GPU computing tutorial

Garland Durham
Quantos Analytics, LLC

March 30, 2013

Abstract

This tutorial will provide an introduction to parallel programming using commodity video card (GPU) hardware. GPU hardware can enable speedups on the order of a factor of 100 relative to conventional serial computing for amenable applications, and is readily available at modest cost. The main focus will be on using the CUDA software development kit (SDK) with Nvidia hardware. The tutorial will include examples in C/C++, Matlab and Python. A large part of the tutorial will consist of hands-on exercises illustrating basic ideas central to GPU computing and a selection of advanced topics. The tutorial is intended for people with little or no experience in GPU programming.

Participants should bring laptops. Those with laptops equipped with Nvidia graphics cards may find it useful to install the Nvidia CUDA driver and SDK. See `https://developer.nvidia.com/cuda-downloads` for details. Matlab with the parallel computing toolkit might also be useful. However, I intend to provide access to remote servers with hardware and software already installed and configured. This will provide a uniform environment for all participants to work on the exercises we will undertake together.

Some experience with C programming would be helpful. A very rudimentary knowledge is sufficient (e.g., a few hours working through an online tutorial such as `http://www.cprogramming.com/tutorial/c-tutorial.html`, excluding the last several sections covering linked lists, recursion, etc).

Linux, OS X and Windows operating systems are all usable.

Outline

1. Introduction
   (a) Historical background; alternative hardware/software platforms (MPI; OpenCL).
   (b) Nvidia CUDA; description of hardware and software environment.
   (c) Possible applications.
2. Installation and configuration
   (a) Available GPU hardware.
   (b) Installing the driver and software development kit (SDK).
   (c) Compiling and running sample programs.
   (d) Setting up a remote GPU compute server; Amazon Elastic Compute Cloud (EC2).
   (e) Working with the remote server (SSH, Sftp, Rsync).
   (f) Version control (Git).
3. Simple examples
   (a) Introduction to GPU computing: C/C++; Thrust library; Accelereyes Arrayfire; Matlab; Python.
   (b) Compile and run a few simple examples in C; illustrate basic ideas of GPU computing.
4. CUDA programming in C
   (a) Working with GPU threads.
   (b) Optimizing memory accesses.
   (c) Compilation options; makefiles.
   (d) Random number generation.
   (e) Linear algebra (CUBLAS).
   (f) Reductions and prefix scans.
   (g) Using the Thrust and Arrayfire libraries.
   (h) Building a custom library (a simple CUDA array class; functor classses; generic operations using functors).
   (i) The CUDA profiler and debugger.
   (j) Thrust library
5. Accelereyes Arrayfire
   (a) Overview

   (b) Array class
   (c) Basic operations
   (d) Matrix operations
   (e) Reductions and prefix scans
   (f) Exercises

6. Matlab
   (a) Working with the gpuArray class.
   (b) Optimizing memory accesses.
   (c) Efficient use of Matlab built-in GPU functions.
   (d) Using custom kernels with Matlab.

7. Python
   (a) Quick overview of Python, SciPy and PyCuda.
   (b) Basic array operations in SciPy.
   (c) Using PyCuda.
   (d) Reductions and prefix scans.
   (e) Random number generation.
   (f) Using custom kernels with Python.

8. Advanced topics
   (a) Choose from a selection of more advanced exercises. Possible projects include Markov chain Monte Carlo, sequential posterior simulation, option pricing, simulation-based forecast construction, particle filtering.