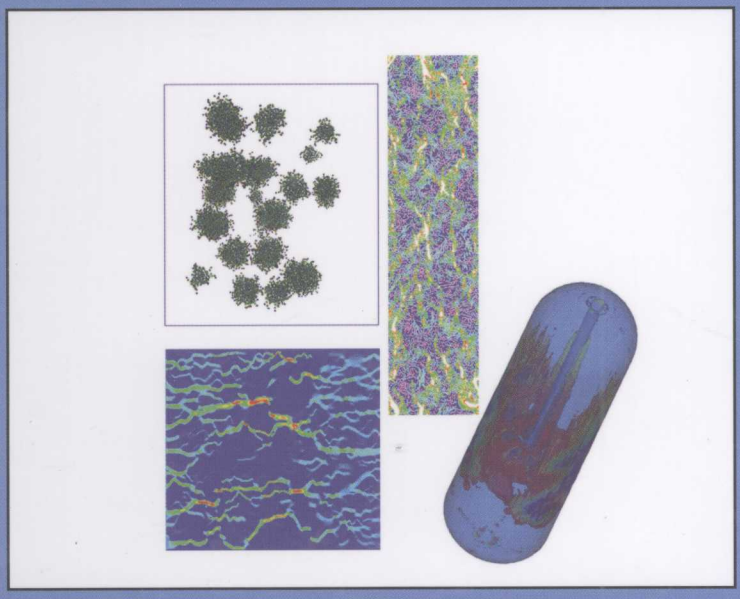
基于GPU的多尺度离散模拟并行计算

多相复杂系统国家重点实验室
多尺度离散模拟项目组 著



科学出版社

基于GPU的多尺度离散模拟并行计算

多相复杂系统国家重点实验室
多尺度离散模拟项目组 著

科学出版社
北京

内 容 简 介

本书介绍了多尺度离散模拟的基本思路、方法和不同应用领域，并就分子动力学模拟、复杂流动和多相流动模拟、数据图像分析等若干重点领域具体讨论了利用图形处理器(GPU)实现其多级并行计算的实施方案和编程技巧. 书中对现有的 GPU 编程环境及其使用方法和注意事项等从应用开发人员的角度作了比较详细的阐述.

本书可供力学、物理、化学、过程工程，以至经济和社会等领域对复杂系统的计算机模拟及其高性能计算感兴趣的研究生、科研人员和工程技术人员参考.

图书在版编目(CIP)数据

基于 GPU 的多尺度离散模拟并行计算/多相复杂系统国家重点实验室多尺度离散模拟项目组著. —北京：科学出版社，2009

ISBN 978-7-03-023942-6

Ⅰ. 基… Ⅱ. 多… Ⅲ. ①图像处理-应用-离散模拟 ②-图像处理-应用-并行算法 Ⅳ. TP301.6 O242.1

中国版本图书馆 CIP 数据核字(2009) 第 006520 号

责任编辑：林 鹏 杨 震／责任校对：鲁 素

责任印制：钱玉芬／封面设计：王 浩

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

中国科学院印刷厂 印刷

科学出版社发行 各地新华书店经销

*

2009 年 1 月第 一 版 开本：B5(720×1000)

2009 年 1 月第一次印刷 印张：13 1/2

印数：1—2 000 字数：258 000

定价：68.00 元

(如有印装质量问题，我社负责调换〈科印〉)

序

解决自然界很多复杂问题的瓶颈在于缺乏对其时空多尺度结构的认识, 这也是复杂性科学研究的焦点问题. 中国科学院过程工程研究所从 1984 年开始就致力于用多尺度方法研究气固两相系统, 逐步发展成变分多尺度方法. 为证明这一方法, 又发展了用于离散模拟的拟颗粒方法. 在推广应用变分多尺度方法的思想和用离散方法证明不同系统的稳定性条件的过程中, 逐步认识到多尺度和离散化是很多工程问题的共性. 并致力于建立针对这一共性的计算模拟方法和软件, 也试图根据这一软件的要求来设计计算机系统, 但这已超出了课题组的能力范围. 虽经多年探索, 并求助于计算机专家, 但没有成功. 2007 年 6 月, NVIDIA 发布了 CUDA 1.0, 课题组的年轻人马上认识到, CPU+GPU 为实现多尺度离散模拟提供了一个可以借用的方案. 为此, 课题组仅用短短 4 个月的时间和有限的资金, 建立了单精度峰值超过 100Tflops 的并行计算系统, 并同时准备了各种应用实例的计算程序. 这台计算机已于 2008 年 2 月正式运行.

