并行文件系统 PARFSNOW + + 中的 Cooperative Cache 技术的研究

赵 欣,陈道蓄,谢 立
(南京大学计算机软件国家重点实验室,南京 210093)

摘 要: 本文介绍了并行文件系统 PARFSNOW + + 中采用的有权限控制的复合协作式缓冲管理机制,并针对当前协作式缓冲机制中的问题,提出了新的全局内存数据同步和数据替换策略. 它们能有效地提高全局内存数据同步和缓冲数据替换的效率. 通过实验,本文证明了引入这种改进的协作式缓冲机制后,并行文件系统效率得到了提高.

关键词: 工作站网络; 并行文件系统; 协作式数据缓冲; 内存一致性规范
中图分类号: TP316.8 **文献标识码**: A **文章编号**: 0372-2112 (2000) 09-0131-04

The Cooperative Cache Techniques Used in PARFSNOW + +

ZHAO Xin, CHEN Dao-xu, XIE Li
(*State Key Lab. of Software, Nanjing University, Nanjing* 210093, *China*)

Abstract: This paper describes the authorization - controlled cooperative cache mechanism used in the NOW-based parallel file system PARFSNOW + +. To solve the problems of current cooperative cache mechanisms, the paper presents a new cache coherence technique and a new data replacement technique. These new techniques can be used to improve the efficiency of the cooperative cache system. Through some test results, this paper also demonstrates that these new techniques indeed improve the performance of PARFSNOW + +.

Key words: NOW; PFS; cooperative cache; MCS

1 引言

当前,许多基于 NOW(Network of Workstation)[1]的并行文件系统都采用了 Cooperative Cache(协作式数据缓冲)机制,如 Sprite 系统等[2]就是这样. Cooperative Cache 是一种数据缓冲机制,它能协调并行环境中各个节点上数据缓冲的内容,使它们能互相协作,将多台工作站上的局部内存通过网络互联成为一个虚拟的全局共享地址空间,提供给系统使用者一个平坦的共享地址空间,以掩盖使用网络上其他主机上内存的底层实现细节,使用户能象访问本地内存一样访问这种分布式共享内存. 使用这种机制,并行文件系统能突破 I/O 节点上本地数据缓冲区容量的限制[3],利用网络共享缓冲区来扩大数据缓冲区,从而提高文件数据在缓存中的访问命中率,减少磁盘的访问次数,从而达到提高系统性能的目的. 但是,当前的协作式高速缓冲机制在其关键操作:缓冲数据的同步和替换上仍存在着性能不高,操作复杂的问题,难以有效提高并行文件系统的 I/O 效率.

本文提出了一种有权限控制的复合式 Cooperative Cache 管理机制,并在此基础上设计了一种简单而高效的内存数据一致性方法和一种基于生命周期的数据替换策略. 通过这些新技术,能有效地提高 Cooperative Cache 系统的效率. 现在,这种复合式 Cooperative Cache 管理机制已应用在我们设计并实现的并行文件系统 PARFSNOW + + 中. 通过实验,文章证明了文中提出的 Cooperative Cache 机制的有效性.

2 PARFSNOW + + 模型简述

PARFSNOW + +[4]是一种基于 NOW 并行文件系统. 系统中每个节点(包括 IBM RS6000 工作站和 PC 机)都通过高速网络互相连接,并运行各自的基于微内核的操作系统,它们包括 IBM AIX4.2 和 WinNT4.0. 并行文件系统则构建于这些操作系统平台之上,作为服务程序向用户提供并行文件服务. PARFSNOW + + 的结构如左图,它包括文件服务器,I/O 服务器,计算节点即用户程序三部分. 并行文件系统通常将文件分片存放,便于并行读写. I/O 服务器(简称 I/O 节点)是负责存放、管理各并行文件分片,并向用户提供对文件分片的读写服务的. 文件服务器是负责并行文件分片信息的存储和管理、文件操作任务的协调等任务的. 当用户要求打开某并行文件时,系统会自动连接文件服务器,将对应的文件分片信息读出,返回给用户和并行文件系统,便于用户和系统的控制. PARFSNOW + + 将各 I/O 节点的内存共享出来,通过 Cooperative Cache 机制构成一个共享的扩大的文件数据缓冲区,来提高数据在缓存中的命中率,减少低速的磁盘访问,从而达到提高系统效率的目的. 所以,Cooperative Cache 机制是否高效直接关系到系统的运行效率. PARFSNOW + + 采用了一种新的有权限控制的复合式Cooperative Cache管理机制,有效的提高了系统效率.

