计算机科学 1996 Vol. 23 №. 2

HPF——FORTRAN 最新版本

李新颖 李晓明

(哈尔滨工业大学计算机系 哈尔滨150001)

摘 要 HPF(高性能 FORTRAN)公布于1993年,是当今世界上 FORTRAN 语言的最新版本。本文系统地介绍了 HPF 相对于以往 FORTRAN 版本(主要是 FORTRAN 90)的新特点,并举出了一个程序实例,供并行计算技术的研究者以及工程技术人员参考。

关键字 FORTRAN 90,HPF,并行计算,数据划分。

FORTRAN 语言自从本世纪五十年代从 IBM 破土而出之后,一直备受工程技术人员的青睐,成千上万用 FORTRAN 语言编写的大型应用程序在全球各地被广泛使用,至今 FORTRAN 仍旧被认为是最受欢迎的数学计算语言。高性能 FORTRAN(High Performance FORTRAN,即 HPF)是这种古老语言的最新扩展集,除汇集了以往 FORTRAN 的全部特性之外,还具有能够实现数据在并行处理机上的划分、数据的并行操作等适用于现代各种计算机结构的语言特色,在国外被通过数据并行程序模型开发并行计算技术的研究人员所广泛采用。HPF 公布于1993年,目前尚不是一个被国际标准化组织承认的标准,当前 ANSI 和 ISO 认可的 FORTRAN 语言标准是 FORTRAN 90,它公布于1990年。

一、从 FORTRAN90到 HPF.

我们知道 FORTRAN90是在 FORTRAN77的基础上扩展了一些数据结构,如:数据指针、结构类型,和并行语句,如:数组操作 A=B*C,已经可以说是一种并行度很高的语言了。但分析起来它有三个弱点:

(1)在数据并行执行上,它对整个数组的操作可以用一条语句很方便地完成。如 sin(A),但对数组内个别元素的操作就无法很简便地并行化。如:对一向量中的个别元素求反,用 FORTRAN90就要写成:

```
DO I=1,N,2
  X[I]=-X[I]
```

这样的串行语句。

(2)它对数据的划分不做要求,这样在程序执行时就不能保证获得最高的性能。

(3)它没有为用其它语言写的程序提供接口。即使一段程序用其它语言可以更好地表达,也无法嵌入到 FORTRAN 程序中。

HPF 针对这些弱点进行了扩展,增加了并行机制(FORALL),数据划分的机制(DISTRIBUTE),以及为 HPF 和其它语言提供接口的方法等。这样上面(1)中的问题就可以用 HPF 写成并行语句:

```
FORALL I=1:N:2
  X[I]=-X[I]
```

可以看出 HPF 在 FORTRAN90基础上增加的这些内容事实上就是在并行计算技术的研究中也曾被广泛接受的以数据划分为特色的FORTRAN版

李新颖 硕士研究生,李晓明 教授,博士生导师。

[10]M. M. Burnett et al., Scaling Up Visual Programming Languages, IEEE Computer, Vol. 28, No. 3, 1995

[11]G. Karsai, A. Configurable Visual Programming Environment: A Tool for Domain-Specific Programming, same to [10]

[12]G. Costagliola et al., Automatic Generation of Visual Programming Environments, same to [10]

[13]A. Repenning, T. Sumner, Agentsheets: A Medium for Creating Domain-Oriented Visual Languages, same to [10]

