EECE5640
High Performance Computing
Homework 5
*Submit your work on Canvas in a single zip file

1. (30) Let us revisit the histogramming problem assigned in Homework 4. Your input to this program will be integers in the range 1-10,000,000 this time (use a random number generator that generates the numbers on the host). Your host-based data set should contain N integers, where N can be varied.
   a. This time you will implement a histogramming kernel in CUDA and run on a GPU. You can choose how to compute each class of the data set on the GPU. Attempt to adjust the grid size to get the best performance as you vary N. Experiment with N = 2^{10}, 2^{15}, 2^{20} and 2^{25}. When the GPU finishes, print one element from each class in class ascending order on the host (do not include the printing time in the timing measurements, though do include the device-to-host communication in the timing measurement). Plot your results in a graph.
   b. Compare your GPU performance with running this same code on a CPU using OpenMP.

2. (30) The code below carries out a "nearest neighbor" or "stencil" computation. This class of algorithm appears frequently in image processing applications. The memory reference pattern for matrix **b** exhibits reuse in 3 dimensions. Your task is to develop a C/CUDA version of this code that initializes **b** on the host and then uses tiling on the GPU to exploit locality in shared memory across the 3 dimensions of **b**:

```
#define n 64
float a[n][n][n], b[n][n][n];
for (i=1; i<n-1; i++)
   for (j=1; j<n-1; j++)
      for (k=1; k<n-1; k++) {
      a[i][j][k]=0.8*(b[i-1][j][k]+b[i+1][j][k]+b[i][j-1][k]
      + b[i][j+1][k]+b[i][j][k-1]+b[i][j][k+1]);
     }
```

   a.) Evaluate the performance of computing a tiled versus non-tiled implementation in your GPU application. Explore what happens when you change the value of n. Consider the performance for at least two additional values for n.

   b.) Explore and report on other optimizations to accelerate your code on the GPU?

3. (20) Read the Pascal whitepaper provided, and then identify the key features that were introduced in the Pascal P100 architecture, comparing those features against the Ampere-based A100 architecture (make sure to identify the source for the information you obtained on the A100). Please do not just repeat what you read in the Pascal whitepaper, go into more detail on each of the features you identify.

4. (20) Consider the following CUDA kernel for computing vector addition. Develop and time code for vector addition using OpenACC on the Discovery cluster. You should Compare the performance of the OpenACC and CUDA implementations on 2 different GPUs. You will need to first run the following: *module add hpc_sdk* and *module load cuda/11.7* to run OpenACC. You may need to first do a *module unload cuda/your-version-of-cuda* first. To compile, you will need to use the following command: *pgcc -acc -Minfo=accel file.c -o fil*e

5. (25 points extra credit for everybody) Let's revisit computing the value of pi, but this time we will use a series. For instance, we provide you code for the Leibniz's series, developed by Jose Cintra. Implement this series on the GPU, allowing the user to enter the number of iterations. Make sure to compute find an efficient computation of this kernel that utilizes the parallelism provided on the GPU. Then modify this code to use single precision math. Show results for at least 10 different number of iterations of the series and discuss how precision plays a role in the rate of convergence.

* Written answers to the questions should be included in your homework 5 write-up in pdf format. You should include your C/C++ programs and the README file in the zip file submitted.