**COP6616 Parallel Computing**

**Instructor: Scott Piersall**

**Fall 2024**

**Assignment 3**

1. **Short Answer Questions (14 points).**

1). What is the difference between data spatial locality and temporal locality?

2). In the MPI Gather routine, does the root contribute data?

3). For the following MPI code, decide whether there is/are any potential problem(s) with it. If there are problems, please fix them.

if (rank =0){

MPI\_Barrier(…);

} else {

do something;

}

4). In CUDA programs, if we compare “structure of arrays” and “array of structures”, which one is offers better memory access performance? WHY?

5). Discuss the difference between the following GPU memory: global, local, shared, texture, and constant. Which one is the fastest?

6). If shared memory in a GPU is defined and used in the following way, under what scenario about the variable “s”, is there a bank conflict? Under what scenario is there no bank conflict? Why?

\_\_shared\_\_ float shared[16];

float foo = shared[baseIndex + s \* threadIdx.x];

7). If shared memory in GPU is defined and used in the following way, what kind of bank conflict (such as 2-way, 4-way, 8-way) does it have? Why?

\_\_shared\_\_ **char** shared[];

foo = shared[baseIndex + threadIdx.x];

1. **(15 points)** Please write **a CUDA program** to compute the Euclidean distance, similar to the problem in assignment 2.

You may design your code using the following steps:

* Declare the arrays (host and device). All arrays should be dynamically allocated; the host arrays can be allocated either with malloc or new, while the device arrays should be allocated with cudaMalloc.
* Print the number of CUDA-enabled hardware devices attached to the system by calling cudaGetDeviceCount.
* Print at least 3 interesting properties of Device 0, including the device name, by calling cudaGetDeviceProperties. The first argument to this function is a pointer to a struct of type cudaDeviceProp.
* Calls InitArray to initialize the host arrays. InitArray initializes an integer array with random numbers within a fairly small range (0 to 99).
* Calls cudaMemcpy to copy the host input arrays to the device.
* Calls the CUDA kernel, which computes the square of the difference of the components for each dimension, reduce all the elements of the output array in parallel (you may need to investigate how to implement an efficient parallel reduce in CUDA), and takes the square root of the sum.

Run experiments using varying size of inputs. Graph and discuss the speedup provided by the GPU/CUDA implementation over varying input sizes. Is there an input size where the speedup stops?

1. **(10 points)** You are working on team that is building an edge detection module. The first step in edge detection is to remove color information and work directly in black-and-white. You do not need to do edge detection, just the first step: removing the color information from images. So, please write **a CUDA program** to convert a color image to grayscale using the Colorimetric method. I suggest that you use PNG or BMP images as input. I want to see the original images that you tested with and the output/resulting image. You may do this assignment in C or Python, and are free to create a Jupyter notebook. If you do a Jupyter notebook, please include the notebook in your assignment report. How does using a GPU for this vs. a serial CPU-based implementation perform? Is there a speedup? How (be specific) is the GPU architecture well-suited to this task?
2. **(11 points)** Write a matrix multiplication program that uses GPU/CUDA. Your matrices should consist of floating-point values (NOT INTEGERS). You may do this assignment in C, Python, and are free to create a Jupyter notebook. If you do a Jupyter notebook, please include the notebook in your assignment report. You should run MANY experiments with varying sized matrices. What is the speedup? Is the speedup affected by matrix size? Explain (diagrams are useful here) how the GPU implementation of this code differs from a serial (CPU-based) implementation.

**What to submit:**

1. Answer all questions and provide any code inline. Do not paste screenshots of code, post your source code, I want to see graphs/visualizations of all of your experimental results results. Crete a PDF and upload the PDF to the assignment specification in canvas.