

HPC领域资深专家推荐，中国最大OpenACC技术社区创建者撰写，不可多得的OpenACC技术专著

全面讲解OpenACC编程规范、语法的行为机理与设计动机，160个完整示例覆盖众多并行编程场景



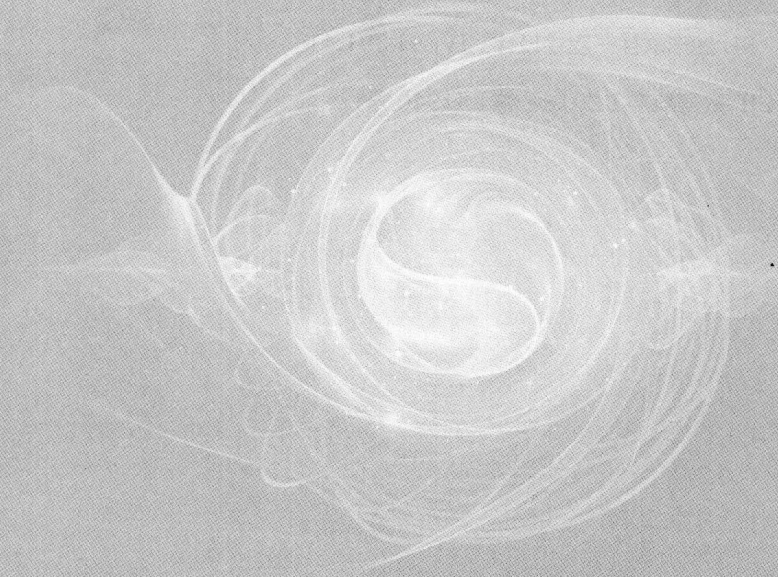
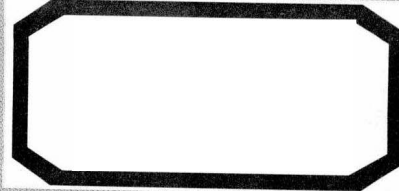
OpenACC Parallel Programming

OpenACC 并行编程实战

何沧平 著



机械工业出版社
China Machine Press



OpenACC Parallel Programming

OpenACC 并行编程实战

何沧平 著



机械工业出版社
China Machine Press

图书在版编目 (CIP) 数据

OpenACC 并行编程实战 / 何沧平著. —北京: 机械工业出版社, 2017.1
(高性能计算技术丛书)

ISBN 978-7-111-54965-9

I. O… II. 何… III. 图形软件—程序设计 IV. TP391.41

中国版本图书馆 CIP 数据核字 (2016) 第 234940 号

OpenACC 并行编程实战



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: 高婧雅

责任校对: 殷虹

印刷: 北京诚信伟业印刷有限公司

版次: 2017 年 1 月第 1 版第 1 次印刷

开本: 186mm×240mm 1/16

印张: 17.25

书号: ISBN 978-7-111-54965-9

定价: 69.00 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

版权所有·侵权必究

封底无防伪标均为盗版

本书法律顾问: 北京大成律师事务所 韩光 / 邹晓东

随着计算机科学技术的飞速发展，作为计算机的核心——处理器的体系结构也经历了从单核、多核到众核的革命性跨越。如今，“异构+众核”已成为超级计算机主流的体系结构，并将引领未来 E 级计算的进一步发展。

然而，这种较为“前卫”的结构设计，也给编程者带来了很大的挑战：他们不得不面对更为复杂的底层结构和更多的存储层次，以往的算法设计和程序代码也不得不随之调整。在这种情况下，诸如 CUDA、OpenCL、OpenACC 等新的异构并行编程语言应运而生。其中，OpenACC 是一种极具发展前景的编程模型，其具有使用方便、代码改动小、平台适用范围广的特性，将为新时代的编程者们带来极大的帮助。

OpenACC 虽然有着易学易用的特点，但是要想全面掌握其丰富的语法特性和使用技巧，以编写高效的程序，还是需要一本有权威性、实用性的技术书籍来指导。本书作者何沧平博士是华为高级系统设计工程师，也是 OpenACC QQ 群的群主，具有多年的程序开发经验，一直致力于 OpenACC 的发展与推广，在并行计算应用领域有着很深的造诣和独到的见解。

对于想要掌握 OpenACC 使用技巧和编程精髓的读者来说，本书是一本非常具有参考价值的学习教程。与传统的技术书籍相比，本书更加注重内容的可读性和易用性，逻辑清晰，内容全面准确，且更加注重编程实践，有大量 C/C++/Fortran 的完整代码实例，便于读者学习和实践。作为第一本中文的 OpenACC 技术书籍，可谓为国内的编程学习者带来了福音。

特别值得一提的是，本书首次引入了 OpenACC 在“神威·太湖之光”超级计算机上应用情况的章节。神威·太湖之光是 2016 年全球 TOP500 排名第一的超级计算机，配备了完全由国人自主研发的异构众核处理器。其超强的计算速度成为了高性能计算应用的强力助推。结合应用的特点和处理器独特的结构设计，系统对 OpenACC 进行了扩展。目前，该系统完成了气候气象、航空航天、船舶工程、药物设计等十多个领域的大型应用课题，其中三个高性能计算应用入围“戈登贝尔奖”。这是我国近 30 年来首次入围该奖项。在这些应用的开发过程中，OpenACC 起到了关键作用。