[14]强军,王兵山,李舟军,可视语言:把思想变成程序的工具,计算机科学. Vol. 21, No. 4, 1994

本--FORTRAN D。因此有人称 HPF 为 FORTRAN 90D。

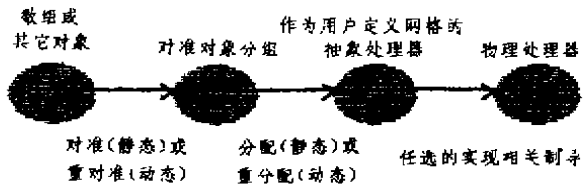
二、HPF 的新特色

现代科学研究与工程项目中经常会面对大量复杂的运算，HPF 正欲成为一种在各种类型计算机，包括传统的向量处理器组，最新的大型 MIMD、SIMD 机器上解决大规模计算问题的标准编程语言。在 HPF 以前的各种并行语言都比较紧密地依赖于机器的结构，比如 PCF 研制的 X3H5 FORTRAN 就依赖于共享存储器的体系结构，由程序员描述同步、通讯等细节问题，HPF 则把这种负担完全交给了编译器，用户只须提供数据映射策略，由编译系统产生通讯细节，控制并行。

下面主要介绍 HPF 在 FORTRAN90基础上扩展的新特色：

1. 数据划分(Data Distribution)

学过并行程序设计我们都知道数据划分就是告诉编译器如何把数据分配到各个处理器上运行，合理的数据划分可以减少通讯量，大大降低整个运算的开销。HPF 采用的是两级映射的机制：数据对象(典型的如数组元素)首先被映射到相关的另一个对象上；然后这个数组被分配到线性安排的抽象处理器组上；抽象处理器组到物理处理器组的映射是依赖于系统的，并且后者可以使用等于或小于前者的处理器个数。这使得编程更加灵活，程序独立于机器(如下图)。



下面就介绍一下如何用 HPF 来实现这种数据的划分：

(1)首先 HPF 定义了一个模板 TEMPLETE，这是一个抽象数组，它的作用就是为数据对准时提供一个模板。我们可以说明：

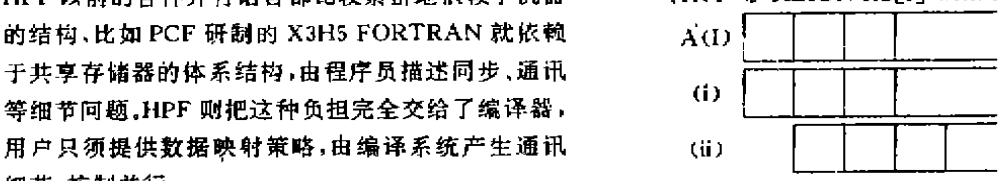
```
!HPF $ TEMPLETE A(N)
!HPF $ TEMPLETE,DIMENSION ::B(N,N)
```

或者干脆把数据的划分也定义出来：

```
!HPF $ TEMPLETE, DISTRIBUTE (BLOCK)::C(N)
```

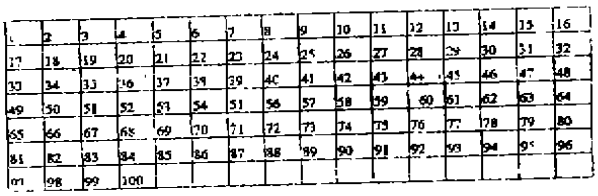
(2)数据对准 ALIGN。ALIGN 语句可以将某些数据对象映射到另一些对象上，使它们一一对应。如：

```
!HPF $ ALIGN X1[I] with A(I)        (i)
!HPF $ ALIGN X2[I] with A(I+1)      (ii)
```

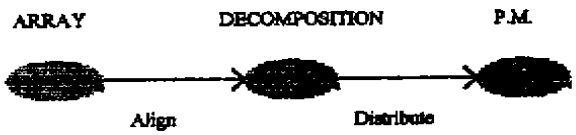


(3)数据分配 DISTRIBUTE。在数据对准后就可以把它们分配到抽象处理器组上。DISTRIBUTE 可以把一个数据对象或一个 TEMPLETE 模板分配到处理器上。如：

```
      REAL AA(100)
!HPF $  PROCESSORS P(16)
!HPF $  DISTRIBUTE AA(CYCLIC) ONTO P
```



