主编 张 舒 褚艳利 副主编 赵开勇 张钰勃

NVIDIA CUDA

GPU

高性能运算之CUDA

- 精选典型实用例程，详解CUDA使用细节
- 重视理论结合实际，介绍并行程序设计方法
- 深入分析硬件架构，揭示模型与底层映射关系
- 精心总结优化经验，解析高性能编程技巧
- 技术支持：论坛（http://bbs.hpctech.com）

中国水利水电出版社

GPU 高性能运算之 CUDA

主　编　张　舒　褚艳利

副主编　赵开勇　张钰勃

中国水利水电出版社
www.waterpub.com.cn

内 容 提 要

本书是全国第一本全面介绍 CUDA 软硬件体系架构的书籍。全面介绍使用 CUDA 进行通用计算所需要的语法、硬件架构、程序优化技巧等知识，是进行 GPU 通用计算程序开发的入门教材和参考书。

本书共分 5 章。第 1 章介绍 GPU 通用计算的发展历程，介绍并行计算的历史、现状以及面临的问题；第 2 章介绍 CUDA 的使用方法，帮助读者理解 CUDA 的编程模型、存储器模型和执行模型，掌握 CUDA 程序的编写方法；第 3 章探讨 CUDA 硬件架构，深入分析 Tesla GPU 架构与 CUDA 通用计算的相互作用；第 4 章总结 CUDA 的高级优化方法，对任务划分、存储器访问、指令流效率等课题进行探讨；第 5 章以丰富的实例展示如何使用 CUDA 的强大性能解决实际问题。

本书可作为 CUDA 的学习入门和编程参考书，主要面向从事高性能计算的程序员与工程师，使用 GPU 加速专业领域计算的科研人员，以及对 GPU 通用计算感兴趣的程序员。开设相关课程的高等院校与科研机构也可选用本书作为教材。

图书在版编目（CIP）数据

GPU高性能运算之CUDA / 张舒，褚艳利主编. -- 北京 : 中国水利水电出版社，2009.10
ISBN 978-7-5084-6543-2

Ⅰ. ①G… Ⅱ. ①张… ②褚… 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(2009)第177649号

策划编辑：周春元　　责任编辑：宋俊娥

书　　名	GPU 高性能运算之 CUDA
作　　者	主　编　张　舒　褚艳利 副主编　赵开勇　张钰勃
出版发行	中国水利水电出版社 （北京市海淀区玉渊潭南路 1 号 D 座　100038） 网址：www.waterpub.com.cn E-mail：mchannel@263.net（万水） sales@waterpub.com.cn 电话：（010）68367658（营销中心）、82562819（万水）
经　　售	全国各地新华书店和相关出版物销售网点
排　　版	北京万水电子信息有限公司
印　　刷	北京市天竺颖华印刷厂
规　　格	184mm×260mm　16 开本　17.75 印张　438 千字
版　　次	2009 年 10 月第 1 版　2009 年 12 月第 2 次印刷
印　　数	4001—7000 册
定　　价	38.00 元

凡购买我社图书，如有缺页、倒页、脱页的，本社营销中心负责调换

版权所有 · 侵权必究

前　言

CUDA 是 NVIDIA 公司于 2007 年推出的 GPU 通用计算产品。CUDA 能够有效利用 GPU 强劲的处理能力和巨大的存储器带宽进行图形渲染以外的计算，广泛应用于图像处理、视频播放、信号处理、人工智能、模式识别、金融分析、数值计算、石油勘探、天文计算、流体力学计算、生物计算、分子动力学计算、数据库管理、编码加密等领域，并在这些应用中对 CPU 获得了一到两个数量级的加速，取得了令人瞩目的成绩。CUDA 在产业界和学术界引发了巨大反响，目前已有 Adobe、Autodesk、HP、联想、浪潮等国内外著名企业推出了基于 CUDA 的产品；哈佛、伯克利、清华等大批国内外高校也开设了相关课程，将 CUDA 作为一种典型的并行系统来帮助学生学习高性能计算的原理与应用。

多核的 CPU 和众核的 GPU 已经成为目前大多数计算机中最重要的两种处理器。传统上，GPU 只用于处理 3D 图像渲染任务，而其他大多数的任务都交给了 CPU。作为一种通用处理器，CPU 的设计必须兼顾各种任务的需要。因此，CPU 中大多数的晶体管都被用于制造庞大的缓存和复杂的控制逻辑，而运算单元占用的面积则并不多。近几年来，CPU 生产厂商已经认识到依靠增加单个核心的缓存尺寸和控制逻辑的复杂程度对提高处理器性能已经没有太大帮助，因此纷纷在一块芯片中集成更多的 CPU 核心。不过，这样做只是能够用更多的晶体管制造一个处理器，并没有提高晶体管的利用率。与此同时，由于图形渲染的并行特性，GPU 与生俱来就是一种拥有大量运算单元的并行处理器。近几年来的发展经验表明，一块高端 GPU 的单精度浮点处理性能可以达到一块同时期高端桌面 CPU 的 10 倍，其显存带宽也达同时期桌面平台的 5 倍左右。并且，由 GPU 提供相同的计算能力，所需要的成本和功耗都要小于基于 CPU 的系统。

