# CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FOREWORD</strong></td>
<td>xvii</td>
</tr>
<tr>
<td><strong>PREFACE</strong></td>
<td>xix</td>
</tr>
<tr>
<td><strong>INTRODUCTION</strong></td>
<td>xxi</td>
</tr>
<tr>
<td><strong>CHAPTER 1: HETEROGENEOUS PARALLEL COMPUTING WITH CUDA</strong></td>
<td>1</td>
</tr>
<tr>
<td>Parallel Computing</td>
<td>2</td>
</tr>
<tr>
<td>Sequential and Parallel Programming</td>
<td>3</td>
</tr>
<tr>
<td>Parallelism</td>
<td>4</td>
</tr>
<tr>
<td>Computer Architecture</td>
<td>6</td>
</tr>
<tr>
<td>Heterogeneous Computing</td>
<td>8</td>
</tr>
<tr>
<td>Heterogeneous Architecture</td>
<td>9</td>
</tr>
<tr>
<td>Paradigm of Heterogeneous Computing</td>
<td>12</td>
</tr>
<tr>
<td>CUDA: A Platform for Heterogeneous Computing</td>
<td>14</td>
</tr>
<tr>
<td>Hello World from GPU</td>
<td>17</td>
</tr>
<tr>
<td>Is CUDA C Programming Difficult?</td>
<td>20</td>
</tr>
<tr>
<td>Summary</td>
<td>21</td>
</tr>
<tr>
<td><strong>CHAPTER 2: CUDA PROGRAMMING MODEL</strong></td>
<td>23</td>
</tr>
<tr>
<td>Introducing the CUDA Programming Model</td>
<td>23</td>
</tr>
<tr>
<td>CUDA Programming Structure</td>
<td>25</td>
</tr>
<tr>
<td>Managing Memory</td>
<td>26</td>
</tr>
<tr>
<td>Organizing Threads</td>
<td>30</td>
</tr>
<tr>
<td>Launching a CUDA Kernel</td>
<td>36</td>
</tr>
<tr>
<td>Writing Your Kernel</td>
<td>37</td>
</tr>
<tr>
<td>Verifying Your Kernel</td>
<td>39</td>
</tr>
<tr>
<td>Handling Errors</td>
<td>40</td>
</tr>
<tr>
<td>Compiling and Executing</td>
<td>40</td>
</tr>
<tr>
<td>Timing Your Kernel</td>
<td>43</td>
</tr>
<tr>
<td>Timing with CPU Timer</td>
<td>44</td>
</tr>
<tr>
<td>Timing with nvprof</td>
<td>47</td>
</tr>
<tr>
<td>Organizing Parallel Threads</td>
<td>49</td>
</tr>
<tr>
<td>Indexing Matrices with Blocks and Threads</td>
<td>49</td>
</tr>
<tr>
<td>Summing Matrices with a 2D Grid and 2D Blocks</td>
<td>53</td>
</tr>
<tr>
<td>Summing Matrices with a 1D Grid and 1D Blocks</td>
<td>57</td>
</tr>
<tr>
<td>Summing Matrices with a 2D Grid and 1D Blocks</td>
<td>58</td>
</tr>
</tbody>
</table>
CHAPTER 4: GLOBAL MEMORY 135

Introducing the CUDA Memory Model 136
   Benefits of a Memory Hierarchy 136
   CUDA Memory Model 137
Memory Management 145
   Memory Allocation and Deallocation 146
   Memory Transfer 146
   Pinned Memory 148
   Zero-Copy Memory 150
   Unified Virtual Addressing 156
   Unified Memory 157
Memory Access Patterns 158
   Aligned and Coalesced Access 158
   Global Memory Reads 160
   Global Memory Writes 169
   Array of Structures versus Structure of Arrays 171
   Performance Tuning 176
What Bandwidth Can a Kernel Achieve? 179
   Memory Bandwidth 179
   Matrix Transpose Problem 180
Matrix Addition with Unified Memory 195
Summary 199

