

Contents

Preface	xxi
Acknowledgments	xxiii
About the Author	xxv

PART I **1**

Chapter 1: Background	3
1.1 Our Approach	5
1.2 Code	6
1.2.1 Microbenchmarks	6
1.2.2 Microdemos	7
1.2.3 Optimization Journeys	7
1.3 Administrative Items	7
1.3.1 Open Source	7
1.3.2 CUDA Handbook Library (chLib)	8
1.3.3 Coding Style	8
1.3.4 CUDA SDK	8
1.4 Road Map	8
 Chapter 2: Hardware Architecture	 11
2.1 CPU Configurations	11
2.1.1 Front-Side Bus	12

2.1.2 Symmetric Multiprocessors	13
2.1.3 Nonuniform Memory Access	14
2.1.4 PCI Express Integration	17
2.2 Integrated GPUs	17
2.3 Multiple GPUs	19
2.4 Address Spaces in CUDA	22
2.4.1 Virtual Addressing: A Brief History	22
2.4.2 Disjoint Address Spaces	26
2.4.3 Mapped Pinned Memory	28
2.4.4 Portable Pinned Memory	29
2.4.5 Unified Addressing	30
2.4.6 Peer-to-Peer Mappings	31
2.5 CPU/GPU Interactions	32
2.5.1 Pinned Host Memory and Command Buffers	32
2.5.2 CPU/GPU Concurrency	35
2.5.3 The Host Interface and Intra-GPU Synchronization	39
2.5.4 Inter-GPU Synchronization	41
2.6 GPU Architecture	41
2.6.1 Overview	42
2.6.2 Streaming Multiprocessors	46
2.7 Further Reading	50
Chapter 3: Software Architecture	51
3.1 Software Layers	51
3.1.1 CUDA Runtime and Driver	53
3.1.2 Driver Models	54
3.1.3 nvcc, PTX, and Microcode	57

3.2	Devices and Initialization	59
3.2.1	Device Count	60
3.2.2	Device Attributes	60
3.2.3	When CUDA Is Not Present	63
3.3	Contexts	67
3.3.1	Lifetime and Scoping	68
3.3.2	Preallocation of Resources	68
3.3.3	Address Space	69
3.3.4	Current Context Stack	69
3.3.5	Context State	71
3.4	Modules and Functions	71
3.5	Kernels (Functions)	73
3.6	Device Memory	75
3.7	Streams and Events	76
3.7.1	Software Pipelining	76
3.7.2	Stream Callbacks	77
3.7.3	The NULL Stream	77
3.7.4	Events	78
3.8	Host Memory	79
3.8.1	Pinned Host Memory	80
3.8.2	Portable Pinned Memory	81
3.8.3	Mapped Pinned Memory	81
3.8.4	Host Memory Registration	81
3.9	CUDA Arrays and Texturing	82
3.9.1	Texture References	82
3.9.2	Surface References	85

3.10 Graphics Interoperability	86
3.11 The CUDA Runtime and CUDA Driver API	87
Chapter 4: Software Environment	93
4.1 nvcc—CUDA Compiler Driver	93
4.2 ptxas—the PTX Assembler	100
4.3 cuobjdump	105
4.4 nvidia-smi	106
4.5 Amazon Web Services	109
4.5.1 Command-Line Tools	110
4.5.2 EC2 and Virtualization	110
4.5.3 Key Pairs	111
4.5.4 Availability Zones (AZs) and Regions	112
4.5.5 S3	112
4.5.6 EBS	113
4.5.7 AMIs	113
4.5.8 Linux on EC2	114
4.5.9 Windows on EC2	115
PART II	119
Chapter 5: Memory	121
5.1 Host Memory	122
5.1.1 Allocating Pinned Memory	122
5.1.2 Portable Pinned Memory	123
5.1.3 Mapped Pinned Memory	124
5.1.4 Write-Combined Pinned Memory	124