该书就是在这个过程中, 同志们应用这台机器进行各种实例计算的经验总结. 考虑到 CPU+GPU 并行将是一个十分有前景的发展方向, 故将各方面的经验总结汇总成书, 希望对开始学习这种计算模式的同志有较大的帮助.

从不同的实际问题归纳共同属性并建立计算模式, 再根据这一共同的计算模式建立计算系统将是未来高性能计算的重要发展方向, 用 CPU+GPU 来实现多尺度离散化并行计算, 就是在这一方向上的一次有益的实践.

希望该书的出版能够为开始学习和使用 GPU 的同志提供帮助, 节约学习的时间, 尽快进入实际应用. 也希望能够推动我国并行计算模拟向应用牵引的模式发展.

由于时间仓促, 该书的系统性、逻辑性还很不完善, 更多的是各方面经验的汇集, 目的只是作为 CPU+GPU 并行这一过渡时期的参考书.

李静海

2008 年 11 月 15 日

前　言

计算机模拟是以强大的计算能力为基础, 根据已有的理论和模型, 运用数值计算方法, 在计算机上进行虚拟实验. 与物理的实验相比, 它费用低、周期短、方便灵活、适用面广, 已成为与理论和实验并列的三大研究开发手段之一. 而高效准确的模拟需要模型、计算方法、算法、计算机软件和硬件系统等方面的共同提高与密切配合. 目前, 计算元器件技术正趋近阶段性的极限, 并行计算已成为提高计算速度的主要途径, 而相应算法的开发相对滞后, 也未能更多地指导硬件设计, 使计算机峰值性能与实际能力间的差别日益扩大.

多尺度离散模拟表达了模拟对象普遍具有的多尺度结构和离散化本质、切合海量处理单元多层次组织的并行计算体系结构发展方向, 为系统提高模拟计算能力展现了广阔的前景. 本书以项目组多年来在此方向的工作为背景, 重点介绍了几种不同尺度与类型的典型离散模拟方法在图形处理器 (graphic processing unit, GPU) 与中央处理器 (central processing unit, CPU) 耦合的新型多级并行系统中的编程实现, 并从应用角度分析了模拟方法改进与程序优化的技巧, 希望能促进此项研究、推广其应用, 并得到读者的大力支持.

在引言中, 我们介绍了多尺度离散模拟的基本范畴、算法特征及与之相应的计算机体系结构, 从而引出 GPU-CPU 多级并行对离散模拟的意义. 第 1 章介绍 GPU-CPU 混合编程的基本方法以及一种主流的 GPU 编程工具 ——CUDA. 第 2 章介绍具有良好离散特性、易于并行的 CT 图像处理算法的 GPU-CPU 多级并行实现. 第 3~5 章分别以分子动力学模拟为例, 结合微流动和材料模拟, 介绍了简单运动粒子、运动晶格粒子和多元复合粒子系统的多级并行模拟. 第 6 章介绍多尺度耦合的多相系统粒子方法的多级并行模拟及其在气固悬浮体系中的应用. 第 7 章则介绍网格化的格子玻尔兹曼方法的多级并行实现及其在多孔介质流中的应用. 第 8 章简要介绍了一些其他的 GPU-CPU 编程环境, 分析了它们各自的特点与优势. 在结束语中我们展望了适合多尺度离散模拟以及 GPU-CPU 多级并行模式的更多模型与应用, 以及需要重点解决的问题.