前 言 *Foreword*

2010 年以来，中国超级计算机建设突飞猛进，欣欣向荣。一个原因是国力强盛，大力投资高新科技；另一个原因是整体科技水平提高，需求旺盛。天气预报、石油物探、工程仿真、基因测序等传统应用对计算资源的需求持续增长，以深度学习为代表的人工智能大爆发，资金雄厚的互联网公司对计算能力极度渴求。超级计算机的建设、应用主战场正在从教育科研单位转向科技企业。

为什么要写这本书

面对浩如烟海的数据，CPU 已经力不从心，因此世界领先的超级计算机都装备大量的加速器或者众核处理器。

目前主流加速器产品是 NVIDIA GPU、AMD GPU 和 Intel 至强 Phi 协处理器。三种加速器使用的编程语言分别为 CUDA C/CUDA Fortran、OpenCL 和 MIC 导语。加速器计算有 4 个困难。

一是 CUDA/OpenCL 等低级语言编程难度大，且需要深入了解加速器的硬件结构。而大部分的用户不是专业编程人员，学习一门新的编程技术将耗费大量时间。

二是加速器的计算模型与 CPU 差别很大，移植旧程序需要几乎完全重写。大量的旧程序在性能优化上已经千锤百炼，稳定性上也久经考验，完全重写是不可完成的任务。

三是低级编程语言开发的程序与硬件结构密切相关，硬件升级时必须升级软件，否则将损失性能。而硬件每隔两三年就升级一次，频繁的软件升级将给用户带来巨大负担。

四是投资方向难以选择。三种加速器均有自己独特的编程语言，且互不兼容。用户在投资建设硬件平台、选择软件开发语言时就会陷入困境，不知三种设备中哪个会在竞争中胜出。如果所选加速器将来落败，将会带来巨大损失；而犹豫不决又将错过技术变革的历史机遇。

OpenACC 应运而生，可以克服这 4 个困难。OpenACC 的编程机制是，程序员只在原程序中添加少量编译标识，编译器根据作者的意图自动产生低级语言代码。无须学习新的

编程语言和加速器硬件知识，便能迅速掌握。只添加少量编译标识，不破坏原代码，开发速度快，既可并行执行又可恢复串行执行。在硬件更新时，重新编译一次代码即可，不必手工修改代码。OpenACC 标准制定时就考虑了目前及将来的多种加速器产品，同一份代码可以在多种加速器设备上编译、运行，无成本切换硬件平台。掌握 OpenACC 后，编写程序省时、省力、省心。

本书特色

笔者学习超算技术时有过苦泪：教材一上来就讲技术细节，只能机械地学习，不清楚这些算法、语法要解决什么问题，花费巨大精力后却发现解决不了自己的难题；新技术的资料往往是英文的，而且零碎，还不一定准确，收集、学习成本很高，求助无门；科学计算领域的祖师语言 Fortran 现在有点小众，新技术资料更少。为节省读者时间，本书特别重视易读性、易用性，具体有如下特点。

- ❑ 第一本中文 OpenACC 技术书籍，方便国内读者系统阅读，快速掌握。
- ❑ 全面、准确：本书覆盖 OpenACC 的所有特性，重要特性结合例子详细讲解，不常用的特性列出标准定义，方便查找。对特性、功能的描述均来自 OpenACC 规范，并用实例验证，确保准确。
- ❑ 大量的例子：书中的示例代码超过 160 个，不是代码片断，而是完整代码，拿来即用。全部由笔者编写、测试，运行无报错。
- ❑ 兼顾 C/C++ 和 Fortran：以 C/C++ 为主来讲解，但所有的例子都有 Fortran 版的代码，Fortran 专用特性会详细讲解。物理、气象等领域积累的优秀 Fortran 代码可以借此移植到新硬件上，重焕生机。
- ❑ 阅读轻松：按照并行程程序的编写、调优顺序，先交代各阶段面临的问题，然后自然引入 OpenACC 提供的解决办法，最后实测验证效果。一切顺理成章，读者不会有云里雾里的感觉。

读者对象

- ❑ 科学家：迅速改造旧程序，快速编写新的原型程序验证算法、理论，将更多的时间投入到高价值的科研创新中去。
- ❑ 企业程序员：一套程序适配多种运行平台，维护简单；性能敏感的部分用 CUDA 等低层语言编写，性能不敏感部分用 OpenACC 编写，相互配合，兼得程序性能和开发效率。
- ❑ 本科生、研究生：花最少的时间掌握一门计算工具，省出时间学习更多的知识。

如何阅读本书

第1章介绍超级计算技术的发展趋势和并行编程概况，可以从中了解 OpenACC 的作用。没有 CUDA C 基础的读者能够掌握基本概念，便于深入理解 OpenACC 的并行化技术。第2章介绍 OpenACC 语言的设计思路。第3~4章是本书的核心，将计算部分并行化，并将数据传递时间减到最少。至此，读者已经能够编写性能良好的 OpenACC 程序。第5~7章介绍高级并行技术，以进行极致性能优化，以及与 CUDA C/CUDA Fortran 和各类库的混合编程。第8章给出 OpenACC 规定的所有运行时例程，不用细读，用到时再参考。第9章指导部署开发环境，以便快速上手。

