Name: *Souham Biswas* ***CWID***: *A20365242* ***Email***: *sbiswas7@hawk.iit.edu*

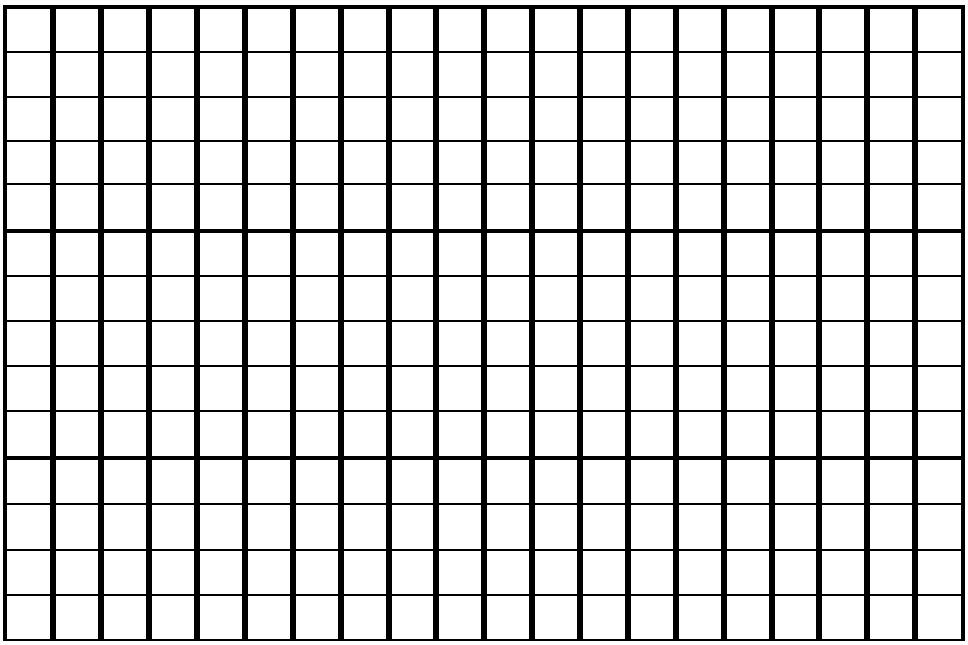
CS 546 – Parallel and Distributed Processing, HW 4

On an NVIDIA GPU, there are present processing units called Streaming Multiprocessors or SMs which handle multiple threads each. Each SM contains sets of Streaming Processors which execute threads in parallel. A given SM has a fixed limit on the number of active threads it can handle. This number is a multiple of the warp size pertaining to the GPU.

The warp size is the number of threads in a threadblock which are executed concurrently at one go on an SM. An SM can handle a fixed number of thread blocks at a given time. Now, if the sizes of the thread blocks deployed on to an SM are multiples of the warp size (currently, the warp size of an NVIDIA GPU is 32), maximum compute efficiency is achieved. Moreover, this also ensures high memory coalescing.

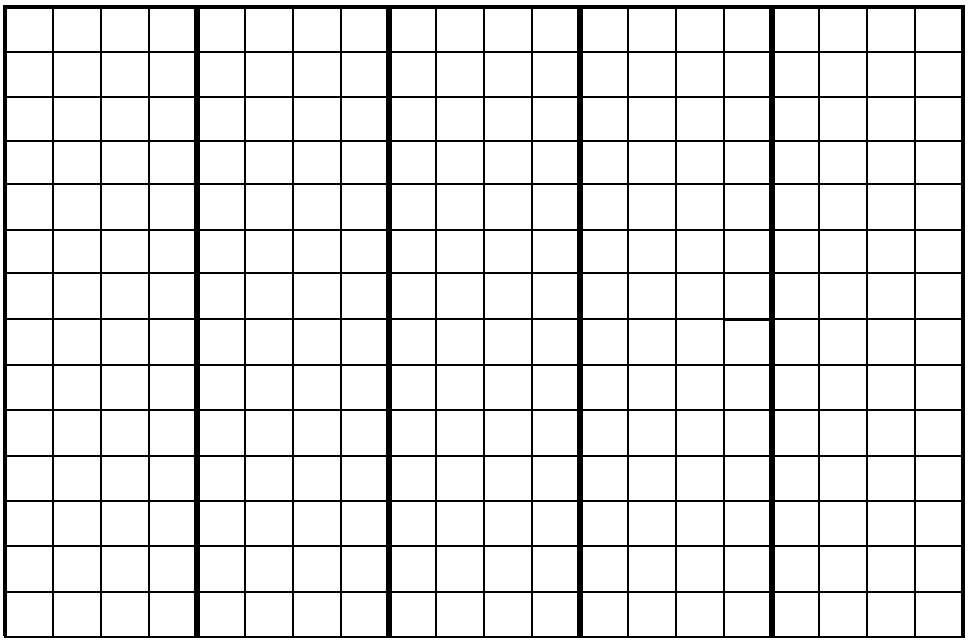
Memory coalescing refers to the phenomena of a given thread along with its neighboring threads accessing a chunk of global memory whenever the global memory on the GPU is accessed. When the neighboring threads read from a memory location which is in that chunk, extra memory accesses are avoided. This is called memory coalescing.

