第34卷 第16期 计 算 机 工 程 2008年8月
Vol.34 No.16 Computer Engineering August 2008
·软件技术与数据库· 文章编号：1000—3428(2008)16—0095—02 文献标识码：A 中图分类号：TP314

程序自动并行化中的数组终写关系分析

罗 勇，张 平，龚雪容
(信息工程大学信息工程学院，郑州 450002)

摘 要：在程序自动并行化中过程中，数据收集阶段可能产生冗余通信，该文利用数组终写关系分析的方法来消除冗余通信，实现嵌套循环中数组数据最后写关系的快速求解，并将结果提供给编译器后端，生成精确数据收集代码。描述数组终写关系的研究目的和内容，将所处理的嵌套循环根据其结构特征进行分类，给出实现算法的过程。测试结果证明了该算法的正确性和高效性，所产生的精确数据收集代码能够有效地消除部分冗余通信，从而优化和提高了并行化程序的性能。

关键词：并行编译；数组终写关系分析；精确数据收集；自干扰；线性不等式

Array Last-write Relation Analysis for Automatic Program Parallelization

LUO Yong, ZHANG Ping, GONG Xue-rong
(College of Information Engineering, University of Information Engineering, Zhengzhou 450002)

【Abstract】 In order to eliminate redundant communication generated in data collection of automatic program parallelization, array last-write relation analysis is proposed to implement fast computation of array data last-write relation in loop nest, whose result is necessary condition for generating accurate data collection code in backend of compiler. This paper describes the aim and content of array last-write analysis, sorts the loop nests into different types according to their construction characteristic, and presents the algorithm implementation process in detail for each type. The test result shows the high performance and accuracy of the algorithm. And the accurate data collection code can efficiently eliminate part of redundant communication, which optimizes the parallel programs.

【Key words】 parallel compiling; array last-write relation analysis; accurate data collection; self-interfere; linear inequality

1 概述

在分布式系统进行串行程序自动并行编译过程中，冗余通信消除是一个非常重要的问题。由于数据交换技术的缺陷，冗余通信可能存在于并行计算的各个阶段(计算前数据分布、计算中数据同步和计算后数据收集)，影响着并行化程序的性能。目前，精确数据收集方法能够有效地减少并行计算后所需收集的数据量，一定程度上消除存在于数据收集过程中的冗余通信，提高并行化程序的运行效率[1]。

然而，精确数据收集必须具备的前提条件为：数组数据的最后一次写，一直以来都没有合适的求解方法。2002年，文献[2]给出数组数据最后一次写的中间表示，但没有给出计算方法。近年来，并行自动识别工作组对数组数据最后一次写的求解方法展开了深入研究：

(1)2006年，首先提出了利用等价类的方法获取数据的最后写关系，并给出了相应的数学证明，但是该方法需要遍历整个迭代空间，求解最后写关系的效率不高[1]；然后提出利用数组数据流分析方法处理嵌套循环中输出依赖问题，将循环迭代空间分割成多个子空间，并给出每个子空间中有效的写引用，进而确定数组数据的最后一次写[3]。

(2)2007年，提出了终写关系分析的概念，讨论了输出依赖关系和自干扰写引用的处理方法[4]。

经过大量的手工分析、理论论证和数学推导，工作组最终提出了数组终写关系分析方法，实现了数组数据最后一次写的快速求解。在并行自动识别工具KAP[3](Kit for Automatic Parallelism)中将结果提供给编译器后端产生精确数据收集代码，得到了较好的实验结果。

2 基础知识

2.1 写引用的迭代空间I[3]

写引用外层是一个深度为 l 的循环嵌套，这个循环嵌套定义了一个 l 维的整型迭代空间，用向量 $\boldsymbol{i}=(i_1,i_2,\cdots,i_l)$ 表示，其中，任意一点表示循环嵌套的一次迭代。

2.2 非自干扰迭代空间I_{ns}

在嵌套循环中，如果一个写引用在不同迭代中，访问的数组元素都不同，那么这个写引用是非自干扰的；如果写引用在不同迭代中，先后访问了同一个数组元素，那么这个写引用是自干扰的[5]。

写引用对所有数组数据的最后一次访问迭代构成其非自干扰迭代空间。写引用在其非自干扰迭代空间中所访问的数组元素都不会被其自身在以后的迭代中再次修改。

2.3 终写迭代空间I_{lw}

在嵌套循环中，定义数组数据最后一次写所发生的迭代为其终写迭代，执行修改动作的写引用为其终写引用。

具有相同终写引用的所有数组元素，构成相应写引用的

作者简介：罗 勇(1981－)，男，助理工程师、硕士研究生，主研方向：并行编译，数据依赖分析；张 平，副教授、博士；龚雪容，助理工程师、博士研究生

收稿日期：2007-12-10 **E-mail：**Ying_dogs@yahoo.com.cn

终写数组空间，其终写迭代构成相应写引用的终写迭代空间。

写引用的终写迭代空间和终写数组空间大小相等，两者之间是一一对应的。

3 算法分析与实现

数组终写关系分析以写数组引用为研究对象，从嵌套循环的索引信息和写引用下标表达式入手，分析各层索引之间、各写引用之间以及写引用与索引之间的相互关系，在线性不等式基础上，求解嵌套循环中每一个写引用的终写迭代空间。

