CS6023: GPU Programming

Assignment 2 (15 marks)
Deadline : March 6, 2022, 23:55 on Moodle

1. Problem Statement

Given four input matrices A, B, C, and D.
Compute the output matrix, $X = (A + B^T)\, C\, D^T$

Write an efficient code to compute the output matrix. While writing the code, consider aspects like memory coalescing, shared memory, degree of divergence, etc.

2. Input and Output

2.1. Input

- 4 integers: p, q, r and s
- Matrix A of size $p \times q$
- Matrix B of size $q \times p$
- Matrix C of size $q \times r$
- Matrix D of size $s \times r$

2.2. Output

- Matrix X of size $p \times s$

2.3. Constraints

- $2 \le p, q, r, s \le 2^{10}$
- All the elements in the input matrices will be in the range [-10, 10]

3. Sample Testcase

- Input matrices $A,\ B,\ C$ and D:

$$A = \begin{bmatrix} 2 & 5 & 0 \\ 3 & -2 & 1 \end{bmatrix}, B = \begin{bmatrix} 6 & 1 \\ -4 & 2 \\ 1 & 3 \end{bmatrix}, C = \begin{bmatrix} 1 & 9 & 6 \\ -6 & 7 & 2 \\ 2 & 4 & -3 \end{bmatrix}, D = \begin{bmatrix} 10 & 0 & 5 \\ 1 & 3 & -3 \end{bmatrix}$$

Input will be given as:

```
2 3 3 2
2 5 0
3 -2 1
6 1
-4 2
1 3
1 9 6
-6 7 2
2 4 -3
10 0 5
1 3 -3
```

First line represents the values $p,\ q,\ r$ and s

Next p lines represents the rows of matrix A

Next q lines represents the rows of matrix B

Next q lines represents the rows of matrix C

Next s lines represents the rows of matrix D

- $(A\ +B^T)$

$$\begin{bmatrix} 2 & 5 & 0 \\ 3 & -2 & 1 \end{bmatrix} + \begin{bmatrix} 6 & -4 & 1 \\ 1 & 2 & 3 \end{bmatrix} = \begin{bmatrix} 8 & 1 & 1 \\ 4 & 0 & 4 \end{bmatrix}$$

- Output matrix, $X\ =\ (A\ +B^T)\ C\ D^T$

$$X = \begin{bmatrix} 8 & 1 & 1 \\ 4 & 0 & 4 \end{bmatrix} \begin{bmatrix} 1 & 9 & 6 \\ -6 & 7 & 2 \\ 2 & 4 & -3 \end{bmatrix} \begin{bmatrix} 10 & 1 \\ 0 & 3 \\ 5 & -3 \end{bmatrix}$$

$$= \begin{bmatrix} 4 & 83 & 47 \\ 12 & 52 & 12 \end{bmatrix} \begin{bmatrix} 10 & 1 \\ 0 & 3 \\ 5 & -3 \end{bmatrix}$$

$$= \begin{bmatrix} 275 & 112 \\ 180 & 132 \end{bmatrix}$$

4. Points to be noted

- The file ‘`main.cu`’ provided by us contains the code, which takes care of taking the input, printing the result and printing the execution time.
- **Don’t write any code in the `main()` function.**

- You need to implement the `compute()` function provided in the ‘`main.cu`’.
- You are free to use any number of functions/kernels.
- You can launch the kernels as you wish.
- **It is compulsory to optimize for coalesced accesses. Also, make use of shared memory.**
- Do not write any print statements.
- Test your code on large input matrices.

5. Submission Guidelines

- Use the file ‘`main.cu`’ provided by us.
- Don’t change anything in the `main()` function.
- Rename the file ‘`main.cu`’, which contains the implementation of the above-described functionality, to <ROLL_NO>.cu
- For example, if your roll number is CS20M039, then the name of the file you submit on the Moodle should be CS20M039.cu (submit only the <ROLL_NO>.cu file).
- After submission, download the file and make sure it was the one you intended to submit.

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.
- Exploit shared memory as much as possible to gain performance benefits.
- Try reducing thread divergence as much as possible.