第33卷 第20期 计 算 机 工 程 2007年10月
Vol.33 No.20 Computer Engineering October 2007
·博士论文· 文章编号：1000—3428(2007)20—0004—04 文献标识码：A 中图分类号：TP393

新树型网络加速器的研究与实现

张志立[1,2]，齐德昱[1]，龙颂潜[1]
(1. 华南理工大学计算机科学与工程学院，广州 510640；2. 许昌学院网络中心，许昌 461000)

摘 要：基于缓存间协同工作的思想，提出一种新的树型网络加速器(TIA)，探讨了关键技术，包括扩大查询范围、增强层次连接、使用动态双亲、对象快速定位等。研究了改善层次缓存性能的元算法，开发并实现了能够稳定运行的网络加速器系统，基于日志驱动的试验表明TIA具有良好的加速效果。

关键词：树型网络加速器；协同缓存；网络加速；元算法

Study and Implementation of New Tree-based Internet Accelerator

ZHANG Zhi-li[1,2], QI De-yu[1], LONG Song-qian[1]
(1. School of Computer Science and Engineering, South China University of Technology, Guangzhou 510640;
2. Computer Network Center, Xuchang University, Xuchang 461000)

【Abstract】 This paper proposes a new tree-based Internet accelerator(TIA), and analyzes the key techniques of the accelerator, such as enlarging the query scope, enhancing layer-connections, using dynamic parents and quick locating of objects. A meta-algorithm to improve hierarchical caches performance is researched. A stable network accelerator system is designed and implemented, and the trace-driven experiments show that the new TIA has a good accelerating rate.

【Key words】 tree-based Internet accelerator(TIA); cooperative caching; Web accelerating; meta-algorithm

目前，单个代理服务器的缓存和服务能力都很有限，代理服务器之间共享缓存文件和请求转发的思想导致了协同缓存的出现[1]。协同缓存通过多个代理间的协同工作，充分利用彼此的缓存空间提高命中率、分散热点数据、均衡各服务器负载，避免单点出错和瓶颈问题[2]。它在网络故障容错、网络可伸缩性和满足大量的用户请求等方面，都具有单个缓存无法比拟的优点。协同缓存主要有层次式和分布式结构。

层次式结构中，代理被组织成树状的层次，同一层和上下层的代理之间进行协同和数据共享，代表系统有Squid[3]。缓存结点之间存在着双亲孩子或兄弟关系。其优点是减少了命中文件的期望距离，可随需发布流行文件；缺点是每一层增加了额外的延迟，接近于根的高层缓存可能变成主要的性能和业务瓶颈。由于文件在每一层被复制，造成巨大的冗余，因此需要强大的中间缓存或负载平衡算法支持。

分布式结构中，各代理之间地位平等，没有从属和层次关系，一般使用路由表和组播技术来发现及定位其他代理服务器上的数据，代表系统有CacheMesh[4]和适应性缓存[5]。其优点是业务流通过底层网络，造成很少的拥塞，对于中间的网络层没有额外的磁盘空间要求，具有较好的负载分配和更高的容错能力；缺点是具有高的连接时间、使用较高的带宽和管理较复杂。

如何优化2个缓存结构抽象概括成为新的缓存加速结构，使其具有低的复杂度、高的命中率和小的时间延迟是一个挑战性的课题[6]。在协同缓存中，需要解决对象定位和对象放置2个核心问题，对象定位研究如何快速地寻找一个满足用户请求的、最近的缓存副本，对象放置则研究一个对象何时应该被缓存，缓存到哪一个服务器上可获得更好的系统总体性能[7]。

1 TIA的体系结构及关键技术

基于Internet树状网络拓扑结构的特点，针对上述问题，本文提出一种树型网络加速体系结构(tree-based Internet accelerator, TIA)。它以分层树型结构组织网络上各代理缓存，将分散的多个代理组成协同加速的缓存池，从网络拓扑上解决网络加速、对象快速定位、对象放置和负载平衡等问题。各代理缓存协同工作，充分共享缓存资源，可以面向大规模的客户群体，提供良好的Web访问服务。对于客户来说，TIA就像一台透明的超级加速器，其体系结构如图1所示。

