第7卷第4期
1996年4月

软 件 学 报
JOURNAL OF SOFTWARE

Vol. 7, No. 4
Apr. 1996

DSM中一种新型Cache一致性管理算法*

房至一　鞠九滨

(吉林大学计算机科学系　长春　130023)

摘要　存储器一致性管理是分布式共享存储器DSM(distributed shared memory)系统的一个重要问题.在基于目录和所有者管理一致性的DSM系统中,如何适时地更新所有者链表以及目录中关于所有者的信息是缩短查表时间的关键.本文介绍一种新型的链表更新算法的设计及其性能分析.分析表明,这种方案对维护存储器一致性来说,具有较灵活的适应性并有助于缩短查表时间,提高系统性能.该算法也可适用于树形层次结构的一致性管理方案.

关键词　高速缓冲存储器,一致性,链表,分布式共享存储器.

构造一个DSM(distributed shared memory)系统的重要问题之一是解决存储器一致性问题.高速缓冲存储器(cache)是一种快速局部存储器,根据局部性原理,它可以拥有最近最常使用的数据副本,从而可以适当地减少对存储器带宽的要求并降低访问延迟.[1]因此,无论是在紧密耦合多机系统中还是在松散耦合分布式计算系统中大都使用了cache机制.这样就使得存储器的一致性问题变成了cache的一致性问题.由于读操作不会引起cache的一致性的改变,因此,一致性的管理都是针对写操作而言.采用的措施可以有:(1)静态定位方式,这种方式简单,但有瓶颈问题;(2)动态迁移方式,这种方式允许数据动态分布,但数据定位困难,在使用集中式服务员服务时会降低并行性和增加系统负担;(3)广播请求式,缺点是系统的伸缩性不好,而且无关节点也要处理该请求.为了避免上述方案的不足,现有DSM系统大多采用基于所有者的分布方案.也就是说,在一个DSM系统中,一个页可以有多个只读型副本但却只能有一个可写副本,这个可写的页拥有一个所有者,它可以对该页进行读/写操作.缺点是当所有者发生变更时,为查找现行所有者可能要进行若干次操作.由此可见,如何管理这个所有者是维持存储器一致性的关键.

对所有者的管理通常是通过目录表的形式实现的.具体划分可以有3种类型,即全匹配型表、有限表和链表.[1]全匹配型表为每个页维持2个状态位.一位标明该页是否有效,另一位标明该页是否可写.这种方案可支持顺序型一致性,但系统的可伸缩性不好.有限表是为解决表的规模问题而设计的.这个方案是基于存储器的局部性原理,把某块数据副本的数量限定为一个常数,使之与系统中处理机数量的增加无关.链表方案通过维护一个链表指针

* 本文研究得到国家自然科学基金资助.作者房至一,1957年生,讲师,主要研究领域为分布计算系统.鞠九滨,1935年生,教授,博士导师,主要研究领域为分布计算系统.

本文通讯联系人:房至一,长春 130023,吉林大学计算机科学系

本文1995－02－25收到修改稿

(单向的或双向的)实现数据副本的共享管理.由于这种方案实现起来灵活方便且有利于系统的伸缩,因此使用得较为广泛.

本文首先介绍关于cache一致性已有的研究工作,接着分别介绍基于链表管理方案实现cache一致性的主要相关工作并加以分析比较,然后提出一个新型的所有者链表管理方案——RTUTH(real-time updating list head).

1 相关工作

针对cache一致性的研究工作已有很多报道.文献[1]指出cache一致性的最好解决办法是使用软硬件结合的策略,并介绍了3种基于目录实现的cache一致性方案;文献[2]介绍了一种由软件辅助实现的、基于时钟和时间戳的cache一致性方案;文献[3]提出将所有分布式存储器都组成大规模的cache存储结构,并介绍了一种cache一致性的硬件模型;文献[4,5]介绍了有效地共享多副本数据的硬件和软件实现方案;文献[6]评估了4种cache一致性协议(写一废除;读一广播;写一广播;竞争侦听协议)的性能;文献[7,8]介绍了cache一致性机制的分层结构;文献[9]介绍了建立在虚拟存储器基础上的由系统软件实现的cache一致性.

