数据线性划分模式下改进的循环分割算法*)

Improved Loop Partition Algorithm in Linear Data Distribution

胡南军　谭文芳　陈道蓄　谢　立

(计算机软件新技术国家重点实验室　南京大学计算机科学与技术系　南京210093)

Abstract　As the distributed memory environment is widely used as the sustaining platform for the parallel computing, how to make a rational data distribution and generate SPMD program efficiently in this environment has become one of the key issues in the researches of parallel compiler. In this article we analyze the shortcomings of a loop distribution algorithm in data linear distribution. And we propose an improved loop partition algorithm which enhances the problem solving ability of the original data distribution algorithm.

Keywords　Distributed memory, Linear data distribution, Loop partition, Algorithm

1. 概述

与共享存储结构相比，分布式存储结构在性价比和可扩展性上有明显的优势，但它缺乏全局的存储空间，存储器间只能通过消息传递的方式进行通讯，因此程序员在编写程序时需要考虑如何分布计算和相应的数据，以便尽可能提高分布式程序的并行性同时尽可能减少处理器间的通讯量。这一工作加大了程序员的负担和算法的复杂度，解决这个问题的一个方案就是借助源程序到源程序的并行编译软件的帮助，对串行程序进行并行化分析，自动地对计算和数据进行分布。

由于循环是占用程序执行时间最长的程序结构，因此比较自然的想法就是将一个规模较大的循环分割成若干较小规模的循环，并将这些循环分布到不同的处理器上执行，以获得加速比。而在大规模的科学计算中循环总是与大规模的数据处理相对应，因此数据划分的结果就成为循环分割的依据。

本文主要分析了Guo[2]中提出的数据线性划分算法中循环分割算法的不足，并给出了改进的循环分割算法。在改进的算法中我们消除了原算法对数组元素引用模式潜在的限制，增强了原数据线性划分算法解决问题的能力。

2. 研究背景

2.1　基本概念

1)循环的迭代空间和数组的数据空间　我们可以将数组中元素的组织形式看作一种空间形式，其中数组的维数定义了数组空间的维数，而数组各维的边界定义了数组空间中各维的边界。同样可以将循环的所有迭代也看作是一种空间形式，其中循环的嵌套重数定义了循环空间的维数，各重循环的上下界定义了循环空间各维的边界。当数组变量的下标表达式为循环控制变量的线性函数时，我们可以定义循环空间到数组空间的映射函数：$f(\vec{I})=A\vec{I}+\vec{I}_0$。其中，A为映射矩阵，$\vec{I}_0$为偏移向量[4]，$\vec{I}$为循环空间向量。如A(2i+j+3,3i−2j−2)，其迭代空间到数组空间的映射函数为：

$$A=\begin{pmatrix}2 & 1\\ 3 & -2\end{pmatrix}\vec{I}_0=\begin{pmatrix}3\\ -2\end{pmatrix}$$

线性映射函数：

$$f(\vec{I})=\begin{pmatrix}2 & 1\\ 3 & -2\end{pmatrix}\vec{I}+\begin{pmatrix}3\\ -2\end{pmatrix}$$

2)超平面　一个n维超平面可定义为一个多元组的集合：

$$\{(a_1,a_2,\cdots,a_n)\mid a_1g_1+a_2g_2+\cdots+a_ng_n=c\}$$

$G=(g_1,g_2,\cdots,g_n)$称为超平面向量。实际上一个超平面向量就定义了一个n维空间的超平面族，不同的c对应不同的超平面。在二维空间中超平面族描述的是直线系。在数组空间中两个点$\vec{I}_1$和$\vec{I}_2$属于同一个超平面，当且仅当$G\cdot\vec{I}_1=G\cdot\vec{I}_2$。这也是两个数组元素属于同一个划分的充要条件。因此，给定一个超平面向量，也就确定了数组空间中的一种划分模式，表1给出了二维数组中的几种典型划分模式。

表1

(1,0)	(0,1)	(1,−1)	(1,1)
按行划分	按列划分	按对角线划分	按反对角线划分

2.2　数据线性分布算法

Guo在文[2]中提出了一种基于超平面的数据线性分布算法。该算法输入待分析的串行循环结构，通过对循环体中数组下标表达式的分析，进行可线性化划

*)本文所涉及的项目得到"863"高技术计划的支持(项目号863-306-ZT02-03-01)。

分的判定。若可划分，则求出以超平面表示的线性划分模式[2]，并根据该模式对数组进行分割生成若干较小规模的数组，然后根据数组划分的结果对原串行循环进行分割生成若干较小规模可并行执行的SPMD程序，通过底层任务调度将这些程序连同相关数据分布到不同处理器上并行执行。以下，我们着重对该数据分布算法中的循环分割算法进行描述并分析其不足，在下一节中，我们将给出改进的循环分割算法。

