科学出版社

金国华　陈福接 著

大规模并行机程序并行化理论与技术

大规模并行机程序并行化理论与技术

金国华　陈福接 著

科学出版社
1995

(京)新登字 092 号

内 容 简 介

本书以开发面向大规模并行系统又兼顾适度并行系统的并行化技术和工具为目的，从尽可能开发程序并行性和优化程序并行执行两方面对有关过程间相关性分析、不规则并行性开发、真假共享 cache 行抖动和并行程序优化等并行处理领域中尚未解决的关键性问题进行了深入研究，着重反映了作者在该领域所取得的最新研究成果。

本书可供从事编译、并行处理技术、并行机系统理论和应用研究的专家和学者参考，也可作为高等理工科院校计算机科学、工程和应用专业攻读博士、硕士学位研究生的教材。

大规模并行机程序并行化
理 论 与 技 术

金国华　陈福接 著

责任编辑　刘晓融

科 学 出 版 社 出版

北京东黄城根北街 16 号

邮政编码：100717

化学工业出版社印刷厂印刷

新华书店北京发行所发行　各地新华书店经售

*

1995 年 1 月第　一　版　开本：787×1092　1/16

1995 年 1 月第一次印刷　印张：8 1/4

印数：1—1000　　字数：185 000

ISBN 7-03-004365-0/TP・399

定价：12.40 元

前　言

由于物理上的极限和工程实现上的问题，单机速度已很难满足高科技领域对计算机性能的要求，因此并行机结构受到普遍关注。大规模、超大规模集成电路，特别是微处理器技术的逐渐成熟为并行计算机结构提供了技术-物质基础，使并行机系统的发展和应用成为可能。然而，并行机结构的迅速发展并没有得到软件的相应支持，并行机系统的性能普遍没有得到充分的发挥。人们希望并行计算机能尽快地得到软件的支持，使它能够更高效地运行，更方便地为人们所用。并行化研究为此提供了一条很重要的途径。

我们在深入分析现有并行化关键技术和工具系统的基础上，尽可能从开发程序并行性和优化程序并行执行两方面对有关过程间相关性分析、不规则并行性开发、真假共享 cache 行抖动、并行程序优化等并行处理领域中尚未解决的或尚未得到很好解决的问题进行了深入研究，旨在开发面向大规模并行处理系统又兼顾适度并行系统的并行化技术和工具系统。

本书是以我们多年来对并行处理的理论和技术所进行的研究为基础撰写而成的，着重反映当前国际上并行化领域的前沿研究情况，并详细介绍了我们近年来在该领域已经取得的研究成果。适合于从事语言和编译、并行处理技术、并行机系统和应用研究的专家和学者阅读参考，可作为高等院校计算机科学、工程和应用专业攻读博士、硕士学位研究生的教材。

全书共分八章，第一章简要介绍大规模并行机并尽可能详细而客观地给出并行化研究的历史介绍和现状述评，然后扼要介绍我们的方法和并行化系统 KD-PARPRO；第二章介绍有关相关性分析的基本概念；第三章着重分析别名对程序并行化的影响，介绍强化别名分析方法；第四章介绍能有效、完全开发循环迭代间不规则并行性的编译方法，包括描述迭代间不规则并行性的两种不同表示方法，它们的编译产生算法和相应的自调度并行代码；第五章主要介绍编译消除多级存储大规模并行处理系统中频繁出现的真共享 cache 行抖动的思想、理论和方法；第六章讨论多级存储大规模并行处理系统中各种访问模式下的假共享 cache 行抖动现象和真假共享抖动并存现象，介绍消除假共享抖动的编译方法——块化错位方法；第七章给出一种并行程序优化方法——异类循环交换。

在本书成文过程中曾得到国防科技大学陈火旺教授、胡守仁教授、周兴铭教授、吴泉源教授，西北工业大学康继昌教授、西安交通大学郑守淇教授等各方面的关心。复旦大学计算中心朱传琪教授，美国密歇根州立大学 Lionel M. Ni 教授，明尼苏达大学 Z. Li 博士，加利福尼亚大学 S. Eggers 博士，密歇根大学 D. Hudak 博士，赖氏大学 D. Callahan 博士、K. McKinley 博士、P. Havlak 博士、W. Hall博士，普林斯顿大学 A. Rogers 博士，卡内基-梅隆大学 H. Ribas，威斯康星大学的 L. Huelsbergen 博士，伊利诺斯大学 D. Lilja，奥地利维也纳大学 M. Gerndt 博士，法国里昂高等师范大学Y. Robert教授和 J. Li 博士等为我们提供了许多最新技术资料，我们一并在此表示衷心的感谢。

由于并行处理还是一门正在迅速发展的新兴学科，许多内容还处于探索阶段，加之我们的研究所涉及的范围有限，书中可能有错误和不当之处，欢迎读者批评指正。

1994 年 6 月

目　录

第一章 绪 论

早在计算机发展的初期，人们就已认识到了大规模并行的重要性，而且在计算机发展过程中也一直试图在系统的各个层次上开发并行性。然而，由于其应用范围与技术条件的限制，早期成果并不显著。近年来，高科技领域对计算机性能提出了越来越高的要求，而物理上的极限和工程实现上的问题使单机速度很难满足应用要求，因此并行机结构，尤其是大规模并行机结构引起了普遍关注。大规模、超大规模集成电路，特别是微处理器技术的逐渐成熟为大规模并行计算机结构提供了技术-物质基础，使大规模并行机系统的发展和应用成为可能，大规模并行机将是21世纪的主流机。

