CS6023: GPU Programming

Assignment 1 (10 marks)

Deadline : February 13, 2022, 23:55 on Moodle

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 matrix A and corresponding complete column of matrix B. In the second kernel ***per_column_row_kernel***, each thread should process a complete column of matrix A and corresponding complete row of matrix A. 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 n x m

2.2 Output

- Output is Matrix **C** of size m x n
- Output is computed as : $\mathbf{C} = (\mathbf{A} + \mathbf{B^T}) \cdot (\mathbf{B^T} - \mathbf{A})$, where X^T is the transpose of matrix X and $X \cdot Y$ is the dot product of the matrices X and Y.

2.3 Constraints

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

3 Sample Testcase

- **Input Matrix A:**

$$\begin{bmatrix} 6 & 18 & -9 & 9 \\ -5 & 13 & 16 & -8 \\ -6 & 3 & -7 & -10 \end{bmatrix}$$

- **Input Matrix B:**

$$\begin{bmatrix} -3 & 8 & 10 \\ -3 & -9 & 13 \\ 13 & -2 & 10 \\ 14 & 16 & 19 \end{bmatrix}$$

- $(\mathbf{A} + \mathbf{B^T})$

$$\begin{bmatrix} 6 & 18 & -9 & 9 \\ -5 & 13 & 16 & -8 \\ -6 & 3 & -7 & -10 \end{bmatrix} + \begin{bmatrix} -3 & -3 & 13 & 14 \\ 8 & -9 & -2 & 16 \\ 10 & 13 & 10 & 19 \end{bmatrix} = \begin{bmatrix} 3 & 15 & 4 & 23 \\ 3 & 4 & 14 & 8 \\ 4 & 16 & 3 & 9 \end{bmatrix}$$

- $(\mathbf{B^T} - \mathbf{A})$

$$\begin{bmatrix} -3 & -3 & 13 & 14 \\ 8 & -9 & -2 & 16 \\ 10 & 13 & 10 & 19 \end{bmatrix} - \begin{bmatrix} 6 & 18 & -9 & 9 \\ -5 & 13 & 16 & -8 \\ -6 & 3 & -7 & -10 \end{bmatrix} = \begin{bmatrix} -9 & -21 & 22 & 5 \\ 13 & -22 & -18 & 24 \\ 16 & 10 & 17 & 29 \end{bmatrix}$$

- $\mathbf{C} = (\mathbf{A} + \mathbf{B^T}) \cdot (\mathbf{B^T} - \mathbf{A})$

$$C = \begin{bmatrix} 3 & 15 & 4 & 23 \\ 3 & 4 & 14 & 8 \\ 4 & 16 & 3 & 9 \end{bmatrix} \cdot \begin{bmatrix} -9 & -21 & 22 & 5 \\ 13 & -22 & -18 & 24 \\ 16 & 10 & 17 & 29 \end{bmatrix}$$

$$= \begin{bmatrix} -27 & -315 & 88 & 115 \\ 39 & -88 & 252 & 192 \\ 64 & 160 & 51 & 261 \end{bmatrix}$$

- **Output Matrix C:**

$$\begin{bmatrix} -27 & -315 & 88 & 115 \\ 39 & -88 & -252 & 192 \\ 64 & 160 & 51 & 261 \end{bmatrix}$$

4 Points to be noted

- The file ’main.cu’ provided by us contains the code, which takes care of file reading, writing etc. The prototypes of the three kernels are also provided in the same file. You need to implement the three kernels provided in the code.
- 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 input graphs.

5 Submission Guidelines

- Use the file ’main.cu’ provided by us.
- Compress the file ’main.cu’, which contains the implementation of the above-described functionality to ROLL_NUMBER.zip
- Submit **only** the ROLL_NUMBER.zip file on Moodle.
- 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.