Contents

Preface xiii
Acknowledgements xix
Dedication xxi

CHAPTER 1 Introduction 1
1.1 Heterogeneous Parallel Computing 2
1.2 Architecture of a Modern GPU 8
1.3 Why More Speed or Parallelism? 10
1.4 Speeding Up Real Applications 12
1.5 Parallel Programming Languages and Models 14
1.6 Overarching Goals 16
1.7 Organization of the Book 17
References 21

CHAPTER 2 History of GPU Computing 23
2.1 Evolution of Graphics Pipelines 23
The Era of Fixed-Function Graphics Pipelines 24
Evolution of Programmable Real-Time Graphics 28
Unified Graphics and Computing Processors 31
2.2 GPGPU: An Intermediate Step 33
2.3 GPU Computing 34
Scalable GPUs 35
Recent Developments 36
Future Trends 37
References and Further Reading 37

CHAPTER 3 Introduction to Data Parallelism and CUDA C 41
3.1 Data Parallelism 42
3.2 CUDA Program Structure 43
3.3 A Vector Addition Kernel 45
3.4 Device Global Memory and Data Transfer 48
3.5 Kernel Functions and Threading 53
3.6 Summary 58
Function Declarations 59
Kernel Launch 59
Predefined Variables 59
Runtime API 60
3.7 Exercises 60
References 62

CHAPTER 4 Data-Parallel Execution Model 63
4.1 Cuda Thread Organization 64
4.2 Mapping Threads to Multidimensional Data 68
4.3 Matrix-Matrix Multiplication—A More Complex Kernel 74
4.4 Synchronization and Transparent Scalability 81
4.5 Assigning Resources to Blocks 83
4.6 Querying Device Properties 85
4.7 Thread Scheduling and Latency Tolerance 87
4.8 Summary 91
4.9 Exercises 91

CHAPTER 5 CUDA Memories 95
5.1 Importance of Memory Access Efficiency 96
5.2 CUDA Device Memory Types 97
5.3 A Strategy for Reducing Global Memory Traffic 105
5.4 A Tiled Matrix–Matrix Multiplication Kernel 109
5.5 Memory as a Limiting Factor to Parallelism 115
5.6 Summary 118
5.7 Exercises 119

CHAPTER 6 Performance Considerations 123
6.1 Warps and Thread Execution 124
6.2 Global Memory Bandwidth 132
6.3 Dynamic Partitioning of Execution Resources 141
6.4 Instruction Mix and Thread Granularity 143
6.5 Summary 145
6.6 Exercises 145
References 149

CHAPTER 7 Floating-Point Considerations 151
7.1 Floating-Point Format 152
Normalized Representation of M 152
Excess Encoding of E 153
7.2 Representable Numbers 155
7.3 Special Bit Patterns and Precision in IEEE Format 160
7.4 Arithmetic Accuracy and Rounding 161
7.5 Algorithm Considerations 162
7.6 Numerical Stability 164
7.7 Summary 169
7.8 Exercises 170
References 171

CHAPTER 8 Parallel Patterns: Convolution 173
8.1 Background 174
8.2 1D Parallel Convolution—A Basic Algorithm 179
8.3 Constant Memory and Caching 181
8.4 Tiled 1D Convolution with Halo Elements 185
8.5 A Simpler Tiled 1D Convolution—General Caching 192
8.6 Summary 193
8.7 Exercises 194

CHAPTER 9 Parallel Patterns: Prefix Sum 197
9.1 Background 198
9.2 A Simple Parallel Scan 200
9.3 Work Efficiency Considerations 204
9.4 A Work-Efficient Parallel Scan 205
9.5 Parallel Scan for Arbitrary-Length Inputs 210
9.6 Summary 214
9.7 Exercises 215
Reference 216

CHAPTER 10 Parallel Patterns: Sparse Matrix–Vector Multiplication 217
10.1 Background 218
10.2 Parallel SpMV Using CSR 222
10.3 Padding and Transposition 224
10.4 Using Hybrid to Control Padding 226
10.5 Sorting and Partitioning for Regularization 230
10.6 Summary 232
10.7 Exercises 233
References 234

CHAPTER 11 Application Case Study: Advanced MRI Reconstruction 235
11.1 Application Background 236
11.2 Iterative Reconstruction 239
11.3 Computing $F^H D$ 241
Step 1: Determine the Kernel Parallelism Structure 243
Step 2: Getting Around the Memory Bandwidth Limitation 249
Step 3: Using Hardware Trigonometry Functions 255
Step 4: Experimental Performance Tuning 259
11.4 Final Evaluation 260
11.5 Exercises 262
References 264

