CS/EE 217
Lab 2
Tiled Matrix Multiplication
Due Fri. Apr 27 at 11:59:59pm

1) Download the lab2-starter zip file from the class website. Unzip lab2-starter into your sdk projects directory or working linux directory.

2) Edit the source files kernel.cu and main.cu to complete the functionality of the matrix multiplication on the device. The two matrices could be any size, but we will not test your code with an output matrix size exceeding 64,000 elements.

3) There are three modes of operation for the application. Check main() for a description of the modes (repeated below). You will support each of these modes using a Tiled matrix multiplication implementation.

> a) No arguments: The application will create two randomly initialized matrices to multiply size (1000x1000). After the device multiplication is invoked, it will compute the correct solution matrix using the CPU, and compare that solution with the device-computed solution. If it matches (within a certain tolerance), if will print out "Test PASSED" to the screen before exiting.
>
> b) One argument: The application will use the random initialization to create the input matrices (size mxm, where m is the argument. Start your testing with small matrices.
>
> c) Three arguments m, k, and n: The application will initialize the two input matrices with random values. A matrix will be of size m x k while the B matrix will be of size k x n, producing a C matrix of size m x n
>
> Note that if you wish, you may add a mode to accept input matrices from files, or to dump input and output matrices to files to facilitate testing.

4) Answer the following questions:

> 1. In your kernel implementation, how many threads can be simultaneously executing? Assume a GPU which has 30 streaming multiprocessors.
>
> 2. Use nvcc –ptxas-options=”-v” to report the resource usage of your implementation. Note that the compilation will fail but you will still get a report of the relevant information. Experiment with the Nvidia visual profiler, which is part of the CUDA toolkit, and use it to further understand the resource usage. In particular, report your branch divergence behavior and whether your memory accesses are coalesced.
>
> 3. How many times is each element of the input matrices loaded during the execution of the kernel?

Grading:

Please upload your zipped directory (after cleaning up executables and any unnecessary files) to iLearn. Your submission will be graded on the following aspects.

Correctness and performance (30%)
1. Produces correct results
2. Shared memory is used correctly (tiling) to improve performance

Functionality (35%)
- Correct usage of CUDA library calls and C extensions. - Correct usage of thread id's in matrix computation.

Report (35%)
Answers to the questions above (10 points for a, and 15 points for b and 10 points for c).