勘误和支持

本人水平有限，错误与疏漏在所难免，恳请批评指正。联系笔者请发送电子邮件至 hpcfan@foxmail.com。更多技术资源请访问 www.gpujisuan.com。技术交流 QQ 群 284876008（将满）、564520462，欢迎加入。

致谢

感谢国家并行计算机工程技术研究中心漆锋滨老师撰写第10章，并为本书作序。感谢 PGI 美女工程师王珍、帅气工程师仰璘歆、Daniel Tian、资深专家 Mathew Colgrove 的技术支持。感谢机械工业出版社的策划编辑高婧雅尽心协调与支持。

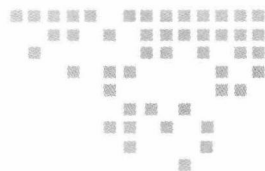
何沧平

Contents 目 录

序	
前言	
第 1 章 并行编程概览	1
1.1 加速器产品	1
1.1.1 英伟达 GPU	3
1.1.2 英特尔至强融核处理器	9
1.2 并行编程语言	12
1.3 CUDA C	14
1.3.1 线程组织方式	16
1.3.2 运行过程	18
1.3.3 内存层级	20
1.3.4 性能优化技术	21
第 2 章 OpenACC 概览	22
2.1 OpenACC 规范的内容	23
2.1.1 抽象加速器模型	25
2.1.2 存储模型	25
2.1.3 计算执行模型	26
2.2 OpenACC 2.5 规范	29
第 3 章 OpenACC 计算构件	36
3.1 条件编译	37
3.2 导语格式	38
3.3 计算构件 kernels	40
3.3.1 构件内有 1 个循环	41
3.3.2 构件内 2 个循环	44
3.3.3 构件内二重嵌套循环	45
3.3.4 kernels 构件内三重嵌套循环	48
3.4 loop 构件	52
3.4.1 independent 子语	53
3.4.2 reduction 归约子语	57
3.4.3 不常用的子语	64
3.5 计算构件 parallel	66
3.5.1 gang 单独模式	68
3.5.2 gang 分裂模式	70
3.5.3 二重循环	73
3.5.4 三重循环	75
3.6 组合导语	77
3.7 案例研究: Jacobi 迭代	78
3.7.1 CPU 上并行化	84
3.7.2 GPU 上并行化	88
3.8 原子操作: atomic 导语	91
第 4 章 数据管理	97
4.1 数据属性、数据区域和数据 生存期	99
4.2 计算构件的伴随数据区域	100

4.2.1 引用计数	101	5.4 routine (名字)	154
4.2.2 present 子语	102	5.5 bind 子语	155
4.2.3 copy 子语	104	5.6 用子语指定并行级别	155
4.2.4 copyin 子语	105	5.6.1 vector 级别并行	156
4.2.5 copyout 子语	107	5.6.2 worker、worker 级别并行	159
4.2.6 create 子语	108	5.7 计算圆周率 π	160
4.2.7 数据子语内的子数组	111		
4.2.8 private 私有子语	112	第 6 章 高级特性	164
4.2.9 承上私有 firstprivate 子语	115	6.1 异步操作	164
4.2.10 带有预置数据属性的变量	116	6.1.1 async 子语	165
4.2.11 default 默认子语	117	6.1.2 wait 子语	165
4.2.12 案例研究: Jacobi 迭代优化		6.1.3 wait 导语	166
数据传输	117	6.2 设备计算与主机计算重叠	166
4.3 data 构件	119	6.3 设备上同时执行多个队列	169
4.3.1 数据管理功能	119	6.4 重叠计算与数据传输	172
4.3.2 deviceptr 子语	121	6.4.1 步骤 0: 串行代码	174
4.3.3 案例研究: data 构件迭代优化		6.4.2 步骤 1: 计算并行化	177
Jacobi 数据传输	122	6.4.3 步骤 2: 分块计算	178
4.4 enter data 导语和 exit data 导语	128	6.4.4 步骤 3: 数据分块传输	179
4.4.1 C++ 类的数据生存期	129	6.4.5 步骤 4: 重叠计算与传输	181
4.4.2 传递设备数据指针	133	6.5 双向传输	183
4.5 update 导语	135	6.6 多个设备同时运算	185
4.6 declare 导语	138	6.6.1 环境变量	186
4.6.1 device_resident 子语	139	6.6.2 运行过程中选择设备	186
4.6.2 create 子语	140	6.6.3 OpenMP 调动多个设备	195
4.6.3 link 子语	140		
4.6.4 用法举例	141	第 7 章 与 GPU 生态环境互操作	202
4.7 特定设备的子语	146	7.1 OpenACC 调用 CUDA C	203
第 5 章 计算区域内的过程调用	148	7.2 OpenACC 调用 CUDA Fortran	205
5.1 routine 导语	150	7.3 CUDA C 调用 OpenACC	207
5.2 seq 子语 (C 版)	151	7.4 捆绑主机地址与设备地址	208
5.3 seq 子语 (Fortran 版)	152	7.5 CUDA Fortran 调用 OpenACC	210

