NSM GPU Programming Assignment 2 Deadline : December 12, 2023, 23:55

1 Problem Statement

Using the concept of shared memory find the computation -

$\mathbf{X} = (\mathbf{A}^T + \mathbf{B}^T)\ \mathbf{C}^T\ \mathbf{D}$

Note that for multiplication and transpose use the shared memory concept strictly.

Also use memory coalescing, degree of divergence etc.

2 Input and Output

2.1 Input

* Four integers p, q, r, s
* Matrix A of size p x q
* Matrix B of size p x q
* Matrix C of size r x p
* Matrix D of size r x s

2.2 Output

* Output is Matrix X of size q x s

2.3 Constraints

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

3 Sample TestCase

* First line represents the values p , q, r, s
  Next p lines represents the rows of matrix A
  Next p lines represents the rows of matrix B
  Next r lines represents the rows of matrix C
  Next r lines represents the rows of matrix D
  2 3 3 3
  1 2 3
  4 5 6
  1 3 2
  4 7 2
  2 3
  3 3

4 5
2 3 5
6 7 8
2 1 3

* Input Matrix A

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

* Input Matrix B

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

* Input Matrix C

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

* Input Matrix D

$$\begin{bmatrix} 2 & 3 & 5 \\ 6 & 7 & 8 \\ 2 & 1 & 3 \end{bmatrix}$$

* $(A^T + B^T)$

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

* $(A^T + B^T)\ C^T$

$$\begin{bmatrix} 28 & 30 & 48 \\ 46 & 51 & 80 \\ 34 & 39 & 60 \end{bmatrix}$$

* $X = (A^T + B^T)\ C^T\ D$

$$X = \begin{bmatrix} 28 & 30 & 48 \\ 46 & 51 & 80 \\ 34 & 39 & 60 \end{bmatrix} \begin{bmatrix} 2 & 3 & 5 \\ 6 & 7 & 8 \\ 2 & 1 & 3 \end{bmatrix}$$

$$= \begin{bmatrix} 332 & 342 & 524 \\ 558 & 575 & 878 \\ 422 & 435 & 662 \end{bmatrix}$$

* Output Matrix X

$$\begin{bmatrix} 332 & 342 & 524 \\ 558 & 575 & 878 \\ 422 & 435 & 662 \end{bmatrix}$$

4 Points to be noted

* You are free to use any number of functions/kernels.
* Do not write any print statements inside the kernel.
* Using multiplication and transpose without shared memory would give 0 marks.
* 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
* Use shared memory concept as much as possible.
* 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.
* Exploit shared memory as much as possible to gain performance benefits
* Try reducing thread divergence as much as possible