An introductory course on General Purpose Computing on GPUs

Additional details about the exercises

Francesco Lettich

Insight Lab
Universidade Federal do Ceará, Fortaleza, Brasil

Introduction

- To complete the exercises you will **need**:
  - Recommended O/S: Linux (e.g., Ubuntu $>=$ 16.04).
  - CUDA 9.
  - A GCC version compatible with CUDA 9 (GCC v5.x or v6.x are ok).
  - A NVIDIA videocard.
  - CMake.
- I will also give you a zip containing the **source code** of a little application that you will have to **complete** by providing the **required solutions**.

Setup

- To download CUDA 9:
https://developer.nvidia.com/cuda-downloads?target_os=Linux

- When installing CUDA 9, please **follow the instructions** provided by NVIDIA, especially the modifications you have to apply to the ".bashrc" file; more info here (**Section 4.1.5**):
http://docs.nvidia.com/cuda/cuda-quick-start-guide/index.html

- To download **GCC 5.x or 6.x**, please use "*apt install*" with gcc-5/g++-5 or gcc-6/g++-6 (or something equivalent).

- To install CMake, type "*apt install cmake*".

Archive details

- Within the archive you will find the following files/folders:

  - **CmakeLists.txt** => file used by the tool “Cmake” to “prepare” the compilation (generates some files).

  - **main.cu** => The file containing the “main” (entry point) of the program. Notice the extension “**cu**”, which tells the nvcc compiler that main.cu contains C++ code augmented with CUDA keywords and syntax.

  - **kernels.h and kernels.cu** => respectively containing the **declarations** and the **definitions** of the kernels (functions) you have to provide as part of the **exercises**.

Compilation and execution

- Assuming that you have a computer **properly configured**, go in the folder where you unzipped the archive...then:

  - To "**prepare**" the **compilation**, type "*cmake CmakeLists.txt*".

  - Then, to **compile** the source code type "*make*".

  - Finally, to **execute** the binary go in the "/bin" subfolder and execute the "*main*" executable.

List of Exercises / 1

- **Exercise 1**: implement the GPU-based version of the **Hillis-Steele** algorithm, limitedly to the case of a **single thread-block.** This is represented by the function "*HillisAndSteele*" in kernels.cu.

- **Exercise 2**: implement the Blelloch's (down/up-sweep) algorithm, limitedly to the case of a **single thread-block**. This is represented by the function "*BlellochSingleBlock*" in kernels.cu.

- **Exercise 3**: Adapt the code written for **exercise 2** to implement the **inclusive prefix sum** (easy!). This is represented by the function "*BlellochSingleBlockInclusive*" in kernels.cu.

List of Exercises / 2

- **<u>Exercise 4</u>**: Extend the code of **exercise 2** to make use of **multiple thread-blocks**. This is represented by the function "*BlellochMultipleBlocks*" in kernels.cu.

- **<u>Exercise 5</u> (optional!)**: Modify the source code of the solution of exercise 2 to avoid **bank conflicts** in shared memory. This is represented by the function "*BlellochNoConflicts*" in kernels.cu.

More on the exercises

- To complete the exercises you will have to complete part of the *main function* in **main.cu** and provide the body of the functions in **kernels.cu**.

- Please, **read carefully** the **notes** I've **left** in the source code!

- Feel free to adapt the source code to your needs: the only requirement is that it should be **easy** to **verify** the **correctness** of your solutions!

- Once you complete the exercises, please **mail back** to me the zip with **your solutions** (for the e-mail, see the next slide).

Questions & Doubts

If you have relevant questions, doubts, problems, or you want to send me the solutions, please contact me at:

francesco.lettich@gmail.com