7.6	OpenACC (C) 调用 cuBLAS	211	8.2.21	acc_copyout	227
7.7	OpenACC (Fortran) 调用 cuBLAS	212	8.2.22	acc_delete	228
第 8 章 运行时函数213					
8.1	运行时库的定义	213	8.2.23	acc_update_device	229
8.2	运行时库例程	215	8.2.24	acc_update_self	230
8.2.1	acc_get_num_devices	215	8.2.25	acc_map_data	231
8.2.2	acc_set_device_type	216	8.2.26	acc_unmap_data	231
8.2.3	acc_get_device_type	217	8.2.27	acc_deviceptr	231
8.2.4	acc_set_device_num	217	8.2.28	acc_hostptr	232
8.2.5	acc_get_device_num	218	8.2.29	acc_is_present	232
8.2.6	acc_init	218	8.2.30	acc_memcpy_to_device	233
8.2.7	acc_shutdown	219	8.2.31	acc_memcpy_from_device	233
8.2.8	acc_async_test	219	8.2.32	acc_memcpy_device	234
8.2.9	acc_async_test_all	220	第 9 章 开发环境搭建235		
8.2.10	acc_wait	220	9.1	Windows 7	236
8.2.11	acc_wait_async	221	9.2	Linux (rhel)	244
8.2.12	acc_wait_all	221	9.3	编译工具、特性支持度	247
8.2.13	acc_wait_all_async	222	第 10 章 在神威·太湖之光上 使用 OpenACC253		
8.2.14	acc_get_default_async	222	10.1	SW26010 众核处理器	253
8.2.15	acc_set_default_async	223	10.2	存储模型	254
8.2.16	acc_on_device	223	10.3	执行模型	256
8.2.17	acc_malloc	224	10.4	数据管理	256
8.2.18	acc_free	224	附录 著名超级计算机259		
8.2.19	acc_copyin	225	后记 码农的悲喜264		
8.2.20	acc_create	226			



并行编程概览

对绝大多数人而言，编程语言只是一个工具，讲究简单高效。科学家的主要精力应该用在科研创新活动上，编程工作仅仅是用来验证创新的理论，编程水平再高也不可能获得诺贝尔奖。对学生和企业程序员而言，技术无穷尽，永远学不完，不用即忘，应该认清技术发展方向，学习有前途的技术，不浪费青春年华。

OpenACC 语言专为超级计算机设计，因此读者需要了解超级计算机的技术演进方向，特别是主流加速器的体系架构、编程模型，看清 OpenACC 的应用场景，有的放矢。普通读者虽然不会用到大型机群，但小型机群甚至单台服务器、普通显卡的计算模式都是相同的。

最近几年的著名超级计算机（见附录 A）均采用加速器作为主要计算部件，可预见未来几年的上层应用仍将围绕加速器展开。

1.1 加速器产品

超级计算机的加速器历史上有多种，本节只介绍当前流行的两种：英伟达 GPU 和英特尔融核处理器。加速器的物理形态是 PCIe 板卡，样子大致如图 1.1 所示，图 1.2 是拆掉外壳后的样子，正中央的是 GPU 芯片，芯片周围的小黑块是显存颗粒，金黄色的边缘处是与 PCIe 连接的金手指，通过 PCIe 插槽与 CPU 相连。图 1.3 中的机架式服务器左下部装有 4 块 GPU 卡，图 1.4 是服务器的主板俯视图，箭头处就是 4 个 PCIe 插槽。

从逻辑关系上来看，目前市场在售的英特尔服务器 CPU 中已经集成了内存控制器和 PCIe 通道。2 颗 CPU 通过 1 个或 2 个 QPI 接口连接，CPU 通过内置的内存控制器连接 DDR3

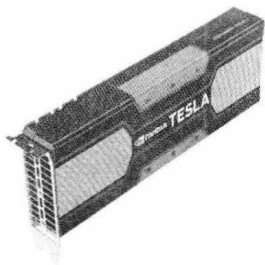
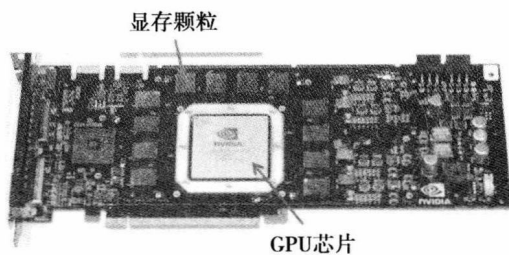
图 1.1 英伟达 Tesla GPU[⊖]

图 1.2 英伟达 GPU 内部构造

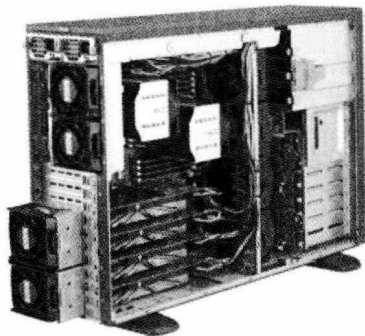
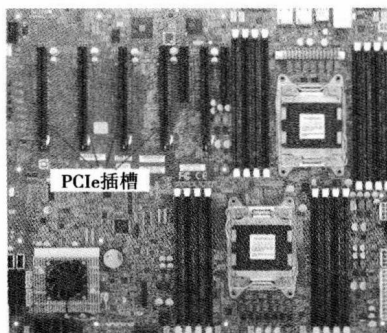
图 1.3 GPU 服务器[⊖]