由于传统 GPU 硬件架构的限制，很难有效利用其资源进行通用计算。NVIDIA 推出的 CUDA 产品则完全扭转了这一局面。与传统 GPU 通用计算开发方式相比，CUDA 编程更简单，功能更强大，应用领域更广泛，支持 CUDA 的硬件性能也更强。CUDA 完全改变了 PC 中的一些游戏规则，使用 GPU 计算无需投入额外的成本，却可以在一些应用中获得一个数量级的加速。在科学计算中，一些过去必须由集群处理的任务，现在也可以由每个研究人员的桌面 PC 完成了，这使得科研人员能够更加自由地将时间投入到自己的研究中。在超级计算机与集群中，过去往往拆除“多余的”显卡以节省功耗，现在的设计却开始采用大量 GPU 来获得更加绿色的计算能力。CUDA 的强大性能引发了一场通用计算革命，这场革命将极大地改变计算机的面貌。

我们很幸运地参与了这场 GPU 通用计算革命。在 2007～2009 年间，我们看到国内有越来越多的开发者尝试将 CUDA 应用到自己感兴趣的领域中，在学术和产业领域都取得了令人欣喜的成绩。与此同时，我们也注意到很多程序员在由 CPU 程序开发转移到 CUDA 程序开发的过程中遇到了一些问题。特别是由于 CUDA 程序的性能往往和并行算法以及硬件架构有很大关系，初次接触 GPU 的程序员往往需要较长的时间才能完全掌握 CUDA 程序的编写与优化。因此，我们结合自己的学习经验，结合例子对 CUDA 语法进行讲解，将 CUDA 编程需要的各

种知识进行归纳而编写了本书。本书既可以作为科研人员与程序开发人者学习 CUDA 的入门指导和编程优化的参考手册，也可以作为高校相关课程的教材。

本书的编写过程耗时一年，在这一年中我们也和国内的其他 CUDA 程序员共同成长，一起经历了 CUDA 从稚嫩走向成熟，渐渐发展壮大的过程。国内最开始只有一些高校中的教师学生在进行一些学术研究，渐渐地有更多的公司和研究机构以及编程爱好者加入了这一行列，讨论的问题也从基本的语法和使用渐渐深入到对算法、硬件架构和优化技巧的分析。在这一过程中，我们和国内的很多开发者和机构进行合作，并建立了友谊。在此，我们要特别感谢许磊给我们的指导和建议，讨论群中的刘伟峰、刘国峰、陈实富、杨常青等活跃的开发人员也提出了很多宝贵建议和意见；我们要感谢 NVIDIA 公司亚太区技术经理的邓培智先生，中国区市场总监魏鸣女士，销售经理谢强先生，以及 NVIDIA 深圳有限公司的 Terence horng，Jerry Zhou，Johnny Qiu，Blues Yu，Jerry Jia，Donovan Chen，Sandy Zou 等同事的支持；电子科技大学的窦衡副教授、杨万麟教授、沈晓峰副教授、况凌博士以及电子工程学院 1201 教研室的各位同学，西安电子科技大学的刘西洋教授、王黎明老师、高海昌老师以及香港浸会大学的褚晓文教授在本书的几位作者攻读学位期间给予的指导和支持；浪潮集团，联想集团，厦门美亚柏科资讯科技有限公司，广州明夏阳人工智能有限公司，以及中科院的张楠、王龙、张先轶等对几位作者在项目上给予了大力支持；感谢 CSDN 的路宏为我们出版本书四处奔走；感谢中国水利水电出版社的各位为了本书能够更早出版付出的巨大努力；最后，还要把最诚挚的感谢留给几位作者的家人。

GPU 和 CUDA 都还在不断发展中，本书也很难对 GPU 高性能计算涉及到的所有知识面面俱到，书中难免存在一些错误和不足之处，恳请诸位读者能够及时指出，给出宝贵的意见和建议。

张　舒

2009 年 8 月于深圳

试读结束：需要全本请在线购买：　www.ertongbook.com

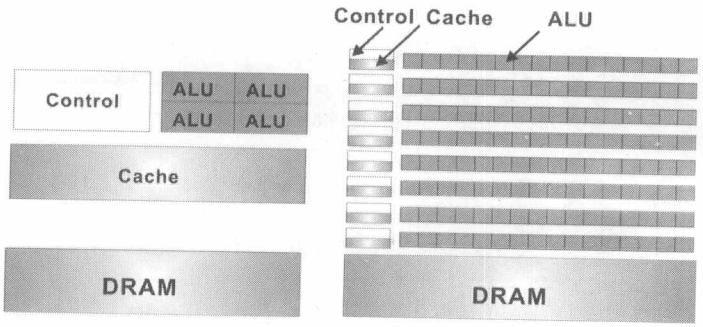
目　　录

第 1 章　GPU 通用计算

目前，主流计算机中的处理器主要是中央处理器 CPU 和图形处理器 GPU。传统上，GPU 只负责图形渲染，而大部分的处理都交给了 CPU。

21 世纪人类所面临的重要科技问题，如卫星成像数据的处理、基因工程、全球气候准确预报、核爆炸模拟等，数据规模已经达到 TB[①]甚至 PB 量级，没有万亿次以上的计算能力是无法解决的。与此同时，我们在日常应用中（如游戏、高清视频播放）面临的图形和数据计算也越来越复杂，对计算速度提出了严峻挑战。

GPU 在处理能力和存储器带宽上相对 CPU 有明显优势，在成本和功耗上也不需要付出太大代价，从而为这些问题提供了新的解决方案。由于图形渲染的高度并行性，使得 GPU 可以通过增加并行处理单元和存储器控制单元的方式提高处理能力和存储器带宽。GPU 设计者将更多的晶体管用作执行单元，而不是像 CPU 那样用作复杂的控制单元和缓存并以此来提高少量执行单元的执行效率。图 1-1 对 CPU 与 GPU 中晶体管的数量以及用途进行了比较。