基于链表的cache一致性管理可以分为双链型和单链型.典型代表是IEEE SCI标准协议[10]和单链分布式目录(SDD)协议.[11]

1.1 IEEE SCI标准协议

可伸缩的一致性接口(SCI)是IEEE的一种标准(P1596)协议.它是一种基于分布式目录的双链型协议.对每个数据块地址、存储器和cache的入口都有附加的标志位.存储器标志位用来识别共享表中第一个处理机(称为表头);cache标志位用来识别共享表的先前的和后续的入口.

开始时,共享表是空的,存储器处于未被缓存状态,cache中的副本也是无效的.当发生读操作时,存储器的状态从未被缓存状态变为缓存状态并给出读请求的数据.这时,cache的入口状态也从无效变为有效的表头状态.表头要存放该块数据的源地址(即所有者所在节点的地址).随着后续的访问操作的增加,链表的长度也随之增加.根据cache的需要,也可以对链表进行删除.由于链表是分布式的和双向链接的,因此可以有多重入口同时进行删除操作.

1.2 SDD协议

SDD协议是典型的单链型协议.在SDD协议中关于数据副本的信息是链式线性分布的.该协议采用了直接的cache-to-cache操作机制,使响应信号的发送不必经过主存,从而减少了出现瓶颈问题的可能性.当发生读/写操作缺页时,用read-miss/write-miss信号进行标识,接着执行read-miss-forward/write-miss-forward操作,进行链式查找所请求的页的副本或页的所有者,查找到时,给出read-miss-reply/write-miss-reply回答信号.回答信号可以和查找到的数据或地址信息一起回送(即载答).

1.3 分析与比较

在DSM系统的管理cache一致性的链表方案中,在每个节点处都建有一个目录,目录

中含有共享数据的状态信息,其中包括该页数据的原所有者(该页建立时拥有写权限的节点)地址信息和当前对该页是否拥有写权限的状态信息.链表指针的指向可以有 2 种,一种是始终指向最新拥有某页数据(即使只是执行了读操作)的节点并把该节点作为该页数据的新的所有者,同时产生一次链接.由于此时该节点的写权限的状态位并未置位,也就是说它只拥有读权限,因此它并不是拥有写权限的真正所有者.当出现写请求操作时,沿链表根据写权限标志位从表中查找真正的所有者;另一种是当进行读操作时,只在读请求节点建立新副本,产生一次链接,但原所有者不发生变更.当进行写操作时,获得写权限后,要修改本节点读/写权限状态信息,执行这次写操作的节点变为该页的新的所有者,并产生一次链接.在这 2 种情况下,当所有者发生变更时,可以采用广播通信方式向所有节点通告所有者变更情况并更新各个节点的目录中关于所有者的状态信息.这种方法的缺点是会增加系统内的通信开销,引起系统性能的下降,因而很少使用.现有多数系统所采用的方法是在执行读操作时,不通告所有者变更情况,也不更新各个节点的目录,而是当某个节点需要进行写操作(需查找所有者)时,首先根据链表找到原所有者地址,再根据链表指针依次查找到新的所有者.我们所讨论的是针对后一种情况而言的.

这里所介绍的 SCI 和 SDD 协议都使用了链式管理方案.从查找时间来看,SCI 协议比 SDD 协议要快,但是要比 SDD 复杂,读/写操作所需要的网络操作的数量也多于 SDD 协议.由于多机系统的互连网络的带宽是较为重要的资源,因此,网络操作数量的增加会引起系统性能的下降,从而抵消查找时间快所带来的益处.从系统的伸缩性来看,由于二者都采用了链式管理方案,因而有利于系统的伸缩性的提高,实现起来也比较简单易行.但是随着系统内节点数的增加,链表长度也会增加.对于读操作来讲,这种增加没有什么大的影响,因为读操作请求节点只要找到所请求的页的一个有效副本并复制到本节点即可.但对于写操作来讲,由于写操作请求者必须找到该页的当前所有者,因此会花费很长的查找时间.由此可见,如何适时更新所有者链表和目录中关于所有者的信息是缩短链表查找时间的关键.

2　链表实时更新协议

2.1　基本原理

如前所述,当发生写请求时,为了查找一个已被更新了的所有者要从原所有者(表头)开始,根据写权限标识沿链表逐一查找,直至查到新所有者为止.查找过程中可能要经过若干只拥有可读副本的非所有者节点,造成不必要的系统开销.如果能适时地将新所有者地址通知原所有者,则查找新所有者的操作可以从表头直接指向新所有者,从而只需 2 次查表操作即可确认新所有者.