The input 2D array is firstly summed up in chunks with the size of each chunk being a multiple of the warp size. It is visually illustrated below –

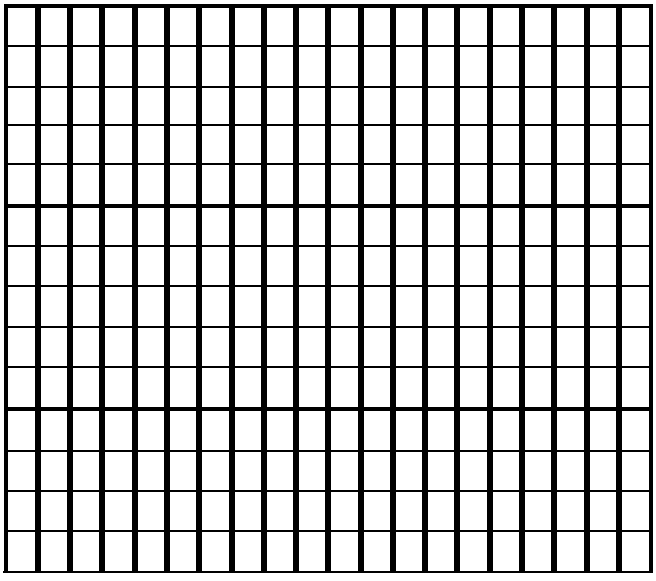


In the illustration above, the figure represents a 2D array and each block enclosed in bold represents a thread block. This diagram is not to scale. So each thread block reduces its contents to their sums using parallel reduction algorithm. The number of elements reduced by each thread is governed by the variable fragSize.

Following this step, sums of the elements in each column is calculated and stored in a device array called d_Avg. In this process, the thread blocks are divided in the following way –

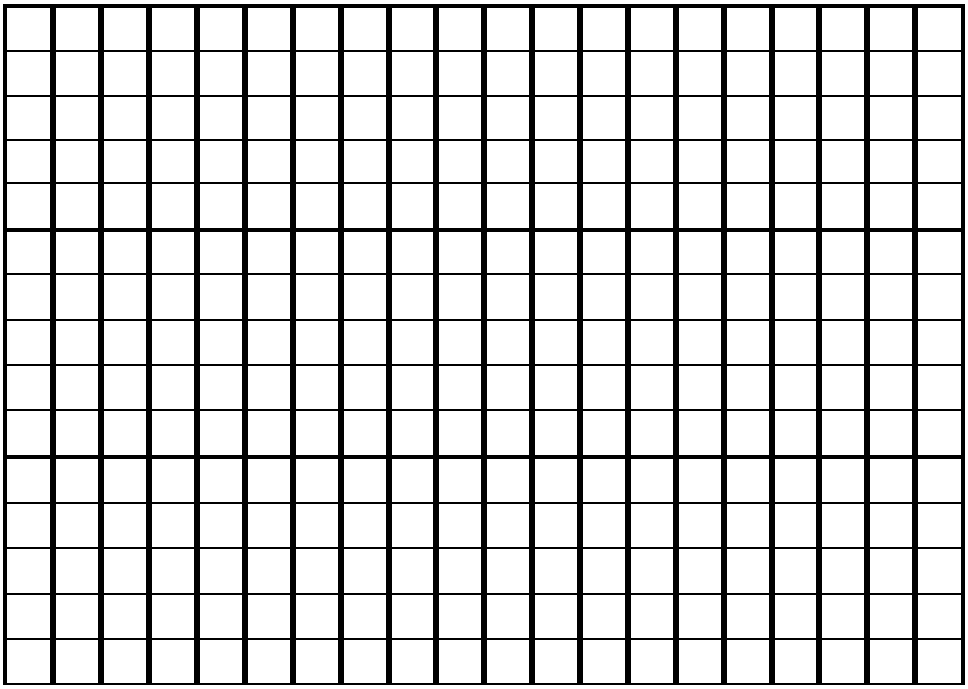


Here, each thread block handles a set of columns and each thread inside the thread block sums up its respective column and calculates the average for that column. Care has been taken to ensure the sizes of the thread blocks be multiples of the warp size.
Subsequently, the variance for each element is calculated. The thread blocks are created in the following manner as explained previously –



