面向 MPP Fortran 的自动数据分布*

唐新春 郭克榕

(长沙工学院计算机研究所 长沙 410073)

摘要 自动数据分布是面向大规模并行处理MPP(massively parallel processing)系统程序自动并行化的一项关键技术.数据分布方式直接影响着应用程序在MPP系统上的并行执行性能.本文以MPP Fortran为例,详细探讨了自动数据分布的有关技术,如对准分析、分布方式的产生、静态性能评估和数据重新分布等,并提出了相应的算法.这些算法将在作者研制的面向MPP Fortran的程序自动并行化工具中实现.

关键词 自动数据分布,数据局部性,负载平衡,静态性能评估,数据重新分布.

中图法分类号 TP311.11

自动数据分布是面向大规模并行处理MPP(massively parallel processing)系统程序自动并行化的一项关键技术.在程序自动并行化过程中,识别出可并行化循环以后,需对循环中所访问的数据进行分布.一个好的数据分布方案应该是:在保证较好的程序并行性前提下,提高数据局部性,减少通讯开销,使得整个并行程序执行时间最短.如何为分布存储多机系统确定一个有效的数据分布方案,许多研究者都提出了自己的解决方案[1-9],这些解决方案主要因语言环境和目标机器等因素而异.本文以MPP Fortran为例,对自动数据分布技术进行探讨,首先介绍MPP Fortran的主要特点,然后介绍程序单元内数据分布方案的确定和程序单元之间数据的重新分布,最后是国际上同类工作的介绍和小结.本文在数据分布方式的产生和静态性能评估等方面提出了一些新的技术.

1 MPP Fortran 的主要特点

MPP Fortran是近年来推出的面向分布存储、全局编址MPP系统的一种数据并行程序设计语言[10],它提供了数据分布与工作共享的机制.数据对象分为私有数据和共享数据,私有数据不分布,在每个处理机上有一个副本;共享数据在整个系统中只有一个副本,其中共享数组可以按维分布在多个处理机上,每维的分布方式如下:

":*BLOCK*":块分布.在该维的每个处理机上分配一个连续块,块大小为维长/本维处理机数.

":*BLOCK*(*M*)":块循环分布.以*M*个元素为一连续块,在本维的所有处理机上循环分布.

":":本维不分布,或叫退化分布.这种情况下,本维分配的处理机数为1,本维所有的元素都在同一处理机上.

在每维的分布方式中,还可说明一个权值,格式为 w:*BLOCK* 或 w:*BLOCK*(*M*).权值指出该维分配的处理机数所占的比例,设n维数组A每维的权值分别为$w_1,w_2,\ldots,w_n$,系统总处理机数为P,令$W=w_1*w_2*\ldots*w_n$,则权值1对应的处理机数为$p=\lfloor\sqrt[n]{P/W}\rfloor$,数组$A$每维分配的处理机数分别为$p*w_1,p*w_2,\ldots,p*w_n$.本文假定目标机可用的处理机总数$P$已知.

例如,设有数组$X(16,16)$,处理机总数为8,X的分布方式为X(1:*BLOCK*, 2:*BLOCK*),其中1和2分别表示每维的权值,则X第一、二维的处理机数分别为2和4,X的分布方式如图1所示.

*PE*0 (0,0) *X*(1:8,1:4)	*PE*2 (0,1) *X*(1:8,5:8)	*PE*4 (0,2) *X*(1:8,9:12)	*PE*6 (0,3) *X*(1:8,13:16)
*PE*1 (1,0) *X*(9:16,1:4)	*PE*3 (1,1) *X*(9:16,5:8)	*PE*5 (1,2) *X*(9:16,9:12)	*PE*7 (1,3) *X*(9:16,13:16)

图1

在MPP Fortran中,并行循环又称为共享循环,由一条指导命令显式说明:

* 本文研究得到国防科技预研基金资助.作者唐新春,1967年生,讲师,主要研究领域为程序自动并行化.郭克榕,女,1957年生,副教授,主要研究领域为程序自动并行化.