（a）CPU　　（b）GPU

图 1-1　CPU 与 GPU 晶体管的使用

受游戏市场和军事视景仿真需求的牵引，GPU 性能提高速度很快。最近几年中，GPU 的性能每一年就可以翻倍，大大超过了 CPU 遵照摩尔定律（每 18～24 月性能翻倍）的发展速度。为了实现更逼真的图形效果，GPU 支持越来越复杂的运算，其可编程性和功能都大大扩展了。

目前，主流 GPU 的单精度浮点处理能力已经达到了同时期 CPU 的 10 倍左右，而其外部存储器带宽则是 CPU 的 5 倍左右；在架构上，目前的主流 GPU 采用了统一架构单元，并且实现了细粒度的线程间通信，大大扩展了应用范围。2006 年，随着支持 DirectX 10 的 GPU 的发布，基于 GPU 的通用计算（General Purpose GPU，GPGPU）的普及条件成熟了。

NVIDIA 公司于 2007 年正式发布的 CUDA（Compute Unified Device Architecture，计算统一设备架构）是第一种不需借助图形学 API 就可以使用类 C 语言进行通用计算的开发环境和软件体系。与以往的传统 GPGPU 开发方式相比，CUDA 有十分显著的改进。经过两年多的发展，CUDA 与支持 CUDA 的 GPU 在性能上有显著提高，功能上也在不断完善。

---

① Tera Byte，2^{40} Byte；Peta Byte，2^{50} Byte。

由于在性能、成本和开发时间上较传统的 CPU 解决方案有显著优势，CUDA 的推出在学术界和产业界引起了热烈反响。现在，CUDA 已经在金融、石油、天文学、流体力学、信号处理、电磁仿真、模式识别、图像处理、视频压缩等领域获得广泛应用，并取得了丰硕的成果。随着 GPU 的进一步发展和更多的开发人员参与到 GPU 通用计算的开发中来，可以预见，GPU 将在未来的计算机架构中扮演更加重要的角色，甚至许多以往被认为不可能的应用也会因为 GPU 的强大处理能力而成为现实。

1.1 多核计算的发展

并行是一个广义的概念，根据实现层次的不同，可以分为几种方式。如图 1-2 所示，最微观的是单核指令级并行（ILP），让单个处理器的执行单元可以同时执行多条指令；向上一层是多核（multi-core）并行，即在一个芯片上集成多个处理器核心，实现线程级并行（TLP）；再往上是多处理器（multi-processor）并行，在一块电路板上安装多个处理器，实现线程和进程级并行；最后，可以借助网络实现大规模的集群或者分布式并行，每个节点（node）就是一台独立的计算机，实现更大规模的并行计算。

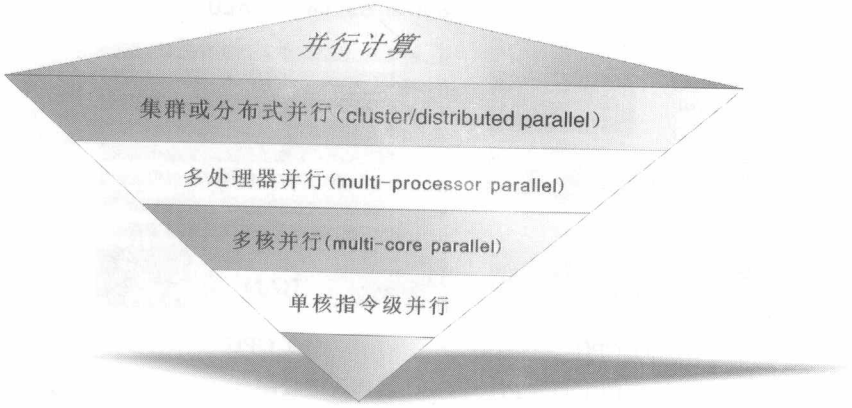


图 1-2 并行层次图

伴随着并行架构的发展，并行算法也在不断成熟与完善。从历史上看，并行算法研究的高峰是在 20 世纪七八十年代。这一阶段，出现了很多采用不同互连架构和存储器模型的 SIMD 机，以及许多优秀的并行算法，在整个并行算法研究历史上谱写了辉煌的一页。20 世纪 90 年代中期以后，并行算法的研究更加注重面向应用，不但研究并行算法的设计与分析，也兼顾并行机体系结构与并行应用程序的设计。

受工艺、材料和功耗的物理限制，处理器的频率不会在短时间内有飞跃式的提高，因此采用各种并行方式提高计算能力势在必行。现代 CPU 的每个核心中都采用了超标量[②]、超级流水线[③]、超长指令字[④]、SIMD[⑤]、超线程、分支预测等手段发掘程序内的指令级并行，并且主流

[②] 超标量体系结构的微处理器设计，通过在 CPU 上内置多条流水线，使 CPU 在一个时钟周期内可以执行多条指令，以空间换取时间。

[③] 也叫深度流水线。通过细化流水、提高主频，使得在一个周期内完成一个或多个操作，以时间换取空间。

的 CPU 中也有多个处理器核心。而 GPU 与生俱来就是一种众核并行处理器，在处理单元的数量上还要远远超过 CPU。

目前计算机中的处理器主流是拥有 2～8 个核心的多核 CPU，及拥有数十乃至上百个核心的众核 GPU，并且两者的核心数量仍会继续增加。多核和众核处理器的普及使得并行处理走入了寻常百姓家，不再是大型机和集群的专利。这也使得并行算法和并行编程成为程序员必须了解和掌握的内容。

