150 points. Individual Work Only. Due Oct 25, 2022 before 12:30 PM.

Objectives:

To implement and evaluate the performance of Jacobi and Gauss-Seidel methods to solve a system of linear equations on a GPU.

Problem Statement:

1. Implement a naïve version of the Jacobi algorithm using CUDA and measure the performance on a single GPU on the DMC machine.
2. Implement an optimized version of the Jacobi algorithm using CUDA and measure the performance on a single GPU on the DMC machine.
3. Implement a modified version of the Jacobi algorithm that uses recently computed values of x within an SM (i.e., uses Gauss-Seidel method within an SM) and uses previously computed values when using values of x not available within an SM.
4. Complete the table below with all the performance measurement results and include this in your report:

Algorithm Versions	# of iterations to converge	Execution Time
CPU Version		
Simple GPU Version		
Optimized GPU Version		
Modified GPU Version		

5. Analyze the performance differences between the different versions of the algorithm and compare the performance of these algorithms and include this analysis in the report.

Guidelines and Hints:

1. Review the related articles and presentations provided in Blackboard.
2. Use the CPU version of the program provided in Blackboard to get started. Feel free to modify this program to use CUDA BLAS, especially to compute the norm of the vector. You can also overwrite the x vector with the new values computed instead of swapping the vector. Whatever changes you make, make sure you compare the performance with similar versions of the program.
3. Use the CUDA Events API to measure the time taken on the GPU (review the article on Performance Metrics in CUDA or the matrix transpose example to understand how to use CUDA Events API).
4. Execute your program on the cluster dmc.asc.edu on a single CPU and a single GPU and note down the time taken. Use the matrix size of 1024x1024 and maximum iterations of 10,000 with a tolerance value of 10^{-6} for all versions of the program.
5. Make sure you submit jobs to the compute nodes and DO NOT run any jobs on the login node/head node on the ASA cluster. Also make sure you submit your jobs ONLY to the “class” queue. Use the *run_gpu* script to submit GPU jobs.

6. Comment out any print statements you might have to print the matrices when you execute with larger problem size. Also execute the program at least three times and use the average time taken.
7. Check-in the final version of your program and the job execution output files (.o<jobid> files) to the *github* server and make sure to share your *git* repository with the Instructor.

Program Documentation:

1. Use appropriate function name and variables names.
2. Include meaningful comments to indicate various operations performed by the program and instructions to compile and run the program.
3. Programs must include the following header information within comments:

```
/*
  Name:
  Email:
  Course: CS 691
  Homework #:
  Instructions to compile the program: (for example: nvcc -o hw2 hw2.cpp)
  Instructions to run the program: (for example: ./hw2 10000)
*/
```

Submission:

Upload the source files and report (.doc or .docx or .pdf file) to Blackboard in the assignment submission section for this homework. Include the github URL in the comment section of the submission.

Grading Rubrics:

The following grading policy will be used for grading programming assignments:

Program Design and Implementation	60% (program with no compiler errors or logical errors with the required functionality)
Program Testing	15% (includes selecting appropriate test cases, performing the tests, comparing CPU and GPU results)
Report including Performance Analysis	20% (performance evaluation and analysis)
Source Code Formatting	5% (indentation, variable names, comments, etc.)