

多机可缩放性高速缓冲存储器一致性协议分析

王 钰

(西安邮电学院 计算机科学与技术系, 陕西 西安 710121)

摘 要:分析了可缩放共享存储器多机系统 Cache 一致性协议的软硬件解决方案。集中式控制方法只适应于规模适中的多机系统,这是因为数据块共享信息存放在系统存储器中,成为性能提高的瓶颈;分布式控制方法允许系统拥有多个存储器控制部件,适合于大规模的多机系统。基于软件的 Cache 一致性技术建立在对源程序的预处理基础上,基本不需要或只需要很少的额外硬件。处理机数量的变化并不改变源程序的属性,编译程序的分析结果适用于任何规模的多处理机系统,具有很强的可缩放性。

关键词:可缩放性; Cache 一致性; 多微机系统

中图分类号: TN957.51

文献标识码: A

文章编号: 1673-629X(2009)02-0094-04

Analysis on Coincidence Protocol of Multi-CPU Machine's Scaleable Cache

WANG Yu

(Computer Science Department, Xi'an Institute of Posts and Telecommunications, Xi'an 710121, China)

Abstract: Presents the scheme of the hardware and software solution for the cache coincidence protocol on the multi-CPU machine with a scalable memory system. Centralized control mode can be only used in the moderate multi-CPU system, just because the shared data is stored in the memory of the system, and which becomes the main reason to improve the system performance; Distributed control mode permits the system to have multi memory control units, and it is suitable to the large-scale multi-CPU system. The software solution of cache coincidence is based on the pretreatment for the source program, so that none or a few hardware is required. The change of the number of the CPU in the system does not change the attributes of the source program, and the analyzing result of the compiler suits to any scale multi-CPU system, so that the software method is highly scalable.

Key words: scalability; cache coincidence; multi-CPU system

0 引 言

并行处理机是满足日益增长的计算量需求的最适合的计算机系统之一,其中共享存储器多处理机系统是最有效、最流行的一类,这是因为共享存储器多机系统具有简单,而且通用的编程模型,有效地支持代码和数据共享,从而使得并行软件开发格外方便。这类多处理机系统的性能由于系统结构的限制,处理机个数增加到一定程度就达到顶峰,如果再增加,系统性能就趋于下降。由此提出了可缩放性(Scalability)概念,即在基本系统结构和工作机制保持不变的前提下,整个系统的性能随着处理机数量的增加而呈近乎线性地增长。目前处于大学研究阶段的可缩放共享存储器多机

系统——SSMP (Scalable Shared Memory Multiprocessors) 的原型有 Stanford 大学开发的 DASH 系统^[1], MIT 开发的 Alewife 系统^[2]和 Carnegie Mellon 大学开发的 Plus 系统^[3];商业化的 SSMP 系统有 KSR, Encore Multimax 和 Sequent Balance。SSMP 所要解决的主要技术问题之一是开发可缩放性多机 Cache 一致性协议。

Cache 一致性协议可以由硬件或软件来具体实施。基于总线连接的共享存储器的体系结构大多采用的硬件协议是侦听 Cache 一致性协议,它要求总线上的各个 Cache 控制器时刻监视着总线,随时接收一致性命令。其局限性在于系统能够容纳的微处理器个数受到总线通信能力的限制。为了支持可缩放性的 Cache 一致性协议,这种总线广播消息的机制已经不能满足要求,而要采用互连网络。由于互连网络不具备公共总线的广播能力,因此只能用点对点或一点对多点 (Multicast) 的通信方式发送 Cache 一致性命令,这

收稿日期:2008-07-11

基金项目:陕西省自然科学基金(2004F28)

作者简介:王 钰(1956-),男,山西万荣人,副教授,从事计算机系统结构的教学与研究。

就出现了适合于可缩放性的目录表 Cache 一致性协议和链表 Cache 一致性协议。

1 目录表协议

可缩放性的多机系统采用一般的互连网络。因为它不能同时满足一台处理器到其它所有处理器的全联通,因此不具备公共总线的广播能力,不可能象共享总线型那样同时向所有的 Cache 直接发送一致性命令,而只能有针对性地向拥有某个数据副本的那些 Cache 直接传送命令,再由那些 Cache 对数据副本实施一致性策略。这就要求在系统中记录下每个 Cache 存放着哪些数据的信息,Cache 控制器通过查询这些信息便可知道哪些 Cache 中存有指定的数据副本,从而向它们发送一致性命令。而且,当一个处理器要访问的数据不在本地 Cache 时,通过查询这些信息就可知道哪一个 Cache 中有此数据。这些信息可以用矢量或矩阵(即目录表)集中存放,也可以由链表将它们分散到各个 Cache 中进行分布式存放。基于目录表的 Cache 一致性协议称为目录表协议。目录表协议也有多种实现方案,如全映像目录表法和有限指针法^[4];基于链表的 Cache 一致性协议称为链表协议。

