普通高等教育“十三五”规划教材

PUTONG GAODENG JIAOYU “13·5” GUIHUA JIAOCAI

超算、云计算与大数据技术专业教程

多核异构并行计算
OpenMP 4.5 C/C++篇

雷洪 编著

冶金工业出版社
www.cnmip.com.cn

普通高等教育“十三五”规划教材

超算、云计算与大数据技术专业教程

多核异构并行计算
OpenMP4.5 C/C++篇

雷 洪 编著

北 京
冶 金 工 业 出 版 社
2018

内 容 提 要

本书主要介绍了共享内存并行编程语言 OpenMP 的基本原理，采用实例方式讲解在 C/C++语言环境中 OpenMP 并行程序的编写和运行，重点介绍了任务并行、向量化和异构计算等 OpenMP 规范的最新进展。本书面向实际应用，简洁易学，使读者能够亲身感受到并行计算的魅力。

本书可作为涉及高性能计算的理工科高年级本科生和研究生的并行计算课程的教材，也可供从事并行计算研究、设计和开发的教师和工程师参考。

图书在版编目(CIP)数据

多核异构并行计算 OpenMP4.5 C/C++篇/雷洪编著. —北京：冶金工业出版社，2018.4

普通高等教育“十三五”规划教材

ISBN 978-7-5024-7657-1

Ⅰ.①多… Ⅱ.①雷… Ⅲ.①C 语言—并行程序—程序设计—高等学校—教材 Ⅳ.①TP311.11

中国版本图书馆 CIP 数据核字(2018)第 080383 号

出 版 人 谭学余

地 址 北京市东城区嵩祝院北巷 39 号 邮编 100009 电话 (010)64027926

网 址 www.cnmip.com.cn 电子信箱 yjcbs@cnmip.com.cn

责任编辑 刘小峰 美术编辑 吕欣童 版式设计 禹 蕊 孙跃红

责任校对 李 娜 责任印制 李玉山

ISBN 978-7-5024-7657-1

冶金工业出版社出版发行；各地新华书店经销；三河市双峰印刷装订有限公司印刷

2018 年 4 月第 1 版，2018 年 4 月第 1 次印刷

787mm×1092mm 1/16；15.75 印张；382 千字；239 页

49.00 元

冶金工业出版社 投稿电话 (010)64027932 投稿信箱 tougao@cnmip.com.cn

冶金工业出版社营销中心 电话 (010)64044283 传真 (010)64027893

冶金书店 地址 北京市东四西大街 46 号(100010) 电话 (010)65289081(兼传真)

冶金工业出版社天猫旗舰店 yjgycbs.tmall.com

前　言

一个国家的高性能计算技术水平，不仅取决于计算机专业人才的技术开发水平，更取决于科技工作者的整体应用水平。现阶段计算应用和数据处理的状况是：一方面，大多数科技工作者日常应用的计算服务器、多核微机虽然具有较强的计算能力，但设备潜力并没有被充分利用；另一方面，由于工程计算大多数是多物理场的耦合计算问题，虽然计算量相对较大，计算耗时较长，但相对于学习并掌握前沿并行计算方法所需耗费的精力，科技工作者往往更愿意关注自身行业的技术发展，从而放弃并行计算带来的效率的提高。因此，简单、易学的并行计算软件，将有效帮助科技工作者充分发挥计算技术的优势，提高科研效率。

作为异构并行计算平台的杰出代表，我国天河一号、天河二号和神威·太湖之光先后蝉联 Top500 世界第一。我国迈进了异构计算时代！虽然我国超级计算机的处理器核心数已经接近 10 万个处理器核，计算能力达到 10 亿亿次每秒。但是，很少有能够利用全部处理器和 GPGPU 核并取得亿亿次性能的突破性应用程序出现。因此，将全机的 CPU+GPGPU 异构并行充分应用于工程技术领域，是当前国际计算科学领域需要解决的一个难题。

当前，PC 机都普遍装备 GPU（独立显卡），使得 CPU/GPU 这种异构计算平台随处可见。但这些平台大多用于大型游戏等娱乐项目，很少用于工程计算。同时，基于 GPU 编程的很多工具包是免费的。因此，采用较低的成本搭建的 CPU/GPU 异构并行平台来满足中小规模的工程计算需求切实可行。

计算机体系结构的演变，都会伴随并行程序设计环境和语言的进化。在众多并行语言中，并行程序开发环境 MPI 和 OpenMP 已经成为被广泛接受和使用的工业标准。随着国际主流体系架构逐渐转变为异构混合并行，OpenMP 于 2013 年推出了 4.0 规范，从而实现了从多核并行计算到异构并行计算的进化。

