超节点 P2P 网络中一种有效的缓存策略*

季美丽,王新华,徐连诚

(山东师范大学 信息科学与工程学院,山东 济南 250014)

摘 要:针对超节点 P2P 系统的特点,提出了一种有效且灵活的缓存策略。该策略使用文件价值来决定缓存替换的对象,并且在替换之前使用"阈值"选择要缓存的文件,使其系统只缓存价值较大的热点文件。最后通过 Trace-Driven 的方法模拟实验,结果表明,与现有的缓存策略 LRU 和 LFU 相比,这种缓存策略具有较好的缓存命中率和字节命中率。

关键词:超节点 P2P;缓存;文件价值;阈值;日志驱动模拟

中图分类号:TP393 **文献标识码**:A **文章编号**:1674-7720(2010)17-0039-04

An effective caching strategy in super-peer P2P networks

JI Mei Li, WANG Xin Hua, XU Lian Cheng

(School of Information Science and Engineering, Shandong Normal University, Jinan 250014, China)

Abstract: This paper presents an effective and flexible caching strategy according to the characteristics of super-peer P2P system. It uses file cost to determine the object which will be replaced in the cache, and before that uses threshold value to select the file being cached. It makes the system only cache the valuable and hot files. Finally the trace-driven simulation, results show that this caching strategy achieves better performance in terms of the hit ratio and the byte hit ratio compared with the LRU and LFU.

Key words: super-peer P2P; cache; file cost; threshold value; trace-driven simulation

P2P(Peer-to-Peer)被称为对等连接或对等网络。P2P 改变了传统的客户机/服务器模式,将网络应用的核心从中央服务器向网络边缘的终端设备扩散。最早出现的 P2P 网络是以 Napster 为代表的集中式 P2P 网络,它采用中央服务器管理 P2P 的各节点,这种中心化的模式容易遭到直接的攻击从而导致网络不稳定。随后出现的分布式 P2P 网络虽然解决了抗攻击问题,但是缺乏快速搜索和扩展性[1]。为克服这些缺陷,一些学者将 P2P 结构模型和 C/S 结构模型相结合,提出了超节点 P2P 网络系统。

P2P 系统在用于信息共享领域时,查询与存取操作是其最基本的操作。因此,如何提高超节点 P2P 网络的存取效率是研究的一个重要内容,对于提高 P2P 系统的性能具有重要意义。当前一种较为常用的方法是合理地使用缓存机制。缓存机制是利用局部性原理来提高系统的性能,其基本思想是利用网络上其他机器缓存中的数据来提高本地机器性能[2]。目前大多数超节点 P2P 网络应用了传统的缓存机制,这种方法没有区分不同节点对资源的需求及关注程度的差异,导致偶尔访问的对象可能替换经常访问的对象。因此,在缓存不大的情况下,使用这种放置方式必然导致替换发生次数过于频繁而降低查询效率[3]。

为了解决上述问题,本文提出了使用文件价值来决定缓存替换的对象,同时在缓存替换之前使用阈值选择缓存文件的策略。这种缓存策略使得系统只缓存价值较大的热点文件,从而通过提高命中率有效地降低了网络流量负载。

1 超节点 P2P 系统

1.1 体系结构

在广域网中存在大量的节点服务器,这些节点服务器通过 P2P 路由机制自组织成一个虚拟的 P2P 网络。在这个虚拟的 P2P 网络中,各节点服务器之间以无结构的 P2P 的形式连接,所有的服务器具有相同的能力和责任,任意两台服务器之间能够相互通信,并且所有的通信都是对等的。系统通过这些服务器为用户提供服务,这些节点服务器被称为超节点,而用户节点被称为普通

*基金项目:山东省自然科学基金资助项目(Y2006G19)

节点。超节点 P2P 网络的拓扑结构如图 1 所示。

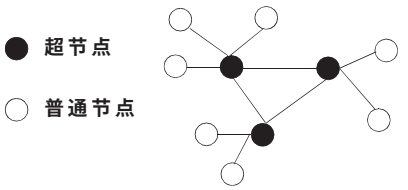

图 1 超节点 P2P 网络的拓扑结构

从图 1 可以看出,超节点具有管理组内普通节点、向用户返回查询结果和转发的功能。普通节点和超节点之间则以传统的 C/S 模式连接在一起,并且每个超节点与若干个普通节点形成组。每个超节点都要维护以下信息表:

