分布阵列处理机(DAP)上的并行算法

程锦松 著

中国科学技术大学出版社

分布阵列处理机(DAP)上的并行算法

分布阵列处理机(DAP)上的并行算法

程锦松 著

中国科学技术大学出版社
1995·合肥

内　容　简　介

本书前三章介绍由几千个简单的位处理单元(PE)构成的分布阵列处理机(DAP)的概况。第四至第九章分别介绍DAP上的排序、矩阵演算与线性方程组求解、数值函数计算、FFT与FNT、多项式求根和伪随机数产生的并行算法。第十至十三章分别介绍在DAP上并行处理语音识别、手写字符识别、芯片等设计自动化中的路径选择和不可压缩流体的Navier-Stokes方程组的求解等四个问题。

本书可供高等院校从事计算机算法与体系结构研究人员阅读;也可作为计算机算法与体系结构研究方向的硕士研究生的参考书。

此书得到安徽大学"211工程"学术专著出版基金资助

图书在版编目(CIP)数据

分布阵列处理机(DAP)上的并行算法/程锦松　著
—合肥:中国科学技术大学出版社,1995年8月
ISBN 7-312-00627-2

Ⅰ　分布阵列处理机
Ⅱ　程锦松
Ⅲ　①并行算法　②分布阵列处理机(DAP)
Ⅳ　TP

中国科学技术大学出版社出版发行
(安徽省合肥市金寨路96号,邮编:230026)
中国科学技术大学印刷厂印刷
全国新华书店经销

开本:787×1092/16　印张:6.75　字数:170千
1995年8月第1版　1995年8月第1次印刷
印数:1—2000册
ISBN 7-312-00627-2/TP·106　定价:7.60元

前　言

分布阵列处理机(Distributed Array Processor，DAP)是一种SIMD型的并行计算机，它由几千个处理单元(PE)组成二维阵列结构，但是DAP不同于著名的ILLIAC-Ⅳ阵列机，ILLIAC-Ⅳ的PE是一台功能完善的64位处理机，而DAP的PE只是一个简单的位处理器。由这些廉价和灵活的PE构成的DAP却可达到很高的性能，同时DAP的体系结构也很适合VLSI的组装，因此，目前已得到广泛应用的DAP将随着计算机技术和VLSI技术的不断进步而得到进一步的发展。

为了能使DAP的性能在实际使用时得到充分发挥，必须深入研究适合DAP特点的并行算法，作者曾对阵列机上的并行算法作过一些研究探讨工作，本书是结合作者的工作和国外的文献撰写而成的。

全书共分十三章、前三章是DAP的概况介绍。中间六章是探讨几类典型问题在DAP上的并行算法，它们是排序、矩阵演算与线性方程组求解、数值函数计算、FFT与FNT、多项式求根和伪随机数产生的并行算法，最后四章是分别介绍如何用DAP来解决目前人们感兴趣的四个问题：语音识别、手写字符识别、芯片等设计自动化中的路径选择和不可压缩流体的Navier-Stokes方程组的求解。

本书可供高等院校计算机及相关专业的师生、从事计算机算法与体系结构研究以及计算机应用的科技人员阅读参考；也可作为计算机算法与体系结构研究方向的硕士研究生的参考教材。

本书在撰写与出版过程中，得到了安徽大学计算机系和科研处的支持，还得到了合肥工业大学张奠成教授、中国科技大学陈国良教授、赵保华副教授等同志的热情鼓励与帮助，在此一并深表感谢。

由于作者水平有限，错误与缺点在所难免，欢迎读者批评指正。

程锦松

1994年冬于安徽大学

目　次

第一章 引 论

1.1 DAP 的发展历史

分布阵列处理机(Distributed Array Processor,DAP)是 ICL(International Computers Limited)研究和高级开发中心(Research and Advanced Development Centre)于 1972 年研制成功的一种单指令流多数据流(SIMD)机器.DAP 的结构是多台处理机互连成的一个矩形网络,这种结构适合于气象预报等多种应用类型的计算.分布阵列处理机是通过存储模块来分散的.DAP 和向量流水线技术设计成的处理阵列结构不同.

DAP 由多个处理机单元 PE(Processor Element)组成,不同于著名的 SIMD 型的 ILLIAC-Ⅳ的 PE.DAP 的 PE 仅是位处理单元,这种结构具有很大的灵活性,有很高的性能.ILLIAC-Ⅳ的 PE 是个功能完善的 64 位处理机.

第一个 DAP 试验模型(Pilot DAP)是由 1024 个 PE 排成 32×32 的二维阵列构成,它由主控制器 MCU(Master Control Unit)控制工作.MCU 尽量保持简单,在阵列上操作用向量模式.MCU 包含有适当寻址和实现循环等硬件设施.在软件方面,开发了 DAP-Fortran(亦称 Fortran-Plus)语言和汇编语言 APAL.