作者长期从事钢铁冶金过程的数值模拟，一直关注并行计算技术的发展。由于现有相关著作大多数是由计算机专家编著，著作中大量的计算机专业术语难以被工科专业学生和广大科技工作者所理解。而且，这些著作多以 MPI 作为

基本的并行计算语言，MPI 又存在学习困难、调试困难等诸多问题，这进一步加大了其在工科专业普及的难度。为此，作者编写本书，目标是提供一本多核、异构并行计算方面的实用参考书，为我国并行计算普及工作略尽绵薄之力。

本书分析了当前流行的并行计算技术，从中遴选出适合大多数工程科技人员应用的 OpenMP 并行计算技术，从实例入手阐明程序运行过程，清晰而简洁地展示 OpenMP 并行计算原理、编程特点和方法。希望本书能够帮助工程技术人员了解并掌握并行计算知识，并在实际工作和学习中顺利运用，避免作者在获取这些经验时所犯过的类似错误。

目前，绝大多数大学均将 C/C++作为本科生必修的计算机编程语言。鉴于此，本书对作者已出版的基于 OpenMP 3.0 规范的《多核并行高性能计算 OpenMP》一书进行了补充和完善。为了避免内容重复，删除了原书第 9 章和第 10 章以及附录，将原书的 Fortran 语言改为 C++语言，并增加了 OpenMP 4.5 规范的内容。实际上，OpenMP 编程在 Fortran 语言和 C/C++语言方面是互通的，本书的大多数程序稍加修改即可应用于 Fortran 语言。同时建议读者在使用本书的过程中，参阅《多核并行高性能计算 OpenMP》的第 9 章和第 10 章以及附录来加深对 Linux 环境、程序的调试、编译和优化的理解。

全书共分为 10 章。主要内容如下：

第 1 章概述并行计算的发展历程，介绍并行计算和异构计算等相关概念，回顾 OpenMP 的发展历史。

第 2 章阐述 OpenMP 的语法，掌握 C/C++程序编写、编译和执行的完整过程。

第 3 章阐明数据环境，研究共享变量和私有变量，全局变量和局部变量的联系和差异。

第 4 章讲解并行区域的构造方法，探讨线程组和子线程数量的确定方式。

第 5 章剖析不同并行结构的差异，实现负载平衡。

第 6 章揭示多线程同步的不同机制，防止数据竞争的出现。

第 7 章明晰运行环境要素，探究锁的操作方式，避免死锁的发生。

第 8 章解析任务的构建、调度和执行，领悟非规则循环和递归的并行特征。

第 9 章了解向量化计算原理，体验并行向量化计算。

第 10 章讲述异构计算的特征，实现多种计算硬件的异步执行。

本书各章节相互独立，部分内容略有重复，供读者根据需要选择阅读。阅读本书之前需对C/C++语言编程有所了解，才能深入体会C/C++语言在并行计算方面的优势。

阅读本书时，如果您对高性能计算感兴趣，建议阅读第1章。如果您是初学者，那么需要关注第1章中编译器部分和第2章简单并行程序的运行。本书的核心部分是第3~10章，本书的精华在第8~10章。其中第3~6章是多核并行的编程基础，学习之后基本能够编写大部分的OpenMP程序；第7章锁操作十分复杂；第8章是OpenMP3.0的精华，也是OpenMP学习的难点，主要用于非规则循环和递归的并行；第9章向量化相对简单，但需具备部分硬件知识；第10章给出了以GPU为代表的异构计算解决方案。

在本人学习和应用OpenMP的过程中，得到了许多老师和同行的帮助，在此特向东北大学计算中心刘小锋老师、大连理工大学张永彬博士、东网科技有限公司胡许冰老师和赵学彬老师、英特尔亚太研发有限公司黄飞龙工程师和周姗工程师以及IBM公司研究中心的Arpith Jacob工程师表示由衷的感谢。特别感谢，牛宏硕士搭建了OpenMP计算平台，刘玉强硕士协助调试了本书的部分程序。

在本书的撰写过程中，参考了其他同行的文章、课件和研究资料，还引用了有关专家和学者的工作。在此一并感谢并向他们所做的工作表示深深的敬意。本书中的所有程序代码可登陆冶金工业出版社网站www.cnmip.com.cn下载。

本书的出版得到了国家自然科学基金委员会-宝钢集团有限公司钢铁联合研究基金（U1460108）和中央高校基本科研业务专项资金（N170906004）的资助。在书稿准备与出版过程中，冶金工业出版社的编辑人员也给予了大力支持，在此一并表示感谢。