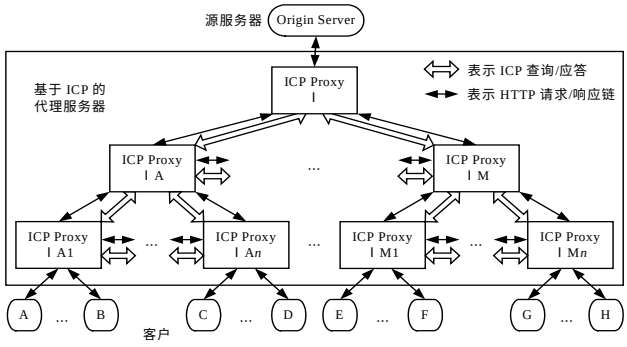

图1 树型网络加速体系结构

加速体系中，缓存之间的关系可定义为2类：(1)双亲孩

基金项目：国家自然科学基金资助项目(60475040)；粤港关键领域重点突破基金资助项目(2005A10307007)；河南省自然科学基金资助项目(0611055000)

作者简介：张志立(1964－)，男，教授、博士研究生，主研方向：Web加速，网格计算；齐德昱，教授、博士生导师；龙颂潜，硕士

收稿日期：2006-10-29 **E-mail：**zzl@xctc.edu.cn

子关系，孩子缓存可以向双亲发送 ICP 查询，双亲作命中或不命中的应答，双亲不会向孩子发送 ICP 查询，孩子查询不命中的请求可以转交给双亲继续请求；(2)兄弟关系，兄弟缓存相互间可以发送 ICP 查询，并作命中或命不中的应答，但是，兄弟缓存不会为另一个兄弟缓存代转不命中的请求。

1.1 TIA 关键技术分析

TIA 仅仅在客户与源服务器之间提供存储转发功能是不够的，涉及到许多问题。这些问题分为 2 大类：(1)架构和模型问题，这里采用树型作为网络模型；(2)算法实现问题，包括替换策略、文件分布策略(对象放置)和 ICP 查询过滤算法。

1.1.1 放大查询范围减少访问延迟

层次式缓存系统为了减少网络流量，孩子只向兄弟发送 ICP 查询，而不向双亲发送 ICP 查询；如果所有的兄弟都应答不命中(MISS)或者超过规定时间时，孩子才向双亲进行 HTTP 请求，然后进行响应，这大大增加了访问延时。

TIA 中，当本地不命中时，允许向所有邻居(包括双亲)发送查询，从而减小了客户的访问延时。另外，向双亲发送查询，也是实现下文所述的“动态双亲”的前提条件。

1.1.2 增强层次连接提高容错性

层次式缓存中，本地缓存和邻居全部不命中时，缓存才向双亲进行 HTTP 请求，由其双亲来处理。如果缓存与双亲之间的网络出现阻塞，或者双亲的负载过重，就会导致响应速度很慢，甚至出现连接超时。

TIA 中，如果本地缓存和邻居全部不命中，通过是否能在限定时间内收到双亲的 MISS 应答来判断双亲是否可用。缓存在向邻居发送查询的同时，也向源服务器的 ECHO 端口试探性地发送 ICP_OP_SECHO 消息 ;如果收到源服务器的回音，就启动 SECHO 超时器。如果 ICP 超时器或者 SECHO 超时器到时，仍然没有收到双亲的应答，就认为在本次请求中双亲不可用，不使用双亲来找目标文件。此时允许孩子缓存使用异构代理(如 Squid)连接源服务器，或者直接连接源服务器。这可提高整个体系的容错性。

1.1.3 使用动态双亲实现负载平衡

层次式缓存没有解决负载平衡问题。通常空闲缓存的响应速度比繁忙缓存快，利用这个特性，通过动态双亲的方法，可使树型缓存体系实现一定形式的负载平衡。动态双亲是指，在程序中允许孩子缓存设置 1 个以上的双亲，当全部查询都不命中时，选择第一个应答 MISS 的双亲进行 HTTP 请求。这样虽然在一段时间内，缓存的双亲数量可能大于 1 个，但在某一时刻，缓存的双亲只有 1 个，树型体系的逻辑结构仍然没变。

