共享存储器多处理机并行计算编译及调度机制*

曾绍华[1,2],魏 延[1,3]
(1. 重庆大学 自动化学院,重庆 400030;2. 重庆师范大学 管理学院,重庆 400047;
3. 重庆师范大学 数学与计算机科学学院,重庆 400047)

摘 要:引入并行程序段标记、程序段归并构造并行程序编译思想,设立并行计算调度状态字构造并行计算调度表,提出了一种有效的共享存储器多处理机程序、作业级并行计算编译调度思想及相应的算法。

关键词:多处理机;并行计算;编译;调度
中图分类号:TP301.6;TP314 文献标识码:A 文章编号:1672-6693(2006)01-0027-04

The Compiling & Calling Ideas of the Parallel Computation Based on the M-shared Multi-processor

ZENG Shao-hua[1,2], *WEI Yan*[1,3]
(1. College of Automation, Chongqing University, Chongqing 400030;
2. College of Management, Chongqing Normal University, Chongqing 400047;
3. College of Mathematics and Computer Science, Chongqing Normal University, Chongqing 400047, China)

Abstract: This paper employs the program segment marks and the compiling ideas and combining the program paragraphs to construct the parallel program, and sets the calling state words of the parallel program to construct the calling table of the parallel program, and raises the useful compiling concepts and the effective calling algorithms of the M-shared multi-processor.

Key words: multi-processor; parallel computation; compiling; calling

并行计算技术有以下几种形式:(1)指令内部的微指令的并行——单处理机流水线技术;(2)指令之间的并行——SIMD、MIMD 等;(3)多任务多道程序并行——单处理机操作系统中的进程管理和作业管理(其实质为操作系统软件提供,宏观上的并行,微观上的串行);(4)程序或作业级的并行——多处理机系统硬件提供[1]。

根据并行的颗粒,可分为细粒度、中粒度和粗粒度[2]。在计算机资源无限多的前提下,忽略通讯开销,细粒度的并行程度最高。但是在运行速度上处理机间通讯与处理机速度相差 1 个数量级以上,粒度过细往往得不偿失。粒度过细,要实现很好的并行,避免过多的等待,要求的处理机资源相应增加,将增加访存冲突、通讯线路和控制的复杂性[1,2]。

在多处理机系统中,中粗粒度并行计算要求的资源相对较细粒度省一些。为了减少共享存储器的访存冲突,降低通讯线路和控制的复杂性,多采用粗粒度进行并行计算,从而减少处理机数量,在保证一定运行速度的基础上降低造价,实现更高的性价比。但是,相应地增加操作系统调度的难度,并要有并行编译系统支撑,才能实现高效的并行计算[2]。

1 共享存储器多处理机并行计算前提

1.1 程序的相关性定义

程序段 1 的输入变量中至少有一个是程序段 2 的输出变量,或者程序段 2 的输入变量中至少有一个是程序段 1 的输出变量,则称程序段 1 与程序段 2 相关。反之,则称程序段 1 与程序段 2 不相关。

1.2 程序、作业级并行计算的判定

1.2.1 输入(出)变量向量 在编译过程中有一变量表,按变量表的先后顺序,对不属于程序段 1 输入变量的变量取 0,属于程序段 1 输入变量的变量取

* 收稿日期:2005-09-06
作者简介:曾绍华(1969-),男,重庆人,博士研究生,研究方向为计算机网络、数据挖掘。

1,所构成的行向量称为程序段1的输入变量向量,记为$[PIV_1]$;对不属于程序段1输出变量的变量取0,属于程序段1输出变量的变量取1,所构成的行向量称为程序段1的输出变量向量,记为$[POV_1]$。

1.2.2 并行计算的判定定理 显然,如果程序段1与程序段2相关,则程序段1与程序段2不能并行计算。

定理1 如果$[PIV_1][POV_2]' \geqslant 1$或者$[POV_1][PIV_2]' \geqslant 1$,则程序段1与程序段2相关。

定理2 如果程序段1与程序段2相关,程序段2与程序段3相关,则程序段1与程序段3相关。

定理3 如果程序段1与程序段2不相关,则程序段1与程序段2可并行计算。

2 程序、作业级并行计算的编译机制

2.1 并行程序段标记

在多处理机系统中,程序段内指令和数据按执行顺序串行书写。程序段前有程序段名、串行号。要求各程序段不同名;并行执行的所有程序段的串行号相同;在某一程序段执行前执行的程序段的串行号比该程序段串行号小。程序段有段结束标记。

2.2 并行程序编译机制

2.2.1 编译前期处理 将各程序段翻译成目标代码[3],评估各程序段的目标码运行时间开销,构造程序段归并准备表[4],其结构见表1,其中$T(f,t)$指单指令的平均时间开销(为处理机主频和主存存取周期的函数)。

