第4卷第4期
1993年8月

软件学报
JOURNAL OF SOFTWARE

Vol. 4, No. 4
Aug. 1993

自动并行编译的新进展

朱根江　谢立　孙钟秀

(南京大学计算机系，南京 210008)

THE DEVELOPMENT OF AUTOMATICALLY PARALLELIZING COMPILER

Zhu Genjiang, Xie Li and Sun Zhongxiu

(*Department of Computer Science, Nanjing University, Nanjing* 210008)

Abstract Automatically parallelizing compiler (APC) is one of the approaches in parallel programming. This paper reviews the development and describes the importance of APC. The techniques for parallelizing compiler are discussed. The paper also concludes with a look at the future.

摘要 自动并行编译是并行程序的主要途径之一. 本文概述了发展自动并行化编译的必要性及其主要进展，讨论了当前采用的主要技术和今后的发展动向.

§1. 引　言

近年来，MIMD 多机系统受到了普遍的关注，市场上已有许多商品化的产品，如 Cray YMP、BBN Butterfly、Convex C－2、Encore、Sequent、Alliant、FX/2800、INMOS 传输计算机等，特别是 Thinking Machines 公司的 CM－1、CM－2 与 CM－5. 这表明，体系结构的研究已取得较大进展，但与之相比，软件技术进展却不大，因而严重阻碍了并行处理的应用. 尤其是对 message passing 类的系统进展更少.

早期的并行化工作往往只是对向量化作直接的模仿或扩充，但 MIMD 系统上的并行化工作与 SIMD 系统上的向量化有着根本的不同. 一般来说，MIMD 机器在执行并发任务时，可发掘出比 SIMD 向量处理机更多的并行性. 并行处理研究的内容十分广泛，包括体系结构、并行操作系统、并行程序设计及并行算法等领域. 进入九十年代，并行处理已成为计算机关键的主流技术之一.

本文主要讨论自动并行编译问题. 一般来说，并行程序设计可采用二种主要途径：

1. 显式并行语言

这种语言的设计有两种方法：传统串行语言加上并行结构的扩充或者设计全新的并行语言，这要求程序员必须具有并行的概念. 这样的语言有：SISAL、Strand、FORCE、

本文1991年4月收到. 作者**朱根江**，博士生，1991年硕士毕业于南京大学，主要研究领域为并行处理，自动并行编译. **谢立**，教授，南京大学副校长，主要研究领域为分布式系统，分布式数据库和智能操作系统. **孙钟秀**，教授，副校长，学部委员，主要研究领域为分布式系统，分布并行处理.

LINDA、Parlog、Concurrent Pascal、PCF FORTRAN 等等.

2. 串行语言,并行编译

用户采用单机上传统的串行语言编制程序,并行编译程序自动检测串行程序中的并行性,转换成在多机系统上可执行的并行代码.

通常,使用显式并行语言解决一个并行任务,用户面临下列问题:①逻辑分解,即寻找一种适应并行处理的数据和代码划分方法. ②逻辑分解到 MIMD 系统的映射. 为取得良好的性能,程序员必须考虑资源分配、负载平衡及调度等问题. ③数据存取方式(局部的还是全局的). 在无共享内存时,还必须处理数据的定位及进程同步等问题.

上述要求远远超出了程序设计中对算法的描述问题,它们期望用户充分具备并行性的知识,深入了解 MIMD 的系统结构,而后者往往是与机器有关的. 因此,由用户考虑并行性,不仅大大增加了其负担,且为诸如死锁、读写竞争这些棘手的故障的发生大开了方便之门;其次目前并行语言尚无统一的标准(PCF Fortran 除外),并行程序设计方法学还不完善,不同的机器往往提供自己的并行语言,而且大都包含与机器有关的成分,故移植性差.

以上所述表明,使 MIMD 多机系统广为接受的唯一途径是自动的并行编译,即用户使用传统的串行语言,由编译程序自动识别程序固有的顺序性,充分挖掘潜在的并行性,加上运行系统的支持负责进程分解、同步、调度和负载均衡这些低级实现. 传统串行语言的使用,用户已经轻车熟路,而现有的正在运行的大量串行程序,已花费了巨额投资,藉手工将这些程序一一转换为并行程序远非易事,因此自动化的并行编译既是行之有效,又是比较现实的途径.

本文第 2 节概述了当前自动并行编译(主要是数据值语言)的发展,第 3 节介绍自动并行编译的主要技术,第 4 节讨论自动并行编译进一步值得研究的问题,最后是结语.