CHAPTER 12 Application Case Study: Molecular Visualization and Analysis 265
12.1 Application Background 266
12.2 A Simple Kernel Implementation 268
12.3 Thread Granularity Adjustment 272
12.4 Memory Coalescing 274
12.5 Summary 277
12.6 Exercises 279
References 279

CHAPTER 13 Parallel Programming and Computational Thinking 281
13.1 Goals of Parallel Computing 282
13.2 Problem Decomposition 283
13.3 Algorithm Selection 287
13.4 Computational Thinking 293
13.5 Summary 294
13.6 Exercises 294
References 295

CHAPTER 14 An Introduction to OpenCL™ 297
14.1 Background 297
14.2 Data Parallelism Model 299
14.3 Device Architecture 301
14.4 Kernel Functions 303
14.5 Device Management and Kernel Launch 304
14.6 Electrostatic Potential Map in OpenCL 307
14.7 Summary 311
14.8 Exercises 312
References 313

CHAPTER 15 Parallel Programming with OpenACC 315
15.1 OpenACC Versus CUDA C 315
15.2 Execution Model 318
15.3 Memory Model 319
15.4 Basic OpenACC Programs 320
Parallel Construct 320
Loop Construct 322
Kernels Construct 327
Data Management 331
Asynchronous Computation and Data Transfer 335
15.5 Future Directions of OpenACC 336
15.6 Exercises 337

CHAPTER 16 Thrust: A Productivity-Oriented Library for CUDA 339
16.1 Background 339
16.2 Motivation 342
16.3 Basic Thrust Features 343
Iterators and Memory Space 344
Interoperability 345
16.4 Generic Programming 347
16.5 Benefits of Abstraction 349
16.6 Programmer Productivity 349
Robustness 350
Real-World Performance 350
16.7 Best Practices 352
Fusion 353
Structure of Arrays 354
Implicit Ranges 356
16.8 Exercises 357
References 358

CHAPTER 17 CUDA FORTRAN 359
17.1 CUDA FORTRAN and CUDA C Differences 360
17.2 A First CUDA FORTRAN Program 361
17.3 Multidimensional Array in CUDA FORTRAN 363
17.4 Overloading Host/Device Routines With Generic Interfaces 364
17.5 Calling CUDA C Via `Iso_C_Binding` 367
17.6 Kernel Loop Directives and Reduction Operations 369
17.7 Dynamic Shared Memory 370
17.8 Asynchronous Data Transfers 371
17.9 Compilation and Profiling 377
17.10 Calling Thrust from CUDA FORTRAN 378
17.11 Exercises 382

CHAPTER 18 An Introduction to C++ AMP 383
18.1 Core C++ Amp Features 384
18.2 Details of the C++ AMP Execution Model 391
Explicit and Implicit Data Copies 391
Asynchronous Operation 393
Section Summary 395
18.3 Managing Accelerators 395
18.4 Tiled Execution 398
18.5 C++ AMP Graphics Features 401
18.6 Summary 405
18.7 Exercises 405

CHAPTER 19 Programming a Heterogeneous Computing Cluster 407
19.1 Background 408
19.2 A Running Example 408
19.3 MPI Basics 410
19.4 MPI Point-to-Point Communication Types 414
19.5 Overlapping Computation and Communication 421
19.6 MPI Collective Communication 431
19.7 Summary 431
19.8 Exercises 432
Reference 433

CHAPTER 20 CUDA Dynamic Parallelism 435
20.1 Background 436
20.2 Dynamic Parallelism Overview 438
20.3 Important Details 439
Launch Environment Configuration 439
API Errors and Launch Failures 439
Events 439
Streams 440
Synchronization Scope 441
20.4 Memory Visibility 442
Global Memory 442
Zero-Copy Memory 442
Constant Memory 442
Texture Memory 443
20.5 A Simple Example 444
20.6 Runtime Limitations 446
Memory Footprint 446
Nesting Depth 448
Memory Allocation and Lifetime 448
ECC Errors 449
Streams 449
Events 449
Launch Pool 449
20.7 A More Complex Example 449
Linear Bezier Curves 450
Quadratic Bezier Curves 450
Bezier Curve Calculation (Predynamic Parallelism) 450
Bezier Curve Calculation (with Dynamic Parallelism) 453
20.8 Summary 456
Reference 457

CHAPTER 21 Conclusion and Future Outlook 459
21.1 Goals Revisited 459
21.2 Memory Model Evolution 461
21.3 Kernel Execution Control Evolution 464
21.4 Core Performance 467
21.5 Programming Environment 467
21.6 Future Outlook 468
References 469

Appendix A: Matrix Multiplication Host-Only Version Source Code 471
Appendix B: GPU Compute Capabilities 481

Index 487