Once this is done, the variances of each column are averaged out in a similar fashion as the average of the original matrix was computed.

Finally, we have the sets of averages & variances of each column. These are finally used to compute the norm values of each column element in the matrix. The thread block division is done in the following way-



To copy the 2D array from the host to GPU, the 2D array had to be flattened to a 1D array. Subsequently, the processed flattened 1D array containing the norm values returned from the GPU had to be unflattened back to a 2D array.

It is documented in cuda's software documentation that the first call to the cuda environment implicitly performs the initialization of the cuda environment. Therefore, to get an accurate time on the parallel algorithm implemented, a dummy call to an empty cuda kernel has been placed in the beginning of the main which implicitly performs the initialization. More information at https://devtalk.nvidia.com/default/topic/392429/first-cudamalloc-takes-long-time-/.

The program can be run with the following simple command –

`./matrixNorm_GPU` ***`[input matrix size]`***

Performance results of the code are presented below –

1.) N=5 :-

```
Matrix dimension N = 5.

Initializing...

A =
        32808.59, 34796.65, 7837.09, 94.84, 4729.39;
        61726.98, 62589.06, 44852.48, 17626.82, 48507.88;
        42266.46, 14097.56, 6001.17, 20746.15, 60565.05;
        3341.93, 27352.25, 17165.40, 1439.11, 65345.25;
        51275.93, 20663.41, 46028.73, 42750.73, 18159.31;

Starting clock.
Stopped clock.

B =
        -0.2747913003, 0.1722568423, -0.9390117526, -1.0590636730, -1.4529112577;
        1.1765253544, 1.8248821497, 1.1624473333, 0.0705725253, 0.3784339130;
        0.1998673379, -1.0585769415, -1.0432417393, 0.2715602517, 0.8828107119;
        -1.7536240816, -0.2704110742, -0.4094198048, -0.9724486470, 1.0827764273;
        0.6520223618, -0.6681506634, 1.2292257547, 1.6893792152, -0.8911098838;

Elapsed time = 0.436 ms.
(CPU times are accurate to the nearest 0.001 ms)
My total CPU time for parent = 0 ms.
My system CPU time for parent = 0 ms.
My total CPU time for child processes = 0 ms.
----------------------------------------
```

2.) N=500 :-

```
Matrix dimension N = 500.

Initializing...

Starting clock.
Stopped clock.

Elapsed time = 29.815 ms.
(CPU times are accurate to the nearest 0.001 ms)
```

```
My total CPU time for parent = 0 ms.
My system CPU time for parent = 0 ms.
My total CPU time for child processes = 0 ms.
------------------------------------------
```

3.) N=1000 :-

```
Matrix dimension N = 1000.

Initializing...

Starting clock.
Stopped clock.

Elapsed time = 90.375 ms.
(CPU times are accurate to the nearest 0.001 ms)
My total CPU time for parent = 0.001 ms.
My system CPU time for parent = 0 ms.
My total CPU time for child processes = 0 ms.
------------------------------------------
```

4.) N=2000 :-

```
Matrix dimension N = 2000.

Initializing...

Starting clock.
Stopped clock.

Elapsed time = 273.456 ms.
(CPU times are accurate to the nearest 0.001 ms)
My total CPU time for parent = 0.004 ms.
My system CPU time for parent = 0 ms.
My total CPU time for child processes = 0 ms.
------------------------------------------
```

5.) N=3000 :-

```
Matrix dimension N = 3000.

Initializing...

Starting clock.
Stopped clock.
```

```
Elapsed time = 590.905 ms.
(CPU times are accurate to the nearest 0.001 ms)
My total CPU time for parent = 0.011 ms.
My system CPU time for parent = 0.001 ms.
My total CPU time for child processes = 0 ms.
----------------------------------------
```