本文通讯联系人:唐新春,长沙 410073,长沙工学院计算机研究所

本文1996-09-09收到原稿,1997-04-09收到修改稿

CDIR $ DOSHARED (I_1,I_2,...,I_n) ON X(f_1(I),f_2(I),...,f_r(I))

该指导命令说明紧接的 n 层紧嵌套循环为共享循环,可并行执行,每个迭代(I_1,I_2,...,I_n)由数组元素 $X(f_1(I),f_2(I),...,f_r(I))$ 所在的处理机执行.其中 $f_1(I),f_2(I),...,f_r(I)$ 为数组下标表达式,必须为 $a*I+b$ 的形式,循环控制变量 I 最多出现在一个数组下标表达式中,a,b 为整型的常量、变量或表达式.数组访问模式 $X(f_1(I),f_2(I),...,f_r(I))$ 称为循环迭代划分的对准目标,下文将其简称为对准目标,而将 X 称为目标数组.MPP Fortran 这种与数组访问模式对准的迭代划分机制,可以使得对准目标以及所有与对准目标对准的其它数组访问全部为局部的,从而提高数据局部性.

2 数据分布

对串行循环进行数据依赖关系分析[11]以后,可以确定程序中的共享循环.在共享循环中定义访问的数组(其下标表达式中含有共享循环的循环控制变量)需定为共享数组,并分布在参与循环执行的所有处理机上,以支持循环的并行执行.共享循环迭代划分的对准目标可选择循环中访问频度最高的共享数组访问模式,以获得较好的数据局部性.

我们分别为每个程序单元确定数据分布方案.本文中程序单元是指 FORTRAN 主程序或外部过程,为程序单元中每个共享数组确定一种分布方式,则形成该程序单元的一个数据分布方案.假定整个程序单元的共享数组已识别出来,共享循环迭代划分的对准目标也已确定.我们的算法分 3 步:(1) 在整个程序单元内作对准分析,确定共享数组之间的对准关系,形成若干对准方案.(2) 对于每个对准方案,产生数组的分布方式,形成若干候选分布方案.(3) 对每个候选分布方案进行静态性能评估,从中选择最佳分布方案.

2.1 对准分析

对准分析主要减少处理机间的数据移动,提高数据局部性.如果共享循环中的数组访问模式与对准目标之间是对准的,则它在所有迭代中都是局部访问.数组访问模式之间的对准是指在整个迭代空间中,这些访问模式所访问的数据对象分布在相同的处理机上,如果某个访问模式是局部访问,则与它对准的访问模式也是局部访问.

Yale 大学的 Li,Chen 和 Choo 提出了 4 种对准关系[1~3]:置换(Permutation),如 $A(I,J)$ 对准 $B(J,I)$,嵌入(Embedding),如 $A(J)$ 对准 $B(I,J,K)$,平移(Shift),如 $A(I,J)$ 对准 $B(I-1,J-2)$,反射(Reflection),如 $A(I)$ 对准 $B(-I)$.当前的对准分析主要基于以上类型.MPP Fortran 中,由于共享数组都是从 0 号处理机上开始分配空间,而且是逐维分配,因此,对准关系不能是置换、嵌入和反射,我们只考虑平移的情况.

对准分析根据共享循环中的数组访问模式确定,两个对准的数组之间维数可以不同,但分布的维数必须相同,如 $DZIN(I)$ 对准 $VC1(I,*)$,其中 * 表示本维不分布.

图 2 为 SPEC92 Benchmark 中 hydro2d.f 的一个程序单元经并行性识别和共享变量识别以后的结果,通过下标分析[2],可以产生 3 个对准方案:

(1) $DZIN(I),DZ(I),VC1(I,*),VZ1(I,*),VR2(I,*),VC2(I,*)$ 之间对准,$DRIN(J),DR(J)$ 之间对准.

(2) $DRIN(J),DR(J),VC2(*,J),VR2(*,J),VZ1(*,J),VC1(*,J)$ 之间对准,$DZIN(I),DZ(I)$ 之间对准.

(3) $VC1(I,J),VZ1(I,J),VR2(I,J),VC2(I,J)$ 之间对准,$DZIN(I),DZ(I)$ 之间对准,$DRIN(J),DR(J)$ 之间对准.

我们将互相对准的所有数组访问模式形成的集合称为一个对准链,如对准方案(1)和(2)具有两条对准链,对准方案(3)具有 3 条对准链.同一条对准链上的所有数组,只要其中任一数组的分布方式确定,则其它所有数组的分布方式也随之确定.