1.1 大规模并行计算机

大规模并行的思想起源于50年代。1950年，计算机之父Von Neumann提出了“自复制细胞自动机”的概念。1958年，Steven Unger提出了构造两维SIMD阵列机的设想；1963年曾按这种构想提出了两种方案：Soloman系统和Illiac Ⅲ，但均以失败告终。1972年，Illinois大学与Burrough公司合作研制的Illiac Ⅳ可以说是大规模并行机的鼻祖。该机原拟做四个象限，每个象限由8×8个处理器(PE)构成，每个PE可以和上下左右四个PE通讯，这种设计思想，对多处理机阵列结构的研究及设计产生了极大的影响。由于当时硬、软件水平所限，Illiac Ⅳ只做了一个象限，且不太成功。80年代中期，随着超大规模集成电路技术的发展，单片微处理机性能的提高和并行处理技术的进步，产生了新一代大规模并行处理机。TMC公司1986研制成CM-1，1987年推出改进型CM-2，1992年又推出世界上最快的机器CM-5。美国Intel公司从80年代中期开始，先后推出了iPSC/1，iPSC/2，iPSC/860三代大规模并行处理机，最近又研制成功Touch stone Delta系统。nCUBE公司1986年研制成nCUBE/10，1989年推出nCUBE2，1991年推出改进型nCUBE2S，进入90年代，MPP开始了大发展时期。除Intel，TMC和nCUBE等公司之外，众多著名的计算机公司如Cray，Convex，Kendall Square，IBM，DEC等都纷纷宣布自己的MPP计划。到目前为止，已有不少公司推出了自己的大规模并行处理机(见表1.1)。

CRAY T3D MPP系统是一个具有可全局寻址分布存储结构的可伸缩MIMD系统，T3D系统采用DEC Alpha微处理器作为处理单元，单个处理单元的峰值性能可达150MFLOPS，每个处理单元内又有一个局部存储器，访问本处理单元的局部存储器要比访问其它处理单元的局部存储器快得多(NUMA)。T3D系统采用三维环形网格的网络拓扑确保短连接路经和高频带。通讯路径是双向的，路由机制采用“Virtual-Cut-Through”和“Wormhole-Routing”相结合。T3D PE还包含有高效同步机制支持控制并行和数据并行。T3D系统通过多个高速通道实现和I/O子系统的连接(每个通道每个方向的传输率为200MBytes/s)。

表 1.1　一些主要的 MPP 系统

机　型	公　司	处理机最大数目	主存最大容量	理论峰值	芯　片
CM-5	Thinking Machine	16K	1TB	ITFLOPS	SPARC
Paragon	Intel	4K	128GB	300GFLOPS	i860xp
Cray T3D	Cray Research	2K	1TB	307GFLOPS	Alpha
nCUBE3	nCUBE	64K	65TB	6.5TFLOPS	自行设计
KSR1	Kendall Square	1088	34GB	43GFLOPS	自行设计
SPP-1	CONVEX	128	32GB	25.6GFLOPS	PA-7100

Kendall Square KSR1 采用自行设计的 64 位超标量芯片作为处理机结点，结点机峰值速度为 40MFLOPS，KSR1 通过 ALLCACHE 实现了共享虚拟存储(SVM)，由 ALLCACHE 引擎自动管理数据的分配，移动和 cache 间的数据一致性。引擎间以胖树拓扑互连形成了一个多级 ALLCACHE 引擎结构。KSR1 中每个处理机有一个 30MBytes/s 的标准 I/O 通道。

TMC 的可伸缩多处理机系统 CM-5 最多可以包含有 16K 个节点处理机，每个结点都有一个标准的 SPARC 微处理器和各自的存储器。节点机又分控制节点和处理节点两类，控制结点用于负责管理工作，处理节点主要完成大量数值计算，每个处理节点又可以包含有 4 个向量处理部件，峰值性能可达 128MFLOPS，CM-5 有两个内部网络支持处理机间通讯。低延迟控制网络负责全局通讯操作，高频带数据网络支持点对点通讯网络操作，它可以和计算操作同时进行，网络拓扑采用冗余四叉树形结构，路由器采用随机路由机制。由于数据网络的支持，每个处理节点至少有 5MBytes/s 的 I/O 频带。

正在研制的 nCUBE3 可伸缩 MIMD 系统处理机节点数目最多为 65536，每个节点内包含有高性能 nCUBE3 微处理器，局部存储器和网络接口，系统最高性能可达 6.5TFLOPS，nCUBE3 采用超立方体结构，通过引入新的直接存储访问(DMA)通道特性，改善 Cut-through 路由策略，提供内部数据缓冲来提高传输速度，减少消息启动和传输延迟，处理机通讯频带高达 24TBytes/s 以上。系统总存储容量可达 65TBytes，虚拟存储实现了节点间的隐式访问，系统还采用专用处理机/存储器接口提高存储频带，降低访存延迟。I/O DMA 通道和 ParaChannel I/O 阵列的节点直接连接使整个系统的 I/O ParaChannel 数最多有 1024，每个 ParaChannel 的数据传输率不低于 2.5GBytes/s，ParaChannel 的内在并行特征为冗余盘阵列的直接实现提供了支持。

