强弱混合一致性语义模型*

李 荣 王鼎兴 沈美明

(清华大学计算机系 北京 100084)

摘要 80年代末出现的分布式共享存储(DSM)技术为解决可扩展性和易编程的矛盾提供了一种可行的方案.但是,由于DSM建立在消息传递机制的基础之上,而且存在对本地内存访问和对远程内存访问的差异,简单地照搬多处理机Cache一致性语义往往效率很低.因此,人们提出了一些弱化的、放松的一致性语义,这些语义与用户采用的顺序一致性语义差别较大,给用户在程序设计、调试上造成很大困难.本文提出了一种强弱混合一致性语义模型,解决了假共享问题,提供了进一步开发时间、空间局部性的可能性,而且还可以保证顺序一致性.试验结果证明,强弱混合一致性语义与严格一致性语义相比,消除了"颠簸"现象,大大改善了系统性能.

关键词 并行处理,DSM,混合一致性语义.

计算机发展进入90年代以后,单处理器的性能受工艺、半导体物理技术水平的限制已逐步接近极限,要想进一步提高计算机系统的运算能力,就只有采用并行技术.随着科学技术的发展,越来越多的应用领域将寻求超级计算能力的支持.传统的超级计算领域,如核物理、空气动力学、生物学、数值气象预报、石油勘探、图象处理等领域对超级计算速度的需求几乎是无限的,而一些新的领域正在不断加入到超级计算的行列中来,如CAD、集成电路设计、超大型数据库、计算机图形学等.在这种情况下,并行处理技术成为90年代最热门的研究课题之一.[1]

现在的并行处理系统可分为2种,一种是基于共享存储器的多处理器系统;另一种是基于分布式存储的多处理机系统.前一种系统拥有物理上共享的存储器,若干个处理器通过一条公共总线访问该处理器.后一种系统,每台处理机都拥有自己的存储器,一般不设置物理上共享的公共存储器,处理机之间通过互连网络连接在一起,这种系统缺乏一个统一的地址空间,不能象前者那样通过对单一、共享的公共存储器的访问进行通讯,只能依靠在互连网络上传递消息彼此通讯.共享内存多处理器系统的主要缺点是可扩展性差(处理器数通常不超过20),优势在于设计程序比较容易,而基于分布式存储的多处理机系统正好相反,可扩展性好,但编程复杂.[2]

80年代末期,李凯率先提出在分布式存储结构上实现一个虚拟的共享存储区

* 本文研究得到国家自然科学基金资助.作者李荣,1967年生,博士生,主要研究领域为分布式共享存储系统,并行算法.王鼎兴,1937年生,教授,博士导师,主要研究领域为并行/分布计算技术及应用.沈美明,女,1938年生,教授,主要研究领域为并行机群系统及并行程序开发环境.

本文通讯联系人:李荣,北京 100084,清华大学计算机系

本文1995-07-31收到修改稿

(SVM)[2],现在通常称为分布式共享存储器(DSM).原理上,DSM 和多处理器系统的 Cache 很相象,但实际上,存在较大的差异.DSM 的共享存储区是虚的而非实的,当作 Cache 用的分布式存储器是通过互连网络而不是通过公共总线相连[2],处理机无法实时获取其它处理机的事件信息,而且事件的原子性无法保证.[3]正因为 DSM 存在一些特殊问题,促使人们针对它的特点提出了许多方案.[2,4~7]

对 DSM 来说最关键的 3 个问题是[4]实现方法、结构和粒度、一致性语义及协议.

实现的方法大致有:

a.硬件实现

初期主要是在网络接口上扩充一些支持 DSM 的功能,现在已经开始研制能直接支持 DSM 的微处理器.

b.软件实现,包括 2 种方式

(1)操作系统级实现方式

以李凯的 IVY 系统为例,在页面错误处理程序中扩充 DSM 管理功能.[2]

(2)编译器和运行系统相结合的方式

大多数实验系统是这样实现的.编译器将访问 DSM 地址单元的指令翻译成特殊的原语,运行时,运行系统动态解释执行这些特殊的 DSM 访问原语.[4]