由以上三条语句可以看出 HPF 与 FORTRAN D 的数据划分并不完全一致。在 FORTRAN D 中必须将数组对准(ALIGN)到一个模板(DECOMPOSITION)上，(数据对象之间不能直接 ALIGN.)然后再将模板映射到具体的物理机器上(下图)。



HPF 对 FORTRAN D 的这种结构做了改动，使得 HPF 更能适用于各种机器结构，并且不受物理处理器个数的限制，方便用户解题。

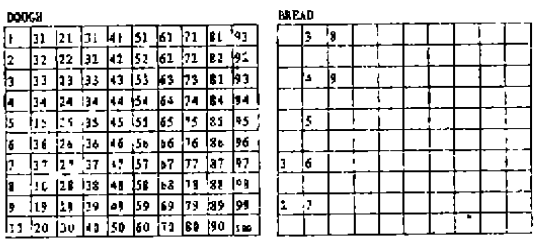
(4)HPF 的动态指令 DYNAMIC,以上介绍的 ALIGN,DISTRIBUTE 都被认为是说明语句,它只是给了编译器一个提示,当在程序中需要改变数据对象之间的对准关系,以减少数据移动时,我们就要重新 ALIGN 和 DISTRIBUTE。在 FORTRAN D 中,允许 ALIGN 和 DISTRIBUTE 又作为可执行语句出现在程序中间来改变数据划分,这样它们既可作为说明语句又可作为可执行语句,事实上,在一种语言标准中做这种规定并不是一种很严谨的做法。

HPF 采用的是 DYNAMIC 动态说明指令,它把可能需要重新划分的数据定义为 DYNAMIC,只有对这种数据才能在程序中用 REALIGN 和 REDISTRIBUTE 这两条可执行语句进行重新划分。比如:

```
!HPF $ TEMPLETE X(N)
!HPF $ DYNAMIC A(N)
!HPF $ ALIGN A(I) WITH X
       ...
!HPF $ REALIGN A(I) WITH X(I+1)
       ...
```

(5)INHERIT 继承指令。这是 HPF 新增加的 FORTRAN90和 FORTRAN D 都没有的指令,它指明了在函数调用时哑元继承了实元的数据划分。这里举一个例子即可说明:

```
        REAL DOUGH(100)
!HPF $ PROCESSORS P(10)
!HPF $ DISTRIBUTE DOUGH (BLOCK(10))
         ONTO P
       CALL PROBATE (DOUGH(7:23:2))
       ...
       SUBROUTINE PROBATE (BREAD)
       REAL BREAD(9)
!HPF $ INHERIT BREAD
       ...
```



2. 数据并行执行

在 HPF 中数据并行执行有三类语句:一种是简便的数据操作语句,如 A=B+C(来自 FORTRAN-90),一种是 FORALL 语句(来自 FORTRAN D),如:

```
FORALL (I=2:4)
  X(I)=X(I-1)+X(I)+X(I+1)
```

它与 X(2:4)=X(1:3)+X(2:4)+X(3:5)是等价的。而对于另一类问题:给出一个下标向量

```
IX[1,2,2,4]
FORALL(I=1:4)
  A(1,IX(I))=X(I)
```

就无法用 FORTRAN90的并行语句编写。这也就是 HPF 引入 FORALL 的好处。

第三类是 INDEPENDENT 语句,为编译器提供有关 DO 循环和 FORALL 语句的信息。例如,它告诉编译器在一个 DO 循环中即使循环不顺序执行也不会引起数据的错误访问,即下面程序段的第一行保证了 INDX 中不会包含重复值:

```
!HPF $ INDEPENDENT
  DO I=1,N
    X(INDX(I))=Y(I)
  END DO
```

有了这个信息,编译器就可以放心地将这段代码并行执行。

3. 扩展的内部函数和库

在 HPF 中有两类库:

