并行程序的设计方法

李三立 杨宋源 编著

THE DESIGN METHOD
OF
PARALLEL PROGRAM

上海科学技术文献出版社

(沪)新登字301号

责任编辑：张秉芬
封面设计：何永平

并行程序的设计方法
李玉茜　杨宗源　编著
*
上海科学技术文献出版社出版发行
(上海市武康路2号　邮政编码200031)
全国新华书店经销　上海市印刷十二厂印刷
*
开本850×1168　1/32　印张6.25　字数174,000
1994年12月第1版　1994年12月第1次印刷
印数：1—1,100

ISBN 7-5439-0494-2/T·335
定价：16.00元
《科技新书目》324-616

前　言

计算机系统已在科学计算、决策支持活动、金融活动等领域中得到广泛的应用。反过来，这种应用也促使计算系统，特别是并行计算机系统的发展。至此，并行软件设计方法的研究已提到议事日程上来。

早在三十年前，为了管理计算机系统的各种资源，提出设计操作系统的任务，操作系统是典型的在顺序计算机上实现并行计算的实例，C. A. R. Hoare 等著名计算机专家提出进程和进程通讯等并行程序的基本概念和理论，实现了在顺序计算机上的并行计算。其次，对于顺序程序，程序设计工作者具有丰富的经验，和大量的软件支撑工具，环境，使得顺序程序的设计变得方便得多。用顺序方式设计并行程序，风险小且成本低；直接用并行方式设计并行程序，困难和风险都将大得多。因此，导致不少并行程序的设计专家致力于研究如何把顺序程序转换成并行程序，以减少设计并行程序的风险和降低其成本；也有不少专家研究和探讨自动完成顺序程序到并行程序的转换，不管人工转换还是自动转换，一般来说，用转换方法所得的并行程序的效率总是低下的。由于当今科学技术的发展，特别是通讯技术、VLSI 技术、嵌入系统的进步，一方面降低了计算机系统的生产成本，另一方面也使并行程序的开发变得容易得多，使程序设计员乐于采用并行方式求解，获得高效率、高质量的并行程序。鉴于这种的进步和变化，使我们感到有必要研究和讨论并行程序设计的方法、理论、正确性验证和映射等问题，并介绍给国内的同行。

为了适应当前并行技术发展的需要，我们试图从方法学的角度，用并行求解方法，讨论并行程序的设计和构造方法。本书主要取材于[7]，初稿曾为研究生试讲过两次。本书共分八章，第一章介

绍并行计算机系统和并行程序的模型，第二章建立描述和证明并行程序性质的逻辑系统，第三章给出描述抽象并行程序的程序设计语言，第四章以例子讨论抽象程序到各种体系结构的映射方法，第五章介绍并行程序的构造方法，第六章、第七章、第八章以并行系统中的进程互斥、进程通讯、程序检测为例，说明并行程序的设计、构造、验证和映射的方法。

宣泰章副教授曾参加讨论本书的编写，并提出宝贵的意见，特此致谢。

本书第一章～第五章和第八章由李玉茜编写，第六章和第七章由杨宗源编写。由李玉茜负责统编全书。

由于水平有限，匆促成书，谬误之处，在所难免。特敬请同行和读者指正。

编　　者

1993 年 11 月于上海

内 容 简 介

本书从方法学的角度讨论并行程序的设计和构造方法。本书建立并行计算机系统和并行程序的模型；基于此模型，建立逻辑系统，用于描述程序的规格说明、程序的性质和验证其正确性；给出并行程序设计语言，用于描述实现规格说明的抽象程序；讨论抽象程序到各种体系结构的映射；以并行系统中的进程互斥、进程通讯、程序检测、废品收集等为例，说明程序的设计、构造、验证和映射的方法，最后给出相应的并行程序。

本书可作为计算机科学理论、软件专业的大学本科高年级学生和研究生的教科书和参考书，也可供从事于计算机科学理论、并行处理和软件研究工作的科技人员及教师阅读和参考之用。

目　录

第一章 基本概念

为了提高程序的质量，缩短程序研制的周期，保证程序的正确性，多年来程序的设计方法，特别在程序的结构、软件环境、软件的生产方式等方面的研究中，提出“软件工程”概念，把软件的设计过程工程化、规范化，推动了程序设计的发展。现今程序设计方法学已成为计算机科学的重要组成部分，是软件工程的柱石。

