一种并行性检测算法

陈镐缨
(西北工业大学计算机软件教研室,西安 710072)

A ALGORITHM FOR PARALLELISM DETECTION

Chen Haoying
(*Northwestern Polytechnical University, Xian* 710072)

Abstract A new and practical algorithm for FORTRAN parallelism detection is described. In this paper, a array—dependence—on—loop concept that is important for detecting parallelism is presented first. Some new theories of detecting parallelism as well as a practical decision algorithm are then given.

摘要 本文介绍一种用于FORTRAN并行化的实用算法.本文探讨串行源程序中的循环横向切分并行化的理论和算法.我们建立了一套新的理论概念,并且在这套概念的基础之上,导出了一系列横向切分理论.最后,文章还根据这些并行化理论得出的结果,介绍了一种实用的循环检测并行化方法.

§1. 基本概念及约定

在串行程序循环的横向并行化过程中,由于要对循环变量的取值范围及方式进行重新确定,故应考虑循环体中循环变量为不同值时数组分量的更新与引用情况.为明确起见,不失一般性,下面给出两种循环程序段模型:

```
      DO  I=L,U,H                    DO  I=L,U,H
S1:   A(f(I))=…                S1:   …=E(A(f(I)))
S1:   …=E(A(g(I)))             S1:   A(g(I))=…
      END  DO                        END  DO
      (Ⅰ类循环)                      (Ⅱ类循环)
```

其中L、U、H分别为整型的循环初值、终值和步长,A为数组变量;E(A)为任意合适的表达式f(I)、g(I)是循环变量I的线性函数:

$$f(I)=a_1I+b_1, g(I)=a_2I+b_2,$$

这里a_1、a_2、b_1、b_2为整数.对于在任一次循环中A元素的更新总是先于引用的循环,我们称其为Ⅰ类循环;而引用与更新次序恰好相反的循环,我们称为Ⅱ类循环.注意,Ⅱ类循环中的语句s_1、s_2也可能退化为一条语句.

为节省篇幅起见,以上这些约定的符号,在本文中将直接引用.

定义 1:一个循环程序段定义了一个迭代空间W,它由INT((U—L)/H)个被称为迭代元素的

本文1990年12月1日收到,1991年2月27日定稿.本文得到了国家自然科学基金资助.作者系霍英东青年教师奖获得者.
陈镐缨,副教授,主要研究领域为并行计算、管理信息系统(MIS)、CAI、文字处理等.

离散点构成(INT是截断取整函数).每个迭代元素表示循环体的一次执行,并定义该次循环时循环变量的值为相应迭代元素的值及名称.

定义 2:迭代空间是有序的.迭代元素K的序号用Φ(K)表示,且有:

$$\Phi(K)=(K-L)/H.$$

关系符$<'$定义为:当$\Phi(K_1)<\Phi(K_2)$时,有关系式$K_1<'K_2$,它描述了对应元素的执行顺序.

根据以上两个定义,对于前面的两个程序段模型,它们对应的迭代空间为:

$$W=\{i|i=kH+L,k=0,1,\cdots,(U-L)/H\};$$

其元素按序排列为:$L<'H+L<'2H+L<'3H+L<'\cdots<'U$.显然,W也是循环变量I的值集合.

定义 3:对于任意数组分量A(I),如果它在迭代元素K_1中被修改之后,又在迭代元素K_2中被引用,这里$K_1<'K_2$,则称该循环与数组A相关,并称迭代元素K_1与迭代元素K_2相关,或称第$\Phi(K_1)$次循环与第$\Phi(K_2)$次循环相关;否则称该循环独立于数组A.

因此,循环横向切分实质上是在保证每次循环的输入集合和输出集合不变的前题下对迭代空间进行的划分,此划分要求所有相关的迭代元素均划分在同一个块中.显然,当循环独立于其体内的所有数组变量时,允许对此循环进行任意方式切分.例如:

```
DO   I=1,10000,1
     A(6*I)=2*I*I
     B(I)=A(2*I+3)/2
END  DO
```

这是一个循环独立于A和B的程序段,故可对其进行任意切分.下面是二种切分方案:

```
法1:DO  I=1,5000,1          DO  I=5001,10000,1
法2:DO  I=1,9999,2          DO  I=2,10000,2
```

其中法1将迭代空间元素简单地以按序等分的方法进行划分,称为简单横向切分.这种切分要求循环独立于其体内的所有数组变量.法2是按"模2同余"的关系对迭代空间进行的一种划分.一般地,对迭代元素值按照模m同余的关系划分迭代空间,将得到m个划分块,这对应于将循环程序段切分为m个并行任务块,每个任务块的循环初值与对应划分块中第一个迭代元素的值相同,而终值与对应划分块中最后一个迭代元素的值相同.任务块的步长为$m\times H$.这种横向切分称为同余横向切分,它能解决部分循环与数组相关的情况.

