大小为 S 的百分比, 考察其在 10% ~ 50% 之间发生变化时, 三种缓存部署算法的性能变化。

图 2(a) (见下页) 显示了平均访问延迟随缓存空间大小的变化情况, 图 2(b) 显示了缓存命中率随缓存空间大小的变化情况, 从图中可以看出, 三种算法的平均访问延迟随着缓存空间的增大而减小, 缓存命中率随着缓存空间的增大而增大, 其中, LRU 算法的缓存命中率最低, 平均访问延迟最大, Sequential Add 算法的缓存命中率最高, 平均访问延迟最小。

5.2.2 缓存节点数量的影响

图3(a)显示了平均访问延迟随节点数量的变化情况,



(a) 平均访问延迟与缓存空间大小的关系

(b) 缓存命中率与缓存大小的关系

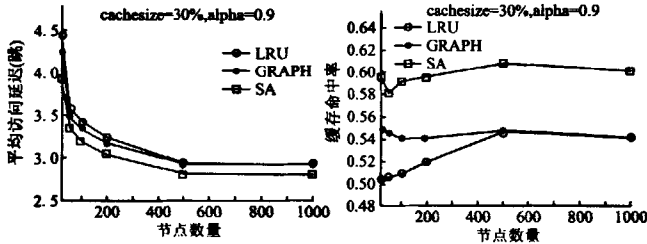
(a) Relation of average access latency and cache size

(b) Relation of hit ratios and cache size

图2

Fig. 2

从图3中可以看出,平均访问延迟随着节点数量的增大而减小,图3(b)显示了缓存命中率随节点数量的变化情况,从图中可以看出,Graph算法和Sequential Add算法的缓存命中率



(a) 平均访问延迟与节点数量的关系

(b) 缓存命中率与节点数量的关系

(a) Relation of average access latency and node number

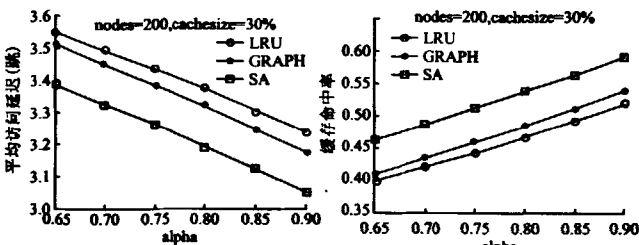
(b) Relation of hit ratios and node number

图3

Fig. 3

对节点数量的变化不敏感,随着节点数量的增多,Graph算法和LRU算法的缓存命中率接近,Sequential Add算法仍然比这两种算法的命中率高。

5.2.3 用户访问模式的影响



(a) 平均访问延迟与α的关系

(b) 缓存命中率与α的关系

(a) Relation of average access latency and α

(b) Relation of hit ratios and α

图4

Fig. 4

图4(a)和图4(b)显示了α从0.65到0.9之间变化时,平均访问延迟和缓存命中率的变化情况。随着α值的增大,

用户的访问越来越集中,3种缓存策略都很容易将热点对象放置在请求节点,而且热点对象被替换出去的概率也渐小,缓存命中率相应提高,访问延时也逐渐降低。

6 算法复杂度分析

设请求路径的长度为 l ,LRU算法每个节点的计算复杂度为 $O(1)$,Graph算法中图的节点数为 n 的复杂度为 $O(l^n)$,本文提出的Sequential Add算法第一个节点需要计算各节点请求对象的最短距离数组,算法复杂度为 $O(n^2)$,其他节点上的计算复杂度为 $O(n)$ 。其中Graph算法的计算都是集中在缓存副本的节点上,在请求路径不长的情況下有很好的执行效率,Sequential Add算法是将计算分布到路径中的各个节点上,只有第一个节点的计算量较大,其他节点的计算量很小。

7 结论

将经常访问的对象缓存到靠近用户的网络节点上,可以有效的节省网络带宽,减少用户的访问延迟。本文研究了分布式缓存系统中的缓存部署问题,提出了一种顺序增加的协同缓存部署算法Sequential Add,利用对象的热门程度和网络距离,以及系统中各节点的请求和缓存信息,进行缓存部署决策,将计算分布到请求路径的各个节点上进行,并通过仿真验证了算法的有效性。实验结果表明,Sequential Add算法优于LRU和Graph算法。

References:

- [1] Laoutaris N, Syntila S, Stavrakakis I. Meta algorithms for hierarchical web caches[C]. IEEE International Conference on Performance, Computing, and Communications, 2004:445-452.
- [2] Li Wen-zhong, Chen Dao-xu, Lu Sang-lu. A graph-based optimal cache deployment algorithm for distributed caching systems[J]. Journal of Software, 2010, 21(7):1524-1535.
- [3] Tewari R, Dahlin M, Vin H M, et al. Design considerations for distributed caching on the Internet[C]. Proceedings of 19th IEEE International Conference on Distributed Computing System, 1999: 273-284.
- [4] Chankhunthod A, Danzig P B, Neerdaels C, et al. A hierarchical internet object cache[C]. Proceedings of USENIX Annual Technical Conference, 1996:13-23.
- [5] Wessels D, Claffy K. Internet cache protocol (ICP), version 2 [EB/OL]. <http://tools.ietf.org/html/rfc2186>, 1997-09.
- [6] Valloppillil V, Ross K W. Cache array routing protocol v1.0[EB/OL]. <http://icp.ircache.net/carp.txt>, 1998-02.
- [7] Fan L, Cao P, Almeida J, et al. Summary cache: a scalable wide-area web cache sharing protocol[J]. IEEE/ACM Trans. Netw, 2000, 8:281-293.
- [8] Rodriguez P, Spanner C, Biersack E W. Analysis of web caching architectures: hierarchical and distributed caching[J]. Networking, IEEE/ACM Transactions on, 2001, 9(4):404-418.
- [9] Ramaswamy L, Liu L. A new document placement scheme for cooperative caching on the Internet[C]. Proceedings of 22nd International Conference on Distributed Computing Systems, 2002:95-103.

