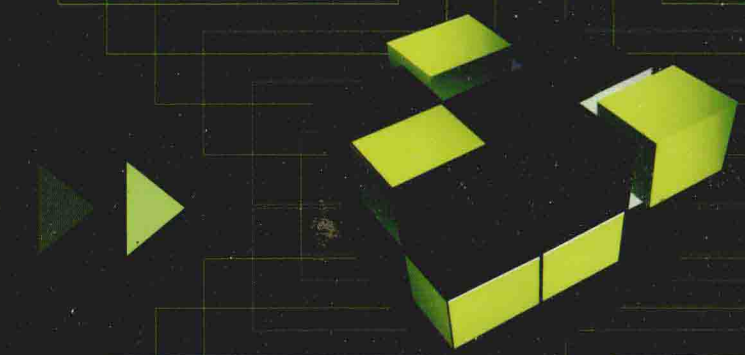
基于CUDA的并行程序设计

刘金硕　邓娟　周峥　曾秋梅　等　编著



- 详细介绍GPU的硬件架构、CUDA程序设计与优化、GPU存储器使用技巧等相关知识
- 并行计算的理论与实践相结合，典型应用案例介绍，关键代码呈现

科学出版社

基于 CUDA 的并行程序设计

刘金硕　邓　娟　周　峥　曾秋梅　等　编著

科学出版社

北　京

内 容 简 介

本书主要介绍基于 CUDA 的并行程序设计的原理、开发方法和硬件基础，并给出了应用实例。全书内容共 9 章。第 1～5 章为基础知识部分，介绍 CUDA 并行程序设计的原理和开发环境，包括并行计算概述、GPU 概述、CUDA 编程基础、GPU 存储器使用技巧和 CUDA 编程优化。第 6～9 章为应用实例部分，分别详细讲解基于 C++的遥感影像处理的 CUDA 优化、基于 OpenGL 的体绘制技术实现剪切波数据三维可视化的 CUDA 优化、基于 MATLAB 的生物细胞图像病理诊断的 CUDA 优化和基于 CUDA 的核外计算集群中间件技术。书中包含实例代码，需要读者具有一定的编程开发基础。

本书适合高等院校计算机、电子工程、通信工程等相关专业的教师、研究生阅读，也可供从事 CUDA 设计与开发的科研技术人员、程序员参考。

图书在版编目(CIP)数据

基于 CUDA 的并行程序设计 / 刘金硕等编著. —北京：科学出版社，2014.5
ISBN 978-7-03-040531-9

Ⅰ. ①基… Ⅱ. ①刘… Ⅲ. ①图象处理－程序设计 Ⅳ. ①TP391.41

中国版本图书馆 CIP 数据核字(2014)第 086978 号

策划编辑：任 静 / 责任编辑：任 静 / 责任校对：杜伟利
责任印制：张 倩 / 封面设计：迷底书装

科 学 出 版 社 出版
北京东黄城根北街 16 号
邮政编码：100717
http://www.sciencep.com

新科印刷有限责任公司 印刷

科学出版社发行 各地新华书店经销

*

2014 年 5 月第 一 版 开本：787×1 092 1/16
2014 年 5 月第一次印刷 印张：17 1/4
字数：411 000

定价：65.00 元

(如有印装质量问题，我社负责调换)

前　言

CUDA 作为一种并行计算架构，是以 GPU 作为数据并行计算设备的软硬件体系。与传统的网格和集群等并行架构相比，GPU 的并行不仅体现了节能、体积小、性价比高等优势，而且加速比更优。CUDA 已经被广泛应用于科研和金融等领域，具体包括：视频与音频处理、石油天然气勘探、产品设计、医学成像、物理效果模拟、计算生物学、计算化学、流体力学模拟、CT 图像重组、地震分析和光线追踪等，并且正在不断向更多领域拓展。基于 CUDA 的并行程序设计已成为一种重要的热点技术。

本书的主要内容分为基础理论部分和应用实例部分。基础理论部分包括：并行计算概述、GPU 概述、CUDA 编程基础、GPU 存储器使用技巧和 CUDA 编程优化等。应用实例部分包括：基于 C++的遥感影像处理的 CUDA 优化、基于 OpenGL 的体绘制技术实现剪切波数据三维可视化的CUDA优化、基于MATLAB的生物细胞图像病理诊断的CUDA优化和基于CUDA的核外计算集群中间件等。通过对学习本书，读者可以充分理解并行计算和 GPU 架构、熟练掌握 CUDA 架构下的多种语言的并行编程技术。

