Programming Assignment 4

Early Due Date (+5%): Mon. 11/18, 11:59pm
Final Due Date: Fri. 11/22, 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

Based on your original serial application from lab 1, create a cuda parallel version of your simplified AMR program.

- You may use the same functional computation for updating DSVs as your original serial program.
- Alternatively or in addition, you may modify your code and datastructures to provide more efficient SIMD processing.
- Write a short report (1 - 3 pages) which provides:
  - a comparative summary of the run-time and Gflops/sec for both your serial and cuda parallel program versions.
  - a description of any changes you elected to make to your serial program to enhance GPU performance, along with a summary of the performance impact of those changes.

Input Data Format

Use data file testgrid_400_12206 from previous labs 1 through 3.

Testing and Submission

Follow the testing and submission guidelines for previous labs, using “lab4” for this assignment, submitting your code on OSC and your report on Carmen.

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 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 lab4p1 jones_jeffrey_lab4p1.cu**

The "-O" flag sets the optimizer to the default level (3), while the "-o lab4p1" flag, specifies the name for the the executable file, "lab4p1," which you can then execute by name:

$ **lab4p1**

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`.