国防科技大学学报
JOURNAL OF NATIONAL UNIVERSITY OF DEFENSE TECHNOLOGY
第18卷第1期 1996年3月 Vol. 18 No. 1

面向 MPP Fortran 的程序自动并行化初探*

郭克榕 唐新春
(国防科技大学计算机系 长沙 410073)

摘 要 MPP Fortran 是由 Cray 公司推出的一种较有代表性的数据并行语言，本文首先介绍了 MPP Fortran 的主要特点，然后，以该语言为例，对面向 MPP 系统程序自动并行化的主要内容进行了初步的探讨。

关键词 程序自动并行化，数据依赖关系分析，数据分布，循环迭代划分，同步控制
分类号 TP312

Automatic Program Parallelization for MPP Fortran

Guo Kerong Tang Xinchun
(Department of Computer Science, NUDT, Shangsha, 410073)

Abstract MPP Fortran is a data parallel programming language released by Cray Research, Incorporated. This paper introduces at first the major features of MPP Fortran. Then based on that, it analyses and discusses the main content as well as the key technique of automatic program parallelization for MPP systems.

Key words automatic program parallelization, data dependence analysis, data distribution, loop iteration distribution, synchronization control

程序自动并行化是并行计算领域一个重要研究课题，是克服并行计算机编程困难、软件移植困难的主要手段，也是充分发挥并行计算机系统潜能的有效途径。

近年来，大规模并行处理机（MPP）得到迅速发展。这种基于分布存储的多机系统为程序自动并行化增加了新的内容，也带来了新的技术难题。例如：如何为一个应用程序确定一个好的数据分布和并行计算模式，以获得最大的数据局部性和程序并行性？如何进行同步及通讯控制以保证并行程序的正确性？这些问题具有一定的理论研究深度和工程实现难度，面向 MPP 系统的程序自动并行化研究仍处于探索阶段。

* 国防科技预研基金资助项目
1995年4月28日收稿

目前，在MPP系统中使用最多的并行程序设计模式有两类：消息传递模式和数据并行模式。近几年来推出的数据并行语言，如HPF[1]、CM Fortran[2]、Cray MPP Fortran[3]等，都支持数据并行编程模式。与消息传递程序设计相比，数据并行程序设计能在一定程度上减轻用户的负担，但是完全依赖用户确定一个好的数据分布和并行划分模式也非常困难，且容易出错。因此，研究面向MPP系统的程序自动并行化技术，改变应用软件并行化完全依赖于用户的状态非常必要。本文将以Cray MPP Fortran (CRAFT)语言为例，讨论如何将串行Fortran 77程序自动并行化为数据并行的CRAFT程序。

1 CRAFT的主要特点

1.1 并行模式与同步机制

CRAFT采用典型的SPMD并行模式，同一个程序配以合适的数据分布和循环迭代划分，同时在多个PE (Processor Element) 上执行。每个执行流对应一个任务。

CRAFT程序可分为多个串行区和并行区。串行区有显式说明 (CDIR $ MASTER与CDIR $ END MASTER)。程序一启动，所有任务就产生(任务数等于该程序获得的PE数)，并且以并行方式开始执行。当进入串行区以后，只有PE_0上的主任务执行，其它任务在串行区的出口处等待。进入下一个并行区时，所有的任务又开始并行执行。程序最后在并行区中结束。

程序在并行区又有共享执行和冗余执行两种执行方式。以共享方式执行时，所有的任务执行相同的代码与不同的数据；冗余执行时，所有任务执行的代码和数据完全相同。

SPMD方式的并行程序使用多个PE解决同一个应用问题，因此每个PE上的任务不可能是完全独立的，而必须作为一个整体的各个部分互相协调。当相互间有某种依赖关系时，需要加入同步控制。CRAFT提供了一组标准的类似共享存储系统的同步机制，如障碍、事件、临界区和锁同步，以保证程序并行执行的正确性。

1.2 数据对象

CRAFT支持程序并行执行时引入的两种数据属性：私有属性和共享属性。私有数据不分布，每个PE上有一个副本，它只能由拥有该副本的PE上的任务访问。共享数据不能被复制，可以被所有任务访问。共享标量不能分布，只能放在某个PE上。共享数组按一定的分布方式分布到多个PE上，支持多个任务的并行执行。