本书由刘金硕、邓娟、周峥、曾秋梅、江庄毅、刘天晓、程力和涂梅生共同编写。全书由刘金硕和邓娟统编定稿。

由衷感谢我在荷兰莱顿大学撰写论文时，导师 Fons Verbeak 对我的引导——“写的东西，要站在读者的角度，想读者是否能够读懂；读者能否因为你写的东西，由浅入深地学会想要学会的东西！”感谢所有能让本书顺利出版的朋友们！感谢武汉大学陈莘萌教授、邹华博士、夏清先生在本书构思之初给我的各种启发。感谢英伟达公司谢强先生对本书的各种建议。感谢英伟达公司对本书和武汉大学刘金硕、邓娟课题组的资助。特别感谢课题组全体师生的辛苦付出，他们朝气蓬勃、孜孜不倦地工作和学习，也深深感染着我。

刘金硕现任职于武汉大学计算机学院，邓娟任职于国际软件学院，周峥现任职于中国舰船研究设计中心工作，曾秋梅现为武汉大学计算机学院研究生。课题组近年获批了英伟达公司全球教学中心、全球研发中心；本书中基于 MATLAB 的生物细胞图像病理诊断，基于刘金硕在荷兰莱顿大学所做的项目；基于 OpenGL 的体绘制实现剪切波数据三维可视化，基于地震行业专项 201008007 部分内容；基于 C++的遥感影像处理，基于课题组遥感相关项目的累积；基于 CUDA 的核外计算集群中间件，基于周峥在美国俄亥俄大学的项目。

由此真心希望读者能够轻松地、由浅入深地学习 CUDA 并行编程技术。同时，由于作者水平有限，书中难免有不足之处，恳请读者批评指正。

刘金硕

2014 年 4 月

目　录

第 1 章　并行计算概述

1.1　并行计算简介

在生活中，所谓“并行”，顾名思义就是一个人同时做几件事情，或者多个人同时做一件事情。对于计算机而言，“并行”就是同一时间间隔内，增加操作量，多个运算部件共同完成一个任务。并行计算(parallel computing)就是由运行在多个部件上的小任务合作来求解一个大规模计算问题的一种方法[1]。

随着计算机在各行业的应用越来越普及，人们对计算速度的期望越来越高。目前，并行计算已经广泛应用于天气预报、石油勘探、地震预报、医学图像诊断、遥感卫星图像的分析处理，以及航空航天技术等领域。在科学计算领域，待求解问题的规模也越来越大，采用并行计算可以降低单个问题求解的时间，增加问题求解规模，提高问题求解精度，改善多机同时执行多个串行程序的容错性、提高可用性和吞吐率。

提高计算机计算性能的方法，笼统地说有：①改善单机性能。例如，改善 CPU、存储、传输速度等硬件配置。②更多运算部件互相协同、互通有无地共同完成一个任务。③合理利用现有的硬件和软件资源，提高处理效率。例如，更合理的算法设计，或者更好的进程调度等。

为提高并行计算效率，计算机的系统结构主要进行以下几个方面的改进[2]。

(1) I/O 通道。将费时的 I/O 操作交给 I/O 处理器(通道去做)，让 CPU 主要进行计算工作。

(2) 交叉存储。将存储体分成多个模块，实现并行存取和削减存取冲突。

(3) 高速缓冲存储器。减少主存与处理器的数据交换时间，改善两者传送速度差异。

(4) 指令预取。一次取出多条待处理指令，使取指令可以与指令译码同样快。

(5) 多功能单元。在中央处理器中设置多功能单元(加法器、乘法器等)可以提高单一程序的吞吐量，缩短周转时间。

(6) 流水线技术。多条指令在同一时间允许在不同的阶段按流水方式执行不同的操作，实现指令重叠。

(7) 向量处理技术。一条向量指令可以同时处理 n 个分量，它们可以同时在各个处理器中进行运算，也可以连续地送入流水线中进行重叠处理。

(8) 多道程序设计。同时把若干作业放在内存中，允许在同一时间有多道程序处于运行状态。

(9) 分时系统。允许多个终端用户同时交互地使用同一台计算机。

(10) 数据驱动。与冯·诺依曼计算机不同，它不是采用指令驱动操作，而是采用操作数驱动操作。这样，若有多个操作数准备就绪，就可以彼此并行地执行而不受指令顺序执行的限制，从而可以充分开拓进度。