表1 程序段前归并准备表

程序段名	串行号	归并标记	目标代码运行时间开销
…	…	…	指令条数$*T(f,t)$

2.2.2 归并程序段 编译系统根据操作系统调度一个程序段的时间开销($T_{调}$)、程序段执行的时间开销($T_{执}$)及同串行号的程序段数(n)判断程序段的归并。

(1)前归并。相同串行号的程序段归并。在被归并程序段的目标代码中,取消被归并程序段的段名、串行号、结束标志;将其目标代码插入到接受归并程序段的目标代码的段名、串行号之后,并重新计算接受归并程序段目标代码入口地址;然后在归并准备表中修改接受归并程序段目标代码入口地址,删除被归并的程序段信息。称此类归并为前归并。

(2)前归并策略。

①提高效率策略。知道要实现并行计算,最坏的情形(并行程度最低)是同串行号的程序段仅有2个,否则是串行程序。所以,假定各程序段执行时间均衡时有

$$2T_{调}+T_{执}<2T_{执}$$

即程序段执行时间开销不大于操作系统调度一个程序段时间开销2倍的程序段,编译系统将其前归并在同串行号的程序段。否则既浪费了资源,又降低了速度。一般$T_{执} \gg T_{调}$。

②$N-1$策略。N处理机系统资源全部为某工程同串行号的n段程序占用,并行执行的时间开销$T_{并}$满足:假定各程序段执行时间均衡时有

$$(\mathrm{int}(n/(N-1))+1)*(T_{调}+T_{执})<T_{并}<(N-2)*T_{调}+(\mathrm{int}(n/(N-1))+1)*(T_{调}+T_{执})$$

同串行号的程序段数等于$N-1$且假定各程序段执行时间均衡时,此$N-1$段程序并行执行的时间开销为

$$T_{并}=(N-1)*T_{调}+T_{执}$$

这样,当$n>N-1$时,同串行号的n段程序归并为$N-1$段,不会出现同串行号的程序段在同一处理机上执行,将节省$\mathrm{int}(n/(N-1))*T_{调}$的时间开销。所以,$n>N-1$时,同串行号的$n$段程序前归并为$N-1$段。

③尽可能提高并行度策略。在N个处理机系统中,假定各程序段执行时间均衡时,同串行号的n($n<N-1$)段程序并行执行的时间开销为

$$T_{并}=n*T_{调}+T_{执}$$

由于$T_{执} \gg T_{调}$,所以,在满足$n<N-1$和各并行程序段$T_{执}>2T_{调}$的条件下,尽可能提高其并行度。

当同串行号被归并程序段执行时间开销之和$>2T_{调}$,且同串行号不被归并程序段数$<N-1$时,将从程序段归并准备表中去掉几个被归并程序段的归并标记,再执行前归并,使各并行程序段$T_{执}>2T_{调}$,且该串行号的程序段数最大。

④程序段均衡策略。同串行号的n段程序都争取到处理机资源,其并行执行的时间开销为

$$T_{调}+\max(T_{执}) \leqslant T_{并} \leqslant n*T_{调}+\max(T_{执})$$

若下一串行号的程序段必须在该串行号的所有程序段执行结束后才能开始执行,程序段不均衡将造成处理机长时间等待,浪费资源。为此,在前归并中,从运行时间开销最长的被归并程序段归并起,接受归并程序段为不被归并程序段中运行时间最短的程

序段,使同一串行号的程序段的时间开销基本均衡。

(3)前归并算法。具体步骤如下。

step1　读取副本(n 为程序段数,N 为处理机数目,n_1 为 $T_{执} \geqslant 2T_{调}$ 的程序段数,$n_2 = n - n_1$);

step2　If $(n > N-1)$ then $n_{并} = N-1$

　　else　$n_{并} = n_1 + \mathrm{int}(n_2/2)$;

step3　If$(n_{并} = n)$ then goto step8;

step4　取 $T_{执}$ 最小的 $n - n_{并}$ 个程序段作归并标记;

step5　在作了归并标记的程序段中,取 max$(T_{执})$程序段前归并到未标记 min$(T_{执})$程序段中;

step6　删除作了标记的 max$(T_{执})$程序段;

step7　If(作了归并标记的程序段不存在) then goto step5;

step8　If$(T_{执} < 2T_{调})$ then 重新读取副本,$n_{并} = n_{并} - 1$,goto step4;

step9　end

(4)后归并。当同串行号的程序段根据前归并原则归并后,若被归并为一个程序段,且仍 $T_{执} < 2T_{调}$ 或某一串行号的程序段仅有一个,且也属被归并程序段,需将其向直接前运行程序段归并。若该程序段的直接前运行程序只有一个,去掉直接前运行程序段的结束标志和被归并程序段的程序段名、串行号,将归并程序段的目标代码加到直接前运行程序段目标代码之后,然后将程序段归并准备表中被归并程序段有关信息删除。称此类归并为后归并。