第一个 DAP 产品是由 4096 个 PE 排成 64×64 的二维阵列构成,每个 PE 具有 4K 位的局部存储器.MCU 增加了一些功能和一些阵列处理的指令.对 DAP-Fortran 的语法做了一些重要修改,以提供较先进的软件诊断机制.

随着 VLSI 技术的发展,推出了第二代的 DAP(DAP-2).DAP-2 有两种型号:一种是面向公共通用的,它具有价廉、大容量但速度较慢的存储器;另一种是面向军方的,其容量小、快速但却是昂贵的存储器.真正生产的是第二种型号的 DAP-2.有时 DAP-2 称为 MiniDAP 或 MilDAP.

DAP-2 又返回到 1024 个 PE 的阵列,利用门阵到技术在一个片子上实现 16 个 PE 的逻辑.它提供了一种新的主机连接器 HCU(Host Connection Unit).HCU 包含有一台微处理机,它主要用于管理 DAP 与主机之间的接口和通道,资源分配以及 DAP 系统内的调度,DAP-2 为阵列存储器和外部链接提供了一种高速类 DMA(DMA-like)接口.

后来又推出了 DAP500,它可以适配多种工作站,例如,可以通过标准的 SCSI 或 VME 总线与 Sun 工作站或 VAX 工作站连接.由于 VLSI 技术的更进一步发展,使得在一片半定制的 VLSI 片子上可实现 64 个 PE,并且存储器规模和时钟周期都得到了提高.DAP500 也是由 1024 个 PE 构成,MCU 为一台 32 位处理机.在软件方面,根据早期的 DAP 系统的经验,改进了库例行程序,增强了可用性.

1.2 DAP 上的程序设计

在 DAP 上解题必须采用适合 DAP 结构的算法和程序设计语言. DAP-Fortran 是在常规的 Fortran 中引入了数组处理操作,这些操作能在 DAP 上有效地并行实现,故 DAP 系统采用 DAP-Fortran.

由于 DAP 的基本处理单元是简单的位处理器,故在 DAP 基本软件中提供了算术运算的函数.

在 DAP 上设计算法和编程还必须注意到的问题有:数据在 DAP 上如何安排组织;怎样进行数据传送和移动;如何进行路径选择(Routing)等.

1.3 DAP 的应用

人们经常提出这样的问题:哪些问题适合用 DAP? 原则上讲,对那些计算机受 CPU 限制而又较多地使用全部有用的处理资源的问题,均适合使用 DAP. 不过具有下列特征的问题更适合使用 DAP.

- 大量的逻辑、字符或整数操作;
- 在较大的结构化数据集合上操作;
- 可以用特殊数据表示的应用问题,例如很短或很长的数值格式,定点数表示和字符表示等.

第二章　DAP的系统结构

在第一章我们已提到有几种类型的DAP，但就其基本结构来说是类似的，故在这一章我们基本上以Pilot DAP为蓝本来讨论DAP的系统结构([1]-[2]).

2.1　概述

DAP是一种SIMD计算机，对许多大型计算有很高的速度，它由排成一个二维阵列的几千个处理单元(PE)组成，PE是简单的位处理器，但用多个PE构成的DAP可以达到很高的性能．各个PE执行由主控制器MCU广播发送的指令流．

PE阵列的规模可根据所需达到的处理能力的要求来定．阵列一般是方形的，它的边长对应标准存储总线宽度．每个PE带有一个几K位的快速随机存储器．存储器与PE之间的快速并行传送与高处理速度相适应．

PE存储器的全体形成一个普通计算机的标准模块．因此在主机(Host)与DAP之间不必分开传送数据．DAP的处理是通过一个简单接口在主机控制下进行的．将DAP作为通用系统的一个部分有两个好处：一是可以方便的访问系统，特别是操作系统、语言与输入—输出设施；另一个是允许用户获取DAP处理的优点．

Pilot DAP由按32×32排列的1024个PE构成，每个PE有1K(可扩展到2K)位的双极存储器．该系统有89块印刷电路板，其中多数是“阵列”板，每块板有16引脚的TTL集成电路形成的16个PE的逻辑和存储器．总功耗(不包括电扇和电流)小于1.5kW.