1.2　并行处理的计算机体系结构

1.2.1　并行计算机分类

对于计算机系统，存在着几种不同的分类方法：Flynn 分类法[3]基于计算机系统中指令流和数据流的重数；Feng（冯译云）分类法基于字和位的串行或并行处理；Handler 分类法[2,4]基于处理机控制部件、运算部件和位-级电路 3 个子系统层次的并行或流水线处理的程度。我们仅引用 Flynn 分类法和 Handler 分类法。

1. Flynn 分类法

Flynn[3]提出指令流、数据流和多倍性的概念，把不同的计算机分为四大类。

（1）单指令流单数据流（single-instruction single-data，SISD）；

（2）单指令流多数据流（single-instruction multi-data，SIMD）；

（3）多指令流单数据流（multi-instruction single-data，MISD）；

（4）多指令流多数据流（multi-instruction multi-data，MIMD）。

SISD 是传统的串行计算机。指令按时序执行，但是它在执行的阶段上可以重叠。大多数的 SISD 单处理机系统是流水线结构的。

SIMD 相当于带分布存储器的阵列机、流水线处理机，SIMD 并行机。SIMD 有多个处理单元（PE），由同一个控制部件管理。所有 PE 接收控制部件发送的相同指令，对来自不同数据流的数据集合序列进行操作。共享存储器系统可包含多个模块。

MISD 是针对同一数据流进行处理的。这类计算机的实际机器并不多，一般认为超标量计算机、超长指令字（VLIW）计算机、退耦（decounted）计算机和专用脉动阵列（systoic arrays）计算机可以归为此类计算机。

MIMD 是多机系统，即多个处理机系统，每个处理机可以独立执行指令和处理数据。一般并行计算机大多采用这种结构。共享存储器为松耦合带共享存储器。MIMD 灵活性好。通过适当的软硬件支持，MIMD 可以用做单用户机器，针对一个应用程序发挥其高性能；也可以用做多道程序机器，同时运行多个任务；还可以是这两种功能的组合。MIMD 性价比好，可以充分利用商品化微处理器在性能价格比方面的优势。实际上，现有的多处理机系统几乎都采用与工作站和单处理机服务器相同的微处理器作为其处理器。

2. Handler 分类法

1977 年，Handler[2,4]根据计算机系统中流水线处理程度和并行程度出现的级别，将一台计算机表示为三对数（如式（1.2.1）），包括处理机控制部件（processing control unit，PCU）或宏流水、运算逻辑部件（arithmatic logical unit，ALU）或指令流水和位级电路（bit level circuit, BLC）或操作流水。PCU 对应 CPU，ALU 对应功能部件或阵列机的处理单元，BLC 对应 ALU 中实现一位操作所需要的逻辑。

一台计算机用三对整数描述为

$$T(C)=<K\times K',D\times D',W\times W'> \tag{1.2.1}$$

式中，K 为 CPU 的数目；K'为能够执行流水线的 PCU 数目；D 为由每个 CPU 控制的 ALU

数目；D'为可以执行流水线的 ALU 数目；W 为 ALU 中的位数或处理单元 PE 的位数；W'为所有 ALU 中或单个 PE 中的流水线阶段数。若括号中任一对中第二个数为 1，则可忽略。×操作用以连接单计算机系统中不同种类处理器描述。

例如，CDC6600 计算机，它有一个 CPU；ALU 有 10 个功能单元，它们都可以流水地执行；CDC6600 字长 60 位；最多有 10 个外围 I/O 处理器，它们可以与 CPU 并行地工作，每个 I/O 处理器有一个字长为 12 位的 ALU，则 CDC6600 可以描述为 T(CDC6600) = T(中央处理器)×T(I/O 处理器) = <1,1×10,60>×<10,1,12>。

然而，Handler 分类又过于依赖机器，目前大家都普遍遵循 Flynn 分类法。

还有一些其他的分类法，如按指令流和执行流的 Kuck 分类法、按并行度的 Feng 分类法等，读者可以参考文献[5]。

1.2.2　并行计算机的物理结构模型

下面介绍一些典型的并行计算机。

1. 阵列处理机

阵列处理机(array processor)[6]是由在同一控制器控制下的多个处理单元组成的，各处理单元是不带指令控制部件的算数逻辑部件。在控制器的作用下，各处理单元各自对分配来的数据并行地完成同一指令所规定的操作，如图 1.1 所示。阵列处理机是一种单指令多数据计算机，也称为并行处理机。阵列处理机的指令执行情况和一般的单机执行指令的情况相同，也是顺序执行的。指令分成两类：一类是通常的指令，由控制器本身解释执行；另一类是并行处理指令，控制器把这类指令传送给所有的处理单元，由处于“活动”状态的处理单元并行地执行这类指令。处理单元之间以互联网络为数据交换的通路。阵列处理机还设有专门系统管理处理机，用以完成系统控制和输入输出功能。

