

# 目 录 *Contents*

译者序	
推荐序	
自序	
作者简介	
技术审校者简介	
前言	
致谢	
<b>第 1 章 基于 CUDA 的异构并行计算</b> ..... 1	
1.1 并行计算..... 1	
1.1.1 串行编程和并行编程..... 2	
1.1.2 并行性..... 3	
1.1.3 计算机架构..... 4	
1.2 异构计算..... 6	
1.2.1 异构架构..... 7	
1.2.2 异构计算范例..... 9	
1.2.3 CUDA: 一种异构计算平台..... 10	
1.3 用 GPU 输出 Hello World..... 12	
1.4 使用 CUDA C 编程难吗..... 15	
1.5 总结..... 16	
1.6 习题..... 16	
<b>第 2 章 CUDA 编程模型</b> ..... 18	
2.1 CUDA 编程模型概述..... 18	
2.1.1 CUDA 编程结构..... 19	
2.1.2 内存管理..... 20	
2.1.3 线程管理..... 24	
2.1.4 启动一个 CUDA 核函数..... 29	
2.1.5 编写核函数..... 30	
2.1.6 验证核函数..... 31	
2.1.7 处理错误..... 32	
2.1.8 编译和执行..... 32	
2.2 给核函数计时..... 35	
2.2.1 用 CPU 计时器计时..... 35	
2.2.2 用 nvprof 工具计时..... 39	
2.3 组织并行线程..... 40	
2.3.1 使用块和线程建立矩阵索引..... 40	
2.3.2 使用二维网格和二维块对 矩阵求和..... 44	
2.3.3 使用一维网格和一维块对 矩阵求和..... 47	
2.3.4 使用二维网格和一维块对 矩阵求和..... 48	
2.4 设备管理..... 50	
2.4.1 使用运行时 API 查询 GPU 信息..... 50	
2.4.2 确定最优 GPU..... 53	
2.4.3 使用 nvidia-smi 查询 GPU	

信息 .....	53	3.6 动态并行 .....	104
2.4.4 在运行时设置设备 .....	54	3.6.1 嵌套执行 .....	105
2.5 总结 .....	54	3.6.2 在 GPU 上嵌套 Hello World .....	106
2.6 习题 .....	55	3.6.3 嵌套归约 .....	109
<b>第 3 章 CUDA 执行模型</b> .....	56	3.7 总结 .....	113
3.1 CUDA 执行模型概述 .....	56	3.8 习题 .....	113
3.1.1 GPU 架构概述 .....	57	<b>第 4 章 全局内存</b> .....	115
3.1.2 Fermi 架构 .....	59	4.1 CUDA 内存模型概述 .....	115
3.1.3 Kepler 架构 .....	61	4.1.1 内存层次结构的优点 .....	116
3.1.4 配置文件驱动优化 .....	65	4.1.2 CUDA 内存模型 .....	117
3.2 理解线程束执行的本质 .....	67	4.2 内存管理 .....	124
3.2.1 线程束和线程块 .....	67	4.2.1 内存分配和释放 .....	124
3.2.2 线程束分化 .....	69	4.2.2 内存传输 .....	125
3.2.3 资源分配 .....	74	4.2.3 固定内存 .....	127
3.2.4 延迟隐藏 .....	76	4.2.4 零拷贝内存 .....	128
3.2.5 占用率 .....	78	4.2.5 统一虚拟寻址 .....	133
3.2.6 同步 .....	81	4.2.6 统一内存寻址 .....	134
3.2.7 可扩展性 .....	82	4.3 内存访问模式 .....	135
3.3 并行性的表现 .....	83	4.3.1 对齐与合并访问 .....	135
3.3.1 用 nvprof 检测活跃的线程束 .....	84	4.3.2 全局内存读取 .....	137
3.3.2 用 nvprof 检测内存操作 .....	85	4.3.3 全局内存写入 .....	145
3.3.3 增大并行性 .....	86	4.3.4 结构体数组与数组结构体 .....	147
3.4 避免分支分化 .....	88	4.3.5 性能调整 .....	151
3.4.1 并行归约问题 .....	88	4.4 核函数可达到的带宽 .....	154
3.4.2 并行归约中的分化 .....	89	4.4.1 内存带宽 .....	154
3.4.3 改善并行归约的分化 .....	93	4.4.2 矩阵转置问题 .....	155
3.4.4 交错配对的归约 .....	95	4.5 使用统一内存的矩阵加法 .....	167
3.5 展开循环 .....	97	4.6 总结 .....	171
3.5.1 展开的归约 .....	97	4.7 习题 .....	172
3.5.2 展开线程的归约 .....	99	<b>第 5 章 共享内存和常量内存</b> .....	174
3.5.3 完全展开的归约 .....	101	5.1 CUDA 共享内存概述 .....	174
3.5.4 模板函数的归约 .....	102		