5.1.5	Registering Pinned Memory	125
5.1.6	Pinned Memory and UVA	126
5.1.7	Mapped Pinned Memory Usage	127
5.1.8	NUMA, Thread Affinity, and Pinned Memory	128
5.2	Global Memory	130
5.2.1	Pointers	131
5.2.2	Dynamic Allocations	132
5.2.3	Querying the Amount of Global Memory	137
5.2.4	Static Allocations	138
5.2.5	Memset APIs	139
5.2.6	Pointer Queries	140
5.2.7	Peer-to-Peer Access	143
5.2.8	Reading and Writing Global Memory	143
5.2.9	Coalescing Constraints	143
5.2.10	Microbenchmarks: Peak Memory Bandwidth	147
5.2.11	Atomic Operations	152
5.2.12	Texturing from Global Memory	155
5.2.13	ECC (Error Correcting Codes)	155
5.3	Constant Memory	156
5.3.1	Host and Device <code>__constant__</code> Memory	157
5.3.2	Accessing <code>__constant__</code> Memory	157
5.4	Local Memory	158
5.5	Texture Memory	162
5.6	Shared Memory	162
5.6.1	Unsize Shared Memory Declarations	163
5.6.2	Warp-Synchronous Coding	164
5.6.3	Pointers to Shared Memory	164

5.7 Memory Copy	164
5.7.1 Synchronous versus Asynchronous Memcpy	165
5.7.2 Unified Virtual Addressing	166
5.7.3 CUDA Runtime	166
5.7.4 Driver API	169
Chapter 6: Streams and Events	173
6.1 CPU/GPU Concurrency: Covering Driver Overhead	174
6.1.1 Kernel Launches	174
6.2 Asynchronous Memcpy	178
6.2.1 Asynchronous Memcpy: Host→Device	179
6.2.2 Asynchronous Memcpy: Device→Host	181
6.2.3 The NULL Stream and Concurrency Breaks	181
6.3 CUDA Events: CPU/GPU Synchronization	183
6.3.1 Blocking Events	186
6.3.2 Queries	186
6.4 CUDA Events: Timing	186
6.5 Concurrent Copying and Kernel Processing	187
6.5.1 concurrencyMemcpyKernel.cu	189
6.5.2 Performance Results	194
6.5.3 Breaking Interengine Concurrency	196
6.6 Mapped Pinned Memory	197
6.7 Concurrent Kernel Processing	199
6.8 GPU/GPU Synchronization: cudaStreamWaitEvent ()	202
6.8.1 Streams and Events on Multi-GPU: Notes and Limitations	202
6.9 Source Code Reference	202

Chapter 7: Kernel Execution	205
7.1 Overview	205
7.2 Syntax	206
7.2.1 Limitations	208
7.2.2 Caches and Coherency	209
7.2.3 Asynchrony and Error Handling	209
7.2.4 Timeouts	210
7.2.5 Local Memory	210
7.2.6 Shared Memory	211
7.3 Blocks, Threads, Warps, and Lanes	211
7.3.1 Grids of Blocks	211
7.3.2 Execution Guarantees	215
7.3.3 Block and Thread IDs	216
7.4 Occupancy	220
7.5 Dynamic Parallelism	222
7.5.1 Scoping and Synchronization	223
7.5.2 Memory Model	224
7.5.3 Streams and Events	225
7.5.4 Error Handling	225
7.5.5 Compiling and Linking	226
7.5.6 Resource Management	226
7.5.7 Summary	228
Chapter 8: Streaming Multiprocessors	231
8.1 Memory	233
8.1.1 Registers	233
8.1.2 Local Memory	234

8.1.3 Global Memory	235
8.1.4 Constant Memory	237
8.1.5 Shared Memory	237
8.1.6 Barriers and Coherency	240
8.2 Integer Support	241
8.2.1 Multiplication	241
8.2.2 Miscellaneous (Bit Manipulation)	242
8.2.3 Funnel Shift (SM 3.5)	243
8.3 Floating-Point Support	244
8.3.1 Formats	244
8.3.2 Single Precision (32-Bit)	250
8.3.3 Double Precision (64-Bit)	253
8.3.4 Half Precision (16-Bit)	253
8.3.5 Case Study: float→half Conversion	253
8.3.6 Math Library	258
8.3.7 Additional Reading	266
8.4 Conditional Code	267
8.4.1 Predication	267
8.4.2 Divergence and Convergence	268
8.4.3 Special Cases: Min, Max and Absolute Value	269
8.5 Textures and Surfaces	269
8.6 Miscellaneous Instructions	270
8.6.1 Warp-Level Primitives	270
8.6.2 Block-Level Primitives	272
8.6.3 Performance Counter	272
8.6.4 Video Instructions	272