§2. 自动并行编译的发展

并行编译有两层含义:串行程序的并行化,其主要过程是将串行程序转换为并行代码;并行程序的编译程序. 本文主要讨论数值型串行程序的并行化.

串行程序的并行编译大体上可看成由两部分组成:与机器无关的分析和与机器有关的处理. 与机器无关部分是编译的预处理,主要进行依赖关系分析及有利于并行化的程序转换;与机器有关的部分涉及进程的分配、通信、同步及调度等. 系统结构的不同,并行编译的工作也不一样. SIMD 的系统(如流水线等),由于有硬件的支持,有指令级的并行,并行粒度可以很小,而 MIMD 系统,由于通信开销的原因(一次通信可能需要成百上千条指令的运行时间),则着重中粒度和大粒度并行性的挖掘. 另外,互连结构和存贮组织对并行化工作也会产生很大的影响.

并行编译过程基本上可分为三个阶段:词法和语法分析、优化和并行代码的生成. 其中优化部分是并行编译的主体,它包括三部分:依赖关系分析,识别各种依赖关系,如数据依赖、控制依赖等;程序转换,主要是循环的转换;进程的分配、同步、通信及调度,调度在理论上应使各处理机做等量的工作,但由于负载的不确定性. 系统中的处理机数目、循环的非均匀分布等诸多因素的影响,调度是非常复杂的,通常,寻找最佳的调度方法一般采用动态的自调度. 在传统的概念中,调度是操作系统的任务,而现在越来越倾向于由并行

编译实现部分的调度，以得到高效的并行性能.

2.1 自动并行编译的进展

七十年代末，美国 ILLINOIS 大学研制的 Parafrase 开创了向量化及并行化的先河，随之出现了更多的 Fortran 向量化(并行化)工具，如 Rice 大学的 PFC 等，加上 Kuck、Banerjee、Wolf 和 Allen 等人的工作，为向量化及后来的并行化工作奠定了理论基础.

八十年代末期，MIMD 系统的并行化工作已相继展开，如 ALLIANT FX/8 的 APC Fortran，Rice 大学的 Parascope，交互式并行化工具 PTOOL 和 PAT[3]，ILLINOIS 大学进一步的工作 Parafrase－2[9]，IBM 的 PTRAN，瑞士的 Oxygen[1]，Kuck&Associates 的 KAP 等等. 并行化工作比向量化要复杂、困难，并行化的依赖关系更为复杂，常常需要同步；并行性的识别不仅仅限于循环，还要挖掘子程序级的并行性；复杂数据结构如指针、结构的依赖关系分析；粒度控制问题；并行化还必须考虑多台处理机的计算的次序等等. 另外，由于 MIMD 系统的可扩充性的伸缩性，并行化工作受到了愈来愈多的重视.

大部分自动并行化工具能处理循环级的并行. Parafrase－2 是多语言的并行化工具，其对象语言是 C、Fortran、Pascal. 交互式的并行化工具 PTOOL 给程序员提供一个接口，显示依赖信息，帮助程序员进行并行代码的调试和程序重构. PAT 所提供的依赖信息使程序员能够逐步地进行程序的转换，而始终保持依赖关系图的正确性，因此 PAT 的输入可以是串行的或部分并行的 Fortran 代码. Oxygen 是分布式存贮多机系统上的并行化编译，而 KAP 则是共享式存贮多机系统上的并行编译. 分布式存贮系统必须进行数据和代码的分解，分解技术应该是数据驱动的分解，即需考虑对数据访问的局部性.

§3. 并行编译技术

3.1 依赖关系测试

3.1.1 数据依赖

数据的依赖关系主要有四种[2]：流依赖、反依赖、输出依赖和控制依赖. 其中流依赖是固有依赖，其他的依赖如输出依赖和反依赖在一定条件下可通过换名扩张及设置必要的附加语句消去，控制依赖可转化为数据依赖[4]. 循环的并行化主要是通过依赖关系分析，改变原串行程序词法次序，以缩短执行时间而不改变程序的语义. 因此，依赖关系分析提供了保证串行程序语义正确性的条件.