(1)本地信息表:保存本地共享资源的索引表。

(2)快捷兴趣信息表:保存与本节点有快捷链接的兴趣相似的超节点信息。该表由(*interest*,*fag*,*num*,*address*)组成,其中 *interest* 代表节点查询的兴趣,*fag* 是快捷链接建立的标志,取值为 0 或 1(建立快捷链接),*num* 代表查询成功的次数,初始值为 0,*address* 代表快捷链接的节点位置。

1.2 超节点间的资源搜索策略

本文采用一种新的搜索策略,该搜索策略可以根据超节点的共享兴趣,逐步在具有相似兴趣的超节点之间建立直接的快捷链接[4]。为了进一步提高查询效率,利用缓存存放访问频率较高的快捷链接。

1.2.1 快捷链接的建立

当一个节点服务器 P_i 刚加入超节点 P2P 网络时,P_i 根据本地的共享资源建立本地信息表,并且该表会随着本地共享资源的变化而更新。当 P_i 第一次发出查询时,它采用 Gnutella 的洪泛式搜索策略在网络中搜索,搜索的同时返回所有拥有该资源的节点表,这些节点都是快捷链接的候选者。当 P_i 成功地从 P_j 中下载资源时,P_i 中本地快捷兴趣信息表的 *num* 加 1;当 P_i 的后续查询继续在 P_j 中搜索成功并完成下载时,*num* 继续加 1;当 *num* 达到设定的阈值时,*fag* 置为 1,表示快捷链接的建立。同时快捷链接还可以通过节点的快捷链接建立,如节点 P_i 向 P_j 发出查询,P_j 首先通过自己的快捷链接进行查询,搜索成功后,将信息返回给节点 P_i,P_i 成功下载后直接在本地兴趣表中建立快捷链接,并将 *num* 赋予规定的阈值。当利用快捷链接搜索信息不成功时,*num* 就减 1,当 *num* 小于设定的阈值时,就取消两节点的快捷链接。

1.2.2 快捷链接的缓存

当本地快捷兴趣信息表建立后,将快捷链接按照 *num* 值的大小进行排队。如果某个快捷链接的 *num* 值最大(说明它经常被使用),就把它放在缓存中,当节点进行搜索时首先调用缓存的快捷链接。其他没被缓存的链接按 *num* 值的大小在快捷兴趣信息表中排队,*num* 值大的排在上面,其他依次类推。同时每个超节点的快捷兴趣信息表可以自我调整,以适应网络的变化。当一个快捷链接的超节点离开网络、或长时间不使用、或命中率很低时,应及时调整它的 *num* 值以及存放的位置,直到最后把该快捷链接去除。

1.3 超节点 P2P 系统中的对象定位

当普通节点 P_0 发出查询请求时:

(1)P_0 首先查询本地缓存,若命中则查询结束;否则转到(2)。

(2)P_0 访问它所连接的超节点 P_i,若在 P_i 中命中则由 P_i 将命中的内容返回给 P_0,查询结束;否则,由 P_i 在超节点间转发查询,具体过程是:①P_i 发出资源搜索时,首先通过缓存中的快捷链接进行查询,若命中,则返回结果后结束;否则转②;②P_i 向自己的快捷兴趣信息表中的快捷链接发出查询,若命中,则返回结果后结束;否则转(3)。

(3)P_i 将查询请求转发给文档原来的存放站点,若命中,则返回结果后结束;否则发回错误报告。

2 缓存策略

2.1 缓存对象的选择

在传统的缓存策略中,当用户访问的文件不在离用户最近的服务器上,那么不管该文件最近是否被访问过,都将该文件缓存到最近的服务器上。在实际应用中,这种缓存策略不能灵活地针对实际情况做实际的调整,有时候会降低缓存的命中率。为了克服这一缺点,本文使用阈值来选择缓存对象[5]。

在超节点 P2P 系统中,当文件和用户的数量足够大时,在短时间内将会有大量的文件被访问,而对于单个的文件来说,被访问的频率是不一样的。本文根据文件的被访问频率用阈值来将它们区分为热点文件和非热点文件。当一个文件的被访问频率超过设定的阈值时称为热点文件,否则为非热点文件。本文只缓存热点文件,这样能够有效地利用服务器的缓存空间,防止在缓存空间不足的情况下,文件副本在缓存空间频繁地替换而导致缓存的命中率下降。