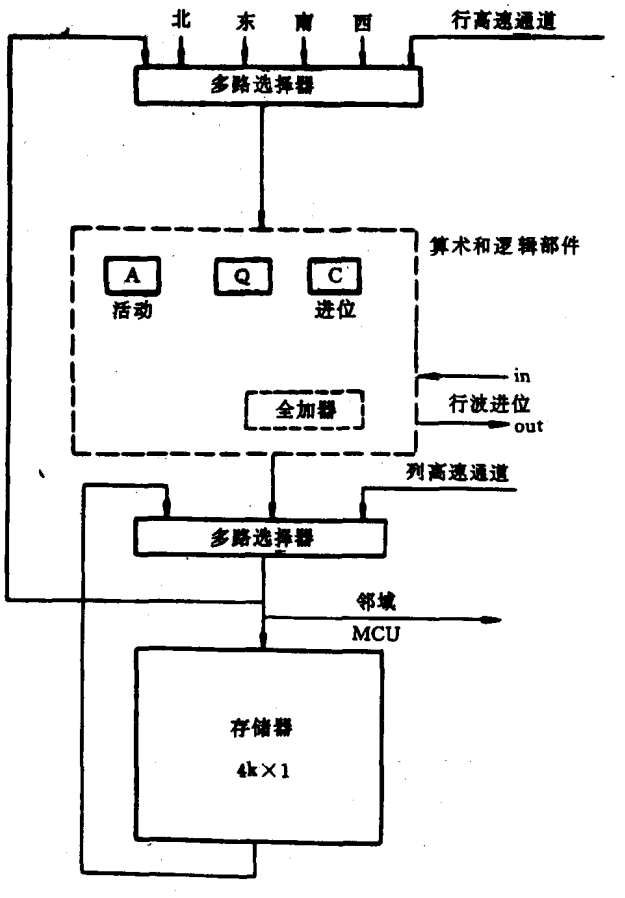

图2.1　处理单元

2.2　处理单元PE

图2.1示出了一个PE的实质性特点，为了清楚起见，省去了一些控制和数据通路．所有数据通路是一位宽的．

图2.1中上面的多路选择器选取算术和逻辑部件(ALU)的输入，这些输入可以是PE自

己的输出或者是从与该 PE 相邻的北、东、南、西面的 PE 来的输出．图 2.1 中下面的多路选择器选取 PE 的输出，这些输出可以是 ALU 的输出或者是存储器的输出并且可以送到存储器、其它的 PE 或 MCU．这两个多路选择器还允许选取从 MCU 沿行或沿列的高速通道广播送来的数据输入．

ALU 有三个 1 位寄存器(A，Q 和 C)和一个 1 位全加器．低级软件(APAL 程序)可以充分发挥 ALU 使用的灵活性．

寄存器 A 提供"活动"控制，某些存储写入操作仅当该寄存器为真时有效．这样就可以利用寄存器 A 的控制作用，使得某种功能仅施于阵列中被选的元素上．寄存器 A 具有逻辑与(AND)功能，这可使条件很快组合出来，还可用来实现一般逻辑功能．

寄存器 Q 为 1 位累加器，寄存器 C 为进位寄存器．加法器将 Q，C 和 ALU 的输入相加，而其和输出可以写回到 Q，其进位输出写回到 C.

图 2.1 中所示的与每个 PE 相关联的 4K 位的存储器仅能由这个 PE 来进行存取．

寄存器 Q 还用于 PE 间的通信，按指令可从一个寄存器 Q 移动数据到邻域的寄存器 Q.

2.3 主控制器 MCU

图 2.2 是 MCU 的框图，它类似于一台小型计算机的指令顺序和控制部分．一条阵列指令的实现包含两个阶段：第一阶段是取指令阶段，指令从阵列存储器中取出；第二阶段是执行阶段，适当的控制信号在阵列上进行广播发送．每一阶段使用一个基本机器周期．

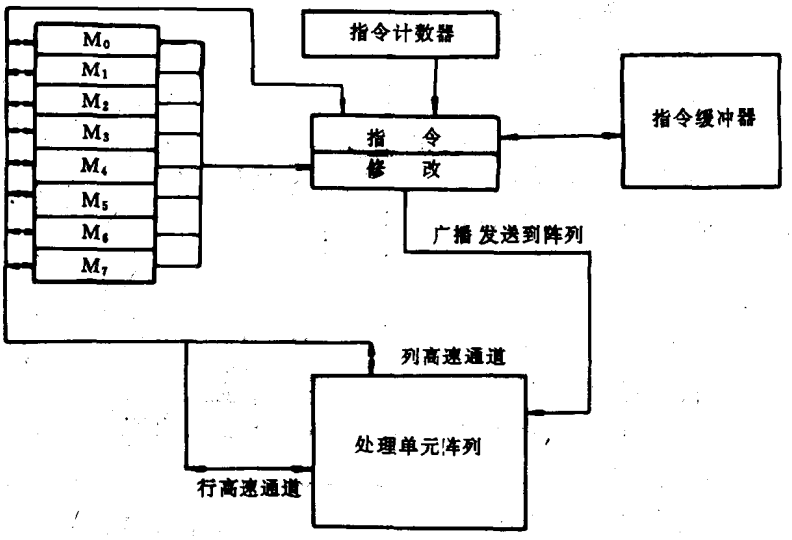

图 2.2 主控制器 MCU

每一条指令有 32 位，沿一行 PE 存放且相继指令是存放在相继的行(或部分行)中，使得代码均匀分布在 PE 存储之间．需要强调指出的是：指令只能由 MCU 解释；PE 不能识别指令本身．