标量数据的依赖最为直观，而对数组仅考虑数组名会失去很多并行性，因此必须分析其下标表达式. 考虑如下的二重循环：

```
for I=1 to N do
  for J=l(I) to u(I) do
        ……
    A(f1(I,J), f2(I,J))=…;
        ……
    =A(g1(I,J),g2(I,J));
        ……
  end;
end;
```

其中，f_i 和 g_i 及 l、u 均为线性函数，依赖问题变为求解方程组：

$$\begin{cases} f_1(i_1, j_1) = g_1(i_2, j_2) \\ f_2(i_1, j_1) = g_2(i_2, j_2) \end{cases} \tag{1}$$

$$\begin{cases} 1 \leqslant i_1 \leqslant N,\ l(i_1) \leqslant j_1 \leqslant u(i_1) \\ 1 \leqslant i_2 \leqslant N,\ l(i_2) \leqslant j_2 \leqslant u(i_2) \end{cases} \tag{2}$$

由(1),四个变量两个方程式,这是求解丢番图方程的问题,其通解是满足(2)式的两个自由变量的线性函数,在平面直角坐标系中形成一个依赖凸边域 DCH(Dependence Convex Hull).

编译时刻依赖关系的测试方法主要有两种:精确测试法和近似测试法.精确测试法试图求出丢番图方程的通解,适用于方程数目较少且循环具有确定的上下界的情况.精确测试失效时,可使用近似测试,近似法仅给出一些存在解的必要条件.DCH 是精确测试法,而 GCD 法,λ—法和 Banerjee 法等都是近似测试法.

· GCD 测试法:

若丢番图方程为 $a_1x_1+a_2x_2+\cdots+a_nx_n=c$,则存在解的充要条件是系数 $a_1, a_2, \cdots, a_n$ 的最大公约数能除尽 c.

· Banerjee 测试法:

设丢番图方程 $h: a_1x_1+a_2x_2+\cdots+a_nx_n=c$ 在区域 R 上连续,则存在解的充要条件是:

$$\min_R h \leqslant c \leqslant \max_R h$$

3.1.2 过程及其调用分析

过程内分析主要涉及别名分析、常数传递分析以及引用分析.

在程序优化之前,采用内联扩展的办法即将被调用过程通过参数代换嵌入到调用处,最终整个程序中只包含唯一的过程,虽然这种方法简单有效,但耗内存太厉害,因而是一种消极的办法.

过程内分析是为了给出被过程隐蔽了的更为详细的信息以利依赖分析.文[11]提出了一种数组线性化处理技术,但线性化过程可能失去精确性.另有人提出了无递归的循环中的过程调用分析[10],依赖关系的分析通过检测一组表示常用语句和被调用过程内的数组引用的线性不等式而得到,但其检测过程相当费时.文献[8]给出的分析是基于所谓原子图(atom image)的方法.原子图用于传递低级过程中的下标及上下界,可用于循环并行化、数组别名,识别它毋需数组线性化等额外手段,能有效地分析递归调用.

3.2 循环的并行化

循环并行化之前应先进行预优化工作(如有必要),包括循环的规范化(尽可能线性化,以便于数据依赖关系的测试)等.循环的并行化分为三个阶段:依赖关系分析(仅考虑跨迭代依赖),循环转换和循环调度.

循环的转换技术有许多种,如循环的交换、合并、分布、排列、分片和环路收缩等等.总的说来,不外乎两类:不改变循环体,而对循环变量的集合进行划分;改变循环体.

3.2.1 对循环体划分

DOPIPE 技术将循环体分成多个等长度的分区,每个分区分配给一台处理机,每分区的各次迭代由一台处理机顺序执行,而一次迭代的执行则跑遍所有的处理机.

循环分布技术把不能并行化的大循环分解成几个顺序执行的小循环,这些小循环本

身有几个甚至全部可以并行执行，而并行循环之间是串行执行的（否则可合并成较大的并行循环）.

3.2.2 对循环变量的迭代空间的划分

对循环变量的迭代空间进行划分，形成互不依赖的子域，各子域间无需信息交换. 如最小距离法[12]，环路收缩[13]，循环分片等方法.

1. 环路收缩

循环的向量化/并行化在循环体存在依赖环路时比较困难. 环路收缩是 ILLINOIS 大学的 Polychronopoulus 提出的方法，考虑单重循环的情形，数组的依赖距离均为常数，环路为：S1δS2δ...SnδS1，不失一般性，设依赖距离为：$\varphi_1 \geqslant \varphi_2 \geqslant \cdots \geqslant \varphi_n$；设 $\varphi_i > 1$，让 $\lambda = \varphi_1$（距离最小者），将迭代（上界为 N）分为 r+1 组：

$$v_0 = [1, \cdots, \lambda], v_1 = [\lambda+1, \cdots, 2\lambda], \cdots, v_r = [r\lambda+1, \cdots, N]$$

则组与组之间须串行执行，每一组迭代可并行执行，因此这种方法的并行度为 λ，当 λ=1 时，环路收缩方法失效. 更一般的情况见[13].