三十多年的程序设计经验告诉我们：程序的设计过程应该把算法设计（和系统体系结构无关的）和算法实现（和系统体系结构有关的）两个步骤尽可能地分开。前者仅给出问题性质的描述，也就是问题的规格说明，求解算法和其正确性的证明，在此基础上，对算法不断地细化和再证明，直到求得使程序设计者满意的解为止，按这个算法写出抽象程序；后者，把算法设计的结果——抽象程序，结合考虑某体系结构的计算机系统，把抽象程序变换到在其上可计算的程序。把算法实现尽可能地推迟到开发阶段的后期，这是程序设计的原则之一。并行程序的设计尤其应遵守这个原则。

管理计算机系统本身资源的操作系统是并行程序的典范。为设计和研制操作系统，E. W. Dijkstra, C. A. R. Hoare 提出进程、进程通讯等概念和基于顺序计算机的并行程序设计理论。随着 VLSI 和通讯技术的发展和进步，并行程序设计研究的深入，和并行系统结构的完善，使并行计算机系统的规模大大地超过传统的向量式巨型计算机系统，也使并行程序的设计变得容易得多，导致程序员摆脱基于顺序计算机开发研制并行程序的求解方式，探讨并行方式的设计和研制方法；随着并行程序的调试，并行技术的进一步研究，今后将会出现形式化的程序验证方法。总之，并行计算机系统的出现，一定促进并行程序的设计理论、方法、程序的构造

等的研究。

本书企图从方法论的角度，讨论并行程序的设计基础理论及其设计方法。本书介绍并行程序的计算模型、与之相关的逻辑证明系统、程序设计的抽象语言、程序的设计和构造方法，并用例子详细说明之。

§1. 并行计算机系统模型

为提高处理速度和资源的利用率，“并行”处理多个相关的事件，起着积极和重要的作用。硬件系统的低层并行加法器，软件系统中的操作系统，和分布式计算机系统都是并行处理的典型例子。

并行概念包括两种含义：并发性和同时性。前者指多个事件在同一时间间隔内发生；后者指多个事件在同一时刻发生。如多位并行加法器，从微观上看，进位信号总是由低位向高位传递，低位完成加法运算在时间上总是处于领先的，因此，应该是并发地进行位相加；从宏观上看，可看成在同一时刻内完成位相加。因此，今后我们不严格地区分并发性和同时性，用并行性泛指它们二者之一。

定义 1.1 理想的计算机系统由 n 个处理器，每个处理器带有一个私有存贮器，为处理器所共有的共享存贮器，和控制总线所组成。

• 处理器：能独立计算具备条件的赋值语句，并计算总是终止的。

• 私有存贮器：属某处理器所私有，只允许该处理器对它作读写访问，读取或写入它所计算的中间结果，其它处理器不可访问它。

• 共享存贮器：属 n 个处理器所共有的，所有处理器都可对它进行读写访问。为正确访问起见，规定如下限制：多个处理器可同时对其中某单元作读访问，但当某处理器对其中某单元作写访问时，将封锁其它处理器对它作读写访问。

当处理器对共享存贮器中某单元作写访问时，称该单元是被占有的；否则称为自由的。

共享存贮器用于描写计算机系统的共享资源，记录处理器之间信息交换等。

·控制总线：负责实现处理器并行计算，协调各处理器对共享存贮器的访问，数据的输入和输出时设备的启动等。

理想的并行计算机系统的如图 1.1 所示。该系统以赋值语句为原子操作，也就是赋值语句的计算是不可中断的，并且总在有限时间内终止的。并行程序由若干个具有这种性质的赋值语句组成。

理想的并行计算机系统中没有显式的调度器，来调度并行程序中语句的计算。任何语句只要具备足够的资源，满足特定的条件，就可被计算，由于语句的计算占有必要的共享资源，因此语句的计算序列和访问共享资源有关。

