分布数据缓存体系*

黄世能, 吴建清

(华南理工大学 计算机工程与科学系,广东 广州 510641)

E-mail: csjqxi@scut.edu.cn

http://www.scut.edu.cn

摘要: 讨论了一种在分布信息访问环境下提高数据利用率和减少通信流量的分布缓存体系.缓存节点能缓存多个数据源的信息,使多个用户相互重用数据缓存,从而提高缓存的命中率.该缓存体系采用了多种不同的方法来解决数据缓存的一致性问题.最后提出了一种虚缓存节点的概念,用于扩展原来的体系.虚节点可以减少全局缓存访问优化计算的成本.

关键词: 分布信息访问环境;信息;数据缓存;远程缓存节点

中图法分类号: TP311 **文献标识码**: A

本文讨论一种分布信息访问环境下的分布缓存技术.它可以有效地提高数据利用的效率并减少数据流量.

在分布信息访问环境下,存在信息访问端和信息发布端两种实体.信息发布端发布信息,信息访问端获取信息.记信息为 I,$I=f(I_1,I_2,\ldots,I_n)$,$I_i(1\leqslant i\leqslant n)$表示从信息发布端 S_i 发布的与 I 相关的信息,称为 I 的子信息,这些子信息经过函数 f 的转换和集成后得到用户需要的最终信息 I.可见,用户得到的信息是对信息发布端所发布信息的综合,而综合函数的执行可以在信息访问端或信息发布端完成.本文把用户获得信息内容的过程称为信息执行.分布信息访问环境的特点是:(1)信息发布端和信息访问端是高度分布的,其物理位置没有任何限制.(2)信息内容具有多源性,信息的内容可以分布在多个信息发布端上.(3)对实时性和一致性有多种要求.

传统上,数据缓存技术主要应用于服务器端,目的是减少 I/O 操作;也有一些客户端数据缓存技术[1,2],但它们研究的是单源数据的缓存问题.本文讨论的是信息分布环境下的通用缓存体系,目的是加快信息检索时间、减少数据传输和减少信息发布端的负载.

1 相关工作

数据缓存技术的主要问题是数据缓存的一致性问题,如果缓存数据与实际信息内容一致,则称该缓存数据有效,否则称为无效缓存数据.分布式环境下维护一致性的数据缓存机制目前有 Time-to-live fields(TTL)[3,4],Client pulling[4]和 Invalidation protocols[4]几种.

在 Time-to-live fields 机制中,每个对象被赋予一个有效期,比如两天或 12 小时,客户端可以在该有效期之内保留该对象的缓存数据,过了有效期之后,则认为缓存数据是无效的.Client pulling 技术是客户端周期性地向服务器询问缓存数据是否有效,客户端询问服务器的时间称为缓

* 收稿日期: 1999-09-28; 修改日期: 2001-03-14

基金项目: 广东省自然科学基金资助项目(109 B62750);广东省重点攻关计划资助项目(109-B25940)

作者简介: 黄世能(1976—),男,广东南海人,工程师,主要研究领域为数据库系统,分布访问技术,知识发现;吴建清(1962—),男,江苏江阴人,博士,教授,博士生导师,主要研究领域为数据库,网络信息发布.

存数据的更新时间,每个缓存数据记录最近的更新时间.确定更新时间的各种方法形成 Client pulling 的各种变种,最常见的是当客户请求该对象时询问服务器.而另一种应用较为广泛的变种起源于 Alex FTP Cache[5],它是基于这样一种假设:新的文件比旧的文件更频繁地被修改.更新时间由对象未被修改的时间和一个叫做陈旧因子的值(staleness)来决定.陈旧因子的定义[6]是