2. 循环的分片

循环的分片（tiling，又称 blocking、partition 等）研究的很多[14~16]，文[7]提出了分布式存贮的多机系统上的循环分片方法. 采用 Lamport[6]的结果，并放宽了 Lamport 给定的条件，将循环的迭代空间分成多个独立的子区域（即分片），当子区域远远大于处理机数目时，定义了每个分片的形状和大小，为了解决分布式存贮的多机系统中启动通信次数过多的问题，采取成组通信的办法，即每个分片执行前必须得到所需的全部数据，执行完后将需要通信的数据一次发送.

循环的分片问题应将编译技术和并行算法中划分技术相结合，编译技术将循环的迭代划分成互不交换信息的子域；并行算法划分将全局数组分成标准的几何形状，如正方形、矩形、六边形等. 互不依赖的子域数往往小于系统中的可用的计算机数而不能充分利用系统中资源，因此分片时，通常首先根据依赖关系对迭代划分，再根据计算机数目、计算量和通信量来决定分片的形状和大小.

3.2.3 其他

目前，大部分循环的并行化方法通过对迭代的分解，由迭代与迭代之间的覆盖来得到并行性，即仅考虑跨迭代的依赖，而不考虑一次迭代内的并行性. [17]给出的方法是一种新的非向量化循环的并行化技术，可以在 MIMD 系统上同时得到迭代内和迭代间两种并行性，其代价是花费一定的通信开销. 分析结果表明，性能优于 DOACROSS[5]等方法.

循环的并行化工作已有许多，限于篇幅，在此不能一一介绍.

§4. 需进一步研究的问题

在并行程序设计方法学还不完备、并行语言的移植性较差及用户负担过重等情形下，串行语言的自动化并行编译是一条现实的道路，但是并行化技术还很不成熟，许多问题有待深入的研究，主要表现在以下几个方面：

· 依赖关系的分析技术

具体的算法中的特殊性，程序员为提高执行时间而使用了技巧，数组下标耦合的非线性化，过程内部分析的复杂性、递归和深层嵌套，这些都使编译时刻的依赖分析变得极为

困难.另外循环的并行化过程,仅局限于数组的依赖关系分析还远远不够(如C及Pascal等包含了指针、结构等复杂数据),因此有效的依赖关系分析技术还有待开发.

• 中间形式表示、程序转换

PDG的方法已广为采用,但它只适用循环级并行性的检测,另外就是中等大小的程序模块用PDG表示时,数据依赖关系已经非常复杂,算法的开销也太大.MIMD系统上的并行编译不仅要发掘循环级的并行性,还应发掘子程级及非DO结构的并行性,寻找合适的中间表示.另外,个别程序转换技术并不一定能显著地提高性能,因此,多种转换技术的结合使用也是值得注意的问题.

• 粒度控制

MIMD系统中,通信和同步的开销对并行性能的影响重大.一个程序的计算量可以完全分解,而通信开销却不然,小粒度并不一定能增加加速比.许多工作表明,并行效率往往并不取决于通信量的问题,而是由通信量与计算量之比决定.因此必须在MIMD系统中解决通信量和计算量之间的平衡问题.

• 运行时刻的分析

由于启发方法(如丢番图求解)往往在编译时不能测试出所有的依赖关系,这时就得求助于运行时刻的分析,由于编译时刻的分析已提供了许多有用的信息,运行时的开销通常是可以接受的.因此编译和运行时刻的分析技术相结合可能是并行化的唯一途径.

§5. 结束语

并行编译的工作具有十分重要的意义,它的发展必将推动MIMD系统的广泛应用.采用显式并行语言编制的程序,不仅对程序员的要求很高,而且用户介入对进程的创建、同步会导致错误的增加、调试的困难以及移植性差等问题.自动并行化卸去了用户的负担,由编译程序自动识别各种可能的并行性,既有效又可靠,是实行处理的一条有效而较为现实的途径,值得大力研究.

