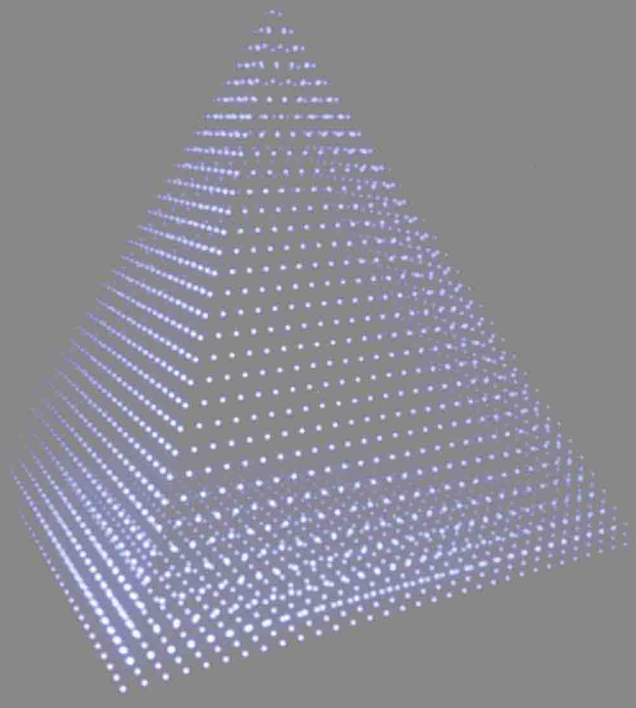
HZ BOOKS 华章IT

ELSEVIER 爱思唯尔

汇集多核/众核时代的主流开发工具，学会选择并综合运用不同的平台和方法

面向真实需求设计和优化代码，在实践中掌握并行计算的编程之道

高性能计算技术丛书



Multicore and GPU Programming

An Integrated Approach

多核与GPU编程

工具、方法及实践

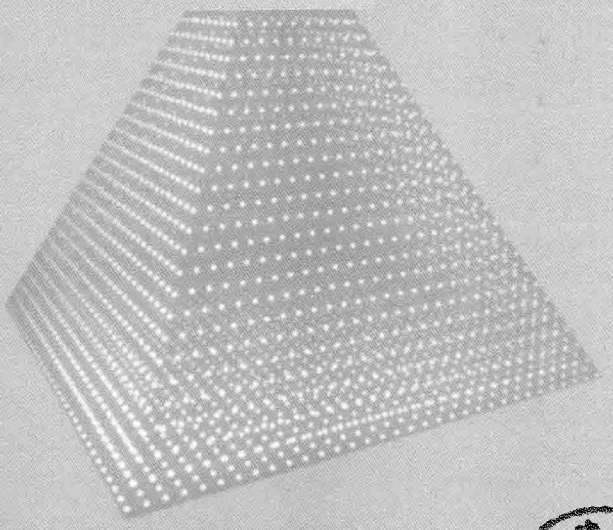
[阿联酋] 杰拉西莫斯·巴拉斯（Gerassimos Barlas）著

张云泉 贾海鹏 李士刚 袁良 等译

机械工业出版社

China Machine Press

高性能计算技术丛书



Multicore and GPU Programming
An Integrated Approach

多核与GPU编程

工具、方法及实践

[阿联酋] 杰拉西莫斯·巴拉斯（Gerassimos Barlas）著

张云泉 贾海鹏 李士刚 袁良 等译

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

多核与 GPU 编程：工具、方法及实践 /（阿联酋）杰拉西莫斯·巴拉斯（Gerassimos Barlas）著；张云泉等译 . —北京：机械工业出版社，2017.2
（高性能计算技术丛书）
书文原名：Multicore and GPU Programming: An Integrated Approach

ISBN 978-7-111-55768-5

I. 多… II. ①杰… ②张… III. 计算机图形学 IV. TP391.41

中国版本图书馆 CIP 数据核字（2017）第 001136 号

本书版权登记号：图字：01-2015-2809

多核与 GPU 编程：工具、方法及实践

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码：100037）
责任编辑：谢晓芳　　责任校对：董纪丽
印　　刷：三河市宏图印务有限公司　　版　　次：2017 年 2 月第 1 版第 1 次印刷
开　　本：186mm×240mm　1/16　　印　　张：34.5
书　　号：ISBN 978-7-111-55768-5　　定　　价：129.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88379426　88361066　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzit@hzbook.com