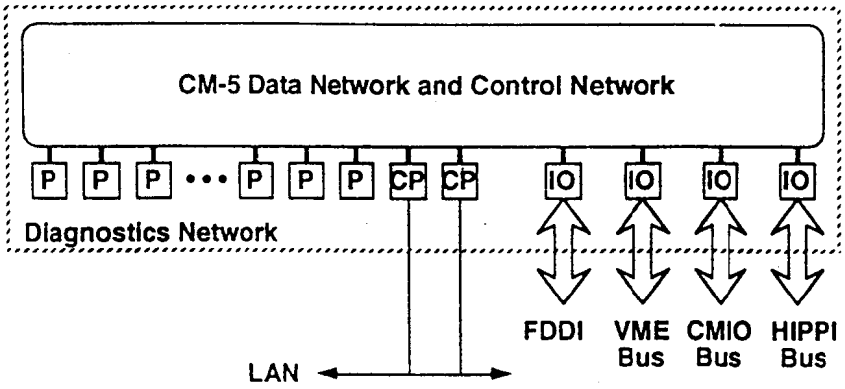

图 1.1 CM-5 系统配置图

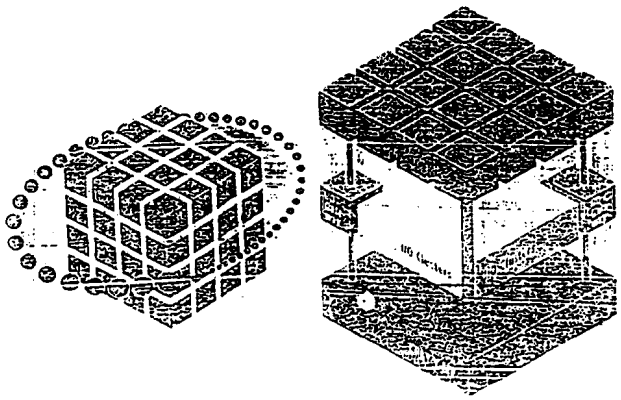

图 1.2 T3D MPP 结构

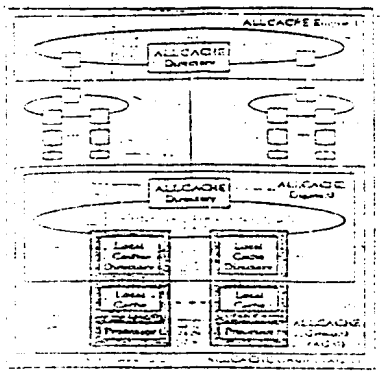

图 1.3 KSR-1 结构

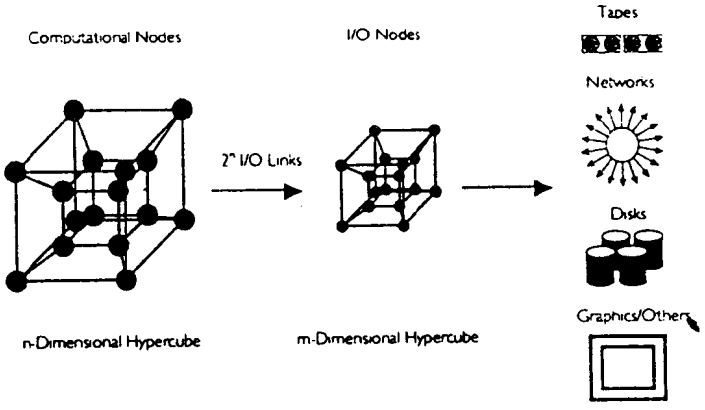

图 1.4　nCUBE3 结构

1.2　并行化研究的背景和意义

并行机结构的迅速发展并没有得到软件的相应支持，并行机系统的性能普遍没有得到充分发挥，据言，一台超级计算机的有效性能通常只有其峰值性能的 5—25%[Hwang 87]，而其中的大规模并行计算机性能的发挥就更低了，通常只能达到峰值的 1—5%[夏唐 93]。人们希望并行计算机能尽快地得到软件的有效支持，使它能够更高效地运行，能更方便地为人们所用。并行化的研究为此提供了一条很重要的途径。

在当前缺乏能被用户普遍接受的并行程序设计语言标准的情况下，开发并行化工具能有效地解决多种并行机结构间的代码可移植性问题，避免了用户为使用新的并行机结构重写其所有的应用程序。因为重写往往会带来新的程序错误和效率问题。这种代码的可移植性在目前并行机结构发展迅猛的形势下是非常需要的。

在程序开发、调试、维护、理解和高效运行方面，并行程序比串行程序困难得多。要使并行程序高效运行，就必须有效地利用目标并行机的结构特性(多功能部件，流水线，向量寄存器，cache，local memory，互连网络结构等)，然而大多数用户(尤其在应用界)并不希望也没有兴趣花费很多额外的时间去了解结构细节开发并行程序。他们更希望借助于并行化工具来免去或大大减轻他们在这方面的劳动。

各个应用领域都存在着大量的经过许多人长时间设计、使用和测试的大型程序。由于缺乏必要的文档，通常，从事并行化改写工作的程序员一般不能完全了解这些代码的全貌。改写过程又往往会引入大量的错误。利用并行化工具只需花费很少的人力就可以并行化现有程序使其能在并行计算机上得到正确而又高效的运行。