项目组的多位同仁直接撰写了本书, 其中第 1、3 章由陈飞国主笔, 第 2 章由孟凡勇主笔, 第 4 章由侯超峰主笔, 第 5 章由徐骥主笔, 第 6 章由李博、熊勤钢主笔, 第 7 章由王小伟主笔, 第 8 章由李曦鹏主笔, 引言和结束语由葛蔚主笔. 全书由葛蔚修改统稿, 王维指导了第 2 章的工作并修改了文稿, 此外, 张云参与了第 7 章的编写工作.

本书是项目组在多尺度离散模拟方面长期工作的结果, 项目组的所有成员都对本书作出了重要贡献. 其中要特别感谢李静海院士的悉心指导和大力支持, 感谢郭力研究员和何牧君、严历、唐德翔、江鹰、秦东明和易峰等在并行计算软硬件系统的开发中给予的紧密合作与热情帮助; 感谢麻景森、王利民、高健、张家元、刘晓星、任瑛、赵辉等在模型建立和系统研制与应用中的优秀工作. 同时要感谢郭慕孙院士对本项目组工作的亲切关心与鼓励, 感谢国家自然科学基金委员会、科技部、财政部和中国科学院对本项目组工作长期和持续的资助, 以及 NVIDIA 和 AMD 公司对本工作的支持. 科学出版社的同志和白雪为本书的编辑付出了辛勤劳动, 在此一并表示感谢.

多尺度离散模拟和 GPU-CPU 多级并行都是计算机模拟领域的新课题, 我们一个项目组的积累有限, 书中难免有不正确、不全面甚至错误之处, 敬请读者不吝指正. 也希望能与各位读者共同努力, 促进计算机模拟与高性能计算的更快发展.

作　者

2008 年 11 月

目　录

第0章 引 言

计算机模拟较物理实验的优势可概括为 “多快好省”, 多 —— 能够在多种条件下、大范围内进行模拟, 突破现实条件的限制 (如微重力实验要用落塔以至航天器, 而模拟只需设重力为零); 快 —— 免去实验装置的建设和运行时间, 许多装备的中试过程以年月计, 而模拟则以天和小时计; 好 —— 模拟的任何细节都可随时获得, 没有测量的困难 (如对高温高压设备的内部状态)、误差以及对系统的干扰; 省 —— 没有建造和运行装置的成本, 没有环境影响和安全隐患, 可无限次复制和改造.

尽管有这些优势, 但是, 在目前大量工程设计和科研开发中模拟还难以撼动实验的主导地位, 其根本原因无外乎模型的准确性和计算的高效性, 而这两者也是相辅相成的. 现代科学对物质的结构和运动基本过程已有非常可信和精确的描述, 如质量、能量和动量守恒定律, 描述微观世界的薛定谔方程, 描述低速宏观世界的牛顿运动定律, 以及由此获得的多种速率方程 (如描述流动的纳维–斯托克斯方程 (Navier-Stokes equation, N-S 方程), 描述导热的傅里叶定律 (Fourier law) 和描述传质的菲克定律 (Fick's law) 等). 但直接根据这些基本原理模拟实际过程的计算量往往庞大到无法想像. 简化和粗化的模型能减少计算量但同时会损失精度, 所以必须在两者间取得平衡. 随着计算技术的发展, 应用相对底层的模型不仅成为可能, 甚至成为需求, 为说明这一点, 需要分析目前高性能计算面临的机遇和挑战.

1. 高性能计算的发展趋势

即使普通的办公室职员也能感受到近年来计算机领域的一些转折性变化, 台式机和笔记本的 CPU 主频从本世纪初的数百兆赫不断增长到 3.xGHz 后似乎停滞了, 但出现了双核、四核的 CPU, 有的台式机还配了两颗 CPU. 实际上, 以半导体为基础的计算机元器件技术发展正逐步进入一个阶段性的平台期, 电路线宽 (已达 45~22nm) 和工作频率 (已达 6GHz) 的增长都因为量子效应等原因而趋缓. 主流 CPU 厂商都开始以提高芯片内并行度的方式来维持和超越摩尔定律, 高性能计算系统的峰值提升更是倚重 CPU(核) 数的增长.