6.) N=4000 :-

```
Matrix dimension N = 4000.

Initializing...

Starting clock.
Stopped clock.

Elapsed time = 873.572 ms.
(CPU times are accurate to the nearest 0.001 ms)
My total CPU time for parent = 0.017 ms.
My system CPU time for parent = 0.002 ms.
My total CPU time for child processes = 0 ms.
----------------------------------------
```

7.) N=5000 :-

```
Matrix dimension N = 5000.

Initializing...

Starting clock.
Stopped clock.

Elapsed time = 1809.33 ms.
(CPU times are accurate to the nearest 0.001 ms)
My total CPU time for parent = 0.03 ms.
My system CPU time for parent = 0.002 ms.
My total CPU time for child processes = 0 ms.
----------------------------------------
```

8.) N=6000 :-

```
Matrix dimension N = 6000.

Initializing...

Starting clock.
Stopped clock.

Elapsed time = 2034.31 ms.
(CPU times are accurate to the nearest 0.001 ms)
My total CPU time for parent = 0.037 ms.
My system CPU time for parent = 0.003 ms.
My total CPU time for child processes = 0 ms.
------------------------------------------
```

9.) N=7000 :-

```
Matrix dimension N = 7000.

Initializing...

Starting clock.
Stopped clock.

Elapsed time = 3050.08 ms.
(CPU times are accurate to the nearest 0.001 ms)
My total CPU time for parent = 0.058 ms.
My system CPU time for parent = 0.005 ms.
My total CPU time for child processes = 0 ms.
------------------------------------------
```

10.) N=8000 :-

```
Matrix dimension N = 8000.

Initializing...

Starting clock.
Stopped clock.

Elapsed time = 3820.71 ms.
(CPU times are accurate to the nearest 0.001 ms)
My total CPU time for parent = 0.07 ms.
My system CPU time for parent = 0.007 ms.
My total CPU time for child processes = 0 ms.
------------------------------------------
```

Furthermore, a test suite was also written (contained in folder 'hw4/test') which compares the output of the parallel algorithm with the serial algorithm's output, checks for correctness and also calculates the speedup of the parallel algorithm over the serial algorithm. It can be run with the following simple command –

./matrixNorm_GPU_test *[input matrix size]*

Its outputs for different input sizes are presented below –

1.) N=5 :-

```
Matrix dimension N = 5.

Initializing...

A =
        17441.27, 33285.46, 32307.12, 62712.92, 43773.04;
        42328.80, 3841.90, 25500.31, 21294.23, 20843.65;
        27678.88, 280.85, 7304.43, 862.06, 48477.38;
        7494.67, 29478.01, 11154.74, 24706.53, 27013.38;
        41486.52, 46428.36, 32212.40, 38149.02, 6594.04;

Starting GPU clock.
Stopped GPU clock.

B_GPU =
        -0.7270663381, 0.5977033377, 1.0063673258, 1.6229443550, 0.9436867237;
        1.1109559536, -1.0590103865, 0.3608160019, -0.4037167132, -0.5555543303;
        0.0290139001, -1.2593815327, -1.3648639917, -1.4034845829, 1.2512803078;
        -1.4616539478, 0.3834676147, -0.9997035861, -0.2367491722, -0.1521454602;
        1.0487511158, 1.3372204304, 0.9973841310, 0.4210060239, -1.4872671366;

Elapsed GPU time = 0.444 ms.

======================================================================

A =
        17441.27, 33285.46, 32307.12, 62712.92, 43773.04;
        42328.80, 3841.90, 25500.31, 21294.23, 20843.65;
        27678.88, 280.85, 7304.43, 862.06, 48477.38;
        7494.67, 29478.01, 11154.74, 24706.53, 27013.38;
        41486.52, 46428.36, 32212.40, 38149.02, 6594.04;

Starting CPU clock.

Computing Serially.
Stopped CPU clock.
```

```
B_CPU =
        -0.7270662189, 0.5977033973, 1.0063673258, 1.6229442358, 0.9436867237;
        1.1109558344, -1.0590102673, 0.3608160317, -0.4037166834, -0.5555543303;
        0.0290138964, -1.2593812943, -1.3648641109, -1.4034844637, 1.2512803078;
        -1.4616538286, 0.3834676743, -0.9997036457, -0.2367491722, -0.1521454602;
        1.0487509966, 1.3372204304, 0.9973841310, 0.4210059941, -1.4872671366;

Elapsed CPU time = 0.012 ms.

SPEEDUP = 0.027027

Array B_GPU & B_CPU are equal.
SOLUTION IS CORRECT
```