共享数据必须用SHARED显式说明，其语法为：

$$\text{CDIR \$ SHARED } var_1, var_2, \cdots, var_n$$

如果共享数据是数组，需说明其按维分布的方式。

1.3 循环共享

循环分为私有循环和共享循环。私有循环由执行该循环的任务独立完成，而共享循环由所有的任务共同完成，每个任务完成一部分循环迭代，迭代的执行次序任意。

私有循环可以在串行区，也可以在并行区。如果在串行区，只能由PE_0上的主任务执行；如果在并行区，则由所有的任务冗余执行。共享循环一般都在并行区，如果出现在串行区，则当成私有循环处理。

共享循环必须用DOSHARED显式说明，语法如下：

CDIR $ DOSHARED (I_1,I_2,…,I_n)ON x($f_1(\bar{I})$,$f_2(\bar{I})$,…,$f_r(\bar{I})$)

其中 n 为循环嵌套的个数，这 n 个循环必须是紧嵌套的。I_1～I_n 为循环控制变量，r 为数组 x 的维数，$f_1(\bar{I})$～$f_r(\bar{I})$为数组下标表达式。

ON 子句用来说明循环迭代的划分方式，编译器将循环迭代按照 x($f_1(\bar{I})$,…,$f_r(\bar{I})$)的分布方式划分到所有任务上执行。也就是说，使得循环执行时，每个 PE 对 x($f_1(\bar{I})$,…,$f_r(\bar{I})$)以及其它与 x($f_1(\bar{I})$,…,$f_r(\bar{I})$)对准分布的数组元素的访问均是局部的。

1.4 数组操作

CRAFT 提供数组操作，包括数组赋值语句和数据并行函数。如果数组是共享的，数组赋值语句就相当于一个共享循环，但与共享循环不同，它的迭代划分完全由编译器控制。编译器选择最佳的划分方式以获得最大的数据局部性。如果数组是私有的，则数组赋值语句相当于一个私有循环。

数据并行函数是数组操作的另一种形式。如果数据并行函数在串行区或共享循环中引用，则计算由执行该函数的任务独立完成，否则视数组的属性而定：如果数组是共享的，则计算由所有任务共同完成；如果数组为私有的，则计算由执行该函数的任务独立完成。

2 面向 CRAFT 的程序自动并行化

传统的程序自动并行化研究，主要集中在数据依赖关系分析和程序变换这两个方面。经过十几年的研究，数据依赖关系分析技术和程序变换技术取得了很大的成就[4][5]。一些成功的技术不仅适用于共享存储多机系统，同样也适用于分布存储的 MPP 系统。但是 MPP 系统毕竟在体系结构上有其特殊性，因而面向该类系统的程序自动并行化有许多新的技术难题。下面将以串行 Fortran 77程序自动并行化为 CRAFT 描述的数据并行程序为例，对这些问题展开讨论。

2.1 DO 循环的并行化

串行 Fortran 77程序中固有的并行性主要体现在 DO 循环结构中，而采用并行程序设计语言编写的程序，则正是用其所特有的并行结构来描述这种并行性。例如，CRAFT 中的数组操作和 DOSHARED 循环，就是这种显式的并行结构。因此，要将串行的 Fortran 77程序自动并行化为并行的 CRAFT 程序，其主要目标就是将串行的 DO 循环转换为并行的数组操作或 DOSHARED 循环，以开发串行程序中的并行性。

数组赋值语句在语义上等同一个 DOSHARED 循环，但由于其相应的迭代划分由编译器选择最优方式，所以在并行化时应尽量识别出程序中蕴含的数组赋值操作。

例1
```
DO I=1, N                    A(I)=B(I)+C(I)   ⇒   A=B+C
ENDDO
```

对于一些完成特殊运算的循环，可对其运算结构进行模式匹配，将其中某些循环用 CRAFT 提供的数据并行函数替代。

例2
```
X=0.0
DO I=1, N                    X=0.0
      X=X+A(I)          ⇒    X=X+SUM(A)
ENDDO
```

对于不能转换成数组操作的其它可并行化循环，则将其转换为 DOSHARED 循环。

例3

```
DO I=1, N                                  CDIR $  DOSHARED (I) ON X(I)
      X(I)=Y(I) * *4                               DO I=1, N
      Y(I)=A * X(I)+Y(I)          ⇒                X(I)=Y(I) * *4
ENDDO                                              Y(I)=A * X(I)+Y(I)
                                                   ENDDO
```

