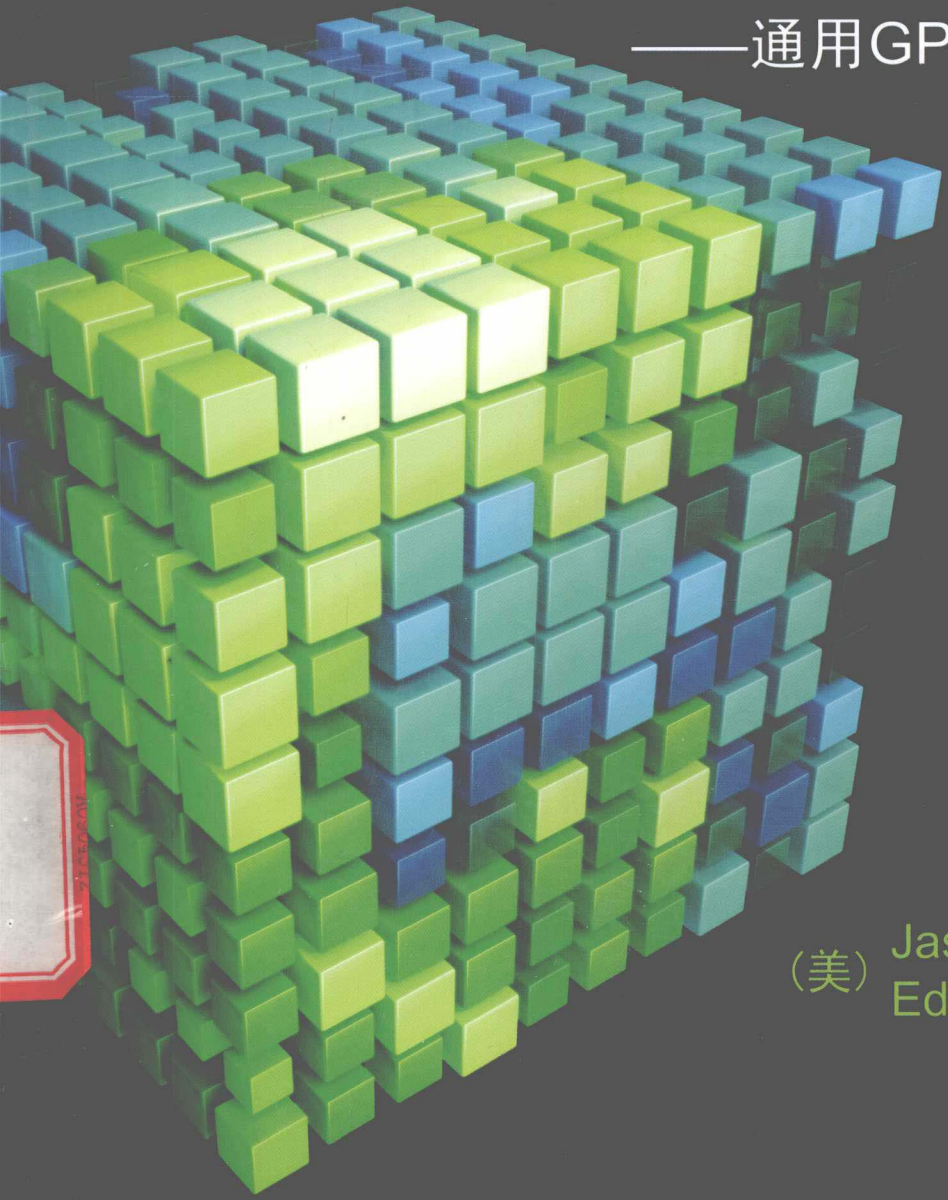
NVIDIA

CUDA 范例精解

——通用GPU编程(影印版)

CUDA by Example:
An Introduction to
General-Purpose
GPU Programming



(美) Jason Sanders
Edward Kandrot 著

清华大学出版社

CUDA 范例精解

——通用 GPU 编程(影印版)