The Translator's Words 译 者 序

由于功耗墙、散热墙等因素的限制，单纯提升单核处理器的性能已经越来越困难，因此多核 / 众核架构成为计算机体系结构发展的重要趋势。然而，多核 / 众核处理器在带来更高计算能力的同时，也增加了并行软件开发、分析和优化的难度。随着多核 / 众核处理器的普及以及并行计算集群应用的日益广泛，如何编写正确、高效的并行程序已经成为软件开发人员面临的一大挑战，同时并行程序的调试、剖析也日益成为并行软件开发的难题。并行程序开发和优化的本质是实现算法特征向底层硬件架构特征的高效映射，这就对并行软件开发人员提出了更高的要求，他们不仅要了解算法特征，还要对底层架构有着清晰的认识。这无疑增加了并行软件开发的难度。

本书作者 Gerassimos Barlas 博士是阿联酋沙迦美国大学计算机科学与工程系的教授，其研究方向包括并行算法及并行软件的设计与开发、负载均衡模型框架研究等。Gerassimos Barlas 博士具有 10 年以上的并行软件开发和教学经验，并且在并行与分布式系统的可分负载理论等领域非常活跃。

本书从并行软件的实现、调试、优化和剖析四个方面，详细讨论了当前主要的并行计算关键技术，包括 CPU 多核编程、分布式内存编程、GPU 编程、负载均衡等。同时针对并行软件的开发效率，本书也详细介绍了 Qt 线程、Thrust 和 Boost MPI 等库和工具。面对当今异构计算平台的架构特点，书中既讲解了选择合适的工具开发高性能并行软件的步骤和方法，同时提供了大量的实际案例，更加深入地讨论不同编程和优化技术的应用方法。通过阅读本书，读者可以学习：如何使用库或者指令制导方式创建多核应用，如何使用 MPI 开发运行在分布式内存系统上的应用程序，如何使用 CUDA 开发高性能 GPU 程序，如何实现负载均衡，以及如何针对目标多核平台进行程序剖析和调试等。总之，本书作者根据自己多年的并行编程

经验，从开发优秀并行软件的本质出发，以通俗易懂的方式对最为关键的基本知识和技术进行了细致讲解。本书既可成为并行软件初学者的参考书籍，也可供具有一定并行编程经验的软件开发人员参考。

由于时间仓促，而且书中某些术语目前业界没有统一译法，所以我们对一些术语采取了保留其英文名称的方法。书中翻译方面的错误和不妥之处，恳请广大读者不吝批评指正。

译者

Preface 前　言

多核架构出现在21世纪的第一个10年里，给并行计算带来了勃勃生机。新平台需要新方法来进行软件开发，其中一个新方法就是把工具和工作站网络时代的惯例同新兴软件平台（如CUDA）相结合。

为满足这种需求，本书将介绍目前主流的工具和技术，不仅是各自独立的工具和技术，更重要的是将它们相互结合。书中会提供多平台和程序设计范例（如消息传递和线程）高效结合的实例。顾名思义，“Hybrid”（混合）计算，是高性能计算的一个新趋势，针对百亿亿级的性能需求，使软件有可能扩展到数百万的线程上。

所有章节都包含丰富的示例和实际问题，并比较了不同的设计脚本，重点在于如何使其实际运作起来。那些能使得高效的软件开发区别于徒劳无功的软件开发训练的所有细节，都以有序的形式呈现出来。

本书介绍了从20世纪90年代继承而来的最新的先进工具（例如OpenMP和MPI标准），并包括更前沿的平台，如具有复杂线程管理功能的Qt库，以及具有在不同多核架构（包括CPU和GPU）上配置相同软件功能的Thrust模板库。

我不可能面面俱到地给出多核开发中的所有可用工具。即使是POSIX线程之类的一些行业标准，书中也并未涉及。

本书不仅旨在介绍主流范型（范围从OpenMP的串行代码半自动并行化到用来巩固MPI的显式通信“plumping”）的实例，还要讲解高效的多核软件开发背后的原理，并指导读者进行实践。

本书内容

本书可以分成如下几个逻辑单元，虽然书中没有明确地这样区分。

多核软件的设计概述概述：第 1 章介绍多核硬件，讨论了一些具有影响力的体系结构范型示例。第 1 章也介绍了加速比和效率，在评估多核和并行软件时这些是基本的度量方式。1.5 节告诉读者如何从多核和众核硬件激动人心的新进展中预测出什么是人们所期待的方法。