使用动态双亲的代价是，缓存需要多发送 1 个 ICP 查询，增加了网络流量，但也增加了查询命中率；另外，相邻 2 组缓存通过设置部分重叠的双亲，一定程度上实现了缓存信息的跨组共享，提高了整个加速体系的命中率。

1.1.4 采用 Bloom Filters 快速定位目标文件

TIA采用Bloom Filters[8]来快速查找目标文件，每个缓存都设置Bloom Filters与本缓存中的Hash表同步一一对应，当ICP查询到达时，使用Bloom Filters来快速查找目标，不用查找复杂数据结构的Hash表，从而能快速作出应答。

Bloom Filters 不是完全精确的，虽然不会把实际存在于缓存的目标对象误报为 MISS，但有可能把不存在的目标对象误报为命中。因此，Bloom Filters 只用于 ICP 查询时应答的查找，而不用于 HTTP 请求时的查找是否命中。

当新缓存一个 URL 文件时，插入 Hash 表，同时在 Bloom Filters 位数组中设置若干位；哪些位应该被设置，是以 URL 为关键字的哈希函数计算得出的，在实际应用中，使用 MD5 作为哈希函数已经取得了很好的效果。

当进行查找时，使用相同的哈希函数确定数组中对应 URL 的位，看是否已被设置。由于某些不同的 URL 可能映射到相同的位，从而产生误报命中(不会误报 MISS)，因此移除一个 URL 时，须考虑如何更新 Bloom Filters。解决方法是，建立一个与位数组同样大小的计数器数组，位数组中每一位被 URL 标记的次数记录为计数器数组中对应下标的数值。移除 URL 时，递减计数器数组中匹配的计数器，当计数器递减为 0 时，重置位数组中匹配的位。

1.2 元算法改善层次缓存系统的性能

层次式缓存中，分层缓存的特征是在*l*层缓存或源服务器的一个文件击中，导致被请求文件的副本被缓存在返回路径上所有缓存中(即层*l*-1,*l*-2,…,1)，即到处留下副本(LCE) [6]。然而，是否需要在每一层存储副本，能否仅仅存储在一个子集中，直到最近才有类似问题的研究出现[9]。

TIA 利用下面 3 个新的“元算法”优化层次缓存性能。之所以称为“元”，在于它们运行在各自的缓存中，独立于实际替代算法之前操作。它们把副本保存在一部分缓存中，而不是全部缓存。

1.2.1 LCD(leave copy down)算法

在客户请求的路径上，被请求文件的副本仅仅保留在击中位置的下方，如图 2 所示。LCD 比 LCE 更加“保守”，因为它需要多次请求才能把一个文件带到叶缓存。

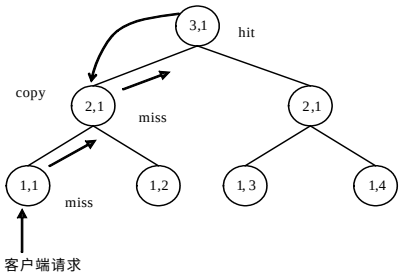

图 2 LCD 算法操作

1.2.2 Prob 算法

如图 3 所示，在从击中位置向下到请求客户端的路径上，每个中间的缓存关于概率 p 保存一个文件的副本，关于概率 $1-p$ 不保存一个文件的副本，这需要调用替代算法。当 $p=1$ 时 Prob 算法就是 LCE。

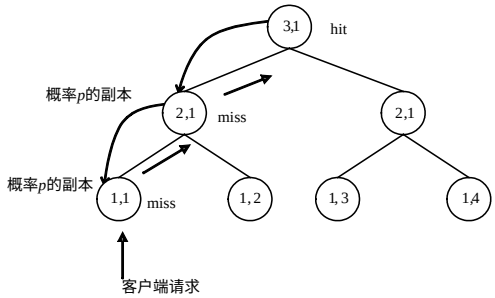

图 3 Prob 算法操作

1.2.3 MCD(move copy down)算法

MCD 算法类似于 LCD 算法，所不同的是在 *l* 层的一个击中，移动被请求的文件到下一层(*l*-1 层)。这要求被请求的

文件从击中发生的缓存中删除掉，但是击中发生在源服务器时，不能删除，如图 4 所示。MCD 的思想是在请求的客户端和源服务器之间的路径上减少同一文件的副本数。这里删除并非是物理删除，而是被标记删除。

