ECEC 413/622: Parallel Computer Architecture
Blocked 2D Convolution

Instructor: Prof. Naga Kandasamy
ECE Department, Drexel University

March 6, 2017

The Lab is due March 19, 2017, via BBLearn. You may work on the assignment in teams of up to two people.

(20 points) Edit the source files `2Dconvolution.cu` and `2Dconvolution_kernel.cu` to complete the functionality of the matrix convolution on the GPU. Matrix convolution is primarily used in image processing for tasks such as image enhancing, blurring, etc. A standard image convolution formula for a 5×5 convolution kernel A with matrix B is

$$C(i,j) = \sum_{m=0}^{4} \sum_{n=0}^{4} A[m][n] * B[i+m-2][j+n-2]$$

where $0 \leq i <$ `B.height` and $0 \leq j <$ `B.width`. Note that the elements at the boundaries, that is, elements that are "outside" the matrix B are treated as if they had the value zero.

This assignment assumes a constant 5×5 convolution kernel, but will have "images" or matrices of an arbitrary size. Your program accepts no arguments. It creates a randomized kernel and matrix. A CPU implementation of the convolution algorithm is used to generate a correct solution which is compared with your GPU program's output. If the solutions match within a certain tolerance, the program prints out "Test PASSED" to the screen before exiting.

Upload the files needed to compile and run your code as a single zip file on BBLearn. Provide a brief report describing the design of your kernel, using code or pseudocode to clarify the discussion, as well as the speedup achieved over the serial version for the following dimensions of the B matrix: 512×512, 1024×1024, and 2048×2048. Quantify the sensitivity of the kernel to thread-block size in terms of the execution time. Also, answer the following questions in your report.

- What is the measured floating-point computation rate for the CPU and GPU kernels on this application? How do they each scale with the size of the input matrix?
- 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, as overhead.

How does the overhead scale with the size of the input? You are free to use any timing library you like, as long as it has a reasonable accuracy. 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.

Your assignment will be graded on the following parameters:

- **(10 points)** Get your code to work using just GPU global memory, at the very least.
- **(10 points)** Efficient use of the GPU memory hierarchy including shared and constant memories. Correct handling of halo elements when using shared memory and the use of constant memory to store the convolution kernel.