COMP 605: Introduction to Parallel Computing Homework 5: GPU/CUDA: Getting Started

Mary Thomas

Department of Computer Science
Computational Science Research Center (CSRC)
San Diego State University (SDSU)

Due: 04/25/16
Posted: 04/07/16
Updated: 04/07/16

Table of Contents

Homework 5: Getting Started with GPU/CUDA Computing

- Read Cuda Tutorial: Volume 1
  - CUDA Programming Model Overview
  - CUDA Programming The Basics
- Do the NVIDIA Exercises found on the Tutorial Web Page:
  - Tutorial: `https://developer.nvidia.com/cuda-training#1`
  - Exercises: `http://www.nvidia.com/content/cudazone/download/Exercises.tar`
  - Instructions: `http://www.nvidia.com/content/cudazone/download/Exercise_Instructions.pdf`
    or `http://www-rohan.sdsu.edu/faculty/mthomas/courses/docs/cuda/CUDA_Exercise_Instructions.pdf`

Notes

- Follow the instructions in the CUDA Exercise Instruction file: `http://www-rohan.sdsu.edu/faculty/mthomas/courses/docs/cuda/CUDAExercise_Instructions.pdf`
- Exercise tar file can be found on tuckoo in the dir */COMP605/cuda*
- There are three problem directories and 5 total problems (instructions say 6).
- Make sure you can see the nvcc compiler: */usr/local/cuda/bin/nvcc*
- Remember that not all exercises will compile - you will need to fix them
- Submit all jobs to the batch queue
- For the libraries, you may need to add the following line to your .bashrc file:

```
export LD_LIBRARY_PATH="/usr/local/cuda/lib:/usr/local/cuda/lib64:$LD_LIBRARY_PATH"
```

simple_hello.cu

Get this code working so you always something that works on the GPU when things get confusing.

```
/*
 * Copyright 1993-2010 NVIDIA Corporation.  All rights reserved.
 *
 * NVIDIA Corporation and its licensors retain all intellectual property and
 * proprietary rights in and to this software and related documentation.
 * Any use, reproduction, disclosure, or distribution of this software
 * and related documentation without an express license agreement from
 * NVIDIA Corporation is strictly prohibited.
 *
 * Please refer to the applicable NVIDIA end user license agreement (EULA)
 * associated with this source code for terms and conditions that govern
 * your use of this NVIDIA software.
 *
 */
#include <stdio.h>

__global__ void kernel( void ) {
}

int main( void ) {
    kernel<<<1,1>>>();
    printf( "Hello, GPU World!\n" );
    return 0;
}
```

Compiling on tuckoo.sdsu.edu

```
[mthomas] nvcc -o simple_hello simple_hello.cu
```

simple_hello batch script

```
#!/bin/sh
#PBS -l nodes=node9:ppn=1
#PBS -N simple_hello
#PBS -j oe
#PBS -r n
#PBS -q batch
cd $PBS_O_WORKDIR

./simple_hello
```

simple_hello output

```
[mthomas] !qsub
qsub simple_hello.bat
6807.tuckoo.sdsu.edu

[mthomas] qstat -a

tuckoo.sdsu.edu:
                                                                  Req'd  Req'd   Elap
Job ID               Username Queue    Jobname          SessID NDS   TSK Memory Time  S Time
-------------------- -------- -------- ---------------- ------ ----- --- ------ ----- - -----
6807.tuckoo.sdsu     mthomas  batch    simple_hello      25347     1   0    --    --  C 00:00

[mthomas]
[mthomas]
[mthomas] cat simple_hello.o6807
Hello, GPU World!
[[mthomas]
```

CUDA Tutorial Instructions

- Follow the instructions in the CUDA Exercise Instruction file: `http://www-rohan.sdsu.edu/faculty/mthomas/courses/docs/cuda/CUDA_Exercise_Instructions.pdf`
- Exercise tar file can be found on tuckoo in the dir */COMP605/cuda*
- CUDA tutorial contains skeletons and solutions for 6 hands-on CUDA exercises
- In each exercise (except for #5), you have to implement the missing portions of the code
  - Each problem is properly completed when you compile and run the program and get the output Correct!
- Solutions are included in the solution folder of each exercise
- Hint: most of the codes need debugging.
- When things don't make sense, run your "hello world" code.