第 2 章讨论可应用于并行和多核软件开发的方法论与设计模式，包括工作分解模式和程序结构模式。

共享内存编程：讨论了两种不同的共享内存并行编程方法——显式并行化和隐式并行化。在显式方面，第 3 章覆盖了线程和两种最常用的同步机制——信号和 monitor。对于通常遇到的设计模式（如生产者 – 消费者模式和读者 – 写者模式），均给出了详细解释并应用于一系列示例中。

在隐式方面，第 4 章介绍了 OpenMP 标准，该标准的设计使得我们能以最少的工作量将现有串行代码并行化。它可使开发时间大大缩短。该标准还有其他的好处，如解决了循环间存在的依赖。

分布式内存编程：第 5 章介绍了分布式内存并行编程的事实标准——消息传递接口（MPI）。MPI 同多核编程相关，因为它旨在把程序从一个共享内存多核机器上扩展到一个具有上百万节点的超级计算机上。就其本身而言，MPI 能够为利用多核机器的多重分解提供基础，以作为独立的虚拟平台。

这部分涉及的特征包括点对点通信和集合通信，也包括单边通信。有一节内容专门针对 Boost.MPI 库，虽然还未实现全部的功能，但它的确简化了使用 MPI 的过程。

GPU 编程：GPU 是把本书内容组织在一起的最初原因之一。以此类推到共享内存编程，这里从两个方面讨论了针对 GPU 软件开发的问题。一方面是 NVIDIA 的 CUDA 中的具体方法，该方法中的内存传输、数据放置和线程执行配置都需要仔细计划。这将在第 6 章进行介绍。

另一方面是高级的 Thrust 模板库算法方法，这部分在第 7 章中讨论。程序设计的类标准模板库（STL-like）方法为 Thrust 提供了把 CPU 和 GPU 平台均作为目标的能力，在本书所介绍的工具中，这是一个独特的特性。

负载均衡：第 8 章讨论在多核开发中经常被低估的一个方面。一般来说，一旦异构计算资源开始运行，就应该认真考虑负载均衡。举例来说，一个 CPU 和一个 GPU 构成这一组资源，我们在满足需求时不能只考虑一群不同机器的集合。第 8 章简要讨论了 Linda 语言，这可以被看做动态负载均衡的高级抽象。

书中主要的关注点在静态负载均衡，以及可以用于驱动负载分区和数据通信序列的数学模型。

很多场景中都应用了一种已经得到认可的方法论——可分负载理论（DLT），并且给出了解释。书中还介绍了一个简单的C++库，它实现了部分过去20年中已经发表过的DLT研究结果。

软件和硬件要求

书中的示例都是在Ubuntu Linux系统上开发和测试的。本书中使用的所有软件都是可以免费获取或者是开源的。其中包括：

- GNU C/C++编译器组件4.8.x（兼容CUDA），组件4.9.x（兼容OpenMP 4.0）
- Digia的Qt 4.x库或者Qt 5.x库
- OpenMPI 1.6
- MPE
- NVIDIA的CUDA SDK 6.5
- Thrust库（版本1.7）

如果拥有近期正确安装的Linux，并装有版本等于或高于上述软件列表中版本号的软件，那么执行本书中提供的示例代码应该不会有任何问题。虽然我没有提供Windows平台下的生成文件或者使用Visual Studio编译和执行示例代码的指令，但未安装Linux[⊖]的用户只需进行少量代码改动就可以把示例移植到Windows上。考虑到我们使用的是标准C/C++库，因此对于代码的改动（如果存在）应该只会影响头文件，也就是那些需要被包括的（include）文件。

硬件部分唯一的实际局限是需要计算能力为2.x或者更高的NVIDIA GPU。早期芯片也可以使用，但是它们的独特性，尤其是关于全局内存的访问，在书中没有给出解释。没有NVIDIA GPU的用户可以通过使用附录E中介绍的方法来成功执行CUDA程序。

示例代码

书中的程序可在Elsevier网站（http://store.elsevier.com/9780124171374）上获取压缩包[⊖]。程序存放在特定的文件夹中，以章名区分，如图1所示。

⊖ 通过虚拟机方法安装Linux是非常简单的，甚至连机器的配置都不需要修改。从Oracle免费获取VirtualBox，就可以在Windows系统的主机上以最小的资源消耗运行Linux。