数组之间的对准关系决定了每个共享循环中所有数组访问模式与对准目标之间的对准关系.如图 2 的 $Loop4$ 中,对于对准方案(1),$VC2(I,J),VR2(I,J+1),VR2(I,J-1),VZ1(I,J+1),VZ1(I,J)$ 与对准目标 $VC2(I,J)$ 完全对准,全部为局部访问.$VZ1(I-1,J+1),VZ1(I-1,J)$ 与 $VC2(I,J)$ 部分对准,在分布块的边界上存在非局部访问.$DRIN(J)$ 与 $VC2(I,J)$ 不能对准,大部分为非局部访问.

2.2 产生分布方式

数组的分布方式包括分布的维数、每维分配的处理机数(或权值)和每维分布的块大小.数组分布的维数在对准分析中已经确定,下面我们为每条对准链上任一数组确定每个分布维的权值和块大小.首先考虑多维分布的情况,如前面对准方案(3)的第 1 条对准链.分析对准链上所有数组第 j 分布维的访问模式在程序单元的每个共享循环中与对准目标的对准情况.如果在第 j 分布维上出现一次完全对准,则在第 j 维上增加权值 k_1,如果为部分对准,则增加权值 k_2,最后累计得到该维权值 w_j.例如,$Loop3$ 中 $VC1(I,J),VR2(I,J),VR2(I,J-1)$ 的第 1 维与对准目标 $VC1(I,J)$ 的

第1维完全对准，而 $VZ1(I+1,J)$，$VZ1(I-1,J)$，$VR2(I+1,J)$，$VR2(I+1,J-1)$与对准目标的第1维是部分对准，因此，第1维增加的权值为 $3k_1+4k_2$. 而 *Loop4* 中第1维增加的权值为 $5k_1+2k_2$，所以对准链上任一数组第1维的权值为 $w_1=8k_1+6k_2$. 同理，对准链上任一数组第2维的权值也可计算出来，$w_2=8k_1+6k_2$. 每维的权值确定以后，对应的处理机数也就确定了.

```
      SUBROUTINE ARTDIF()
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER (MP=102,NP=40)
      PARAMETER (M1=MP-1,N1=NP-1)
      COMMON /PRMT/ TS,DT,GAM,CFL,VCT,TZ,TR,FTZ,FTR,BSZ,BSR,QTZ,QTR,IS
      COMMON /ADVC/ ROBMQ,ROBNQ,MQ,MQ1,MQ2,NQ,NQ1,NQ2,MQFLG,NQFLG
      COMMON /GRID/ Z(MP),ZB(0:MP),DZ(MP),DBZ(0:MP),FZ(M1),
     c              R(NP),RB(0:NP),DR(NP),DBR(0:NP),FR(N1),DUM(8)
      COMMON /VARH/ VZ1(0:MP,NP),VR2(MP,0:NP),VC1(0:MP,NP),VC2(MP,0:NP),DU3(288)
      DIMENSION DRIN(NP),DZIN(MP)
CDIR$ SHARED DZIN,DZ,DRIN,DR,VC1,VC2,VZ1,VR2
*Loop1:
CDIR$ DOSHARED (I) ON DZIN(I)
      DO 20 I=1,MQ1
      DZIN(I)=VCT*DT/DZ(I)
20    CONTINUE
*Loop2:
CDIR$ DOSHARED (J) ON DRIN(J)
      DO 40 J=1,NQ1
      DRIN(J)=VCT*DT/DR(J)
40    CONTINUE
*Loop3:
CDIR$ DOSHARED (J,I) ON VC1(I,J)
      DO 230 J=1,NQ
         DO 100 I=1,MQ1
            VC1(I,J)=DZIN(I)*(ABS(VZ1(I+1,J)-VZ1(I-1,J))+
     c                 0.5D0*ABS(VR2(I+1,J)+VR2(I+1,J-1)-VR2(I,J)-VR2(I,J-1)))
100      CONTINUE
230   CONTINUE
*Loop4:
CDIR$ DOSHARED (J,I) ON VC2(I,J)
      DO 220 J=1,NQ1
         DO 200 I=1,MQ
            VC2(I,J)=DRIN(J)*(ABS(VR2(I,J+1)-VR2(I,J-1))+
     c                 0.5D0*ABS(VZ1(I,J+1)+VZ1(I-1,J+1)-VZ1(I,J)-VZ1(I-1,J)))
200      CONTINUE
220   CONTINUE
      END
```