因此, 如何让大量的处理器或处理核心充分发挥其效能是当今高性能计算发展中的一个关键问题, 这里需要克服所谓存储墙 (内存访问远慢于处理器计算)、通信墙 (结点间通信带宽和延迟远低于处理器的吞吐能力) 和编程墙 (如何开发能充分利用大量处理器的并行程序) 等障碍, 而这需要软件与硬件相互配合来解决.

如果对应用的软件没有任何限制, 建立完全通用的硬件系统, 就需要全局性的

快速数据交换, 包括处理器与存储器之间以及处理器之间直接或间接的数据交换. 这样, 当处理器数量增加时, 通信方面的硬件开销必然非线性地增加, 而通信的效率也必然逐步降低, 使系统的实际速度无法随处理器数量线性增长, 成为提高机器性能的主要瓶颈. 另一方面, 开发针对特定算法和问题的专用计算机虽然可以获得很高的效率, 但其应用面狭窄, 业务量小, 无法成批生产和充分利用, 也难以成为高性能计算的主流.

因此, 有必要寻找介于通用和专用高性能计算之间的第三条道路. 我们希望找到一种能够涵盖尽可能多的数学物理模型的算法框架, 以尽量扩大其应用范围, 而同时又希望这种框架有很突出的专门性和简单性, 使针对它进行的硬件设计能极大地简化. 我们认为, 多尺度离散模拟很可能就是这样一种算法框架, 下面分别对多尺度方法和离散模拟作一简单介绍.

2. *多尺度方法*

多尺度结构几乎是所有复杂系统的共同特征, 而这些系统的模拟大多迫切需要高性能计算. 微观上, 从电子、核子到原子、分子以及纳微结构是多尺度的; 宏观上, 湍流中的涡和材料中的裂纹都具有多尺度结构; 宇观上, 从星球、星系到星系团也是多尺度结构. 仅从小尺度上描述这些结构即使可能也是不经济的, 而仅从大尺度上描述显然也是不够的. 多尺度方法就是要研究如何在大尺度模型中考虑小尺度结构的影响而又不必直接研究小尺度结构的细节. 如能成功, 自然是处理这种复杂性的切实手段.

多尺度方法总体上在 20 世纪 90 年代后才引起广泛关注, 目前它的应用也还相当有限. 单尺度方法现在依然流行, 但它们不仅会掩盖所考虑尺度之下结构的效应, 也会抹平大尺度上的结构, 从而造成显著的误差, 特别是对于那些结构敏感的过程. 按照参考文献 (Li and Kwauk 2003) 的划分, 多尺度方法大致有三种类型: ① 描述型, 即在同一模拟的不同时空区域采用不同尺度的描述; ② 关联型, 即由小尺度模拟为上一尺度的模拟提供本构关系; ③ 变分型, 即不同尺度上的模型通过稳定性条件相关联给出系统的总体描述. 按上述顺序, 它们的侧重点也从数值方法转向系统机理的理论阐述. 其中关联型方法的历史相对较长, 通过分子动力学 (molecular dynamics, MD) 模拟测量流体物性作为纳维–斯托克斯 (N-S) 方程数值求解的参量来预测多孔介质的渗透率已体现了它的基本思想. 大涡模拟 (large eddy simulation, LES, Lesieur and Metais 1996) 和雷诺应力模型也是这方面的典型实例. 但近来尝试最多的还是描述型方法, 如分子动力学模拟与有限元计算相结合描述材料中裂纹的发展 (E and Huang 2001; Huang 2002; Curtin and Miller 2003) 和微流动过程 (O'Connell and Thompson 1995; Garcia et al. 1999). 变分型方法还很不成熟, 但我们相信它对复杂性研究有着更深远的意义, 当然它的发展依赖于通过其他

多尺度方法积累的认识. 对气固两相流, 能量最小多尺度模型 (energy minimization multi-scale, EMMS (Li and Kwauk 2003, 1994)) 是这一类型中比较典型的实例.

虽然多尺度方法提供了处理复杂性的一个合理框架, 其成功还依赖于不同尺度上的模型组件的优劣. 对此, 无论从反映和探索机理的能力, 还是并行计算的效率上考虑, 离散化模拟方法都是很有吸引力的候选组件.