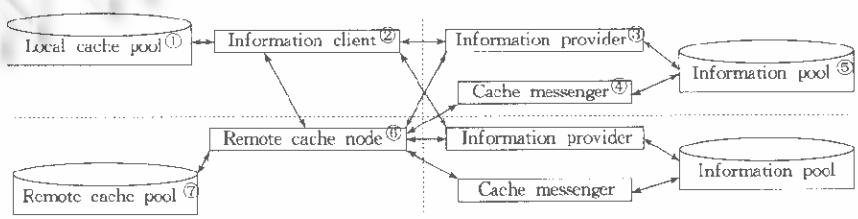
$$\text{Staleness}=\frac{\text{现在的时间}-\text{上次更新的时间}}{\text{上次更新的时间}-\text{上次修改的时间}}.$$

这意味着,如果某个文件 30 天内未被修改,3 天后它的陈旧因子为 10%,当陈旧因子大于某个预定义的值时,客户端就要询问服务器缓存数据是否有效.

Client pulling 技术和 TTL 是弱一致性的,它不能保证缓存数据与源数据在所有的时间内都一致,只能保证大部分时间内是一致的.保证缓存数据在所有时间内有效是强一致性问题.要保证强一致性,可以采用 Invalidation protocols 技术.与 Client pulling 不同,它是服务器端主动.客户端被动,服务器跟踪每个被缓存的对象,当该对象改变时,通知每个缓存该对象的客户端更新或丢弃相应的缓存数据.

2 信息分布访问缓存体系

分布信息访问环境下信息分布访问模型及其缓存体系结构如图 1 所示.



①本地缓存池,②信息访问端,③信息发布端,④缓存提示器,⑤信息池,⑥远程缓存节点,⑦远程缓存池.

Fig. 1 The model of information distributed accessing and its cache architecture

图1 信息分布访问模型及其缓存体系

这一结构中有 4 个主要组件:

(1) 信息访问端与信息发布端.信息访问端接受执行信息请求,并负责本地缓存池的管理.信息发布端负责执行由信息访问端发送过来的执行信息请求,搜索信息池,并返回相应的信息内容.

(2) 远程缓存节点.缓存数据管理器从信息访问端取回可以长期保存的缓存数据,保存到远程缓存池里,并负责远程缓存池里缓存数据的更新与淘汰.引入远程缓存节点是为了提高缓存命中率.由于分布信息访问环境是一个高度分布的系统,信息内容具有多源性,而且单个用户执行的信息是有限的,把缓存数据的重用局限在单个用户的范围内会大大减少缓存数据的命中率.因此引入了远程缓存节点,由它专门管理可以在用户之间重用的缓存数据,提高缓存数据的命中率.它可以与信息访问端或信息发布端在同一机器上,但在一般情况下,它作为一个单独的节点而存在.信息访问端上可以设置远程缓存节点的位置.

(3) 缓存提示器.信息发布端上的一个单独进程,维护一张缓存数据跟踪表,记录需要向其发送更新提示的远程缓存节点及相应的缓存数据标识,并监测信息池.如果信息池中的信息内容有改变,缓存提示器则根据缓存数据跟踪表的记录,发送更新提示到相应的远程缓存节点.

远程缓存节点的引进使得在该环境中存在两种数据缓存:

(1) 本地缓存池里的缓存,称为短期缓存.信息访问端将每一个符合条件(详见第 3 节)的信息内容放到由它管理的本地缓存池里.这些信息内容一直位于本地缓存池里,直到客户会话结束或被

用户删除为止.本地缓存池的缓存数据是私有的,信息访问端只能使用自己管理的短期缓存数据,而不能访问其他信息访问端上的短期缓存数据.

(2) 远程缓存池里的缓存称为长期缓存.该数据缓存是由远程缓存节点管理的.当信息访问端的一个会话结束后,在信息访问端上的本地缓存池中的所有缓存数据被转移到远程缓存池.长期缓存是共享的,任何能访问远程缓存节点的信息访问端都可以请求该远程缓存节点管理的长期缓存数据.

在该体系下,用户执行信息的流程如下:

(1) 信息访问端搜索本地缓存池,查看是否有相应的缓存数据,如果有,则返回该缓存数据.

(2) 否则,向指定的远程缓存节点发送查询缓存数据请求,远程缓存节点收到请求后,搜索远程缓存池,如果有相符合的缓存数据,则返回查询成功响应,并附上相应的缓存数据,否则返回查询失败响应.