从结构和粒度上看,除李凯的平坦、分页式虚拟地址空间外[2],DSM 还可以表现为一个分段的单级存储区或者一个物理地址空间.DSM 和多 Cache 一样,希望提高访问命中率,通常的办法是采用较大的页,这样从外部调入页面时可多调入一些数据,提高时空局部性.但是并行系统中存在多个并发的进程,它们访问同一页时可能会出现竞争现象,导致这一页在处理机间往返传递,称为"颠簸"现象[2],最严重情况下,这些进程忙于争夺该页,谁也无法前进一步.造成"颠簸"的原因在于共享.实际上,真正在逻辑上相关的共享都位于程序员的控制之下,严重影响系统性能的是那些逻辑上不相关的、不为程序员所知的假共享现象,那些并发的访问者实际上访问的是不同的地址单元,但因为这些地址单元位于同一页面内,如果以页为维护一致性和数据交换的单位,它们就成了该页的共享者,这种现象称为假共享.页面越大,越容易引起假共享,从而降低时空局部性.因此,有人提出缩小维护一致性的粒度以降低假共享发生的概率,PLUS 的一致性粒度甚至缩小到了 32 位的字[5],这样势必导致目录体积膨胀,占用过多的存储单元,系统开销过大.

在一致性语义这个问题上,顺序一致性语义更符合程序员的思维模式[8],IVY 采用的就是这种语义.人们认为语义的严格性影响了 DSM 的效率,因此提出了许多放松的一致性语义模型,有代表性的包括弱一致性[9]、释放一致性[3]、处理机一致性等,放松了对读写事件顺序的约束,与顺序一致性语义差别较大,给程序员编程和调试带来许多不便之处.[3,10]

针对以上这些问题,我们提出强弱混合一致性语义模型.

1　强弱混合一致性语义

1.1　基本思想

对某一数据来说,如果至多只有一台处理机对它进行更新操作,而其它处理机暂时不访问该数据,那么这些处理机上该数据的拷贝(如果存在)就不必立即更新或作废,可以推迟到

其它处理机访问该数据之前进行,即更新延迟.

一个数据块(例如页和段)通常包含若干个数据,如果这些数据全部满足可更新延迟的条件,则允许存在多个写入者和读出者同时访问属于自己的数据块拷贝.这些数据块拷贝可以不同、不一致,需要更新的时候,只要把这些不一致的拷贝合并成一个新拷贝,作废所有的旧拷贝,就可以保证顺序一致性.

如果一个数据块拥有至少 2 个同时进行操作的写入者或者同时进行操作的写入者和读出者,则称该数据块处于弱一致状态,否则处于强一致状态.数据块处于弱一致状态时,它拥有若干弱一致拷贝,退出这种状态时,需将这些弱一致拷贝合并成一个新拷贝,这个合一的过程称为合一操作.

强弱混合一致性语义具有下列特点:

(1)在不改变读写事件顺序的基础上,开发每个数据块内在的并行度.

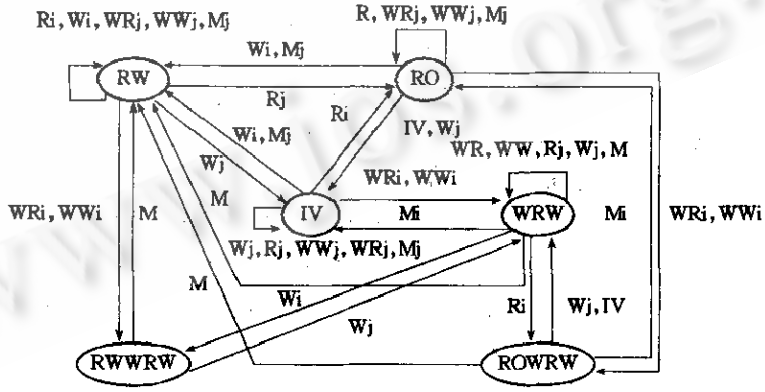
(2)可避免假共享造成的"颠簸"现象.通过使数据块处于弱一致状态,产生若干弱拷贝,解除假共享关系,恢复每个读写者的独立性和并发性,使每台假共享某数据块的处理机都拥有一份该数据块的弱拷贝,从而不必去争夺数据块的所有权,"颠簸"自然就被避免了.

(3)维护一致性的粒度不必缩小,以页为单位就可以,避免了目录体积膨胀的问题.