⊖ 关于本书教辅资源，使用教材的教师需通过爱思唯尔的教材网站（www.textbooks.elsevier.com）注册并通过审批后才能获取。具体方法如下：在www.textbooks.elsevier.com教材网站查找到该书后，点击“instructor manual”便可申请查看该教师手册。有任何问题，请致电010-85208853。——编辑注

对于书中的代码清单，均在第 1 行给出了相对于该章目录的对应文件的位置。

单文件程序的第一行注释里包含编译和链接命令。多文件项目存放在它们自己的目录中，其中也包含一个生成文件或者一个项目（.pro）文件。如果需要输入样本值，该部分数据也在该目录下的文件中提供。

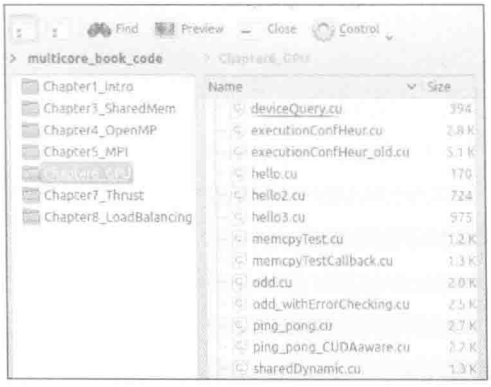


图 1　截屏显示了示例代码在特定章节的文件夹中的组织方式

教学建议

本书中所涉及的内容适合高年级本科生或者研究生课程。学生需要具备的背景知识包括掌握 C 和 C++ 编程（两种语言的使用贯穿整本书）、基本的操作系统概念，以及基本的计算机体系结构常识。

根据各自的需求，教师可以选择使用以下列出的一些教学建议。前两章是为后面的章节奠定基础，因此它们在所有的路径中都包括：

重点在并行编程（本科生）：

- 第 1 章：Flynn 分类法、当代的多核设备、性能标准。1.1 ～ 1.5 节。
- 第 2 章：设计、PCAM 方法论、分解模式、程序结构模式。2.1 ～ 2.5 节。
- 第 3 章：线程、信号、monitor。3.1 ～ 3.7 节。
- 第 4 章：OpenMP 的基本概念、工作共享结构，4.1 ～ 4.4 节。
- 第 5 章：MPI、点对点通信、集合操作、目标 / 结构通信，以及调试和性能分析。5.1 ～ 5.12 节、5.15 ～ 5.18 节和 5.20 节。
- 第 6 章：CUDA 编程模型，内存层次结构，针对 GPU 的优化。6.1 ～ 6.6 节、6.7.1 节、6.7.3 节、6.7.6 节、6.9 ～ 6.11 节和 6.12.1 节。

- 第 7 章：Thrust 基本的知识。7.1 ～ 7.4 节。
- 第 8 章：负载均衡。8.1 ～ 8.3 节。

重点在多核编程（本科生）：

- 第 1 章：Flynn 分类法、当代的多核设备、性能标准。1.1 ～ 1.5 节。
- 第 2 章：设计、PCAM 方法论、分解模式、程序结构模式。2.1 ～ 2.5 节。
- 第 3 章：线程、信号、monitor。3.1 ～ 3.10 节。
- 第 4 章：基本的 OpenMP、工作共享结构，以及正确性和性能问题。4.1 ～ 4.8 节。
- 第 5 章：MPI，点对点通信、集合操作、目标 / 结构通信，以及调试和性能分析。5.1 ～ 5.12 节、5.16 ～ 5.18 节和 5.21 节。
- 第 6 章：CUDA 编程模型、内存层次结构，以及针对 GPU 的优化。6.1 ～ 6.10 节、6.12.1 节。
- 第 7 章：Thrust 基本的知识。7.1 ～ 7.4 节。
- 第 8 章：负载均衡。8.1 ～ 8.3 节。

高级多核编程：

- 第 1 章：Flynn 分类法、当代的多核设备、性能标准。1.1 ～ 1.5 节。
- 第 2 章：设计、PCAM 方法论、分解模式、程序结构模式。2.1 ～ 2.5 节。
- 第 3 章：线程、信号、monitor、高级线程管理。3.1 ～ 3.10 节。
- 第 4 章：OpenMP 基本的知识，工作共享结构，以及正确性和性能问题。4.1 ～ 4.8 节。
- 第 5 章：MPI、点对点通信、集合操作、目标 / 结构通信，以及调试和性能分析。5.1 ～ 5.12 节、5.15 ～ 5.18 节和 5.21 ～ 5.22 节。
- 第 6 章：CUDA 编程模型、内存层次结构，以及针对 GPU 的优化。6.1 ～ 6.12 节。
- 第 7 章：Thrust 数据类型和算法。7.1 ～ 7.7 节。
- 第 8 章：负载均衡、基于 DLT 的分割。8.1 ～ 8.5 节。