(3) 信息访问端若接收到查询成功响应,则把所附的缓存数据作为信息内容;若接收到查询失败响应,则根据 I 的定义 $I=f(I_1,I_2,\ldots,I_n)$,向各信息发布端请求 $I_i(1\leqslant i\leqslant n)$,并执行函数 f,得到最终信息内容.

3 缓存的一致性维护

从上面的执行信息流程可以看出,信息访问端认为本地缓存池和远程缓存池里的缓存数据都是有效的,信息访问端本身不采取任何手段判断已有缓存数据的一致性.

对于缓存数据的一致性问题,对本地缓存池和远程缓存池采用不同的维护方法:

(1) 本地缓存池的一致性维护

基于用户的一个会话一般不会持续很久的假设,不采取任何措施去判断短期缓存的一致性,而认为它们是有效的.但需采用 TTL 技术避免更新频率过高的信息内容被放到本地缓存池里.设用户执行的信息为 $I=f(I_1,I_2,\ldots,I_n)$,各信息发布端对返回的信息内容 $I_i(1\leqslant i\leqslant n)$ 附上一个有效期域,记录 I_i 的有效时间(估计不会被更新的时间)记为 TTL_i,最终信息内容 I 的有效时间 $TTL=\mathrm{Min}(TTL_1,TTL_2,\ldots,TTL_n)$.当 TTL 小于用户预先设定的值时,信息访问端将不把该结果保存到本地缓存池里,否则将保存到本地缓存池里.另外,信息访问端也提供一个刷新功能,让用户强制信息访问端不到本地缓存池和远程缓存节点里取缓存数据,而到各信息发布端请求实际信息内容.

(2) 远程缓存池的一致性维护

远程缓存池里的缓存数据分为两种:一种标记为"已登记",表示在源信息发布端的缓存数据跟踪表上已登记了该缓存数据的信息;另一种标记为"未登记",表示源信息发布端没有该缓存数据的任何信息.对于"已登记"的对象的一致性,采用缓存无效协议(invalidation protocol)机制,而对于"未登记"的对象的一致性,则采用 Client Pulling 机制,由远程缓存节点管理员指定陈旧因子的最大值,记为 α.具体步骤如下:

· 在远程缓存节点端:

① 当用户与信息访问端的会话结束时,信息访问端会把本地缓存池上的内容传到指定的远程缓存节点 C,C 把它们放到远程缓存池里,并标记为"未登记",同时记录有效时间 TTL、产生时间 T_c 和更新时间 T_u,置 $T_u=T_c$.

② 对于 C 里每个标记为"未登记"的缓存数据 O,记 O 的源信息发布端为 $S_1,S_2,\ldots,S_n$,C 向每个 $S_i(1\leqslant i\leqslant n)$ 上的缓存提示器发送一个申请缓存请求包,请求包包括 O 的定义和请求跟踪时间

T_R.T_R 的值表示远程缓存节点请求缓存提示器跟踪该缓存数据更新变化的时间.

③ 若接收到各个源服务器允许缓存的响应,则把 O 标记为"已登记",并用返回的最小 T_R 值替代原来的 T_R 值;若接收到一个服务器的拒绝缓存请求,则不改变 O 的状态;若没有完全接收到各个服务器的响应,则认为是拒绝缓存请求.

④ 对于标记为"已登记"的缓存数据,若请求跟踪时间 T_R 期限已到,则把该对象标记为"未登记".

⑤ 对于每个"未登记"的缓存数据 O,按公式

$$\text{Staleness}=\frac{T_n-T_u}{T_u-T_c}\ (T_n \text{为当前时间})$$

计算 O 的陈旧因子.若陈旧因子大于 α,则发送询问请求到各源信息发布端询问变化情况,若没有变化,则置 $T_u=T_n$,否则,根据情况(详见第 5 节)更新 O 或丢弃 O.

⑥ 等待一段时间 T,返回第②步重新执行.