2.2.1 程序模型 算法假定待分割的循环已标准化，即循环变量 i_k、j_k 均满足 $1\leqslant i_k\leqslant n_1$ 及 $1\leqslant j_k\leqslant n_2$，其中 n_1、n_2 为正整数，循环体中的数组为2维。

```
do i=1 to n1
   do j=1 to n2
   f(A(i11,j11),A(i12,j12),…,A(im,jm),B(i21,j1),B(i22,
   j22),…,B(in,jn),…)=0
```

其中 $i_{mn}=g_{mn}(i,j)$，$j_{mn}=h_{mn}(i,j)$；g_{mn}、h_{mn} 均为2维空间中的线性函数。

2.2.2 线性划分 在文[2]中给出了线性划分的形式化定义。

定义2.1（线性划分） 以超平面向量 $(g_1,g_2,\cdots,g_n)$ 形式给定 n 维数组 A 上的一种划分模式，$\exists\ m, g_m\neq 0$，将数组 A 划分为 n 个数据块 $D_1,D_2,\cdots,D_n$，称为对数组 A 的线性划分，其中

$D_k=\{(a_1,a_2,\cdots,a_n)\mid a_1g_1+a_2g_2+\cdots+a_ng_n=c_k,\ low(k)\leqslant a_k\leqslant up(k)\}$；Low(k)和up(k)分别为数组A第k维的上下界。

每个数据块 D_k 对应一个超平面，且若 $\exists\ a_1,a_2\in A$，a_1,a_2 具有依赖关系[6]，则 $\exists\ d, a_1\in D_d$，且 $a_2\in D_d$。

2.2.3 数组的分割 设数组A上的线性划分模式为 $Mod,d_1,d_1,\cdots,d_m$ 表示数组A在模式Mod下划分得到的超平面，分布式存储环境中有n个处理器，且 $n<m$，则该算法分配到第k个处理器上的数据集为 $S_k=\{d_p\mid p\bmod n=k,1\leqslant p\leqslant m\}$，其数据形式为将这些超平面顺序排列组成得到的一维数组。在此，算法为每一个新数组引入辅助数组Bound，元素 $Bound_k(i)$ 为分配在该处理器上的第i个超平面在 S_k 中的起始位置（数组分割示例见图1）。

```
P0: A'={A(1,1),A(1,4),A(2,3),A(3,2),A(4,1),A(2,
        6),A(3,5),A(4,4),A(5,3),A(6,2),A(5,6),A
        (6,5)
    Bound0={1,2,6,10};
P1: A'={A(1,2),A(2,1),A(1,5),A(2,4),A(3,3),A(4,
        2),A(5,1),A(3,6),A(4,5),A(5,4),A(6,3),A
        (6,6);
    Bound1={1,3,8,12};
P2: A'={A(1,3),A(2,2),A(3,3),A(1,6),A(2,5),A(3,
        4),A(4,3),A(5,2),A(6,1),A(4,6),A(5,5),A
        (6,4)}
    Bound2={1,4,10};
```

说明：P0、P1、P2为处理器，A为原数组，A'为划分后的新数组，划分模式为(1,1)

图1

2.2.4 循环的分割 在数组进行分割后，该循环分割算法根据数组分割的结果对循环进行相应的分割，分割得到分配到第k个处理器上的SPMD程序形式如图2

```
do j=1,J
   do i=t1,t2,t3
      F(A'(i),A'(i-e))=0
                        (a)
do j=1,J
   do i=t1,t2,t3
      F(A'(i),B'(i))=0,
                        (b)
```

(a)同名数组变量处理方式，(b)异名数组变量处理方式；A'、B'为数组A、B变换后得到的新数组

图 2

该算法的主要思想为：以分配到第k个处理器上的超平面个数为新循环外重循环的上界，选择参照数组变量其变换后形式为A'(i)，内循环的上、下界及步长分别为Bound[j]，Bound[j+1]及参照数组变量的变化步长，且该算法隐含要求 $|t_3|=1$，这个条件实际上要求参照数组变量在数组引用时具有连续性。与参照数组同数组名的数组变量变换后的形式为A'(i+e)，e为在原循环的一次迭代中，该数组变量与参照数组变量在同一个数组超平面内引用点间的距离值，且必须为常数。在变换与参照数组异名的数组变量时，算法要求该数组变量与参照数组变量进行对齐(alignment)，则其下标表达式与参照数组相同。