1.1.1　CPU 多核并行

过去 20 年间，Intel、AMD 等厂家推出的 CPU 性能在不断提高，但发展速度与 20 世纪 80 年代末 90 年代初的飞跃相比，已经不能相提并论。按照摩尔定律，每 18～24 个月芯片上可集成的晶体管数量就会翻倍，这一速度带来了处理能力和速度的飞速增长。

CPU 提高单个核心性能的主要手段是提高处理器的工作频率，以及增加指令级并行。这两种传统的手段都遇到了问题：随着制造工艺的不断提高，晶体管的尺寸越来越接近原子的数量级，漏电流问题愈加显著，单位尺寸上的能耗和发热也越来越大，使得处理器的频率提高速度越来越慢。另一方面，通用计算中的指令级并行并不多，因此费尽苦心设计获得的性能提高与投入的大量晶体管相比，显得很不划算。使用流水线可以提高指令级并行，但更多更深的流水线可能会导致效率问题，例如，Intel 的 Pentium 4 架构中有更长的流水线，但运算性能还不及同频的 Pentium III。为了实现更高的指令级并行，就必须用复杂的猜测执行机制和大块的缓存保证指令和数据的命中率。现代 CPU 的分支预测正确率已经达到了 95%以上，没有什么提高余地。缓存的大小对 CPU 的性能有很大影响，过去低端的赛扬和主流的奔腾处理器用于计算的单元基本一样，主要区别就在于缓存的大小。但是继续增加缓存大小，最多也就是让真正用于计算的少量执行单元满负荷运行，这显然无助于 CPU 性能的进一步提高。

由于上述原因限制了单核 CPU 性能的进一步提高，CPU 厂商开始在单块芯片内集成更多的处理器核心，使 CPU 向多核方向发展。2005 年，Intel 和 AMD 正式向主流消费市场推出了双核 CPU 产品，2007 年推出 4 核 CPU，2009 年 Intel CPU 进入 8 核时代，Intel CPU 核心数量及制程的变化如图 1-3 所示。随着多核 CPU 的普及，现代的普通 PC 都拥有数个 CPU 核心，实际上已经相当于一个小型集群。可以预见，未来 CPU 中的核心数量还将进一步增长。与此同时，多核架构对传统的系统结构也提出了新的挑战，如存储器壁垒、芯片、板极、系统级均衡设计以及可移植性等方面的问题。

随着 CPU 从单核发展为多核，越来越多的程序员也意识到了多线程编程的重要性。多线程编程既可以在多个 CPU 核心间实现线程级并行，也可以通过超线程等技术更好地利用每一个核心内的资源，充分利用 CPU 的计算能力。除了直接使用操作系统提供的线程管理 API 实现多线程外，还可以通过一些库或者语言扩展等方法实现多线程并行计算。目前常见的多线程编程语言扩展有 OpenMP 和 Intel 的 TBB（Thread Building Block）等。

---

④ VLIW，这里的指令是由编译器提取出来的可并行的若干指令组成的长指令，一条长指令用来实现多个操作的并行。这减少了内存访问，提高了执行单元的利用率。

⑤ Flynn 分类法，见表 1-1。

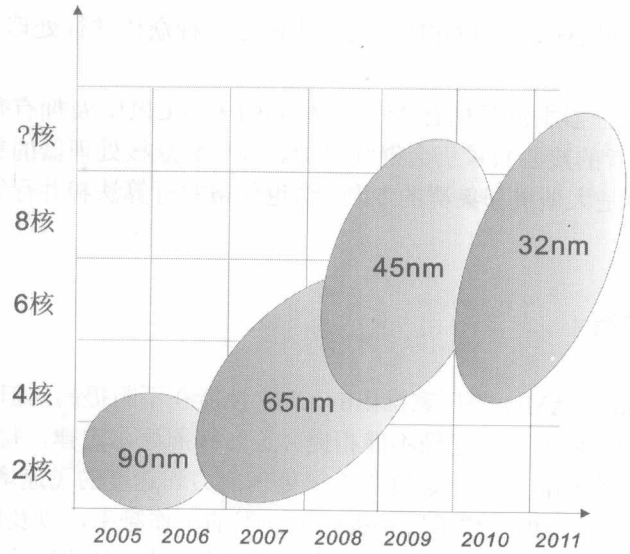


图 1-3　Intel CPU 核心数量变化

表 1-1　计算机的 Flynn 分类

分类	说明
SISD（Single Instruction Single Data）系统	传统的计算机是单 CPU，同一时刻只能执行一条指令，处理一个数据，即一个控制流和一个数据流的顺序执行。SISD 也可以并行，只要各 CPU 执行不同的指令流，处理不同数据空间的数据流，就仍是 SISD 模式
SIMD（Single Instruction Multiple Data）系统	向量机中，一个指令部件同时控制多个执行部件，一条指令可以同时操作多个数据，即一个控制流多个数据流。向量机执行操作时，向量处理单元并行处理由同类数据组成的数组，结果按若干个周期的频率输出。相对于细粒度的 SIMD，粗粒度的有 SPMD、SPMT 等，可视作 SIMD 的子类
MISD（Multiple Instructions Single Data）系统	MISD 系统结构只适用于某些特定的算法。MISD 机中，各个处理单元排成线性阵列，对流经的同一数据流，执行不同的指令流，这种应用非常有限
MIMD（Multiple Instruction Multiple Data）系统	多处理器在不同的数据流上执行不同的指令流，与 SISD 的本质区别在于不同处理器处理的指令与数据彼此相关，往往是并行地在执行同一工作的不同部分。高性能服务器与超级计算机大多是 MIMD 机

