第32卷 第24期 计 算 机 工 程 2006年12月
Vol.32 No.24 Computer Engineering December 2006
·博士论文· 文章编号：1000—3428(2006)24—0026—03 文献标识码：A 中图分类号：TP311

并行编译中一种线性数据和计算划分算法

董春丽，韩 林，赵荣彩
(解放军信息工程大学信息工程学院，郑州 450002)

摘 要：对于高性能并行计算机而言，如何找到一种好的计算和数据划分，对数据和计算进行合理划分，增加数据本地化来减少处理器间的通信是提高其并行性能的关键。该文讨论了一种线性的自动进行无数据重组的计算和数据划分算法。
关键词：并行编译；数据划分；计算划分；循环级并行

Linear Computation and Data Decomposition Algorithm in Parallel Compilation

DONG Chunli, HAN Lin, ZHAO Rongcai
(School of Information Engineering, PLA Information Engineering University, Zhengzhou 450002)

【Abstract】 For high-performance parallel computer, finding a good decomposition of computation and data, minimizing communication by increasing the locality of data references is an important optimization for achieving high performance. This paper describes a linear decomposition algorithm that automatically finds computation and data decomposition and no data reorganization communication.
【Key words】 Parallel compiler; Data decomposition; Computation decomposition; Loop level parallel

随着高性能并行计算机[1]的发展，程序自动并行化成为计算机领域一个重要的研究课题。使用先进的并行化编译技术[2]，自动将串行程序转换为等价的能在并行计算机上高效运行的并行程序，是克服并行计算机编程因难、软件移植困难的主要手段，是充分发挥高性能并行计算机系统潜能的有效途径。20 世纪 70 年代中期以来，并行化编译在共享内存并行机和向量机上有了很大的突破和发展，但在分布内存体系结构上，并行编译技术的发展一直较慢。在分布存储模式下，各个处理器有自己的存储器，处理器间的通信只有通过消息传递来进行，而消息传递的开销在很大程度上抵消了并行带来的好处，甚至在有些情况下并行执行时间大于串行执行时间，所以分布体系结构上的并行化编译必将从传统的以计算分布为核心转为以数据分布为核心。希望找到一个数据划分方案使得并行计算中各处理器尽量引用本地数据，减少通信开销，因此如何确定一个好的数据分布和计算分布模式以获得最大的数据局部性和程序并行化成为分布存储器并行计算机并行化编译的关键技术，分布算法的优劣将直接影响到并行性和通信代价的高低。

本文将讨论一种对计算和数据进行线性分解的算法，通过分析此算法，对所处理程序中的数组将得到一致的数据分布，即在整个程序中数据的分布方式只有一种，不会出现数据重组现象，在程序执行过程中不会出现由数据重组引起的通信，减少了通信开销。此算法适用于分布和共享内存体系结构。

1 计算和数据划分算法的数学模型

1.1 问题的提出

在分布存储模式下，对单一循环的最大限度并行化和本地化[3]的技术已经比较成熟，而如何对多循环进行最优的并行化和本地化则是一个需要解决的问题。

对例 1 来讲，如果按单循环分别考虑循环 L1、L2，则循环 L1 没有依赖关系，最内最外层均可并行，数组 X、Y 可按行也可按列划分；循环 L2 中最外层（i1）有依赖关系，只可内层并行，数组 Z 按行划分，Y 按列划分。若按多循环考虑，则循环 L2 中 i2 层并行，数组 Z 按行划分，Y 按列划分，循环 L1 中 i2 层并行，数组 X、Y 按列划分。所以数组的最终分布是数组 X、Y 按列划分，数组 Z 按行划分。

例 1
```
L1:    For i=0 to N
       For j=0 to N
            X[i][j]+=Y[i][j];
            L2:    For i=0 tn N
       For j=0 to N
            Z[j][i]=Z[j][i-1]+Y[i][j-1];
```