3. 离散模拟

所谓离散模拟方法主要是指将模拟对象离散为大量相互独立的单元, 并定义单元间相互作用的规则或方式 (如作用势等), 通过模拟这些单元的运动或变化来描述系统的演化, 并经过各种统计、分析获得所关心的模拟结果. 复杂系统往往在多个尺度上呈现离散化特征, 实际上, 这正是其多尺度结构的表现方式, 两者对复杂系统是内在统一的.

所谓单元, 在不同系统中或一个系统的不同尺度上有不同的表现. 各种粒子方法就是对力学、物理和化学系统中不同尺度结构的抽象. 在微尺度上, 粒子主要呈现分子属性, 即有显著的热运动并以简单的保守力相互作用. 像压力和黏度这样的连续介质属性需要对许多粒子进行积分统计才能获得. 分子动力学模拟 (Alder and Wainwright 1956, 1957; Rapaport 1987) 可能是微观上最早出现的粒子方法, 它将原子、原子团或分子简化为通过有势力和刚性约束等方式相互作用的质点来描述分子、分子团以至材料的微观行为. 在宏观上, 粒子可大致被看作物质微元的拉格朗日 (Lagrange) 表达, 粒子间的应力和能量耗散与它们的状态变量相关. 其典型例子是离散单元法 (distinct element method, DEM (Tsuji et al. 1993; Xu and Yu 1997; Cleary 2001; Liu et al. 2008), 用以模拟像沙石、谷物等固体颗粒物的运动) 和光滑粒子动力学 (smoothed particle hydrodynamics, SPH (Gingold and Monaghan 1977; Lucy 1977; Takeda et al. 1994; Monaghan 1992)), 移动粒子半隐式方法 (moving particle semi-implicit, MPS (Koshizuka et al. 1995; Koshizuka and Oka 1996)) 一定意义上也属此类. 而在位于两者之间的介观尺度上, 有格子玻尔兹曼 (lattice Boltzmann, LB) 方法、耗散粒子动力学 (dissipative particle dynamics, DPD(Hoogerbrugge and Koelman 1992)) 及其后续发展的一些模型 (参见 Español 1998; Español and Revenga 2003). 这类粒子既有热运动又耗散能量, 大致可认为是分子团的一种物理模型.

在宇观尺度上, 世界的离散特性也是非常明显的. 多体动力学 (如 Portegies et al. 2007) 通过计算巨大而遥远的星体间的万有引力来跟踪它们的轨迹和集体行为, 是天体动力学模拟的一种主流手段. 这种方法为探索宇宙的形成与演化及未来的航天领域提供了有力手段.

另外粒子方法也不局限于直观上能处理为粒子集合的系统. 像 DPD 和 SPH 就是针对流体的流动和材料的变形等传统上采用连续介质方法模拟的行为而构造

的粗粒化模型, 它们突破了计算量随系统自然含有的粒子数量必然增大的问题, 为粒子方法的应用开辟了更广阔的前景.

更一般的意义上, 单元还可以是社会中的人、交通流中的车辆、神经网络 (neural network) 中的结点、信号与图像处理中的点元或像素等. 尽管它们形形色色, 但对计算机模拟而言, 又具有显著的共同特征, 即具有良好的并行计算性能. 这主要表现在:

首先, 无论是自然存在的单元还是人为构造的模型单元, 它们之间的作用强度普遍随距离的增长而迅速降低. 物理粒子间的作用本质上无非是四种基本力造成的 (实为三种或更少), 其中引力和电磁力的强度与质点间距离平方成反比, 而强和弱相互作用的衰减更快, 因此一般可忽略相距足够远的粒子间的作用, 或者通过估计大量粒子的合力来代替每对粒子间的受力计算. 这就导致了局部性, 即尽管整个系统可拥有任意多个单元, 但直接决定任一单元瞬时行为的却主要是很少量的邻近单元. 这种局部性特别适合区域分解 (space decomposition) 的并行计算, 即由不同计算结点上的不同进程负责不同空间区域中单元演化的模拟, 而每个进程只与一定数量的相邻进程通信或共享内存, 从而避免计算结点间的通信拓扑随系统规模的扩大而趋于复杂, 保持线性的加速比.

