

# CONTENTS

<b>FOREWORD</b>	<b>xvii</b>
<b>PREFACE</b>	<b>xix</b>
<b>INTRODUCTION</b>	<b>xxi</b>
<b>CHAPTER 1: HETEROGENEOUS PARALLEL COMPUTING WITH CUDA</b>	<b>1</b>
Parallel Computing	2
Sequential and Parallel Programming	3
Parallelism	4
Computer Architecture	6
Heterogeneous Computing	8
Heterogeneous Architecture	9
Paradigm of Heterogeneous Computing	12
CUDA: A Platform for Heterogeneous Computing	14
Hello World from GPU	17
Is CUDA C Programming Difficult?	20
Summary	21
<b>CHAPTER 2: CUDA PROGRAMMING MODEL</b>	<b>23</b>
Introducing the CUDA Programming Model	23
CUDA Programming Structure	25
Managing Memory	26
Organizing Threads	30
Launching a CUDA Kernel	36
Writing Your Kernel	37
Verifying Your Kernel	39
Handling Errors	40
Compiling and Executing	40
Timing Your Kernel	43
Timing with CPU Timer	44
Timing with nvprof	47
Organizing Parallel Threads	49
Indexing Matrices with Blocks and Threads	49
Summing Matrices with a 2D Grid and 2D Blocks	53
Summing Matrices with a 1D Grid and 1D Blocks	57
Summing Matrices with a 2D Grid and 1D Blocks	58

<b>Managing Devices</b>	<b>60</b>
Using the Runtime API to Query GPU Information	61
Determining the Best GPU	63
Using nvidia-smi to Query GPU Information	63
Setting Devices at Runtime	64
<b>Summary</b>	<b>65</b>
<b>CHAPTER 3: CUDA EXECUTION MODEL</b>	<b>67</b>
<b>Introducing the CUDA Execution Model</b>	<b>67</b>
GPU Architecture Overview	68
The Fermi Architecture	71
The Kepler Architecture	73
Profile-Driven Optimization	78
<b>Understanding the Nature of Warp Execution</b>	<b>80</b>
Warps and Thread Blocks	80
Warp Divergence	82
Resource Partitioning	87
Latency Hiding	90
Occupancy	93
Synchronization	97
Scalability	98
<b>Exposing Parallelism</b>	<b>98</b>
Checking Active Warps with nvprof	100
Checking Memory Operations with nvprof	100
Exposing More Parallelism	101
<b>Avoiding Branch Divergence</b>	<b>104</b>
The Parallel Reduction Problem	104
Divergence in Parallel Reduction	106
Improving Divergence in Parallel Reduction	110
Reducing with Interleaved Pairs	112
<b>Unrolling Loops</b>	<b>114</b>
Reducing with Unrolling	115
Reducing with Unrolled Warps	117
Reducing with Complete Unrolling	119
Reducing with Template Functions	120
<b>Dynamic Parallelism</b>	<b>122</b>
Nested Execution	123
Nested Hello World on the GPU	124
Nested Reduction	128
<b>Summary</b>	<b>132</b>

<b>CHAPTER 4: GLOBAL MEMORY</b>	<b>135</b>
<b>Introducing the CUDA Memory Model</b>	<b>136</b>
Benefits of a Memory Hierarchy	136
CUDA Memory Model	137
<b>Memory Management</b>	<b>145</b>
Memory Allocation and Deallocation	146
Memory Transfer	146
Pinned Memory	148
Zero-Copy Memory	150
Unified Virtual Addressing	156
Unified Memory	157
<b>Memory Access Patterns</b>	<b>158</b>
Aligned and Coalesced Access	158
Global Memory Reads	160
Global Memory Writes	169
Array of Structures versus Structure of Arrays	171
Performance Tuning	176
<b>What Bandwidth Can a Kernel Achieve?</b>	<b>179</b>
Memory Bandwidth	179
Matrix Transpose Problem	180
<b>Matrix Addition with Unified Memory</b>	<b>195</b>
<b>Summary</b>	<b>199</b>
<b>CHAPTER 5: SHARED MEMORY AND CONSTANT MEMORY</b>	<b>203</b>
<b>Introducing CUDA Shared Memory</b>	<b>204</b>
Shared Memory	204
Shared Memory Allocation	206
Shared Memory Banks and Access Mode	206
Configuring the Amount of Shared Memory	212
Synchronization	214
<b>Checking the Data Layout of Shared Memory</b>	<b>216</b>
Square Shared Memory	217
Rectangular Shared Memory	225
<b>Reducing Global Memory Access</b>	<b>232</b>
Parallel Reduction with Shared Memory	232
Parallel Reduction with Unrolling	236
Parallel Reduction with Dynamic Shared Memory	238
Effective Bandwidth	239

<b>Coalescing Global Memory Accesses</b>	<b>239</b>
Baseline Transpose Kernel	240
Matrix Transpose with Shared Memory	241
Matrix Transpose with Padded Shared Memory	245
Matrix Transpose with Unrolling	246
Exposing More Parallelism	249
<b>Constant Memory</b>	<b>250</b>
Implementing a 1D Stencil with Constant Memory	250
Comparing with the Read-Only Cache	253
<b>The Warp Shuffle Instruction</b>	<b>255</b>
Variants of the Warp Shuffle Instruction	256
Sharing Data within a Warp	258
Parallel Reduction Using the Warp Shuffle Instruction	262
<b>Summary</b>	<b>264</b>
<b>CHAPTER 6: STREAMS AND CONCURRENCY</b>	<b>267</b>
<b>Introducing Streams and Events</b>	<b>268</b>
CUDA Streams	269
Stream Scheduling	271
Stream Priorities	273
CUDA Events	273
Stream Synchronization	275
<b>Concurrent Kernel Execution</b>	<b>279</b>
Concurrent Kernels in Non-NULL Streams	279
False Dependencies on Fermi GPUs	281
Dispatching Operations with OpenMP	283
Adjusting Stream Behavior Using Environment Variables	284
Concurrency-Limiting GPU Resources	286
Blocking Behavior of the Default Stream	287
Creating Inter-Stream Dependencies	288
<b>Overlapping Kernel Execution and Data Transfer</b>	<b>289</b>
Overlap Using Depth-First Scheduling	289
Overlap Using Breadth-First Scheduling	293
<b>Overlapping GPU and CPU Execution</b>	<b>294</b>
<b>Stream Callbacks</b>	<b>295</b>
<b>Summary</b>	<b>297</b>