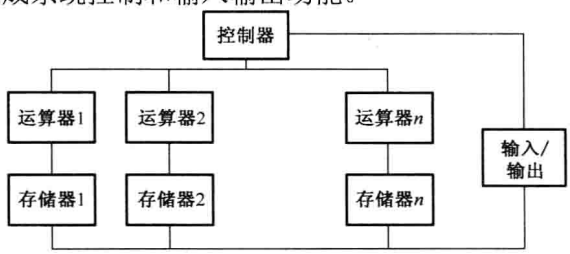


图 1.1　阵列处理机原理图

阵列处理机根据存储方式不同可分为分布存储器的阵列处理机和共享存储器的阵列处理机。分布存储器的阵列处理机中，各个处理单元有独立的存储器，用来存放程序和数据。此外，阵列控制器(CU)中也具有自己的存储器，以存放系统程序和用户程序，它还可存放各个 PU 所需的共享数据存储器。每个处理单元仅和自己的存储器直接相连。美国 Burroughs 公司和伊利诺伊大学于 20 世纪 60 年代联合研制的 ILLIAC-IV 型阵列处理机处理单元(processing unit，PU)如图 1.2 所示，美国 Goodyear 宇航公司在 1979 年研制的巨型并行处理机(MPP)、英国 ICL 公司在 1980 年研制的分布式阵列处理机(DAP)都采用这种结构。共享存储器的阵列处理机，多个处理单元通过互联网与共享存储器相连。一般情况下，存储器的数目等于或大于处理单元的数目。美国的 Burroughs 公司和伊利诺伊大学于 1979 年联合研制的 Burroughs 科学处理器(Burroughs Scientific Processor，BSP)就采用这种结构。

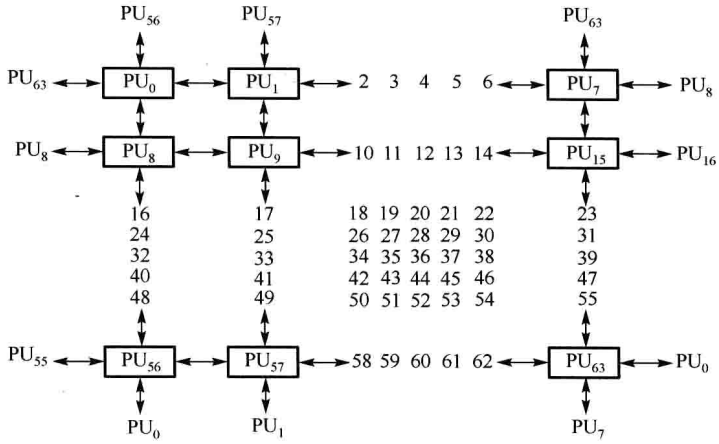


图 1.2　ILLIAC-IV 型阵列处理机处理单元

2. 流水线处理机

流水线处理机(也称为向量机)[4]将流水线技术应用于计算机中。把计算机的运算部件或控制部件等装配成一些有序的子部件，利用功能部件分离与时间重叠的办法，使每个被操作对象处在整个操作流程的不同功能部件中，且保持在不同的阶段完成，从而达到操作级的并行。流水线处理器的思想不仅是在指令上，而且在功能划分的基础上，以流水线方式组成高速中央处理器，进而达到由一台机器的中央处理器设置多条专用流水线。这种技术是目前巨型机的主要方式，如图 1.3 所示。