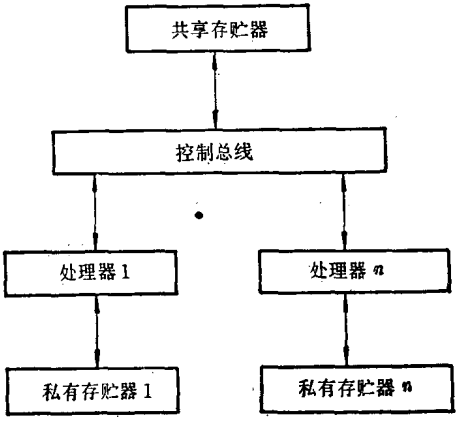

图 1.1 理想并行计算机系统

§2. 并行程序模型

并行程序的特征是程序计算的并行性和资源的共享性。前者

指多于一个进程在计算时间上的重叠；后者指多于一个进程共享某种资源或访问共享变元。

在理想的计算机系统定义支持下，将给出并行程序的概念、并行程序的逻辑系统、并行程序设计语言和并行程序的构造方法。

在下面讨论中，语句是并行程序的基本单位。

定义 1.2 进程的定义

(1) 任何语句是进程。

(2) 假如 u, v 是进程，那末 $u \| v$ 也是，其中小长方形符 $\|$ 为分隔符。

(3) 只有(1)和(2)构成进程。

从定义 1.2 可知，简单的进程只含有一个语句；复杂的进程由若干个语句并行地、嵌套地组成。

例 1.1 语句 $x:=y+1$ 和 $y:=x$ 分别为进程，由它们组成的

$$\begin{array}{l} x:=y+1 \\ \| \; y:=x \end{array}$$

也是进程。

例 1.2 计算 n 阶单位阵的进程可表示为

$$\langle \| i: 1 \leqslant i \leqslant n :: u[i, i]:=1 \rangle$$
$$\| \langle \| i, j: 1 \leqslant i \leqslant n \wedge 1 \leqslant j \leqslant n \wedge i \neq j :: u[i, j]:=0 \rangle$$

其中 $\langle \| i: 1 \leqslant i \leqslant n :: u[i, i]:=1 \rangle$ 表示 n 个并行进程

$$\begin{array}{l} u[1, 1]:=1 \\ \| \; u[2, 2]:=1 \\ \quad \vdots \\ \| \; u[n, n]:=1 \end{array}$$

的缩写；类似地 $\langle \| i, j: 1 \leqslant i \leqslant n \wedge 1 \leqslant j \leqslant n \wedge i \neq j :: u[i, j]:=0 \rangle$ 表示 $(n-1)^2$ 个进程 $u[i, j]:=0$ 的缩写。

定义 1.3 并行程序是四元组 (V_p, V_s, I_s, S)，其中，V_p 表

示由有限个私有变元集所组成的类，称为私有变元类。其元素和固定的进程相联系，仅供该进程作读写访问。

V_s 表示由有限个共享变元所组成的集，称为共享变元集。其元素允许多于一个进程作读写访问。

I_s 表示由有限个进程所组成的集，称为初始进程集。其元素在并行程序体 S 的计算之前被计算，从而确定程序中某些私有变元和共享变元的初值，程序中不赋初值的变元可取任何合法的值。I_s 中的进程计算且只计算一次。

S 表示由有限个进程所组成的集，称为并行程序体。它是并行程序的核心部分，其中的进程被重复地、协调地计算，即满足条件的进程就可被计算，且会终止的。

并行程序的模型如图 1.2 所示。

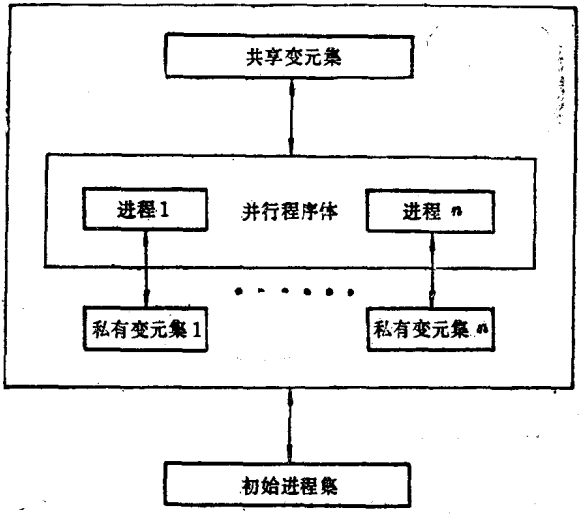

图 1.2 并行程序的模型

并行程序的计算特征是语句(进程)计算的无序性和计算结果的不确定性。

• 无序性　由于计算的并行性和资源的共享性，使得并行程序中语句(进程)的计算次序和顺序程序的情形完全不同。后者是一

句一句依次地计算，而前者由当时系统所处的环境和资源决定计算哪条语句(进程)，因此不可能像顺序程序一样，用控制流程图描写语句(进程)的计算序列，和事先规定语句(进程)的计算速度。

