EE/CSCI 451
Fall 2023
Programming Homework 5

Assigned: October 25, 2023
Due: November 6, 2023
Total Points: 100

General Instructions

- You may discuss the algorithms. However, the programs have to be written individually.
- Submit **the source code, makefile and a simple report** via **Blackboard**. The report should include the screenshots on the output from HPC. The source codes should be named 'p1.cu' and 'p2_1.cu'. Put all the files in a zip file with file name `<firstname>_<uscid>_phw<programming homework number>.zip` (do not create an additional folder inside the zip file). For example, `alice_123456_phw1.zip` should contain only the source codes and the report.
- Your program should be written in C or C++. You can use any operating systems and compilers to develope your program. However, we will test your program on a x86 linux with the latest version of g++ and gcc. Make sure you makefile could compile the executables for **all** the problems (hint: set multiple targets for the makefile) and the name of the executables are correct. If your program has error when we compile or run your program, you will lose at least 50% of credits.

1 Examples

"vector_add.cu" implements the vector addition using 64K threads. You can either run it on CARC or your local machine with Nvidia GPU. To run it on CARC, first load the CUDA module using "module load nvidia-hpc-sdk", and compile it using the nvcc compiler "nvcc vector_add -o vector_add". Then, submit the job to a node with GPU using the slurm file we provided. If you want to run GPU program in interactive mode, you need to reserve a node with GPU, please refer to the details here: https://www.carc.usc.edu/user-information/user-guides/software-and-programming/using-gpus

2 Matrix Multiplication [100 points]

In the lecture and discussion, we discussed two approaches to compute matrix multiplication ($C = A \times B$) using CUDA: (1) unoptimized implementation using global memory only and (2) block matrix multiplication using shared memory.

In this assignment, your task is implementing 1024×1024 matrix multiplication using these two approaches.

- Approach 1 (unoptimized implementation using global memory only):
  - Name this program as 'p1.cu'
  - The value of each element of A is 1
  - The value of each element of B is 2
  - Thread block configuration: 16×16
  - Grid configuration: 64×64
  - After computation, print the value of $C[451][451]$
- Approach 2 (block matrix multiplication using shared memory):
  - Name this program as 'p2.cu'
  - The value of each element of A is 1
  - The value of each element of B is 2
  - Thread block configuration: 32×32
  - Grid configuration: 32×32
  - More details of this algorithm can be found in the paper 'Matrix Multiplication with CUDA' on Piazza > Resources > General Resources.
  - After computation, print the value of $C[451][451]$
- Report: measure the execution time of the kernel of Approach 1 and Approach 2, respectively. Briefly discuss your observations.