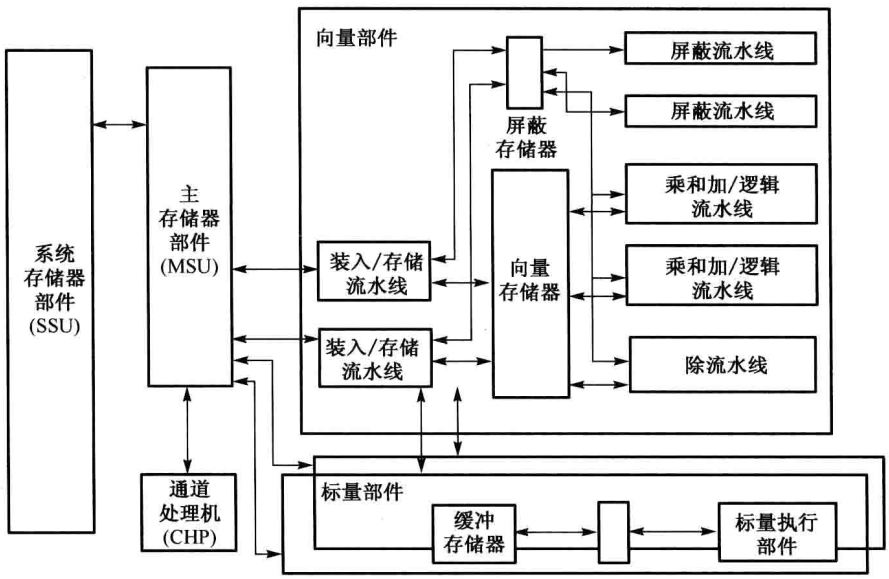


图 1.3　Fujitsu 公司的 VP2000 系列向量计算机系统

3. 多处理机和多计算机

多处理机系统和多计算机系统都属于 MIMD 系统。多处理机是利用系统内的多个 CPU 来并行执行用户的几个程序，以提高系统的吞吐量，或用来进行冗余操作以提高系统的可靠

性。多个处理机(器)在物理位置上处于同一机壳中，有一个单一的系统物理地址空间，每个处理机均可访问系统内的所有存储器。整个系统由统一的操作系统控制，在处理器和程序之间实现作业、任务、程序段和数组元素等各级的全面并行。多处理机系统更多属于紧耦合系统，各处理机既独立又联系紧密。

从目前的实际应用看，MIMD 系统是并行处理计算机系统中的主流，而多处理机系统又在 MIMD 系统中占据了主导地位。根据多处理机系统的组成结构，现有的多处理机系统[7]主要分为对称式共享存储器结构多处理机(symmetric shared-memory multiprocessor，SMP)系统、分布式共享存储器结构多处理机(distributed shared-memory multiprocessor，DSM)系统和大规模并行处理机(massively parallel processor，MPP)系统等。

1) 对称式共享存储器结构多处理机系统

通常 SMP 系统使用微处理器(具有片上或外置高速缓存)作为其处理机，它们经互联网络与一个共享存储器互连。共享存储器可以被所有处理器通过互联网络进行访问，就如同一个单处理器访问它的存储器一样。所有处理器对任何存储单元有相同的访问时间。互联网络可以是单总线、多总线或者是交叉开关。因为对共享存储器(share memory，SM)的访问是平衡的，每个处理器对存储器读/写的机会相等，也有相同的访问速度，故这类系统称为对称多处理机。因为这种对称多处理器的存储器是共享的，所以又称为共享存储器多处理机系统。SMP 系统的体系结构如图 1.4 所示。

对称多处理器的优点是并行度高。因为系统是对称的，每个处理器可等同地访问共享存储器、I/O 设备和操作系统。这种对称使得系统能开拓较高的并行度。但共享存储器的系统总线的带宽是有限的，使系统中的处理器不能太多(一般少于 64 个)，同时因为总线和交叉开关互连，所以也难于扩展。

2) 分布式共享存储器结构多处理机系统

分布式共享存储器多处理机具有独立的局域物理存储器(local memory，LM)。系统将各台处理机所拥有的局部存储器在逻辑上统一编址，形成一个统一的虚拟地址空间，以实现存储器的共享。其存储器采用 cache 目录表来支持 9 分布高速 cache 的一致性。系统中每个节点包含了处理机、存储器、I/O 设备和互联网络接口(network interface connection，NIC)。DSM 系统的体系结构如图 1.5 所示。