2.2.5 该循环分割算法的不足

1）无法处理同名数组变量间距离不为常数的情况，见图3。图3(a)对应的循环体为F(A(i,j),A(i,2j))=0，由图可见可将A的数组空间列，即(1,0)进行无通讯的划分，由两个同名数组变量的下标表达式可知两个数组变量在数组空间中的同一个超平面内的伸展方向一致，但步长不一致。

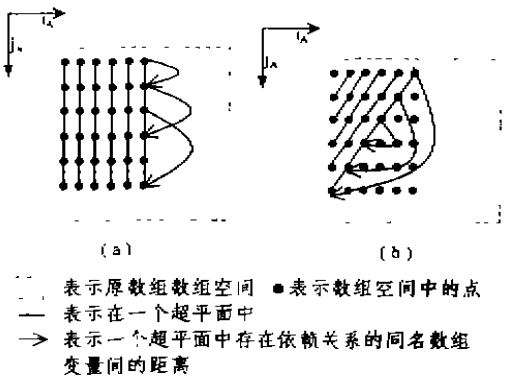

图3

图3(b)对应的循环体为F(A(i,j),A(j,i))=0，由图可见可将A的数组空间按对角线，即(1,1)进行无通讯的划分，由两个同名数组变量的下标表达式可知两个数组变量在数组空间中的同一个超平面内的伸展

方向不一致，但步长一致。

由于原算法仅能处理同名数组变量在一个超平面内距离为常数的情况，所以虽然图3(a)和图3(b)两种情况可以进行无通讯的数组划分，但由于原循环分割算法的处理能力的限制，而无法得到可并行的 SPMD 程序。

ii)由图2(b)可见在异名数组的变换中，原算法要求一个数组的变换要参照另一个已变换的参照数组进行一一对应，这样做增加了描述数组的变换信息，使得数组的分割和重组不能高效进行。

iii)要求参照数组具有数组元素引用的连续性。

3. 改进的循环分割算法

在数据分布中，循环作为处理大规模数组的程序结构，其划分必然是以数组划分的结果为依据的，实际上数组的划分和循环的分割都是按照线性划分模式计算的结果进行的。

循环分割的要求如下：

i)由于在线性划分算法中要求对数组的划分是无通讯的(即数据及程序分布后，在并行计算中，处理器间无需进行通讯)，因此要求相应的循环分割得到的 SPMD 程序间在并行计算中也不能产生通讯。

ii)为了充分利用并行环境中处理器的并行效能，必须考虑如何对数据进行均匀的划分及相应的循环分割，以达到负载的平衡，故在改进算法中仍使用文[2]中超平面循环分配的方式，并以分布式存储环境中的处理器个数作为循环因子。

iii)在确定划分粒度时应考虑并行计算中的网络传输代价问题，若划分粒度过细，并行计算的效益无法超过数据传输的代价，则无法得到预期的加速比。在算法中我们依据处理器的个数确定划分的 SPMD 程序的个数。

3.1 线性划分下循环分割的形式化定义

定义3.1(函数 DPar(A,Mod)) A 表示数组的数据集，Mod 为数组 A 的线性划分模式，函数结果为数组 A 在该划分模式下得到待分配到 n 个处理器上的子数据集集合，即 DPar(A,Mod)={$A_1,A_2,\cdots,A_n$}，A_k 表示分布到第 k 个处理器上的子数据集，$A_1\cup A_2\cdots\cup A_n=A$ 且 $A_1\cap A_2\cdots\cap A_n=\Phi$。

定义3.2(函数 DCon({$A'_1,A'_2,\cdots,A'_n$},Mod)) {$A'_1,A'_2,\cdots,A'_n$}为将 DPar(A,Mod)得到的数据集集合中的元素分配到 n 个处理器上并行计算得到的新数据集集合，若 A'为上述串行循环执行后数组 A 的值，则函数 DCon({$A'_1,A'_2,\cdots,A'_n$})=A'。

定义3.3(线性划分下的循环分割) 设串行循环 L 的输入数据集为 M={$A_m,B_m\cdots$}，循环结构为：

```
L:: do i=1 to n1
      do j=1 to n2
      f(K)=0;
```