(4)假共享问题解决后,可以进一步加大数据块的长度,采用更大的页或者建立段页 2 级结构,以增大预取数据量、增强时空局部性.

1.2 强弱混合一致性语义

语义规定,数据可处于强、弱和无效 3 种状态.数据块的状态分 2 大类:单一状态类与混合状态类.单一状态是指该数据块上的数据都处于相同状态(强、弱或无效),否则就处于混合状态.单一状态类包括下列状态:只读(RO)、读写(RW)、弱读写(WRW)和无效(IV);混合状态类包括:只读和弱读写并存(ROWRW)、读写和弱读写并存(RWWRW).强弱混合一致性语义的状态转换图如图 1.



状态结点:RO:只读,WRW:弱读写,RW:读写,ROWRW:强只读和弱读写并存,IV:无效,RWWRW:强读写和弱读写并存.输入的请求:R:读,M:合一,W:写,WR:弱读,IV:无效,WW:弱写.

图 1 强弱混合一致性语义的状态转换图

其中每种请求又分为本地请求和远程请求,用下标 i 和 j 区分本地请求和远程请求,不加下标则表示不必区分来源.

1.3 强弱混合一致性协议

根据强数据和弱数据的特点,我们按照"写——无效"协议维护强数据的一致性,按照"合一——无效"协议维护弱数据的一致性,即:合一时,全部弱拷贝与强拷贝合并成一个新拷贝,将旧的强拷贝、弱拷贝全部作废.下面给出一个简单的协议.为便于叙述,我们假设DSM为分页式的平坦、无结构的线性地址空间结构,以页为单位维护一致性,采用固定分布管理者算法[11],忽略页替换.

定义.　　OP(input(i))=(route(j),state(k))

接收到input(i)请求,则执行route(j),然后该页状态变为state(k).其中state(i),state(k)∈{RO,RW,IV,WRW,ROWRW,RWWRW}, input(i)∈{R,W,IV,M,WR,WW}.

route(1):若是非管理者结点,则向管理者结点发读请求,收到该页后,通知管理者.若是管理者结点,则向该页的所有者发读请求.

route(2):向该页的所有者转发读请求,收到请求者的应答后,将请求者结点加入到拷贝集中.

route(3):若是非管理者结点,则向管理者结点发写请求,收到该页后,通知管理者.若是管理者结点,则判断自己是不是所有者,若不是,则向该页的所有者发写请求,索取该页,向拷贝集中的结点发送"无效"请求,待确认消息全部到达后,管理者成为新的所有者.

route(4):若是非管理者结点,则把该页传给请求者.若是管理者结点,则向拷贝集中的结点发送"无效"请求,待确认消息全部到达后,判断自己是不是所有者,若是,则直接把该页传给请求者;若不是,则将该请求转发给所有者结点.收到请求者的应答后,请求者成为新的所有者.

route(5):若是非管理者结点,则向管理者结点发弱访问请求,收到该页以后,通知管理者.若是管理者结点,则向该页的所有者发弱访问请求,收到该页后,将自己加入到弱拷贝集中.

route(6):若是非管理者结点,则把该页传给请求者.若是管理者结点,则将该请求者从拷贝集中删除后,加入到弱拷贝集中.如果该请求者没有拷贝,则把该页传给请求者;否则,向请求者结点发确认消息.等待收到请求者的应答到达.

route(7):若是非管理者结点,则向管理者结点发合一请求和数据,收到确认消息后,通知管理者.状态变为state(k).若是管理者结点,当弱拷贝集中的结点的合一请求及数据已经全部收到时,若自己不是所有者,则向该页的所有者发写请求,将该页取到本地.并将该页和收到的合一数据合并成一个新页,向拷贝集中的结点发送"无效"请求,待确认消息全部到达后,向弱拷贝集中的远程结点播发确认消息,待应答消息全部到达后,管理者成为新拷贝的所有者,状态变为RW;否则,收下合一数据,状态不变.

route(8):若是非管理者结点,则把该页传给请求者.若是管理者结点,则把该页传给请求者,收到请求者的应答后,将请求者结点加入到拷贝集中.

route(9):向管理者结点发确认消息.