(美) Jason Sanders
Edward Kandrot 著

清华大学出版社

北 京

北京市版权局著作权合同登记号　图字：01-2010-5495

本书封面贴有 Pearson Education**(培生教育出版集团)激光防伪标签，无标签者不得销售。**

图书在版编目(CIP)数据

CUDA 范例精解——通用 GPU 编程=CUDA by Example：An Introduction to General-Purpose GPU Programming：英文/(美) 山德尔 (Sanders,J.)，(美)康洛特(Kandrot,E.) 著. —影印本.
—北京：清华大学出版社，2010.10
ISBN 978-7-302-23995-6

Ⅰ.C… Ⅱ.①山… ②康… Ⅲ.计算机图形学－英文 Ⅳ.TP391.41

中国版本图书馆 CIP 数据核字(2010)第 191928 号

责任编辑：王　军　张立浩　　**装帧设计：**孔祥丰
责任校对：胡雁翎　　**责任印制：**王秀菊
出版发行：清华大学出版社　　**地　　址：**北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编：**100084
社　总　机：010-62770175　　**邮　　购：**010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：北京四季青印刷厂
装 订 者：三河市溧源装订厂
经　　销：全国新华书店
开　　本：185×230　**印　张：**19.25　**字　数：**431 千字
版　　次：2010 年 10 月第 1 版　　**印　　次：**2010 年 10 月第1 次印刷
印　　数：1～3000
定　　价：39.00 元

产品编号：040345-01

Foreword

Recent activities of major chip manufacturers such as NVIDIA make it more evident than ever that future designs of microprocessors and large HPC systems will be hybrid/heterogeneous in nature. These heterogeneous systems will rely on the integration of two major types of components in varying proportions:

- **Multi- and many-core CPU technology**: The number of cores will continue to escalate because of the desire to pack more and more components on a chip while avoiding the power wall, the instruction-level parallelism wall, and the memory wall.
- **Special-purpose hardware and massively parallel accelerators**: For example, GPUs from NVIDIA have outpaced standard CPUs in floating-point performance in recent years. Furthermore, they have arguably become as easy, if not easier, to program than multicore CPUs.

The relative balance between these component types in future designs is not clear and will likely vary over time. There seems to be no doubt that future generations of computer systems, ranging from laptops to supercomputers, will consist of a composition of heterogeneous components. Indeed, the *petaflop* (10^{15} floating-point operations per second) performance barrier was breached by such a system.

And yet the problems and the challenges for developers in the new computational landscape of hybrid processors remain daunting. Critical parts of the software infrastructure are already having a very difficult time keeping up with the pace of change. In some cases, performance cannot scale with the number of cores because an increasingly large portion of time is spent on data movement rather than arithmetic. In other cases, software tuned for performance is delivered years after the hardware arrives and so is obsolete on delivery. And in some cases, as on some recent GPUs, software will not run at all because programming environments have changed too much.

CUDA by Example addresses the heart of the software development challenge by leveraging one of the most innovative and powerful solutions to the problem of programming the massively parallel accelerators in recent years.

This book introduces you to programming in CUDA C by providing examples and insight into the process of constructing and effectively using NVIDIA GPUs. It presents introductory concepts of parallel computing from simple examples to debugging (both logical and performance), as well as covers advanced topics and issues related to using and building many applications. Throughout the book, programming examples reinforce the concepts that have been presented.

The book is required reading for anyone working with accelerator-based computing systems. It explores parallel computing in depth and provides an approach to many problems that may be encountered. It is especially useful for application developers, numerical library writers, and students and teachers of parallel computing.

I have enjoyed and learned from this book, and I feel confident that you will as well.