8.6.5 Special Registers	275
8.7 Instruction Sets	275
Chapter 9: Multiple GPUs	287
9.1 Overview	287
9.2 Peer-to-Peer	288
9.2.1 Peer-to-Peer Memcpy	288
9.2.2 Peer-to-Peer Addressing	289
9.3 UVA: Inferring Device from Address	291
9.4 Inter-GPU Synchronization	292
9.5 Single-Threaded Multi-GPU	294
9.5.1 Current Context Stack	294
9.5.2 N-Body	296
9.6 Multithreaded Multi-GPU	299
Chapter 10: Texturing	305
10.1 Overview	305
10.1.1 Two Use Cases	306
10.2 Texture Memory	306
10.2.1 Device Memory	307
10.2.2 CUDA Arrays and Block Linear Addressing	308
10.2.3 Device Memory versus CUDA Arrays	313
10.3 1D Texturing	314
10.3.1 Texture Setup	314
10.4 Texture as a Read Path	317
10.4.1 Increasing Effective Address Coverage	318
10.4.2 Texturing from Host Memory	321

10.5 Texturing with Unnormalized Coordinates	323
10.6 Texturing with Normalized Coordinates	331
10.7 1D Surface Read/Write	333
10.8 2D Texturing	335
10.8.1 Microdemo: <code>tex2d_opengl.cu</code>	335
10.9 2D Texturing: Copy Avoidance	338
10.9.1 2D Texturing from Device Memory	338
10.9.2 2D Surface Read/Write	340
10.10 3D Texturing	340
10.11 Layered Textures	342
10.11.1 1D Layered Textures	343
10.11.2 2D Layered Textures	343
10.12 Optimal Block Sizing and Performance	343
10.12.1 Results	344
10.13 Texturing Quick References	345
10.13.1 Hardware Capabilities	345
10.13.2 CUDA Runtime	347
10.13.3 Driver API	349

PART III **351**

Chapter 11: Streaming Workloads	353
11.1 Device Memory	355
11.2 Asynchronous Memcpy	358
11.3 Streams	359
11.4 Mapped Pinned Memory	361
11.5 Performance and Summary	362

Chapter 12: Reduction	365
12.1 Overview	365
12.2 Two-Pass Reduction	367
12.3 Single-Pass Reduction	373
12.4 Reduction with Atomics	376
12.5 Arbitrary Block Sizes	377
12.6 Reduction Using Arbitrary Data Types	378
12.7 Predicate Reduction	382
12.8 Warp Reduction with Shuffle	382
Chapter 13: Scan	385
13.1 Definition and Variations	385
13.2 Overview	387
13.3 Scan and Circuit Design	390
13.4 CUDA Implementations	394
13.4.1 Scan-Then-Fan	394
13.4.2 Reduce-Then-Scan (Recursive)	400
13.4.3 Reduce-Then-Scan (Two Pass)	403
13.5 Warp Scans	407
13.5.1 Zero Padding	408
13.5.2 Templated Formulations	409
13.5.3 Warp Shuffle	410
13.5.4 Instruction Counts	412
13.6 Stream Compaction	414
13.7 References (Parallel Scan Algorithms)	418
13.8 Further Reading (Parallel Prefix Sum Circuits)	419

Chapter 14: N-Body	421
14.1 Introduction	423
14.1.1 A Matrix of Forces	424
14.2 Naïve Implementation	428
14.3 Shared Memory	432
14.4 Constant Memory	434
14.5 Warp Shuffle	436
14.6 Multiple GPUs and Scalability	438
14.7 CPU Optimizations	439
14.8 Conclusion	444
14.9 References and Further Reading	446
 Chapter 15: Image Processing: Normalized Correlation	 449
15.1 Overview	449
15.2 Naïve Texture-Texture Implementation	452
15.3 Template in Constant Memory	456
15.4 Image in Shared Memory	459
15.5 Further Optimizations	463
15.5.1 SM-Aware Coding	463
15.5.2. Loop Unrolling	464
15.6 Source Code	465
15.7 Performance and Further Reading	466
15.8 Further Reading	469
 Appendix A The CUDA Handbook Library	 471
A.1 Timing	471
A.2 Threading	472

A.3 Driver API Facilities	474
A.4 Shmoos	475
A.5 Command Line Parsing	476
A.6 Error Handling	477
Glossary / TLA Decoder	481
Index	487