2.2　更新时机

设 L 为链表长度或链表链接的节点个数,N 为表内节点序号,M 为新所有者节点的序号($M \leqslant L$),那么从表头($N=0$)开始至 M 节点($N=M$)共需执行 M 次查表操作.事实上,从表头($N=0$)开始至第 1 个节点($N=1$)需执行 1 次查表操作.从第 1 个节点($N=1$)至第 2 个节点($N=2$)需执行 1 次查表操作.共需执行 2 次查表操作.依此类推,从表头($N=0$)开始至 M 节点($N=M$)共需执行 M 次查表操作.

由于在松散耦合分布式系统中,传递几个字节的数据和传送几千个字节的数据所需要

的时间开销的差别不大[12],因此我们给出推理1.

推理 1. 设节点间的每次读/写操作与每次查表操作所需节点间操作的时间开销相同,那么当 $M>2$ 时,可进行链表更新并形成新的关于所有者的链表.

关于推理1的进一步说明如下.当 $M>2$ 时,意味着查找新所有者的操作需进行2次以上.考虑到更新链表(即将新所有者 M 节点地址通知原所有者)需1次节点间的操作,根据新链表从表头(原所有者)直接查到新所有者需1次操作,共需2次操作,小于按未更新的链表进行查找所需的操作次数.因此推理1是成立的.

推理 2. 如果每当形成1个新的所有者节点就更新1次关于所有者的链表,那么至多只增加1次节点间数据传送操作.

这里所谓"增加1次节点间数据传送操作"是指以下2种情况.第1种是在更新操作后不再发生写请求,那么这次更新是1次多余的操作;第2种情况是当链表内节点序号为1($N=1$)时,如果不进行更新,那么从表头到该节点只需1次查表操作,而进行更新仍需1次查表操作,却增加了1次节点间数据传送操作(更新操作).其它情况下将不会造成这种通信开销的无谓增加.值得指出的是,所增加的这1次节点操作开销对一个大的 *DSM* 系统来说是微乎其微的,但是给缩短查表时间带来的益处却是很大的,同时也使得链表更新协议的设计与实现更为简明规范.为此我们的更新协议中将采用推理2的策略.

2.3 链表结构

这里我们以单向链表为例规定链表结构.表头结构如图1所示.其它表的结构如图2所示.和 *SDD* 协议相比,这里只是在表头增加了"所有者地址"这一项,在其它表中增加了"表头地址"这一项.

对于表头,当"写权限"为1时,表示本节点仍是所有者,因此"所有者地址"即是本节点地址,如果"写权限"为0,则表示本节点已不是所有者,"所有者地址"应该是新所有者的地址."链表指针"指向拥有该页数据副本的下一个节点地址.

N	写权限 WR	所有者地址 OD	链表指针

图1 表头结构

N	写权限 WR	表头地址 LHD	链表指针

图2 其它表的结构

2.4 链表更新协议

对于写操作请求处理如下:①如果 $N=0$ 且"写权限"为1,则表明该表头仍是所有者,可以对该页进行写操作,同时对链表中所有副本进行废除或更新操作.②如果 $N=0$ 但"写权限"为0,则表明该表头已不是所有者,需将"所有者地址"代表的节点的"写权限"改为0,本节点(发出写请求的节点)的写权限改为1,之后执行写操作并对链表中其它副本进行废除或更新操作.③如果 $N\neq0$ 且"写权限"为1,则表明本节点即是现行的所有者,可以对该页进行写操作,同时对链表中所有副本进行废除或更新操作.④如果 $N\neq0$ 但是"写权限"为0,则表明本节点不是现行的所有者,需到表头根据"所有者地址"查到现行的所有者所在节点,然后将那个节点的"写权限"改为0,本节点的"写权限"改为1,并对该页数据执行写操作,同时对链表中的其它副本进行废除或更新操作,最后还要将表头中"所有者地址"更新为本节点地址.该协议的形式化描述如图3所示.

以上协议是假定发出写请求的节点不存在缺页的情况下建立的,即本节点已是链表中的一员,拥有了至少是只读权限的副本.如果发生缺页情况,则可先将本节点链接到链表中,

然后执行与上述协议相同的操作即可.只是在查到当前所有者节点时,要从该节点将所缺数据页复制到本节点.