1.1 全映像目录表法(Full-MapDirectory——Dirp)

这种方法为存储器中的每一块数据设置一个位向量和一位修改标记,位向量的每一位对应一个处理器,表明某个处理器是否在其 Cache 中有此块数据的副本;修改标记表明是否有某些处理器对本块数据进行了修改。因此通过这个二维表可以知道整个系统中数据的共享情况。当需要进行写作废或写更新操作时,只需要从这个目录表中获取信息,对需要修改的 Cache 发送一致性命令。

对于有 P 个处理器的多机系统,每个处理器拥有 M 个字节,每一块数据含 B 个字节,那么整个目录表所占用的资源是 $\frac{P \cdot M}{P \cdot M} \cdot P = O(P)^2$ 位(忽略修改

位),它与处理器个数呈平方倍增长。因此,全映像目录表法对于含处理器数量太多的多机系统不适应。但如果处理器个数适中,则它是一个十分有效的方法。例如,Stanford 的 DASH 多机系统由 64 个处理器组成,4 个为一组,共 16 组,组内的 Cache 一致性协议采用侦听策略,组间 Cache 一致性协议采用全映像目录表法。目录表占用存储器资源的 13.3%。

表面上看来,为了减少目录表的开销,可以增加数据块的大小,但是这样做的副作用是假共享现象的增加,从而大量增加维护 Cache 一致性操作和通信量,严

重降低系统性能。

1.2 有限指针法(Limited Pointer Directory)

一些研究表明,在任意时间内,只有极少数的处理器会共享同一个数据。根据这一现象,可以压缩目录表的开销,具体做法是:存储器中的每一个数据块配备一定数量的指针,指向含有数据副本的 Cache。每个指针需要 $\log_2 P$ 位,如果存储器每个数据块有 i 个指针,那么有限指针法所占用的资源是 $(i \log_2 P) \frac{P \cdot M}{B} = O(P \log_2 P)$ 位,随着处理器个数呈 $P \log_2 P$ 倍增长,比全映像目录表法的缩放性好。

这一策略的关键是如何处理含有数据副本的 Cache 多于指针数量的情况,按照处理指针溢出的处理器方法不同,可以细分为三类:

(1) 广播式有限指针法(DIR_B)。

通过给存储器每个数据块增加一个广播位来解决指针溢出问题。当指针溢出时,广播位置位,如果后续操作中对这块数据进行写操作,则对所有的 Cache 发写作废命令。这样会发一些多余的命令,延迟写操作的完成,浪费通信带宽。当大部分数据块同时被多于指针数量的处理器共享时,这一策略的性能很差。

(2) 非广播式有限指针法(DIR_{NB})。

当指针出现溢出时,作废一个共享此块数据的 Cache,腾出一个指针空间存放新申请的处理器指针。这样存储器每一块数据的指针永远不会超过 i 个。其缺点是如果经常读某个数据的处理器个数多于指针数量 i ,那么不断地重复作废和修改指针,会使系统出现颠簸。

(3) 超集法(Dir_X)。

存储器中的每一个数据块只用两个指针,当共享数据的处理器个数多于两个时,这两个指针变成一个复合指针,复合指针的每一位假设有三个状态:0,1,X(表示既为 0,也为 1),占用两个比特位。当要增加一个指针时,新指针与已有的指针进行比较,如果某一位不同,则这一位变换成 X 状态。当出现写操作而要进行写作废时,复合指针中的每一个 X 扩展成为 0 和 1,这样形成的一组指针是实际共享该块数据的 Cache 的超集。

(4) 向量法(Dir_{CV})。

这种方法中,当指针没有溢出时, i 个指针作为普通指针用;当指针溢出时,那么这 i 个指针所占的资源用来存储向量,每一位代表一组处理器(r) 个,每次操作都针对一组处理器进行。