循环执行结束后的输出数据集为 N={$A_{out},B_{out}\cdots$}。

对如上的循环 L，DPar(A_m)={$A_{m1},A_{m2},\cdots,A_{mn}$}，DPar($B_m$)={$B_{m1},B_{m2},\cdots,B_{mn}$}…n 为系统中处理器的个数。分配到第 k 个处理器上的输入数据集 M_k={$A_{mk},B_{mk}\cdots$}。

对循环按照数据划分的结果进行分割，第 k 个处理器上的循环结构为：

```
do i=1 to m
  do j=1 to n
    f(Mk)=0;
```

设各个处理器上执行得到的输出数据集为{$N'_{out1},N'_{out2}\cdots N'_{outn}$}，其中 N'_{outk}={$A_{outk},B_{outk}\cdots$}，并且：DCon({$A_{out1},A_{out2},\cdots,A_{outn}$})=$A'_{out}$ DCon({$B_{out1},B_{out2},\cdots,B_{outn}$})=$B'_{out}\cdots$，则并行后得到的输出数据集 N'={$A'_{out},B'_{out}\cdots$}且 N'=N。

3.2 改进的循环分割算法

1)求单个数组变量在线性划分下每个超平面中变化的步长 设数组变量 $A_l(i_l,j_l)$，$u=(i_l,j_l)$表示 A_l在数组空间 A 中的点，α 为 A_l的映射矩阵，α_0为 A_l的偏移向量，$G=(g_1,g_2)$为 A 上的线性划分，step 为所求之步长：

$$\alpha=\begin{pmatrix}\alpha_{11}&\alpha_{12}\\ \alpha_{21}&\alpha_{22}\end{pmatrix}\quad \alpha_0=\begin{pmatrix}\alpha_{10}\\ \alpha_{20}\end{pmatrix}$$

联立方程组：

$$\begin{cases}u^T=\alpha I+\alpha_0 & (1)\\ G\cdot u^T=c & (2)\end{cases}$$

其中$|\alpha|\neq 0$

由(1)(2)$\Rightarrow(G\alpha)\cdot I=c-G\cdot\alpha_0$ (3)

$\Rightarrow(a',b')\cdot I=c'$ (4)

由于循环空间具有连续性

$\Rightarrow(d_1,d_2)=(b'/\gcd(a',b'),a'/\gcd(a',b'))^T$ (5)

(d_1,d_2)为循环空间中点的变化向量，gcd 为求最大公约数

将循环空间中点的变化向量向数组空间进行映射，得到数组空间中点在超平面内的变化向量$(d_{l1},d_{l2})=\alpha\cdot(d_1,d_2)^T$，则 step=gcd($d_{l1},d_{l2}$)为所求之变化步长。

2)求数组变量在线性划分下在每个超平面中引用的下限点 low 和上限点 high 及其在该超平面中的变化次数 p 设$G\cdot u^T=c_1$表示线性划分 G 下得到的数组空间中的一个超平面：

$$\begin{cases}u^T=\alpha I+\alpha_0 & (1)\\ G\cdot u^T=c_1 & (2)\end{cases}$$

其中$|\alpha|\neq 0$

由(1)(2)$\Rightarrow(a',b')\cdot I=c'_1$ (3)

循环空间的四个边界分别对应下列四个方程：

$$i=0 \qquad (1,0)I=0 \tag{4}$$
$$j=u_1 \qquad (1,0)I=u_1 \tag{5}$$
$$j=0 \qquad (0,1)I=0 \tag{6}$$
$$j=u_2 \qquad (0,1)I=u_2 \tag{7}$$

将(3)与(4)(5)(6)(7)分别联立：

i)若无解则原程序在 $G\cdot u^T=c_1$ 超平面上没有引用，所求变化次数 $p=-1$；

ii)若解得一个值 (i_1,j_1)，则原程序在 $G\cdot u^T=c_1$ 超平面上仅引用一个点，带入(1)求得该点坐标，所求变化次数 $p=0$。

iii)若解得两个值 (i_1,j_1)，(i_2,j_2)，则根据它们在循环空间中的相对次序判断哪一个对应所求的下限值并将这两个值带入(1)求得 low 和 high，且所求变化次数 p=(high-low)/step。

因为在每个超平面内，所有的数组变量同步变化，所以所求的变化次数也为所有数组变量在该超平面内的变化次数。

3)改进的循环分割算法

输入：二重循环语句　数组线性划分模式

输出：SPMD 程序

1)如果数组可进行线性划分则执行2)

2)分别计算每个数组变量在超平面内的步长值，记为 $stepA_1$，$stepA_2$，…，$stepA_n$，$stepB_1$，$stepB_2$，…，$stepB_n$…