1.1.2　超级计算机、集群与分布式计算

超级计算机一般指在性能上居于世界领先地位的计算机，通常有成千上万个处理器，以及专门设计的内存和 I/O 系统。它们采用的架构与通常的个人计算机有很大区别，使用的技术也随着时代和具体应用不断变动。尽管如此，超级计算机与 PC 的关系仍然十分紧密。一部分超级计算机使用的处理器在 PC 市场上也能找到；而超级计算机使用的一些技术，如 SIMD 向量机、多核处理器，以及处理器封装技术等也都已普及到普通计算机中。支持 CUDA 的 GPU 可以看成是一个由若干个向量处理器组成的超级计算机，性能也确实可以和小型的超级计算机相比。为了将更多的资源用于计算，过去的超级计算机的界面往往都很简单，往往拆掉显卡以降低功耗。CUDA 技术推出以后，越来越多的超级计算机开始安装 GPU 以提高性能，降低计算成本。

试读结束：需要全本请在线购买：　www.ertongbook.com

计算机集群（简称集群）是一种通过一组松散集成的计算机软件和/或硬件连接起来高度紧密地协作完成计算工作的系统。分布式计算则将大量的计算任务和数据分割成小块，由多台计算机分别计算，在上传运算结果后，将结果统一合并得出结果，组织较为松散。两者一般都采用网络将各个节点连接起来，性价比都高于专门的超级计算机。例如进行蛋白质折叠计算的 Folding@home 分布式计算项目，现在有大约十万台计算机参加，计算能力甚至超过了大多数超级计算机。值得一提的是，在该项目中，仅有的 11370 颗支持 CUDA 的 GPU 提供了总计算能力的一半；而运行 Windows 的 CPU 共计 208268 颗，却只能实现 198 万亿次浮点运算，仅相当于该项目总处理能力的 6%。

目前在超级计算机、集群与分布式计算程序开发中常用的工具是 MPI（Message Passing Interface，消息传递接口）。MPI 是一个免费和开源库，可以被 C/Fortran77/C++/ Fortran90 调用，几乎得到所有并行计算机制造商的支持。它是一种消息传递编程模型，广泛应用于多类并行机，特别适用于分布式存储的并行机和服务器集群。

1.1.3　CPU+GPU 异构并行

目前主流计算机的处理能力主要来自 CPU 和 GPU。CPU 与 GPU 一般经北桥[®]通过 AGP 或者 PCI-E 总线连接，各自有独立的外部存储器，分别是内存和显存。在一些芯片组中使用的集成 GPU 没有采用独立的显存芯片，直接从内存中分出一块区域作为显存。Intel 和 AMD 提出的 CPU-GPU 融合产品还准备直接将 CPU 和 GPU 通过 QPI 或者 HT 总线连接，集成在一块芯片内。可以预见，受面积、功耗和散热的限制，此类融合产品不大可能集成性能很高的 GPU，并且有可能不为 GPU 提供独立显存。

传统的 CPU+GPU 异构并行处理的典型任务是图形实时渲染。在这类应用中，CPU 负责根据用户的输入和一定的规则（如游戏的 AI）确定在下一帧需要显示哪些物体，以及这些物体的位置，再将这些信息传递给 GPU，由 GPU 绘制这些物体并进行显示。两者的计算是并行的：在 GPU 绘制当前帧的时候，CPU 可以计算下一帧需要绘制的内容。

在这些处理中，CPU 负责的是逻辑性较强的事务计算，GPU 则负责计算密集度高的图形渲染。为了满足事务计算的需要，CPU 的设计目标是使执行单元能够以很低的延迟获得数据和指令，因此采用了复杂的控制逻辑和分支预测，以及大量的缓存来提高执行效率；而 GPU 必须在有限的面积上实现很强的计算能力和很高的存储器带宽，因此需要大量执行单元来运行更多相对简单的线程，在当前线程等待数据时就切换到另一个处于就绪状态等待计算的线程。简而言之，CPU 对延迟将更敏感，而 GPU 则侧重于提高整体的数据吞吐量。CPU 和 GPU 的设计目标的不同决定了两者在架构和性能上的巨大差异，具体来说有：

1. CPU 线程与 GPU 线程

CPU 的一个核心通常在一个时刻只能运行一个线程的指令。CPU 的多线程机制通过操作系统提供的 API 实现，是一种软件粗粒度多线程。当一个线程中断，或者等待某种资源时，操作系统就保存当前线程的上下文，并装载另外一个线程的上下文。这种机制使得 CPU 切换线程的代价十分高昂，通常需要数百个时钟周期。某些具有超线程功能的 CPU 可以将一个物

[®] 北桥，主板上最大最重要的芯片，负责 CPU 和内存、显卡之间的数据交换，往往覆盖有散热片或风扇。

理核心虚拟成多个核心，但每个虚拟核心在一个时刻也只能运行一个线程。

GPU 采用的则是由硬件管理的轻量级线程，可以实现零开销的线程切换。线程的切换在这里成为一件好事：当一个线程因为访问片外存储器或者同步指令开始等待以后，可以立即切换到另外一个处于就绪态的线程，用计算来隐藏延迟。当线程中的计算指令需要的时间较多，而访存相对较少，即计算密集度比较高时，延迟就可以被计算隐藏，而且线程越多，延迟隐藏得更好。

2. 多核与众核

当前的主流 CPU 中一般有 2～8 个核心，每个核心中有 3～6 条执行流水线。这些核心极采用了很多提高指令级并行的技术，如超标量超深流水线、乱序执行、预测执行，以及大容量缓存等，也采用了如 SSE、3Dnow!一类的数据级并行技术。由于缓存和控制逻辑需要很大面积，因此在一块芯片上能够集成的核心数量也就被限制了。

