Programming Assignment 4

Early Due Date: submission ends Sun., 4/4, 11:59pm, Carmen Time Zone
Due Date: submission ends Thu., 4/8, 11:59pm, Carmen Time Zone
Final Due Date: submission ends Sun, 4/11, 11:59pm, Carmen Time Zone

Please read all instructions carefully. You will be graded based upon meeting all requirements stated in this assignment.

Background and Problem Statement

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, those applications which perform heavy computations and/or conform to an SIMD organization can greatly improve computational runtimes in a GPU environment.

In this assignment we will compare GPU implementations of the producer - consumer problem to your serial version and also to contrast two key variant styles of GPU programming.

Assignment and Program Requirements

Unless otherwise stated in this assignment, assume that all relevant requirements from previous labs apply to lab 4.

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

- version 1:
  - accept as input the same data files as the previous labs 1 through 3, processing transform calls in the order presented by the input (“arbitrary” order).
  - the implication here is that different concurrent threads will be performing different transforms.
- version 2
  - pre-process the input, reordering the data so that all like work occurs together. That is, all input lines specifying transformA are grouped together, followed by all references to transformB, followed by all references to transformC and lastly all references to transformD (“SIMD” order).
  - your re-ordering may be done into a new intermediate file, or may be done in memory.
  - the implication here is that you can now structure your kernel so that all concurrent threads are performing the same transform.
- both versions
  - other than (optional) debugging output to stderr, your main() function should perform no I/O.
  - use separate kernels for producer and consumer functions. Do not chain producer and consumer functions together within the same kernel.
  - return the results of your producer kernel calls to the host and, on the host, determine the max and min values of the transformed keys over the entire set of input data.
  - carefully organize your source files such that all components required to be in .cu files are in .cu files, and all components which are not required to be in .cu files are in .c files.
  - your program should excute without using any command-line parameters.

For this assignment, you are encouraged to use sound software engineering principles, including creating prototype serial versions of your program which are structured to include function calls representing what will become kernel calls. Implement and test those new serial versions for correctness prior to porting them to cuda.

Input Data Format

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

Testing and Instrumentation

- The nVidia cuda compiler, *nvcc*, is located in module “cuda/10.2.89” on owens. Please allow time to reference the OSC module documentation and the nVidia compiler documentation as needed to complete your assignment.
- Measure the overall run-time of your programs using the Unix time(1) utility to report elapsed wallclock time.
- Using time(2), measure the total run-time of all producer computations, including any necessary host-device data transfers.
- Using time(2), measure the total run-time of all consumer computations, including any necessary host-device data transfers.
- You may use file I/O to stderr (standard error) to report your instrumentation and other debugging information.
  - You may but need not remove these statements from your program.
  - Your program will be tested with stderr redirected to /dev/null.
  - If you are not familiar with using standard error, please educate yourself and come to office hours with any questions.
- As with previous labs, programs which do not follow stdio guidelines will receive a 0 for this assignment. If you don’t know, ask.

Program Output

Your programs should each generate the exact output as your lab 1.

Report Requirements

- 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.
- the max and min values for transformed keys.

Submission

Generally, follow the testing and submission guidelines for lab3, instead 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_<osc id>_<osu id>_arbitrary” and “lab4_<osc id>_<osu id>_simd,” according to the input ordering that they expect.
  - your report in .pdf format.

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:

$ **sinteractive -A PAS1906 -g 1**

This will request a single interactive node with one gpu for (the default time of) 1 hour.

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/ess/PAS1906/transform_nvcc.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`.