为了提高缓存的作用,阈值的大小应根据实际情况来调整。可以根据缓存空间的大小和文件类型来设置阈值的大小。当缓存空间较大时,可以适当减小所有文件的阈值;当缓存空间较小时,应适当增大文件的阈值,使得缓存策略更关注那些较热的文件。同时,可以为更新较少的文件类型设置一个较小的阈值,这样文件将更容易被缓存;为更新较多的文件类型设置一个较大的阈值,那么该类型的文件就比较难于被缓存。

2.2 缓存替换策略

本文提出了使用文件价值决定缓存替换对象的替换策略,该策略要求所有的节点都维护同一请求列表。请求列表中记录着该节点访问过的所有文件的基本信息,如表 1 所示。

表 1 请求列表

fn	*qc*	*fat*	qc_1	*mat*	qc_2	*lat*	*qf*
A	6	t_1	2	t_2	4	t_3	$(\alpha\times2+\beta\times4)/(t_2-t_1)$
B	…	…	…	…	…	…	…
C	…	…	…	…	…	…	…

表 1 中，*fn*、*qc*、*fat*、qc_1、*mat*、qc_2、*lat*、*qf* 分别是 *file name*、*query count*、*first access time*、*query count*$_1$、*mid access time*、*query count*$_2$、*last access time*、*query frequency* 的缩写。根据实际情况，定义中间访问时间为：

$$mid\ access\ time=first\ access\ time+(current\ time-first\ access\ time)/2 \quad (1)$$

定义文件被访问的频率为：

$$Frequency_{file}=query\ frequency=\alpha\times query\ count_1/(mid\ access\ time-first\ access\ time)+\beta\times query\ count_2/(current\ time-mid\ access\ time)=(\alpha\times query\ count_1+\beta\times query\ count_2)/(mid\ access\ time-first\ access\ time),\ \alpha+\beta=1,\ \alpha<<\beta \quad (2)$$

其中，*query count*$_1$ 为文件在前半个时间段内被访问的次数，*query count*$_2$ 为文件在后半个时间段内被访问的次数，*query count* 为两者之和。根据当前的时间和表中最后一次访问的时间，可以得到文件最后一次被访问后流逝的时间为：

$$Recency_{file}=current\ time-last\ access\ time \quad (3)$$

在这里，文件的价值是由文件被访问的频率和文件最后一次被访问后流逝的时间来决定的。文件被访问的频率越高越不容易被替换，文件被缓存的价值也就越大；反之就越小。同样，文件最后一次被访问后流逝的时间越长越容易被替换，文件被缓存的价值也就越小；反之就越大。因此，本文将文件的价值定义为：

$$Cost_{file}=Frequency_{file}/Recency_{file} \quad (4)$$

并且可以按照表中的数据计算出缓存中现有文件的 $Cost_{file}$ 值。

当一个节点从服务器中成功地下载文件 A，并判断其被访问的频率大于设定的阈值后，按照以下步骤进行缓存替换：

(1)如果节点的缓存空间足够容纳文件 A，那么文件被缓存。否则转到(2)。

(2)比较文件 A 和缓存中候选文件的 $Cost_{file}$ 值。先与 $Cost_{file}$ 值最小的候选文件相比，若文件 A 的值小，则不进行缓存替换；若文件 A 的值大，则将 $Cost_{file}$ 值最小的候选文件从缓存中去除，然后判断缓存空间的大小，转到(1)。

3 模拟实验和结果

为了验证缓存策略的实际性能，这里选取了一组 Web Cache 的访问日志作为工作负载并使用 Trace-Driven 的方法模拟实验。在实验中，为了进行性能比较，对 LRU[6]和 LFU[7]也进行了性能测试。

实验选取的 Web Cache 访问日志为学校网络中心的 Web 服务器在一天 24 小时内的日志记录。在每一条日志记录中包括的主要字段有：独立用户的 IP 地址、上网时刻、目的网站的域名、目的网站的 IP 地址、URL、访问文件的大小等。然后实验模拟了分布在广域网上的 10 台节点服务器。所有这些节点服务器按照 P2P 路由机制组成一个完全分布式无结构的 P2P 网络，并使用散列的方法把从日志中提取到的独立的用户均匀分布到这些服务器上去，同时假设用户在物理上距离自己所分配到的服务器最近，然后用同样的方法把从日志中提取到的互不重复的 URL 所对应的文件均匀分布到这些服务器上去，假设每一个文件都位于自己所分配到的服务器上。这样，就把 Web Cache 的访问日志纪录映射到虚拟的超节点 P2P 网络中了。