CHAPTER 5: SHARED MEMORY AND CONSTANT MEMORY 203

Introducing CUDA Shared Memory 204
   Shared Memory 204
   Shared Memory Allocation 206
   Shared Memory Banks and Access Mode 206
   Configuring the Amount of Shared Memory 212
   Synchronization 214
Checking the Data Layout of Shared Memory 216
   Square Shared Memory 217
   Rectangular Shared Memory 225
Reducing Global Memory Access 232
   Parallel Reduction with Shared Memory 232
   Parallel Reduction with Unrolling 236
   Parallel Reduction with Dynamic Shared Memory 238
   Effective Bandwidth 239
## CONTENTS

Coalescing Global Memory Accesses 239
  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

Constant Memory 250
  Implementing a 1D Stencil with Constant Memory 250
  Comparing with the Read-Only Cache 253

The Warp Shuffle Instruction 255
  Variants of the Warp Shuffle Instruction 256
  Sharing Data within a Warp 258
  Parallel Reduction Using the Warp Shuffle Instruction 262

Summary 264

## CHAPTER 6: STREAMS AND CONCURRENCY 267

Introducing Streams and Events 268
  CUDA Streams 269
  Stream Scheduling 271
  Stream Priorities 273
  CUDA Events 273
  Stream Synchronization 275

Concurrent Kernel Execution 279
  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

Overlapping Kernel Execution and Data Transfer 289
  Overlap Using Depth-First Scheduling 289
  Overlap Using Breadth-First Scheduling 293

Overlapping GPU and CPU Execution 294

Stream Callbacks 295

Summary 297
## CHAPTER 7: TUNING INSTRUCTION-LEVEL PRIMITIVES

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

## CHAPTER 8: GPU-ACCELERATED CUDA LIBRARIES AND OPENACC

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introducing the CUDA Libraries</td>
<td>328</td>
</tr>
<tr>
<td>Supported Domains for CUDA Libraries</td>
<td>329</td>
</tr>
<tr>
<td>A Common Library Workflow</td>
<td>330</td>
</tr>
<tr>
<td>The cuSPARSE Library</td>
<td>332</td>
</tr>
<tr>
<td>cuSPARSE Data Storage Formats</td>
<td>333</td>
</tr>
<tr>
<td>Formatting Conversion with cuSPARSE</td>
<td>337</td>
</tr>
<tr>
<td>Demonstrating cuSPARSE</td>
<td>338</td>
</tr>
<tr>
<td>Important Topics in cuSPARSE Development</td>
<td>340</td>
</tr>
<tr>
<td>cuSPARSE Summary</td>
<td>341</td>
</tr>
<tr>
<td>The cuBLAS Library</td>
<td>341</td>
</tr>
<tr>
<td>Managing cuBLAS Data</td>
<td>342</td>
</tr>
<tr>
<td>Demonstrating cuBLAS</td>
<td>343</td>
</tr>
<tr>
<td>Important Topics in cuBLAS Development</td>
<td>345</td>
</tr>
<tr>
<td>cuBLAS Summary</td>
<td>346</td>
</tr>
<tr>
<td>The cuFFT Library</td>
<td>346</td>
</tr>
<tr>
<td>Using the cuFFT API</td>
<td>347</td>
</tr>
<tr>
<td>Demonstrating cuFFT</td>
<td>348</td>
</tr>
<tr>
<td>cuFFT Summary</td>
<td>349</td>
</tr>
<tr>
<td>The cuRAND Library</td>
<td>349</td>
</tr>
<tr>
<td>Choosing Pseudo- or Quasi- Random Numbers</td>
<td>349</td>
</tr>
<tr>
<td>Overview of the cuRAND Library</td>
<td>350</td>
</tr>
<tr>
<td>Demonstrating cuRAND</td>
<td>354</td>
</tr>
<tr>
<td>Important Topics in cuRAND Development</td>
<td>357</td>
</tr>
</tbody>
</table>
CUDA Library Features Introduced in CUDA 6 358
  Drop-In CUDA Libraries 358
  Multi-GPU Libraries 359