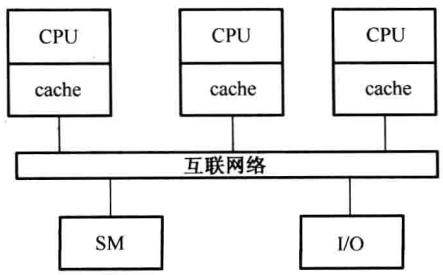


图 1.4　对称式共享存储器结构多处理机系统

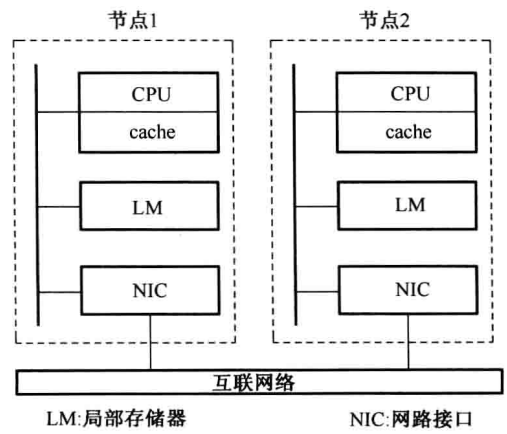


图 1.5　分布式共享存储器结构多处理机系统

将存储器分布到各节点有两个好处：第一，如果大多数的访问是针对本结点的局部存储器，则可降低对存储器和互联网络的带宽要求；第二，对局部存储器的访问延迟低。

DSM 和 SMP 的主要差别是：SMP 中的处理机没有自己的局部存储器，系统的存储器由所有处理机共享；而 DSM 在物理上有各自独立的局域存储器，并进行统一编址，形成一个共享的虚拟存储器。随着处理器性能的迅速提高和处理器对存储器带宽要求的不断提高，在更小规模的多处理机系统中，采用分布式存储器结构也优于采用对称式共享存储器结构。

3）大规模并行处理机系统

MPP 适合中等粒度并行。MPP 可以满足庞大的数据量、异常复杂的运算、极不规则的数据结构和极高的处理速度，这些是高科技应用领域对计算机和通信网络在计算、处理和通信性能上提出的更高的要求，尤其是科学计算中的重大课题都要求计算机系统能提供 3T 性能，即 TeraFLOPS 计算能力、Terabyte 主存储器和 Terabyte/s 输入输出频带宽度；于是伴随着超大规模集成电路(very large scale integration，VLSI)技术和微处理技术的发展，大规模并行处理机就成了 20 世纪 80 年代中期计算机发展的热点。

MPP 一般是指超大型并行计算机系统，大规模并行处理需要有新的计算方法、存储技术、处理手段和结构组织方式。实现的方法是将数百乃至数千个高性能、低成本的 RISC 微处理器用专门的互联网络互连，组成大规模并行处理机。这种处理机可进行中粒度和细粒度大规模并行处理，构成 SIMD 或 MIMD 系统。它具有性价比高、扩展性好的优点。

MPP 一般具有如下特性：

(1) 处理节点采用微处理器；

(2) 系统中有物理上的分布存储器；

(3) 采用高通信带宽和低延迟的互联网络(专门设计和定制的)；

(4) 能扩展至成百上千个处理器；

(5) 它是一种异步的 MIMD 机器，程序由多个进程组成，每个进程都有其私有地址空间，进程间采用传递消息相互作用。

大规模并行处理机系统采用的关键技术主要是超大规模集成电路、可扩展技术和共享虚拟存储技术；其适用的领域主要是科学计算、工程模拟和信号处理等以计算为主的一些重大课题和领域，例如，全球气候预报、基因工程、飞行动力学、海洋环流、流体动力学、超导建模、半导体建模、量子染色动力学和视觉等[8]。

4）多计算机系统

多计算机系统则是由多个独立的计算机组成的，它们通过某种方式连接起来，实现并行处理或计算。一般来讲，多计算机系统属于松耦合系统，构成系统的可以是独立的计算机，多计算机以各处理器经通信链路传递消息为特征。

分布式系统可以认为是由常规网络连接起来的多计算机系统。它拥有多个系统界面，每个节点有自己的操作系统。同时，分布式系统的每台计算机可以是 MMP、SMP、集群或单个计算机。

网络并行集群(cluster)系统是一群以网络技术连接起来的工作站或计算机的组合，适合粗粒度并行。一般以高速网络连接起来的高性能的工作站或计算机组成。集群在工作中所有节点使用单一界面。

多处理器系统/多计算机系统与阵列机的区别在于：阵列机中每个处理器只能执行中央处理器的指令。前者可以执行自己的指令，这样可以达到指令级和任务级的并行。

1.3　并行算法的设计方法

并行算法是适合在并行计算机上实现的算法。一个好的并行算法必须具备充分发挥并行计算机潜在性能的能力。并行算法的目的是用增加空间复杂度来降低时间复杂度。

1.3.1　并行算法的相关概念

1）粒度

粒度(granularity)是各个多处理机可独立并行执行的任务大小的量度[9]。

大粒度：(粗粒度)反映可执行的运算量与程序量很大。通常指基于向量和循环级并行的算法。

中粒度：通常指基于较大的循环级并行。

细粒度：是任务级、语句级的并行粒度。通常指基于子任务级并行的算法，如通常的基于区域分解的并行算法。

并行粒度是个相对的概念，如果处理器的计算功能强大，则有些粗粒度算法也可以被认为是细粒度算法。

2）并行算法的时间复杂性[10-11]