尽管作者在编写本书的过程中投入了大量的精力，但受计算机专业水平所限，书中难免存在不当之处，恳请专家和读者给予批评指正。

雷洪于东北大学
2018年2月

目　录

1　并行计算概论 …… 1

1.1　并行计算机的种类 …… 2

1.1.1　多核 CPU …… 2

1.1.2　GPU …… 2

1.1.3　CPU 与存储器的连接方式 …… 3

1.1.4　数据的通信方式 …… 4

1.1.5　常见的并行计算硬件系统 …… 5

1.1.6　指令和数据之间的工作方式 …… 5

1.2　并行计算 …… 5

1.2.1　并行计算、高性能计算与超级计算 …… 5

1.2.2　并行处理技术 …… 6

1.3　高性能并行计算特征 …… 7

1.4　并行编程模式 …… 7

1.4.1　共享内存模式 …… 8

1.4.2　消息传递模式 …… 8

1.4.3　数据并行模式 …… 9

1.4.4　异构计算 …… 10

1.5　OpenMP 和 MPI 的特点 …… 13

1.6　并行计算中常用概念 …… 14

1.6.1　并发、并行和并行计算 …… 14

1.6.2　程序、线程、进程和超线程 …… 15

1.6.3　单核编程和多核编程 …… 16

1.6.4　线程绑定 …… 17

1.6.5　多线程编程和多进程编程 …… 17

1.6.6　并行算法评价 …… 18

1.7　OpenMP 多核编程 …… 20

1.7.1　OpenMP 历史 …… 20

1.7.2　OpenMP 特点 …… 21

1.8　Linux 系统 …… 22

1.9　常用编译器与 OpenMP …… 22

1.9.1　Windows 环境下 Visual Studio 2008 命令行界面的编译和执行 …… 23

1.9.2　Windows 环境下 Visual Studio 2008 菜单界面的编译和执行 …… 24

1.9.3 Windows 环境下 MinGW-W64 的安装 …… 25
1.9.4 Linux 环境下 Intel C/C++编译器 icc 的安装 …… 26
1.9.5 在 Windows 系统下远程操作服务器端 Linux 环境下的编译器 …… 26
1.9.6 Windows 和 Linux 环境下常用命令和系统资源检查 …… 27
1.10 小结 …… 27
练习题 …… 28

2 OpenMP 编程简介 …… 30

2.1 编译指导语句 …… 30
2.2 并行执行模式 …… 31
2.2.1 编译指导语句格式 …… 31
2.2.2 主要指令 …… 32
2.2.3 主要子句 …… 33
2.2.4 指令的作用域 …… 34
2.2.5 指令和子句的配套使用 …… 35
2.3 头文件 …… 36
2.4 常用库函数 …… 37
2.5 最简单的并行程序 …… 37
2.6 小结 …… 42
练习题 …… 42

3 数据环境 …… 43

3.1 子句 private、子句 shared 和子句 default …… 43
3.2 子句 firstprivate 和子句 lastprivate …… 49
3.3 指令 threadprivate …… 52
3.4 子句 copyin 和子句 copyprivate …… 58
3.5 子句 reduction …… 62
3.6 数据竞争 …… 67
3.7 伪共享 …… 69
3.8 小结 …… 71
练习题 …… 72

4 并行控制 …… 73

4.1 指令 parallel …… 73
4.2 设定线程数量 …… 76
4.3 默认模式 …… 77
4.4 静态模式 …… 77
4.5 动态模式 …… 78
4.6 嵌套模式与 num_threads 子句 …… 80

4.7　条件并行子句 if …… 84
4.8　动态设置并行循环的线程数量 …… 86
4.9　小结 …… 86
练习题 …… 86

5　并行构造 …… 87

5.1　负载平衡 …… 88
5.1.1　静态负载平衡 …… 89
5.1.2　动态负载平衡 …… 89
5.2　依赖关系 …… 90
5.2.1　循环依赖 …… 90
5.2.2　内存依赖 …… 95
5.2.3　任务依赖 …… 96
5.3　指令 for …… 98
5.3.1　单重循环 …… 100
5.3.2　嵌套循环 …… 102
5.3.3　循环工作量的划分与调度 …… 104
5.3.4　子句 collapse …… 112
5.4　指令 sections …… 115
5.5　指令 single …… 117
5.6　合并的并行工作共享结构 …… 119
5.7　小结 …… 119
练习题 …… 119

6　线程同步 …… 120

6.1　互斥锁机制 …… 121
6.2　事件同步机制 …… 121
6.3　指令 barrier …… 121
6.4　指令 nowait …… 124
6.5　指令 master …… 125
6.6　指令 critical …… 128
6.7　指令 atomic …… 129
6.8　指令 ordered …… 132
6.9　指令 flush …… 135
6.10　小结 …… 139
练习题 …… 139

