ME759
High Performance Computing for Engineering Applications
Blocked 2D Convolution

Date Assigned: October 14, 2013
Date Due: October 21, 2013
Cutoff Time: 11:59 PM

In this problem, you have to implement a blocked matrix convolution. You will work with a constant 5x5 so called “convolution kernel matrix” (the **M** matrix), but will have arbitrarily sized "images" (the **N** matrix). Matrix convolution is primarily used in image processing for tasks such as image enhancing, blurring, etc.

A standard image convolution formula for a 5x5 convolution kernel **M** with matrix **N** is

$$\mathbf{P}[i][j] = \sum_{m=0}^{4}\sum_{n=0}^{4}\mathbf{M}[m][n]\cdot\mathbf{N}[i+m-2][j+n-2]$$

where $0 \le i < \mathbf{N}.\text{height}$ and $0 \le j < \mathbf{N}.\text{width}$. Elements that are "outside" the matrix **N**, for this exercise, are treated as if they had value zero.

For this problem, edit the source files to complete the functionality of the matrix convolution on the device. A CPU implementation of the convolution algorithm will be used to generate a correct solution which will be compared with your kernel’s output. If it matches (within a certain tolerance), it will print out "**Test PASSED**" to the screen before exiting.

Report.
This problem will require a performance testing and analysis process. Included in the distribution folder is a subfolder called "test", which contains two test case input sets. Using these test cases or any others that you wish to create to support your findings, provide answers to the following questions, along with a short description of how you arrived at those answers.

A. Generate a **png** plot that shows how the CPU and GPU convolution implementations scale with the size of the input. For this question, consider images **N** of increasing size (width from 2^4 to 2^{12}, assume square images). For the GPU, report the inclusive times.

B. How much time is spent as an overhead cost of using the GPU for computation? Consider all code executed within your host function, with the exception of the kernel itself and the CPU computation of the convolution, as overhead. How does the overhead scale with the size of the input?

Finally, remember that kernel invocations are normally asynchronous, so if you want accurate timing of the kernel's running time, you need to insert a call to `cudaThreadSynchronize()` after the kernel invocation. Alternatively, you can use what in the lecture we called the “Lazy Man’s profiler” to get timing information.

Grading.
Your submission will be graded as follows:
i) Demo/knowledge: 25%
- Computation runs on Euler, producing correct outputs files.

ii) Functionality: 40%
- Correct handling of boundary conditions
- Uses shared and/or constant memory to cover global memory latency.

iii) Report: 35%