并行算法的时间复杂性用函数 $f(n)$ 表示。其中，n 为输入规模，f 为从第一台处理机开始执行算法直到最后一台处理机完成该算法所执行时间的最大值。

例 1.3.1　n 个数求和，串行算法需要 $n-1$ 次加法，所以它的时间复杂度是 $O(n)$。如果使用 $n/2$ 台处理机，并行执行 n 个数求和，采用如图 1.6 所示的树形结构方式计算，则 n 个数求和需要 $\lceil \log_2 n \rceil$ 步（$\lceil\ \rceil$ 表示向上取整），因此，它的时间复杂度是 $O(\log_2 n)$。

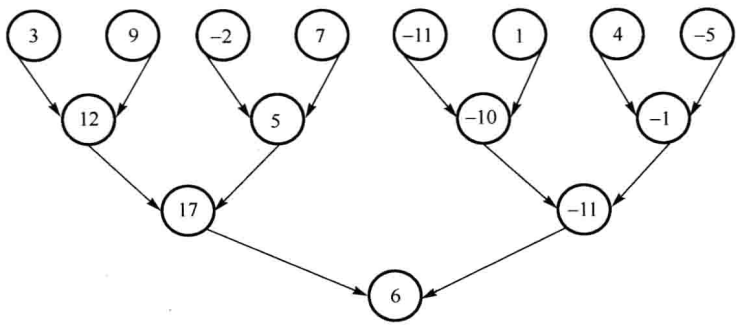


图 1.6　树形方式并行求解 K 个值之和

3）并行算法的耗费

并行算法的耗费 = 时间复杂度×P　　(P 为并行处理机台数)

4）并行算法的加速比[10]

并行加速比是表示采用多个处理器计算速度所能得到的加速的倍数。设 t_{seq} 表示用串行机求解某个计算问题所需的时间，t_P 是用 P 个处理机求解该问题所需的时间。

定义 1.3.1　P 个处理机加速比为[12]

$$S_P = \frac{t_1}{t_P} \tag{1.3.1}$$

式中，t_1 表示用单处理器求解同一问题所需的时间。

定义 1.3.2　并行加速比

$$\hat{S}_P = \frac{t_{\text{seq}}}{t_P} \tag{1.3.2}$$

式中，t_{seq} 表示用串行机求解该问题所需的时间。

一般来说，$\hat{S}_P < S_P$，因为 t_1 是在并行操作系统下用单个处理器计算的结果，t_{seq} 是串行操作系统，而并行操作系统的开销大于串行系统。

5）Amdahl 定律[13]或 Ware 定律

$$S_P = \frac{P}{1 - s + s \times P} \tag{1.3.3}$$

式中，s 为只能串行执行的运算量百分比。其余 P 个处理器可以并行执行的运算量的百分比为 1–s, 忽略通信与同步等由并行引起的开销，$0 \leqslant s \leqslant 1$；对任意多个处理器，有 $S_p \leqslant 1/s$。

6）并行效率

$$E_P = \frac{S_P}{P} \tag{1.3.4}$$

当加速比 S_P 接近于 P 时，效率 E_P 接近于 1。

7）算法并行度

算法并行度是算法中可并行实现部分的操作数。例如，两个 n 维向量 $\boldsymbol{a}$ 和 $\boldsymbol{b}$ 之和：$a_i + b_i$；它们是独立的，并且可以并行执行，该算法的并行度为 n。

8）并行算法成本 $C(n)$

$$C(n) = t_P(n) \cdot P(n) \tag{1.3.5}$$

式中，$t_P(n)$ 为并行算法的运行时间，$p(n)$ 为处理器的个数。

成本的最坏情况是求解一个问题时所需的执行步数。并行算法最优成本是在数量级上等于最坏情况下串行求解此问题所需的执行步数。

9）作业层和进程[13]

对于一个求解问题 P，$(N, \prec)$ 是由输入元、输出元和算子确定的基本算法元的有序集。如果保持前趋关系的次序执行 $(N, \prec)$ 能实现 P 的求解，则称 $(N, \prec)$ 是 P 的一个解法。P 的解法与确定的 $\prec$ 的一个执行次序一起称为 P 的由 $(N, \prec)$ 导出的一个算法。