同时, 一对单元间的作用函数一般可通过常微分方程描述, 而且很多情况下一个单元同时受到的各对作用是可叠加的. 也就是说, 我们可以按任意顺序分别独立处理每对单元间的作用, 加和得到各单元的新状态. 虽然在对硬球粒子或由多个粒子通过一些约束组成的复合粒子 (如链状的高分子) 的具体处理上并非那么简单, 但在稍大的尺度上, 如对复合粒子的整体, 依然具有此性质. 这种可加性十分适合进程内的多线程并行, 可将各单元与邻近单元的作用分别赋予大量不同线程而互不干扰地计算, 比如说, 由同一处理器中的多个计算核心执行, 也特别适合单指令多数据 (single instruction multiple data, SIMD) 地执行.

4. 多尺度离散模拟系统的建立

上述讨论表明, 我们可以用离散化方法对不同尺度上的复杂系统模拟提出一种比较通用的算法框架, 即多尺度离散模拟. 在此框架下, 大量单元间发生近程作用, 同时与若干上层单元相互作用, 而上层单元间又发生近程作用, 并与更上层单元相互作用, 以此类推组成多尺度结构系统. 举例来说, 社会系统有最基本的组成单元 —— 人, 人大多与他/她周围的人关系较密切, 远程的则联系较少. 但社会不仅是单个人之间的作用, 每个人还会参加不同的团体和组织, 如家庭、工作单位、社区、俱乐部等, 它们对组成或参与它们的个体的行为有约束作用, 同时作为整体也会发生相互关系, 并且能组成更高层次的团体. 在物质世界中, 从基本粒子到整个宇宙也有相似的特征.

利用此框架, 代表各种应用的各种单元间的作用方式可模块化地嵌入通用的总体算法和数据结构中, 而无需独立编写相应的计算软件. 2003~2005 年期间, 我们基于上述思想初步开发了一个面向集群并行系统和消息传递模式的粒子模拟通用软件工具库 (唐德翔等 2006), 采用了范型编程、条件编译和动态负载平衡等手段与技术, 并在分子动力学、多相流动和颗粒流的模拟中得到了成功应用.

同时针对这种算法框架, 还可以设计专门的高性能计算硬件系统. 与一般的通用高性能计算机相比, 其应用范围虽然有所缩小, 但仍会有大量的需求. 而硬件成本的降低和效率的提高将十分显著. 因此发展这样的系统将具有非常广阔的前景. 为此, 我们提出了一种以局部内存共享为基础的可扩展并行体系结构 (葛蔚, 李静海 2005), 其中包括多个计算和存储部件, 分别排成阵列, 使每个存储部件与多个与其邻近的计算部件连接; 每个计算部件与多个与其邻近的存储部件连接. 每个存储部件可仅有极少量缓存, 使得同一芯片或板卡上能集成大量的存储与计算部件, 实现细粒度的高效并行. 同时由一个多级树型网络连接所有结点, 使单元间的作用方式与系统体系结构直接对应, 并有更强的通用性. 但是这种系统的实现需要从芯片和板卡的层次出发, 非应用人员所能. 因此我们又提出了一种集群并行计算系统 (葛蔚等 2007), 它由多层排成多维阵列的计算结点组成, 同层的相邻结点之间有直接通信连接, 不同层的结点通过交换机连接. 它可以采用市售的工作站、服务器和网络设备方便地搭建.