⑦ 若接收到缓存提示器发送过来的更新提示,则根据情况更新或丢弃相应的缓存数据.

·在缓存提示器端:

① 如果接收到远程缓存节点 C 发送过来的申请缓存请求包,若服务器的负载允许跟踪该缓存数据,则缓存提示器根据情况确定跟踪时间 T_R,把缓存数据的标识、缓存数据基于的信息池内容、远程缓存节点的网络地址及 T_R 记录到缓存数据跟踪表里(T_R 表示这些跟踪信息在缓存数据跟踪表里存在的时间),返回允许缓存的响应,并附上 T_R 的值.否则,返回不允许缓存的响应.

② 如果检测到信息池里的信息 I_b 发生变化,则扫描缓存数据跟踪表,若存在基于 I_b 的缓存数据标识,则发送一个更新提示到相应的远程缓存节点上.

③ 每隔一段时间,扫描缓存数据跟踪表,若存在 T_R 时间已到的记录,则删除该记录.

4 缓存体系的性能分析

假设信息定义为 $I=f(I_1,I_2,\ldots,I_n)$,信息访问端与信息发布端 $S_1,S_2,\ldots,S_n$ 的网络连接速度分别是 $k_{S_1},k_{S_2},\ldots,k_{S_n}$(字节/秒),远程缓存节点与信息发布端 $S_1,S_2,\ldots,S_n$ 的网络连接速度分别是 $k'_{S_1},k'_{S_2},\ldots,k'_{S_n}$(字节/秒),信息访问端与远程缓存节点的网络连接速度是 k_c 字节/秒.I_1,$I_2,\ldots,I_n$ 的平均大小为 $m_1,\ldots,m_n$ 个字节,而 I 的平均大小是 m_c 个字节,信息访问端每天有 q 个执行查询请求.忽略发送执行信息请求和更新提示的时间及执行函数 f 的时间,并假定以串行方式执行信息.

若没有采用缓存技术,则 q 次执行信息的时间为

$$T_a=q*(m_1/k_{S_1}+m_2/k_{S_2}+\ldots+m_n/k_{S_n}).$$

若采用了本文所讨论的缓存体系,且 q 次执行信息都能命中远程缓存池上的缓存数据,设远程缓存池中的缓存数据的平均更新频率是 f 次/天,则

执行信息的时间为

$$T_1=q*m_c/k_c,$$

更新缓存的时间为

$$T_2=f*(m_1/k'_{S_1}+m_2/k'_{S_2}+\ldots+m_n/k'_{S_n}),$$

总时间为

$$T_b=T_1+T_2,$$

前后时间之比为

$$y=T_a/T_b=\frac{q*(m_1/k_{S_1}+m_2/k_{S_2}+\ldots+m_n/k_{S_n})}{q*m_c/k_c+f*(m_1/k'_{S_1}+m_2/k'_{S_2}+\ldots+m_n/k'_{S_n})}.$$

设信息访问端与各服务器之间的网络连接速度相同,即 $k_{S_1}=k_{S_2}=\ldots=k_{S_n}=k_S$,远程缓存节点与各信息发布端之间的网络连接速度相同,即 $k'_{S_1}=k'_{S_2}=\ldots=k'_{S_n}=k'_S$,由于远程缓存节点一般比较靠近信息访问端,因此假设 $k_S=k'_S$,则

$$y=\frac{q*(m_1+\ldots+m_n)/k_S}{q*m_c/k_c+f*(m_1+\ldots+m_n)/k'_S}=\frac{q*(m_1+\ldots+m_n)/k_S}{q*m_c/k_c+f*(m_1+\ldots+m_n)/k_S}.$$

记缓存数据更新频率与用户执行信息频率之比为 $a=f/q$,子信息 $I_i(1\leqslant i\leqslant n)$ 大小之和与 I 的大小之比为 $b=(m_1+\ldots+m_n)/m_c$,信息访问端与远程缓存节点和信息访问端与各信息发布端的网络连接速度之比为 $c=k_c/k_S$,则

