Lab 3: Convolution (Due on 11/06/2020)

Objective

Implement a convolution routine using texture and shared memory in CUDA. Your code should be able handle arbitrary 2D input sizes (1D kernel and 2D image pixel values should be randomly generated between 0~15). You only need to use a 1D kernel (mask) for both row and column convolutions that can be performed separately.

Instructions

Make a new folder for your project. Copy and modify the code of "convolutionTexture" in CUDA sample code (3_Imaging folder) to include the following key functions:

1. allocate device memory
2. copy host memory to device
3. initialize thread block and kernel grid dimensions
4. invoke CUDA kernel
5. copy results from device to host
6. deallocate device memory
7. implement the 2D image convolution kernel using texture and shared memories
8. handle thread divergence when dealing with arbitrary 2D input and kernel sizes

Your final executable can be run using the following command:
./convolution2D -i <dimX> -j <dimY> -k <dimK>

Where:
<dimX>, <dimY> are the X- and Y-dimensions of the input 2D image. <dimK> is the size of the 1D kernel (mask) for convolution.

Questions

(1) Name 3 applications of convolution.

(2) How many floating-point operations are being performed in your convolution kernel (expressed in dimX, dimY and dimK)? Explain.

(3) How many global memory reads are being performed by your kernel (expressed in dimX, dimY and dimK)? Explain.

(4) How many global memory writes are being performed by your kernel (expressed in dimX, dimY and dimK)? Explain.

(5) What is the minimum, maximum, and average number of real operations that a thread will perform (expressed in dimX, dimY and dimK)? Real operations are those that directly contribute to the final output value.

(6) What is the measured floating-point computation rate for the CPU and GPU kernels in this application? How do they each scale with the size of the input?

(7) What will happens if the mask (convolution kernel) size is too large (e.g. 1024)? What do you end up spending most of your time doing? Does that put other constraints on the way you'd write your algorithm (think of the shared/constant memory size)?

Report Submission

Your report should be submitted to Canvas along with the source code folder (zipped) via email no later the required project due date. In the report, you have to

1. Answer all the above questions.
2. Include important implementation details for developing your CUDA program.
3. Demonstrate extensive experimental results, such as compute throughputs (GFLOPS), using different combinations of input sizes, thread block sizes

and kernel sizes. Input size may not be multiples of the thread block size. The final results should be checked by comparing to the ones on CPU.

4. Discuss your results in your report using the knowledge learnt from our classes.