CS4532 Concurrent Programming

Take Home Lab 2

Due – July 08 before 11:55 PM

Learning Outcomes

In this lab we will learn how to develop parallel programs using the CUDA programming. We will also compare the performance of different implementations. At the end of the lab you will be able to:

- develop a simple program to solve a parallel program using GPU threads
- collect performance data and use them to compare the performance of different implementations used to solve a problem

Getting Started:

i. SSH to one of CSE GPU machines. IP addresses and login details are as follows.

Machine 1: IP address – 10.8.106.6
1. Username – cstudent
2. Password- cstudent@123

Machine 2: IP address- 10.8.106.7
3. Username- cstudent
4. Password- cstudent@123

Then create a folder (use mkdir) with your index number. All the code files and executables should be stored in this folder.

ii. Upload device_details.cu file (this file can be found in the extracted lab2 folder) to the folder you created in step ii.

iii. You can compile this code using the nvcc compiler which can we called by invoking the command nvcc.
nvcc device_details.cu -o device_details

iv. Use ./device_details to execute the program.

This program will output the details of the device.

1. Write a brief description about each of the property displayed from the device_details program. You may refer https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html for additional information.

2. Upload vecadd.cu file (this file can be found in the extracted lab2 folder) to the folder you created in the above first step.
   a. Complete the __global__ void vecAdd(int *A,int *B,int *C,int N) function to add A and B vectors and put the result in C vector.
   b. Measure the time performance for both CPU and GPU code to add the given vectors for n=100, 10000,1000000 and 10000000.

3. Mean filter is a simple method used in image processing to reduce noise in images. You are required to write a CUDA program to apply a $k \, x \, k$ kernel filter to a given image (Gray Scale image) and produce the result. CPU (host) program should read the image file and convert to a raw image matrix. Then pass the matrix to the GPU and GPU kernel should apply the mean filter

to the image and return the result image back to the CPU. You don't need to display the result image.

a. Write a C or C++ code (sequential code) for the same application. Compare the time for CPU implementation and GPU implementation for following image sizes and window sizes.
    i. 640x640 image size with 3x3 window size
    ii. 640x640 image size with 5x5 window size
    iii. 1280x1280 image size with 3x3 window size
    iv. 1280x1280 image size with 5x5 window size
b. What are your conclusions regarding:
    i. When is the speedup (i.e. time of CPU version / time of CUDA version) at its lowest? And why?
    ii. When is the speedup at its highest? And why?

Important:

Always run your programs several times and take the average to report the time. Also when you access the server please make sure no one else will access (you can use top or htop commands to see the usage).