route(10):若是非管理者结点,则向管理者结点发读请求,收到该页后,将本地弱拷贝中已经被更新的弱地址单元与之合并成一个新页,废弃掉旧的弱拷贝,然后通知管理者.若是管理者结点,则向该页的所有者发读请求,收到该页后,将本地弱拷贝中已经被更新的弱地址单元与之合并成一个新页,废弃掉旧的弱拷贝.

route(11):若是非管理者结点,则向管理者结点发写请求.收到该页后,将本地弱拷贝中已经被更新的弱地址单元与之合并成一个新页,废弃掉旧的弱拷贝,然后通知管理者.若是管理者结点,而自己又不是所有者,则向该页的所有者发写请求,收到该页后,向拷贝集中的结点发送"无效"请求,待确认消息全部到达后,管理者成为新的所有者.

页处于 IV 状态时,输入及其处理方法为:

OP(Ri)=(route(1),RO);OP(Rj)=(route(2),IV);OP(Wi)=(route(3),RW); OP(Wj)=(route(4),IV);OP(WRi/WWi)=(route(5),WRW);OP(WRj/WWj)=(route(6),IV);

OP(Mj)=(route(7),IV);

页处于 RO 状态时,输入及其处理方法为:

OP(Rj)=(route(8),RO);OP(IV)=(route(9),IV);OP(Wi)=(route(3),RW); OP(Wj)=(route(4),IV);OP(WRi/WWi)=(route(5),ROWRW);

OP(WRj/WWj)=(route(6),RO);OP(Mj)=(route(7),RO);

页处于 RW 状态时,输入及其处理方法为:

OP(Rj)=(route(8),RO);OP(Wj)=(route(4),IV);OP(WRi/WWi)=(route(5),RWWRW); OP(WRj/WWj)=(route(6),RW);OP(Mj)=(route(7),RW);

页处于 WRW 状态时,输入及其处理方法为:

OP(Ri)=(route(10),ROWRW);OP(Rj)=(route(2),WRW);OP(Wi)=(route(11),RWWRW);

OP(Wj)=(route(4),WRW);OP(WRj/WWj)=(route(6),WRW);OP(M)=(route(7),IV);

页处于 ROWRW 状态时,输入及其处理方法为:

OP(Rj)=(route(8),ROWRW);OP(IV)=(route(9),WRW);OP(Wi)=(route(11),RWWRW);

OP(Wj)=(route(4),WRW);OP(WRj/WWj)=(route(6),ROWRW);OP(M)=(route(7),RO);

页处于 RWWRW 状态时,输入及其处理方法为:

OP(Rj)=(route(8),RWWRW);OP(Wj)=(route(4),WRW);

OP(WRj/WWj)=(route(6),RWWRW);OP(M)=(route(7),RW);

合一操作的算法与实现方式有关,可以采用硬件和操作系统配合的方式,也可以采用软件实现.前一种方法,需在操作系统的页表中增添一项弱更新指示向量,指示该页相应地址单元的弱更新情况,如果某位被置为"1",则表示相应的地址单元已被弱更新,反之,若为"0",则表示相应的地址单元未被弱更新,处理器扩充弱访问指令,在弱写某地址单元的同时置位相应的向量位.合一数据由 2 部分组成:弱拷贝及其弱更新指示向量.硬件电路在弱更新指示向量的控制下,将弱拷贝中的更新后的弱地址单元与强拷贝合并起来.也可以用软件模拟实现上述机制,但效率显然不如前者.

2 性能评价

我们在清华大学 PGR 智能工作站上实现了一个 DSM 系统——THSVM. THSVM 先将对 DSM 地址的访问变换成相应的原语操作,然后在实际运行过程中由运行系统解释执行 DSM 访问原语及维护 DSM 的一致性.[13]

我们在 THSVM 系统上测试、比较了在严格一致性(SC)语义和强弱混合一致性(SWHC)语义下计算矩阵乘 A=B*C 的运行时间. 矩阵规模为 128*128,矩阵 A,B,C 均为共享变量,按行主顺序排列. 矩阵 A 被均匀地按列分成若干条带,每台处理机计算一条数据带. 表 1 列出的是实测结果的平均值.

表 1　实际运行结果

处理机数	SC 的运行时间(s)	SWHC 的运行时间(s)
1	453.8695	455.3082
2	442.2909	239.7673
3	513.6147	173.7581
4	752.9127	147.7729

加速比曲线如图 2 所示.