图 2

每维的分布块大小也根据对准情况确定. 如果对准链上所有数组第 j 分布维上的访问模式在程序单元的每个共享循环中与对准目标为完全对准或不能对准，则该维引起的通讯开销一般不随分布块大小变化，而采用数据循环分布 *BLOCK*(1)，可获得最佳的处理机负载平衡，如对准方案(3)的对准链 $\{DZIN(I),DZ(I)\}$ 和 $\{DRIN(J),DR(J)\}$. 如果存在部分对准的情况，设该分布维的数组访问模式与对准访问时的最大偏差为 d，我们从数据局部性角度出发(参见 2.3)，取分布块大小 M_j 为 d 的 k_3 倍，即 $M_j=K_3*d$. 如对准方案(3)的第1条对准链，两维最大偏差均为1，因此取 $M_1=M_2=k_3$. 除此之外，我们考虑到处理机负载平衡性，还取 *BLOCK* 分布作为一种候选分布方式.

以上讨论的是对准链上的数组多维分布的情况. 对于对准链上的数组只有一维分布的情况，如对准方案(1)、(2)和对准方案(3)的第 2,3 条对准链，则不需确定权值，而分布块大小的确定方法与上相同. 对准方案(1)中，第 1 条对准链上的数组分布方式有 $BLOCK(k_2)$和 $BLOCK$，第 2 条对准链上的数组分布方式为 $BLOCK(1)$. 对准方案(2)产生的分布方式与方案(1)相同.

对准方案中每条对准链上任一数组的分布方式产生以后，将它们组合为候选分布方案，如对准方案(3)产生两个候选分布方案：

(1) $VC1(w_1:ELOCK(k_3),w_2:BLOCK(k_3)),DZIN(:BLOCK(1)),DRIN(:BLOCK(1))$

(2) $VC1(w_1:BLOCK,w_2:BLOCK),DZIN(:BLOCK(1)),DRIN(:BLOCK(1))$

其中每条对准链我们只列出了一个数组. 本例中"$w_1=w_2$"可省略.

在以上的分析中，k_1,k_2,k_3 为并行化工具给定的参数，可以根据应用程序在目标机上的实际运行结果凭经验选择一组或多组数值.

2.3 分布方案评估

当程序单元产生的候选分布方案超过 1 个时，需要对它们进行静态性能评估，从中选出最佳分布方案. 静态性能评估不一定要求十分精确，它主要用于区分各个候选分布方案的优劣.[9]分布方案对共享循环的执行时间具有较大的影响，我们以共享循环为单位进行静态性能评估. 根据共享循环的计算量、通讯量、处理机的计算能力、互联网的通讯能力和循环并行执行时的处理机数，静态估算循环的并行执行时间. 对每个候选分布方案计算程序单元内所有共享循环的总执行时间，则总执行时间最短的方案为最佳数据分布方案.

设目标机系统的单机处理能力为 ω(MIPS)，任一处理机一次非局部访问的时间(包括启动时间和传输时间)为 τ(秒)，循环体的计算量为 Nc(Instructions)，循环迭代空间大小(总的迭代次数)为 N_i，并行执行循环的处理机数为 N_p，每个处理机的非局部访问次数为 N_r，则循环的执行时间为：$T=\frac{N_i * N_c}{\omega * N_p}+N_r * \tau$(秒). 其中 ω 和 τ 是由系统所决定的；N_i 是由程序本身所固有的，N_c，N_r 和 N_p 则与数据的分布方式密切相关.

N_c 根据循环体编译后产生的指令数确定. 其中数组地址计算所需的指令数与分布方式关系较大，如 $VC1(I,J)$ 的地址计算，如果 $VC1$ 两维都是 $BLOCK(M)$分布，约需 30 条指令，若一维 $BLOCK$ 分布，一维不分布，约需 21 条指令. 其它操作和运算所需的指令数一般与分布方式无关. 我们可以将每种操作和运算所需的指令数保存在一个内部表中，以便性能评估时查找.

N_r 根据循环中数组访问模式与对准目标之间的对准关系确定. 完全对准的访问模式不会引起非局部访问. 不能对准的访问模式我们保守地估计为 100%的非局部访问，次数为 N_i/N_p. 部分对准的访问模式在分布块的边界上引起非局部访问，这种访问模式的对准比例为：$\rho=\frac{(M_1-d_1) * \ldots * (M_n-d_n)}{M_1 * \ldots * M_n}$. 其中 $d_1,\ldots,d_n$ 为该访问模式与对准访问模式比较时每维的偏差，而 $M_1,\ldots,M_n$ 为数组每维分布的块大小，如图 3 所示. 所以部分对准访问模式引起的非局部访问次数为 $(1-\rho) * N_i/N_p$.

