

CS6023 (GPU Programming): A2 (7 marks)

Deadline: 01-Mar-2020 23:55

Submission Link: <https://courses.iitm.ac.in/mod/assign/view.php?id=39725>

0. Aim

This assignment helps to understand/learn two concepts: Synchronization and Memory coalescing.

1. Problem Specification

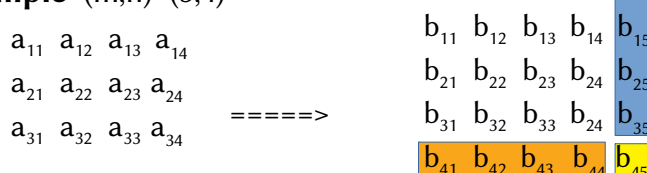
Input: Matrix A of size $m \times n$

Output: Matrix B of size $(m+1) \times (n+1)$

Given a matrix $A_{m \times n}$. Construct a matrix $B_{p \times q}$ where $p = m+1$ and $q = n+1$ using the following:

- The last entry of i^{th} row of B should hold the $\text{rowSum}(i) = \text{sum of all } i^{\text{th}} \text{ row entries of A}$, $1 \leq i \leq m$.
- Similarly, the last entry of j^{th} column should hold $\text{colSum}(j) = \text{sum of all } j^{\text{th}} \text{ column entries}$, $1 \leq j \leq n$.
- Find the minimum entry (say x) from last row and last column of B (computed in previous steps).
- For other entry: $B(i,j) = A(i,j) + x$ for all $1 \leq i \leq m$ and $1 \leq j \leq n$.
- Store x as the last entry of B. That is, $B(p,q) = x$.

Example $(m,n)=(3,4)$



Objective

Compute B matrix using three different kernels (if sticking to Section 4 guidelines) following the launch configuration and input/output specification.

- $(m*n)/k$ threads where k is the number entries processed by a thread (Here, $\text{DoC} = 32 - k$). If $k=1$ then one thread operates on one entry, that is, otherwise called as fully coalesced.
- Assume the matrix A is stored as a single dimension integers in row-major order ($\text{int}^* A$). For $k > 1$, $m*n$ is divisible by k .
- B matrix ($\text{int}^* B$) must be populated in row-major order and printed as in Section 2.2.

2. Input and Output Formats

1. Input

$m \ n \ k$

$a_1 \ a_2 \ a_3 \ \dots$ // all $m*n$ entries separated by space

2. Output

$b_{11} \ b_{12} \ \dots \ b_{1q}$

$b_{21} \ b_{22} \ \dots \ b_{2q}$

$\dots \ b_{3q}$

$b_{p1} \ b_{p2} \ \dots \ b_{pq}$

// Print as 2D Matrix format

3. Limits and Constraints

- $-100 \leq a_{ij} \leq 100$
- $2^1 \leq m \leq n \leq 2^{13}$
- $k = \{1, 2, 4, 8, 16, 32\}$. Default $k = 1$.
- $\text{INT_MIN} < \text{RowSum}, \text{ColSum} < \text{INT_MAX}$.

4. Example $(m, n) = (3, 4)$

Step 0	Step 1. rowSum/colSum	Step 2. FindMin. Here $x=6$	Step 4. Update
0 1 2 3 -	0 1 2 3 6	0 1 2 3 <u>6</u>	6 7 8 9 6
4 5 6 7 -	4 5 6 7 22	4 5 6 7 22	10 11 12 13 22
8 9 10 11 -	8 9 10 11 38	8 9 10 11 38	14 15 16 17 38
- - - - -	12 15 18 21 -	12 15 18 21 -	12 15 18 21 <u>6</u>

3. Submission Guidelines

Please follow the submission guidelines seriously to avoid penalty.

1. The `main()` function reads the input and writes the output from/to `stdin/out` as specified in Section 2. Do NOT use file reading.

e.g After compilation, to run you can use the input redirection. `./a.out < 1.txt`

where 1.txt is the input file.

2. There should NOT be any other `printf/scanf` statement. The evaluation is semi-automated.
3. Submit a single CUDA (.cu) file containing the required kernels and `main()` on the [Moodle](#).
4. File name should be `ROLLNUMBER.cu`. Replace ROLLNUMBER with your roll number in capital letters.
5. Download your file, and make sure it was the one you intended to submit.
6. Non-compliance to any of the above guidelines invites **penalty**.

4. Kernel Signatures and Hints

- `__global__ void sumRandC (int* A, int* B, int m, int n, int p, int q, int k=1)`
- `__global__ void findMin (int* A, int* B, int m, int n, int p, int q, int k=1)`
- `__global__ void updateMin (int* A, int* B, int m, int n, int p, int q, int k=1)`
- There is no strict constraint on using the above kernels, but we think this is a good /recommended design decision for the three steps/phases. Some arguments are redundant. You may use other device functions and/or device/shared variables as required.
- [Hint] Coding the CPU computation first always helps in writing the GPU kernels better.
- [Hint] It is advisable to code for $k = 1$ and later incorporate computation for $k > 1$.
- [Learning purpose only] Plot as a graph (or in a spreadsheet) with executing times of the kernels vs varying k values on the same input. Not for submission! `nvprof ./a.out < 14.txt`