若该程序段的直接前运行程序段有若干个,将该程序段后归并到直接前运行程序段中执行时间开销最长的程序段上。

当执行前归并将某串行号的所有程序段归并为一个程序段,且直接前运行程序段只有一个,将该程序段后归并到其直接前运行程序段;若其直接后运行程序段仅有一个,将直接后运行程序段后归并到该程序段。

(5)后归并算法。具体步骤如下。

step1　读取副本(已作前归并);

step2　Scode = 最后一个串行号,S_0 = 倒数第二个串行号;

step3　If(串行号为 Scode 的程序段数 > 1) then goto step7;

step4　If(串行号为 S_0 程序段数 = 1) then 将串行号为 Scode 的程序段后归并到串行号为 S_0 程序段中;

step5　If$(T_{执} < 2T_{调})$ then 将串行号为 Scode 的程序段后归并到串行号为 S_0 的 max$(T_{执})$程序段中;

step6　删除串行号为 Scode 的程序段;

step7　If(Scode 为第一个串行号) then goto step9;

step8　Scode = S_0, S_0 = S_0 前一串行号, goto step3;

step9　end。

2.2.3　构建并行计算调度表　共享存储器多处理机系统分处理机数目大于机器字长和处理机数目小于机器字长的两类。在处理机数目大于机器字长的系统中并行程序的调度最为复杂。以此类系统为例(本文假定机器字长为 64 位,处理机数目为 256,程序段数 $> 4 \times 256$)构建并行计算调度表,其结构见表 2。并行计算状态字为 $4 \times$ 处理机数目;一般情况下,处理机数目为 $2^n \times$ 机器字长。

表 2　并行计算调度表

程序段名	串行号	并行计算准备就绪否	是否已排队	程序段入口地址	是否执行完成
…	…	…	…	…	

(1)分页。编译系统按串行号递增序,同串行号下执行时间开销递减序对归并准备表排序[4]。读出此表中($i \times 4 \times$ 处理机数目)和($i \times 4 \times$ 处理机数目 +1)行(其中,$i = 1, 2\cdots$为页号)的串行号的值,判断其是否相同。若相同,在($i \times 4 \times$ 处理机数目)行串行号的起始行之前插入虚拟程序段名(虚拟程序段名编译系统用 $*1$、$*2$、…、$*j$ 代替),直到该串行号的起始行为($i \times 4 \times$ 处理机数目 +1)行为止。依次类推,直到分页完成。

(2)定义并行计算状态字。构建一个并行计算状态字生成表,表结构见表 3。

表 3　并行计算状态字生成表

串行号	程序段名	Word1	Word2	…	Word16

按归并准备表的顺序调入一页,即此表 $16 \times$ 机器字长个记录。第一条记录对应 Word1 的最高位,第二条记录对应 Word1 的次高位……,其余依次类推。用数量化理论[5]方法,在每一程序段中对应的位上,逐记录确认该位对应的程序段是否为此记录

对应程序段的前行程序段(指此页中),不是置1,是置0。虚拟程序段所对应的行,位全置1。建立一个状态字生成表结构的副本,将状态字生成表的结果追加到状态字生成表副本中,再调入一页,重复计算状态字生成表,将结果追加到状态字生成表副本中,直到完成。

2.2.4 编译 (1)向并行计算调度表填入数据。按归并准备表的顺序用归并准备表和编译的结果向计算调度表添加数据,将非虚拟程序段所在行的并行计算准备就绪否、是否已排队和是否执行完成字段置0,虚拟程序段对应位置置1;(2)存储编译结果——并行计算调度用表(计算调度表和状态字生成表副本)和各程序段目标代码。

3 程序、作业级并行计算的调度算法

step1 某一工程提交共享存储器多处理机运行时,操作系统从磁盘将该工程的并行计算调度用表的第一页和该页各程序段目标代码调入共享存储器,并根据机器主辅存储器管理方式相应的地址映象和替换算法,修改并行计算调度表中各程序段目标代码的入口地址[1,2,6]。

step2 调度处理机定时逐记录检查调入的状态字生成表副本(仅有一页),如 word1—word16 中各位全为1,将在调入的计算调度表副本中对应的该程序段的并行计算准备就绪否标记为1。

step3 操作系统定时逐记录扫描计算调度表副本,按先后顺序将并行计算准备就绪否标记为1,且是否已排队标记为0的程序段的入口地址提交排队队列,并将其是否已排队标记为1。