(1)在一个 MODULE(模块)中定义的库函数。HPF 把一些库函数放在一个叫做 HPF_LIBRARY 的 MODULE 中。每当用户需要使用这些函数时,就要在相当的程序单元中声明:

```
USE HPF_LIBRARY
```

才能使用这些函数。比如:

IAND,SUM 等。

(2)普通内部函数和标准库。这些就是正常使用的,如

MAXLOC(ARRAY 中最大值的位置)

MINLOC(ARRAY 中最小值的位置)

等。

4. 外部函数 EXTRINSIC PROCEDURE

HPF 提供了与其它语言规范的接口,即把它们做为一种特殊的外部过程在接口块中定义:

EXTRINSIC (C_LOCAL, HPF_LOCAL, HPF(隐含)等)

计算机科学 1996 Vol. 23 №. 2

35-40

Agent 研究*)

朱 冰 TP18
(北京大学计算机科学技术系 北京 100871)

摘 要 This paper introduces agent learning, multi-agent systems analysis, agent negotiation, multi-agent systems programming language etc.

关键词 Agent, Multi-agent systems, Object, Active object.

agent、多 agent 系统已成为分布式人工智能研究的一个十分活跃领域。agent 通常指在一定的环境下持续自主地运行的实体，agent 有两个特点，一是自主性，一是相互合作性。agent 作为独立的个体，能自学习，自增长，同时又可以和别的 agent 进行磋商，合作以完全任务。现在，agent 不仅仅是分析人工智能问题的高层概念，而且有可能成为一个可实现的概念。本文主要参考第13界人工智能国际会议论文集，简单介绍了 agent 学习、多 agent 系统分析、agent 磋商与合作以及多 agent 程序设计等方面的研究现况。最后，初步分析了被动对象、主动对象和 agent。

一、agent 学习

1.1 环境历史活动的学习[1]

agent 的行为与其环境关系很大，它与环境的紧耦合表现在三个方面：开发时比较注重与外部世界的交互作用；为特定的目标和环境设计特殊的 agent；agent 的模型重点在于 agent 的相互作用，而不是对其行为的规划。

agent 通过环境中的历史活动来学习时，涉及到"例程"和"产地"两个概念，前者是指 agent 重复发生的活动，开发例程可以免除规划的烦恼，一般可定义

*)本项目研究得到863国家高技术计划的资助

HPF_LOCAL 说明了单 PROCESSOR、单 NODE 的程序接口。

这样 HPF 就可以与前面多种 FORTRAN 版本、C、PASCAL、ADA 等中高级语言结合起来使用。

三、HPF 中 CHOLESKY 分解的并行程序

```
      PROGRAM CHOL
      PARAMETER N=1024
      REAL A(N,N)
!HPF $ PROCESSOR PROCS(4)
!HPF $ DISTRIBUTE A (CYLIC, * )ONTO
      PROCS
      READ * M
      CALL CHOLSKY(A,N,M)
      END
      SUBROTINE CHOLSKY(A,N,M)
      REAL A(N,N)
      DO K=1,M
        A(K,K)=SQRT(A(K,K))
        FORALL(I=(K+1):M)
          A(I,K)=A(I,K)/A(K,K)
        DO J=K+1,M
          FORALL(I=MAX(K+1,J):M)
          A(I,J)=A(I,J)- A(I,K) * A(J,K)
        END DO
      END DO
      RETURN
      END
```

参 考 文 献

[1] High Performance Fortran Forum (HPFF), High Performance Fortran Language Specification, May 3,1993

[2] Charles H. Koelbel 等, The High Performance Fortran Handbook, The MIT Press,1994

[3] Geoffrey Fox, Seema Hiranandan 等, Fortran D Language Specification, April 15,1991

[4] ISO/IEC JTC1, Fortran 90 Language, Information Technology,1991

[5] K. Dincer 等, High Performance Fortran and Possible Extensions to Support Conjugate Gradient Algorithms, NPAC Technical Report SCCS 703