3)对待分割的循环分割成多个循环，从而得到 SPMD 程序。分割后分配到第 k 个处理器的循环具有以下形式：

```
do j=1 to l
   I. 计算数组变量在分配到该处理器的第 j 个超平面内的
      变化次数，记为 u_j，并计算各个数组变量在第 j 个超平
      面内的初始位置。
   II. 将 I 中求得的初始位置映射为在新数组中的位置，记
      为
        δ_j1，δ_j2，…δ_jn，δ_j2…δ_jn…
   do i=1 to u_j+1
      f(A(δ_j1+(i-1)*stepA_1)，A(δ_j2+(i-1)*stepA_2)…A(δ_jn+(i-1)
          *stepA_n)，
        B(δ_j1+(i-1)*stepB_1)，B(δ_j2+(i-1)*stepB_2)，…，B(δ_jn
          +(i-1)*stepB_n)…)=0
```

在循环中 l 为从原数组中划分并分配到第 k 个处理器的超平面个数。

3.3　对原算法的改进

由循环分割算法的叙述可知，因为在改进算法中各个数组变量的变换仅依赖数据线性划分的结果，而不依赖于其它数组变量的划分，所以对于同名数组变量和异名数组变量的变换可以采用统一的形式进行。同时由于消除了参照数组的影响，因而去除了要求数组变量访问的连续性条件并增加了对同名数组变量间在划分的超平面内距离不为常数的处理，增强了循环分割算法解决问题的能力。

4. 相关工作及其比较

作为数据分布算法的一部分，循环的分割与数据划分模式的计算有着密切的关系，实际上循环分割是以数据划分模式的计算结果为依据的。Gupta 和 Banerjee[1]提出的划分算法以行、列和块的方式对数组进行划分，这种规则的数据划分方式对于循环的分割是容易处理的。但在很多情况下，按这些模式进行划分并不是最佳的，有时甚至是不可能的。Wolf[5]提出了一些循环变换的方法，这些方法侧重于循环形式的变换。通过这些循环变换我们可以将一些形式上数据分布算法无法分析的循环转换成可以处理的循环形式。Utpal Banejee[6]也讨论了如何从串行程序的循环得到 SPMD 程序，但其循环分割是以数据依赖分析的结果进行的，其适用范围受到数据依赖分析技术本身的限制。Minyi Guo[2]提出了一种数据的线性划分算法，该算法增加了按斜线划分的模式。但该算法的循环分割部分存在不足，使得一些满足数据划分条件的循环，由于循环分割算法的限制而不能进行分布。本文主要针对该算法的这一不足进行改进，使得原分布算法解决的问题范围得到扩充。

总结　本文主要讨论了如何对一种称为数据线性分布算法中的循环分割算法进行改进，从而使得原数据分布算法解决问题的范围得到扩充。在改进的算法中，循环分割中数组变量的变换完全取决于数组线性划分后的结果，并且同名数组变量和异名数组变量能够以相同的形式进行变换。

参 考 文 献

1 Gupta M, Banerjee P. Demonstration of Automatic Data Partitioning Techniques for Parallelizing Compilers on Multicomputers. IEEE Transactions on Parallel and Distributed Systems, 1992, 3(2). 179～193

2 Guo M. Efficient Techniques for Data Distribution and Redistribution in Parallelizing Compilers: [PhD thesis]. University of Tsukuba, 1998

3 Kennedy K, Kremer U. Automatic Data Layout for High Performance Fortran. In: Proc. Suppercomputing 95, San Diego, Calif. Dec. 1995

4 Kandemir M, et al. A Linear Algebra Framwork for Automatic Determination of Optimal Data Layouts. IEEE Transactions on Parallel and Distributed Systems, 1999 (Feb). 115～135

5 Wolfe M. High Performance Compilers for Parallel Computing. Addison-Wesley, 1996

6 Banerjee U. Dependence Analysis for Supercomputing Norwell, Mass, Kluwer Academic Publishers, 1988

7 Ramanujam J. Compiler-time Techniques for Data Distribution in Distributed Memory Machines. IEEE Transactions on Parallel and Distributed Systems, 1991, 2(4): 472～482

8 Du Jiancheng, Chen Daoxu, Xie Li. JAPS: An Automatic Parallelizing System Based on JAVA. Science in China, 1999, 3: 279～288

9 胡南军，陈道蓄，谢立. 并行编译中数据分布的线性划分模式计算. 待发表