Take-Home Exam CUDA_SCAN2

Parallel Programming & Architectures

Consideration

- ✓ Your code is automatically graded using a script, and therefore, if your file/folder names are wrong you will receive a grade of **zero**. Please read and follow the instructions carefully. Common mistakes include
  - Different file or folder names
  - Different formatting of input or output
  - Not paying attention to case sensitiveness of C++ and Linux
- ✓ Go to the folder ~/the/cuda_scan2/ in your home directory on the server and put your codes in this directory and remove any compiled binaries and test cases.
- ✓ Make sure your code compiles and runs without any error **on the server**. Your grade will be **zero** if any compile or runtime error occurs on the server. **Any!**
- ✓ The provided test cases, examples and sample codes (if any) are only to better describe the question. They are **not** meant for debugging or grading. It is your responsibility to think of and generate larger and more complex test cases (if necessary) in order to make sure your software works correctly for all possible scenarios.
- ✓ Start early and don't leave everything to the last minute. Software debugging needs focus and normally takes time.
- ✓ Just leave your final programs on the server. **Don't** email anything!
- ✓ Your grade is divided into several parts. In all cases, if you miss **correctness** (i.e. your code doesn't satisfy desired functionality), you miss other parts (e.g. speed, coding style, etc.) too. This rule is applied separately for each section of a take-home exam. So for example, in cuda_mm, your code might not be correct for M >= x but still you will get your grade for lower M values.
- ✓ Talking to your friends and classmates about this take-home exam and sharing ideas are *OK*. Searching the Internet, books and other sources for any code is also *OK*. However, copying another code is **not OK** and will be automatically detected using a similarity check software. In such a case, grades of the copied parts are **multiplied by -0.5**. Your work must be 100% done only by yourself, and you **should not** share parts of your code with others or use parts of other's codes. Online resources and solutions from previous years are part of the database which is used by the similarity check software.

***scan2.cu* – Scan Algorithm – Distributing across multiple GPU kernels**

Grade: *50%* correctness, *50%* speed

Implement parallel scan algorithm (inclusive scan) on an input array "A" with N 32-bit "float" values using the Blelloch algorithm. N=2^M and 20 <= M <= **29**. The output array "C" also has N elements. Your program must fill array A with random float values -2.0 <= A[i] <= +2.0

Hint: Blelloch algorithm computes the exclusive scan not inclusive scan, thus you need to do some extra work.

Hint: You need to distribute your work across multiple GPU kernels because large M values such as 29 does not fit into the GPU.

Only modify scan2.cu file and do not modify other files, e.g., scan2_main.cu or scan2.h. We replace the other files with their original copies before grading your work.

Compile: *nvcc scan2.cu scan2_main.cu -o scan2*

Execute: *./scan2 M*