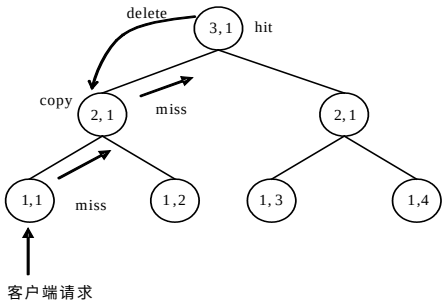

图 4　MCD 算法操作

1.2.4　3 个元算法分析

元算法要求很少额外的协同操作，每个缓存只须知道它的直接双亲以便上行转发请求，知道它的直接孩子以便下行转发文件。

元算法具有如下几个优点：(1)明显减少了平均击中距离，适合于限制存贮的情况；(2)具有较低的复杂度，较少的存贮，不要求额外的信息(如目标请求频率等)；(3)较少或者不改变现有的层次缓存协议(如 ICP 协议)。

3 个元算法根据达到一个缓存命中期望的距离，瞄准改善层次缓存的性能目标，利用以下设计原理。

(1)避免替换错误扩大

当一个不流行的文件引起去除一个流行文件时，就会发生替换错误。对于 LCE 元算法，一个 *l* 层的多层缓存，一个不流行的文件的请求将引起其副本缓存在请求路径上的 *l* 个缓存中，产生了 *l* 个替换错误，导致替换错误的放大。Prob, LCD 和 MCD 通过减少被单个请求触发的副本数来减少错误扩大的程度。

(2)过滤掉一次性请求的文件

由于存在大量的一次性请求的文件，在可测量的业务流量中大约占总请求的 45%和总的不同文件的 75%[10]。缓存一个一次性请求的文件将导致存贮容量的浪费，因为一次性请求的文件在缓存中不再被请求。LCE通过缓存一次性请求的文件剥夺了流行文件的存贮，阻塞了所有有价值的缓存；对于LCD和MCD来说，除根缓存外，一次性文件并不能影响任何缓存；Prob通过用一个小的缓存概率*p*过滤掉了一次性文件。

(3)合理化复制的次数

LCE 在所有的中间缓存存贮文件副本达到 2 个目标：1)有一个临近的副本来服务连接到叶子缓存的其他客户；2)为了文件请求的客户有一个“后援”副本，以防叶子缓存被去除。然而，当需求模式不相似或存储容量是有限的时，存储大量的副本并非总是有益的。Prob, LCD 和 MCD 引起较少的文件被缓存，因此，对每一个缓存文件造成较小的副本数，允许更多的不同的文件被缓存。

2　TIA 的实现模型

2.1　TIA 程序的总体架构

TIA 程序的总体架构见图 5，虚线方框内为 ICP 代理程序的各个执行处理部件。在程序实现中，每个部件都有一个 Java 类与之对应，部件名称后面的括号内容为与之对应的 Java 类名。

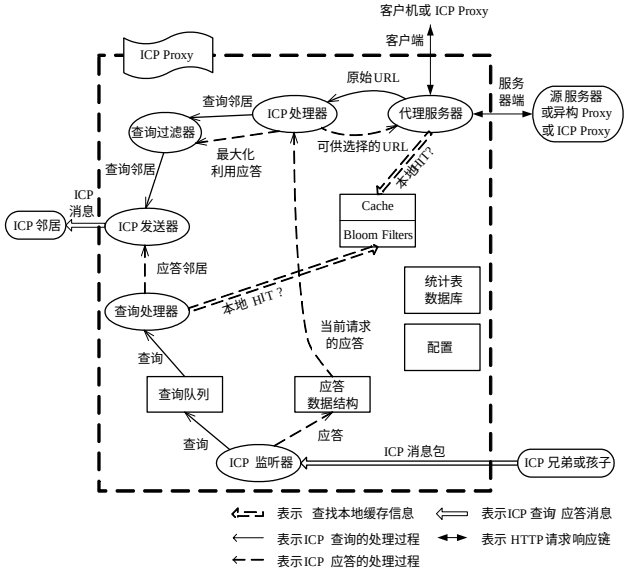

图 5　TIA 程序的总体架构

在 ICP 代理的总体架构中，配置(config)和统计表数据库(statisticDB)为其他各个部件所共用，提供各种公共信息。其中，在程序实现时，配置类分为 2 个类：(1)代理服务器关于 HTTP 以及缓存的基本配置(proxy.Config)；(2)关于 ICP 的配置(Icp.Config)。