N_p 为共享循环中实际参与并行执行的处理机数. 设对准目标第 j 维的下标表达式为 $a_j * I_j-b_j$，I_j 的初值为 l_j，终值为 u_j，目标数组该维分配的处理机数为 N_j，分布块大小为 M_j，则下标表达式为 $a_j * I_j+b_j$. 跨越的处理机数为 $P_j=\lceil a_j(u_j-l_j)/M_j \rceil$，若 $P_j \geqslant N_j$，则该维所有处理机都参与执行，否则，只有 P_j 个处理机执行. 取 $N_{pj}=\min(P_j,N_j)$，则共享循环中实际参与并行执行的处理机数为 $N_p=\prod_{j=1}^{n} N_{pj}$，$n$ 为目标数组分布的维数.

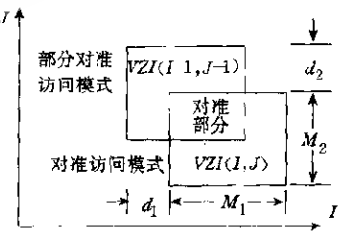

图3

当循环的初值和终值为变量时，我们首先通过过程间常数传播和向前替代等方法确定变量的取值，如果仍然不能确定，则根据循环中数组的定义来确定循环控制变量的范围. 如图 2 的 $Loop1$ 中，$MQ1$ 根据 $DZIN$ 的定义可取为 102.

当共享循环中含有外部过程调用时，只能调用串行版[12]，我们将外部过程串行版估算的计算量和共享哑数组、共享公用块数组私有化(类似重新分布，见下节)的开销传给本程序单元一起评估.

当共享循环中含有条件分支时，可根据每个分支发生的概率分别对每个分支评估.[13]

3 重新分布

在上节中，我们确定了每个程序单元所要求的最佳数据分布方案，本节将讨论程序单元之间的数据重新分布.

3.1 共享实参数组重新分布

如图4(a)所示，程序中含有外部过程调用.经过第2节的分析，*SUB*1要求实参*A*具有分布方式1，*SUB*2要求哑参*D*具有分布方式2，哑参和实参要求的分布方式不一致.对于这种情况，编译器自动进行隐式重新分布，即在子程序入口将实参重新分布为哑参的分布方式，而在子程序出口再将实参重新分布为原来的分布方式，如图4(b)所示.但重新分布具有一定的动态开销，如果被调用程序单元的计算量不是很大，重新分布带来的好处不足以抵消它引入的额外开销，则没有必要进行重新分布，如图4(c)所示.

3.2 共享公用块数组重新分布

如图5(a)所示，多个程序单元中的同一公用块数组在各自的程序单元中要求不同的分布方式，这是语言文本所不能支持的.[10]对于这种情况，我们先将共享公用块数组易名，转换为参数传递的形式，如图5(b)所示，再按3.1节的方法重新分布.当程序单元中的计算量不是很大时，也可不进行重新分布，如图5(c)所示.

```
SUBROUTINE SUB1()
A 要求分布方式 1
   …
CALL SUB2(A)
   …
END
SUBROUTINE SUB2(D)
D 要求分布方式 2
   …
END
```

(a) 哑参和实参要求不同的分布方式

```
SUBROUTINE SUB1()
A 保持分布方式 1
   …
CALL SUB2(A)
   …
END
SUBROUTINE SUB2(D)
D 保持分布方式 2
(A 隐式重新分布为方式 2)
   …
(A 恢复原来的分布方式 1)
END
```

(b) 实参重新分布

```
SUBROUTINE SUB1()
A 保持分布方式 1
   …
CALL SUB2(A)
   …
END
SUBROUTINE SUB2(D)
D 继承分布方式 1
(A 不重新分布)
   …
END
```

(c) 实参不重新分布

图 4

```
SUBROUTINE SUB1()
COMMON /X/A
A 要求分布方式 1
   …
CALL SUB2
CALL SUB3
   …
END
```

```
SUBROUTINE SUB2()
COMMON /X/B
B 要求分布方式 2
   …
END
```

```
SUBROUTINE SUB3()
COMMON /X/C
C 要求分布方式 3
   …
END
```