• 不确定性　并行程序中语句(进程)计算的相互牵制和影响，将会改变它的独立计算的行为，因此，两个静态功能等价的并行程序，其动态计算的结果往往是不同的，甚至于同一程序从相同的变元值出发，其计算结果往往也是不同的。这种现象为计算结果的不确定性。

不确定性使并行程序的测试极为困难，不能像测试顺序程序那样，用相同的数据多次重复计算发现和纠正程序中的错误，和在语句的前后附设前置和后置断言来验证程序的正确性。因此必须从并行程序的整体来考察程序的测试和正确性。

尽管并行程序的计算无序性和不确定性，但语句(进程)的计算总是服从某种约束而协调地工作，完成并行程序的总体目标。称并行程序服从特定的约束而工作为同步。这也是并行程序计算的特征之一。产生同步的原因是程序访问共享变元，可以断言，对共享变元访问的次序将直接影响语句(进程)的计算结果和速度。同步的特例是互斥，它规定在同一时刻最多只允许一个语句(进程)访问某共享变元。

在理想计算机系统中，语句(进程)间的同步和互斥是隐式发生的，它由系统控制实现。

由于并行程序完成某特定的任务，因此，它应具有如下的性质：

1) 不变性　程序的计算变元值(包括私有变元和共享变元)总满足某个关系式，称为程序的不变式。它表明并行程序的计算"不会做坏事"，因此也称为安全性。

例1.3　对于整数 m 和 n，设除法的并行程序为

初始进程集

$$x, y, z, k:=0, m, n, 1$$

并行程序体：

$$z,\ k:=2*z,\ 2*k \qquad \text{IF } y\geqslant 2*z$$
$$[\!]\ z,\ k:=n,\ 1 \qquad \text{IF } y<2*z$$
$$[\!]\ x,\ y:=x+k,\ y-z \qquad \text{IF } y\geqslant z$$

其中 x 为进程 $x:=x+k$ IF $y\geqslant z$ 的私有变元，y，z 和 k 是程序的共享变元。

容易验证除法并行程序有不变式

$$z*n+y=m \wedge 0\leqslant y<n$$

和

$$y\geqslant 0\wedge k\geqslant 1 \wedge z=n*k \wedge x*n+y=m$$

2) 进展性　并行程序的计算过程中，某事件一定会发生，或者在某事件出现的前提下，另一事件一定会发生，这种性质称为程序的进展性。它表明程序的计算“总会做好事”。

例1.4　例1.3的程序的进展性，可表示为

当 $y\geqslant z\wedge y=M$ 时，程序计算后一定会有 $y<M$。

3) 公平性　并行程序中的任何语句（进程），在限定的时间内一定有机会被计算，也就是说，不会有语句（进程）永远被遗忘而不被计算，称为公平性。

一般地说，系统的公平性依赖于调度算法。在我们理想的计算机系统中，并没有显式的调度器，而使公平性被隐含在问题求解的算法中，在算法设计时，就要考虑公平性（例见第六章§4 哲学家就餐问题）。

4) 不动点　对于并行程序，假如所有变元，在计算并行程序中任何语句（进程）后，其值总保持不变，称该变元的值为并行程序的不动点。

假如并行程序有不动点，则称程序的计算是终止的，或者收敛的。不动点就是并行程序的计算结果。假如不存在不动点，则程序的计算是不终止的，或者发散的。

例1.5　除法程序例1.3是终止的，其不动点 $(x,\ y,\ z,\ k)$ 取 $(\mathrm{DIV}(m,\ n),\ \mathrm{REM}(m,\ n),\ n,\ 1)$。

由语句 $x:=y+1$ 和 $y:=x$ 所组成的并行程序不存在不动点，

因此它是不终止的。

5）死锁　设系统中有一组进程，其中每个进程都处在等待被组中另一进程所占有，又不能抢占的资源，使组中每个进程都处于阻塞而不能继续工作下去的状态，这种现象称为该组进程处于死锁状态。

由进程对共享资源的使用要求，进程并行计算的速度，和并行程序本身的错误等等原因，可诱发死锁的产生。系统一旦出现死锁，将导至局部瘫痪或全局瘫痪，造成极大的损失。因此，并行程序系统中必须采取措施，以防止、避免死锁现象出现。第八章将研究死锁的检测。