Kennedy和Kremer的研究表明寻找最佳的数据分布模式的问题是NP完全问题[4]，但很多研究者针对特定的程序模式提出了许多数据分布方案。本文所讨论的算法是针对程序中的多个循环嵌套进行循环级并行，能够进行并行处理的循环嵌套所要满足的条件为：循环边界和数组下标是循环索引变量和符号常量的线性函数，且循环迭代次数远大于处理器数。本文中的算法首先要建立一组满足分解条件的仿射关系方程式，通过解方程来找到满足条件的数据划分和计算分解。问题的关键在于如何建立这样一组方程以及如何有效地求解方程。

1.2 算法的数学模型

在分布存储的机器上，循环级并行的并行性是通过循环嵌套迭代空间的分解来实现的。将循环及数组分布过程分为

作者简介：董春丽(1970－)，女，博士生，主研方向：计算机软件与并行编译；韩 林，博士生；赵荣彩，博导
收稿日期：2005-12-27 **E-mail：**DCLLDH@sina.com

两步：(1)通过将循环迭代和数组元素映射到一个规模不受限的虚拟处理机阵列上，建立每个迭代与它所访问的数组元素之间的对应关系。(2)将虚拟处理器阵列通过一个模板映射到实际的物理处理器上。第(2)步和具体机器的物理结构有关，本文所涉及的算法不考虑具体目标机的处理器结构，即只讨论第(1)步。

1.2.1 处理的循环类型

在对问题进行分析时，使用仿射函数来表示数据分布和计算分解[5]，且假定循环和数组在从源代码到中间代码的变换中已经过标准化处理，即循环是下界为 0、步长为 1 的标准化循环，数组下标从 0 开始，循环嵌套深度为L，循环边界为循环索引变量的仿射函数(例 2)。

例 2 L1: For i=0 to N

For j=0 to N

X[i11][j11]=X[i12][j12]+…X[i1n][j1n]+Y[i21][j21]+…;

其中，数组下标 $i_{m*n} = f_{mn}(i, j)$，$j_{m*n} = g_{mn}(i, j)$ 为循环索引变量的线性函数。

1.2.2 涉及的基本概念

在建立数学模型时，所需表示的向量空间有

(1)循环迭代空间 L ：它是一个 l 维空间，用于表示深度为 l 的循环嵌套的迭代空间，其空间中的任意一点表示循环嵌套的一次迭代，可用向量 $\vec{i} = (i_1, i_2, ..., i_l)$ 来表示；

(2)数组下标空间 A ：它是一个 m 维的空间，用于表示 m 维数组的数据分布空间，其空间中的任意一点为一个数组元素的下标，可用数组下标向量 $\vec{a} = (a_1, a_2, ..., a_m)$ 来表示；

(3)处理器空间 P ：它是一个 n 维的空间，用于表示 n 维的处理器，其空间中的任意一点表示一个处理机的位置，可用其位置向量 $\vec{p} = (p_1, p_2, ..., p_n)$ 表示。

基于上述 3 种向量空间，可以对处理器、循环迭代和数组元素建立以下 3 种对应关系。

定义 1 l 维循环迭代空间到 m 维数组下标空间的映射，用于表示循环迭代与数组元素之间的引用关系：数组访问仿射函数 $L -> A: f(\vec{i}) = F\vec{i} + \vec{k}$ ，其中 F 是一个 $l * n$ 的线性转换矩阵，$\vec{k}$ 是常向量。

定义 2 m 维数组下标空间到 n 维处理器空间的映射，用于说明数组元素与处理器空间之间的对应关系：数组分布仿射函数 $A -> P: d(\vec{a}) = D\vec{a} + \vec{\delta}$ ，其中 D 是一个 $n*m$ 的线性转换矩阵，$\vec{\delta}$ 是常向量。

定义 3 l 维循环迭代空间到 n 维处理器空间的映射，用于表示计算分解：计算划分仿射函数

$$L -> P: c(\vec{i}) = C\vec{i} + \vec{\gamma}$$

其中 C 是一个 $n * l$ 的线性转换矩阵，$\vec{\gamma}$ 是常向量。

在上述 3 个定义中，将计算和数据划分的线性部分用线性矩阵 C、D 表示，划分的偏移部分用常向量 $\vec{\delta}$ 、$\vec{\gamma}$ 表示。