MCU 中的指令缓冲器可存储用低级编码明确指定的循环(在 Pilot DAP 上可多至 15 条指令)，故在首次运行循环之前，这指令组仅需一次取出．在一个字的相继位上操作时，地址可自动地步进．

有八个通用寄存器，M0 到 M7，其长度等于 PE 阵列的边长．这些寄存器在指令取出或执

行中用于保存数据或地址．寄存器中的内容可以沿行或列高速通道送出到阵列或从阵列送入．

提供了正常的控制转移指令 GOTO,LINK 和 EXIT 以及对地址进行算术运算的指令．MCU 还包含有对主机的存储和控制接口设施．当 DAP 处理时,主机仍可通过周期挪用来访问存储器．

图 2.3 说明了行高速通路、列高速通路与 PE 矩阵的连接方式．图 2.4 是示出了将 DAP 看作一种三维(3D)的存储阵列,在二维(2D)处理器阵列下面垂直地安排了一个 4K 位的存储模块．MCU 提供了一系列在 3D 存储阵列与 1D(一维)寄存器之间的通信设施．

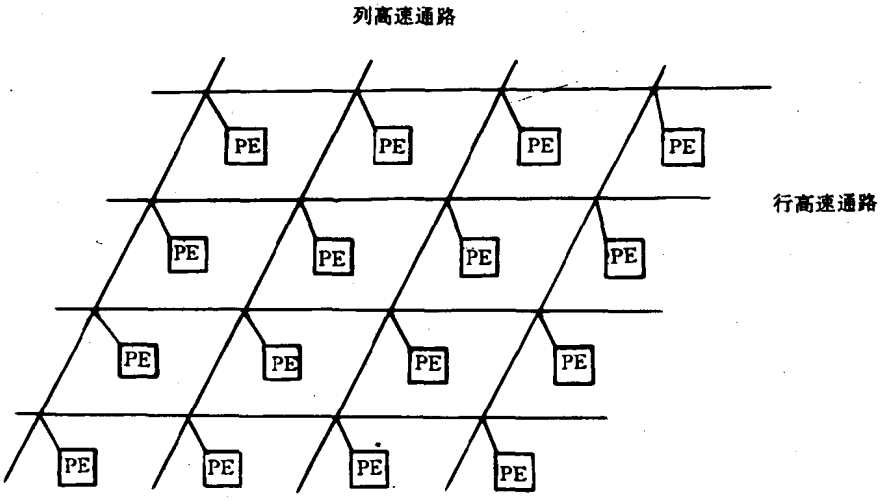

图 2.3 PE 互连

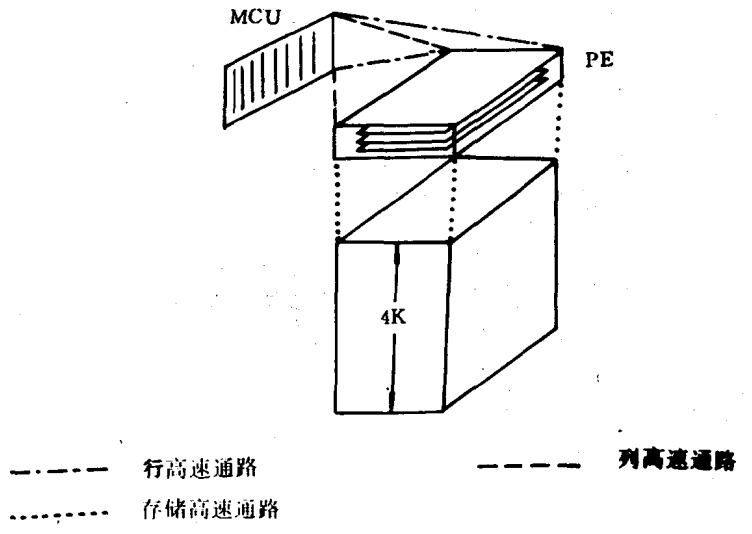

图 2.4 DAP 概貌

2.4 输入/输出

一般的超级计算机系统用一台标准的计算机作为其'前端'.DAP 系统用一台 ICL2900 机器作为其前端处理机,且配置 DAP 作为前端主机的一个存储模块(见图 2.5),从而省去了一般系统中有的数据通道.