Jack Dongarra
University Distinguished Professor, University of Tennessee Distinguished Research Staff Member, Oak Ridge National Laboratory

Preface

This book shows how, by harnessing the power of your computer's graphics process unit (GPU), you can write high-performance software for a wide range of applications. Although originally designed to render computer graphics on a monitor (and still used for this purpose), GPUs are increasingly being called upon for equally demanding programs in science, engineering, and finance, among other domains. We refer collectively to GPU programs that address problems in nongraphics domains as *general-purpose*. Happily, although you need to have some experience working in C or C++ to benefit from this book, you need not have any knowledge of computer graphics. None whatsoever! GPU programming simply offers you an opportunity to build—and to build mightily—on your existing programming skills.

To program NVIDIA GPUs to perform general-purpose computing tasks, you will want to know what CUDA is. NVIDIA GPUs are built on what's known as the *CUDA Architecture*. You can think of the CUDA Architecture as the scheme by which NVIDIA has built GPUs that can perform *both* traditional graphics-rendering tasks *and* general-purpose tasks. To program CUDA GPUs, we will be using a language known as *CUDA C*. As you will see very early in this book, CUDA C is essentially C with a handful of extensions to allow programming of massively parallel machines like NVIDIA GPUs.

We've geared *CUDA by Example* toward experienced C or C++ programmers who have enough familiarity with C such that they are comfortable reading and writing code in C. This book builds on your experience with C and intends to serve as an example-driven, "quick-start" guide to using NVIDIA's CUDA C programming language. By no means do you need to have done large-scale software architecture, to have written a C compiler or an operating system kernel, or to know all the ins and outs of the ANSI C standards. However, we do not spend time reviewing C syntax or common C library routines such as `malloc()` or `memcpy()`, so we will assume that you are already reasonably familiar with these topics.

You will encounter some techniques that can be considered general parallel programming paradigms, although this book does not aim to teach general parallel programming techniques. Also, while we will look at nearly every part of the CUDA API, this book does not serve as an extensive API reference nor will it go into gory detail about every tool that you can use to help develop your CUDA C software. Consequently, we highly recommend that this book be used in conjunction with NVIDIA's freely available documentation, in particular the *NVIDIA CUDA Programming Guide* and the *NVIDIA CUDA Best Practices Guide*. But don't stress out about collecting all these documents because we'll walk you through everything you need to do.

Without further ado, the world of programming NVIDIA GPUs with CUDA C awaits!

Acknowledgments

It's been said that it takes a village to write a technical book, and *CUDA by Example* is no exception to this adage. The authors owe debts of gratitude to many people, some of whom we would like to thank here.

Ian Buck, NVIDIA's senior director of GPU computing software, has been immeasurably helpful in every stage of the development of this book, from championing the idea to managing many of the details. We also owe Tim Murray, our always-smiling reviewer, much of the credit for this book possessing even a modicum of technical accuracy and readability. Many thanks also go to our designer, Darwin Tat, who created fantastic cover art and figures on an extremely tight schedule. Finally, we are much obliged to John Park, who helped guide this project through the delicate legal process required of published work.

Without help from Addison-Wesley's staff, this book would still be nothing more than a twinkle in the eyes of the authors. Peter Gordon, Kim Boedigheimer, and Julie Nahil have all shown unbounded patience and professionalism and have genuinely made the publication of this book a painless process. Additionally, Molly Sharp's production work and Kim Wimpsett's copyediting have utterly transformed this text from a pile of documents riddled with errors to the volume you're reading today.

Some of the content of this book could not have been included without the help of other contributors. Specifically, Nadeem Mohammad was instrumental in researching the CUDA case studies we present in Chapter 1, and Nathan Whitehead generously provided code that we incorporated into examples throughout the book.