在此集群的建设过程中, 我们注意到 GPU 以共享内存、SIMD 方式的大规模并行流处理可获得远高于 CPU 的计算速度, 并在计算–存储器件比例等方面与我们设想的局部内存共享方式类似. 由此, 我们设计了 CPU-GPU 耦合的异构集群和相应的多级并行计算模式 (葛蔚等 2008), 并于 2008 年初建成了首套实验系统. 其基本配置与性能如下:

理论峰值:	124Tflops 单精度
可用峰值:	82Tflops 单精度
计算结点:	124×HP8600 工作站
通用处理器:	248×Intel E5430 2.66GHz
图形加速卡:	200×NV Tesla C870 +
	20×NV GeForce 9800 GX2
内存:	124×16GB ECC
硬盘:	248×250GB SATA
网卡:	Broadcom BCM5755 4×RJ45
网络交换机:	华为 H3C 7506R
后处理结点:	2×HP8600 工作站
	2×32GB ECC 内存 (内部存储器)

12×500 GB SATA 硬盘
4×NV Quada FX5600 图卡
8×HP LP3065 显示器

操作系统：　CentOS 5.1/5.2 (Linux)

编程环境：　C/C++, MPI, CUDA

这套系统被命名为 Mole-9.7, 因为其理论峰值大致为 $10^{-9.7}$Moleflops(Mole, 简写为 mol(摩尔), 是微观粒子数的基本计量单位, 1mol=6.023×10^{23}). 本书下面各章将详细介绍我们在此系统上的编程经验.

第1章 CUDA 使用初步

1.1 GPU 介 绍

GPU 是英文 graphic processing unit 的缩写, 汉语的意思就是图形处理器. 早期的计算机也需要处理图形, 但相对简单, 多数工作都交由中央处理器 (central processing unit, CPU) 来完成. 其实当时的业界也有一些图形处理产品, 例如 20 世纪 80 年代的 GE(geometry engine), 但是其功能还是较弱, 直到 NVIDIA 公司在 1999 年推出了具有标志意义的图形处理器 GeForce256, GPU 才真正开始了迅速的发展. 到目前为止, GPU 已经过了八代的发展, 每一代都拥有比前一代更强的性能和更完善的可编程架构. 第一代 GPU(到 1998 年为止) 包括 NVIDIA 的 TNT2, ATI 的 Rage 和 3dfx 的 Voodoo3; 第二代 GPU(1999~2000 年) 包括 NVIDIA 的 GeForce256 和 GeForce2, ATI 的 RV200, S3 的 Savage3D; 第三代 GPU(2001 年) 包括 NVIDIA 的 GeForce3 和 GeForce4 Ti, 微软的 Xbox, 及 ATI 的 R200; 第四代 GPU(2003 年) 包括 NVIDIA 的 GeForce FX(具有 CineFX 架构), ATI 的 R300; 第五代 GPU(2004 年) 主要以 NVIDIA 的 GeForce 6800 为代表; 第六代 GPU(2006 年) 主要以 NVIDIA GeForce 7800 为代表; 第七代 GPU(2007 年) 主要以 NVIDIA 的 GeForce 8800 为代表; 目前的第八代 GPU(2008 年) 主要有 NVIDIA 的 GeForce 200 系列以及 AMD 的 RV670、RV770 等. GPU 的计算能力也在过去 10 年间有了飞速的发展, 基本上是平均每 6 个月就有性能翻倍的产品面市. 同时, GPU 的计算性能发展速度也大大快于 CPU, 如图 1.1 所示.

目前, 高端的 GPU 计算性能已经达到 Teraflops(每秒万亿次) 级别, 相当于一个高性能计算集群系统, 远远大于主流 CPU 的计算能力. GPU 的极高计算性能也吸引了很多人的关注, 人们期望能够将 GPU 用于图形显示以外的通用计算领域, 如数据处理, 科学计算等.

从微架构上看 (图 1.2), CPU 擅长的是像操作系统、系统软件和通用应用程序这类拥有复杂指令调度、循环、分支、逻辑判断以及执行等程序任务. 程序逻辑的复杂度也限定了程序执行的指令并行性, 在 CPU 上基本无法完成上百个线程并行执行程序. GPU 擅长的是图形类的或者是非图形类的高度并行数值计算, GPU 可以容纳上千个没有逻辑关系的数值计算线程, 它的优势是无逻辑关系数据的并行计算. 目前 GPU 数值计算的优势主要是浮点运算, 它执行浮点运算快是靠大量