Introduction to Parallel Computer Architecture CUDA Programming Assignment: Matrix-Vector Multiplication

Instructor: Prof. Naga Kandasamy
ECE Department
Drexel University

February 23, 2018

The assignment is due March 4, 2018 by 11:59 pm via BBLearn. You may work on the problems in teams of up to two people. The submitted code must be your own. Please do not copy code from other students/teams or from online sources. Violation of this policy will result in a score of zero for the entire assignment.

You will multiply a dense $n \times n$ matrix A with an $n \times 1$ vector x to yield the $n \times 1$ result vector y. The serial algorithm is shown below.

```
1: procedure VEC_MAT_MULT(A, x, y)
2: int i, j;
3: for i := 0 to n − 1 do
4:     y[i] := 0;
5:     for j := 0 to n − 1 do
6:         y[i] := y[i] + A[i, j] × x[j];
7:     end for
8: end for
```

Answer the following questions:

- **(10 points)** Edit the `vec_mat_mult_on_device_using_global_memory()` function and the corresponding "naive" kernel function to complete the functionality of the vector-matrix multiplication on the GPU using global memory.
- **(15 points)** Edit the `vec_mat_mult_on_device_using_shared_memory()` function and the corresponding optimized kernel function to complete the functionality of the vector-matrix multiplication on the GPU using shared memory.

The CUDA source files for this question are available on BBLearn. Your program should accept no arguments. The application will create a randomly initialized matrix and a vector to multiply. After the GPU-based multiplication kernel is invoked, it will then compute the correct solution using the CPU and compare that solution with the GPU-computed solutions. If the solutions match within a certain tolerance, the application will print out "Test PASSED" to the screen before exiting.

Upload all of the files needed to run your code as a single zip file via BBLearn. This question will be graded on the following parameters:

- Report the speedup achieved by the GPU kernels over the CPU implementation for the following matrix sizes: 512×512, 1024×1024, and 2048×2048.
- Include a brief report describing how you designed your kernels, using code or pseudocode to clarify the discussion, and the speedup obtained over the serial version for both GPU-based versions.
- The GTX 1080 GPU can achieve a peak processing rate of about 8800 GFLOPs. The memory bandwidth on the device is 320 GB/s. How many floating-point operations must be performed per load operation to achieve the peak processing rate? What is the performance of your kernels (both naive as well as the one that uses shared memory), in terms of GFLOPs?