定义 4:所谓分段横向切分,是指对迭代空间这样的划分:部分划分块集合对应的任务块将保持串行执行,而另一部分划分块集合将进一步被横向切分.设迭代元素K_1是迭代元素K_2的直接前趋,并且它们属于不同的划分块集合$\prod_1$和$\prod_2$,则称K_1是$\prod_1$和$\prod_2$的界点元素.

§2. 横向切分定理

在本节将给出若干个横向切分的判定定理,因篇幅所限,详细证明省略.

定理 1:对于Ⅰ类和Ⅱ类循环,下式不成立是循环独立于数组A的充分条件:

$$\gcd(a_1,a_2)\mid b_1-b_2$$

例如,对于前例,gcd(6,2)=2,不能整除3,所以该循环独立于数组A.

定理 2:令$h(x,y)=f(x)-g(y)$,h_1及h_2为$h(x,y)$在某一相关解集合中的最大和最小值,则当$h_1h_2>0$时,相应循环独立于数组A.

尽管有些循环与数组A相关,不能按原步长H进行横向切分.但是,如果将其步长按一定的关

系修改，就可以对它进行同余横向切分. 定理 4 给出了同余横向切分的判定条件.

定理 3：对于 I 类或 II 类循环，当 $a_1=a_2$ 且 $b_1\neq b_2$ 时，令：

$$N=|(b_1-b_2)/(a_1H)|$$

则以 N 的因子 n 与 H 之积为步长可对循环按模 n 进行同余横向切分.

应当明确，当按模 n 同关系划分后，各并行任务块的循环步长为 $n\times H$. 显然按定理 4 求出 N 值后，将迭代空间按模 N 同余的关系划分能获得最大的并行度. 因将得到 N 个并行任务块，所以此时最佳的硬件配置是系统具有 N 个并行处理机.

定理 4：对于与数组 A 相关的 I 类或 II 类循环，若 $a_1\neq a_2$，令：

$$x=(b_1-b_2)/(a_1-a_2)$$

$$k=L+INT((x-L)/H)*H,$$

则当 $L\leqslant x\leqslant U(H>0)$ 或 $U\leqslant x\leqslant L(H<0)$ 时，循环可被分段横向切分. 而当 $H>0$ 时界点元素为 k，否则为 k 的直接前趋.

§ 3. LHPD 算法

根据前面给出的定理，我们设计出了一个实用的横向切分的算法：

1. 当 $(a_1=a_2)\wedge(b_1=b_2)$ 时，可以对两类循环进行任意横向切分，终止.

2. 若 $gcd(a_1,a_2)|(b_2-b_1)$ 为假，则可对循环进行任意横向切分，终止.

3. (1)对于 I 类循环，若 $H>0$，计算(关于函数 a^+ 和 a^- 的定义见[6])：

$$h_1=(a_1-a_2)L-a_2+b_1-b_2+(a_1^+-a_2)^+(U-L-1)$$

$$h_1=(a_1-a_2)L-a_2+b_1-b_2-(a_1^-+a_2)^+(U-L-1)$$

否则 $h_1=(a_1-a_2)U+a_1+b_1-b_2+(a_2^-+a_2)^+(L-U-1)$

$$h_2=(a_1-a_2)U+a_1+b_1-b_2-(a_2^+-a_2)^+(L-U-1)$$

(2)对于 II 类循环，若 $H>0$，计算：

$$h_1=(a_1-a_2)L+a_1+b_1-b_2+(a_2^-+a_1)^+(U-L-1)$$

$$h_2=(a_1-a_2)L+a_1+b_1-b_2-(a_2^+-a_1)^+(U-L-1)$$

否则 $h_1=(a_1-a_2)U-a_2+b_1-b_2+(a_1^+-a_2)^+(L-U-1)$

$$h_2=(a_1-a_2)U-a_2+b_1-b_2-(a_1^-+a_2)^+(L-U-1)$$

则当 $h_1h_2>0$ 时，可对循环进行任意横向切分，终止.