HW5.P1: cudaMallocAndMemcpy (Tutorial P1)

- There are 5 parts or steps: do all of these.
- cudaMallocAndMemcpy.cu is an exercise in learning to copy between the host and the device.
- there are several "Bonus" steps in the code, do all of these.
- Provide data to support the fact that your code worked (e.g. create a simple math kernel and change the data in the memory).

Note: The comments above are just hints, see the CUDA tutorial and code for complete directions.

HW5.P2: myFirstKernel (Tutorial P2)

- There are 3 parts or steps: do all of these.
- learn to launch kernels and set thread dimensions.
- learn about blocks, grids, and how to assign threads.

Note: The comments above are just hints, see the CUDA tutorial and code for complete directions.

HW5.P3: reverseArray-single block (Tutorial P3)

- Given an input array $\{a_0, a_1, \ldots, a_{n-1}\}$ in pointer d_a, store the reversed array $\{a_{n-1}, a_{n-2}, \ldots, a_0\}$ in pointer d_b.
- You will implement the kernel, called
  $$\_\_global\_\_ \; void \; reverseArray\_singleblock()$$
- You will launch only one thread block, to reverse an array of size $N = numThreads = 256$ elements
- Each thread moves a single element to reversed position:
  - Read input from d_a pointer
  - Store output in reversed location in d_b pointer

Note: The comments above are just hints, see the CUDA tutorial and code for complete directions.

HW5.P4: reverseArray - multiblock (Tutorial P4)

- Given an input array $\{a_0, a_1, \ldots, a_{n-1}\}$ in pointer d_a, store the reversed array $\{a_{n-1}, a_{n-2}, \ldots, a_0\}$ in pointer d_b.
- You will implement the kernel, called
  $\_\_global\_\_\ voidreverseArray\_multiblock()$
- You will launch multiple 256-thread blocks in order to reverse an array of size N, with $N/256$ blocks
- Each thread moves a single element to reversed position
  - Read input from d_a pointer
  - Store output in reversed location in d_b pointer

Note: The comments above are just hints, see the CUDA tutorial and code for complete directions.

HW5.P5: reverseArray-profiling (Tutorial P5)

- The program, *cudaprof* is not installed on tuckoo.
- Instead we will insert timers around critical blocks and look for bottlenecks.
- See code in /COMP605/cuda/setNthdsFromCmdArg
- Time key blocks of the code and use that data to make improvements to your code*:
  - Run code, make table of critical timings, identify where most of the time is spent.
  - Modify code and note if there significant changes/improvements, if so, record the data
  - repeat
  - summarize
- Compare the performance of the three approaches.
- Include relevant plots and tables of data.

Note: The comments above are just hints, see the CUDA tutorial and code for complete directions.

Part 6: Optimizing Array Reversal

- Goal: Get rid of incoherent loads/stores and improve performance
- Use shared memory to reverse each block
- Profile the working code
- Include relevant plots and tables of data.

Note: The comments above are just hints, see the CUDA tutorial and code for complete directions.

What to Report/Turn in for all problems:

- Create the homework directory USER/hw/hw4 with correct access permissions.
- Short lab report with comments and output showing code state/progress as you do the different exercises.
- Images are not helpful, unless you use a terminal coloring scheme that produces contrast between the text and the background.
- Evidence you ran your jobs using the batch queue (short/small job), and examples of batch scripts
- Relevant snippets of the code as you edit/complete key steps/parts.
- Reference key sources of information in your report.