Programming in Parallel with CUDA

CUDA is now the dominant language used for programming GPUs; it is one of the most exciting hardware developments of recent decades. With CUDA, you can use a desktop PC for work that would have previously required a large cluster of PCs or access to an HPC facility. As a result, CUDA is increasingly important in scientific and technical computing across the whole STEM community, from medical physics and financial modelling to big data applications and beyond.

This unique book on CUDA draws on the author's passion for and long experience of developing and using computers to acquire and analyse scientific data. The result is an innovative text featuring a much richer set of examples than found in any other comparable book on GPU computing. Much attention has been paid to the C++ coding style, which is compact, elegant and efficient. A code base of examples and supporting material is available online, which readers can build on for their own projects.

RICHARD ANSORGE is Emeritus University Senior Lecturer at the Cavendish Laboratory, University of Cambridge and Emeritus Tutor and Fellow at Fitzwilliam College, Cambridge. He is the author of over 170 peer-reviewed publications and co-author of the book *The Physics and Mathematics of MRI* (2016).

Programming in Parallel with CUDA

A Practical Guide

Richard Ansorge

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE
UNIVERSITY PRESS

University Printing House, Cambridge CB2 8BS, United Kingdom

One Liberty Plaza, 20th Floor, New York, NY 10006, USA

477 Williamstown Road, Port Melbourne, VIC 3207, Australia

314–321, 3rd Floor, Plot 3, Splendor Forum, Jasola District Centre, New Delhi – 110025, India

103 Penang Road, #05–06/07, Visioncrest Commercial, Singapore 238467

Cambridge University Press is part of the University of Cambridge.

It furthers the University's mission by disseminating knowledge in the pursuit of education, learning, and research at the highest international levels of excellence.

www.cambridge.org
Information on this title: www.cambridge.org/9781108479530
DOI: 10.1017/9781108855273

First published 2022

Printed in the United Kingdom by TJ Books Limited, Padstow Cornwall

A catalogue record for this publication is available from the British Library.

ISBN 978-1-108-47953-0 Hardback

To Catherine and Lydia

Contents

Figures

Appendix Figures

Tables

Appendix Tables

Examples

Appendix Examples

Preface

This book has been primarily written for people who need lots of computing power, including those engaged in scientific research who need this power to acquire, process, analyse or model their data. People working with medical data who need to process ever-larger data sets and more complicated image data are also likely to find this book helpful.

Complicated and demanding computations are something I have been doing for my entire research career, firstly in experimental high-energy physics and more recently in various applications of medical imaging. The advent of GPU computing is one of the most exciting developments I have yet seen, and one reason for writing this book is to share that excitement with readers.

It seems to be a corollary of Moore's law that the demand for computing power increases to always exceed what is currently available. Since the dawn of the PC age in the early 1980s, vendors have been providing supplementary cards to improve the speed of rendering displays. These cards are now known as graphics processing units or GPUs, and, driven by the demands of the PC gaming industry, they have become very powerful computing engines in their own right. The arrival in 2007 of the NVIDIA CUDA Toolkit for writing software that exploits the power of GPUs for scientific applications was a game changer. Suddenly we got a step up in computing power by a factor of 100 instead of the usual doubling every 18 months or so. Since then, the power of GPUs has also continued to grow exponentially over time, following and even exceeding Moore's law. Thus, knowing how to program GPUs is just as useful today as it was in 2007. In fact, today, if you want to engage with high-performance computing (HPC) perhaps on world-class supercomputers, knowing how to use GPUs is essential.

Up till about 2002 the exponential growth in PC computing power was largely due to increasing clock speeds. However, since then, clock speeds have plateaued at around 3.5 GHz, but the number of cores in a CPU chip has steadily increased. Thus, *parallel programming*, which uses many cooperating cores running simultaneously to share the computing load for a single task, is now essential to get the benefit from modern hardware. GPUs take parallel programming to the next level, allowing thousands or even millions of parallel threads to cooperate in a calculation.