另外，在 ICP 代理的总体架构中，与外部有 2 个 HTTP 连接端：客户端和服务器端。2 个连接端的相关操作都是由同一部件——代理服务器负责执行，在 ICP 代理的总体架构中，与外部的 ICP 邻居或孩子通过 UDP 方式进行通信，由 2 个部件来负责执行相关的操作：ICP 发送器负责向 ICP 邻居发送 ICP 消息包，ICP 监听器负责接收 ICP 兄弟或孩子发来的 ICP 消息包。

程序启动以后，后台程序(daemon)在配置(config)指定的 HTTP 客户端口进行监听，当接收到来自客户机或其他 ICP 代理的 HTTP 请求时，创建一个新的代理服务器(proxy)线程来处理该请求。

2.2　TIA 的实现方式与运行环境

为了具有良好的跨平台性，选择使用 Java 语言来实现加速器系统，本系统可以运行在所有支持 Java 虚拟机(JVM)的软硬件平台上。

在开发编程过程中，使用操作系统 Windows2000，采用 Java2 Runtime Environment, SE v1.4.1_02 作为运行平台，采用 J2SDK, SE v1.4.1_02 的编译器，IDE 开发工具使用 IntelliJ IDEA 3.0.5。另外，本程序只使用 J2sdkSEv1.4 提供的基本类库，实现代码略。

2.3　TIA 的功能

(1)树型结构有利于减轻中心节点负载，实现层次化管理，降低管理复杂度；实现任务负载平衡，提高查找效率。

(2)支持大规模用户群。支持地域上分散的大规模用户群，每个加速器可以为一个客户群服务，通过多个加速器的协同工作，提供比单个服务器更好的服务。

(3)可扩展性强。加速器可以动态地加入和退出，当客户过多时，可以通过增加加速器的数量来缓解各服务器的负载，从而可以接入更多的用户。一台服务器要加入 TIA，只须简单配置就可提供加速服务。

(4)提高缓存命中率，优化数据分布。通过使用适当的缓存替换算法，可以提高缓存命中率，使频繁访问的热点数据靠近用户端分布，从而加快响应时间，提供更好的服务能力。

(5)新加入节点可以根据其位置和网络情况来配置父节点和邻居节点，提高节点之间的网络通信性能。

3 试验系统演示

搭建由 6 台 PC 构成的高度为 3 的树型加速器的试验系统，运行本文实现的 TIA。试验系统网络为 100Mb/s 校园网。其中，6 台 PC 的 CPU 为奔腾 4 (2.4GHz)，内存为 512MB，操作系统为 Windows 2000。

TIA 试验系统的树型逻辑结构图如图 6 所示，节点以主机 IP 地址最后一个字节的整数值代表该主机。

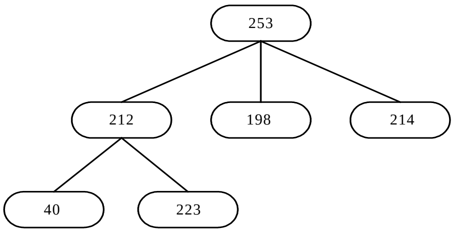

图 6 TIA 加速器的树型逻辑结构

采用Boston大学计算机科学系代理服务器日志进行测试[11]，日志从 1994 年 11 月中旬到 1995 年 2 月底。综合试验系统所得数据，ICP应答HIT个数与MISS个数的比例约为 1:4.6，命中率约为 18%。考虑到本地命中率一般在 25%~50%之间，因此，TIA加速器实际命中率在 40%~70%之间。说明TIA加速器确实可以减少网络拥塞、减轻服务器负载、减小访问延时，面向网络应用层取得了理想的加速效果。

4 结束语

TIA 树型加速器吸取层次结构和分布式结构的优点，同时提出了许多新的技术和算法，抽象概括为一种新的缓存加速结构。本文基于日志驱动的试验说明 TIA 具有良好的加速效果，下一步的研究工作包括：从理论上定量分析 TIA 加速结构的有效性、完备性；部署在 Internet，接受实践的检验，进一步完善其功能，使 TIA 成为互联网上结构先进、功能强大、布局合理实用的网络加速系统。

参考文献

1 Malpani R, Lorch J, Berger D. Making World Wide Web Caching Servers Cooperate[C]//Proc. of the 4th Int'l World Wide Web Conf., Boston, MA. 1995.

