Programming Assignment 4

Early Due Date (+5%): Fri., 10/20, 11:59pm
Final Due Date: Tue., 10/24, 11:59pm

Background

The cuda API provides an environment for the development of host/device paradigm parallel applications which can make use of thousands of synchronous cores. Although subject to inherent communications overhead applications which perform heavy computations and/or conform to an SIMD organization can greatly improve computational runtimes in a GPU environment.

Assignment

Create both serial and CUDA parallel programs based on the following matrix operation:

```
double A[4096][4096], C[4096][4096];

// insert code to initialize elements of A[ ][ ] to random values between 1.0 and 2.0

for (i = 0; i < 4096; i++)
    for (j = 0; j < 4096; j++)
        for (k = 0; k < 4096; k++)
            C[ i ][ j ] += A[ k ][ i ] * A[ k ][ j ];
```

Input Data Format

An input file is not needed for this program.

Testing and Submission

Execute both of your programs, serial and cuda parallel, on one full node (28ppn) on Owens. Measure the elapsed time of both instances with time(1). Compute the performance of both program versions in terms of GFlops/sec.

Submit a short report in .pdf format on carmen which includes:
a) your GFlops/sec. results for both program versions.
b) the CUDA compute structure (Grid, Block and Thread) you used and explain your results.

For your code submission:

- Create a directory "lab4". Within this directory, place:
  - source code files required to build your programs(.c, .cu, .h);
  - makefile
    - your program should build with the command “make” with no parameters.
    - name your executables “lab4_serial” and “lab4_cuda.”
- submit the entire directory "lab4."

Using CUDA at Ohio Supercomputer Center

The OSC Owens cluster is equipped with Tesla P100 GPUs. Some relevant stats for the P100:

```
Total amount of global memory:                 16 GB
Compute Capability:                            6.0
(56) Multiprocessors, (64) CUDA Cores/MP:     3584 CUDA Cores
GPU Clock rate:                                1.328 GHz
L2 Cache Size:                                 4.0 MB
Total amount of constant memory:               65536 bytes
Total amount of shared memory per block:       49152 bytes
Total number of registers available per block: 65536
Warp size:                                     32
Maximum number of threads per multiprocessor:  2048
Maximum number of threads per block:           1024
Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535
```

To use CUDA on the OSC cluster, you must allocate a node with an attached GPU. To interactively allocate such a node, use:

$ **qsub -I -l walltime=0:59:00 -l nodes=1:gpus=1**
(qsub -EYE -ell walltime ... -ell nodes ...).

To ensure the best resource availability for everyone, please only log-on to a GPU host node when you are ready to compile and run, then please exit when you are not actively testing.

To compile and test your programs you will need to load the CUDA environment:

$ **module load cuda**

in order to use the Nvidia compilers. For example:

$ **nvcc -O -o lab4_cuda jones_jeffrey_lab4p2.cu**

The "-O" flag sets the optimizer to the default level (3), while the "-o lab4_cuda" flag, specifies the name for the executable file, "lab4_cuda," which you can then execute by name:

$ ./**lab4_cuda**

Note that compilation (use the nvidia compiler, nvcc) can be performed on the login nodes, and does not require a node with a GPU. You will need to load the cuda module on the login node if you wish to do this. You will not be able to test your programs successfully on the login nodes, as they have no GPUs.

Nvidia CUDA drivers available free on-line

If your laptop/desktop has an Nvidia graphics card, you can download the CUDA drivers directly from Nvidia for your own local development and testing. Please see `https://developer.nvidia.com/cuda-downloads`. Nvidia's "CUDA Zone" also provides a wide array of tools and documentation: `https://developer.nvidia.com/cuda-zone`.