Scientific research is difficult, and competitive, available computing power is often a limiting factor. Speeding up an important calculation by a factor of, say, 200 can be a game changer. A running time of a week is reduced to less than one hour, allowing for same-day analysis of results. A running time of one hour would be reduced to 18 seconds, allowing for exploration of the parameter space of complex models. A running time of seconds is reduced to milliseconds, allowing for interactive investigation of computer models. This book should be particularly useful to individual researchers and small groups who can equip their own in-house PCs with GPUs and get these benefits. Even groups with good access to large HCP facilities would benefit from very rapid tools on their own desktop machine to explore features of their results.

Of course, this book is also suitable for any reader interested in finding out more about GPUs and parallel programming. Even if you already know a little about the subject, we think you will find studying our coding style and choice of examples rewarding.

To be specific, this book is about programming NVIDIA GPUs in C++. I make no apology for concentrating on a specific vendor's products. Since 2007 NVIDIA have become a dominant force in HPC and, more recently, also AI. This is due to both the cost-effectiveness of their GPUs and, just as importantly, the elegance of the C++-like CUDA language. I know that some scientific programming is still carried out in various dialects of Fortran (including Fortran IV, a language I was very fond of in the early 1980s). But C++ is, in my opinion, more expressive. Fans of Fortran may point out that there is a technical problem with optimising C++ code using pointers, but that problem was overcome in C++11 with the introduction of the restrict keyword in C11. This keyword is also supported by modern C++ compilers, and it is used in many of our examples.

The examples are one feature that distinguishes this book from other current books on CUDA. Our examples have been carefully crafted from interesting real-world applications, including physics and medical imaging, rather than the rather basic (and frankly boring) problems often found elsewhere. Another difference between this book and others is that we have taken a lot of care over the appearance of our code, using modern C++ where appropriate, to reduce verbosity while retaining simplicity. I feel this is really important; in my experience most scientific PhD students learn computing by modifying other people's code, and, while much of the CUDA example code currently circulating works, it is far from elegant. This may be because in 2007 CUDA was launched as an extension to C, not C++, and most of the original SDK examples were written in a verbose C style. It is unfortunate that that style still persists in many of the online CUDA tutorials and books. The truth is that CUDA always supported some C++, and nowadays CUDA fully supports up to C++17 (albeit with a few restrictions). In November 2019 the venerable "NVIDIA C Programmers Guide" was renamed the "NVIDIA C++ Programmers Guide", and, although then there was no significant change to the content of the guide, it did signal a change in NVIDIA's attitude to their code, and since 2020 some more advanced uses of C++ have started to appear in the SDK examples.

This book does not aim to teach you C++ from scratch; some basic knowledge of C++ is assumed. However Appendix I discusses some of the C++ features used in our examples. Modern C++ is actually something of a monster, with many newer features to support object-orientated and other high-level programming styles. We do not use such features in this book, as, in our view, they are not appropriate for implementing the algorithmic code we run on GPUs. We also favour template functions over virtual functions.

To get the most out of our book, you will need access to a PC equipped with an NVIDIA GPU supporting CUDA (many of them do). The examples were developed using a Windows 10 PC with a 4-core Intel CPU and an NVIDIA RTX 2070 GPU (costing £480 in 2019). A Linux system is also fine, and all our examples should run without modification. Whatever system you have, you will need a current version of the (free) NVIDIA CUDA Toolkit. On Windows, you will also need Visual Studio C++ (the free community version is fine). On Linux, gcc or g++ is fine.

Sadly, we cannot recommend CUDA development on macOS, since Apple do not use NVIDIA cards on their hardware and their drivers do not support recent NVIDIA cards. In addition, NVIDIA have dropped support for macOS starting with their Toolkit version 11.0, released in May 2020.

All of the example code can be downloaded from https://github.com/RichardAns/CUDA-Programs. This site will also contain errata for the inevitable bugs that some of you may find in my code. By the way, I welcome reader feedback about bugs or any other comments. My email address is rea1@cam.ac.uk. The site will be maintained, and I also hope to add some additional examples from time to time.

I hope you enjoy reading my book as much as I have enjoyed writing it.