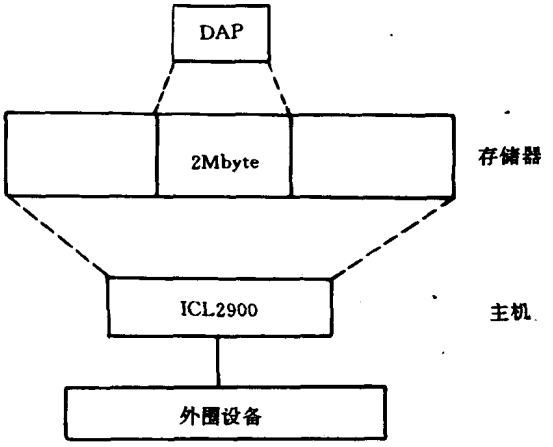

图 2.5 DAP 与主机

DAP 的每个 PE 有一个 4096 位的存储器,故在 64×64DAP 上总的存储器是 4096×64×64=2Mbyte.DAP 的存储器作为 ICL2900 主机的存储器的一个存储模块.主机和 DAP 处理单元均可存取这存储模块中的数据.主机从外设读入的数据送入这个模块或将该模块内的数据输出到外设.

主机装载程序到 DAP 且启动 DAP 进行处理.图 2.5 示出了主机 2900 和 DAP 的 MCU 两者均要访问的存储空间.

2.5 数据组织格式

64×64DAP 的存储器可以看作是 64×64×4096 位的三维结构.DAP 硬件没有对任何特殊数值格式做算术运算的内部设施,因此汇编级程序员根据需要可自由地选用数据格式,例如,可选用各种位数的浮点数或整数.运算时间取决于操作数据的长度,操作的复杂程度和程序员的技巧.

输入到 DAP 的数据通常是主机可识别的格式,因此要得到各种不同的格式将必须进行转换.格式转换是在 DAP 内完成的.

由于 DAP 存储器的三维结构,则数据在存储器中有两种存放方式:"垂直"方式和"水平"方式.图 2.6 给出了在 4×4DAP 上的数据存放格式.

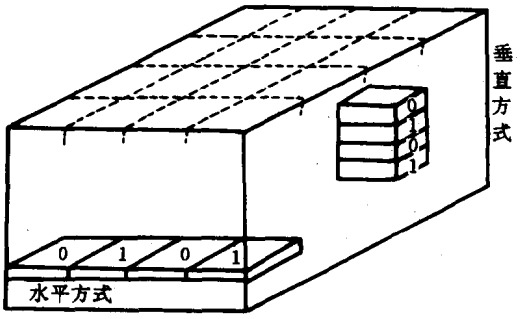

图 2.6 数据存储模式

当数据按垂直方式存放时,则所有位是存放在一个处理器的存储器中,而按水平方式存放时,则是分开存在几个处理器(在一行或一列中)的存储器中.用垂直方式,在 64×64×DAP 上,4096 个数可同时处理,当然一次处理 1 位.用水平方式,有 64 个 64 位的数可同时处理,其操作是在所有 64 位上同时进行.

至于哪种存储方式好,不能简单一概而论.一般情况下,垂直方式适合于有较大并行度的算法;水平方式对仅有低并行度的算法较合适.

第三章　DAP的软件

3.1　软件结构

如第二章所述，可以把DAP作为正规的主存储器，故在主机(2900)上使用标准设施运行的任一应用程序将运行在2900/DAP配置系统上．但是这样应用没有发挥DAP的性能，是不可取的．为了适合DAP的使用，必须用一种新的语言，DAP-Fortran(见3.2节)．这种语言可方便的描述DAP的并行处理．

在一台N×N DAP系统上，可以有几种不同的处理方式、存储方式以及并行程度，如下列表3.1所示．

要有效地使用DAP系统的关键在于：在任何情况下，只要改变方式的代价不是主要的，就使处理方式适合于所选择的并行度．

表　3.1

处理方式	存储方式	数据并行性
主机-标量	水平	1
DAP-标量	水平	1
DAP-矢量	水平	N
DAP-矩阵	垂直	N^2

在DAP上运行的一个程序由一个标准的Fortran程序和若干个DAP-Fortran子程序组成．DAP-Fortran源程序经一个特殊的编译器编译后产生出适于DAP执行的代码．可将DAP-Fortran模块与用任何一种常规语言编写的模块组合在一起．

通常在DAP上运行的任一应用程序有图3.1所示的结构．其中DAP-Fortran模块实现处理计算任务，2900模块处理输入/输出和整个控制任务．所有数据是保存在Fortran COMMON块中，这些公共块是驻留在DAP的存储器中，它们可以由2900和DAP两者来访问，没有任何数据传送．