当前的 NVIDIA GPU 中有 1～30 个包含完整前端的流多处理器，每个流多处理器可以看成一个包含 8 个 1D 流处理器的 SIMD 处理器。CUDA 就利用了多个流多处理器间的粗粒度任务级或数据级并行，以及流多处理器内的细粒度数据并行。

尽管 GPU 的运行频率低于 GPU，但更多的执行单元数量还是使 GPU 能够在浮点处理能力上获得优势，即使是芯片组里集成的低端 GPU，在单精度浮点处理能力上也能和主流的 CPU 打成平手，而主流 GPU 的性能则可以达到同时期主流 CPU 性能的 10 倍左右，如图 1-4 所示。

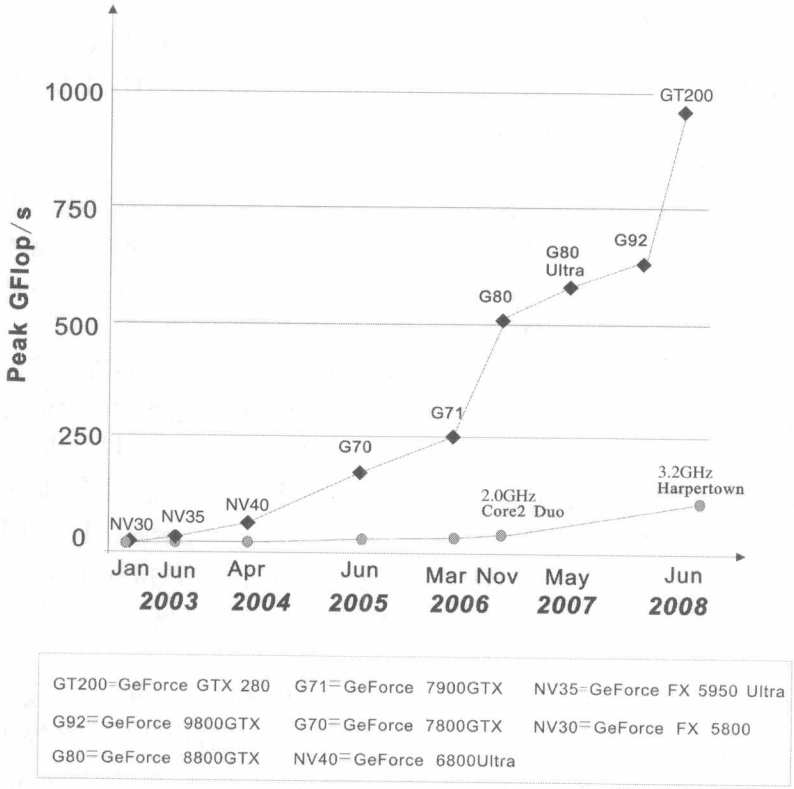


图 1-4　GPU 与 CPU 峰值浮点计算能力比较

3. 外部存储器

GT200 GPU 的显存带宽达到了 140GB/s，是同时期 CPU 最高内存带宽的 5 倍，图 1-5 是

CPU 与 GPU 同期带宽的比较。造成 GPU 与 CPU 外部存储器带宽差异的主要原因有两个：

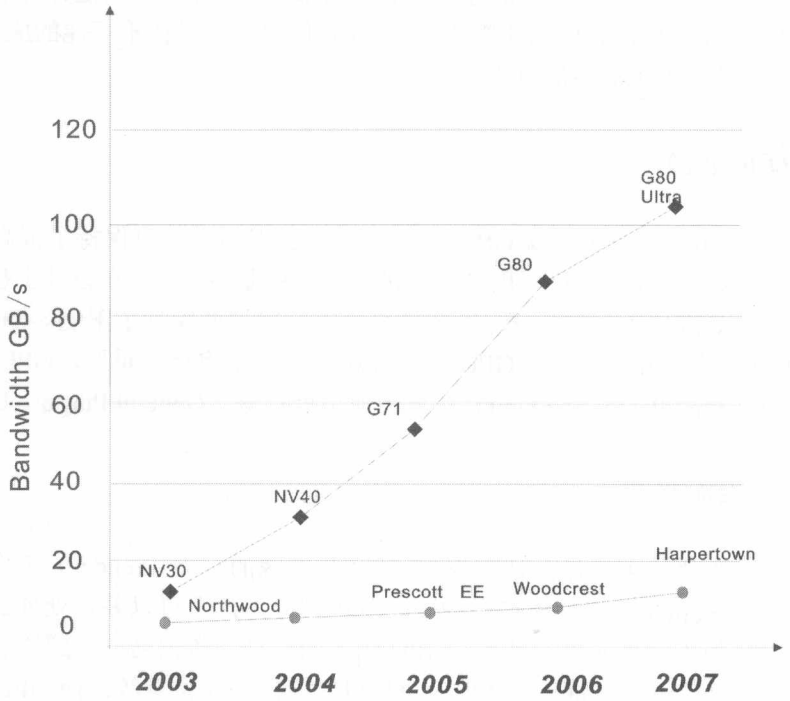


图 1-5　GPU 与 CPU 带宽比较

首先，显存中使用的 GDDR 存储器颗粒与内存的 DDR 存储器颗粒在技术上基本相同，但显存颗粒是直接固化在显卡的 PCB 板上的，而内存则为了兼顾可扩展性的需要，必须通过 DIMM 插槽与主板相连。直接焊在 PCB 板上的显存的信号完整性问题比通过插槽连接的内存更容易解决，显存的工作频率也通常比使用相同技术的内存要高一些。

