第 35 卷 第 4 期 计 算 机 工 程 2009 年 2 月
Vol.35 No.4 Computer Engineering February 2009
·软件技术与数据库· 文章编号：1000—3428(2009)04—0085—03 文献标识码：A 中图分类号：TP314

基于 Define-Use 分析的冗余通信消除算法

王军委[1,2]，赵荣彩[1]，李 妍[3]
(1. 解放军信息工程大学信息工程学院，郑州 450002；2. 解放军信息工程大学理学院，郑州 450001；3. 郑州大学软件学院，郑州 450002)

摘 要：针对并行代码自动生成过程中产生的大量冗余通信代码，提出基于 Define-Use 分析的冗余通信消除算法。将中间代码的每一个过程划分为不同的块，同时收集各块中对数组变量的定义和引用信息。以块为节点，按控制流关系构造控制流图。以控制流图为基础，根据块间各数组变量的 Define-Use 关系，确定需要通信的位置，从而消除冗余通信代码，达到优化通信的目的。测试结果表明，该算法可有效提高并行程序的执行效率。

关键词：并行代码；冗余通信；优化

Algorithm to Eliminate Redundancy Communication Based on Define-Use Analysis

WANG Jun-wei[1,2], ZHAO Rong-cai[1], LI Yan[3]
(1.Institute of Information Engineering, PLA Information Engineering University, Zhengzhou 450002; 2. Institute of Science, PLA Information Engineering University, Zhengzhou 450001; 3. Software School, Zhengzhou University, Zhengzhou 450002)

【Abstract】 Against the redundant communication code produced in the process of automatic parallel code generation, this paper presents an algorithm to eliminate redundancy communication based on Define-Use analysis. Intermediate code is divided into all kinds of blocks by process, and the information of defining and using of array is collected. Blocks are constructed into Control Flow Graph(CFG). Based on CFG, the positions needed communication are fixed according to Define-Use analysis to eliminate redundancy communication code and optimize communication. Test result shows that the algorithm effectively improves the performance of parallel code.

【Key words】 parallel code; redundancy communication; optimization

1 概述

在分布存储结构并行计算机系统中，通信开销成为影响并行程序执行效率的关键因素之一。因此，在针对分布存储结构并行机的串行程序自动并行化研究中，必须充分考虑分布存储系统的特点，对通信进行优化，降低通信开销。通常，一次通信开销可表示为[1]

$$T_{comm} = T_{start} + T_{copy(n)} + T_{transit(n)}$$

其中，T_{comm} 是总通信时间；T_{start} 是通信启动时间，对于 1 个确定的目标系统而言是常数；$T_{copy(n)}$ 是通信的发起方建立 1 个消息或通信接收方从 1 个消息中接收数据的时间，是消息长度为 n 的函数；$T_{transit(n)}$ 是消息的传输时间，也是消息长度的函数。

可以看出，通信优化的途径主要有 2 个：

(1)减少通信次数，即减少 T_{start}；

(2)减少通信的数据量，即减少 $T_{copy(n)}$ 和 $T_{transit(n)}$。

通信量的大小与并行软件的结构和数据分布、计算划分的策略有必然联系。在分布存储结构并行系统中，只有保证数据分布和计算划分的一致性和对齐关系，通信量才会最小，但在实际应用中是不可能的，常用的方法是通过执行消除冗余通信、消息合并和消息向量化等通信优化措施来减少 T_{start}，$T_{copy(n)}$ 和 $T_{transit(n)}$，达到降低通信开销的目的。

本文主要讨论如何利用数据流分析找出单个过程内数组变量间的 Define-Use 关系，由 Define-Use 关系确定需要广播的位置，从而实现消除冗余广播、优化通信的目的。

2 基本概念

2.1 冗余并行执行模型

在本文的并行化系统中生成的目标并行程序采用冗余并行执行模型(Redundant Parallel Execution Model, RPEM)。该模型的程序执行分为多个阶段，每个阶段的执行模式是 3 种类型之一[2]：冗余并行执行，完全并行执行，流水并行执行。不能并行化的串行代码被复制，在所有的处理机上冗余并行执行。并行循环被划分到多个处理机上以完全并行方式执行。流水循环被划分到多个处理机，在多个处理机间以流水线方式并行执行。程序在 RPEM 模型下的执行过程如图 1 所示。