3.2　DAP-Fortran

3.2.1　一般原理

DAP-Fortran[3]基本上是对熟悉的Fortran的语法的一种扩充．但是，在很多方面，DAP-Fortran是比Fortran更高级的语言，它允许用户用一种较接近他的真正数学记号的方法表示他的问题．由于在DAP硬件中提供的设施支持DAP-Fortran语言中的设施，故DAP

系统在运行 Fortran 时是非常有效的．因此用户不必去使用低级的汇编语言．

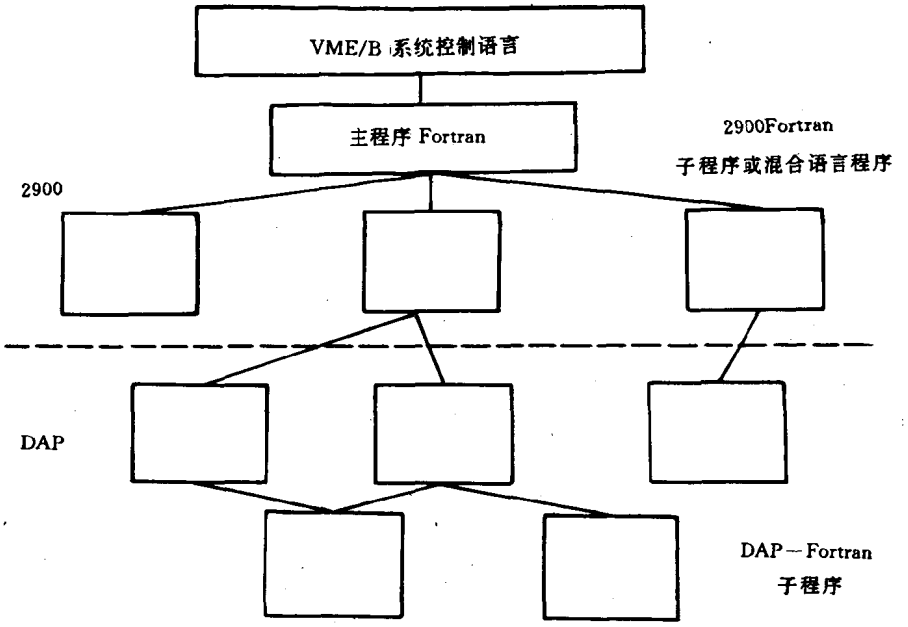

图 3.1　DAP 软件结构

DAP-Fortran 与 Fortran 之间的主要差别是能将整个数组作为一个单项．在 Fortran 中唯一的计算单元是一个简单的标量数据项，而数组是作为标量的集合．DAP-Fortran 识别三种基本单元：标量、向量和矩阵．一个标量项真正对应一个 Fortran 标量．向量和矩阵分别是 N 和 N×N 项的数组，这里的项可以作为一个各个别项的集合或更有效地作为整个数组．所有三种数据类型可以在表达式和赋值语句中使用．

例如，两个 N×N 元素矩阵同时相加的典型的 Fortran 程序为：

```
      DO  1  I=1,N
      DO  1  J=1,N
1     A(I,J)=B(I,J)+C(I,J)
```

这个程序用 DAP-Fortran 来编写则可简化为：

$$A = B + C$$

这里 A，B 和 C 已事先说明为矩阵．在 DAP 上计算和赋值将按并行方式在所有元素上同时实现．

另一种扩充是：取向量和矩阵作为基本算术函数的自变量，而返回的结果是一个向量或矩阵．

例如，要求对 N^2 个 a，b，c 的值解下列方程：

$$ax^2 + bx + c = 0$$

则解可简写为：

```
X1 = (－B + SQRT(B**2 － 4*A*(C))/(2*A)
X2 = (－B － SQRT(B**2 － 4*A*C))/(2*A)
```

这里 X1，X2，A，B 和 C 全是矩阵．

B1

3.2.2 数据方式

标准 Fortran 仅有标量方式变量，而 DAP-Fortran 除标量外，还有两种新的数据方式 VECTOR(向量)和 MATRIX(矩阵). 变量方式的定义用一种扩充的类型说明语句给出，范围说明中的一个(或两个)是省略，则其变量是向量(或矩阵)方式. 例如：

REAL A,B(),C(,),D(,64),E(,,3,6)

说明一个实标量 A，一个实向量 B，一个实矩阵 C，一个 64 个实向量的集合 D 和一个 3×6 个矩阵的集合 E.

假定考虑的是 64×64DAP，则上例中 C 和 D 均是总共有 4096 个元素，但 D 不是一个矩阵，它是一个 64 个向量的数组.