图 1.4 GPU 服务器的主板

或 DDR4 内存条，GPU 加速器通过 PCIe 总线与 CPU 相连（图 1.5）。2016 年 6 月主流的英特尔至强 E5-2600 v3 和 E5-2600 v4 CPU 每颗拥有 40 个 PCIe Lan，而每块 GPU 的接口需要 16 个 PCIe Lan，因此每颗 CPU 上最多全速挂载 2 块 GPU。有些服务器的 PCIe $\times 16$ 插槽上只安排 $\times 8$ ，甚至 $\times 4$ 的信息速率，可以挂载更多的 GPU。

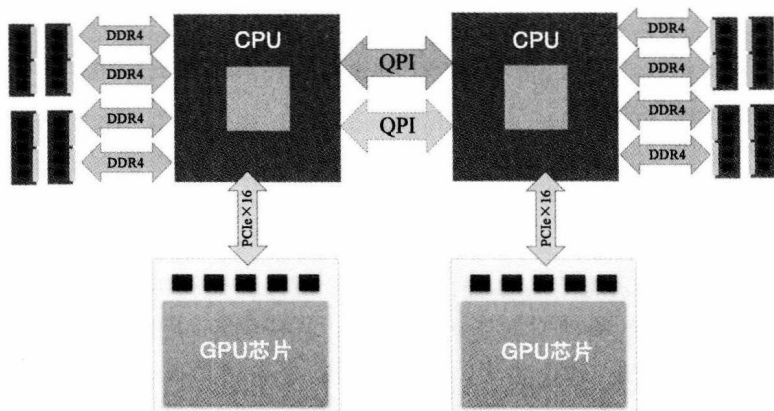


图 1.5 加速器在服务器上的逻辑位置

[⊖] 图片来自 www.nvidia.com。

[⊖] 图片来自 www.supermicro.org.cn。

英特尔融核处理器在服务器上的位置与 GPU 一样，不再赘述。

1.1.1 英伟达 GPU

通用计算 GPU 是英伟达公司发明的，每隔 2~3 年就会升级硬件架构。通用计算 GPU 的第一代架构代号是 G80，每 8 个核心封装在一起称为一个流式多处理器（Streaming Multiprocessor，SM），2 个流多处理器共用一块 L1 缓存，所有的核心共用 L2 缓存，任意核心都能访问芯片外部的 GPU 显存（图 1.6 中的 FB）。从图 1.6 中可以看出，G80 架构 GPU 产品最多可以有 128 个核心。

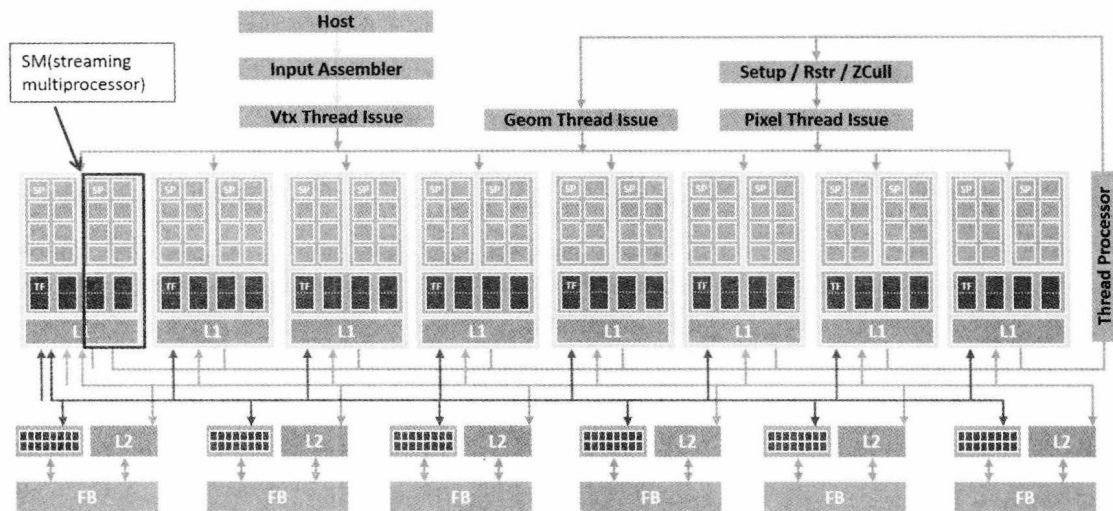


图 1.6 通用计算 GPU 的 G80 架构

接下来的架构代号是 GT200，然后是 Fermi。Fermi 架构（图 1.7）最多包含 14 个流式多处理器，每个流式多处理器包含 32 个核心。6 个 DRAM 接口，1 个 PCIe 接口（图 1.16 中的 Host Interface）。