- [10] Bhattacharjee S, Calvert K L, Zegura E W. Self-organizing wide-area network caches[C]. INFOCOM, 1998:600-608.
- [11] Tang X, Chanson S T. Coordinated en-route web caching[J]. IEEE Transactions on Computers, 2002, 51(6):595-607.
- [12] Wu J, Shih S, Liu P, et al. Optimizing server placement in distributed systems in the presence of competition[J]. J. Parallel Distrib. Comput., 2011, 71:62-76.
- [13] Zegura E W, Calvert K L, Bhattacharjee S. How to model an internetwork[C]. INFOCOM '96, 1996:594-602.
- [14] Waxman B M. Routing of multipoint connections[J]. Selected Areas in Communications, IEEE Journal on. 1988, 6(9):1617-1622.
- [15] Breslau L, Cue P, Cao P, et al. Web caching and Zipf-like distributions: evidence and implications[C]. INFOCOM, 1999:126-134.
- [16] Li W, Chan E, Feng G, et al. Analysis and performance study for coordinated hierarchical cache placement strategies[J]. Comput. Commun., 2010, 33(15):1834-1842.
- 附中文参考文献:
[2] 李文中, 陈道蓄, 陆桑璐. 分布式缓存系统中一种优化缓存部署的图算法[J]. 软件学报, 2010, 21(7):1524-1535.

万方数据

本刊检索与收录

国内

中文核心期刊

中国学术期刊文摘(中英文版)收录

中国科技论文与引文数据库来源期刊(中国科技核心期刊)

中国科技论文统计源期刊

中国期刊全文数据库(CJFD)收录期刊

中国科技期刊精品数据库收录期刊

中国学术期刊综合评价数据库(CAJCED)收录期刊

中国核心期刊(遴选)数据库收录期刊

中文科技期刊数据库收录期刊

国际

英国《科学文摘》(INSPEC)

俄罗斯《文摘杂志》(AJ, VINITI)


美国《剑桥科学文摘(自然科学)》CSA(NS); Cambridge Scientific Abstracts(Natural Science)

美国《剑桥科学文摘》CSA(T); Cambridge Scientific Abstracts(Technology)

美国《乌利希期刊指南》UPD(Ulrich's Periodicals Directory)

日本《日本科学技术振兴机构中国文献数据库》(JST, Japan Science & Technology Agency Chinese Bibliographic Database)

波兰《哥白尼索引》(IC, Index of Copernicus)

作者: 王必尧, 王劲林, 吴刚, 刘学, WANG Bi-yao, WANG Jin-lin, WU Gang, LIU Xue
作者单位: 王必尧, 王劲林, WANG Bi-yao, WANG Jin-lin(中国科学技术大学自动化系, 合肥230027;中国科学院声学研究所国家网络新媒体工程技术研究中心, 北京100190), 吴刚, WU Gang(中国科学技术大学自动化系, 合肥, 230027), 刘学, LIU Xue(中国科学院声学研究所国家网络新媒体工程技术研究中心, 北京, 100190)
刊名: 小型微型计算机系统 
英文刊名: Journal of Chinese Computer Systems
年, 卷(期): 2012, 33(8)

参考文献(17条)

1. [Laoutaris N;Syntila S;Stavarakakis I Meta algorithms for hierarchical web caches](#) 2004
2. [Li Wen-zhong;Chen Dao-xu;Lu Sang-lu A graph-based optimal cache deployment algorithm for distributed caching systems](#)[期刊论文]-[Journal of Software](#) 2010(07)
3. [Tewari R;Dahlin M;Vin H M Design considerations for distributed caching on the Internet](#) 1999
4. [Chankhunthod A;Danzig P B;Neerdaels C A hierarchical internet object cache](#) 1996
5. [Wessels D;Claffy K Internet cache protocol \(ICP\), version 2](#) 1997
6. [Valloppillil V;Ross K W Cache array routing protocol v1.0](#) 1998
7. [Fan L;Cao P;Almeida J Summary cache:a scalable widearea web cache sharing protocol](#) 2000
8. [Rodriguez P;Spanner C;Biersack E W Analysis of web caching architectures:hierarchical and distributed caching](#) 2001(04)
9. [Ramaswamy L;Liu L A new document placement scheme for cooperative caching on the Internet](#) 2002
10. [Bhattacharjee S;Calvert K L;Zegura E W Self-organizing widearea network caches](#) 1998
11. [Tang X;Chanson S T Coordinated en-route web caching](#) 2002(06)
12. [Wu J;Shih S;Liu P Optimizing server placement in distributed systems in the presence of competition](#) 2011
13. [Zegura E W;Calvert K L;Bhattacharjee S How to model an internetwork](#) 1996
14. [Waxman B M Routing of multipoint connections](#)[外文期刊] 1988(09)
15. [Breslau L;Cue P;Cao P Web caching and Zipf-like distributions:evidence and implication](#) 1999
16. [Li W;Chan E;Feng G Analysis and performance study for coordinated hierarchical cache placement strategies](#) 2010(15)
17. [李文中;陈道蓄;陆桑璐 分布式缓存系统中一种优化缓存部署的图算法](#)[期刊论文]-[软件学报](#) 2010(07)

本文链接: http://d.g.wanfangdata.com.cn/Periodical_xwxjsjxt201208003.aspx