收稿日期:1999-07-05;修回日期:1999-12-20

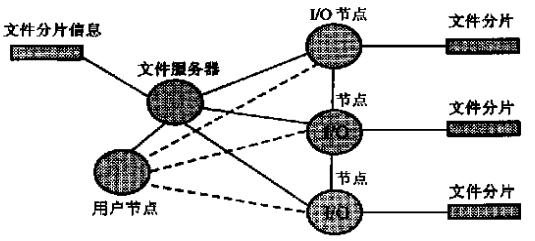

图 1 PARFSNOW + + 结构图

3 复合的 Cooperative Cache 技术

3.1 现有的 Cooperative Cache 管理方式存在的问题

过去提出的 Cooperative Cache 系统往往采用两类 Cache 管理模式:基于多备份的分布式共享内存管理模式和基于单备份的分布式共享内存管理模式. 基于多备份的分布式共享内存管理模式允许对同一个数据块在全局内存中维护多个对等的数据备份,用户节点能在本地数据备份中直接读写数据,从而能减少用户到远地缓存中访问数据的次数,降低网络数据传输的开销,提高数据访问效率. 但是,其复杂的数据同步操作开销巨大,影响了系统的效率的提高. 另一种模式就是基于单备份的分布式共享内存管理模式. 这种模式基于减少数据一致性维护代价的考虑,采用了全局数据单备份机制 COMA[5]. 即仅维护一个数据缓冲备份,没有主、副备份之分. 当某用户访问数据时,就必须访问对应的远地数据缓冲备份,并将数据取回本地. 但大量网络传输给网络带来沉重的负担,也会影响数据访问效率. 由于存在这些问题,所以设计一个新的 Cooperative Cache 管理机制就是非常重要的了.

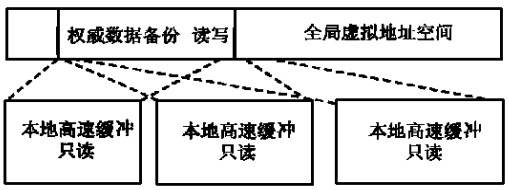

图 2 复合式 Cooperative Cache 管理模式图

3.2 有权限控制的复合 Cooperative Cache 管理模式

针对现有的 Cooperative Cache 管理技术存在的问题,提出有权限控制的复合 Cooperative Cache 管理机制(以下简称"复合模式"). 在复合模式下,所有节点都将本机上的部分内存提供出来,通过网络互联,构成一个全局虚拟共享内存. 其具体结构如图 2,系统在全局虚拟地址空间内为共享数据块保留且只保留一份权威数据备份. 权威数据备份是具有可修改权限的,允许用户在上面进行修改操作,并定期自动将更新数据刷新到文件中去,从而能确保是相应的文件数据块的最新数据备份. 系统在用户本地还维护了一个本地高速缓冲,这个缓冲同权威缓冲数据备份相映射,使得用户能尽可能地在本地读取数据,而不必到远地缓冲数据备份中读取数据,从而有效提高数据访问效率,减轻网络负载. 但是,这种本地高速缓冲没有修改权限,所以,用户若要修改文件数据,就必须到对应的权威数据备份中进行修改. 系统并不向其他节点通告修改操作的发生. 由上所述,整个系统实现了一种层次型的缓冲机制:当用户要读取某块数据的时候,首先在本地缓冲中查找数据,若本地缓冲有该数据并确实同权威的缓冲数据备份相一致,那么用户就在本地读取数据. 当本地缓冲没有该数据或本地缓冲数据同权威的缓冲数据备份已经不一致(由于用户对权威缓冲数据备份进行修改)的时候,系统就自动到存放在全局共享内存中的权威缓冲数据备份中取得当前最新数据来更新本地高速缓冲,并提交给用户. 当然,由于内存空间有限,所以,当虚拟地址空间中数据难以保证全部存放在内存中时,系统就要采用操作系统的虚存管理机制配合上数据替换策略来处理难以完全放于内存中的数据,或者抛弃较少使用的缓冲数据以腾出空间,或者用虚存管理机制将数据保留在磁盘上. 由于复合模式在本地高速数据缓冲和权威数据备份上引入了权限控制机制,只允许对权威数据备份进行修改,并且不通告数据修改状况,从而能有效减少数据修改通告和数据同步的开销,系统也不需要时刻维护每个本地缓冲的状态,这样能降低系统实现的复杂度. 但是,由于这种机制,权威数据备份同本机高速缓冲在一定时间内会出现数据不一致的状态. 为了保证用户数据访问时,本地缓冲同权威数据备份的数据保持一致,设计了一种主动式内存同步策略.