7　运行环境 …… 140

7.1　环境变量 …… 140

7.1.1 OMP_DYNAMIC …… 141
7.1.2 OMP_SCHEDULE …… 141
7.1.3 OMP_NUM_THREADS …… 141
7.1.4 OMP_NESTED …… 141
7.1.5 OMP_STACKSIZE …… 142
7.1.6 OMP_WAIT_POLICY …… 142
7.1.7 OMP_PROC_BIND …… 142
7.1.8 环境变量的设置方法 …… 142
7.1.9 段错误和环境变量的应用 …… 143
7.2 库函数 …… 144
7.2.1 运行环境操作函数 …… 144
7.2.2 OpenMP 时间函数 …… 146
7.2.3 热点分析 …… 148
7.2.4 锁函数 …… 151
7.3 小结 …… 156
练习题 …… 156

8 任务 …… 157

8.1 任务简介 …… 157
8.1.1 任务结构 …… 158
8.1.2 任务类别 …… 159
8.2 任务的创建 …… 163
8.2.1 指令 parallel 和子句 single …… 164
8.2.2 指令 for …… 166
8.2.3 指令 sections …… 168
8.2.4 包含任务 …… 169
8.2.5 递归 …… 171
8.3 任务调度原则 …… 174
8.3.1 栅障 barrier …… 176
8.3.2 指令 taskwait …… 176
8.3.3 指令 taskgroup …… 177
8.3.4 指令 taskyield …… 179
8.3.5 子句 if …… 180
8.4 任务的执行和完成 …… 181
8.5 任务的数据环境 …… 181
8.5.1 共享变量和私有变量 …… 182
8.5.2 任务与对栈数据的引用 …… 184
8.5.3 全局变量 …… 187
8.6 任务依赖子句 depend …… 188
8.7 指令 taskloop …… 191

8.8 小规模任务 …… 193
8.8.1 子句 final …… 193
8.8.2 子句 mergeable …… 194
8.9 子句 priority …… 195
8.10 小结 …… 195
练习题 …… 195

9 向量化 SIMD …… 196

9.1 SIMD 的发展 …… 196
9.2 代码风格 …… 198
9.3 循环的串行向量化指令 simd …… 199
9.3.1 子句 aligned …… 204
9.3.2 子句 safelen …… 204
9.3.3 子句 simdlen …… 205
9.3.4 子句 linear …… 205
9.4 循环的并行向量化指令 for simd …… 207
9.5 函数的向量化指令 declare simd …… 207
9.5.1 子句 inbranch 和 notinbranch …… 208
9.5.2 子句 uniform …… 210
9.6 小结 …… 212
练习题 …… 212

10 异构计算 …… 213

10.1 目标设备查询 …… 213
10.2 控制权的移交指令 target …… 215
10.2.1 子句 device …… 216
10.2.2 子句 map …… 216
10.2.3 子句 defaultmap …… 218
10.2.4 子句 if …… 220
10.2.5 指令 target data …… 221
10.2.6 指令 target enter data 和 target exit data …… 224
10.2.7 指令 target update …… 225
10.2.8 指令 declare target …… 227
10.3 线程组群指令 teams …… 228
10.4 工作共享指令 distribute …… 230
10.5 异步执行和依赖性 …… 232
10.6 OpenMP 并行执行模式比较 …… 234
10.7 小结 …… 237
练习题 …… 237

参考文献 …… 238

1 并行计算概论

并行计算的优点是具有强大的数值计算和数据处理能力，能够被广泛地应用于国民经济、国防建设及科技发展中具有深远影响的重大课题，如石油勘探、地震预测、天气预报、新型武器设计、天体和地球科学等[1~5]。并行计算离不开硬件和软件两大系统，如图1-1所示。并行计算系统既可以是专门设计的、含有多颗CPU的超级计算机，也可以是以某种方式互连的若干台独立计算机构成的集群。在这样的背景下，对编程人员提出了更高的要求。硬件系统中CPU、GPU和存储器等的连接形式决定了科技人员采用的并行方式。绝大多数科技人员使用的是个人计算机、服务器和工作站，这一类的硬件系统均可采用OpenMP进行并行。通常，编程人员需要对现有的串行程序进行修改，对CPU之间、CPU与GPU之间的通信和控制进行协调从而解决并行程序所带来的数据竞争、同步等潜在问题，实现并行程序的高稳定性和高并行加速比。