目　录 Contents

⊖ 请参见华章网站 www.hzbook.com。

第 1 章 Chapter 1

概　述

本章目标：

- 了解计算机（计算机体系架构）设计的发展趋势以及该趋势如何影响软件开发。
- 学习基于 Flynn 分类的计算机分类方法。
- 学习评估多核 / 并行程序性能即加速比和效率的必备工具。
- 学习测量和报告程序性能的正确实验方法。
- 学习 Amdahl 和 Gustafson-Barsis 定律，并使用这两个定律预测并行程序性能。

1.1　多核计算机时代

在过去的 40 年中，数字计算机已经成为技术和科学发展的基石。遵循 20 世纪 70 年代摩尔（Gordon E. Moore）发现的摩尔定律，计算机的信息处理速度（性能）呈指数提高，这使得我们可以处理更加复杂的问题。

令人惊讶的是，即使在今天，摩尔定律也描述了行业的发展趋势。然而，在大众科学中有一个被忽视的问题需要澄清一下：摩尔定律描述的是晶体管数量呈指数级增长，而不是运行性能。图 1-1 描述了摩尔定律。[⊖]

这是一个非常容易犯的错误，因为晶体管数目的增加伴随着运行频率（时钟频率）的提高。但是，时钟频率的增加会导致产热的增加。为此，芯片设计者不断降低电子电路的操作电压（目前的运行电压为 1.29V）。然而，这并不足以解决这个问题。因此，时钟频率的

⊖ 这是在维基百科公用许可下发布的一个可编辑版本：http：//commons.wikimedia.org/wiki/File：Transistor_Count_and_Moore's_Law_-_2011.svg。

发展不可避免地陷入停滞。在过去 10 年中，主流时钟频率维持在 2 ～ 4GHz 之间。

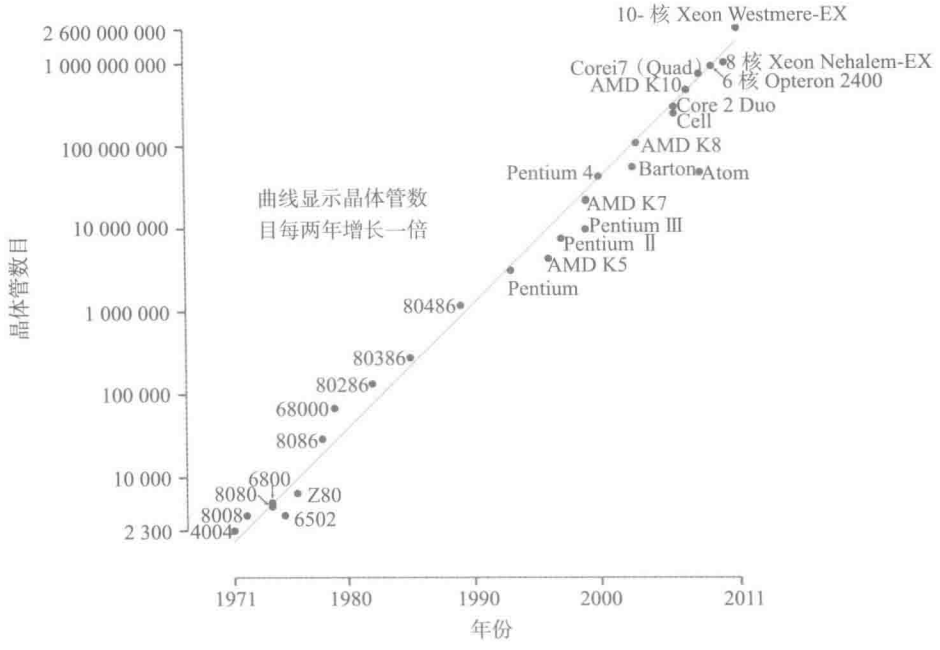


图 1-1 A　CPU 晶体管数目逐年增加的示意图

所以，获取更高计算能力的唯一途径就是在芯片内部集成更多的计算逻辑和计算核心。随着 AMD 于 2015 年推出第一款双核芯片（AMD 64 X2），更多的多核芯片也被不断推出。这其中不仅包括拥有大量计算核心的同构芯片（如 64 核 Tilera，TILE64）[⊖]，而且包括异构芯片，如 Cell BE，它采用 Power 架构，并用于 Sony Playstation 3。