根据嵌套循环和写引用的结构特点，将串行程序进行了分类：将写引用分为非自干扰写引用和自干扰写引用；将循环根据循环中写引用的数量分为单引用循环和多引用循环。下面将针对这些不同类型的程序分别给出数组终写关系分析的过程和算法。

3.1 单引用循环下的非自干扰写引用分析

单引用循环下的非自干扰写引用是最简单的一种情况。非自干扰写引用的非自干扰迭代空间与其外层索引构成的迭代空间相同，嵌套循环中不存在其他写引用，则其终写迭代空间与非自干扰迭代空间相等。

3.2 单引用循环下自干扰写引用分析

自干扰分析的内容是确定自干扰写引用的非自干扰迭代空间。在嵌套循环中，写引用发生自干扰的情况有很多种。文献[5]定义了未使用索引(unused loop indices)，并说明完美循环(perfect loop)中大部分写引用自干扰都是由于嵌套循环中的未使用索引所引起的。

定义 未使用索引[5]

如果某层索引不在写引用下标表达式中出现，也不会对任何一个出现在写引用下标表达式中的索引的取值范围造成影响(直接或间接)，对于相应写引用来说，这个索引就是未使用索引。

在未使用索引取不同值时，写引用会对相同的数组元素进行多次访问。若未使用索引有 k 个不同的取值，则写引用对每一个数组元素都会访问 k 次，而只有当未使用索引取最大值时才是最后访问。求解由未使用索引引起自干扰写引用的非自干扰空间时，需要将未使用索引取最大值作为其非自干扰空间的约束。

3.3 多引用循环分析

如果嵌套循环中都只有单个写引用，写引用的非自干扰迭代空间I_{ns}与终写迭代空间I_{lw}相等。如果嵌套循环中存在多个写引用对同一个数组进行访问(同名写引用)，则在求解每一个写引用的终写迭代空间时，需要考虑同名写引用之间的互写关系，也就是输出依赖关系。嵌套循环中写引用之间的输出依赖关系可以用精确数组数据流分析LWT(Last Write Tree[5])算法得到。LWT算法是针对嵌套循环中数据流依赖关系而提出的，它能指出一对读、写引用对存在流依赖的具体迭代范围。实践证明，LWT算法同样适用于处理嵌套循中的输出依赖关系。

对于存在输出依赖关系的同名写引用对，可以利用LWT算法进行输出依赖分析，建立依赖关系不等式，对这个不等式进行傅立叶消元(Fourier Motzkin Elimination, FME)[6]，可以得到相应写引用受其他写引用干扰的迭代空间I_{ww}。终写迭代空间为$I_{lw}=I_{ns}-I_{ww}$。

3.4 算法流程

输入

(1)循环嵌套 L。

(2)L 的索引信息 ni。

(3)L 中所有写引用集合 R。

输出 R 中所有写引用的终写迭代空间注释

步骤如下：

(1)将 R 复制两个备份 $P=Q=R$。

(2)从P中提取一个写引用a，根据ni求得其外层索引迭代空间I_1，$R=Q-a$。

(3)判断a是否自干扰，若是，则查找未使用索引，对未使用索引取最大值构成约束条件添加到I_1中，得到a的非自干扰空间I_2；若a是非自干扰引用，则$I_2=I_1$。

(4)若R为空，则a的终写迭代空间$I_3=I_2$。

若 R 不为空，对任意一个写引用 $b\in R$ 分别与 a 进行输出依赖分析：

1)从 R 中提取一个写引用 b。

2)*Dependence_test*(b,a)判断 b 是否依赖于 a，$P=P-a$。

3)若 b 依赖于 a，则 *LWT_test*(b,a)建立依赖关系不等式 F。

4)用FME消去F中的指定列，得到a受b干扰的迭代空间I_4。

5)$I_2=I_2-I_4$，返回 1)，重复 1)~5)，直到分析完R中所有写引用。

6)$I_3=I_2$。

(5)将a和其终写迭代空间I_3写入中间注释。

(6)判断 P 是否为空，若空则结束；不空则返回(2)，重复(2)~(6)。

4 性能测试

4.1 算法适用范围

(1)for 循环都经过规范化：步长为 1，内层索引变量上界、下界是外层索引和符号常量的线性表达式；

(2)写数组引用是规则的：数组访问函数是外层索引变量和符号常量的线性表达式；

(3)索引上界、下界和写引用访问函数中所有变量的常系数为 0 或±1。

4.2 测试平台

测试目标机为 SunWay 机群。该机群由 8 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。节点间通过千兆交换机相连。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。

4.3 测试结果

数组终写关系分析算法通过了对 PPOPP 和 NPB 测试集的正确性测试，串行代码和相应自动并行程序的运行结果都是一致的。在串行程序并行编译过程中，发现引入数组终写关系分析前后的编译时间并没有明显的变化，证明了数组终写关系分析算法的高效性。性能测试主要是通过比较引入精确数据收集前后并行化程序在相同条件下的运行时间，以 NPB 中 Lu、Bt 和 Sp 程序为例，结果见表 1。

表 1 数组终写关系分析性能提高百分比

程序名	处理器个数			
	2	3	8	16
Lu	1.8	9.1	26.2	12.1
Bt	1.7	7.8	30.8	23.2
Sp	12.7	0.5	29.8	15.0

由表 1 可以看出，在经过精确数据收集和对程序结构的优化后，程序的性能提高的范围为 0.5%～30.8%。由此可知，数组终写关系分析的引入是切实有效的。 (下转第 161 页)