2 Barish G, Obraczka K. World Wide Web Caching: Trends and Techniques[J]. IEEE Communications Magazine, 2000, 38(5): 178-184.

3 The Squid Project[Z]. [2006-10]. http://www.squid.cache.org.

4 Wang Zheng. Cachemesh: A Distributed Cache System for World Wide Web[C]//Proc. of the NLANR Web Caching Workshop, Boulder, Colorado, USA. 1997.

5 Zhang L, Floyd S, Jacobson V. Adaptive Web Caching[C]//Proc. of the NLANR Web Cache Workshop, Boulder, Colorado, USA. 1997.

6 Che Ha, Tung Ye, Wang Zhijun. Hierarchical Web Caching Systems: Modeling, Design and Experimental Results[J]. IEEE Journal on Selected Areas in Communications, 2002, 20(7).

7 李文中，顾铁成，李春洪，等. GCaching——种网格协同缓存系统[J]. 计算机研究与发展, 2004, 41(12): 2221-2227.

8 Mullin J K. A Second Look at Bloom Filters[J]. Communications of the ACM, 26(8): 570-571.

9 Laoutaris N, Syntila S, Stavrakakis I. Meta Algorithms for Hierarchical Web Caches[C]//Proc. of IEEE International Performance Computing and Communications Conference, Phoenix, Arizona. 2004-04.

10 Mahanti A, Williamson C, Eager D. Traffic Analysis of a Web Proxy Caching Hierarchy[J]. IEEE Network Magazine, 2000, 14(3): 16-23.

11 Davison B D. Boston University CS Dept Client Traces[Z]. [2006-10]. http://www.web-caching.com/traces-logs.html.

~~~~~~

(上接第 3 页)

表 3 几种算法结果比较

标杆文件	KCR 算法	改进 DME 算法	多级 SA
r1	1 627	1 497	1 503
r2	3 349	3 013	3 006
r3	4 360	3 902	3 927
r4	8 580	7 782	7 785
r5	12 928	11 665	11 633

表 3 中数据表明，多级模拟退火算法可以得出比其他所列启发式算法更好的结果。但计算复杂性明显增加，但随着计算机技术的不断发展，时间因素在该问题的求解过程中将会得到解决。

4 结论

本文阐述了一种形成多级时钟二叉树的模拟退火算法，从理论上说明了该算法的求解思路和求解过程及相关策略，在此基础上对节点合并策略和单节点二叉树的情况作出了重点的论述。实验中，给出了该算法形成时钟二叉树的过程。同时将该算法用于对随机测试例子和标杆文件的测试，并和传统的启发式算法得出的结果进行了比较，说明所述算法较之传统启发式算法有一定的改进作用。

参考文献

1 Jackson M A B, Srinivasan A, Kuh E S. Clock Routing for High-performance ICS[C]//Proc. of ACM/IEEE Design Automation Conference. 1990: 573-579.

2 Kahng A B, Cong J, Robins G. High-performance Clock Routing Based on Recursive Geometric Matching[C]//Proc. of ACM/IEEE Design Automation Conference. 1991: 322-327.

3 Tsay R S, Exact Zero Skew[C]//Proc. of IEEE International Conference on Computer-aided Design. 1991: 336-339.

4 Rubinstein J, Penfield P, Horowitz M A. Signal Delay in RC Tree Networks[J]. IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, 1983, 2(3): 202-211.

5 Ting-Hai Chao, Yu-Chin Hsu, Jan-Ming Ho, et al. Zero Skew Clock Routing with Minimum Wavelength[J]. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, 1992, 39(11): 799-814.
~~~~~~