其次，目前的 CPU 存储器控制器一般基于双通道或者三通道技术的，每个通道位宽 64bit；而 GPU 中则存在数个存储器控制单元，例如 GTX280 GPU 中就有 8 个存储器控制器，每个控制两片位宽 32bit 的显存芯片，使总的存储器位宽达到 512bit。这样，尽管内存中的 DDR 存储器颗粒比显卡上的 GDDR 存储器颗粒还要多，但 CPU 能够同时访问的存储器颗粒反而少于 GPU。

4. 缓存

CPU 中的缓存主要用于减小访存延迟和节约带宽。缓存在多线程环境下会发生失效反应：在每次线程上下文切换之后，都需要重建缓存上下文，一次缓存失效的代价是几十到上百个时钟周期。同时，为了实现缓存与内存中数据的一致性，还需要复杂的逻辑进行控制。

在 GPU 中则没有复杂的缓存体系与替换机制。GPU 缓存是只读的，因此也不用考虑缓存一致性问题。GPU 缓存的主要功能是用于过滤对存储器控制器的请求，减少对显存的访问。所以 GPU 上缓存的主要功能不是减小访存延迟，而是节约显存带宽。

从上面的分析可以看出，GPU 的主要设计目标是以大量线程实现面向吞吐量的数据并行计算，适合于处理计算密度高、逻辑分支简单的大规模数据并行任务。而 CPU 则有复杂的控制逻辑和大容量的缓存减小延迟，能够适应各种不同的情况，尤其擅长复杂逻辑运算。使用

GPU 处理数据并行任务，而由 CPU 进行复杂逻辑和事务处理等串行计算，就可以最大限度地利用计算机的处理能力，实现桌面上的超级计算。CPU+GPU 并行不只是计算能力的提高，也不只是节省了成本和能源；将高性能计算普及到桌面，使得许多以往不可能的应用成为了现实，这种灵活性本身就是一种革命性的进步。

1.2 GPU 发展简介

自 1999 年 NVIDIA 发布第一款 GPU 以来，GPU 的发展就一直保持了很高的速度。为了实时生成逼真 3D 图形，GPU 不仅采用了最先进的半导体制造工艺，在设计上也不断创新。传统上，GPU 的强大处理能力只被用于 3D 图像渲染，应用领域受到了限制。随着以 CUDA 为代表的 GPU 通用计算 API 的普及，GPU 在计算机中的作用将更加重要，GPU 的含义也可能从图形处理器（Graphic Processing Unit）扩展为通用处理器（General Purpose Unit）。

1.2.1 GPU 渲染流水线

GPU 的渲染流水线的主要任务是完成 3D 模型到图像的渲染（render）工作。常用的图形学 API（Direct3D/OpenGL）编程模型中的渲染过程被分为几个可以并行处理的阶段，分别由 GPU 中渲染流水线的不同单元进行处理。GPU 输入的模型是数据结构（或语言）定义的对三维物体的描述，包括几何、方向、物体表面材质以及光源所在位置等；而 GPU 输出的图像则是从观察点对 3D 场景观测到的二维图像。在 GPU 渲染流水线的不同阶段，需要处理的对象分别是顶点（vertex）、几何图元（primitive）、片元（fragment）、像素（pixel）。如图 1-6 所示，典型的渲染过程可以分为以下几个阶段：

1. 顶点生成

图形学 API 用简单的图元（点、线、三角形）表示物体表面。每个顶点除了(x,y,z)三维坐标属性外还有应用程序自定义属性，例如位置、颜色、标准向量等。

2. 顶点处理

本阶段主要是通过计算把三维顶点坐标映射到二维屏幕，计算各顶点的亮度值等。这个阶段是可编程的，由 vertex shader 完成。输入与输出一一对应，即一个顶点被处理后仍然是一个顶点，各顶点间的处理相互独立，可以并行完成。

3. 图元生成

根据应用程序定义的顶点拓扑逻辑，把上阶段输出的顶点组织起来形成有序的图元流。顶点拓扑逻辑定义了图元在输出流中的顺序，一个图元记录由若干顶点记录组成。

4. 图元处理

这一阶段也是可编程的，由 geometry shader 完成。输入和输出不是一一对应，一个图元被处理后可以生成 0 个或者多个图元，各图元处理也是相互独立的。本阶段输出一个新的图元流。

5. 片元生成

这一阶段将对每一个图元在屏幕空间进行采样，即光栅化。每一个采样点对应一个片元记录，记录该采样点在屏幕空间中的位置、与视点之间的距离以及通过插值获得的顶点属性等。

6. 片元处理

片元处理阶段是可编程的，由 pixel shader 完成，主要完成图形的填色功能。模拟光线和物体表面的交互作用，决定每个片元的颜色及透明程度等属性。

7. 像素操作

用每个片元的屏幕坐标来计算该片元对最终生成图像上的像素的影响程度。本阶段计算每个采样点离视点的距离，丢弃被遮挡住的片元。当来自多个图元的片元影响同一个像素时，往往都根据图元处理输出流中定义的图元位置进行像素更新。

下面以绘制一朵玫瑰为例来说明 GPU 图形流水线的工作流程。首先，GPU 从显存读取描述玫瑰 3D 外观的顶点数据，生成一批反映三角形场景位置与方向的顶点；由 vertex shader 计算 2D 坐标和亮度值，在屏幕空间绘出构成玫瑰的顶点；顶点被分组成三角形图元；geometry shader 进行进一步细化，生成更多图元；随后，GPU 中的固定功能单元对这些图元进行光栅化，生成相应的片元集合；由 pixel shader 从显存中读取纹理数据对片元上色和渲染；最后一个阶段，根据片元信息更新玫瑰图像，主要是可视度的处理。由 ROP 完成像素到帧缓冲区的输出，帧缓冲区中的数据经过 D/A 输出到显示器上以后，就可以看到绘制完成的玫瑰图像了。

