Learning CUDA: Lab Exercises and Experiences, Part 2

Christopher T. Mitchell Jens Mache
Lewis & Clark College
Portland, OR 97219, USA
{chrism, jmache}@lclark.edu

Karen L. Karavanic
Portland State University
Portland, OR 97207-0751, USA
karavan@cs.pdx.edu

Abstract

The rise of multi-core computer hardware has introduced new urgency to learning parallel programming. In this presentation, we again focus on CUDA exercises suitable for undergraduate students. Trying to appeal to a wide audience of today's learners, we have developed a "Game of Life" exercise and an introductory CUDA summary. We discuss our classroom-test of the exercise, our experiences, and our lessons learned.

Categories and Subject Descriptors D.1.3 [*Software*]: Programming Technique—Concurrent Programming; K.3.2 [*Computer and Information Science Education*]: Computer Science

General Terms Algorithms, Design, Human Factors, Languages, Measurement, Performance

Keywords parallel computing, GPGPU, CUDA, computer science education

1. Introduction

The rise of multi-core computer hardware has introduced new urgency to learning parallel programming, and CUDA is a well-known approach to general-purpose computing on graphics processing units (GPGPU). Although a typical CPU today has two, four or six cores, today's graphics cards can already have hundreds of cores.

Existing learning materials for parallel programming focus on scientific computing. Many of the applications presented in current textbooks include: matrix operations, numerical integration, finite element computations, and FFTs.

This focused set of applications is insufficient for two reasons: it is not representative of the broad range of applications that must soon be implemented in parallel, and the importance of these applications is only clear to students with a background in scientific computing.

Thus, there is a need for more widely-accessible examples that demonstrate the diverse applicability of parallel programming and motivate a broader group of students. To help accomplish this, we began developing a "Game of Life" exercise and a website that would help students quickly become familiar with important CUDA concepts.

2. Pieces of the Puzzle

While learning CUDA, we identified essential concepts necessary for understanding how CUDA applications are designed. We wanted to impress these concepts on students so that they would have a cohesive (though not necessarily detailed) mental model of how a CUDA application works. The hope is that from this model, students will be better able to internalize the details learned from other, more detailed, sources. We produced an informational web-page and a related lab exercise to teach and affirm these concepts [2].

The first concept involves understanding the differences between a CPU and a GPU — the differences in processor design encourage radically different ways of composing a program to solve a problem. Understanding the architecture of a GPU can help illuminate the rationale behind some of the ideas found in CUDA. The other pieces that we discuss in our webpage are:

- running a kernel on the GPU
- executing many copies of the kernel in configurable grids.
- the idea of being able to execute many distinct iterations of certain loops simultaneously by using `blockIdx` and `threadIdx`.
- allocating and copying to and from device memory

Each of these concepts are relevant to all but the most trivial CUDA programs, and having a grasp on them should make understanding CUDA source code and other materials much easier.

SPLASH'11 Companion, October 22–27, 2011, Portland, Oregon, USA.
ACM 978-1-4503-0940-0/11/10.

2.1 Game of Life Exercise

The undergraduate students who helped write part one of this series [3] worked through four labs associated with the book "Programming Massively Parallel Processors: A Hands-on Approach" [5]. The students reported that many of the labs rewarded student effort only with simple pass/ fail messages and that these textual messages were not very engaging or motivating. Students wished that their efforts could be met with a more rewarding outcome such as a visual. For a more engaging and visual lab, we explored the idea of using Conway's Game of Life.

We realized that instructing students to port a CPU-only version of the Game of Life to CUDA would serve well to reinforce the essential concepts that we identified. We settled on a prompt that offered no guidance and only explained that the student should get practice "putting all of the CUDA pieces together" by converting a serial Game of Life implementation into a CUDA one. This would require students synthesize what they have learned from other resources.

Our first attempt at developing the CPU-only code and the CUDA-enabled solution used the console for output. Because both versions were being bottlenecked by the slow console output, discerning a speed difference between the two was impossible. We took issue with this because without a perceptible speed difference, students would not get feedback demonstrating that their CUDA port had any benefit over the CPU-only version. To rectify this, we augmented the program with Xlib to draw the board in a window with one pixel per cell. With 1000×1000 sized boards, the speed increase that CUDA enables is easily observable, even on machines with fast CPUs.

3. Evaluation

We evaluated the Game of Life exercise with students enrolled in a 2011 summer Special Topics course at Portland State University titled "General Purpose GPU Computing" (GPGPUC). The course was open to senior undergraduates and graduate students, and covered both GPU hardware and CUDA. Because the exercise was added after the course was designed and already underway, we added it as an extra credit exercise, with the credit contingent upon the students' completion of a short survey. Student preparation prior to completing the exercise included readings from the CUDA C Programming Guide [1], the first three articles in the Dr. Dobbs series [4], and several recent research papers. They had completed two short programming assignments: the first one was designed to familiarize them with the CUDA environment and course machines; the second one required them to implement a simple matrix-vector multiplication in C, Pthreads, OpenMP, and CUDA. They were partway through more complex group programming projects using CUDA at the time they completed the extra credit exercise. Ultimately, eight students provided feedback.

Several respondents mentioned difficulty using tiling (described in Chapter 4 of [5]) to accommodate a Game of Life board with more cells than the maximal number of threads that can be in a single block. This was not an intended sticking point of the exercise and suggests that tiling (especially given it's general utility and ubiquity) should be introduced by our summary website.

We had anticipated that the exercise would take one or two hours to complete. The seven students who answered the question "Approximately, how many hours did you spend on the exercise?" answered with 1, 2, 3, 4, 4, 5, and 5 hours. A few students reported that the bulk of their time was spent on things related to tiling. Other students wished that they were able to use the CUDA debugger, but their environment did not readily support using it.

Students found the exercise to be interesting (4 vs. 0), worthwhile (6 vs. 0) and helpful for understanding course materials (5 vs. 1). Some felt that the Game of Life was a compelling problem for parallel computing (3 vs. 1). Four of the eight students thought that the exercise was slightly difficult while three found it to be slightly easy. Two students found our summary website to be sufficient for helping them understand the exercise, and three found our website insufficient; the rest felt neutral, indicating room for improvement.

Acknowledgments

We would like to thank David Bunde for discussions, and Julian Dale for his role in creating the exercise and website. This material is based upon work supported by the National Science Foundation under grant 1044932, by the John S. Rogers Science Research Program at Lewis & Clark College, by NSF award 1044973, and by a PSU Miller Foundation Sustainability Grant.

References

[1] NVIDIA CUDA C programming guide. `http://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf`.

[2] `http://www.lclark.edu/~jmache/CUDA/`.

[3] N. Anderson, J. Mache, and W. Watson. Learning CUDA: Lab exercises and experiences. SPLASH '10, pages 183–188, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0240-1. doi: http://doi.acm.org/10.1145/1869542.1869571.

[4] R. Farber. CUDA, supercomputing for the masses: Part 1. `http://www.drdobbs.com/high-performance-computing/207200659`, 2008.

[5] D. B. Kirk and W.-m. W. Hwu. *Programming massively parallel processors: a hands-on approach.* Morgan Kaufmann Publishers, Burlington, MA, 2010. ISBN 0123814723.