Assignment 2 : CS6023 GPU Programming

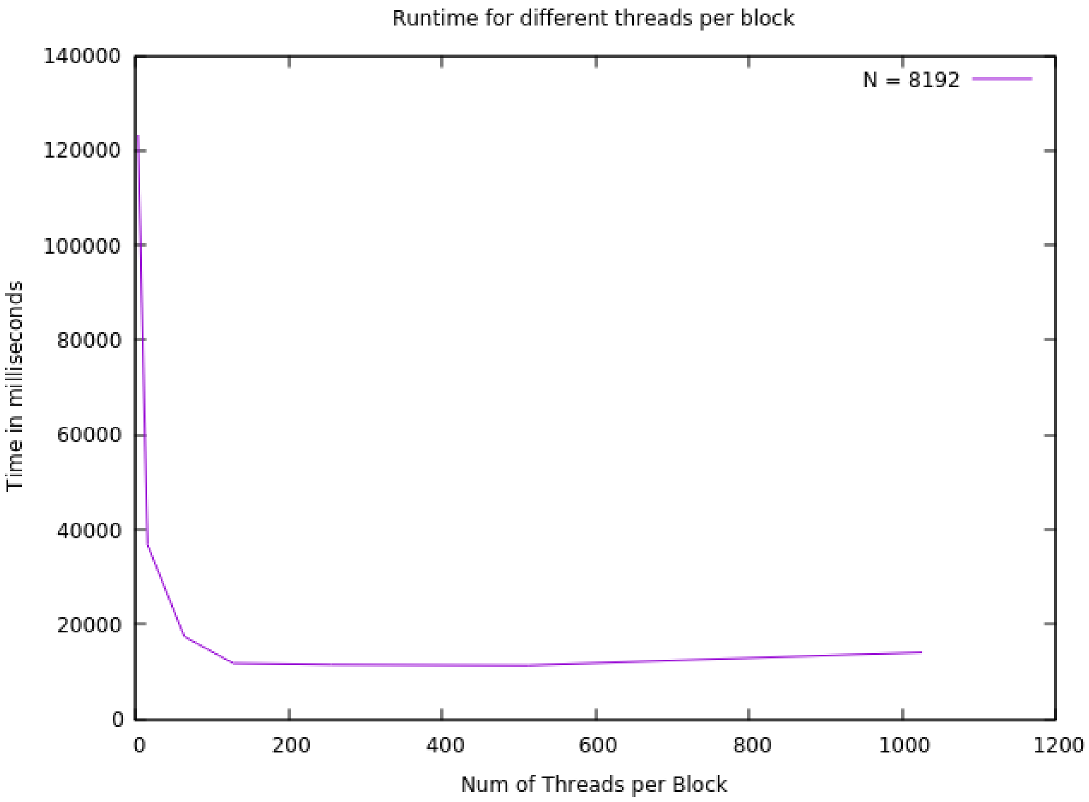
by CS15B049

Q1 : The code is included.

Q2: Matrix Size : 8192*8192

S.No	Threads per Block	Runtime in milliseconds
1	2*2	123207.99
2	4*4	36875.53
3	8*8	17413.35
4	16*8	11756.33
5	16*16	11465.96
6	16*32	11328.14
7	32*32	14089.15
8	64*64	Error as numof threads >1024

The Runtime graph :

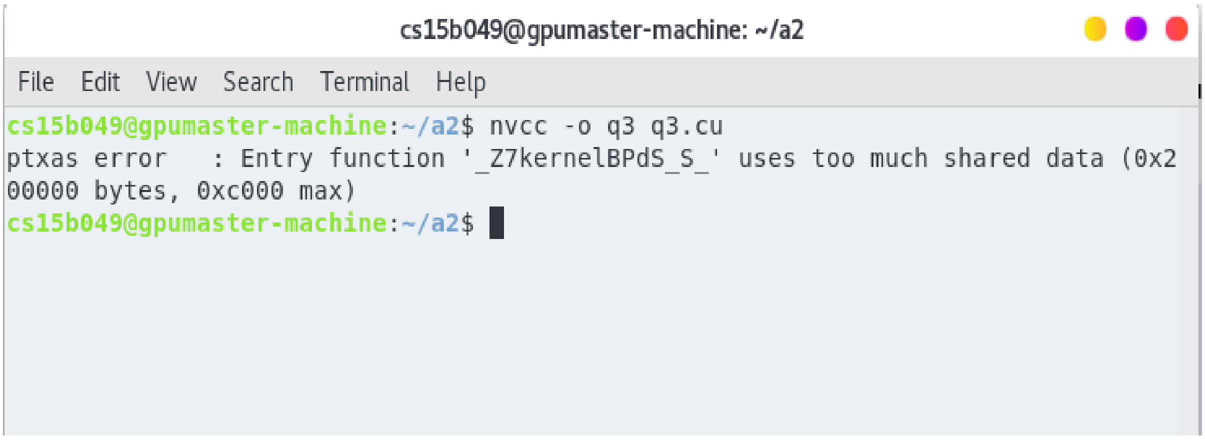


As we increase the number of threads per block the run time decreses exponentially and then after 512(32*16) it increases.