step4 操作系统定时侦测各非调度处理机所处状态(busy or ready),若侦测到某一处理机 ready =1(准备就绪),操作系统从排队队列中取出一程序段的入口地址传递给该空闲非调度处理机执行该程序段,并记录该处理机号,运行的程序段所属工程,程序段名。

setp5 非调度处理机执行完程序段后,向调度处理机发出该程序段已执行完成信号。操作系统调度中断处理程序中断调度处理机的现行工作,将非调度处理机执行完成的程序段所属工程、程序段名转存于缓存中 如果排队队列中有排队的程序段,则从中取出最前面程序段的入口地址给该非调度处理机,并记录该非调度处理机将执行程序段的所属工程、程序段名。调度处理机随后根据转存于缓存中对计算调度表副本中该程序段是否完成标记为1。同时,扫描计算调度表副本中各程序段是否完成标记是否全为1,如果是,则调入新的一页,不是则记录该完成程序段的记录号 n,$i = \mathrm{int}(n/64) + 1$,$j = \mathrm{mod}(n/64)$,取 chang 为从高位起第 j 位为1,其余全为0的64位二进制数,逐记录修改状态字生成表副本中的[wordi] = [wordi] or [chang]。结束后返回中断。

4 应用和存在的问题

4.1 应用

(1)即使任何一个工程,只要它有足够的长度,它都包含有许多可并行的成分,应用文献[7~9]的思想可进行并行化处理,再应用前述的归并策略可将单处理机上的串行工程,有效地移植到共享存储器多处理机系统上,提高程序的计算效率。

(2)程序段的前归并和后归并策略,使同串行号程序段的运行时间开销相对均衡,且避免了片面强调并行性——细粒度的并行运算而得不偿失地发生。采用处理机主频和主存存取周期与指令的平均执行时间的函数关系及处理机数目作为评判是否归并的因素,使编译系统直接与机器实体结合,针对实体机器进行优化,降低了工程并行计算及调度的时间开销,更主要的是降低了整个工程的时间开销。

(3)前面假定机器字长为64位,处理机数目为256,程序段数 $>4\times256$。以4倍处理机数个程序段进行分页,也可以使用1倍、2倍……进行分页。使用1倍的分页每页包含的程序段太少,倍数取得太大,每页包含的程序段太多,同样会降低调度效率。一般取3或4倍。对处理机数 < 机器字长,程序段数少于一页的情形采用类似的手段都能建立计算调度表和状态字生成表副本,实现相应的编译和调度。

4.2 存在的问题

(1)正如上所述,编译策略直接针对实体机器进行优化,在这台共享存储器多处理机上编译的程序到另一台机器上执行,因处理机主频和主存存取周期变化,因而不一定也是优化的。例如,处理机主频发展较快,而主存频率相对发展迟缓,它影响指令平均时间开销与处理主频和主存存取周期的函数关系,从而影响归并的决策,因而要使其优化,就必须在该处理机上重新编译。

(2)归并优化策略,增大了编译的时间开销。

(3)工程在调度处理机处的并行计算调度表一次最多仅能调入处理机一页程序段的信息。当下一页的部分程序段并不需要先调入程序段全部执行完成后才能提交执行时,操作系统不能调度这些程序段,这样将增加计算的时间开销,甚至可能造成不能充分利用处理机资源。若以 1 倍处理机数个程序段进行分页,也不能充分利用处理机资源;若倍数过大(>4),将增加操作系统搜索、修改并行计算调度表的时间开销。

(4)编译的优化策略受到程序设计人员设计程序并行度的影响较大。

参考文献:

[1] 李学干. 计算机系统结构[M]. 第 3 版. 西安:电子科技大学出版社,2001.

[2] HWANG K. Advanced Computer Architecture——Parallelism Scalability Programmability[M]. New York:McGraw-Hill Companies,1993.

[3] STALLINGS W. Operating System——Internals and Design Principles[M]. 4th ed. New York:Prentice Hall,2001.

[4] 严蔚敏,吴伟民. 数据结构[M]. 第 2 版. 北京:清华大学出版社,2002.

[5] 董文泉. 数量化理论及应用[M]. 长春:吉林人民出版社,1979.

[6] 吕映芝,张素琴,蒋维杜. 编译原理[M]. 北京:清华大学出版社,2000.

[7] 于勐,阳雪林,臧婉瑜,等. 基于方法调用一般化模型的并行性分析[J]. 计算机学报,2002,25(4):403-408.

[8] 陈国良. 并行算法的设计与分析[M]. 修订版. 北京:高等教育出版社,2000.

[9] 张广泉. 并发程序模型的可信性研究[J]. 重庆师范学院学报(自然科学版),2000,17(1):12-18.

(责任编辑　游中胜)