在算法模型中，对于循环嵌套内的所有语句作为一个整体考虑，不考虑具体的每一条指令，即对循环嵌套的每一次迭代，仿射函数 $c(\vec{i})$ 指出此次迭代中的所有语句分布到哪一个虚拟处理器。对程序中的循环嵌套进行数据划分和计算分解的问题可以分为以下 3 步进行：

(1)数据和计算划分：数据和计算划分决定了可以被分到同一处理器的数组元素和循环迭代，即决定了 D 矩阵和 C 矩阵的核。

由定义 2 可知，当由 $\vec{a}_1$、$\vec{a}_2$ 确定的两个数组元素分配到同一处理器时，不考虑偏移量则有

$$D\vec{a}_1 = D\vec{a}_2 => D(\vec{a}_1 - \vec{a}_2) = \vec{0} => \vec{a}_1 - \vec{a}_2 \in \ker D$$

若 $\vec{a}_1 - \vec{a}_2 \notin \ker D$ ，则由 $\vec{a}_1$、$\vec{a}_2$ 确定的数组元素分配到不同的处理器。(注： $\ker D$ 是矩阵 D 的零空间，它是由所有使得 $D\vec{a} = \vec{0}$ 的 $\vec{a}$ 向量组成。)

同样，由定义 3 可知，当两次循环迭代被分配到同一处理器时，则有

$$C\vec{i}_1 = C\vec{i}_2 => C(\vec{i}_1 - \vec{i}_2) = \vec{0} => \vec{i}_1 - \vec{i}_2 \in \ker C \quad (1)$$

若 $\vec{i}_1 - \vec{i}_2 \notin \ker C$ ，则由 $\vec{i}_1$、$\vec{i}_2$ 确定的循环迭代分配到不同的处理器。

(2)数据分布和计算划分的定向：由 D 矩阵和 C 矩阵中元素的正负号决定，其描述了数组元素、循环迭代和处理器之间的映射方向。

(3)数据分布和计算划分的平移：给出了数据分布和计算划分相对于处理器的初始偏移量，由常向量 $\vec{\delta}$ 和 $\vec{\gamma}$ 决定。

2 线性分解算法

2.1 基本线性分解等式

无流水通信和无数据重组的数据和计算划分算法仅适用于 doall 循环，它可以使数据分布和计算划分问题简化，并可清楚地体现出算法基本思想。算法基本思想如下：设循环嵌套 j 的计算划分矩阵为 C_j，数组 x 的数据划分矩阵为 D_x，且在循环嵌套 j 中数组 x 的第 k 次访问函数为 $f^k{}_{xj}$ ，对于所有的循环迭代 $\vec{i}$ ，当且仅当：

$$D_x(f^k{}_{xj}(\vec{i})) + \vec{\delta}_x = C_j(\vec{i}) + \vec{\gamma}_j \quad (2)$$

则该循环迭代和它所引用的数组元素分布在同一处理器，对于由式(2)得到的分解称之为线性分解。

由于由平移引起的通信量可以通过块划分来减少，使问题简化，因此在处理时先不考虑平移，即不考虑常向量 $\vec{\delta}$、$\vec{\gamma}$ 和 $\vec{\kappa}$ ，问题可简化为求解矩阵方程：

$$D_x F^k{}_{xj}(\vec{i}) = C_j(\vec{i}) \quad (3)$$

若数组元素和循环迭代满足式(3)的关系，则这种划分是一种无重组无流水通信的划分，这种分解称为基本线性分解。

在式(3)中，F_{xj} 为数组元素下标和循环索引变量之间的映射关系，因为其为已知，所以数据和计算划分问题简化为求解矩阵 D_x 和 C_j。对式(3)来讲，若忽略等式两边的迭代空间向量 $\vec{i}$ ，则等式变为 $D_x F^k{}_{xj} = C_j$ ，对此等式求解可以得到很多组解，包括把所有数据和计算分配到一个处理器的解，我们的目的在于求出一组解使得数据和计算尽可能分到不同的处理器，即找到一组能发现程序最大并行性的解。从数学上来讲，将找到一组解使得矩阵 C_j 的秩尽可能大。