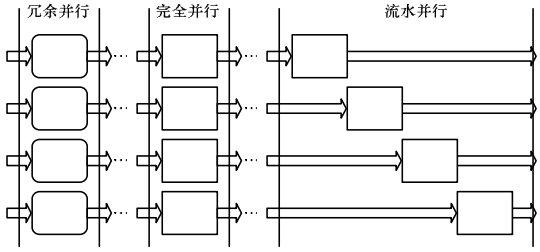

图 1 RPEM 执行模型

在 RPEM 模型中，不同的执行阶段间需要进行通信。每个阶段开始前的通信称为前通信，其作用是从数据的拥有者

作者简介：王军委(1979－)，男，硕士研究生，主研方向：计算机软件，并行编译；赵荣彩，教授、博士生导师；李 妍，硕士研究生
收稿日期：2008-07-04 **E-mail：**junwei79@yahoo.com.cn

处获得数据。每个阶段结束后的通信称为后通信，其作用是将在本处理机上修改过的数据发送给数据的拥有者，保证数据的一致性。另外，在流水并行执行阶段为了保证流水的正确性，因此，存在流水通信。

2.2 数据划分和数据收集

在分布存储计算机系统中，各个处理器拥有本地存储模块，每个并行实体均有自己独立的地址空间，一个并行实体不能直接访问另一个并行实体中的数据[3]。处理器间通过消息传递机制进行通信。由于处理器访问本地存储模块的速度远快于通过网络访问异地处理器存储模块的速度，因此需要在自动并行化过程中对串行代码进行正确分析，找出其中可并行的部分，然后进行合理的计算和数据划分，理想的结果是每个处理器计算所需的数据均为本地数据，但实际上由于数组应用不可能完全对齐等原因很难实现这种理想划分，因此，在每个并行迭代计算结束之后，根据数据划分结果，把各个处理器计算结果进行收集，再将收集到的数据进行广播，从而达到理想划分的效果。但这种方法会带来大量的冗余通信。本文主要针对该问题进行研究。

3 基于 Define-Use 分析的冗余通信消除

本文所讨论的通信优化在前端程序并行性分析结果的基础上展开，其基本思想是：(1)将中间文件的每一个过程划分为不同的块，同时收集各块中数组变量的 Define 和 Use 信息；(2)以块为节点，按控制流关系构造控制流图；(3)以控制流图为基础，根据块间各数组变量的 Define-Use 关系，确定需要通信的位置，最终达到消除冗余通信的目的。

3.1 块和控制流图的生成

基于前端提供的并行性分析结果和通信优化的目的，本文将中间文件划分为以下 3 种类型的块：

(1)位于并行区开始标志和并行区结束标志之间的代码作为 1 个块，称为可并行节点。

(2)每一条过程调用语句分别划作为一个块，称为过程调用节点。

(3)其他代码按传统的基本块划分方法，分别划分为不同的块，称为串行节点。

块的具体划分算法如下：

(1)设置一个布尔型的全局标志变量 *InParallel*，表示当前代码是否可并行执行。若该变量为假，表示当前代码为串行代码；若该变量为真，则当前代码可并行执行，缺省值为假。处理每条语句之前，首先判断 *InParallel* 的值。

(2)若标志变量 *InParallel* 为假，则分别求出各块的入口语句为：

1)标识并行区开始的语句，遇到该语句时，将 InParallel 设置为真；

2)程序的第 1 个语句；

3)过程调用语句；

4)能由转移语句转移到的语句；

5)紧跟在条件转移语句后面的语句；

6)紧跟在标识并行区结束的语句后面的语句；

7)紧跟在过程调用语句后面的语句。

(3)若全局标志变量 *InParallel* 为真，则当前代码可并行执行，从当前代码处继续向后查找，直至标识并行区结束的语句为止，将 *InParallel* 设置为假。

(4)对以上所求出的每一个入口语句，构造其所属的块[4]：它是由该入口语句到另一入口语句(不包括该入口语句)，或到一转移语句(包括该转移语句)，或到一停语句(包括该语句)之间的语句序列组成。

(5)将控制流信息增加到块的集合上。如果一个块中包含程序的第 1 条语句，则称此节点为首节点。如果在某个执行顺序中，块 B_2 紧接在块 B_1 之后，则从 B_1 到 B_2 有一条有向边。即，如果[4]：