We would be remiss if we didn't thank the others who read early drafts of this text and provided helpful feedback, including Genevieve Breed and Kurt Wall. Many of the NVIDIA software engineers provided invaluable technical

assistance during the course of developing the content for *CUDA by Example*, including Mark Hairgrove who scoured the book, uncovering all manner of inconsistencies—technical, typographical, and grammatical. Steve Hines, Nicholas Wilt, and Stephen Jones consulted on specific sections of the CUDA API, helping elucidate nuances that the authors would have otherwise overlooked. Thanks also go out to Randima Fernando who helped to get this project off the ground and to Michael Schidlowsky for acknowledging Jason in his book.

And what acknowledgments section would be complete without a heartfelt expression of gratitude to parents and siblings? It is here that we would like to thank our families, who have been with us through everything and have made this all possible. With that said, we would like to extend special thanks to loving parents, Edward and Kathleen Kandrot and Stephen and Helen Sanders. Thanks also go to our brothers, Kenneth Kandrot and Corey Sanders. Thank you all for your unwavering support.

About the Authors

Jason Sanders is a senior software engineer in the CUDA Platform group at NVIDIA. While at NVIDIA, he helped develop early releases of CUDA system software and contributed to the OpenCL 1.0 Specification, an industry standard for heterogeneous computing. Jason received his master's degree in computer science from the University of California Berkeley where he published research in GPU computing, and he holds a bachelor's degree in electrical engineering from Princeton University. Prior to joining NVIDIA, he previously held positions at ATI Technologies, Apple, and Novell. When he's not writing books, Jason is typically working out, playing soccer, or shooting photos.

Edward Kandrot is a senior software engineer on the CUDA Algorithms team at NVIDIA. He has more than 20 years of industry experience focused on optimizing code and improving performance, including for Photoshop and Mozilla. Kandrot has worked for Adobe, Microsoft, and Google, and he has been a consultant at many companies, including Apple and Autodesk. When not coding, he can be found playing World of Warcraft or visiting Las Vegas for the amazing food.

To our families and friends, who gave us endless support.
To our readers, who will bring us the future.
And to the teachers who taught our readers to read.

Contents

Foreword vii

Preface ix

Acknowledgments xi

About the Authors xiii

1 WHY CUDA? WHY NOW? 1

1.1 Chapter Objectives 2

1.2 The Age of Parallel Processing 2

1.2.1 Central Processing Units 2

1.3 The Rise of GPU Computing 4

1.3.1 A Brief History of GPUs 4

1.3.2 Early GPU Computing 5

1.4 CUDA 6

1.4.1 What Is the CUDA Architecture? 7

1.4.2 Using the CUDA Architecture 7

1.5 Applications of CUDA 8

1.5.1 Medical Imaging 8

1.5.2 Computational Fluid Dynamics 9

1.5.3 Environmental Science 10

1.6 Chapter Review 11

2 GETTING STARTED 13

2.1 Chapter Objectives 14
2.2 Development Environment 14
2.2.1 CUDA-Enabled Graphics Processors 14
2.2.2 NVIDIA Device Driver 16
2.2.3 CUDA Development Toolkit 16
2.2.4 Standard C Compiler 18
2.3 Chapter Review 19

3 INTRODUCTION TO CUDA C 21

3.1 Chapter Objectives 22
3.2 A First Program 22
3.2.1 Hello, World! 22
3.2.2 A Kernel Call 23
3.2.3 Passing Parameters 24
3.3 Querying Devices 27
3.4 Using Device Properties 33
3.5 Chapter Review 35

4 PARALLEL PROGRAMMING IN CUDA C 37

4.1 Chapter Objectives 38
4.2 CUDA Parallel Programming 38
4.2.1 Summing Vectors 38
4.2.2 A Fun Example 46
4.3 Chapter Review 57

5 THREAD COOPERATION 59

5.1 Chapter Objectives 60
5.2 Splitting Parallel Blocks 60
5.2.1 Vector Sums: Redux 60
5.2.2 GPU Ripple Using Threads 69
5.3 Shared Memory and Synchronization 75
5.3.1 Dot Product 76
5.3.2 Dot Product Optimized (Incorrectly) 87
5.3.3 Shared Memory Bitmap 90
5.4 Chapter Review 94