当一个多机系统由若干个用户共同使用时,一个用户可以分配一组处理器进行运算,那么所有的写共

享和读共享数据的情况都发生在组内,写作废只在组间处理器进行。这种情况下,向量法有很明显的优势。

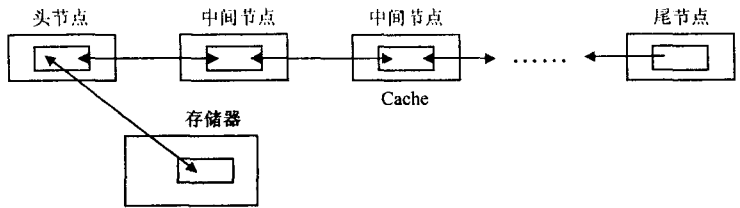


图 1 一个 SCI 共享链

出了各个状态及其简单描述。

表 1 共享链的状态

存储器块状态	头指针状态	中间指针状态	尾指针状态	说明
未共享				数据未进入任务 Cache
净状态	独干净			一个 Cache 有此数据副本,未修改
净状态	头干净		尾有效	两个 Cache 有此数据副本,均未修改
净状态	头干净	中间有效	尾有效	两个以上 Cache 有此数据副本,均未修改
脏状态	独脏			一个 Cache 有此数据副本,已修改
脏状态	独脏		尾有效	两个 Cache 有此数据副本,均已修改
脏状态	头脏	中间有效	尾有效	两个以上 Cache 有此数据副本,均已修改
脏状态	头专用		尾陈旧	两个 Cache 有此数据副本,头已修改,尾未修改
脏状态	头陈旧		尾专用	两个 Cache 有此数据副本,头未修改,尾已修改

2 链表协议

与目录表完全不同的保持可缩放性的方法是采用链表结构,可以用单向链表(如 Stanford 的分布式目录协议^[5])和双向链表等多种形式,在这些结构的基础上形成了 IEEE 的标准 P1596—SCI(Scalable Coherence Interface)协议。链表结构的可缩放性比目录表的强,这是因为它没有集中式控制,也没有全局共享的资源,不同数据块的共享链可以同时进行操作,支持高度的并行化。链表的规模与存储器数据块数、Cache 块数和处理器个数呈线形关系。

2.1 单向链表协议(Single-linked Distributed Directory protocol—SDD)

单向链表协议^[6]的共享链用单向链表实现,每个节点只有一个后继指针,存储器数据块指向共享链的首节点。存储器数据块的初始状态是任何 Cache 都没有它的副本,因而此时的指针为空;如果这时有一个 Cache 要求共享该数据,那么就要给存储器节点的空指针赋值,同时存储器给出回答信号。假如已经有几个 Cache 共享一个数据,这时有一个新的 Cache 要求共享该数据,那么将这个新节点插入到存储器节点和第一个 Cache 节点之间,存储器节点的指针必须修改,但是由共享链的最后一个节点给出回答信号。

2.2 SCI 协议

SCI 基于作废机制之上,支持强排序(Strong Ordering)存储器一致性模型和弱排序(Weak Ordering)存储器一致性模型。

SCI 共享链是一个双向链。存储器某一块数据指向共享链的首节点,共享链中的每一个节点都有一个前驱指针和一个后继指针,如图 1 所示。SCI 共享链上有三个基本操作:插入、删除和简化。插入操作是一个新的 Cache 要求共享数据时使用;删除操作作用于一个 Cache 块要用做其他用途;简化操作用于写作废操作时使用,除了最新做写操作的项以外,此操作删除链表中的全部项。

每一个 Cache 块和存储器数据块的状态域不仅定义指针的意义,而且定义了数据的权限和状态,表 1 给

为了进一步优化 SCI 的操作效率,扩展 SCI 协议,可以对共享链进行改善,文献^[4]提到了一种增加“冗余指针”的方法。如图 2 所示,每一个 Cache 的每一块除了原有的前驱指针和后继指针以外,还设置了第三个指针,即冗余指针,它指向共享链中非邻接的节点,指向较远的节点,从而形成一个树状结构。未改进的 SCI 链的性能是线形的,复杂度为 $O(P)$,增加了冗余指针之后的共享链性能呈对数关系,复杂度为 $O(\log P)$ 。

SCI 协议的开销大约占用 3.5% 的存储器资源和 7.4% 的 Cache 资源,理论上支持 64k 个 Cache 共享同一块数据。SCI 协议是一种点对点通信的协议,一致性操作被分为请求和响应两个子操作完成。SCI 协议还有恢复错误传输和由此引起的错误数据的功能。

链表协议也有缺点:其一,协议要为每一个存储器块维护一条链,这比使用目录表复杂得多;其二,每一个写作废操作要将一条链表从头到尾、一个 Cache 接一个 Cache 的访问一遍,速度慢;而目录表能够以网络可以承受的通信能力尽快发送写作废命令;其三,目录表法可以以存储器的速度进行操作,存储器可用价格