Q3: - Using 16x16 threads per block.

a) The code is included.

b) If we change the size of the array to 8192 , the code doesnt compile through nvcc , it is due to the fact that size of shared memory is limited and the size of the aaray we initaialize goes beyond its size. It gives compile time error as arrays are statically allocated and compiler knows the size of shared memory and does a check for the allocation is possible or not.



```
cs15b049@gpumaster-machine:~/a2$ nvcc -o q3 q3.cu
ptxas error   : Entry function '_Z7kernelBPdS_S_' uses too much shared data (0x2
00000 bytes, 0xc000 max)
cs15b049@gpumaster-machine:~/a2$
```

Q4: Size of matrix = 8192*8192
Threads per block = 16*16

Runtime using tiling : 3833.963135 milliseconds
Runtime using q1 (without tiling and shared memory) : 11465.964844 milliseconds

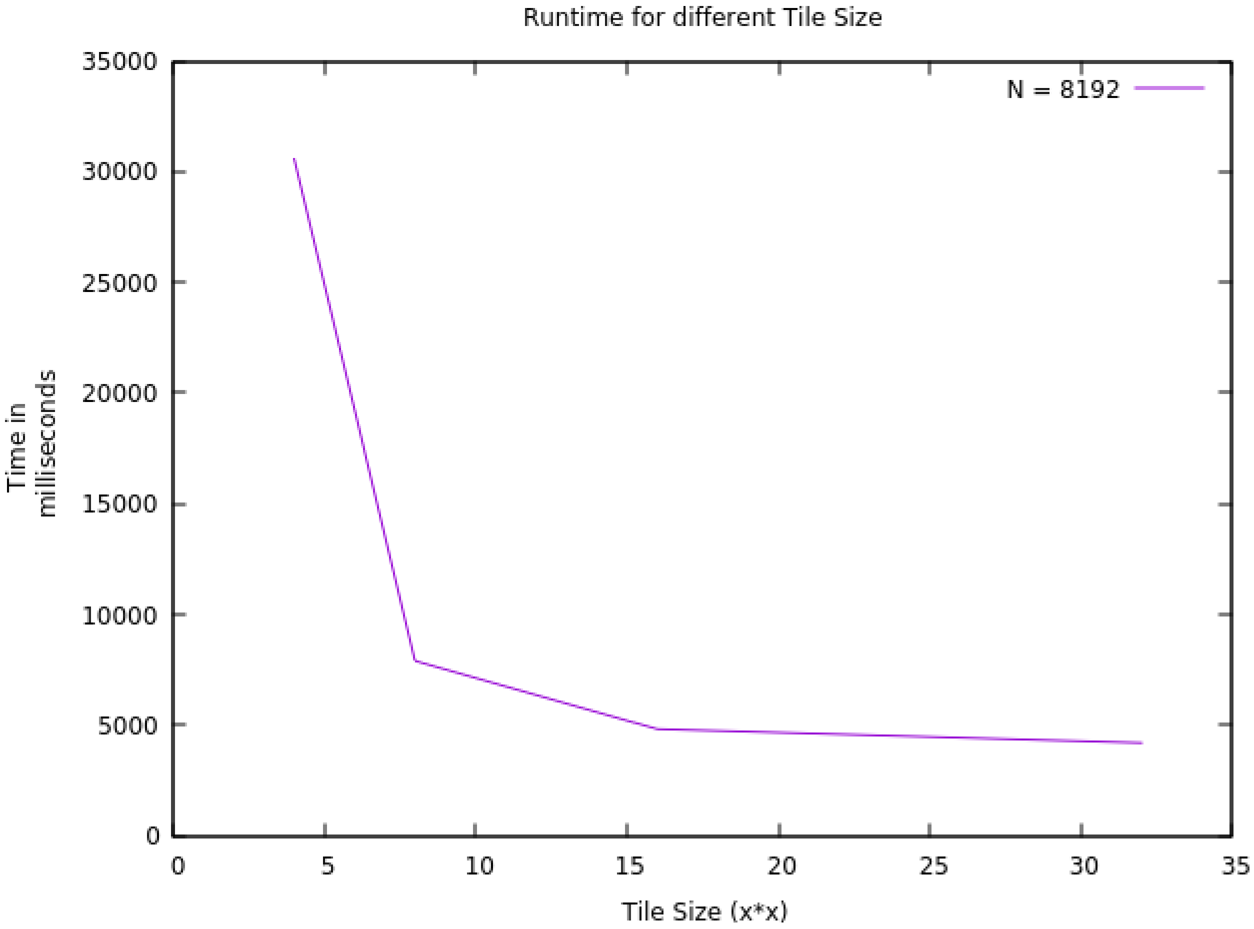
as the above with tilling runtime is reduces to 1/3 of without tiling.

It is due the fact that in tiling we are utilising shared memory which is much faster then the DRAM global memory, so the latency of accessing the element from DRAM is reduced as we utilise the temporal and spatial locality as well as brust section with the use of tiling, so overall latency reduces.

Q5: Matrix Size : 8192*8192
Number of threads per block = TILE_SIZE * TILE_SIZE

Optimal Tilse Size = 32*32

below is th runtime graph with diffenrent tile size.



Q6 : Matrix A size : 4096*8192
Matrix B size : 8192*16384

Runtime : 11539.041016 milliseconds.