由此可见，并行化工具和环境是在并行机系统结构和应用软件之间架设了一座桥梁。一方面，在系统结构飞速发展的今天，它使得用户无需过多地受到这种变化的牵制；另一方面，即使面对一类稳定的并行系统结构，要想发挥机器的性能，往往需要针对结构细节来编程，这不符合计算机作为简单适用的工具的本意，因而提供使用户透明使用计算机结构的并行化工具是必要的。这也是并行化工具更加光明的应用前景。

并行化工具正在也必将越来越受到并行机(尤其是大规模并行机)系统软件研究人员的重视,成为系统软件不可缺少的组成部分。为此国际上投入了许多研究力量。但由于研究的复杂性和对上涉及用户编程语言对下涉及并行机系统结构所带来的多样性,工具的研究难度较大,问题很多,研究工作正方兴未艾。

我们通过广泛分析这方面的技术与工具的研究情况,探讨了其中的重要课题,并在一定程度上解决了其中一些难度很大的关键性问题。在下一节中,我们将尽可能详细而客观地给出并行化研究的历史介绍和现状述评,然后介绍我们的研究情况。

1.3 并行化研究的历史与现状

1.3.1 关键技术:数据相关性分析和程序重构

1. 数据相关性分析

有关数据相关性分析的工作早在60年代就已经开始[Bernstein 66],然而这方面成就突出的应该算是Kuck, Banerjee, Wolfe等人所做的工作[Kuck 78, Banerjee 76, Banerjee 79, Wolfe 82]。Banerjee于1976年在他短短的36页硕士论文中率先提出了用于判别循环中语句间相关关系的GCD测试法和界限测试法(后被称为Banerjee不等式测试法)。此后,Banerjee, Wolfe, Burke, Triolet, Callahan, Li等人又分别在他们的博士论文和后续发表的一些论文和著作中提出了方向向量[Wolfe 82],多级相关性测试[Burke 86]等一系列重要概念,讨论并给出了多维下标表达式情况的相关性求解方法[Banerjee 88, Li 89]和过程间相关性分析方法[Burke 86, Triolet 85, Callahan 87, Li 89, Balasundaram 90],进一步充实了早期的相关性分析工作,从而为循环结构的向量化和并行化奠定了很好的理论基础,也为开发过程间并行性创造了一定的条件。

相关性分析按精度分类可以分精确和近似两种方法,精确测试法给出的是相关关系存在的充要条件,所以采用精确测试法可以准确地判别相关关系的存在与否;而近似测试法判别的是存在相关的必要条件,采用近似测试法可以确定相关关系的不存在,但在不能证实不相关的情况下,为保证向量化和并行化的正确性,只能作出保守估计,假设相关关系的存在。

另外,相关性分析也可以按数组访问的下标线性度分成线 性下标相关性分析和非线性下标相关性分析两种,或根据分析范围的不同分过程内相关性分析和过程间相关性分析两种,下面我们分别介绍它们的研究现状。

(1) 线性下标情况

对于单层循环的一维线性数组访问,目前已经有有效的精确测试方法可以运用[Banerjee 79, Wolfe 87, Banerjee 88]。然而对于多层循环的一维线性数组访问尤其是多维线性数组访问,除部分特殊情况外[Barerjee 88],大多数相关性分析算法仍然只能给出近似的答案。

早期提出的一些方法[Wolfe 82, Burke 86, Girkar 88],为了提高效率,在处理多维线性数组访问的相关性问题上,或采用数组下标逐维测试方法或通过线性化方法将多维相关性分析问题转化为单维相关性问题。一般地说,在不出现耦合下标[Shen 90]的简单情况下,逐维测试法可以获得较为精确的结果,线性化方法则由于在丢番图方程中引入过多的变量,精度较差;

然而在另一方面，当出现耦合下标时，逐维测试法因孤立地对待数组的每一维下标，结果往往过于保守，线性化方法虽然能改善部分情况的测试精度，但多数情况下效果并不明显。

Banerjee 的多维 GCD 测试法[Banerjee 88]和 Z. Li 的 λ-多维联立测试法[Li 90]对相关性分析的精度有了一定的改善(根据 Li 的实验结果，应用 λ-联立测试法，从 Eispack 库中检测出的不存在相关的数组引用比逐维测试法增加 10.4%)，表现在多维 GCD 测试法能用于判别和求解同时满足所有丢番图方程的整数通解(不考虑约束条件)，λ-多维联立测试法可以判别同时满足所有丢番图方程和约束条件的实数解(不考虑整数解)。

从理论上来说，更为精确的相关性分析信息可以通过整数规划或线性规划来求得。近年来，在针对相关性分析的求解特性(大量小型方程组求解)，应用、改进 LP 和 IP 算法方面投入了较多的研究。Triolet 在他的相关性分析算法中[Triolet 85]使用了 Fourier-Motzkin 消元法(一种基于线性不等式成对比较的线性规划方法，可以获得实数解)。Wallace 的约束矩阵测试法[Wallace 88]采用了修改后的单纯形算法，Lu 和 Pugh 提出了用于相关性分析的整数规划算法[Lu 90, Pugh 91]。大多数算法使分析的精度有了不同程度的提高，但效率问题仍然是这类算法的关键问题。要获得一维多变量尤其是多维多变量数组访问情况下的精确相关性信息，算法往往具有指数时间复杂性[Goff 91, Pugh 91]。