5.1.1 共享内存	175	第 6 章 流和并发	230
5.1.2 共享内存分配	176	6.1 流和事件概述	231
5.1.3 共享内存存储体和访问模式	176	6.1.1 CUDA 流	231
5.1.4 配置共享内存量	181	6.1.2 流调度	234
5.1.5 同步	183	6.1.3 流的优先级	235
5.2 共享内存的数据布局	185	6.1.4 CUDA 事件	235
5.2.1 方形共享内存	185	6.1.5 流同步	237
5.2.2 矩形共享内存	193	6.2 并发内核执行	240
5.3 减少全局内存访问	199	6.2.1 非空流中的并发内核	240
5.3.1 使用共享内存的并行归约	199	6.2.2 Fermi GPU 上的虚假依赖关系	242
5.3.2 使用展开的并行归约	202	6.2.3 使用 OpenMP 的调度操作	244
5.3.3 使用动态共享内存的并行归约	204	6.2.4 用环境变量调整流行为	245
5.3.4 有效带宽	205	6.2.5 GPU 资源的并发限制	246
5.4 合并的全局内存访问	205	6.2.6 默认流的阻塞行为	247
5.4.1 基准转置内核	205	6.2.7 创建流间依赖关系	248
5.4.2 使用共享内存的矩阵转置	207	6.3 重叠内核执行和数据传输	249
5.4.3 使用填充共享内存的矩阵转置	210	6.3.1 使用深度优先调度重叠	249
5.4.4 使用展开的矩阵转置	211	6.3.2 使用广度优先调度重叠	252
5.4.5 增大并行性	214	6.4 重叠 GPU 和 CPU 执行	254
5.5 常量内存	215	6.5 流回调	255
5.5.1 使用常量内存实现一维模板	215	6.6 总结	256
5.5.2 与只读缓存的比较	217	6.7 习题	257
5.6 线程束洗牌指令	219	第 7 章 调整指令级原语	258
5.6.1 线程束洗牌指令的不同形式	220	7.1 CUDA 指令概述	259
5.6.2 线程束内的共享数据	222	7.1.1 浮点指令	259
5.6.3 使用线程束洗牌指令的并行归约	226	7.1.2 内部函数和标准函数	261
5.7 总结	227	7.1.3 原子操作指令	262
5.8 习题	228	7.2 程序优化指令	264
		7.2.1 单精度与双精度的比较	264
		7.2.2 标准函数与内部函数的比较	266
		7.2.3 了解原子指令	272