这些芯片是多路（multisocket）平台（即，20 世纪 90 年代中后期出现的搭载多个 CPU 的计算机）的自然演化。然而，GPGPU（通用计算图形处理单元）的出现是一个意外。GPGPU 是指利用 GPU（Graphical Processing Unit，图形处理器）进行通用计算。虽然单个 GPU 核与同时代的 CPU 核相比性能很差，但是 GPU 采用了大规模并行架构，拥有通过高带宽、高性能 RAM 相连的成百上千个计算核心。因此，同 CPU 相比，GPU 的性能以数量级提升。

在能源日益紧张的今天，GPGPU 还有一个额外优势：它提供了卓越的 GFlop/W 的性能。换句话说，可以使用同样的能源进行更多的计算。这在服务器和云基础设施领域是非常重要的。在这些领域中，CPU 在其运行寿命中消耗的能源费用要比其购买价格高得多。

GPGPU 技术被认为是颠覆性的，在很多层面上确实是这样：它为使用现代单核甚至多核 CPU 技术仍然无法解决的问题提供了解决方案。但是，GPGPU 需要新的软件设计、开

⊖ http://www.tilera.com/products/processors/TILE64.

发工具和技术。据预测，在不久的将来，需要数百万个线程来开发下一代高性能计算硬件的性能。

然而，所有这些多核芯片带来的性能提升都不是免费的：需要对按部就班执行的传统算法进行重新设计。

1.2 并行计算机的分类

使用多种资源获取更高性能并不是最新的技术，这个技术最早开始于20世纪60年代。因此，定义一种描述并行计算机架构特征的方法是非常重要的。1966年，Michael Flynn引入了一种计算机体系结构分类方法：根据能够并发处理的数据量和同时执行的不同指令数目进行分类。根据这两个条件，计算机体系结构可以分为四类：

单指令单数据（Single Instruction Single Data，SISD）：只能同时执行一条指令并处理一个数据的串行计算机。出人意料的是，绝大多数现代CPU都不属于这个类别。甚至现代微控制器都是多核配置，每个核都可以认为是一个SISD机器。

单指令多数据（Single Instruction Multiple Data，SIMD）：一条指令可以处理多个数据的计算机。最早出现的这类机器是向量处理器。GPU的流多处理器[⊖]（Streaming Multiprocessor，SM；针对NVIDIA）和SIMD单元（针对AMD）遵循这样的设计。

多指令单数据（Multiple Instructions Single Data，MISD）：这好像是一个“怪胎”，多条指令如何处理相同数据呢？正常情况下，这是没有意义的。然而，当系统（军事或航天应用中）需要容错时，数据被多个机器处理，并根据多数原则做出最终决定。

多指令多数据（Multiple Instructions Multiple Data，MIMD）：最通用的类别。多核计算机（包括GPU）都遵循这种设计。GPU由多个SM/SIMD单元组成，每个SM/SIMD单元都能够运行自己的程序。所以，即使单个SM属于SIMD类别，多个SM集成起来的工作方式也是MIMD。

这个分类方法经过细化，又添加了几个子类（特别是MIMD），如图1-2所示。MIMD分成两个子类。

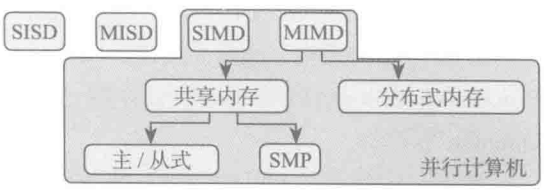


图1-2 计算机系统的Flynn扩展分类方法

- 共享内存MIMD：具有共享存储空间的计算机体系架构。一方面，共享内存在最小化额外开销的前提下，简化了所有CPU间的操作；另一方面，共享内存限制了系统的可扩展性。解决这个问题的一个方法就是划分内存，为每个CPU分配一块独立的内存。这样，CPU不仅能以较高的性能访问本地内存，还能以较低的性能访问属于其他CPU的非本地内存。值得注意的是，内存划分不会影响寻址机制。这

⊖ 更多信息见6.3节。