4. 当 $(a_1=a_2)\wedge(b_1\neq b_2)$ 时，计算 $N=|(b_1-b_2)/(a_1H)|$，对循环以 N 的因子 n 与 H 之积为步长按模 n 进行同余横向切分，终止.

5. 计算 $x=(b_2-b_1)/(a_1-a_2)$，若 $L\leqslant x\leqslant U$(或 $U\leqslant x\leqslant L$)，则循环可被分段横向切分，转步骤 7.

6. 对于 I 类循环，若 $(x<L)\wedge(x<U)\wedge[(H>0)\wedge(a_1<a_2)\vee(H<0)\wedge(a_1>a_2)]\vee(x>L)\wedge(x>U)\wedge[(H<0)\wedge(a_1<a_2)\vee(H>0)\wedge(a_1>a_2)]$ 为真时，可对循环进行任意横向切分，终止；否则，不能并行化，终止.

对于 II 类循环，当 $(x<L)\wedge(x<U)\wedge[(H<0)\wedge(a_1<a_2)\vee(H>0)\wedge(a_1>a_2)]\vee(x>L)\wedge(x>U)\wedge[(H>0)\wedge(a_1<a_2)\vee(H<0)\wedge(a_1>a_2)]$ 为真时，可对循环进行任意横向切分，终止；否则，不能并行化，终止.

7. 令 $k=L+INT((x-L)/H)*H$，若 $H<0$ 则 $k=k-H$.

对于 I 类循环，当 $a_1<a_2$ 时，保留 $I=L\sim k$ 的循环串行执行，而可对其后的循环进行任意横向

切分；当 $a_1>a_2$ 时，可对 I=L～k 的循环进行任意横向切分，而剩余部分串行执行，终止.

对于 Ⅱ 类循环，当 $a_1<a_2$ 时，保留 I=L～k 的循环进行任意横向切分，而剩余部分串行执行；当 $a_1>a_2$ 时，保留 I=L～k 的循环串行执行，而可对其后的循环进行任意横向切分，终止.

§4. 后 记

循环是程序中最大的并行性来源，特别是若整个程序的绝大部分时间都耗费在少数几个循环中，这种情况并行的效率最高，LHPD 对于多重循环在处理上与单重循环没有什么根本的差别，而分割这类循环的外层时可获得较高的并行效率. 对于一个可切分的循环，按照嵌套层次以先外后内的原则进行处理，直至第一次找到满足并行化条件的循环层.

最近，我们已在 Transputer 并行系统中实现了此算法的实验系统 FPS. 经对多个程序进行的试验证明，此算法具有较强的实用性. 如对于一个约 200 行语句的 FORTRAN 程序，并行化处理仅耗费约 2 分，并且其中至少有 5/4 的时间耗费在磁盘 I/O 上. 另外，它能有效地提高那些将大部分运行时间耗费在个别几个循环语句的程序的运行速度. 由于在并行化中仅对程序中少数几个最大的循环进行处理，所以不仅使得算法及其及实现大为简化，同时又避免了因产生过多并行任务碎块而实质上降低了系统运行效率的弊病. 我们曾对一个矩阵幂运算的程序做并行化处理，最终得到的并行程序并行效率达 86%.

参考文献

[1] Chen haoying, A Algorithm for Parallel Decision, Proceeding of the Fourth Japanese—Sino Sapporo International Conference on Computer Applications, 1990.

[2] J. C. Browne, Muhammed Azam and Stephen Sobek, CODE; A Unified Approach to Parallel Programming, IEEE Software, July 1989.

[3] Vincent A, Faust; An Integrated Environment for Parallel Programming, IEEE Software, July 1989.

[4] Bill Appeibe and Kevin Smith, Start/Pat: A Parallel—Programming Toolkit, IEEE Software, July 1989.

[5] Herbert C. Conn, Software Tools and Techniques for Embedded Distributed Processing, Noyes Publication, 1986.

[6] Randy Alley and Ken Kennedy, Automatic Translation of FORTRAN Program to Vector Form, ACM Transactions on Programming Languages and Systems, Vol. 9, No. 4, October 1987, 491—542.

[7] Kai Hwang, Advanced Parallel Processing with Supercomputer Architectures, Proceedings of the IEEE, Vol. 75, No. 10, October 1987.