3.3 内存主动同步机制

由于采用复合模式,用户既可以从权威缓冲数据备份中读取数据,又可以从本地缓冲中读取数据. 显然,从本地高速缓冲中读取数据从远地的权威数据备份中读取数据开销要小. 所以,用户将尽可能从本地高速缓冲中读取数据. 但是,由于系统不会对权威数据备份的修改进行通告,这就可能造成权威数据备份同本地高速缓冲数据的不一致状态. 所以,必须实现一套高效内存一致性规范 MCS. 但是,我们看到的常见内存同步策略中往往存在一些问题:(1)不必要的一致性检查和数据维护操作. 系统运行中,只要在使用本地缓冲时,它的数据同权威数据备份一致就可以了. 但许多系统中每次修改操作后都要进行全局同步操作,带来了不必要的一致性操作延迟. (2)操作通告代价高昂. 通常的 MCS 系统每次数据修改操作后,就需要通告各相关运行进程,来维持系统数据的一致性. 这种通告代价高昂,严重影响了系统效率.

通过对实际应用的分析,我们发现,许多时候用户虽然保留了某块数据的本地高速缓冲,但是并不一定会立即使用,若在用户使用该高速缓存的数据前,有另外的用户对权威数据备份进行多次修改,那么,由于数据一致性维护的需要,本地高速缓冲就必须一次次更新. 这实际上是没有必要的. 实际上,本地高速缓冲只需要在用户访问前更新一次就可以了. 而且许多时候系统虽然维护了该本地缓冲,但可能用户不会再使用到它,所以,不顾数据是否会被继续使用而持续进行数据更新是对有限的系统资源的浪费. 所以,只有在必要的时候才进行数据同步将会更加有效. 基于这种认识,设计了"主动一致性方法".

这种"主动一致性方法"是由两部分思想组成的. 第一,权威数据备份和本地高速缓冲分别同时间戳相联系. 时间戳是一个逻辑非负数,用来描述两个动作之间的先后关系. 在"主

动一致性方法"中有三种时间戳:系统时间戳、权威数据备份时间戳 *Ta*、本地高速缓冲时间戳 *Tl*. 系统时间戳是 I/O 节点在初始化的时候创建的. 每次对权威数据备份进行更新后该系统时间戳就会增大 1. 权威数据备份时间戳 *Ta* 等于权威数据备份被更新时的系统时间戳. 本地高速缓冲时间戳 *Tl* 则等于它从权威数据备份中获得数据时的权威数据备份时间戳 *Ta*. 由于每次对权威数据备份进行修改都会增大权威数据备份时间戳. 这样,通过比较本地高速缓冲的时间戳 *Tl* 和权威数据备份的时间戳 *Ta*,就能判定权威数据备份是否已经被修改过. 若 $Ta = Tl$,则表明当前高速数据缓冲内的数据就是对应文件数据块的最新数据. 此时可以直接使用本地数据. 若 $Ta > Tl$,则表明由于用户修改,该高速缓冲对应的文件数据块已经被刷新过,而本地高速缓冲却还未刷新,则本地高速缓冲中的数据是失效的数据. 此时系统会根据权威数据备份向用户本地传送最新数据,并刷新本地高速缓冲. 第二,"使用时主动同步". 即用户要使用缓存于本地高速缓冲的某数据块的时候,才主动向权威数据备份发出数据同步请求. 系统在请求中附上本地高速缓冲的时间戳,这种同步报文非常小,所以系统开销不大. 当权威数据备份收到该同步请求时,它通过比较请求中附的时间戳和自身时间戳,就能确定用户本地缓冲中的数据是否是最新的. 若是,则发送"无需更新"的回答. 若不是,则直接发送更新后的数据. 这样,在权威数据备份上进行"写"操作后,系统允许出现临时性的数据不一致. 在用户访问本地缓冲时,它才主动地询问当前数据状况并进行同步操作,而文件缓冲数据的更新无需向每个相关进程发送更新通告,能省去许多通告开销.

这种主动式方法能有效支持临时性的数据不一致状态,减少系统通告操作的开销,提高系统操作的效率. 最重要的是它省去了许多不必要的一致性维护的开销. 只有那些要用到这块数据的进程才会提出 synchronize 操作请求,并获得最新通告. 也只有那些通过 synchronize 操作请求的数据才参与数据一致性维护操作,从而进一步减少了数据的不必要的共享操作.

