Programming Assignment 4 - cuda

Due Date: Tues., 11/27 (early submission +10%) – Thurs. 11/29 (final submission date)

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.

The purpose of this assignment is both to compare GPU implementations of the producer - consumer problem to a serial version and also to contrast two key variant styles of GPU programming.

Assignment

Based on your original serial application from lab 1, create 2 cuda parallel versions of your producer-consumer application.

- Parallel producer / consumer version 1 should accept as input the same input data files as the previous labs 1 through 3, and should process transforms calls in an arbitrary order.
- Parallel producer - consumer version 2 should pre-process the input, reordering the data so that all like work occurs together. That is, calls to each transform should be performed in clusters. For example, all calls to transformA are performed, then all calls to transformB, followed by all calls to transformC and lastly all calls to transformD. (Any ordering may be used, as long as they are performed in clusters.)
- Write a short report (no more than 3 pages) which provides:
  - a comparative summary of run-time results for each of your cuda implementations of producer - consumer along with a reference comparison to your serial version.
  - your observations and analysis regarding your results.

Input Data Format

Use the same input data files as previous labs 1 through 3.

Testing and Submission

Follow the testing and submission guidelines for lab2, using directory "lab4."

- Create a directory "lab4". Within this directory, place:
  - all source code files (.c, .cu and .h files);
  - makefile
    - your program should build with the command "make" with no parameters.
    - name your executables "lab4_arbitrary" and "lab4_simd."
- submit your report in .pdf format via Carmen.

cuda on OSC

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
```

and then use the Nvidia compiler to build your base program. For example:

```
$ nvcc   -dc   lab4_arbitrary.cu
```

Then, run **nvcc** again to link your program with the transform library:

```
$ nvcc   lab4_arbitrary.o   /fs/project/PAS1421/transform_cuda.o
```

Note that compilation using 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`.