A Survey of CUDA Library Performance 361
  cuSPARSE versus MKL 361
  cuBLAS versus MKL BLAS 362
  cuFFT versus FFTW versus MKL 363
  CUDA Library Performance Summary 364

Using OpenACC 365
  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
  Summary 384

CHAPTER 9: MULTI-GPU PROGRAMMING 387

Moving to Multiple GPUs 388
  Executing on Multiple GPUs 389
  Peer-to-Peer Communication 391
  Synchronizing across Multi-GPUs 392

Subdividing Computation across Multiple GPUs 393
  Allocating Memory on Multiple Devices 393
  Distributing Work from a Single Host Thread 394
  Compiling and Executing 395

Peer-to-Peer Communication on Multiple GPUs 396
  Enabling Peer-to-Peer Access 396
  Peer-to-Peer Memory Copy 396
  Peer-to-Peer Memory Access with Unified Virtual Addressing 398

Finite Difference on Multi-GPU 400
  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

Scaling Applications across GPU Clusters 409
  CPU-to-CPU Data Transfer 410
  GPU-to-GPU Data Transfer Using Traditional MPI 413
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU-to-GPU Data Transfer with CUDA-aware MPI</td>
<td>416</td>
</tr>
<tr>
<td>Intra-Node GPU-to-GPU Data Transfer with CUDA-Aware MPI</td>
<td>417</td>
</tr>
<tr>
<td>Adjusting Message Chunk Size</td>
<td>418</td>
</tr>
<tr>
<td>GPU to GPU Data Transfer with GPUDirect RDMA</td>
<td>419</td>
</tr>
<tr>
<td>Summary</td>
<td>422</td>
</tr>
<tr>
<td><strong>CHAPTER 10: IMPLEMENTATION CONSIDERATIONS</strong></td>
<td>425</td>
</tr>
<tr>
<td>The CUDA C Development Process</td>
<td>426</td>
</tr>
<tr>
<td>APOD Development Cycle</td>
<td>426</td>
</tr>
<tr>
<td>Optimization Opportunities</td>
<td>429</td>
</tr>
<tr>
<td>CUDA Code Compilation</td>
<td>432</td>
</tr>
<tr>
<td>CUDA Error Handling</td>
<td>437</td>
</tr>
<tr>
<td>Profile-Driven Optimization</td>
<td>438</td>
</tr>
<tr>
<td>Finding Optimization Opportunities Using nvprof</td>
<td>439</td>
</tr>
<tr>
<td>Guiding Optimization Using nvvp</td>
<td>443</td>
</tr>
<tr>
<td>NVIDIA Tools Extension</td>
<td>446</td>
</tr>
<tr>
<td>CUDA Debugging</td>
<td>448</td>
</tr>
<tr>
<td>Kernel Debugging</td>
<td>448</td>
</tr>
<tr>
<td>Memory Debugging</td>
<td>456</td>
</tr>
<tr>
<td>Debugging Summary</td>
<td>462</td>
</tr>
<tr>
<td>A Case Study in Porting C Programs to CUDA C</td>
<td>462</td>
</tr>
<tr>
<td>Assessing crypt</td>
<td>463</td>
</tr>
<tr>
<td>Parallelizing crypt</td>
<td>464</td>
</tr>
<tr>
<td>Optimizing crypt</td>
<td>465</td>
</tr>
<tr>
<td>Deploying Crypt</td>
<td>472</td>
</tr>
<tr>
<td>Summary of Porting crypt</td>
<td>475</td>
</tr>
<tr>
<td>Summary</td>
<td>476</td>
</tr>
<tr>
<td><strong>APPENDIX: SUGGESTED READINGS</strong></td>
<td>477</td>
</tr>
<tr>
<td><strong>INDEX</strong></td>
<td>481</td>
</tr>
</tbody>
</table>