我们认为精确而又高效的相关性分析算法应该是多种算法的有效合成，Maydan 和 Goff 等人在这方面的研究成果[Maydan 91, Goff 91]就是两个很好的例子。

(2) 非线性下标情况

和线性下标相对应，非线性下标指的是数组的下标是循环控制变量的非线性表达式。非线性下标可以部分地出现于数组下标的某些维，构成部分线性访问模式，也可以出现于数组下标的全部维，构成非线性访问模式。

目前大部分相关性测试方法仍然局限于对线性下标的分析。对于部分线性和非线性的访问模式可以通过使用重构技术、向前替换、归纳变量替换和常数传递等，部分地将其转化为线性下标，但根据沈志宇等人在 Illinois 大学对 6 个库程序的分析结果[Shen 90]，转换后的程序中部分线性和非线性的访问模式仍然占有全部数组访问的 13.45%和 33.95%。

近年来出现了一些符号处理方法[Lichnewsky 88, Polychronopoulos 90]，但它们或分析代价太大[Lichnewsky 88]或处理情况相对简单[Polychronopoulos 90]，效果并不理想。要获得精确的相关性分析结果还需要有更为有效的符号处理方法。

(3) 含过程调用情况

含过程调用情况下的相关性分析(即过程间相关性分析)，以开发调用过程和被调用过程中的各种潜在并行性为目的，包含有三个方面的内容：别名分析、过程间常数传递和过程访问信息计算。

1) 别名分析

使用别名分析可以提高相关性分析的精度。一般地说，别名分三类：最简单的别名关系由显式说明引起(FORTRAN 中的 EQUIVALENCE 语句和 C 中的 UNION 语句都具有这种功能)，我们称它为显式别名；另外可以通过复制参数产生别名(一个实参同时传给两个或多个不同的形参，或全局变量作为实参传给形参)，即所谓的参数别名；或由指针引起，被称作为指针别名。这里我们主要介绍别名参数的研究工作。

传统的别名分析方法以变量名作为基本分析单位，不考虑数组变量的下标细节，它们对非循环调用图的处理非常简单，通过遍历调用图的每个过程结点标出程序中的所有别名。对于循环调用图情况(即含递归调用)，Banning[Banning 78]提出了检测别名的迭代算法。Cooper[Cooper 85]又将别名分析划分为检测别名引入和追踪别名传递两部分，提出了别名传递的数据流分析框架。但由于忽视了数组变量的下标细节，这种变量名分析方法往往精度较差，不能满足并行化的要求。

2) 过程间常数传递

和别名分析一样，使用过程间常数传递可以提高被调用过程的相关性分析精度，开发被调用过程中更多的并行性。

简单的过程间常数传递可以通过过程内的局部常数传递[Wegman 85]和对检测出的不变形参的常量替换的反复迭代来实现，但这种方法的效率较低，迭代过程常常渗杂有许多不必要的工作。Callahan 的过程间常数传递方法[Callahan 86]通过引入跳跃函数 J_s^f (s 为调用点，f 为被调用过程的一个形参)，压缩过程体信息，将上述迭代过程转换成了对等式

$$f = \wedge_{s=P\to Q} J_s^f$$

的迭代求解，从而可以节省许多迭代时间，提高了常数传递效率。

3) 过程访问信息计算

相关性分析的一个重要组成部分是计算语句的 IN 和 OUT 集合。一般说来，在忽略别名影响和过程调用的情况下，IN 和 OUT 集合的计算较为简单。当语句包含过程调用时，编译无法直接确定所访问变量的集合，这时编译必须进行过程间访问分析，否则只能基于调用点所获得的信息作出保守假设。

访问信息计算的最简单方法是通过分析每个过程 P，确定对 P 的一次调用会引起对 P 的哪些参数的修改。这里包括了对 P 中直接修改的和 P 中通过 CALL 语句间接修改的所有参数的分析。如果我们用 REF(P)表示过程 P 的所有参数访问对集合(访问对由参数和参数的访问类型构成)；LREF(P)表示过程 P 直接访问的所有参数对集合；MAP(e,s)表示调用点 $e = P\to Q$ 处(P 调用 Q)形参到实参的映射函数，那么采用这种方法的计算可描述为

$$\mathrm{REF}(P)=\mathrm{LREF}(P)$$

$$\mathrm{REF}(P)=\mathrm{REF}(P)\cup[\cup_{e=P\to Q}\mathrm{MAP}(e,\mathrm{REF}(Q))]$$

对于非循环调用图计算过程比较简单，只要对调用图作深度优先遍历，对每个过程 P 分析其直接修改的和其孩子(已分析)修改的参数。对于循环调用图，整个计算过程需要迭代进行。虽然迭代过程总能终止，但由于 MAP 函数的出现，算法不够高效[Cooper 84]。

Cooper[Cooper 84,Cooper 88a]进一步将访问信息计算问题分成全局变量访问和形参访问两部分：

$$\mathrm{REF}(P)=\mathrm{LREF}^{+}(P)\cup[\cup_{e=P\to Q}\mathrm{GREF}(Q)]$$

$$\mathrm{LREF}^{+}(P)=\mathrm{LREF}(P)\cup[\cup_{e=P\to Q}\mathrm{MAP}(e,\mathrm{RMOD}(Q))]$$

