基于弱一致性模型软件数据预取策略*

窦 勇 周兴铭

(国防科技大学计算机系 长沙 410073)

摘要 本文针对分布共享存储器中存在的远地访问大延迟问题,提出了基于弱序一致性模型的存储访问优化策略,主要是利用并行程序中同步操作提供的信息,在同步点成块预取将要被使用的数据.该方法能够有效地掩盖远地存储访问的大延迟.

关键词 并行处理,体系结构,编译优化,数据预取.

分布存储器体系结构具有良好的系统可伸缩性,同时分布存储器也具有存储访问时间不一致的特点,处理机对本地存储器的访问时间要远远小于对远地存储器访问的时间.特别是处理机速度的提高远远快于存储器性能的改进,并且随着系统规模的增大,存储器的访问延迟也在增大,因此分布存储器性能的优化成为非常关键的问题.

已经提出了许多优化分布存储器性能的技术[1,2],以克服分布存储器的远地访问大延迟.如传统的cache技术用于减少远地访问次数来提高存储性能.另外,目前广泛研究的弱存储器一致性模型优化方法是采用弱存储器一致性模型放松对存储访问流出次序的严格限制[3,4],以达到优化的目的.数据预取技术和重缓冲技术也是有效提高分布存储器性能的方法.

数据预取技术被用来掩盖大的存储访问延迟,主要是利用处理机的计算与数据访问之间的时间重叠.预取可以分为软件控制和硬件控制2种方法,硬件控制的方法要求处理机内部有硬件部件支持,在运行时硬件自动预取数据;而软件方法依靠编译技术通过对数据流分析,在编译时自动插入预取命令.另外预取又可以分为依赖性预取(Binding Prefetching)和非依赖性预取(Non-binding Prefetching),在依赖性预取中,被预取的数据在预取后的值依赖于预取时的值.而在非依赖性预取中,其后对被预取数据的引用不一定是被预取的值,而由系统的一致性策略保证其正确性.

以前的软件控制预取策略主要是基于顺序一致性模型研究如何掩盖由于cache失效所导致的大延迟,编译优化结合cache参数和数据流分析确定预取的时间和内容.一般需要lock-up free cache保证预取指令以流水方式无阻塞流出.

本文提出一种新的方法在弱存储器一致性模型中,实现软件控制的依赖性自动预取.我

* 本文研究得到国家863高科技项目基金资助.作者窦勇,1966年生,博士,讲师,主要研究领域为并行处理,体系结构.周兴铭,1938年生,教授,主要研究领域为巨型机体系结构,分布、并行处理.

本文通讯联系人:窦勇,长沙 410073,国防科技大学计算机系

本文1996-01-08收到修改稿

们研究的背景是具有非一致存储访问特点的大规模共享存储器体系结构,利用弱存储器一致性模型对并行程序的约束信息,实现自动的数据预取.主要思想是编译器在程序的同步点插入预取命令,预取在此同步块中将被引用的数据.预取操作是一个块传输指令,它代替若干个单字远地访问,类似于向量取访问.提高性能的主要机制是合并频繁的单字取操作为少数块传输操作实现掩盖远地访问延迟的目的.

本文第1节介绍有关体系结构和一致性模型的背景知识,第2节讨论预取策略的细节,第3节给出了确定预取数据集编译方法,第4节给出预取策略的应用及性能分析,第5节总结本文的工作.

1 体系结构模型和一致性模型概述

1.1 体系结构模型

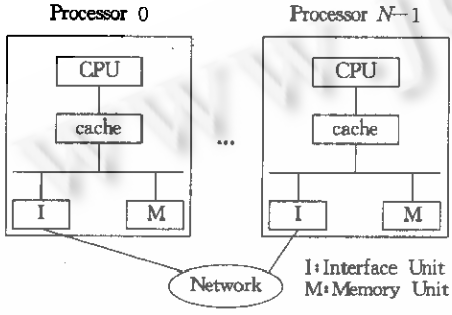

图1 NUMA体系结构模型

为了方便下文的介绍,假设一个具有非统一存储访问特点的NUMA(non-uniform memory access)体系结构,如图1所示.NUMA体系结构包括若干个处理结点,处理结点之间由可伸缩互连网络连接.每个处理结点主要包括3部分:微处理器、私有cache和局部存储器.私有cache只缓冲其局部存储器中的数据,因此处理机之间不存在数据一致性问题.NUMA体系结构的存储系统是一个物理分布的、逻辑共享的存储器.物理分布是因为每个处理结点有一个局部存储器;逻辑上共享是因为任意处理机有以访问任何处理结点的局部存储器中的数据.处理结点发出的访问远地结点局部存储器的存储器地址由硬件支持自动地实现地址变换.NUMA体系结构支持块传输操作主要有4类:常数间距读、常数间距写以及Gather的Scatter操作.