注意 N 是由算法元组成的有限集。设 $A, B \in N$ 是两个算法元，如果 B 的输入元含有 A 的输出元，则称 A 是 B 的直接前趋，B 是 A 的直接后继。如果 N 中存在前趋相关的算法元，那么 N 与 $\prec$ 一起成为一个有序集 $(N, \prec)$；如果 N 中也存在前趋无关的算法元时，$(N, \prec)$ 是偏序集。

假设 P 是个求解问题，$(N, \prec)$ 是 P 的一个解法。$N = \{A_1, A_2, \cdots, A_N\}, A_j \nprec A_i (i < j)$。下面介绍的是对 $(N, \prec)$ 按照前趋关系进行分层。

(1) 每层包含 $(N,\prec)$ 中若干相连的操作步(或称为时间步)，称为作业层或时间层。设 $(N,\prec)$ 分 n 层，第 i 层有 l_i 个操作步，于是总共有 $l=\sum_{i=1}^{n} l_i$ 操作步，第 $j(1\leqslant j\leqslant l)$ 个操作步由 $(N,\prec)$ 中能在第 j 步执行的各单个操作和相应的操作元组成。

(2) 一个算法元和一个作业层只有全交和不相交两种情况。全交即算法元 A_i 有 l_i 个操作和相应的操作元属于第 i 层。构成 A_i 与第 i 层的交称为一个作业。一个 A_i 至少与一个作业层全交或者与相连的几个作业层全交。每个作业层由若干能并行执行的作业组成，记第 i 层的作业个数为 k_i，作业为 $T_1^{(i)},T_2^{(i)},\cdots,T^{(i)}k_i$。

(3) 任何相邻的两层，一层作业所属的算法元与另一层作业所属的算法元不全相同。

(4) 依层次 1～n 的顺序执行分布在各作业层的所有作业，可实现 P 的求解。

图 1.7 是 $(N,\prec)$ 分层示意图，其中实线块表示算法元，虚线表示分层。每个作业层至少有一个作业。如果 $k_i=1$，则第 i 作业层为串行层；如果 $k_i>1$，第 i 层为并行层。并且对于并行层，如果 k_i 个作业全部彼此相同，则称为 SIMD 层；若存在相互不同的作业则称为 MIMD 层。

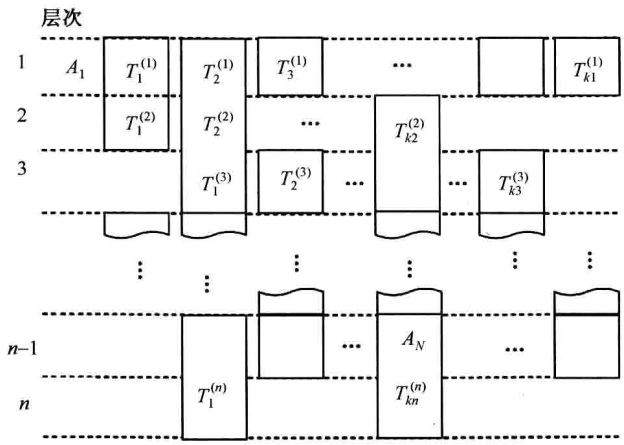


图 1.7　$(N,\prec)$ 分层示意图

综上所述，按照横向作业层分解可将并行算法分为 SIMD 算法和 MIMD 算法。如果一个并行算法的每个并行层都是 SIMD 层，则称为 SIMD 算法；若部分与全部并行层是 MIMD 层，则称为 MIMD 算法。按照纵向进程分解可以将算法分为同步算法和异步算法。所谓同步算法，是并行算法有 $k>1$ 个进程，这些进程的若干操作要等待另一些进程中的操作执行后才能执行。异步算法是每个进程中的操作都不需要等待其他进程的操作执行。SIMD 算法一定是同步算法；异步算法是 MIMD 算法，但是同步算法也可以是 MIMD 算法。同步算法需要进程之间的通信取得同步，这会引起某些进程封锁等待需要信息。异步算法进程之间不需要通信以同步，而是通过读取存放在存储器中的公有变量动态地更新来实现同步。

1.3.2　设计并行算法应注意的问题

1. 问题的并行性分析

对于一个待求解问题，我们需要认真分析其内在的“并行性”，将问题划分为可并行执行的子问题。当然，不同的划分方法，将产生不同方式的并行。