**CHAPTER 7: TUNING INSTRUCTION-LEVEL PRIMITIVES 299**

<b>Introducing CUDA Instructions</b>	<b>300</b>
Floating-Point Instructions	301
Intrinsic and Standard Functions	303
Atomic Instructions	304
<b>Optimizing Instructions for Your Application</b>	<b>306</b>
Single-Precision vs. Double-Precision	306
Standard vs. Intrinsic Functions	309
Understanding Atomic Instructions	315
Bringing It All Together	322
<b>Summary</b>	<b>324</b>

**CHAPTER 8: GPU-ACCELERATED CUDA LIBRARIES AND OPENACC 327**

<b>Introducing the CUDA Libraries</b>	<b>328</b>
Supported Domains for CUDA Libraries	329
A Common Library Workflow	330
<b>The CUSPARSE Library</b>	<b>332</b>
cuSPARSE Data Storage Formats	333
Formatting Conversion with cuSPARSE	337
Demonstrating cuSPARSE	338
Important Topics in cuSPARSE Development	340
cuSPARSE Summary	341
<b>The cuBLAS Library</b>	<b>341</b>
Managing cuBLAS Data	342
Demonstrating cuBLAS	343
Important Topics in cuBLAS Development	345
cuBLAS Summary	346
<b>The cuFFT Library</b>	<b>346</b>
Using the cuFFT API	347
Demonstrating cuFFT	348
cuFFT Summary	349
<b>The cuRAND Library</b>	<b>349</b>
Choosing Pseudo- or Quasi- Random Numbers	349
Overview of the cuRAND Library	350
Demonstrating cuRAND	354
Important Topics in cuRAND Development	357

<b>CUDA Library Features Introduced in CUDA 6</b>	<b>358</b>
Drop-In CUDA Libraries	358
Multi-GPU Libraries	359
<b>A Survey of CUDA Library Performance</b>	<b>361</b>
cuSPARSE versus MKL	361
cuBLAS versus MKL BLAS	362
cuFFT versus FFTW versus MKL	363
CUDA Library Performance Summary	364
<b>Using OpenACC</b>	<b>365</b>
Using OpenACC Compute Directives	367
Using OpenACC Data Directives	375
The OpenACC Runtime API	380
Combining OpenACC and the CUDA Libraries	382
Summary of OpenACC	384
<b>Summary</b>	<b>384</b>
<b>CHAPTER 9: MULTI-GPU PROGRAMMING</b>	<b>387</b>
<b>Moving to Multiple GPUs</b>	<b>388</b>
Executing on Multiple GPUs	389
Peer-to-Peer Communication	391
Synchronizing across Multi-GPUs	392
<b>Subdividing Computation across Multiple GPUs</b>	<b>393</b>
Allocating Memory on Multiple Devices	393
Distributing Work from a Single Host Thread	394
Compiling and Executing	395
<b>Peer-to-Peer Communication on Multiple GPUs</b>	<b>396</b>
Enabling Peer-to-Peer Access	396
Peer-to-Peer Memory Copy	396
Peer-to-Peer Memory Access with Unified Virtual Addressing	398
<b>Finite Difference on Multi-GPU</b>	<b>400</b>
Stencil Calculation for 2D Wave Equation	400
Typical Patterns for Multi-GPU Programs	401
2D Stencil Computation with Multiple GPUs	403
Overlapping Computation and Communication	405
Compiling and Executing	406
<b>Scaling Applications across GPU Clusters</b>	<b>409</b>
CPU-to-CPU Data Transfer	410
GPU-to-GPU Data Transfer Using Traditional MPI	413

GPU-to-GPU Data Transfer with CUDA-aware MPI	416
Intra-Node GPU-to-GPU Data Transfer with CUDA-Aware MPI	417
Adjusting Message Chunk Size	418
GPU to GPU Data Transfer with GPUDirect RDMA	419
Summary	422
<b>CHAPTER 10: IMPLEMENTATION CONSIDERATIONS</b>	<b>425</b>
<b>The CUDA C Development Process</b>	<b>426</b>
APOD Development Cycle	426
Optimization Opportunities	429
CUDA Code Compilation	432
CUDA Error Handling	437
<b>Profile-Driven Optimization</b>	<b>438</b>
Finding Optimization Opportunities Using nvprof	439
Guiding Optimization Using nvvp	443
NVIDIA Tools Extension	446
<b>CUDA Debugging</b>	<b>448</b>
Kernel Debugging	448
Memory Debugging	456
Debugging Summary	462
<b>A Case Study in Porting C Programs to CUDA C</b>	<b>462</b>
Assessing crypt	463
Parallelizing crypt	464
Optimizing crypt	465
Deploying Crypt	472
Summary of Porting crypt	475
Summary	476
<b>APPENDIX: SUGGESTED READINGS</b>	<b>477</b>
<b>INDEX</b>	<b>481</b>