其中 $\mathrm{LREF}^{+}(P)$ 表示 P 中直接访问的和通过传地址方式传递到其它过程被访问的参数对集合；RMOD(P)表示由 P 直接访问的或通过 P 中过程调用访问的 P 的形参；GREF(P)表示全局参数访问对集合。RMOD(P)可以通过计算 map 矩阵的自反传递闭包 map^{*} 来确定[Cooper 88a]，有效地避免了 MAP 函数的计算。此后 Cooper 又提出了 binding 图

概念[Cooper 88b]，简化了形参访问信息的计算，使简单访问信息计算问题在线性时间内能得到求解。然而，Cooper 方法的不足在于它和其它简单访问信息计算方法一样，仅仅计算了变量名访问信息。这对于数组来说是不够的，要确定更详细的数组访问信息，需要有更为精确的数组区域访问分析。

Li 的原子和原子映象法[Li 88,Li 89]所做的区域分析最为全面，它不仅记录了过程中每一数组访问的每一线性下标表达式，也保存了所有循环变量的界限(用循环不变量和外层循环变量表示)。存储这些信息的数据结构是原子和原子映象。原子包含了数组访问的局部信息，每个原子由一个二维数组 A 来表示，A 的第 i 行第 j 列(A_{ij})包含了第 i 维下标表达式中第 j 个循环变量的系数(循环变量按最外层到最内层依次排列)，A_{i0}代表常数项。另外，对应数组的每一行还有一位标志，当且仅当该维表达式是线性时，置该标志。原子映象在原子的基础上又增加了每个循环变量的界限信息。采用原子和原子映象法进行过程间相关性分析，首先将被调用过程中每个数组访问的原子映象转换成调用点处的原子映象，然后使用标准的相关性测试方法对这些原子映象做相关性分析。由于记录的这些信息本质上等价于源代码，这种方法获得的访问信息最为精确，但问题是无法对访问信息作合并操作，子程序中每个数组的每次访问都需要对应有一个原子和原子映象，存储空间消耗很大，这个问题在其它的一些方法中是以信息精度为代价来解决的。

Burke 和 Cytron 的线性化方法[Burke 86]将所有不同的数组访问转化为对同一单维数组不同区域的访问，保存的是线性化后的数组访问信息和循环界限信息。如，语句

$$A(i,j)=(A(i,j-1)+A(i,j+1)+A(i-1,j)+A(i+1,j))/4$$

中的 A 数组访问将被表示为 $\mathrm{MEM}[N\times(i-1)+j+\mathrm{INIT}_A]$，$\mathrm{MEM}[N\times(i-1)+(j-1)+\mathrm{INIT}_A]$，$\mathrm{MEM}[N\times(i-1)+(j+1)+\mathrm{INIT}_A]$，$\mathrm{MEM}[N\times(i-2)+j+\mathrm{INIT}_A]$和 $\mathrm{MEM}[N\times i+j+\mathrm{INIT}_A]$，这里 INIT_A 指数组 A 在存储器中的启始地址。线性化后的访问信息可以单独保存，也可以合并后保存(两个访问区域的合并是取最小下限和最大上限)。单独保存获得的访问信息精度较高，但存在和 Li 方法同样的问题，合并可以节省存储空间，但结果往往过于保守。

Triolet 提议用线性不等式表示的 K 维凸形空间来描述一个数组的访问区域[Triolet 85,Triolet 86]，其算法分为两步：1)用自顶向下的分析方法计算所有过程的执行环境；2)用自底向上的分析方法计算所有过程的访问区域。精确地说，过程的执行环境定义了过程中每一局部变量的取值范围是一个线性不等式集合；数组的访问区域由二元序偶$<A,\sigma>$表示，其中，A 是区域所属的数组名字，σ 为形如 $l\leqslant\varphi_k\leqslant u$ 的线性不等式集合，φ_k 对应 A 中所访问元素在第 k 维的变化范围。数组的局部访问区域是单个语句访问的区域(其中，不等式可能含局部变量)，数组的全局区域是整个过程访问的区域，由过程中所有局部区域合并而成(不含任何局部变量)。过程调用语句的数组访问区域由被调用过程的数组全局区域在调用点处的执行环境下转换得到。相关性测试就是测试表示访问区域和执行环境的线性不等式组的一致性。这里，对线性不等式组的求解代价很高，理论上的时空复杂性是指数级的[Banerjee 86]。另外，一系列的合并操作也增加了结果区域的复杂性(可能产生和合并的原始访问数同样多的不等式)。这是 Triolet 不等式方法的不足之处。

如果说前面的方法是试图以效率为代价换取最高精度的话，那么 Callahan[Callahan 87]提出的受限规则域描述(RRSD)方法实际上采用的是一种以损失部分精度来换取效率的策