对于读操作请求,如果不缺页,则只需读出该数据副本即可.如果缺页,则先将本节点链接到链表中,之后从前一个节点读取该数据页的副本即可.

```
if N = 0 and WR = 1
  then write the page and invalidate or update other copies
  else if N = 0 and WR = 0
       then access the owner (set the owner's WR =0 and local WR =1),
            write the page and invalidate or update other copies
       else if N ≠ 0 and WR = 1
            then write the page and invalidate or update other copies
            else access the owner via list head
                 (set the owner's WR =0 and local WR =1),
                 write the page, invalidate or update other copies
                 and write the local address to the OD in list head
            end if
       end if
end if
```

图 3 协议的形式化描述

3 结　论

这里所介绍的针对所有者的链表更新协议最大限度地缩短了对所有者的链表查询时间,使查找所有者的操作次数至多为 2 次.这种协议对于分布式共享存储器系统的存储器一致性的保持以及系统规模的变化具有较灵活的适应性.对于写修改较频繁和共享存储器的节点数量较大的情况效果更为明显.这种实时更新协议也可适用于文献[7,8]中提出的 cache 一致性的树形层次结构.

这里讨论的实时更新协议是每当产生新的所有者时都要对表头中的"所有者地址"进行更新操作的策略(注意:自身就是原所有者时则不必进行这种更新),通信开销的最大浪费量是一次节点间的数据传送时间.所增加的这一次节点操作开销对一个大的 DSM 系统来说是微不足道的,但是给缩短查表时间带来的益处却是很大的,同时也使得链表更新协议的设计与实现更为简明规范.

参考文献

1 Chaiken D *et al*. Directory—based cache coherencein large—scale multi—processors. IEEE Computer, 1990,**23**(**6**):49~58.

2 Min S L, Baer J L. Design and analysis of a scalable cache coherence scheme based on clocks and timestamps. IEEE Trans. on Parallel and Distributed Systems, 1992,**3**(**1**):25~44.

3 Hagersten E, Landin A, Haridi S. DDM—a cache—only memory architecture. IEEE Computer, 1992,**25**(**9**):44~54.

4 Yang Q, Thangadurai G, Bhuyan L N. Design of an adaptive cache coherence protocol for large scale multiprocessors. IEEE Trans. on Parallel and Distributed Systems, 1992,**3**(**3**):281~292.

5 Bennett J K, Carter J B, Zwaenepoel W. Adaptive software cache management for distributed shared memory architectures. Proc. of 17th Int'l Symp. on Computer Architecture, 1989. 125～134.

6 Eggers S J, Katz R H. Evaluating the performance of four snooping cache coherency protocols. Proc. of 16th Annual Int'l Symp. on Computer Architecture, May 1989. 2～15.

7 Lenoski D *et al*. The directory—based cache coherence protocol for the DASH multiprocessor. Proc. of 17th Int'l Symp. on Computer Architecture, 1989. 148～158.

8 Anderson C, Jean—Loup Baer. A multi—level hierarchical cache coherence protocol for multiprocessors. Proc. of 7th Int'l Parallel Processing Symp., April 1993. 142～148.

9 Petersen K, Li K. Cache coherence for shared memory multiprocessors based on virtual memory support. Proc. of 7th Int'l Parallel Processing Symp., April 1993. 49～55.

10 James D V *et al*. Distributed—directory scheme: scalable coherent interface. IEEE Computer, 1990, 23(6): 74～77.

11 Thapar M, Delagi B, Flynn M J. Linked list cache coherence for scalable shared memory multiprocessors. Proc. of 7th Int'l Parallel Processing Symp., April 1993. 34～43.

12 Li K, Hudark P. Memory coherence in shared virtual memory systems. ACM Trans. on Computer Systems, 1989, 7(4): 321～359.

A NEW ALGORITHM FOR CACHE COHERENCE IN DSM

Fang Zhiyi Ju Jiubin

(*Department of Computer Science Jilin University Changchun* 130023)

Abstract The management of memory coherence is an important problem in DSM (distributed shared memory) system. In a directory—based and owner—based DSM system, when to update the owner's linked list is the key to reduce the time of searching the list. This paper describes the design and the analysis of a new algorithm for cache coherence in DSM—RTULH (real—time up—dating list head). The analysis result shows that this algorithm can improve the adaptability and performance of a DSM system. It can also be suitable to manage the cache coherence in tree—like hierarchical architecture.

Key words Cache, coherence, linked list, DSM.