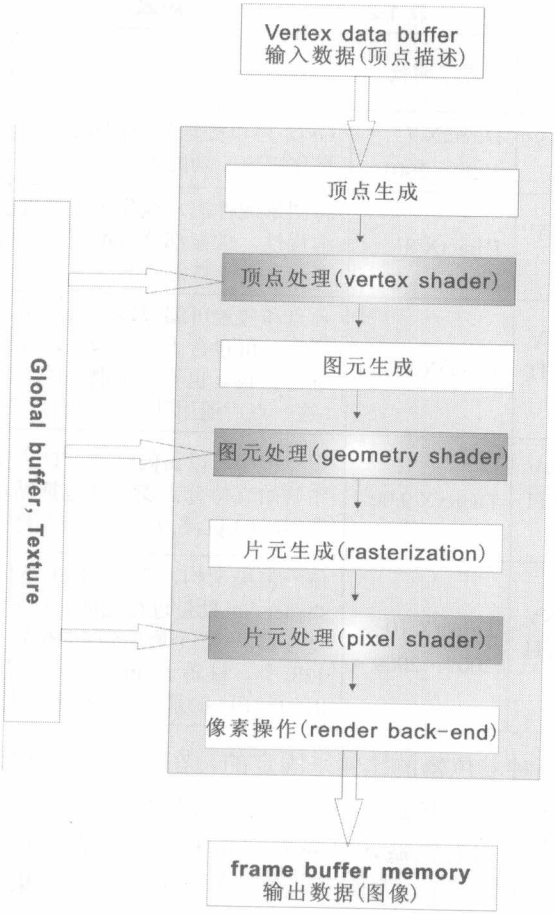


图 1-6　GPU 渲染流水线（DX10）

图形渲染过程具有内在的并行性：顶点之间、图元之间、片元之间的数据相关性很弱，对它们的计算可以独立并行进行。这使得通过并行处理提高吞吐量成为可能。

首先，渲染流水线具有时间上的功能并行。流水线的各级可以同时工作，当各级都能满负荷工作时，能够获得最高性能。

其次，渲染流水线具有数据并行性。不仅可以通过 SIMD 提高每一个可编程着色器的性能，还可以在一个 GPU 内集成多条渲染流水线实现更高的吞吐量。

1.2.2 着色器模型

在图形渲染中，GPU 中的可编程计算单元被称为着色器（shader），着色器的性能由 DirectX 中规定的 shader model 来区分。GPU 中最主要的可编程单元是顶点着色器和像素着色器。为了实现更细腻逼真的画质，GPU 的体系架构从最早的固定功能流水线到可编程流水线，再到 DirectX 10 时代的以通用的可编程计算单元为主、图形固定功能单元为辅的形式。在这一过程中，着色器的可编程性也随着架构的发展不断提高。附录 G 给出了不同版本着色器模型的性能参数，这里的表 1-2 给出了每代模型的大概特点。

表 1-2 Shader Model 版本

Shader Model	GPU 代表	显卡时代	特点
	1999 年第一代 NV GeForce 256	DirectX 7 1999～2001	GPU 可以处理顶点的矩阵变换和进行光照计算（T&L），操作固定，功能单一，不具有可编程性
SM 1.0	2001 年第二代 NV GeForce3	DirectX 8	将图形硬件流水线作为流处理器来解释，顶点部分出现可编程性，像素部分可编程性有限（访问纹理的方式和格式受限，不支持浮点）
SM 2.0	2003 年第三代 NV GeForceFx 和 ATI R300	DirectX 9.0b	顶点和像素可编程性更通用化，像素部分支持 FP16/24/32 浮点，可包含上千条指令，纹理使用更加灵活：可用索引进行查找，也不再限制在[0,1]范围，从而可用作任意数组（这一点对通用计算很重要）
SM 3.0	2004 年第四代 NV GeForce 6 和 ATI X1000	DirectX 9.0c	顶点程序可以访问纹理 VTF，支持动态分支操作，像素程序开始支持分支操作（包括循环、if/else 等），支持函数调用，64 位浮点纹理滤波和融合，多个绘制目标
SM 4.0	2007 第五代 NV G80 和 AMD/ATI R600	DirectX 10 2007～2009	统一渲染架构，支持 IEEE754 浮点标准，引入 geometry shader（可批量进行几何处理），指令数从 1K 提升至 64K，寄存器从 32 个增加到 4096 个，纹理规模从 16+4 个提升至 128 个，材质 texture 格式变为硬件支持的 RGBE 格式，8192*8192 的最高纹理分辨率比原先的 2048*2048 高很多

传统的分离架构中两种着色器的比例是固定的。在 GPU 核心设计完成时，各种着色器的数量便确定下来，比如著名的“黄金比例”——顶点着色器与像素着色器数量为 1∶3。但不同的游戏对顶点着色器和像素着色器计算能力的需求是不同的，如果场景中有大量的小三角形，顶点着色器就必须满负荷工作，而像素着色器则会被闲置；如果场景中只有少量的大三角形，又会发生相反的情况。因此，固定比例的设计无法完全发挥 GPU 中所有计算单元的性能。

试读结束：需要全本请在线购买： www.ertongbook.com