3.2.3 数据类型

DAP-Fortran 有如同标准 Fortran 一样的类型：实型、整型、字符型和逻辑型. 逻辑型变量在 DAP-Fortran 中是经常使用的. 逻辑变量仅有值是 TRUE 和 FALSE(或 0 和 1)，它不必作为一个字节来存储，也不宜 作为一个低精度的整数处理. 默认的浮点变量的精度是一个标准 32 位(4 字节)十六进制格式，但用户可以按需要在 3 到 8 字节之间指定任意精度. 对整数也可类似的指定精度范围.

3.2.4 赋值与表达式

运算符+，−，*，/，** 所进行的操作如同标准 Fortran. 例如：

$$X = Y + Z * W$$

表示集合 Z 和 W 的对应元素相乘(逐个元素)，加相乘的结果到对应的 Y 的元素上且存储结果在 X 中. 如果所有量 X,Y,Z 和 W 已定义为是同样的数据方式，则这赋值语句是合法的. 要特别提醒注意的是：在矩阵方式变量上运算时，* 不是表示一般意义下的矩阵乘法，而是矩阵的对应元素的相乘.

3.2.5 函数

DAP-Fortran 中的函数按标准 Fortran 的类似方法操作，但返回的除标量外，还可是向量或矩阵. 因此，对 3.2.4 节举的例子，当 X,Y,Z 和 W 是矩阵变量且要进行矩阵相乘，则可写成：

$$X = Y + MATMULT(Z,W)$$

这里 MATMULT 是一个用户已编写(或库)矩阵乘法程序. 为了编译程序可做语法检查，需要定义任何用户编写的函数的类型，这可以用扩充的 EXTERNAL 语句来描述，内部公共函数 SIN,COS,SQRT 等的结果类型无须说明，编译程序产生代码使得结果与自变量有相同的类型. 例如，COS(A)返回结果与 A 是同一类型，它的元素是 A 的对应元素的余弦.

用户自己也可以定义函数，例如，程序员可以定义一个用某种方法计算向量场偏差的函数 DIV，则他编写程序时，可把 DIV 作为类似于正弦、余弦等一样的操作符来使用.

DAP-Fortran 中常用的计算函数有 ABS,SQRT,EXP,SIN,COS,TRAN 等.

为了向量-矩阵组合的需要，DAP-Fortran 提供了两种函数：MATC 和 MATR. MATC 是产生一个列向量的矩阵，即矩阵的各列均等于同一个列向量. 例如，X=MATC(V)，即是将一

个列向量 V 扩展成一个矩阵 X，X 中每一列均为 V. 类似的，MATR 产生一个行向量矩阵.

还有一个常用的合并函数 MERGE. 例如，

$$Z = MERGE(X,Y,L)$$

这里 X，Y，Z 是 N×N DAP 上的 N×N 矩阵，L 是 N×N 的位逻辑矩阵，则对应 L 的元素为真的位置，置 Z 的元素等于 X 的对应元素，而对应 L 的元素为假的位置，置 Z 的元素等于 Y 的对应元素.（参见下面 3.2.7 小节的条件操作）. 有关矩阵移位的函数在下面 3.2.6 小节中叙述.

3.2.6 下标

下标是用来选取一个集合中的某些项的指示说明. DAP-Fortran 中提供了方便的下标表示和操作的方法. 我们在 3.2.2 节已提到了这方面的问题，我们可以用空下标位置来说明整个一行或一列. 例如，A(3,2)选取集合的第二个矩阵的第三行，而 A(,3,2)选取第三列，但 A(,,2)选取整个第二个矩阵.

应注意的是：形如 B(,3)的下标是语法上正确的，它表示是选取一个矩阵中的一列或是一个向量集合中的一个向量，因为从程序员的观点来看，这两者在逻辑上是相同的.

在一般简单的情况下，每个数组元素仅与另一数组在同一位置（具有相同 Fortran 下标）的元素互相影响. 但有不少情况，不是这么简单的关系，在 DAP-Fortran 中提供了一些新的技巧来处理这些问题.

例如，我们考虑在一个 N×N 的格网上解拉普拉斯方程的迭代法，在每一次迭代，在任一格网点上的值是用四个邻域的值的平均来代替. 用 Fortran 表示为：

```
      DO  1  I=1,N
      DO  1  J=1,N
1     X(I,J)=(Y(I+1,J)+Y(I-1,J)+Y(I,J+1)+Y(I,J-1))/4.0
```

但是，这迭代当 I 或 J 在 1 或 N 的情况（即在格网的边界上）将无法进行，因为下标计算要访问边界以外的点. 人们必须要小心处理在这种情况下的下标变量，一般将要编写一段特殊的程序代码来处理边界影响.