流式多处理器内部组件也相当多（图 1.8）：1 个指令缓存（Instruction Cache），2 组 Warp 调度器（Warp Scheduler）、分发单元（Dispatch Unit），一堆寄存器；最重要的是 32 个核心，每个核心拥有一个单精度浮点单元和一个整数单元；16 个 Load/Store 单元；4 个特殊功能单元（Special Function Unit），负责计算双精度浮点数、三角函数、超越函数、倒数、平方根等。共享内存 + L1 缓存共 64KB，可以灵活配置。

接下来是 Kepler 架构（图 1.9），这一代架构最大可以包含 15 个流式多处理器（Streamong Multiprocessor extreme，SMX）和 6 个 64 位内存控制器。不同的产品型号将使用不同的配置，可以包含 13 或 14 个 SMX，多种参数都有升级和更改。Kepler 架构的主要设计目标是提高用电效率，台积电的 28nm 制造工艺在降低功耗方面起着重要的作用。Kepler 架构还提高了双精度计算能力。

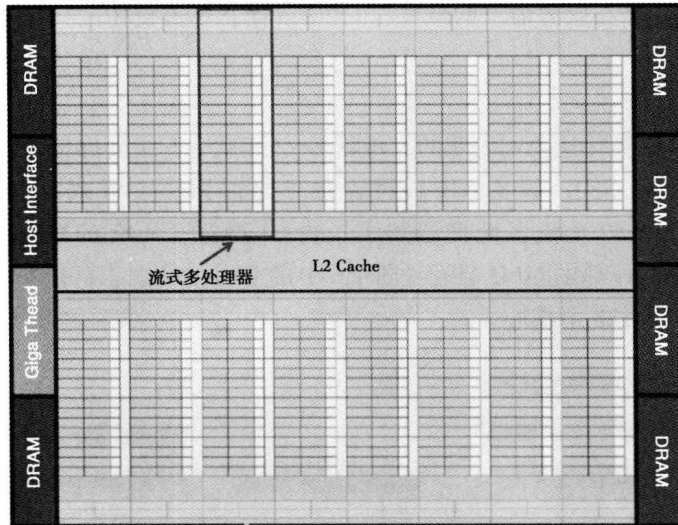


图 1.7 Fermi 架构图

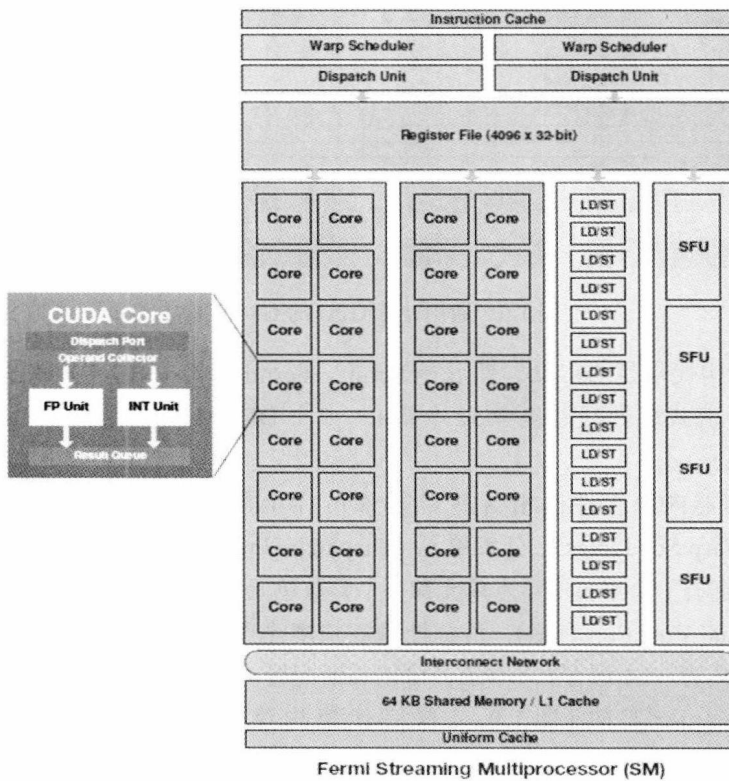
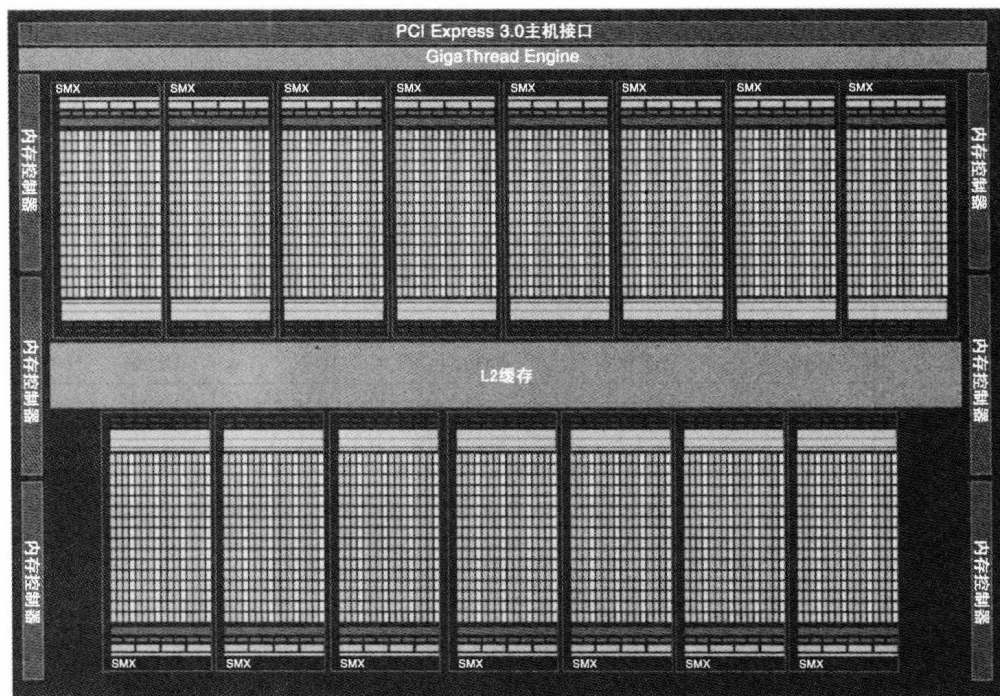