6 CONSTANT MEMORY AND EVENTS 95

6.1 Chapter Objectives 96
6.2 Constant Memory 96
6.2.1 Ray Tracing Introduction 96
6.2.2 Ray Tracing on the GPU 98
6.2.3 Ray Tracing with Constant Memory 104
6.2.4 Performance with Constant Memory 106
6.3 Measuring Performance with Events 108
6.3.1 Measuring Ray Tracer Performance 110
6.4 Chapter Review 114

7 TEXTURE MEMORY 115

7.1 Chapter Objectives 116
7.2 Texture Memory Overview 116

7.3 Simulating Heat Transfer 117
7.3.1 Simple Heating Model 117
7.3.2 Computing Temperature Updates 119
7.3.3 Animating the Simulation 121
7.3.4 Using Texture Memory 125
7.3.5 Using Two-Dimensional Texture Memory 131
7.4 Chapter Review 137

8 GRAPHICS INTEROPERABILITY 139

8.1 Chapter Objectives 140
8.2 Graphics Interoperation 140
8.3 GPU Ripple with Graphics Interoperability 147
8.3.1 The GPUAnimBitmap Structure 148
8.3.2 GPU Ripple Redux 152
8.4 Heat Transfer with Graphics Interop 154
8.5 DirectX Interoperability 160
8.6 Chapter Review 161

9 ATOMICS 163

9.1 Chapter Objectives 164
9.2 Compute Capability 164
9.2.1 The Compute Capability of NVIDIA GPUs 164
9.2.2 Compiling for a Minimum Compute Capability 167
9.3 Atomic Operations Overview 168
9.4 Computing Histograms 170
9.4.1 CPU Histogram Computation 171
9.4.2 GPU Histogram Computation 173
9.5 Chapter Review 183

10 STREAMS 185

10.1 Chapter Objectives 186
10.2 Page-Locked Host Memory 186
10.3 CUDA Streams 192
10.4 Using a Single CUDA Stream 192
10.5 Using Multiple CUDA Streams 198
10.6 GPU Work Scheduling 205
10.7 Using Multiple CUDA Streams Effectively 208
10.8 Chapter Review 211

11 CUDA C ON MULTIPLE GPUS 213

11.1 Chapter Objectives 214
11.2 Zero-Copy Host Memory 214
11.2.1 Zero-Copy Dot Product 214
11.2.2 Zero-Copy Performance 222
11.3 Using Multiple GPUs 224
11.4 Portable Pinned Memory 230
11.5 Chapter Review 235

12 THE FINAL COUNTDOWN 237

12.1 Chapter Objectives 238
12.2 CUDA Tools 238
12.2.1 CUDA Toolkit 238
12.2.2 CUFFT 239
12.2.3 CUBLAS 239
12.2.4 NVIDIA GPU Computing SDK 240

12.2.5 NVIDIA Performance Primitives 241
12.2.6 Debugging CUDA C 241
12.2.7 CUDA Visual Profiler 243
12.3 Written Resources 244
12.3.1 Programming Massively Parallel Processors: A Hands-On Approach 244
12.3.2 CUDA U 245
12.3.3 NVIDIA Forums 246
12.4 Code Resources 246
12.4.1 CUDA Data Parallel Primitives Library 247
12.4.2 CULAtools 247
12.4.3 Language Wrappers 247
12.5 Chapter Review 248

A ADVANCED ATOMICS 249

A.1 Dot Product Revisited 250
A.1.1 Atomic Locks 251
A.1.2 Dot Product Redux: Atomic Locks 254
A.2 Implementing a Hash Table 258
A.2.1 Hash Table Overview 259
A.2.2 A CPU Hash Table 261
A.2.3 Multithreaded Hash Table 267
A.2.4 A GPU Hash Table 268
A.2.5 Hash Table Performance 276
A.3 Appendix Review 277

Index 279