数据依赖关系分析技术是并行化的基础，近十几年来发展很快，如 gcd 测试、Banerjee 逐维测试等，在共享存储的向量并行编译器或工具中得到了很好的应用。这些技术对于分布存储系统的并行化也同样有效。就 MPP 系统而言，还需更深入地研究精确而高效的数据依赖关系分析技术。这包括复杂下标依赖关系分析技术(如耦合下标的各维联立测试法，非线性下标的符号分析法)，过程间相关性分析技术（如过程间变量别名分析、常数传播等)，非循环区域或不规则循环区域（非紧嵌套多层循环、复杂区域循环）的依赖关系分析技术，依赖关系增值分析技术等。

2.2 数据分布与并行划分

识别出串行程序中可并行化的 DO 循环，要使这些循环能够高效地由处理机集合并行执行，必须用 CRAFT 提供的数据分布和并行划分机制对这些循环所访问的数组进行分布，对循环的迭代空间进行划分。

首先，对可并行化的循环所访问的数组进行对准分析，将那些大小、维数、访问方式有某种确定关系的数组归为一类。然后，进行分布分析，给每类数组制定出分布方案，并且将循环中访问频度最高的数组的访问模式，作为该循环迭代空间划分模式。一旦分布与划分的候选方案确定，就可按照静态性能评估模型。根据该循环的计算量、通讯量、处理机的计算能力、互联网的通讯能力和处理机数，来估算循环的执行时间，以确定针对该循环的最佳数据分布与并行划分模式。在工程实现时，为减少系统开销，可能要以多个循环或整个程序段为单位确定数据分布方案。为此，要对上述各个循环的数据分布方案进行归并，并按照归并的数据分布方案，对各循环的划分模式进行相应的调整[6]。

下面以矩阵乘为例，说明如何确定合理的数据分布与并行划分模式。

经数据依赖关系分析，例4中的多层紧嵌套 DO 循环可以并行化。因为是对二维数组进行运算，所以最多可有两层循环并行。显然，选择外两层循环可获得最大的并行粒度(DOSHARED (K,I))。为获得最大的计算并行性，将 C 数组在处理机间进行均匀分布(C(:BLOCK, :BLOCK))。为获得最佳的数据局部性，迭代空间(K,I)按 C 数组划分(ON C(I,K))，而且 A 数组按行均匀分布、列不分布(A(:BLOCK,:)，因为一次并行运算用到一行元素)，B 数组按列均匀分布，行不分布(B(:,:BLOCK)，因为一次并行运算用到一列元素)。

并行化后的程序如下：

例4

```
CDIR $  SHARED A(:BLOCK,:), B(:,:BLOCK), C(:BLOCK,:BLOCK)
CDIR $  DOSHARED (K,I) ON C(I,K)
        DO K=1, N
```

```
      DO I=1, L
            DO J=1, M
                  C(I,K)=C(I,K)+A(I,J) * B(J,K)
            ENDDO
      ENDDO
ENDDO
END
```

MPP 系统具有极强的并行计算能力和大容量的分布主存，要使这些系统特性得以充分发挥，必须尽可能地减少处理机间的通讯开销，维持处理机的负载平衡。一个应用程序的数据分布与并行划分模式，直接影响着程序并行性的开发、处理机间的通讯量和计算的均衡性。因此，自动数据分布与并行划分是面向 MPP 系统程序并行化的一项关键技术。

2.3 变量属性分析与并行控制

从上例中可见，将串行 DO 循环并行化为 DOSHARED 循环，并且确定需分布的数组之后，需对程序中的变量属性进行分析与说明。CRAFT 提供了这种说明能力，如上例中的数组 A、B、C 均被说明为共享数组。

