CS 33 Spring 2013
Lab 5: CUDA

Introduction

In this lab assignment, you will improve the performance of the edge-detection code from Lab 4 by translating parts of it to CUDA. The code has been modified in a few places to make it more straightforward to translate.

Grading

Your grade for this assignment will be proportional to the amount of speedup you achieve, compared to the original sequential code. For full credit, you must achieve a 20X speedup. To ensure fairness in measuring speedup, you will submit your code to be run on the Hoffman computing cluster. To get any credit, your code must produce the same output as the original code and have no memory leaks. A memory leak is defined as a region of memory that was allocated by your code but never freed.

Environment Setup

The CUDA compilation tools are installed on `lnxsrv` for you to test the correctness of your code.
Before you can use them, you need to set up a few shell environment variables:

1) Determine if your `lnxsrv` shell is **CSH** / **TCSH** or **Bash**.
Run the command `ps` and note whether **csh**, **tcsh**, or **bash** is printed.

2a) If you use **CSH** / **TCSH** then edit the file **`.cshrc`** in your home directory (notice the dot “.”).
If the file does not exist, then create it.
Add the following lines at the bottom of the file:

```
setenv PATH ${PATH}:/usr/local/cuda/bin
setenv MANPATH ${MANPATH}:/usr/local/cuda/man
setenv LD_LIBRARY_PATH /usr/local/cuda/lib64
```

2b) If you use **Bash** then edit the file **`.profile`** in your home directory (notice the dot “.”).
If the file does not exist, then create it.
Add the following lines at the bottom of the file:

```
export PATH=$PATH:/usr/local/cuda/bin
export MANPATH=$MANPATH:/usr/local/cuda/man
export LD_LIBRARY_PATH=/usr/local/cuda/lib64
```

3) Log out of `lnxsrv`, then log back in.
You can now use the CUDA compiler `nvcc` and the compile command `make cuda`.

Lab Files

The included file `edgedetect.c` is the reference implementation of the edge-detection code.
The included file `edgedetect.cu` is the same code. You should edit, test, and submit `edgedetect.cu`.

Compiling and Running

To compile the reference version:	`make seq`
To compile your CUDA version:	`make cuda`
To run with default options:	`make run`
To check that your output is correct:	`make check`
To check the memory trace for leaks:	`make checkmem`
To remove the executable and output files:	`make clean`

Emulation Mode on lnxsrv

The `lnxsrv` machines do not have Nvidia GPUs. When you run CUDA code, it will use “emulation mode”. The thread blocks will be executed one-at-a-time. Within each thread block, normal CPU threads will be used for parallel execution. This can be very slow (10-20sec) compared to the reference version or running on a real GPU when you submit.

When using emulation mode, limit the size of your thread blocks to around 16 threads to avoid a huge slowdown when running. You can use the following predefined macro to avoid having to change your block size before each time you submit:

```
#ifdef __DEVICE_EMULATION__
     // The code here will be compiled when emulation mode is enabled.
     // Set a small block size...
#else
     // Otherwise, the code here will be compiled.
     // Set an arbitrary block size...
#endif
```

Submission

To use the submission scripts, you must be logged in to `lnxsrv02`.

To submit a source file: `./submit edgedetect.cu`
A unique cookie will be printed to allow you to identify your submissions.

To check the status of your submission in progress: `./status`

To check the results of completed submissions: `./results`
You can also check the web scoreboard (see below).

To clear a submission that has not yet completed: `./clear`

To view your recent submission’s output log: `./output`

When you submit your code, it will be run normally and with `cuda-memcheck`, an Nvidia tool which reports memory errors such as out-of-bounds accesses, and other errors and possible issues with a CUDA program. The `output` command will print the output log from `cuda-memcheck`.

Scoreboard

A full scoreboard is available via the web, and is updated every 30 seconds:
`http://www.seas.ucla.edu/~vitanza/cs33s13/cudalab.html`

Hoffman Cluster Hardware

For those who may be in interested in tuning their CUDA code for the specific hardware: When you submit to the cluster, your code will be run on an Nvidia Tesla M2070 which has 5GB RAM and compute capability 2.0.