Prog. HPC: Homework 6

CS 4001/7001

April 16, 2015

DUE: Sunday, 2015 April 26 @ 23:23

Topic: Parallel Programming with CUDA

The goal of this homework is to:

- develop your understanding of CUDA;
- develop an understanding of the various tradeoffs that can be made in logically structuring CUDA into a grid;
- understand the design alternatives of a kernel for solving variations of the same problems.

Problem: Median Filter

A median filter was discussed in class as a nonlinear image filter.
Please see: `http://en.wikipedia.org/wiki/Median_filter` for more details or review.
Data set:

- An image is being provided in the most simple of image formats, portable gray map (PGM), at `/content/cs/hpc/lena.pgm`
- This file is a 512x512 pixel, single band image.

Implementation

This homework may be implemented using the Amazon Cloud or on any other hardware resources you have available to you with GPU computing capabilities. You will implement median filters for windows of 3x3, 7x7, 11x11, and 15x15. This homework must be done in CUDA.

You may either write your kernel to handle the edge/corner cases, or you can buffer the processing to *active pixles* that support a full NxN window. For example, if you are performing a 11x11 median filter, the window over each pixel has a *square radius* of 5, e.g., (11 >> 1). Therefore, there will be a border (buffer) of 5 pixels around the outer edge of the active pixel input data.

Partnerships This homework can be implemented as either a one-some or two-some. However, the each student must submit an individual report.

Portable Gray Maps (PGM or PPM) Portable gray maps (PGM) and portable pixel maps (PPM) are extremely simple image file formats with minimal header information. See: `http://en.wikipedia.org/wiki/Portable_graymap`.

You may find sample PGM reading and writing code within the CUDA SDK or online. Please complile this into a separate file, away from your primary host code and device kernel code. CUDA SDK code exists as a templatized header in the file `sdkHelper.h`; is included in the following two locations of the SDK or Toolkit examples:

- `C/common/inc/`
- `CUDALibraries/common/inc/`

Specifically the functions that are relevant to his homework are

- **sdkLoadPGM(const char* file, T** data, unsigned int *w, unsigned int *h)**, and
- **sdkSavePGM(const char* file, T *data, unsigned int w, unsigned int h)**

Program Flow Your program should run as follows:

1. The command line should take a) an integer filter size {3, 7, 11, or 15}; b) an input file name; and c) an output file name.
2. The input file should be loaded into a data buffer (the file is one byte per pixel).
3. A timer should mark this time
4. The file data should be pushed onto the device memory.
5. The logical structure for processing should be defined
6. The kernel should be launched
7. The output data should be copied from device to main memory
8. The timer should be stopped to capture the device copy-compute-copy time
9. A *golden standard* host code version of the answer should be produced and compared to the device generated output
10. The output should be written to file in PGM format.
11. The percent of correctness between the host's *golden standard* and the device created answer should be reported
12. The timing statistics should be reported.

Report

Your report should document the following:

- Group members
- Your logical shape of the kernel execution, e.g., Grid/Block dimensions.
- Your timing trends for the four different size median windows, and using at least two different Grid/Block logical configurations.
- Examples images of the *input*, and the resulting *3x3*, *7x7*, *11x11*, *15x15* output.
- Your analysis of the timing trends as the window size increases, and how the logical structure (Grid) affects the timing.
- Any other insights and analysis you can provide that demonstrates what have have learned about CUDA.

Deliverables

EACH GROUP WILL SUBMIT A SINGLE CODE ARCHIVE, EACH PERSON WILL SUBMIT THEIR INDIVIDUALLY WRITTEN REPORT.

1. A tar-ball of a directory containing all the source code and appropriate Makefiles or Autotools. This should include a top level *README* file that details the simple procedure to build and run the program(s), as well as the necessary analysis. The naming convention should be: **pawprint1_pawprint2_pawprint3_hw6.tgz**. Where the *pawprintN* is your actual pawprint id, for upto three members.

2. A *PDF* report with content as detailed above. The report should identify your *Name* and *Pawprint* at the top of the front page. The naming convention should be: **pawprint_hw6.pdf** Again, the *pawprint* is your actual pawprint id; and **each member is responsible for their own report**.

- Use **cs_submit** with assignment key HW6 for the CODE tarball
- Email REPORT to GrantScott.phd+HPC@gmail.com
  with SUBJECT: HPC HW6 Report