用 DAP-Fortran，解拉普拉斯方程的迭代法可以省去显式的下标变量，而以一种简明记法表示成：

$$X = (Y(+,) + Y(-,) + Y(,+) + Y(,-))/4.0$$

其中 Y(+,)称为移动下标，它用来访问数组中每个点的邻域，不过对应形如 Y(I+4,J+3)的 Fortran 的表达式是要特殊处理的.

如同用 Fortran 一样，对于格网问题的 DAP-Fortran 代码必须访问整个数组，但不必像 Fortran 情况那样，在边界上要由程序员显式定义. 程序员能用 GEOMETRY 语句告诉 DAP，他希望如何处理边界. 在多数通常情况下，两种边界状况之一将成立. 第一种状况，称为平面边界，假定数组的边界之外是值为 0 的一个区域，从而访问数组之外产生 0 结果. 第二种状况，称为循环边界，假定数组的一条边是直接连接到对边. 按北-南和东-西方向组合这两种几何状况，DAP 可构造出一个平面，有两个定向的圆柱面或环形圆纹曲面. 另外的一些边界情况可以用屏蔽的技巧（见 3.2.7 节）来处理.

关于矩阵存储与下标表示我们可以用图 3.2 更清晰的表示.

通常 A(I,J)是带下标表示的矩阵，第一个下标标记行，第二个下标标记列. DAP 习惯标

记阵列的第一行为北(NORTH),而标记阵列的第一列为西(WEST). 类似的最后一行标记为南(SOUTH),最后一列标记为东(EAST).

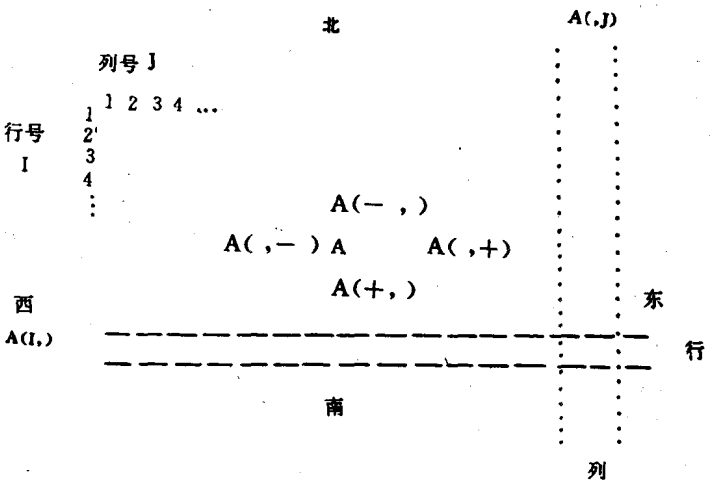

图 3.2 DAP 上的矩阵存储与下标表示

这种习惯表示用于移位函数. 例如,移位函数 SHEP(A,N)表示矩阵 A 往东移动 N 列,即 A 的第 1 列变成第 N+1 列. 这里 P 表示平面移动,使 A 的最后 N 列移出阵列,而 A 的前 N 列则填"0". 如果 N 没有给出,默认 N=1. 如果是在 64×64DAP 上,若 N≥64,则按 mod 64 计.

SHEC 函数按东-西方向矩阵往东实现一次循环移位.

当移动 1 行或 1 列时,可用±记号来表示,如前面所提到的移动下标的形式. 例如,A(,+)是等价于 SHEP(A). 当用±记号时,移位默认为是平面移位.

如需要东-西方向循环移位或北-南方向循环移位或两个方向均循环移位,则必须用 GEOMETRY语句说明.

循环移位的函数名中均包含有字母 C,这里 C 为 Cyclic 的首字符.

例如,SHWC 函数表示往西进行循环移位. 循环移位时,阵列的东、西两边或南、北两边是看作直接相连的. 对于二维情况的移位还可以用函数复合的方法. 例如,在平面情况,

$$M1 = SHNP(SHEP(M,3),4)$$

首先实现数组 M 向东移动 3 个位置,接着向北移动 4 个位置. 在边界上用 0 填入. 这等价于下列 Fortran 代码:

```
      DO  1  I=1,N-4
      DO  1  J=4,N
1     M1(I,J)=M(I+4,J-3)
```

如果要求循环移位,等价的 Fortran 代码将更复杂.

3.2.7 条件操作

我们前面举的一些例子是假定在阵列的每一个点上执行同样的动作. 对于很多实际应用不总是这种情况,从而 Fortran 用户必须使用 IF 语句. 例如,我们再考虑在 3.2.1 节提到的一元二次方程的例子. 假定给定

$$ax^2 + bx + c = 0$$