(a) 公用块数组要求不同的分布方式

```
SUBROUTINE SUB1(newA)
COMMON /X/A
定义 newA
newA 要求分布方式 1
   … } A 易名为 newA
CALL SUB2(newA)
CALL SUB3(newA)
   … } A 易名为 newA
END
```

```
SUBROUTINE SUB2(newB)
COMMON /X/B
定义 newB
newB 要求分布方式 2
   … } B 易名为 newB
END
```

```
SUBROUTINE SUB3(newC)
COMMON /X/C
定义 newC
newC 要求分布方式 3
   … } C 易名为 newC
END
```

(b) 公用块数组转换为参数传递

```
SUBROUTINE SUB1()
COMMON /X/A
A 保持分布方式 1
   …
END
```

```
SUBROUTINE SUB2()
COMMON /X/B
B 改为分布方式 1
   …
END
```

```
SUBROUTINE SUB3()
COMMON /X/C
C 改为分布方式 1
   …
END
```

(c) 公用块数组不重新分布

图 5

程序单元之间数据重新分布与不重新分布各有利弊，重新分布时会引入额外的系统开销，不重新分布时所取的分布方式将与程序单元评出的最佳分布方式不一致，我们采用上节介绍的静态性能评估方法分别对它们进行评估，从而决定是否进行重新分布.

4 相关工作

当前的自动数据分布技术一般都包括对准分析和分布分析，但各自采用的分析方法有所不同.

Yale 大学的 Li，Chen 和 Choo 为 Crystal 编译器提出的自动数据分布算法[1~3]，将一个过程分为多个程序段，为每个程序段执行对准分析和分布分析，形成一个数据分布方案，最后，将不同程序段的数据分布方案合并. 他们的对准分析分为维间对准(置换、嵌入)和维内对准(平移、反射). 对准分析后在程序段中插入通讯例程，每个通讯例程有一个时间开销函数，以分布方式、问题规模和机器特征作为参数，由此确定程序段中开销最小的分布方式.

Illinois 大学的 Gupta 和 Banerjee 为 Parafrase-2 并行化工具提出的自动数据分布算法[4,5]，为每条语句作对准分析和分布分析. 对准分析基于 Li，Chen 和 Choo 的方法. 每条语句有一个时间开销函数，以问题规模、目标机处理机数和分布方式作为参数，当问题规模和目标机处理机数编译已知时，整个用户程序产生一个唯一的数据分布方案，程序单元或程序段之间没有重新对准或重新分布.

Carnegie Mellon 大学的 Wholey 为 ALEXI 语言开发的编译器中，在编译时执行对准分析，在运行时执行分布分析.[6]对准分析主要考虑维内对准，分布方式仅考虑 *BLOCK* 分布. 该方法以原始操作(Primitive Operation，如并行计算结构或内部通讯函数)为时间开销的评估单位，时间开销函数以分布方式、问题规模、目标机处理机数和目标机的拓扑结构作为参数，整个程序的时间开销为所有原始操作的时间开销之和. 该方法没有重新对准或重新分布，但进行过程间性能分析.

Vienna 大学的 Chapman，Fahringer，Blasko，Herbeck 和 Zima 提出的自动数据分布算法为 SUPERB-2 并行化工具的一部分.[7,8]他们的方法主要基于 Gupta 和 Banerjee 的工作. 另外，他们的静态性能评估还采用了知识库和高层模式匹配的方法. 数据分布方案主要根据关键程序段确定. 该工具执行过程间性能分析和过程间重新分布.

Rice 大学的 Kremer 提出的面向 Fortran D 的自动数据分布算法为整个用户程序确定最佳数据分布方案.[9]它首先对整个程序作全局对准分析，形成若干对准方案. 然后将程序分为若干程序段，为每个程序段产生若干局部分布方案，采用培训集(Training Set)和模式匹配的方法对每个局部分布方案进行评估. 最后合并所有程序段的局部分布方案，程序段之间考虑重新对准和重新分布. 这种方法需要一个有效的修剪算法来减小对准方案和局部分布方案的搜索空间. 另外，当程序中含有外部过程调用时，程序段之间的分布方案合并将十分复杂，不仅要作过程间分析，而且与编译器是否支持过程繁衍和过程嵌入有关.