3.4　缓冲数据替换

通常情况下,权威数据缓冲备份保存在该备份所对应的文件所在的 I/O 节点上. 但是,当 I/O 节点被大量访问的时候,仍然可能出现本地内存不足以保存所有的权威数据缓冲备份的问题. 此时,就必须要考虑将缓冲内存中的某些页面替换出去. 过去常用的技术是将缓冲数据替换到磁盘上,由于磁盘速度较低,所以替换到磁盘上的操作往往效率不高. 利用 Cooperative Cache 提供的服务,权威数据缓冲备份可以在全局内存空间中自由迁移. 这意味着缓存的数据在内存中的位置可以是动态调整的. 这种机制非常有利于并行计算的工作模式. 它能在某个节点内存不够时,将某些缓冲数据迁移到其他节点上去. 我们注意到,在高速网络环境下,当本地内存不够缓冲所有需要缓冲的数据的时候,把数据缓存在其他空闲节点比将之替换到磁盘中更能有效提高性能. 特别是对于某些节点使用数据过多的情况下,配合有效的替换算法,其他节点实际上就变成了此节点的文件服务器,能将对单个节点的 I/O 请求分流.

要将数据缓冲备份替换到它机上,就涉及两个问题:第一,应该将哪个页面替换到其他节点上去的问题. 第二,由于一个权威数据缓冲备份不应该在没有用的时候仍然占用内存空间,所以存在选择将一个权威数据缓冲备份抛弃的合适时间的问题. 对于替换页面的选择问题,采用的是一种改进的 LRU 算法[6]. 该算法在节点内部维护了一个 LRU 先进先出队列,当节点创建了某文件数据的权威数据备份后,就将该数据备份的索引加到队列尾部. 当某权威数据备份被访问后,系统就从该队列中将这个权威数据备份块对应的索引从原来位置删除,然后添加到队列尾部. 这样,就可以保证队列首部是当前最少访问的权威数据备份的索引. 通常的 LRU 算法是将当前最少用的缓冲数据块替换出去,但是,由于并行文件系统存在着"延迟写"机制,所以当前可能存在已经被修改但是尚未刷新到磁盘上的"脏"的缓冲数据块. 为了提高将"脏"缓冲数据块刷新到磁盘上的效率,应该尽可能不将这种"脏"缓冲数据块替换到其他节点上. 根据上面的原则,当需要进行缓冲数据替换的时候,系统就从该 LRU 队列首部开始向尾部查找,寻找第一个"干净"的权威数据备份块,将它迁移到一个具有空闲内存资源的节点上,并修改本机上对该文件数据块的索引,使它指向接受该权威数据备份的节点. 而接受这个权威数据备份块的节点则将该迁移来的权威数据备份块添加到它自身的 LRU 队列尾部. 当然,一个权威数据缓冲备份不应该在没有用的时候仍然保存在内存中,占用内存空间,所以,每个权威数据缓冲备份块都有一个 *TTL*(Time To Live)域,缺省情况下,*TTL* = 0,每次将权威数据缓冲备份替换到其他节点都要使它的 *TTL* 域加 1,当 *TTL* 达到阈值 *N* 后,系统就不会再替换该数据块,而只是简单抛弃这块权威数据备份块. PARFSNOW + + 设定 $N = 2$.

4　性能评估

为了测试上面各项技术的效果,建立了一个试验环境,该环境由多台 IBM RS6000 工作站组成每台工作站的内存为 64M,硬盘 2G,其上运行的操作系统是 AIX4.2.1.

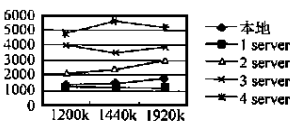

图 3　文件写操作性能测试图

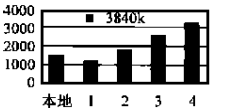

图 4　文件读操作性能测试图

上面图 3 显示了传输大小为 1200kB、1440kB、1920kB 时写操作的数据传输率(kB/s)测试图.

上面图 4 显示的是投入多个 I/O 节点读 3840kB 数据的数据传输率(kB/s)的直方图.

它们表明采用了协作式缓冲的并行文件系统通过并行磁盘读写,有效提高了文件读写操作的效率.