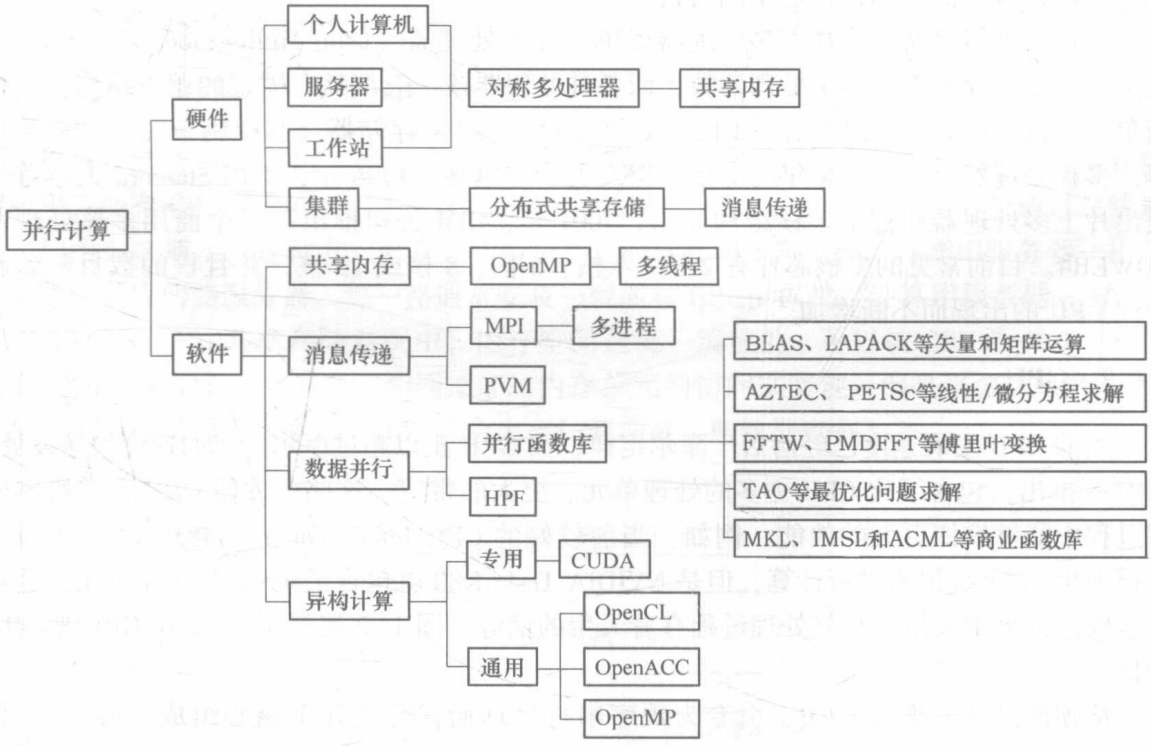

图1-1 并行计算所需硬件和软件

需要指出的是，工程中的计算问题不是固定不变的，而是随现代工业的不断进步而不断发展演化的。现代工业的发展要求工程计算朝着多物理场耦合、跨尺度计算、增加网格扩大计算规模等方向发展，即程序中需要并行计算的代码比例越来越大，因此对并行计算需求也越来越强烈。

1.1 并行计算机的种类

并行计算机通常包含多颗自带高速缓存（Cache）的CPU，而且这些CPU需要一定数量的内存才能工作；同时，这些CPU通常需要借助于网络传递数据从而实现CPU之间的协同工作；数据的显示则需要借助于显卡（GPU）。因此，并行计算机通常可分为四大部件：CPU、GPU、存储器和网络。通常，并行计算机的种类可通过CPU与存储器的连接方式、数据的通信方式以及指令和数据之间的工作方式来进行划分[3,6~8]。

1.1.1 多核CPU

计算机运算速度的提高能够有效地提高科研人员的工作效率。在过去的几十年里，个人计算机CPU的主频一直依照摩尔定律发展。但是当单核CPU的主频达到3GHz以后，过高的功耗和高散热问题成为瓶颈限制了CPU频率的提高。虽然单核CPU的性能可能达到了极限4GHz，但是多媒体、大规模科学计算等多个应用领域却对处理器性能不断地提出了更高的需求。现代电子工业的发展使芯片上晶体管的密度仍可以不断地增加，于是各主流处理器厂商将产品战略从提高芯片的时钟频率转到了多内核的研发。因此，多核处理器的出现是应用需求和科技进步的时代产物。

多核处理器，又称为片上多处理器或单芯片多处理器（Chip Multi-Processor，CMP），是指在一个芯片上集成多个处理器核，而各种处理器核一般都具有固定的逻辑结构：指令级单元、执行单元、一级缓存（L1）、二级缓存（L2）、存储器及其控制单元、总线接口等。多核处理器仅有二十余年的历史，但发展十分迅猛。1996年，美国Stanford大学首先提出片上多处理器和首个多核结构原型。2001年，IBM公司推出第一个商用多核处理器POWER4。目前常见的多核芯片有2核、4核、6核、8核或16核，并且核的数目随着新一代CPU的出现而不断增加[2~4]。