我们提出的自动数据分布算法以程序单元为单位进行对准分析和分布分析，在程序单元之间考虑重新分布，并执行过程间的性能分析. 我们的算法最大的特点是：根据共享循环中所有数组访问模式与对准目标之间的对准关系(完全对准、部分对准)产生每维的权值和分布块大小，尽量缩小候选分布方案的搜索空间. 另外，我们根据 MPP Fortran 数据并行语言的特点，建立了自己的静态性能评估模型，以提高评估效率.

5 结束语

自动数据分布一直是面向大规模并行处理系统的程序自动并行化有待攻克的难题. 尽管许多研究者提出了各种解决方案，但这方面的技术仍欠成熟，大部分算法都未实现或者仅部分实现于原型系统中，还有待经受实际应用程序的考验. 本文提出的自动数据分布算法将用于我们正在研制的面向 MPP Fortran 的程序自动并行化工具中. 下一步，我们将对算法进一步完善，如在程序单元内实现数据重新分布、使过程间的性能分析更精确等，并且从工程的角度出发，使算法更加实用、有效. 自动数据分布技术的深入研究，对大规模并行处理系统的推广和应用有着重要的意义.

参考文献

1 Chen M, Choo Y, Li J. Theory and pragmatics of compiling efficient parallel code. Technical Report of YALEU/DCS/TR-760, Yale University, New Haven, CT, December 1989

2 Li J, Chen M. Index domain alignment: minimizing cost of cross-referencing between distributed arrays. In: Frontiers90: The 3rd Symposium on the Frontiers of Massively Parallel Computation. College Park, MD, October 1990

3 Li J, Chen M. Compiling communication-efficient programs for massively parallel machines. IEEE Transactions on Parallel and Distributed Systems, July 1991, 2(3): 361~376

4 Gupta M, Banerjee P. Automatic data partitioning on distributed memorymulti processors. In: Proceedings of the 6th Distributed Memory Computing Conference. Portland, OR, April 1991

5 Gupta M, Banerjee P. Demonstration of automatic data partitioning techniques for parallelizing compilers on multicomputers. IEEE Transactions on Parallel and Distributed Systems, April 1992

6 Wholey S. Automatic data mapping for distributed-memory parallel computers. In: Proceedings of the ACM International Conference'92 on Supercomputing. Washington, DC, July 1992

7 Chapman B, Herbeck H, Zima H. Automatic support for data distribution. In: Proceedings of the 6th Distributed Memory Computing Conference. Portland, OR, April 1991

8 Fahringer T, Blasko R, Zima H P. Automatic performance prediction to support parallelization of Fortran programs for massively parallel systems. In: Proceedings of the ACM International Conference'92 on Supercomputing. Washington, DC, July 1992

9 Kremer U. Automatic data layout for distributed-memory machines. Technical Report of CRPC-TR93299, Houston, Center for Research on Parallel Computation, Rice University, 1993

10 Douglas M Pase, Tom Mcdonald, Andrew Meltzer. MPP Fortran programming model. Eagan, Minnesota: Cray Research, Inc., 1993

11 Utpal Banerjee. Dependence analysis for supercomputing. Boston: Kluwer Academic Publishers, 1988

12 Guo Ke-rong, Tang Xin-chun. Multi-version technique for external procedure calls in MPP Fortran programs. Computer Engineering and Design, 1996, 17(6): 49~54

13 Ken Kennedy, Nathaniel McIntosh, Kathryn S McKinley. Static performance estimation in a parallelizing compiler. Technical Report of CRPC-TR9292204, Houston: Center for Research on Parallel Computation, Rice University, 1992.

Automatic Data Distribution for MPP Fortran

TANG Xin-chun GUO Ke-rong

(Institute of Computers Changsha Institute of Technology Changsha 410073)

Abstract Automatic data distribution is a key technique in the area of automatic program parallelization for MPP (massively parallel processing) systems. Data distribution schemes directly influence the parallel execution performance of application programs in MPP systems. Taking MPP Fortran as an example, this paper discusses techniques of automatic data distribution in detail such as alignment analysis, shared array distribution, performance estimation for shared array distribution schemes, and shared array redistribution. The corresponding algorithms are presented and they will be implemented in these automatic program parallelization tools for MPP Fortran.

Key words Automatic data distribution, data locality, load balance, static performance estimation, data redistribution.

Class number TP311.11