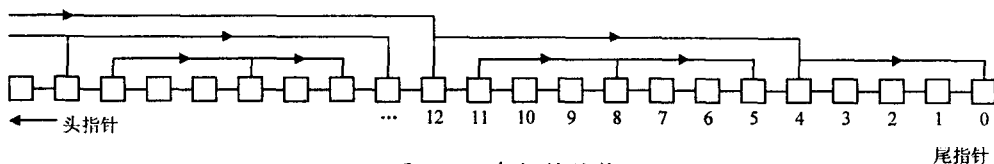


图 2 冗余指针结构

便宜且密度较高的 DRAM 来实现,而链表需要用高速度的 Cache 来实现。

3 基于软件的方法

基于硬件的 Cache 一致性协议均需要额外的硬件机构来维护数据的一致性。随着处理机数量的增多,相应的硬件机构将趋复杂。而且,任何基于硬件的协议只能在程序运行的时候发现 Cache 一致性问题。那么是否可以通过对源程序及其数据的分析,预先找出可能产生一致性问题的变量,并且规定在程序运行时,这些变量不能放入 Cache,任何处理器只能从存储器中存取这类变量呢?如果这样的话,就避免了 Cache 的不一致现象。这一分析工作由专门设计的编译程序或预处理程序完成,故称为基于软件的方法,如 Cedar 系统和 RP3 系统。基于软件的方法实际上是将检测 Cache 一致性问题的开销从运行期间转移到编译期间,把设计的复杂性从硬件转移到软件^[7]。

这类方法的基本策略是,对并发程序进行分析,把各个变量分别标识为“可装入 Cache”(简称 Cache 型)或“不可装入 Cache”(简称非 Cache 型)。各个进程的专用数据或任何只读数据都是 Cache 型的。对共享的可写数据的处理是个复杂的问题,最简单的办法是把共享的可写变量都标为非 Cache 型。这显然过于保守,因为即使是这样的数据也可能在一段时间内只由一个进程单独访问,或者虽由多个进程访问,但所进行的都是读操作,那么在这段时间里此共享数据仍可安全地放入 Cache,不会产生不一致。较为理想的办法是由编译程序分析共享变量何时可以安全地存于 Cache 之中,在这段时间里把变量标识为 Cache 型。至这段时间终结时,该变量在内存中的值必须与 Cache 中的副本相符,而 Cache 中的副本此刻必须作废。这样便提高了多处理机的访存效率,但是增加了编译程序分析工作的复杂性。

基于软件的 Cache 一致性技术建立在对源程序的预处理基础上,基本不需要或只需要很少的额外硬件。处理机数量的变化并不改变源程序的属性,编译程序的分析结果适用于任何规模的多处理机系统,具有很强的可缩放性。

4 结束语

目录表法是集中控制的结构,这使得目录表成为性能瓶颈,阻碍系统的扩展,同时目录表的存储量随着系统的扩大呈指数倍增长。基于链表的 Cache 一致性协议是分布式的控制结构,它的可缩放性很强,并且已经形成了 IEEE 标准 SCI 协议。

基于软件的 Cache 一致性技术建立在对源程序的预处理基础上,基本不需要或只需要很少的额外硬件。处理机数量的变化并不改变源程序的属性,编译程序的分析结果适用于任何规模的多处理机系统,因此软件 Cache 一致性协议具有很强的可缩放性。

在实际的多机系统中往往是软件协议和硬件协议结合使用,侦听法和目录表法或链表法并存,写作废与写更新相结合,发挥各个协议的优势,使得整个 Cache 一致性协议的开销最小。

参考文献:

- [1] Lenoski D. The DASH Prototype: Implementation and Performance[C]// In: Proceedings of 19th Annual International Symposium on Computer Architecture. Gold Coast, Australia: [s. n.], 2000: 92-103.
- [2] Agarwal A. The MIT Alewife Machine: Architecture and Performance[C]// In: Proceedings of 22nd Annual International Symposium on Computer Architecture. Gold Coast, Australia: [s. n.], 1999: 2-13.
- [3] Yousif M S, Thazhuthaveetil M J, DAS C R. Cache Coherence in Multiprocessors: A Survey[J]. In: Advanced in Computers, 1995, 40: 56-101.
- [4] Dubois M, Thakkar S. Scalable Shared Memory Multiprocessors[M]. Norwell, MA: Kluwer Academic Publisher, 1992.
- [5] Thaper M, Delagi B. Stanford Distributed - Directory Protocol[J]. Computer, 1990, 23(6): 78-79.
- [6] Thapar M, Delagi B, Flynn M J. Linked List Cache Coherence for Scalable Shared Memory Multiprocessors[C]// In: Proceedings of 7th International Parallel Processing Symposium. Newport Beach, CA, USA: [s. n.], 1993: 34-43.
- [7] Stallings W. Computer Organization and Architecture Design for Performance[D]. [s. l.]: Prentice - Hall International Inc, 2002.

(上接第 93 页)

- [3] Compact Flash Association CF + and Compact Flash Specification Revision 3.0[S]. [s. l.]: [s. n.], 2004.
- [4] 陈代军. 解析 FAT 文件系统对长文件名的支持[J]. 成都信息工程学院学报, 2003, 18(4): 381-385.
- [5] 陈卫东. 嵌入式系统的数据存储和交换[D]. 北京: 北京邮电大学, 2005.
- [6] 张晓培, 李 详. 从 Unicode 到 GBK 的内码转换[J]. 微计算机应用, 2006, 17(6): 757-759.