略。在[Callahan 87]里,一个 RRSD 被定义为由一个数组名和一组表达式组成,每个表达式对应数组的一维,记录了数组访问空间的这一维中被访问的元素。表达式的取值有三种(以第 j 维为例):1) $i_j=\perp$ 表示任意值;2) $i_j=\alpha$ 表示不变量;3) $i_j=\alpha\pm i_k(k\neq j)$ 表示相对位移量。使用 RRSD 方法可以描述数组空间的点、行、对角线、平面和其它子区域。区域间的合并操作是逐维进行的。两个相同常数 α 的合并结果是常数 α;两个相同表达式 $\alpha\pm i_k$ 的合并结果是表达式 $\alpha\pm i_k$;其它情况的合并结果取任意值 $\perp$。相关性分析表现为对区域的求交,如交集非空,则存在相关;否则不存在相关。两区域(r_1 和 r_2)间的相交关系由下列条件决定:1) 相应元素至少有一个具有 $\perp$ 形式;2) 一个具有 $\alpha\pm i_k$ 形式,另一个不是 i_k 的函数;3) 二者都具有 $\alpha+i_k$ 或 $\alpha-i_k$ 形式且相同的常数 α;4) 二者为相同的常数 α。如果 r_1,r_2 中至少有一对相应元素不满足上述任何条件,则两规则域互不相交,否则交集非空。RRSD 方法具有高效的合并、求交操作,但所能描述的区域很有限,任何过程只要访问了任意两个不同行、列或对角线都被认为访问了整个数组。

为了更精确地描述数组访问的区域,又能执行高效的合并和求交操作,Balasundaram 提出了数据访问描述(DAD)方法[Balasundaram 90]。DAD 用简化区域来描述一个数组的被访问部分,简化区域由数组访问空间的正交界和对角界表示。正交界是每一维的上、下界,对角界是每两维间的和、差界限。n 维数组访问空间的简化区域以如下不等式形式存储:

①正交边界	$\alpha\leqslant x_i\leqslant\beta$,	$1\leqslant i\leqslant n$
②正对角边界	$\alpha\leqslant x_i-x_j\leqslant\beta$,	$1\leqslant i,j\leqslant n,i\neq j$
③反对角边界	$\alpha\leqslant x_i+x_j\leqslant\beta$,	$1\leqslant i,j\leqslant n,i\neq j$

简化区域的合并是取每一对对应正交界或对角界的最小下界和最大上界,简化区域间的相交关系通过比较对应边界的上、下界来判别。如果存在一对对应的正交边界或对角边界,其中一个的上界小于另一个的下界,则区域不相交,否则相交。和其它方法相比,DAD 方法具有合并、求交操作简单的优点,而且相关性计算也比较简单,所描述的简化区域也非常一般,可以包括各种规则形状,精度上要比 RRSD 方法更好一些。

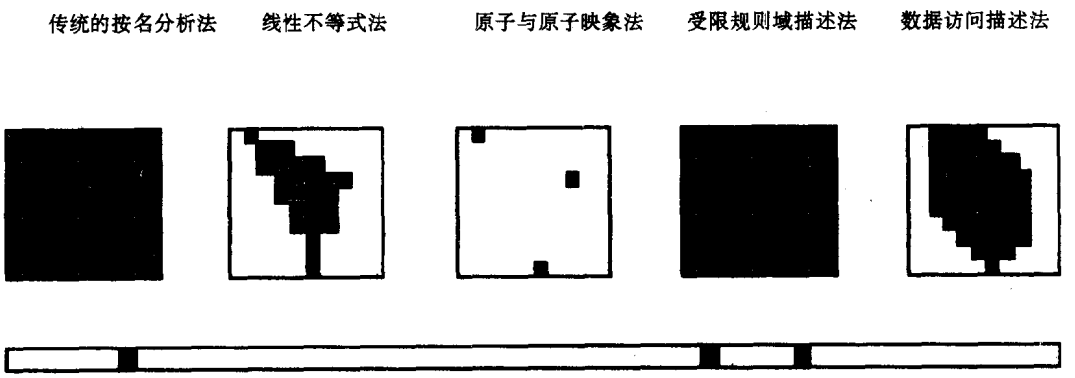

图 1.5 不同方法对数组访问 $A[1,2]$, $A[4,8]$, $A[10,6]$的不同描述

图 1.5 给出了上述各种访问信息计算方法对数组元素访问的不同描述。从中我们不难发现，Li 的原子、原子映象法和 Burke 的线性化方法最为精确(这里线性化方法没有做合并区域操作)，Triolet 的线性不等式方法和 Balasundaram 的 DAD 方法其次，Callahan 的 RRSD 方法在这种情况下和传统的按名分析方法一样不精确。

2. *程序重构*(program restructuring)

以相关性分析为基础，对程序作语义等价的(自动或手工)重构(也称改写 rewriting[Lamport 74]或转换 transformation[Kuck 81])提高程序在目标机系统上的实际运行性能是程序并行化和向量化的最终目的。

向量化程序重构经过多年的研究目前已基本成熟，并行化程序重构却并非如此。和向量化相比，一方面，并行化程序重构需要面对更加复杂多样的程序结构，以更为复杂的全局数据流分析，过程内、过程间相关性分析为基础，开发更多、更大粒度的并行性；另一方面并行化程序重构需要更多地考虑并行机的各种结构和机器特性，使目标机性能能得到充分的发挥。所以程序的并行化改写要比程序的向量化改写更为复杂。目前，这方面的工作主要还停留在对单个重构技术的研究上，具体表现在以下几个方面(这里我们仅例举一部分)：