1.2 弱序存储器一致性模型

文献[5]中提出了弱序存储器一致性模型(Weak Order Consistency Model).弱序存储器模型包括2部分,一部分为程序模型,用于约束程序员对并行程序的设计;另一部分为存储器实现模型,约束存储器实现所必须满足的条件.符合弱序存储器模型的存储器系统保证按照程序模型设计的并行程序能够象在顺序一致性模型一样被正确执行,否则弱序存储器系统不保证程序的正确执行.

弱序存储器的程序模型只要求所有同步操作都必须被显式标识,以区别于普通的存储访问操作,在此基础上,程序模型对并行程序有如下约束:(1)同步操作对存储系统是可见的或称为显式标识的.(2)没有数据竞争.

如果2个存储访问操作的对象是同一存储地址,而且至少有一个操作是写操作,则称这对存储访问是冲突访问.如果不同处理机上的2个存储访问操作是冲突访问,而且它们之间的发生次序被规定,它们可能同时在不同处理机上执行,则引起数据竞争,此对操作被称为竞争操作.没有数据竞争,即为没有竞争操作存在.

对于上述程序模型,弱序存储器只约束存储器的实现满足如下条件:(1)存储访问操作

必须满足每台处理机自身的数据、控制相关关系.(2)对同步变量的存储访问操作一定是顺序一致的.(3)在一台处理机中,仅当同步操作以前的所有数据存储访问操作被完成后,对同步变量的存储访问操作才能流出.(4)在一个对同步变量的访问被完成之前,所有的访问都不能在此处理机中流出.

弱序存储器模型通过程序中显式地标识对同步变量的操作,只需要在同步点维持存储器的一致性.它不只具有简单的程序设计界面,而且减弱了对存储访问操作发生次序的约束,为提高存储器性能创造了条件.

2　数据预取的基本原理

2.1　基本假设和定义

本文只考虑 *fork/join* 结构的并行 Fortran 语言,并行程序由若干个任务组成,每个任务是传统的多任务概念中的子程序.任务是静态的概念,它是一段可执行的代码;进程是动态的概念,它是任务的一次执行过程.程序员通过显式的任务并发语句 *fork*(),表示并行程序之间的控制依赖关系.*fork*()语句的功能是启动一个进程执行指定的任务.*fork* 执行完成时,父进程派生出一个子进程,完成任务的并发.假设用 *fork/join* 表示显式并行程序设计模型.

程序员使用共享变量的方式,表示进程间的通信关系,同时为了保持对共享变量的互斥访问,程序员要显式地使用同步原语,安排进程间的同步、互斥,统称为同步操作 *Syn-op*.

并行程序的执行过程中,并行任务间的操作是并行的,同一任务内部的操作是顺序的,而且一次任务执行开始之前,任务内部操作的发生次序是非确定的.只有当一次任务执行结束,任务内部操作的发生先后次序才是已经确定的.用任务执行图 $G(T)$ 表示任务 T 的一次执行.$G(T)=\langle N,E,No,Nn\rangle$,$N$ 表示操作的结点集,E 表示操作之间发生次序的有向边集;No 表示任务第 1 个操作的结点;Nn 表示任务最后一个执行操作的结点.如果在图 $G(T)$ 中有一条有向边 e 从 P 结点到 Q,则有向边 e 表示任务 T 的此次执行中操作 P 先于操作 Q 发生.图 $G(T)$ 是一个无环、有向、连通图.$G(T)$ 中的结点可以进一步分为同步结点、非同步结点.非同步结点表示非同步操作,包括计算操作、对私有数据的引用和赋值、对共享数据的引用和赋值及转移、判断操作.同步结点表示同步操作,包括对同步操作、任务并发控制操作 *fork/join* 和任务头尾隐含的同步操作.

定义. 同步块 *Syn-B*(*Syn-op*)

同步操作 *Syn-op* 的同步块 *Syn-B*(*Syn-op*) 是任务图中 *Syn-op* 操作到其第 1 个后继同步操作之间的非同步操作的集合.

定义. 同步块的预取数据集 *In_prefetch*(*Syn_op*)

In_prefetch(*Syn_op*)是同步块 *Syn-B*(*Syn-op*)中只引用或先引用后定义的共享数据集合.

定义. 预取操作 *prefetch*(x)

prefetch(x)将变量 x 表示的数据取到离处理机较近的存储层次中存放.