1.1.2 GPU

当前CPU发展速度已经落后于摩尔定律，而GPU正以超过摩尔定律的速度快速发展。GPU一推出就包含了比CPU更多的处理单元、更大的带宽。这些条件使GPU在多媒体处理过程中能够发挥更大的效能。例如：当前较好的CPU Intel Xeon E7-4850 v4有16核，可模拟出32个线程来进行计算，但是NVIDIA Tesla K80就包含了4992个处理单元，这对于多媒体计算中大量的重复处理过程有着天生的优势。图1-2展示了CPU和GPU架构的对比。

从硬件设计上来讲，CPU由专为顺序串行处理而优化的几个核心组成。另一方面，GPU则由数以千计的更小、更高效的核心组成，这些核心专为同时处理多任务而设计。

需要注意的是，GPU与经常提到的显卡是有区别的。GPU的范围更大：显卡一定是GPU，但GPU不一定是显卡。目前，NVIDIA是生产GPU最重要的公司。自1999年以来，NVIDIA系列显卡性能不断更新，其在硬件设计上或者命名方式上也有很多的变化，主要内容如下：

（1）GPU架构：Tesla、Fermi、Kepler、Maxwell、Pascal。

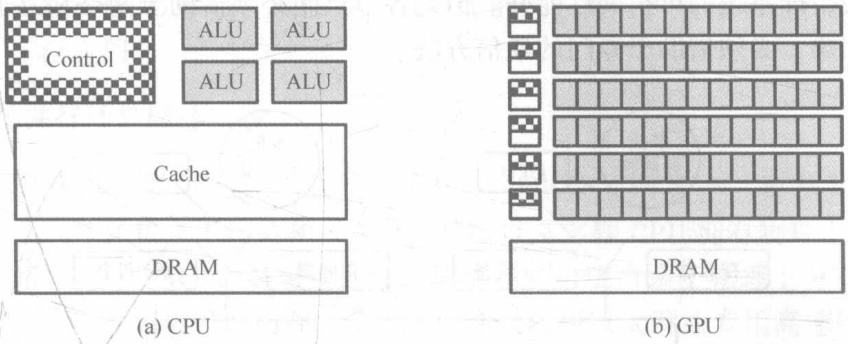

图 1-2 CPU 和 GPU 的架构

（2）芯片型号：GT200、GK210、GM104、GF104 等。

（3）显卡系列：GeForce、Quadro、Tesla。

GPU 架构是指硬件的设计方式，例如流处理器簇中有多少个核心、是否有 L1 或 L2 缓存、是否有双精度计算单元等。每一代的架构代表一种设计思想，而芯片则是对上述思想的实现。而显卡系列在本质上并没有什么区别，只是 NVIDIA 根据主要用途分为三类：GeForce主要用于家庭电脑，Quadro 用于工作站，而 Tesla 用于服务器。Tesla 的 K 型号卡是专门为高性能科学计算而设计，比较突出的优点是双精度浮点运算能力高。需要注意的是 Tesla 系列没有显示输出接口，它专注于数据计算而不是图形显示。

1.1.3 CPU 与存储器的连接方式

根据存储器与 CPU 的连接方式可分为共享存储系统和分布存储系统[2,9]。在共享存储系统中，所有 CPU 共同使用同一个存储器和输入输出（I/O）设备，并且一般通过总线连接，如图 1-3 所示。这种方式适合于实验室常见的计算服务器系统。计算用服务器一般为两路服务器或四路服务器。每一路通常安装一颗多核 CPU，因此，计算用服务器一般有 2 颗或 4 颗 CPU。在共享存储系统中，内存空间是统一编址的，可以被 CPU 所共享；CPU 之间数据通信依靠 CPU 对具有相同地址的内存单元的访问来实现。但是当多颗 CPU 对同一地址的内存单元进行读写操作时，会出现访问冲突，即数据竞争。