1)有 1 个条件或无条件转移语句从 B_1 的最后一条语句转移到 B_2 的第 1 条语句；

2)在程序的序列中，B_2 紧接在 B_1 的后面，并且 B_1 的最后一条语句不是一个无条件转移语句。

就称 B_1 是 B_2 的前驱，B_2 是 B_1 的后继。至此完成中间文件块的划分和基于块的控制流图的建立。

3.2 冗余通信的消除

在 RPEM 下，由于不可并行的串行代码被复制，在所有的处理机上冗余并行执行，串行节点中对变量所作的定义在所有的处理机上都是有效的，无须进行广播，因此只要对并行节点中的变量定义处理即可。

在控制流图中，依次对每一个并行节点中的各个数组定义进行分析。如果当前并行节点中对数组变量定义之后没有对该变量的引用，或再次引用前被重新定义，或在本地引用该变量，则该定义可不广播。消除冗余通信的算法如下：

输入 *CurrentNode*：当前正在处理的并行节点
FromNode：调用本算法遍历控制流图时的起始节点
A：当前正在处理的数组变量

输出 *Annote_Bcast*：需要广播的标志

(1)若 CurrentNode 节点后已有变量 A 的广播标志，返回；否则，转(2)。

(2)设置 FromNode 为已遍历。

(3)判断 FromNode 是否有未处理的后继节点，若有，转(5)；否则，转(4)。

(4)设置 FromNode 为未遍历，返回。

(5)取一个 FromNode 未处理的后继节点 Node，若 Node 已遍历，转(3)；否则，转(6)。

(6)若 Node 是过程调用节点，转(7)；否则，转(8)。

(7)若 Node 中的过程调用是 I/O，转(21)；否则，转(25)。

(8)若 A 在 Node 中存在定义或引用，转(9)；否则，转(21)。

(9)若 Node 是并行节点，转(10)；否则，转(20)。

(10)若 A 在 Node 后做归约，转(3)；否则，转(11)。

(11)若 A 在 CurrentNode 后做归约，转(12)；否则，转(14)。

(12)若 A 在 Node 中存在引用，转(25)；否则，转(13) 。

(13)若 A 在 Node 后做精确收集，转(3)；否则，转(25)。

(14)若 A 在 CurrentNode 和 Node 中的数据划分一致，转(15)；否则，转(18)。

(15)若 A 在 Node 中存在引用，转(16)；否则，转(17)。

(16)若 A 在 CurrentNode 和 Node 中的数据划分对不齐，转(25)；否则，转(17)。

(17)若 A 在 Node 中存在定义，转(3)；否则，转(21)。

(18)A 在 CurrentNode 和 Node 中的数据划分不一致，若 A 在 Node 中存在引用，转(25)；否则，转(19)。

(19)若 A 在 CurrentNode 后做精确收集，转(3)，否则，转(25)。

(20)Node 是串行节点，若 Node 中存在对 A 的引用，转(25)；否则，转(21)。

(21)若 Node 是控制流图的尾节点，转(22)；否则，转(24)。

(22)若当前过程是 main，转(3)；否则，转(23)。

(23)若 A 是当前过程中的局部变量，转(3)；否则，转(25)。

(24)以 CurrentNode, Node 和 A 作为输入递归调用本算法，转(3)。

(25)在 CurrentNode 节点后插入变量 A 的广播标志，返回。

上述算法将当前控制流图中的每一个并行节点作为 *CurrentNode*，对其中的每一个变量定义 *A*，都通过调用本算法判断 *A* 是否需要在当前并行节点所包含的代码执行完毕后广播，每处理一处变量定义 *A*，当前并行节点都要调用 1 次本算法，每次调用本算法时，*CurrentNode* 和 *FromNode* 均指向当前要处理的并行节点。

4 性能分析

本文选取基准测试集进行测试。使用优化前的算法生成并行程序，称为优化前的生成；在该算法中加入冗余通信消除算法生成优化的并行代码，称为优化后的生成。对优化前后生成的并行程序分别进行测试，程序都能正确执行且通信部分都有不同程度的消除，通信量比优化前减少 15%~25%，在执行时间上部分程序优化后明显较优化前减少。可见，本文的算法提高了生成的并行化程序的性能。为说明算法效果，给出实例代码如下：

```
#define N 100
int i,j,k;
int a[N][N],b[N][N][N];
for(i=0; i<N;i++)
for(j=0; j<N; j++)
a[i][j]=i+j;
for(i=0; i<N;i++)
for(j=0; j<N; j++)
for(k=0; k<N; k++)
b[i][j][k]=i+j-k;
for(i=0; i<30;i++)
for(j=0; j<N; j++)
for(k=0; k<N; k++)
a[3*i][j]=b[i][j][k];
for(i=0; i<30;i++)
for(j=0; j<N; j++)
a[2*i+5][j]=b[3*i+1][j][3];
```