由以上分析，可得出结论：在一段程序中，如果所有的循环嵌套和数组的 C、D 矩阵都为全零阵，则可以不引起重组通信，但此时找不到可利用的并行性，每个循环嵌套的所有迭代都分配到同一处理器中，因此程序只能串行执行，即 $C_j = \vec{0}, D_x = \vec{0}$ ，$\ker C_j$ 为 L 的整个循环迭代空间，$\ker D_x$ 为 A 的整个数组下标空间。$\ker C_j$ 的维数越小，划分出的超平面越多，被利用的并行度越大。当 $\ker C_j$ 为仅含零向量的空间时，所有循环迭代都分配到不同的处理器中，达到完全并行。

2.2 偏移量的计算

线性分解算法是由基本线性分解等式和偏移量组成，下

面将讨论如何求解偏移量。由式(2)和式(3)及定义 1 可得对循环嵌套 j 的计算分解偏移量方程为

$$\vec{\gamma}_j = D_x \vec{k}^{\,k}{}_{xj} + \vec{\delta}_x \tag{4}$$

同理，可得对数组 y 的数据划分偏移量方程为

$$\vec{\delta}_y = \vec{\gamma}_j - D_y \vec{k}^{\,k}{}_{yj} \tag{5}$$

在上述等式中，数据划分矩阵已知，偏移量 $\vec{k}$ 已知，通过一种简单的贪心法策略即可求得分解中的偏移量。

3 一个简单的例子

首先来看一个例子。

例 3

```
int   x[N][N],y[N][N],z[N][N]
L1:       For i=0   to N do
                For j=0 to N do
                        x[i][j]=y[i][j]+z[i][j];
L2:       For i=0 to N do
                For j=0 to N do
                        y[i][j]=y[i][j]+x[i][j];
```

在对数组进行静态分解时，首先需求出数组的访问函数，在上述例 3 中的数组访问函数 F 分别为：在第 1 个循环嵌套 L_1 中数组 x、y 和 z 的访问函数分别为

$$F_{x1}{}^{1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}、F_{y1}{}^{1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}、F_{z1}{}^{1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

在第 2 个循环嵌套L_2中数组y和x的访问函数分别为

$$F_{y2}{}^{1} = F_{y2}{}^{2} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}、F_{x2}{}^{1} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

根据式(3)不考虑偏移可得

$$D_x F_{x1}{}^{1} = D_y F_{y1}{}^{1} = D_z F_{z1}{}^{1} = C_1$$

$$D_x F_{x2}{}^{1} = D_y F_{y2}{}^{1} = D_y F_{y2}{}^{2} = C_2$$

由上述等式可得

$$C_1 = C_2 = [1 \quad 1]$$

$$D_x = D_y = D_z = [1 \quad 1]$$

经过编写的具有自动查找并行的编译程序分析划分结果与上述手工分析结果一致，算法有一定的应用价值。

4 结论

并行化编译系统是并行计算机的系统软件中十分重要的一部分，是充分发挥并行计算机性能、实现超级计算的关键。本文仅就过程内循环嵌套所引用数组如何进行一致性划分进行了分析，在实践中，通过程序测试，自动划分结果和手工划分结果一致，验证了算法的正确性。由于此算法得到的数据划分是过程内全程不变的，对于大型程序，可能找不到一致的数据划分。因此，对于数据重组不可避免的情况下，怎样更好地利用数据的性质，找到一种动态的数据和计算划分方法，最大限度地减少通信开销，以及过程间的数据和计算划分算法的研究将是今后工作的重点。

参考文献

1 金怡濂, 黄永勤, 陈左宁等. 高性能计算机的关键技术和发展趋势[J]. 中国工程科学, 2001, 3(6): 1-8.