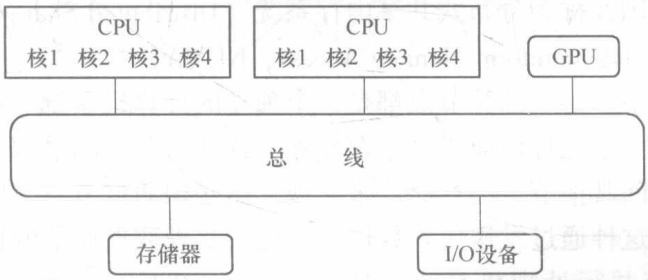

图 1-3 共享存储系统的基本结构

图 1-4 给出了分布存储系统的结构。通常，每颗 CPU 均具有各自的存储器和输入输出设备，它们组成了一个计算节点；多个计算节点通过网络相互连接形成了分布存储系统。这种方式适合于实验室常用的集群系统。由于每颗 CPU 的计算结果都有自己的存储器，因此可以保证 CPU 访问存储器速度，不会出现访问冲突。另外，在网络中增加计算

节点比较方便，即系统的可扩展性能好。但是各节点间必须借助于网络相互通信，因此数据通信比较困难，必须借助于专门的通信方法。

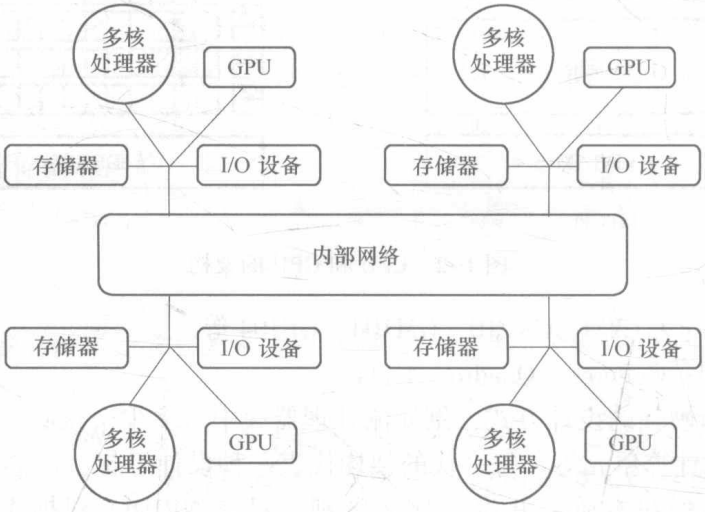

图 1-4 分布存储系统的基本结构

1.1.4 数据的通信方式

根据数据通信方式，可以将并行计算系统分为共享地址空间系统和消息传递系统两大系统。

在共享地址空间系统中，存储器的地址空间是统一的，因此可称为单地址系统或共享存储多处理器（Shared Memory Multiprocessors，SMM）系统。根据 CPU 与存储器的连接方式可将共享存储器多处理器系统进一步进行分类。如果存储器是集中式的，那么所有的处理器能够以相同的速度访问内存，这种系统称为对称共享内存多处理器系统（Symmetric Shared-memory Multiprocessors 或 Symmetric Multiprocessors，SMP）或均匀存储访问系统（Uniform Memory Acess，UMA）。如果内存是分布式的，那么 CPU 访问内存的速度就与内存的位置有关。毫无疑问，CPU 访问本地内存的速度最快。换言之，由于处理器访问内存的速度是不一样的，因此称为分布式共享内存系统（Distributed Shared-Memory，DSM）或非均匀存储访问系统（Nonuniform Memory Access，NUMA）。

在消息传递系统中，每个计算节点都是一个独立的计算机系统，而每个节点的存储器均单独编址，因此同一个地址对应于多个存储器。这样，节点间数据的传递不能通过本地节点的处理器直接访问其他节点的存储器来实现，而必须通过节点之间相互发送含有数据信息的消息来实现。这种通过发送包含数据的消息来实现数据通信的系统称为消息传递系统。它可分为大规模并行处理机系统（Massively Parallel Processor，MPP）和集群系统（Cluster）。大规模并行处理机系统是指由几百或几千台处理机组成的大规模并行计算系统。此系统的很多硬件设备是专门设计制造的，它的网络传输速度较高但扩展性稍差，开发十分困难，通常标志着一个国家的综合实力。而集群系统是相互连接的多个同构或异构的独立计算机的集合体，节点之间通过高性能互联网相连接。每个节点都有自己的存储器、I/O 设备和操作系统，可以作为单机使用；并行任务的完成则需通过各节点之间的相

互协同工作来完成。近10年来，集群系统以高性价比、高可扩展性和结构的灵活性在多个领域得到了广泛应用。

1.1.5 常见的并行计算硬件系统

目前在实验室比较常见的计算系统，大体上可以分为两类：一类是共享内存系统（SMP），例如个人计算机、工作站和服务器，其特点是多颗CPU拥有物理上共享的内存；一类是分布存储系统（DMP），如集群系统，其特点是系统由多个物理上分布的计算节点组成，每个计算节点拥有自己的内存，节点之间通过高速以太网或专用高速网络连接。它们各自的特点如表1-1所示。