7.2.4 综合范例 .....	277	8.7 CUDA 函数库的性能研究 .....	310
7.3 总结 .....	279	8.7.1 cuSPARSE 与 MKL 的比较 .....	310
7.4 习题 .....	280	8.7.2 cuBLAS 与 MKL BLAS 的比较 .....	311
<b>第 8 章 GPU 加速库和 OpenACC</b> .....	<b>281</b>	8.7.3 cuFFT 与 FFTW 及 MKL 的比较 .....	311
8.1 CUDA 库概述 .....	282	8.7.4 CUDA 库性能小结 .....	312
8.1.1 CUDA 库支持的作用域 .....	283	8.8 OpenACC 的使用 .....	312
8.1.2 通用的 CUDA 库工作流 .....	283	8.8.1 OpenACC 计算指令的使用 .....	315
8.2 cuSPARSE 库 .....	285	8.8.2 OpenACC 数据指令的使用 .....	321
8.2.1 cuSPARSE 数据存储格式 .....	286	8.8.3 OpenACC 运行时 API .....	325
8.2.2 用 cuSPARSE 进行格式转换 .....	289	8.8.4 OpenACC 和 CUDA 库的 结合 .....	327
8.2.3 cuSPARSE 功能示例 .....	289	8.8.5 OpenACC 小结 .....	328
8.2.4 cuSPARSE 发展中的重要 主题 .....	291	8.9 总结 .....	329
8.2.5 cuSPARSE 小结 .....	291	8.10 习题 .....	329
8.3 cuBLAS 库 .....	292	<b>第 9 章 多 GPU 编程</b> .....	<b>331</b>
8.3.1 管理 cuBLAS 数据 .....	293	9.1 从一个 GPU 到多 GPU .....	332
8.3.2 cuBLAS 功能示例 .....	294	9.1.1 在多 GPU 上执行 .....	333
8.3.3 cuBLAS 发展中的重要主题 .....	295	9.1.2 点对点通信 .....	334
8.3.4 cuBLAS 小结 .....	296	9.1.3 多 GPU 间的同步 .....	335
8.4 cuFFT 库 .....	296	9.2 多 GPU 间细分计算 .....	336
8.4.1 使用 cuFFT API .....	296	9.2.1 在多设备上分配内存 .....	336
8.4.2 cuFFT 功能示例 .....	298	9.2.2 单主机线程分配工作 .....	337
8.4.3 cuFFT 小结 .....	299	9.2.3 编译和执行 .....	337
8.5 cuRAND 库 .....	299	9.3 多 GPU 上的点对点通信 .....	338
8.5.1 拟随机数或伪随机数的 选择 .....	299	9.3.1 实现点对点访问 .....	338
8.5.2 cuRAND 库概述 .....	300	9.3.2 点对点的内存复制 .....	339
8.5.3 cuRAND 介绍 .....	303	9.3.3 统一虚拟寻址的点对点 内存访问 .....	341
8.5.4 cuRAND 发展中的重要主题 .....	306	9.4 多 GPU 上的有限差分 .....	342
8.6 CUDA 6.0 中函数库的介绍 .....	307	9.4.1 二维波动方程的模板计算 .....	342
8.6.1 Drop-In 库 .....	307		
8.6.2 多 GPU 库 .....	308		

9.4.2	多 GPU 程序的典型模式	343	10.1.2	优化因素	367
9.4.3	多 GPU 上的二维模板计算	344	10.1.3	CUDA 代码编译	370
9.4.4	重叠计算与通信	347	10.1.4	CUDA 错误处理	373
9.4.5	编译和执行	348	10.2	配置文件驱动优化	374
9.5	跨 GPU 集群扩展应用程序	350	10.2.1	使用 nvprof 寻找优化因素	375
9.5.1	CPU 到 CPU 的数据传输	351	10.2.2	使用 nvvp 指导优化	379
9.5.2	使用传统 MPI 在 GPU 和 GPU 间传输数据	353	10.2.3	NVIDIA 工具扩展	381
9.5.3	使用 CUDA-aware MPI 进行 GPU 到 GPU 的数据传输	356	10.3	CUDA 调试	383
9.5.4	使用 CUDA-aware MPI 进行节点内 GPU 到 GPU 的数据传输	357	10.3.1	内核调试	383
9.5.5	调整消息块大小	358	10.3.2	内存调试	390
9.5.6	使用 GPUDirect RDMA 技术进行 GPU 到 GPU 的数据传输	359	10.3.3	调试小结	395
9.6	总结	361	10.4	将 C 程序移植到 CUDA C 的案例研究	396
9.7	习题	362	10.4.1	评估 crypt	396
第 10 章	程序实现的注意事项	364	10.4.2	并行 crypt	397
10.1	CUDA C 的开发过程	364	10.4.3	优化 crypt	398
10.1.1	APOD 开发周期	365	10.4.4	部署 crypt	404
			10.4.5	移植 crypt 小结	407
			10.5	总结	407
			10.6	习题	407
			附录	推荐阅读	409