2 沈志宇, 胡子昂, 廖湘科等. 并行编译方法[M]. 北京: 国防工业出版社, 2000.

3 Kennedy K, McKinley K S. Optimization for Parallelism and Data Locality[C]. Proceedings of the ACM International Conference on Supercomputing, 1992: 323-334.

4 Kennedy K, Kremer U. Automatic Data Layout for High Performance Fortran[C]. Proc. of Supercomputing, San Diego, Calif., 1995.

5 Anderson J M, Lam M S. Global Optimizations for Parallelism and Locality on Scalable Parallel Machines[C]. Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation, 1993: 112-125.

(上接第 5 页)

图 3(b)是去除定位横线后的灰度图像，图 3(e)是本文方法的二值化结果。在二值化第 1 步过程中，大部分的黑笔画被成功保留，少部分印章处的笔画被去除，导致笔画断裂，如图 3(c)中字符“元”。在二值化第 2 步过程中，一些印章碎块被去除，避免了后期增长的可能，如图 3(e)。经过最后的区域增长过程，断裂的笔画被修复，如图 3(e)。本文还给出了Cheriet算法[1]和逻辑层算法[6]对图 3(b)的二值化结果，分别如图 3(f)、图 3(g)。可以看出，本文提出的方法优于这两种方法。Cheriet算法是一种全局阈值法，由于过分割，造成了字符笔画的断裂。逻辑层法是一种局部方法，背景中的印章字符被作为目标误提取。在 60 张大写金额子图像上的实验显示，本文的方法总体上获得了比Cheriet算法和逻辑层算法更好的效果。但是实验结果也表明，对于书写的笔画较细、灰度较淡，而印章灰度较深时，本文方法还不能获得令人满意的效果。

3 结论

票据图像的预处理是票据自动处理系统中的重要环节之一，其处理效果的好坏直接影响到系统的整体性能。我国支票背景复杂，尤其是大写金额子图像，背景中包含定位线、纹理线以及使用过程中盖上的各式各样印章图像，其中一些盖在了字符上。现有的一些目标提取方法不能很好地解决这种情况。本文设计了用于大写金额子图像目标提取的方法。针对颜色较深，形状已知的定位线采用形态方法去除，通过灰度和双边缘分析以及区域增长的方法对去线后的图像进行二值化处理，实验表明了本方法的有效性。

参考文献

1 Cheriet M, Said J N, Suen C Y. A Recursive Thresholding Technique for Image Segmentation[J]. IEEE Trans. on Image Processing, 1998, 7(6): 918-921.

2 Palumbo P W, Swaminathan P, Srihari S N. Document Image Binarization: Evaluation of Algorithms[C]. Proc. of SPIE, 1986.

3 Parker J R. Gray Level Thresholding in Badly Illuminated Images[J]. IEEE Trans. on PAMI, 1991, 13(8): 813-819.

4 Liu Y, Srihari S N. Document Image Binarization Based on Texture Features[J]. IEEE Trans. on PAMI, 1997, 19(5): 540-544.

5 Xiangyun Y, Cheriet M, Suen C Y. Stroke-model-based Character Extraction from Gray-level Document Images[J]. IEEE Trans. on Image Processing, 2001, 10(8): 1152-1161.

6 Kamel M, Zhao A. Extraction of Binary Character/graphics Images from Grayscale Document Images[J]. CVGIP, 1993, 55(3): 203-217.

7 Zhang C Y, Chen Q, Lou Z, et al. Extraction of Courtesy Amount Item from Chinese Check[C]. Proc. of the 8th International Conference on Control, Automation, Robotics and Vision, 2004: 6-9.

8 Ye X, Cheriet M, Suen C Y, et al. Extraction of Bankcheck Items by Mathematical Morphorlogy[C]. Proc. of J. Doc. Anal. Recognit., 1999: 53-66.

9 Otsu N. A Threshold Selection Method from Gray-level Histograms[J]. IEEE Trans. on Systems, Man, and Cybernetics, 1979, 9(1): 62-66.