表1-1 实验室常见的计算系统特点及并行计算方式

计算系统	个人计算机	服务器和工作站	集 群
硬件系统	单一主机，集成或独立GPU，单颗有多个核心的CPU	单一主机，集成或独立GPU，多颗有多个核心的CPU	多台主机，每台主机有集成或独立GPU和一颗或多颗CPU
操作系统	单一	单一	多个
高性能计算系统	对称多处理器（SMP）	对称多处理器（SMP）	分布式共享存储（DMP）
常用并行模式	共享内存模式（如OpenMP）	共享内存模式（如OpenMP）	消息传递模式（如MPI）

1.1.6 指令和数据之间的工作方式

根据指令和数据之间的工作方式可分为四大类[2,3]。第一类是单指令流单数据流系统（Single Instruction Stream Single Data Stream，SISD），具有一个单处理器核的个人计算机可归为此类。第二类是单指令多数据流系统（Single Instruction Stream Multiple Data Stream，SIMD），它是指在多颗CPU上运行相同的指令，但是每颗CPU所处理的数据对象并不相同。第三类是多指令单数据流系统（Multiple Instruction Stream Single Data Stream，MISD），在实际应用中，这种系统是不存在的。第四类是多指令多数据流系统（Multiple Instruction Stream Multiple Data Stream，MIMD），它是指每颗CPU上执行的指令和处理的数据各不相同。目前常见的多核个人计算机和集群计算机可归为此类。

1.2 并行计算

1.2.1 并行计算、高性能计算与超级计算

并行计算（Parallel Computing）、高性能计算（High Performance Computing，HPC）和超级计算（Supercomputing，简称超算）这三者的概念是不同的，它们之间的相互关系如图1-5所示。

并行计算是指利用多个CPU（或多个CPU核）的协同来解决同一个问题，即在计算任务中存在多核心并行即可视为并行计算。并行计算的实质是将一个待求解的问题分解成若干个子问题，各个子问题均由独立的CPU同时进行计算。这样，各CPU（或CPU核）

图 1-5 并行计算、高性能计算和超级计算的关系

在并行计算过程中往往需要频繁地交换数据，具有细粒度和低开销的特征。并行计算的重要特征是短的执行时间和高的可靠性，它主要是指以高精度浮点运算为主的科学计算。

高性能计算要求针对所使用的硬件环境（多核和 GPU），通过向量化、提高 Cache（缓存）命中率、采用多核心同时执行计算任务。高性能计算所面对的计算环境为 10~100 量级的 CPU 核心或 GPU。这是普通高校和研究中心中常见的工作站或服务器的标准硬件配置，并行加速比为 10~100 量级。

超算（超级计算）指在少量节点的高性能计算性能不足以满足实验计算量和运算规模需要，而必须在超级计算机或巨型机上解决的大型、复杂运算。不同专业、不同领域对超级计算的标准不同，很难给出一个超级计算的阈值。本书中仅给出一个参考值：并行计算中使用超过 128 个当代计算节点或并行加速比达到 1000 量级。

在实际应用中，并行计算与分布式计算是十分相似的概念。它们之间的界限十分模糊。分布式计算的目的是提供方便。这种方便性主要体现在可用性、可靠性以及物理分布三个方面。在分布式计算中，CPU（或 CPU 核）之间并不需要频繁地交换数据，具有粗粒度的特征。分布式计算的重要特征是长的正常运行时间，它主要是指以整数运算为主同时具有少量简单浮点运算的事务处理型计算。

1.2.2 并行处理技术

早期的计算机采用的是串行处理，计算机的各个操作只能串行地完成，即任一时刻只能进行一个操作。而并行处理能够同时进行多个操作，从而极大地提高了计算机的速度。

计算机的并行处理技术，概括起来主要有三种形式：

（1）时间并行。时间并行是指时间重叠，即多个处理过程在时间上相互错开，轮流重叠地使用同一套硬件设备的各个部分，从而加快硬件周转而赢得速度。时间并行的实现方式是采用流水处理部件，这是一种非常经济实用的并行技术，能保证计算机系统具有较高的性能价格比。目前的高性能计算机几乎均使用了流水技术。

例如，食品工厂生产食品的步骤可分为：

1）清洗：将食物冲洗干净。

2）消毒：将食物进行消毒处理。

3）切割：将食物切成小块。

4）包装：将食物装入包装袋。

如果不采用流水线操作，当一个食品完成上述四个步骤后，下一个食品才能开始进行处理，耗时长，效率低；如果采用流水线技术，就可以同时处理四个食品。这就是并行算法中的时间并行：在同一时间启动两个或两个以上的操作，可以大大提高计算性能。