变量有共享与私有之分，这是并行程序的特点之一。针对循环而言，私有变量是在共享循环的一次迭代中先定义后引用的变量，循环中除私有变量以外的变量均为共享变量。共享数组的分布方式与循环的并行方式密切相关。由此，变量属性的分析可相对于循环而进行。但要确定一个变量在整个程序段的变量属性，则要对该变量在段内所有循环的属性分析信息进行归并。在归并分析过程中，尤其要注意的是程序并行性与数据局部性的折衷。另外，由于私有变量有多份拷贝，当它的值在共享循环之后还要引用，则需要在共享循环出口处保留最后一次迭代计算的值。见下例。

例5

```
DO I=1, M                          CDIR $  SHARED A(:BLOCK)
      T1=SIN(A(I)) * 2             CDIR $  SHARED C(:BLOCK)
      C(I)=T1+C(I)/T1              CDIR $  DOSHARED (I) ON C(I)
ENDDO                                      DO I=1, M
T2=T1+S                          ⇒               T1=SIN(A(I)) * 2
                                                 C(I)=T1+C(I)/T1
                                           ENDDO
                                           T1=SIN(A(M)) * 2
                                           T2=T1+S
```

以冗余方式执行程序，对共享变量进行访问时，当共享变量出现在非共享循环区，则产生共享变量一致性问题。所以，要保证程序的正确性，在程序并行化时需加以适当的并行控制。CRAFT 的串行区说明与同步原语对此提供了支持。

经程序分析，例6中的 DO 循环可并行化。将迭代区间进行划分，由各处理机求出 A、B 两数组的部分点积之和。为此，增设临时变量 ST 存放部分结果，并将其说明为私有变量（缺省），A、B 两数组定为共享数组，结果变量 S 定为共享标量。为保证该段程序并行执行时的正确性，加以适当的并行控制如例6中右部所示。

例6

```
S=SO                                    ⇒   CDIR $  SHARED A (:BLOCK)
DO I=1, N                                   CDIR $  SHARED S, B (:BLOCK)
      S=S+A(I) * B(I)                       CDIR $  MASTER
ENDDO                                               S=SO
                                            CDIR $  END MASTER
                                                    ST=0. 0
                                            CDIR $  DOSHARED (I) ON A (I)
                                                    DO I=1, N
                                                        ST=ST+A (I) * B (I)
                                                    ENDDO
                                            CDIR $  CRITICAL
                                                    S=S+ST
                                            CDIR $  END CRITICAL
```

3 结束语

本文仅对串行程序并行化为 CRAFT 数据并行程序的基本内容进行了初步的探讨，仍遗留了一些问题未作深入研究。例如，数据共享给 Fortran 程序中的等价、公用带来了新的限制，给数据依赖关系分析增加了复杂性。共享数据的分布与跨越函数和子程序的数据再分布，给过程间分析和程序优化带来了新的困难。

并行程序还有进一步优化的问题。大多数面向共享存储系统的并行程序优化技术，对面向分布存储的 MPP 系统都行之有效。除此之外，面向 MPP 系统的程序并行化，还应重点考虑数据局部性优化、同步通讯优化、负载平衡优化，以进一步提高并行程序在 MPP 系统的运行效率。

参 考 文 献

1 High Performance Fortran Forum, High Performance Fortran Language Specification. Technical Report CRPC－TR92225, Center for Research on Parallel Computation, Rice University, Houston, TX., 1993

2 Thinking Machine Corporation, Cambridge, Massachusetts, CM Fortran Referance Manual, 1991

3 Douglas M. Phase, Tom Mcdonald and Andrew Meltzer, MPP Fortran Programming Model. Cray Research, Inc.. Eagan, Minnesota, 1993

4 Polychronopoulos C, Girkar M, Haghighat MR, etal. Paratrase－2: A new generation parallelizing compiler. Int. J. High Speed Computing, 1989: 45～72

5 Allen JR, kennedy K. Automatic translation of Fortran programs to vector form. ACM Transactions on Programming Languages and Systems, 1987: 491～542

6 Kremer U. Automatic Data Layout for Distributed－Memory Machines. Technical Report, CRPC－TR93299, Center for Research on Parallel Computation, Rice University, 1993

（责任编辑　朱宝龙）