2.) N=500 :-

```
Matrix dimension N = 500.

Initializing...

Starting GPU clock.
Stopped GPU clock.

Elapsed GPU time = 29.566 ms.

==============================================================================

Starting CPU clock.

Computing Serially.
Stopped CPU clock.

Elapsed CPU time = 60.304 ms.

SPEEDUP = 2.039640

Array B_GPU & B_CPU are equal.
SOLUTION IS CORRECT
```

3.) N=1000 :-

```
Matrix dimension N = 1000.

Initializing...

Starting GPU clock.
Stopped GPU clock.

Elapsed GPU time = 99.05 ms.

================================================================================

Starting CPU clock.

Computing Serially.
Stopped CPU clock.

Elapsed CPU time = 268.713 ms.

SPEEDUP = 2.712903

Array B_GPU & B_CPU are equal.
SOLUTION IS CORRECT
```

4.) N=2000 :-

```
Matrix dimension N = 2000.

Initializing...

Starting GPU clock.
Stopped GPU clock.

Elapsed GPU time = 297.7 ms.

================================================================================

Starting CPU clock.

Computing Serially.
Stopped CPU clock.

Elapsed CPU time = 993.053 ms.

SPEEDUP = 3.335751

Array B_GPU & B_CPU are equal.
SOLUTION IS CORRECT
```

5.) N=3000 :-

```
Matrix dimension N = 3000.

Initializing...

Starting GPU clock.
Stopped GPU clock.

Elapsed GPU time = 610.851 ms.

==============================================================================

Starting CPU clock.

Computing Serially.
Stopped CPU clock.

Elapsed CPU time = 2270.21 ms.

SPEEDUP = 3.716465

Array B_GPU & B_CPU are equal.
SOLUTION IS CORRECT
```

6.) N=4000 :-

```
Matrix dimension N = 4000.

Initializing...

Starting GPU clock.
Stopped GPU clock.

Elapsed GPU time = 1019.59 ms.

==============================================================================

Starting CPU clock.

Computing Serially.
Stopped CPU clock.

Elapsed CPU time = 3936.4 ms.

SPEEDUP = 3.860753

Array B_GPU & B_CPU are equal.
SOLUTION IS CORRECT
```

7.) N=5000 :-

```
Matrix dimension N = 5000.

Initializing...

Starting GPU clock.
Stopped GPU clock.

Elapsed GPU time = 1803.89 ms.

==============================================================================

Starting CPU clock.

Computing Serially.
Stopped CPU clock.

Elapsed CPU time = 6373.64 ms.

SPEEDUP = 3.533283

Array B_GPU & B_CPU are equal.
SOLUTION IS CORRECT
```

8.) N=6000 :-

```
Matrix dimension N = 6000.

Initializing...

Starting GPU clock.
Stopped GPU clock.

Elapsed GPU time = 2038.39 ms.

==============================================================================

Starting CPU clock.

Computing Serially.
Stopped CPU clock.

Elapsed CPU time = 9221.49 ms.

SPEEDUP = 4.523910

Array B_GPU & B_CPU are equal.
SOLUTION IS CORRECT
```

9.) N=7000 :-

```
Matrix dimension N = 7000.

Initializing...

Starting GPU clock.
Stopped GPU clock.

Elapsed GPU time = 3169.13 ms.

================================================================================

Starting CPU clock.

Computing Serially.
Stopped CPU clock.

Elapsed CPU time = 12855.4 ms.

SPEEDUP = 4.056449

Array B_GPU & B_CPU are equal.
SOLUTION IS CORRECT
```

10.) N=8000 :-

```
Matrix dimension N = 8000.

Initializing...

Starting GPU clock.
Stopped GPU clock.

Elapsed GPU time = 3969.62 ms.

================================================================================

Starting CPU clock.

Computing Serially.
Stopped CPU clock.

Elapsed CPU time = 16950.1 ms.

SPEEDUP = 4.269961

Array B_GPU & B_CPU are equal.
SOLUTION IS CORRECT
```

As seen above, the speedup of the parallel code keeps increasing with the input size. Spikes in the speedup at specific values of N may be observed due to the dynamic nature of the GPU and the different applications using it at a given time. Furthermore, it has been observed that the results are consistent with the results of the serial algorithm establishing its correctness.