图 5 描述了两个用户的随机共享读写,其中读写的记录及次数均随机产生,1/3 为写操作,2/3 为读操作. 图中纵坐标是两个用户各读写 100 次的时间(s). 其中横坐标是 I/O 节点的 Cache 大小(单位 kB). 从图中可以看出,在复合 Cooperative

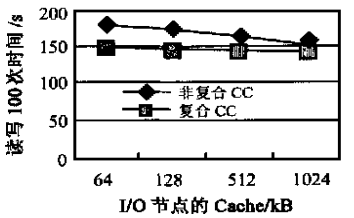

图 5 两用户随机共享读写

Cache 机制下,系统充分使用了网络连接的局部内存,使得系统读写性能明显优于没有 Cooperative Cache 机制时的系统性能.由于 Cooperative Cache 机制充分利用了网络共享内存,所以数据缓冲区基本是够用的,故单机 Cache 的增大并不会对系统性能有所改变,系统性能较为稳定.而无 Cooperative Cache 机制的共享访问时间则随 Cache 的增大逐渐减少,显示出 I/O 节点靠本地缓冲难以保证数据的充分缓冲,所以当单机 Cache 增大到能满足系统缓冲要求时,系统性能较好,但若 Cache 不足时,系统性能就较差.由于单机内存有限,若无法充分利用它机内存,将难以应付大量数据缓冲的要求.这显示出复合 Cooperative Cache 机制能有效提高系统资源利用率,提高系统效率.

5 结束语

本文介绍了基于 NOW 的并行文件系统 PARFSNOW + + 的设计中涉及 Cooperative Cache 技术.经过测试表明,这些技术的引入确实提高了并行文件系统的效率.

参考文献:

[1] T. E. Anderson, M. D. Culler, D. A Patterson et al. A case for NOW (Networks of Workstations) [J]. IEEE Micro, February 1995:54 - 64.

[2] M. Nelson, B. Welch and, J. Ousterhout. Caching in the sprite network file system [J]. ACM Trans. on Computer Systems, February 1988, 6(1):45 - 55.

[3] D. Kotz and, N. Nieuwejaar. Flexibility and Performance of Parallel File Systems [J]. ACM Operating Systems Review, April 1996, 30(2):63 - 73.

[4] 李群,谢立,孙钟秀. 并行文件系统的设计 [J]. 计算机科学, 23(4):36 - 39.

[5] Erik Hagersten et al. DDM-A Cache-Only Memory Architecture [J]. IEEE Computer Sep. 1992:44 - 54.

[6] A. Leff, J. L. Wolf, S. Yu. Efficient LRU-Based Buffering in a LAN Remote Caching Architecture [J]. Transactions on Parallel and Distributed Systems, February 1996, 7(2):191 - 206.

作者简介:

赵 欣 1974 年出生. 1996 年获南京大学计算机系学士学位, 1998 获南京大学计算机系硕士学位. 现在南京大学计算机系攻读博士学位. 研究方向:网络与分布式系统. 发表文章 10 余篇.

陈道蓄 1982 年毕业于南京大学. 现任南京大学计算机系教授,系主任. 主要研究方向:分布式处理和并行计算. 在国内外发表论文 30 余篇,合作出版著作 2 本,并 4 次获教育部和江苏省科技进步奖.

谢 立 1964 年毕业于南京大学. 现任南京大学计算机系教授,南京大学副校长. 主要研究方向:分布式处理和并行计算. 在国内外发表论文 100 余篇.

(上接第 122 页)

参考文献:

[1] D. Hush, B. Horne. Progress in supervised neural networks [J]. IEEE Signal Processing Magazine, January 1993:8 - 39.

[2] J. Park, I. W. Sandberg. Approximation and radial basis function networks [J]. Neural Computa., 1993, 5(1):305 - 316.

[3] J. Park, I. sandberg. Universal approximation using radial basis function networks [J]. Neural Computa., 1991, 3(2):246 - 257.

[4] N. B. Karayiannis, G. W. Mi. Growing radial basis neural networks: merging supervised and unsupervised learning with network growth techniques [J]. IEEE Trans. Neural Networks, 1998, 8(6):1492 - 1506.

[5] S. Chen, C. F. N. Cowan, P. M. Grant. Orthogonal least squares learning algorithm for radial basis function networks [J]. IEEE Trans. Neural Networks, 1991, 12(2):302 - 309.

[6] S. Mallat, Z. Zhong. Matching pursuits with time-frequency dictionaries [J]. IEEE Trans. Signal Processing, 1993, 41(12):3397 - 3415.

[7] R. Parisi, E. D. D. Claudis, G. Orlandi, B. D. Rao. Fast adaptive digital equalization by recurrent neural networks [J]. IEEE Trans. Signal Processing, 1997, 45(11):2731 - 2739.

[8] S. Chen, G. J. Gibson, C. F. N. Cowan, P. M. Grant. Reconstruction of binary signal using an adaptive radial-basis-function equalizer [J]. Signal Processing, 1991, 22(1):77 - 93.