图 1.8 Fermi 架构的流式多处理器[⊖]

[⊖] 图片来自 www.nvidia.com。

图 1.9 Kepler 架构[⊖]

流式多处理器 SMX (图 1.10) 包含 192 个单精度核心、64 个双精度单元、32 个特殊功能单元 (SFU) 和 32 个加载 / 存储单元 (LD/ST)、3 个 Warp 调度器、6 个分发器、一大堆寄存器。每个核心由 1 个浮点计算单元和 1 个整数算术逻辑单元组成, 支持融加 (FMA) 运算。每个 SMX 拥有 64KB 的片上存储器, 可配置为 48KB 的共享存储器和 16KB 的 L1 缓存, 或配置为 16KB 的共享内存和 48KB 的 L1 缓存。

这里简要介绍市面的主力 GPU 产品型号, 见表 1.1。

表 1.1 2014~2016 年市场主力 GPU 产品规格

型 号 特 性	Tesla K80	Tesla K40	Tesla K20X	Tesla K20	Tesla K10
GPU 数量类型	2×Kepler GK210	1 个 GK110	1 个 Kepler GK110		2 个 Kepler GK104
双精浮点峰值 (基础 / 提速)	1.87 Tflops 2.91 Tflops	1.43 Tflops 1.66 Tflops	1.31 Tflops	1.17 Tflops	190 (2×95) Gflops
单精浮点峰值 (基础 / 提速)	5.6 Tflops 8.74 Tflops	4.29 Tflops 5 Tflops	3.95 Tflops	3.52 Tflops	4576 (2×2288) Gflops
存储器带宽 (ECC 关闭)	480GB/s	288GB/s	250GB/s	208GB/s	320 (2×160) GB/s

⊖ 图片来自《英伟达™ (NVIDIA®) 下一代 CUDA™ 计算架构: Kepler™ GK110》。

(续)

特 性 \ 型 号	Tesla K80	Tesla K40	Tesla K20X	Tesla K20	Tesla K10
存储器容量 (GDDR5)	24GB (2×12GB) /s	12GB/s	6GB/s	5GB/s	8GB (2×4GB) /s
CUDA 核心数	4992	2880	2688	2496	3072 (2×1536)
核心时钟	562MHz	745MHz	732MHz	706MHz	745MHz
超频时钟	875MHz	810/875MHz	不支持	不支持	不支持
存储器时钟	5GHz DDR5	6GHz GDDR5	5.2GHz DDR5	5.2GHz DDR5	5GHz DDR5
最大功耗	300W	235W	235W	225W	225W
制程	28nm	28nm	28nm	28nm	28nm