2.2　预取策略

因为弱存储器一致性模型的程序模型保证程序中没有数据访问冲突,因此在一个任务的一次执行中,在同步块中的所有访问是互斥的,在一个同步块中被访问的共享数据不会有另一进程更新它,直到进程退出同步块.因此,在同步块 $Syn\_B(Syn\_op)$ 中被引用的数据值可以在进入同步块之前成块预取进入局部存储器的临时单元,其后对这些数据的引用直接从局部存储器的临时单元取值,而不必向远地存储器发请求.下面给出了同步点预取策略(Prefetch at Synchronization Point)的一般性描述[4]:

对于符合弱存储器一致性模型的程序 P,已知每个同步操作 Syn_op 的预取数据集 $In\_prefetch(Syn\_op)$,在同步操作后插入预取操作 $prefetch(x)$ ($x \in In\_prefetch(Syn\_op)$),得到重构程序 P',被重构程序 P' 在 NUMA 体系结构中的执行结果与非重构程序 P 相同.

3 预取数据集的确定

在同步点预取策略的一般性描述中,假设可以精确地确定在每一个同步点要预取的数据集,但是实际上不可能在编译时准确地分析出这些信息.因为所定义的同步块是一个运行后才能确定的概念.在程序每次执行之前,无法精确计算同步块的组成,因此编译器计算上述数据集,只能根据程序的数据流分析做可能性估计.下面设计的编译策略,尽可能准确估计同步块中的预取数据集.对预取数据集估计必须遵守如下原则,以保证对程序的重构不破坏弱序存储器一致性模型对并行程序的约束.

在同步块中被引用数据,一定在同步点被预取.

同步点预取策略主要通过 2 条途径提高存储器的性能:①利用成块数据传输,实现通信与计算的重叠;②利用数据访问的局部性,多次重用临时存储区的数据,减少远地访问次数.因此本文只考虑循环中对共享数组元素访问的优化.而且假设由程序员显式地分割数据.这样,对于给定的一个数组元素,编译器可以分析出拥有此数据的所有者处理机号.因此,对给定的成块数据访问,编译器能够安排成块操作正确地向其所有者处理机发访问请求.设计编译器实现同步点预取策略的数据流分析方法:

第 1 步:全局数据流分析[6]

对程序做数据流分析,一个程序由控制流图 $CFG\langle N,E,N0\rangle$ 表示.N 是代表基本块的结点集合,基本块是 CFG 中一串顺序执行的语句序列,其中只有 1 个入口和 1 个出口.E 是程序中代表控制流的边集合.$N0$ 是程序的唯一入口点.

第 2 步:寻找循环

自然循环 L 是控制流图中具有单一入口点的环.定义 $Def(L)$ 为循环 L 中被赋值数组元素的集合:$Use(L)$ 为循环 L 中被引用数组元素的集合.

第 3 步:归并循环为循环簇

因为预取的另一个特点是利用数据访问的局部性,增加数据被重用的机会,以减少远地访问的次数.第 2 步中将同一循环内部对数组的访问统一起来,但是循环之间也有可能数据被重用,所以,第 3 步将循环合并为簇,其目的就是利用循环之间数据重用的机会.

$$Prefetch\_set(s)=\bigcup_{B\in succ(s)} Pre(B)$$

$$Pre(B)=\begin{cases}(Use(B)-Pst(P))\cup Pre(Q) & \text{if } B \text{ is a non-syn-loop}\\ P\in pred(B),\ Q\in succ(B)\\ \Phi \quad \text{otherwise}\end{cases}$$

$$Pst(B)=\begin{cases}Def(B)\cup Def(P) & \text{if } B \text{ is a non-syn-loop}\\ P\in pred(B)\\ \Phi \quad \text{otherwise}\end{cases}$$

图 2

通过上述分析可以得到以循环簇和同步操作为结点的程序控制流图 CFG_L. 按照预取数据集的定义,*prefetch-set*(*s*)是同步点 S 之前所有非同步结点被引用数据集的并集,并从中除去被先定义的元素. 图 2 给出了计算每个同步点预取数据集的数据流方程. 方程可以采用迭代法求解. 其中 *succ*(*P*)表示 P 的后继结点集合,*pred*(*P*)表示 P 的前驱结点集合. 显然,预取的数据量大于程序运行时所引用的数据集. 因为为了保证程序运行的正确性,在编译时,对于无法确定其定义集合的结点,设其 *Def* 为∅,导致预取了程序运行时不被引用或先被赋值后引用的数据. 可以采用其它编译策略降低预取不精确所带来的额外开销,这里暂不讨论.