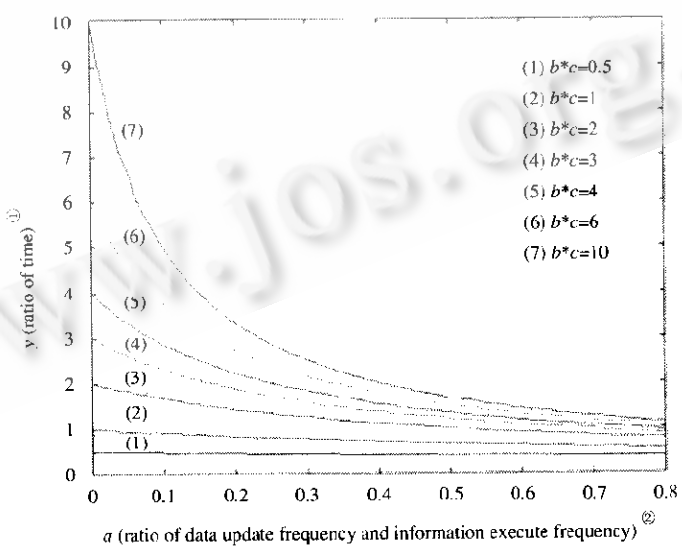
$$y=\frac{1}{\frac{1}{b*c}+a}.$$

图 2 是在不同的 $b*c$ 值下,y 与 a 之间的函数曲线图.从图中可以看出:

(1) y 随 a 的增大而减少,随 $b*c$ 的增大而增大.

(2) $b*c$ 的值越大,曲线就越陡峭,y 受 a 的影响越大.

可见,当 $a\leqslant 0.25$ 且 $b*c\geqslant 4$ 时,该缓存体系能使分布信息访问环境的性能提高两倍以上.因此,要使该缓存体系得到好的性能:(1) a 必须足够小,即缓存数据的更新频率要远小于用户执行信息频率,如果缓存数据的更新频率过高,就要丢弃该缓存数据;(2) 信息访问端应该访问与自己的网络连接速度较快的远程缓存节点;(3) 远程缓存节点不应保留最终信息内容比子信息内容之和大的缓存数据.



①y(时间比),②a(更新频率与执行信息频率之比).

Fig. 2 The function curve of y and a

图 2 y 与 a 的函数曲线图

5 远程缓存池的维护

远程缓存节点上的磁盘资源是有限的,不能无限制地满足缓存的需要,必须淘汰一些缓存数据.因此在每个缓存对象上附有一个叫做有用度的值 M,设定一个有用度的最低阈值 M_{min},当缓存对象的 $M \leqslant M_{min}$ 时,表示应该丢弃该缓存对象.根据第 4 节的分析,有用度应与 y 值有关,y 越大,M 就越大,反之亦然.同时,M 也应与缓存数据的驻留时间有关,所以

$$M = y + m * R^r,$$

其中参数 m 和 r 规定了 y 和 R 之间的相对重要性.

由于信息访问端与远程缓存节点的网络连接速度和信息访问端与各信息发布端的网络连接速度之比在远程缓存节点无法得知,因此用参数 s 替代,由管理员根据系统环境来设定.记缓存数据更新频率与用户访问缓存频率之比为 A,中间信息内容与最终信息内容的大小之比为 B,因此

$$M = \frac{B}{s + A * B} + m * R^r (s, m, r \text{ 为管理员设定的参数}).$$

为了能及时淘汰缓存数据,在以下两种情况下要计算 M 值:

(1) 每隔一段时间,远程缓存节点都会计算缓存池里每个对象的 M 值,如果小于最低阈值,则丢弃该缓存数据.

(2) 当判定某个缓存数据为无效时,计算该数据的 M 值,如果小于最低阈值,则丢弃该缓存数据,否则更新该缓存数据.