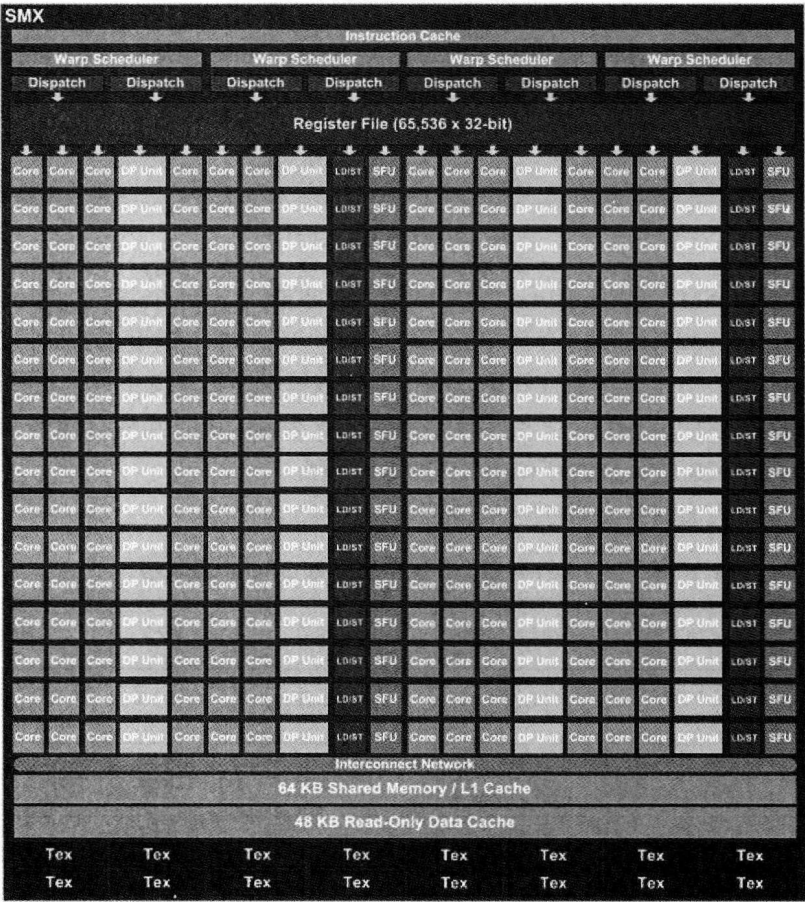


图 1.10 kepler 架构中的流式多处理 SMX[⊖]

⊖ 图片来自《英伟达™ (NVIDIA®) 下一代 CUDA™ 计算架构: Kepler™ GK110》。

2016年4月,英伟达在GPU技术会议(GPU Technology Conference)上发布了新一代Pascal架构和旗舰产品Tesla P100,引入一些新特性。

- 极致性能:为高性能计算、深度学习等计算领域设计。双精度浮点峰值5.3 Tflops,单精度浮点峰值10.6 Tflops,专为深度学习设计的半精度浮点峰值达到惊人的21.2 Tflops。按双精度峰值对比,Tesla P100是同期主力高端CPU英特尔至强E5-2680 v4的10倍。按照深度学习应用性能对比,Tesla P100是E5-2680 v4的20倍。
- NVLink:为应用扩展性全新设计的高速、高带宽互连协议。
- HBM2:快速、大容量、高效片上堆叠式内存。
- 统一内存:用统一的代码来管理主机CPU内存和GPU内存,方便开发代码。

Pascal最强劲的是GP100硬件架构(图1.11),包含60个Pascal流式多处理器和8个512位内存控制器(共4096位)。每个流式多处理器拥有64个CUDA核心和4个纹理单元。GP100共有3840个单精度核和240个纹理单元。每个内存控制器附带512KB的L2缓存,每个HBM2堆叠内存由一对内存控制器管理。L2缓存共计4096KB。Tesla P100共配置有56个流式多处理器。

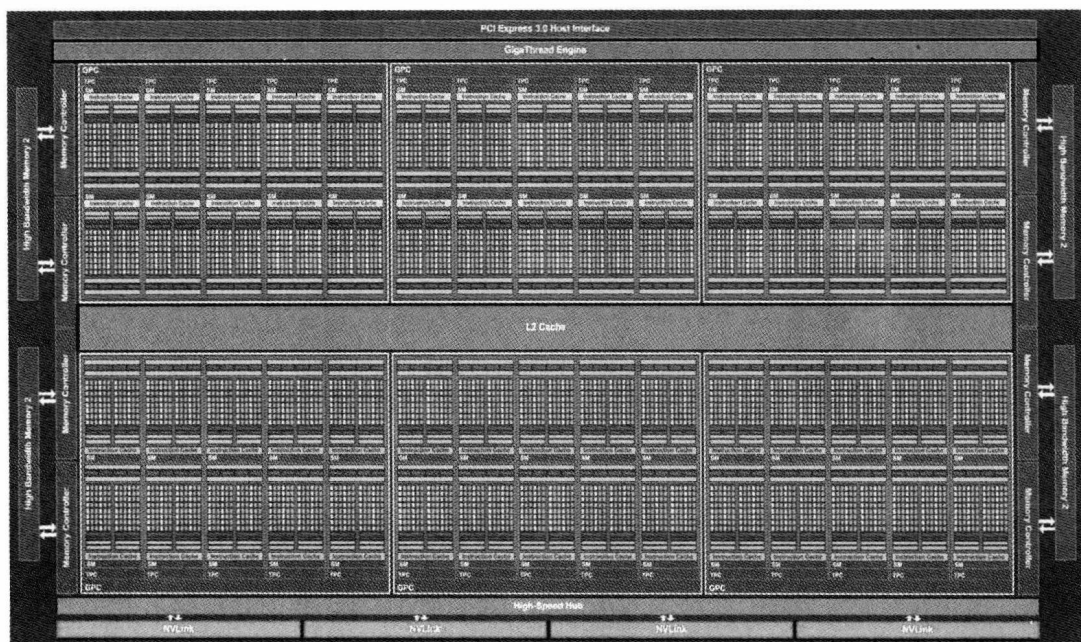


图 1.11 GP100 架构全景图^①

GP100的第6代流式多处理器架构提升了CUDA核心的利用率和能源效率,可以运行在更高的频率上。每个流多处理器包含64个单精度(FP32)CUDA核心,分成两部分,每

^① 图片来自《Pascal architecture whitepaper》。