利用弱序存储器一致性模型对程序的约束,同步点预取策略采用成块数据传输的方法掩盖分布存储器中远地访问的大延迟. 在同步块中被预取的数据存放于本地址存储器的临时存储区中,增加了数据被重用的机会. 连续分配的临时存储区增强了私有 cache 的访问局部性,同时避免了多机 cache 数据不一致性的干扰. 理想情况下,除同步点的成块预取之外,其它该操作全部命中局部存储器而且大部分在私有 cache 中缓存,因此预取策略能够充分改善分布存储器体系结构的访问性能.

4　同步点数据预取策略应用举例

已知偏微分方程 $\frac{\partial^2 U}{\partial X^2}+\frac{\partial^2 U}{\partial Y^2}=F(x,y)$

求解 Possion 方程第 1 边界值问题. 采用 JACOBI 迭代法计算 $Uij(i,j=1,2,\ldots,n)$. 假设 p 台处理机,并行程序如图 3 所示. 当误差 *esp* 大于指定值 *delt* 时,迭代结束. 处理机 Pk 计算 U 矩阵的第$(n/p*k+1)$列到第$(n/p*(k+1))$列. 假设数据按块分布在 p 台处理机的存储器中,则每步迭代中,每台处理机需取 2 列数据放置于其相邻处理机中的元素. 如果不采用预取策略则每步计算需发出 $2n$ 个单字远地读请

```
prefetch(U*,tid*(n/p))                          同步点预取策略
prefetch(U*,(n/p)*(tid+1)+1)
task(tid)
while (esp>delt)
do  j=tid*(n/p)+1,(n/p)*(tid+1)                 非预取策略
    do I=1,n
       remote_load(Ui,j)
       (j=tid*(n/p)  or  j=(n/p)*(tid+1)+1  i=1,n)
       Ui,j=(Ui+1,j+Ui-1,j+Ui,j+1+Ui,j-1)-h2*Fi,j/4
    enddo
  enddo
enddo
esp=|Unew-Uold|
barrier(p)                                      同步点预取策略
prefetch(U*,tid*(n/p))
prefetch(U*,(n/p)*(tid+1)+1)
endwhile
```

图3　采用预取和非预取策略的JACOBI迭代并行程序

求，时间为 T_{np}. 如果采用预取策略则每步成块取 $2n$ 个元素，发出 2 个成块读请求，时间为 T_p. 比较远地访问时间，设 T_0 为读操作启动时间，T_1 为单字传输时间，$T_0/T_1=k$，则 $T_p=2*(T_0+n*T_1)$，$T_{np}=2n*(T_0+T_1)$.

$$\frac{T_{np}}{T_p}=k+1 \qquad (n>>k)$$

所以预取访问的效率约是非预取访问的 k 倍.

5 总 结

本文基于弱存储器一致性模型对并行程序的约束提出同步点预取策略，主要解决了 2 个问题，即何时发预取命令和预取数据的确定. 基于上述策略实现程序优化，编译器自动插入预取命令，成块预取将要使用的数据有效地掩盖了 NUMA 体系结构存储访问的大延迟.

参考文献

1 Mowry T, Gupta A. Tolerating latency through software-controlled data prefetching in shared memory multiprocessors. Journal of Parallel and Distributed Computing, 1990,12(2):210～217.

2 Chen T F. Data prefetching for high-performance processors [Ph. D. thesis]. Dept. of Computer Science and Engineering, Univ. of Washington, 1993.

3 Mosberger David. Memory consistency models. Operating System Overview, 1993,27(1):7～15.

4 Dou Yong, Zhou Xingming. Prefetching at synchronization points for programs respecting weak memory consistency model. In: Proc. 1995 Intl. Conference on Young Computer Scientists, Beijing, 1995.

5 Sarita V Adve, Mark D Hill. Weak ordering-a new definition. In: Proc. 1990 Intl. Conference on Computer Architecture, 1990. 2～14.

6 Muchnik Steven S, Jones Neil D. Program flow analysis: theory and applications. Prentice Hall, EngleWood Cliff, NJ, 1981.

A SOFTWARE-CONTROLLED DATA PREFETCHING SCHEME BASED ON WEAK ORDER CONSISTENCY MODEL

DOU Yong ZHOU Xingming

(*Department of Computer Science National University of Defence Technology Changsha* 410073)

Abstract This paper presents a software-controlled data prefetching scheme to overcome the delay of remote memory accesses in the physically distributed and logically shared memory system. With the information provided by the parallel programs respecting weak order consistency model, prefetch operations can be arranged at synchronization points to get the data before they are used. This scheme can effectively hide the large latency of non-uniform memory access architechtrol model through block operations.

Key words Parallel processing, architecture, compiler optimization, data prefetching.