缓存策略性能的好坏可以从缓存命中率和缓存字节命中率这两个方面来衡量。缓存命中率(Hit Ratio)表示从缓存得到服务的请求占总请求的百分比，缓存字节命中率(Byte Hit Ratio)表示从缓存得到服务的字节量占总请求字节量的百分比[8]。在实验中，只比较缓存空间在存储空间中所占比重小于一半的情况，因为缓存继续增加对提高性能而言效果不再明显。图 2 和图 3 显示了该缓存策略与传统缓存策略 LRU 和 LFU 的性能比较。

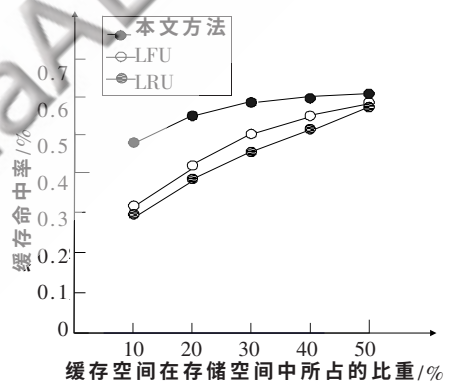

图 2 缓存命中率的比较

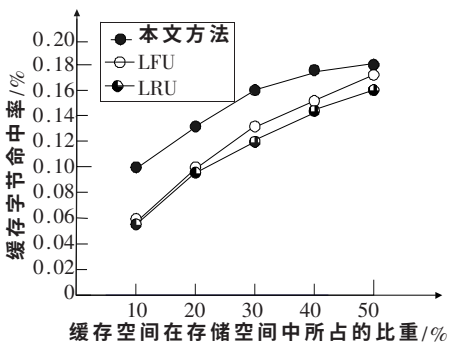

图 3 缓存字节命中率的比较

从图 2 和图 3 可以看出，本文提出的缓存策略在性能上要好于传统的 LRU 和 LFU。当缓存空间比较小时，本文提出的缓存策略表现出更突出的性能，随着缓存空

间的逐渐增大，这三种方法所表现出的性能趋于相同。另外，阈值设置的大小也会影响缓存的命中率和字节命中率，因此，在实验过程中要根据缓存空间的大小和文件的类型及时调整阈值，以达到提高性能的目的。在参考文献[5]中的部分结论选择较为理想的，如1、2、3、4等较小的阈值，具体设置在此不做详细论述。

本文提出了一种有效且灵活的缓存策略，该策略使用文件价值来决定缓存替换的对象，同时在缓存替换之前使用阈值选择要缓存的文件，这使得系统只缓存价值较大的热点文件。最后使用Trace-Driven的方法模拟实验，并且将其与传统的LRU和LFU做了性能上的比较，验证了该缓存策略在性能上比传统的LRU和LFU要好。

参考文献

[1] 张文，赵子铭.P2P网络技术原理与C++开发案例[M].北京：人民邮电出版社，2008.

[2] PATTERSON D A, HENNESSY J L. Computer architecture: a quantitative approach[M]. Elsevier, 2002.

[3] 陶烨，王义麟，王远，等.一种超节点P2P网络中基于语义的协同缓存管理机制[J]. 计算机科学，2007，34(11)：32-36，40.

[4] 杨振会，程楠.非结构化P2P网络的资源搜索算法研究[J].现代计算机，2007，261:128-129，130.

[5] 高伟，韩华，代亚非.一种P2P环境下分布式文件存储系统的缓存策略[J].计算机工程与应用，2004，30:45-49.

[6] MORI T, ASAKA T, TAKAHASHI T. A novel cooperative caching scheme for unstructured peer-to-peer networks[C]. In: IEEE Consumer Communications and Networking Conference, Jan, 2009:1-5.

[7] LAOUTARIS N, SMARAGDAKIS G, BESTAVROS A, et al. Distributed selfish caching [C].In: IEEE Transactions on Parallel and Distributed Systems, 2007,18(10):1361-1376.

[8] SONG Jin-Woo, PARK Kyo-Sung, YANG Sung-Bong. An effective cooperative cache replacement policy for mobile P2P environments [C].In: IEEE International Conference on Hybrid Information Technology(ICHIT'06), 2006,2:24-30.

(收稿日期：2010-03-10)

作者简介：

季美丽，女，1985年生，硕士研究生，主要研究方向：P2P网络技术及应用。