(1) 消除相关

属于这类重构技术的主要有：私有化(privatization)，标量扩展(scalar expansion)，结点分割(node splitting)，下标集分割(index set splitting)和循环剥离(loop peeling)等，它们或引入额外的存储空间，或分割循环下标集来消除或打开输出相关、反相关和跨迭代相关，促进其它重构技术的应用和并行性的开发。

· **私有化**[Eigenmann 91a] 一个数组或标量变量使该变量局部于每个循环迭代，用于消除跨迭代相关。

· **标量扩展**[Kuck 81, Padua 86] 将一个标量变量转换为一个一维数组，用于消除可能阻止并行性开发的输出相关和反相关。

· **结点分割**[Kuck 81, Polychronopoulos 88] 主要用于存在含反相关的相关圈的场合，通过将一反相关的源复制到一个新引入的临时数组，并换名相关源为这个新数组，将圈打开，促成后续的循环分布开发出循环中的并行性。

· **下标集分割**[Banerjee 79, Wolfe 82] 将一个循环的迭代空间分成两个相邻的部分，用于打开相关圈或打开跨某一迭代的相关。

· **循环剥离**[Zima 90] 将循环的前 k 个迭代或后 k 个迭代分离出来，可用于打开和循环前 k 个迭代或后 k 个迭代有关的相关。

(2) 调整执行序开发并行性

这类重构主要通过改变迭代内或迭代间的语句执行序来开发或增加循环级并行性，或优化数据局部性。属于这类重构技术的有：循环分布(loop distribution)，循环交换(loop interchange)，循环扭曲(loop skewing)，循环熔合(loop fusion)，循环对准(loop alignment)，周期压缩(cycle shrinking)和语句交换(statement interchange)等。

· **循环分布**[Kuck 78, Kennedy 90] 通过划分循环中的语句将一个循环转变成为多个具有相同循环头的循环，用于从那些必须串行执行的语句中抽取出可以并行执行的语句。

·**循环交换**[Wolfe 82, Allen 84a] 将两个相邻的循环进行交换，改变迭代空间的遍历次序，可用于开发并行性，调整并行循环的粒度和优化数据局部性。

·**循环扭曲**[Wolfe 86b, Wolfe 89a] 以开发波前并行性为目的，通过"扭曲"迭代空间改变内层循环的相关距离，使所有相关在循环交换后都能转变为外层循环携带相关，从而实现内层循环并行化。循环扭曲本身不能直接开发出任何并行性，它的意义在于辅助循环交换完成并行性开发。

·**循环熔合**[Kuck 81] 是循环分布的逆变换，它通过熔合两个相邻的循环来增加并行区域的粒度和提高数据的重用性。

·**循环对准**[Callahan 87] 通过将一个或多个语句实例从一个迭代移到另一个迭代使外层循环携带相关转化为内层循环携带相关或循环独立相关。

·**周期压缩**[Polychronopoulos 88] 通常用于存在流相关圈(仅由流相关组成的圈)的循环结构，抽取出隐含在其中的并行性。转换后的循环结构由外层串行循环和内层并行循环组成。周期压缩只有当所有相关距离都大于1时才能开发出并行性，这时的折减因子 $\lambda>1$。

·**语句交换**[Allen 83, Callanhan 87] 或更一般地，语句重排序(statement reordering)，改变循环中的语句序。可以用来将向后相关(backward dependence)转变为向前相关(forward dependence)促成循环分布转换，也可以用来优化 DOACROSS 循环[polychronopoulos 88]。

(3) 优化多级存储

这类重构的目的是调整一个循环在计算和存储访问间的平衡，更好地利用多级存储和功能流水线。属于这类重构技术的有：条化(strip mining)，标量置换(scalar replacement)，循环展开(loop unrolling)和展开与合并(unroll and jam)等。

·**条化**[Loveman 77, Padua 86] 将步距为1的一个循环转换为一个由步距大于1的外层循环和完成步内迭代的内层循环组成的嵌套循环。它可以用来开发并行性(类似于周期压缩)，也可以用来优化向量寄存器和 cache 的使用(根据向量寄存器长度和 cache 容量决定新步距的大小)。对于多重嵌套循环，它还可以和循环交换一起使用，调整迭代执行序，增加数据访问的局部性和数据的可重用性。和条化类似的重构技术还有块化(blocking)([Kuck 84]中的块化等同于条化，[Lam 91]中的块化实际上是条化加循环交换)，迭代空间片化(tiling)[Wolfe 89b, Wolfe 89c](片化是条化加循环交换)和[Wolfe 90a]中的 sectioning 和 combing(前者等同于条化，后者等同于片化)等。

·**标量置换**[Callahan 90] 将具有一致相关的数组访问替换成标量临时变量以便分配到寄存器。它通过减少要求的存储访问数来改善程序的性能。

·**循环展开**[Zima 90] 通过展开一个循环的循环体，减少循环开销，增加标量替换的潜在机会，减少存储器和寄存器间的数据传输。

·**展开与合并**[Callahan 90] 通过展开嵌套循环中一个外层循环的循环体和合并所得到的内层循环，可以增加标量替换的潜在机会，改善流水线利用率(在出现递归时)，改善 cache 性能，提高 cache 命中率。

1.3.2 并行化系统

一些典型的并行化系统如表1.2所示。

最早的向量化和并行化工具[Kuck 84, Padua 80](以向量化为主)PARAFRASE 由 Illinois 大