Lab 3 Report: Cameron Young, Matthew Kellermann
cky39v, mtkmc6

Work Distribution:

Code:

CUDA Code- Cameron Young
File I/O- Cameron Young
Matrix Addition Algorithm- Cameron Young
Matrix Input Format- Cameron Young
README- Cameron Young
Error Handling- Cameron Young
Bugfixing- Cameron Young

Question Responses:

Cameron Young

Lab Report:

Cameron Young

Milestones:

CUDA Code:

This program was written using the CUDA api and hardware to allow for the addition of two m x n matrices. These matrices must have the same number of rows and columns. This resulted in a fairly simple program that utilized cuda efficiently.

Matrix Input Format:

The program takes in two matrices stored in a text file. The matrices are preceded by their row and columns then the matrices are stored as they would appear if drawn out on a piece of paper. Separated by spaces for each row.

Matrix Addition Algorithm:

The algorithm allowed for the computational sum of two matrices using CUDA. This was achieved by writing a cuda kernel and implementing all of the host code in C.

Implementation Decisions:

Using Cuda:

The decision to use CUDA for this lab was a requirement of the lab and therefore was not necessary to make a decision on what library to use for GPU compute.

Question Responses:

1. If cuda waited for threads that were in other blocks to complete in separate blocks this would create much longer delays in the program and slow down the execution of large operations. This defeats one of the core purposes for cuda which is to accelerate this kind of workload.
2. CUDA allows for execution of large chunks of data at once. Operating sequentially is rather frowned upon when operating on data like that. Instead the program could perform the entire reduction at the same time and result in a much faster execution.