6 虚拟缓存节点

有些访问节点(信息访问端节点和缓存节点)在物理上较为紧密地相连或具有宽的互联带宽,这些节点的访问行为可能有较大的相似性.例如,在一个单位的内部网中往往是这种情形.按照上述的缓存方法,这些节点的缓存池内可能有同样的缓存信息.为了提高存储效率,把这些缓存池组织成一个局部的缓存系统,该缓存系统把这些物理上分散的缓存池作为逻辑上一体的缓存池进行管理和数据访问控制,进一步提高缓存数据的利用率,减少主干网络或远程数据传输量以及刷新消耗.这样的局部系统在更大的缓存系统内称为虚拟的缓存节点.用 BN 表示缓存节点,VCN 表示虚拟缓存节点.VCN 的引入意味着以下几点:

(1) 各种局部缓存信息可以被同一个 VCN 中的其他节点所共享;

(2) 多个 VCN 和 CN 可以构成更大的 VCN;

(3) 访问端在访问信息时,首先在本地缓冲池中查找,其次是在所处的各级 VCN 中查找,最后才访问信息发布端.

在实现带有 VCN 的缓存系统时,有多种实现策略可以选择.例如,一个访问节点在搜索所在的 VCN 中的信息时,既可以采用广播方式查询,也可以在一个中央信息目录中查询,中央信息目录登记了该 VCN 中各个缓存信息的详细物理路径.各级 VCN 的设立,可以是静态的,也可以是动态的.每个 VBN 对应的缓存数据,可以是各个低层 VCN 和 CN 的缓存数据的交集,或者根据每个缓存数据的访问概率、网络速度和访问端点的分布来确定是否进入上层的 VCN.我们在以后的工作中将进一步展开这方面的研究.

References:

[1] Wilkinson, K., Neimat, M. A. Maintaining consistency of client-cached data. In: Proceedings of the 16th International Conference on Very Large Data Bases. 1990. 122~133. http://dblab.ce.cnu.ac.kr/dolphin/db/conf/vldb/Wilkinson-N90.html.

[2] Wang, Y., Rowe, L. A. Cache consistency and concurrency control in a client-server DBMS architecture. In: Proceedings of the 1991 ACM SIGMOD International Conference on Management of Data. 1991. 367~376. http://dblab.ce.cnu.ac.kr/~dolphin/db/conf/sigmod/sigmod91.html.

[3] Luotonen, A. Proxy caching. 1994. http://www.w3.org/hypertext/WWW/Daemon/User/Config/Caching.html.

[4] Gwertzman, J., Seltzer, M. World Wide Web cache consistency. In: Proceedings of the USENIX 1996 Annual Technical Conference. 1996. 22~26. http://www.eecs.harvard.edu/~vino/web/usenix.196/.

[5] Cate, V. Alex——a global filesystem. In: Proceedings of the 1992 USENIX File System Workshop. 1992. 1~12. ftp://alex.sp.cs.cmu.edu/doc/intro.ps.

[6] Wessels, D. Intelligent caching for world-wide web objects [MS. Thesis]. Boulder, Colorado: University of Colorado, 1995. http://itp-www.colorado.edu/.

Distributed Data Cache Systems*

HUANG Shi-neng, XI Jian-qing

(*Department of Computer Engineering and Science, South China University of Technology, Guangzhou* 510641, *China*)

E-mail: csjqxi@scut.edu.cn

http://www.scut.edu.cn

Abstract: A distributed cache system in distributed information access environment is discussed in this paper, which is used to increase the access efficiency and reduce the communication throughout. The cache node can store the data from multiple data sources. Different access of information can reuse the same cached data and thus improve the hit rates. The cache system adopts various methods to keep the consistency of cached data. Finally, concept of virtual cache node is proposed, which extends the original caching system. The adoption of virtual nodes is for reducing the amount of replicated cached data and reduce the computation cost for a global optimized accessing plan computation.

Key words: distributed information access environment; information; data cache; remote cache node

* Received September 28, 1999; accepted March 14, 2001

Supported by the Natural Science Foundation of Guangdong Province of China under Grant No. 109-B62750; the Key Sci-Tech Project of Guangdong Province of China under Grant No. 109-B25940