参考文献

1 R. Ruhl and M. Annaratone, Parallelzing of Fortran Code on Distributed Memory Parallel Processing, In International Conf. on Supercomputing, 1990, 342—353.

2 U. Banerjee, Dependence Analysis for Supercomputing, Kluwer Academic Publishers, Boston, MA, 1988.

3 K. Smith, B. Appelbe and K. Strirewalt, Incremental Dependence Analysis for Interactive Parallelization, In International Conf. on Supercomputing, 1990, 330—341.

4 J. R. Allen, K. Kennedy, C. Porterfield and J. Warren, Conversion of Control Dependence to Data Dependence, In Proceedings of the 1983 Symposium on Principles of Programming Languages, 177—189.

5 R. Cytron, Doacross: Beyond Vectorizing for Multiprocessors, In Proc. of 1986 ICPP, Vol. 2, 834—844.

6 L. Lamport, The Parallel Execution of Do Loops, Comm. ACM, 17, 1974. 2, 83—93.

7 J. Ramanujam and P. Sadayappan, Tiling of Iteration Space for Multiprocessors, In Proc. of 1990 ICPP, Vol. 2, 179—186.

8 Z. Li and P. C. Yew, Interprocedural Analysis for Parallel Computing, In Proc. of 1988 ICPP, Vol. 2, 221—228.

9 C. D. Polychronopoulos, M. Girkar, M. R. Haghighat, C. Lee and B. Leung, Parafrase—2: An Environment for Parallelizing, Partitioning, Synchronizing, and Scheduling Programs to Multiprocessors, In Proc. of 1989

ICPP, Vol. 2,39—48.

10 R. Triolet, F. Irigoin and P. Feautrier, Direct Parallelization of CALL Statements, In Proc. of the ACM SIGPLAN'86 Symp. on Compiler Construction, ACM SIGPLAN Not. Vol. 21, No. 6, June 1986.

11 M. Burke and R. Cytron, Interprocedural Dependence Analysis and Parallelization, In Proc. of the ACM SIGPLAN'86 Symp. on Compiler Construction, ACM SIGPLAN Not. Vol. 21, No. 7, July 1986.

12 J. K. Pier and R. Cytron, Minimum Distance: A Method for Partitioning Recurrence for Multiprocessors, In Proc. of 1987 ICPP, 217—225.

13 C. D. Polychronopoulos, Compiler Optimizations for Enhancing Parallelism and Their Impact on Architecture Design, IEEE Trans. on Computer, Vol. 37, No. 8, Aug. 1988, 991—1004.

14 F. Irigoin, and R. Triolet, Supernode Partitioning, In Proc. of 15th Annual ACM Symp. Principles of Programming Languages, Sun Diego, CA, Jan. 1988, 319—329.

15 C. King and L. Ni, Grouping in Nested Loops for Parallel Execution on Multiprocessors, In Proc. of 1989 ICPP, Vol. 2, 31—38.

16 M. Wolf, More Iteration Space Tiling, In Proc. of Supercomputing 89, Reno NV, Vol. 13—17, 1989, 655—664.

17 K. Kim and A. Nicolau, Parallelizing Non—Vectorzable Loops for MIMD Machines, In Proc. of 1990 ICPP, Vol. 2, 114—118.

本刊推荐使用“词表”一书选择论文的关键词

词表的编者曾对某刊 3 期作者所用的关键词作过统计，一共用了 111 个关键词. 从中看到同一类的关键词却用了不同关键词标识，如：计算机辅助制造，有的用中文全称，有的用英文缩写 CAD，还有的用 CAD/CAM. 一个词的多种表示，给用户查找使用带来不便.

《**计算机科学技术汉语叙词表**》适用于标引和检索计算机科学技术的理论基础、生产技术、制造工艺、应用管理以及市场等各方面的国内外文献；同时也适用于与计算机专业密切相关的计算数学、自动化、电子、电工等专业的文献标引的检索.

为了计算机检索的需要，使同一类论文的关键词相对集中，保证用户查准查全，以使文章被充分的检索利用，建议投稿者使用《**计算机科学技术汉语叙词表**》一书作选择论文关键词的依据. 该词表中没有的新词，作者可加新词，但应在该词的右上角加 * 表示，如四叉树*.

《**计算机科学技术汉语叙词表**》一书于 1990 年由清华大学出版社出版，单价：10.00 元. 欲购者请按下述地址联系：**北京 中关村 中科院计算所 19 室 王能琴同志　邮政编码** 100080　**联系电话**：2565533—479.