6）瓶颈口　系统中共享资源的有限性，各进程计算时间的不等性等原因，使在并行计算中含有顺序计算的成分，它将影响并行程序并行计算的速度。在程序中，影响并行计算的卡口，称为程序的瓶颈口。

设法疏通瓶颈口，提高程序的计算速度，是程序细化工作之一。

§3. 映射和体系结构

定义 1.4　把由算法设计所得到的独立于体系结构的抽象程序，变换到特定体系结构的计算机系统上的可计算程序，这种变换称为程序的映射，简称映射。

映射是已设计的算法在具体计算机系统上的实现，它应考虑语句（进程）的处理器分配，存贮器分配和计算的控制流（语句的隐式或显式的计算序列），以提高程序的效率。

下面讨论到各种体系结构上映射的特点。

1. 到顺序体系结构的映射

顺序结构的并行计算机由一个处理器、一个存贮器、和控制总线组成。

该处理器一句一句地计算满足条件的语句（进程），并且计算总是终止的；程序中每个语句（进程）将被无穷次地计算；程序动态

计算的语句(进程)次序和其静态的语句(进程)次序无关。

2. 到异步共享体系结构的映射

异步共享的并行计算机由多个独立的处理器，与处理器个数相同的私有存贮器，一个共享存贮器，和控制总线组成。

多个处理器同时地计算满足条件的语句(进程)；私有存贮器供各自的处理器直接访问；共享存贮器允许多于一个处理器的访问。

映射到异步共享体系结构的程序的限制：

1) 共享变元是多个处理器访问的对象。允许同时对某存贮单元作读访问，但当有一个处理器对某存贮单元作写访问时，就封锁其它处理器对它作读和写访问。

2) 赋值语句

$$x:=f(y,\ z,\ \cdots)$$

其中赋值号":="左边的变元 x 是可写的，右边的变元 y, z, …是可读的。由于赋值语句是程序计算的基本单位，因此，假如赋值号左边的变元 x 是共享变元，它就不允许同时在其右边出现。

3) 处理器无条件且无穷次地计算分配给它的任何语句(进程)。

3. 到同步共享体系结构的映射

同步共享的并行计算机由多个独立的处理器，与处理器个数相同的私有存贮器，一个共享存贮器，一个公共的时钟，和控制总线组成。规定，公共时钟滴嗒一声为一个时间步。

映射到同步共享体系结构的程序的限制：

1) 处理器的分配，为简单起见，限定在一个时间步内，只计算一个语句(进程)。例如系统中有三个处理器，假如计算二重赋值语句

$$x,\ y:=f,\ g$$

那末，分配两个处理器计算它们，多余的一个处理器为空闲而不计

算其它语句；假如计算四重赋值语句

$$w, x, y, z := f, g, h, k$$

那末，就要把它分解成在几个时间步内完成计算，在每一时间步内，允许有空闲的处理器，当然，空闲处理器的个数越少，处理器的利用率就越高。

2) 在一个时间步内，允许多于一个处理器以相同的内容对某存贮单元作写访问；允许多于一个处理器对某存贮单元同时作读访问；不允许多于一个处理器对某存贮单元作既读又写的访问。

3) 赋值语句

$$x := f(y, z, \cdots)$$

其中赋值号":="左边的变元 x 是可写的，右边的变元 y, z, 是可读的，读写规则满足第2)条规定。

4) 处理器无条件且无穷次地计算分配给它的语句(进程)。

4. 到分布式体系结构的映射

分布式的并行计算机由多个独立的处理器，与处理器个数相同的私有存贮器，连接处理器之间的通道，和控制总线组成。

属处理器私有的存贮器只供该处理器读写访问；通道替代并行计算机模型中的共享存贮器，交换处理器之间的消息，其工作方式为：处理器沿着通道发送消息给其它处理器，或者从通道接收其它处理器发送来的消息。其构造如图 1.3 所示。通道可由一个缓冲器组成，当缓冲器不满时，发送处理器可发送消息至缓冲器；当缓冲器不空时，接收处理器接收来自缓冲器的消息。

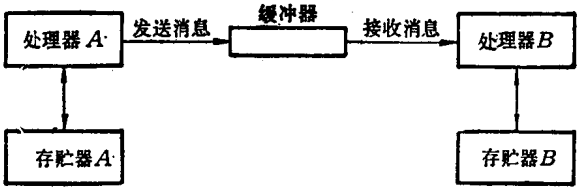

图 1.3 分布式体系结构的基本单元