对该实例优化前后的性能进行测试，测试环境为 SunWay 机群，SunWay 机群由 7 个节点构成，每个节点含有 2 个主频为 2.8 GHz 的 Xeon 处理器，4 GB 内存，16 MB 二级缓存，128 KB 一级缓存。节点间通过百兆交换机相连。操作系统为 Red Hat Linux 7.2 2.96-118.7.2 smp，MPI 编译器为 MPICH 1.2.6。测试结果为：生成的并行代码优化前含有 4 个广播，优化后仅有 1 个广播，冗余通信减少了 75%，加速比的结果见图 2。

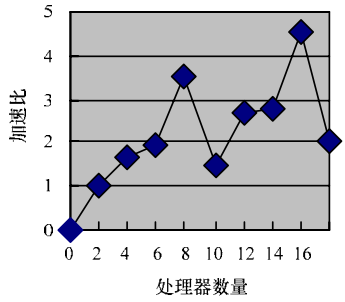

图 2 加速比结果

5 结束语

SUIF 自动生成的并行代码有很大的优化空间，本文通过冗余通信代码的消除，可有效提高并行程序性能。但这只是初步结果，在今后的工作中，将测试大量程序，对各种情况进行分类，继续改进算法，同时对其他优化方法进行研究，进一步提高并行程序的执行效率。

参考文献

[1] 张 平. 并行化编译器中并行程序自动生成和性能优化技术研究[D]. 郑州: 解放军信息工程大学, 2006.

[2] 钟洪涛, 舒继武, 温冬婵, 等. 基于区域图数据流分析的通信优化算法[J]. 软件学报, 2003, 14(2): 175-176.

[3] 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.

[4] 陈火旺, 刘春林, 谭庆平, 等. 程序设计语言编译原理[M]. 3 版. 北京: 国防工业出版社, 2000.

(上接第 84 页)

5 结束语

本文针对相关窗口匹配方法中费用函数计算量大的问题，通过对计算过程的分析和研究，提出了一种能有效消除冗余计算、增加匹配速度而不改变匹配精确度的优化方法，且对相关窗口大小的选择具有不敏感的特性。实验证明本方法是可行和有效的。

参考文献

[1] 周秀芝, 文贡坚, 王润生. 自适应窗口快速立体匹配[J]. 计算机学报, 2006, 29(3): 473-479.

[2] Kanade T, Okutomi M. A Stereo Matching Algorithm with an Adaptive Window: Theory and Experiment[J]. IEEE Trans. on Pattern Analysis and Machine Intelligence, 1994, 16(9): 920-932.

[3] Roberto F V, Trucco E. Efficient Stereo with Multiple Windowing[C]//Proc. of IEEE Conference on Computer Vision and Pattern Recognition. [S. l.]: IEEE Computer Society Press, 1997.

[4] Veksler O. Stereo Matching by Compact Windows via Minimum Ratio Cycle[C]//Proc. of ICCV'01. [S. l.]: IEEE Computer Society Press, 2001.

[5] Veksler O. Fast Variable Window for Stereo Correspondence Using Integral Images[C]//Proc. of Conference on Computer Vision and Pattern Recognition. [S. l.]: IEEE Computer Society Press, 2003.

[6] 李海滨, 张 强. 一种新的基于子线段的立体匹配算法[J]. 光学学报, 2007, 27(5): 907-912.

[7] 文贡坚. 一种基于特征编组的直线立体匹配全局算法[J]. 软件学报, 2006, 17(12): 2471-2484.

[8] 徐彦君, 杜利民, 侯自强. 基于相位的尺度自适应立体匹配方法[J]. 电子学报, 1999, 27(7): 38-41.

[9] Stefano L D, Marchionn M, Mattoccia S, et al. A Fast Area-based Stereo Matching Algorithm[C]//Proc. of the 15th International Conference on Vision Interface. [S. l.]: IEEE Press, 2004.