由于这种数据分布方式在 SC 语义下易引起"颠簸",随着处理机数目的增加,运行时间基本上呈现增长的趋势,而在 SWHC 语义下则可消除这种访问冲突,从而加快运行速度,提高加速比.

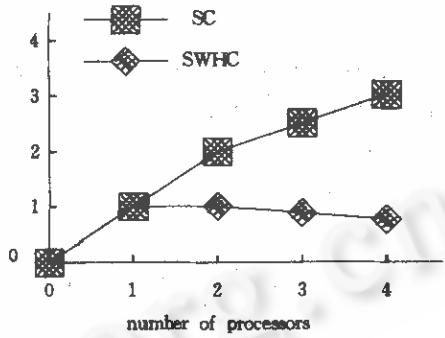


图2 加速比曲线

3 总　结

本文提出了一种强弱混合一致性语义模型,解决了假共享问题,同时也为提高顺序一致性系统的效率提供了一种可行方案. 本文在固定分布管理者算法的基础上讨论了状态的转换过程及相应的处理方法,给出了 2 种合一机制的实现方案. 试验结果证明,SWHC 可以消除"颠簸",显著提高加速比.

参考文献

1 Gordon Bell. Scalable, parallel computers: alternatives, issues, and challenges. International J. Parallel Program, 1994,22(1):3～46.

2 Li Kai, Schaefer Richard. A hypercube shared virtual memory system. ICPP'89, 1989. 125～132.

3 Gharachorloo K, Lenoski D, Laudon J *et al*. Memory consistency and event ordering in scalable shared-memory multiprocessors. ACM Computer Architecture News, 1990,18(2):15～26.

4 Nitzberg Bill, Lo Virginia. Distributed shared memory: a survey of issues and algorithms. IEEE Computer, 1991,(8):52～60.

5 Dasgupta Partha *et al*. The clouds distributed operating system. IEEE Computer, 1991,(11):34～44.

6 Bisiani Roberto, Ravishankar Mosur. PLUS: a distributed shared-memory system. Proc. 17th International Symp. Computer Architecture, IEEE CS Press, Los Alamitos, Calif., Order No. 2047, 1990. 115～124.

7 Fleisch B D, Popek G J. Mirage: a coherent distributed shared memory design. Proc. 14th ACM Symp. Operating System Principles, ACM, NewYork, 1989. 148～159.

8 Lamport Leslie. How to make a multiprocessor computer that correctly executes multiprocess programs. IEEE Trans. on Computers, 1979, (9): 690～691.

9 Dubois Michel, Scheurich Christoph, Briggs Faye. Memory access buffering in multiprocessors. AISCA'86, June 1986. 434～442.

10 Mosberger D. Memory consistency models. ACM Operating Systems Reviews, 1993, 28(1): 18～26.

11 Li Kai, Hudak Paul. Memory coherence in shared virtual memory systems. ACM Trans. on Computer Systems,, 1989, 7(4): 321～359.

12 高瞿清. 博士后研究人员工作期满出站报告. 清华大学计算机系, 1992.

13 邢浩. 分布共享存储系统的研究与实现[硕士论文]. 清华大学计算机系, 1994.

A STRICT WEAK HYBRID COHERENCE MODEL

Li Rong Wang Dingxing Shen Meiming

(*Department of Computer Science and Technology Tsinghua University Beijing* 100084)

Abstract DSM (SVM) is a memory model which was proposed at the 1980s'. DSM provided a possible way to solve the problem that scalability was contradict with easy programming. Because DSM was built on the top of message passing scheme, and the cost gap between local reference and remote reference is quite large, the efficiency of DSM will be very low if the authors simply adopt the cache coherence semantic for multiprocessors. So, some weaken, relaxed coherence semantic model have been proposed, but these models are quite different from sequential coherence which was assumed by most programmers intuitively, so programming and debugging becomes very difficult. This paper proposes SWHC (strict weak hybrid coherence) model which can eliminate false sharing and provide more temporal and spatial locality, also provides a scheme to keep sequential coherent. Experiment results proved that strict and weak hybrid consistency semantic can eliminate the "thrashing" and improve system performance greatly.

Key words Parallel processing, DSM, strict weak hybrid coherence.