NSM GPU Programming Assignment 1 Deadline : December 11, 2023, 23:55

1 Problem Statement

Write three separate CUDA C++ kernels for performing computations on two input matrices (**A** and **B**) and generating the output matrix **C**. In the first kernel **per_row_column_kernel**, each thread should process a complete row of the input matrices. In the second kernel **per_column_row_kernel**,each thread should process a complete column of the input matrices . In the third kernel **per_element_kernel**, each thread should process exactly one element from both the input matrices. For the evaluation purpose, per_row_column_kernel will be invoked with ***1D grid*** and ***1D blocks***, per_column_row_kernel will be invoked with ***1D grid and 2D blocks*** and per_element_kernel will be invoked with ***2D grid and 2D blocks***.

2 Input and Output

2.1 Input

* Matrix A of size m x n
* Matrix B of size m x n

2.2 Output

* Output is Matrix C of size n x m
* Output is computed as : C = (A + B)^T - (B - A)^T, where X^T is the transpose of matrix X.

2.3 Constraints

* $2 \leq m \leq 2^{13}$, $2 \leq n \leq 2^{13}$

3 Sample TestCase

* Input Matrix A

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

* Input Matrix B

$$\begin{bmatrix} 1 & 3 & 2 \\ 4 & 7 & 2 \end{bmatrix}$$

* (A + B)^T

$$\begin{bmatrix} 2 & 8 \\ 5 & 12 \\ 5 & 8 \end{bmatrix}$$

* (B - A)^T

$$\begin{bmatrix} 0 & 0 \\ 1 & 2 \\ -1 & -4 \end{bmatrix}$$

* C = (A + B)^T - (B - A)^T

$$C = \begin{bmatrix} 2 & 8 \\ 5 & 12 \\ 5 & 8 \end{bmatrix} - \begin{bmatrix} 0 & 0 \\ 1 & 2 \\ -1 & -4 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 8 \\ 4 & 10 \\ 6 & 12 \end{bmatrix}$$

* Output Matrix C

$$\begin{bmatrix} 2 & 8 \\ 4 & 10 \\ 6 & 12 \end{bmatrix}$$

4 Points to be noted

* The number of threads launched for each of the three kernels will be more than or equal to the number of threads required to do the computation.
* Do not write any print statements inside the kernel.
* Test your code on large inputs.

5 Submission Guidelines

* Submit your file with your **full_name.cu** which contains the implementation of the above-described functionality
* For transpose don't use tiling method.
* After submission, download the file and make sure it was the one you intended to submit.
* Kindly adhere strictly to the above guidelines.

6 Learning Suggestions

Write a CPU-version of code achieving the same functionality. Time the CPU code and GPU code separately for large matrices and compare the performances.