Matrix Multiplication Runtime

M.A.S SURANGA (CST140043)

Generated on Friday 27th October, 2017 14:49

Contents

1 Introduction . . . 3

2 CPU Program runtime . . . 4

3 GPU Global Memory Program(CUDA) runtime . . . 5

4 GPU Shared Memory Program(CUDA) runtime . . . 6

5 CPU vs GPU Performance Comparison . . . 7

6 Shared Memory vs Global memory of GPU . . . 8

7 Conclusion . . . 10

8 References . . . 11

1 Introduction

Report Content

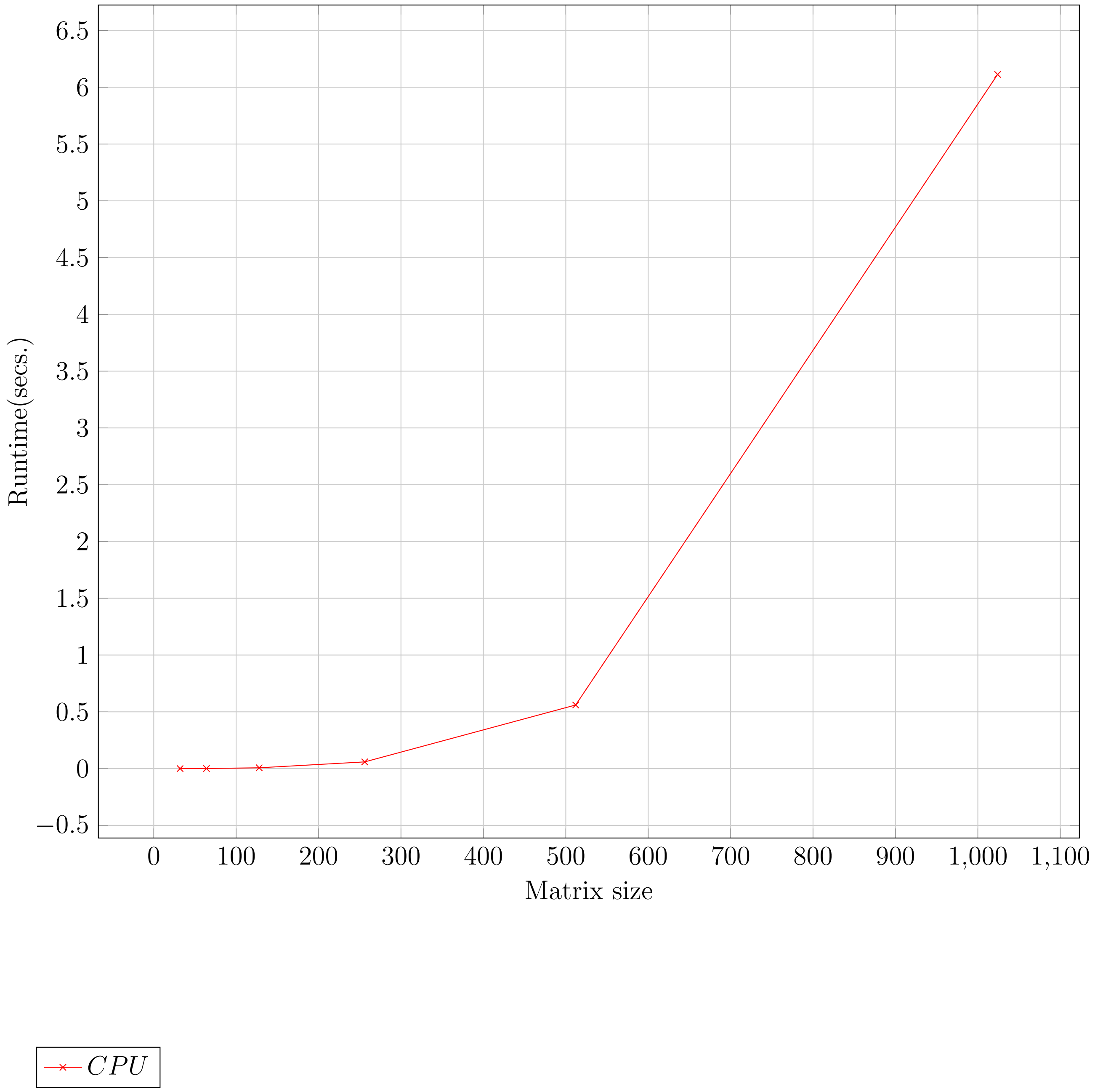
This report compares the performance of matrix multiplication on CPU and Nvidia GPU. CPU program uses the naive approach for computation and GPU programs contain two parallel source codes that uses global memory and shared memory.

Testcases Details

Matrix Sizes	32 64 128 256 512 1024
Block Size(Threads Per Block)	16 64 256 1024
No. of iterations	4
Fixed Matrix Size	1024
Fixed Block Size	256

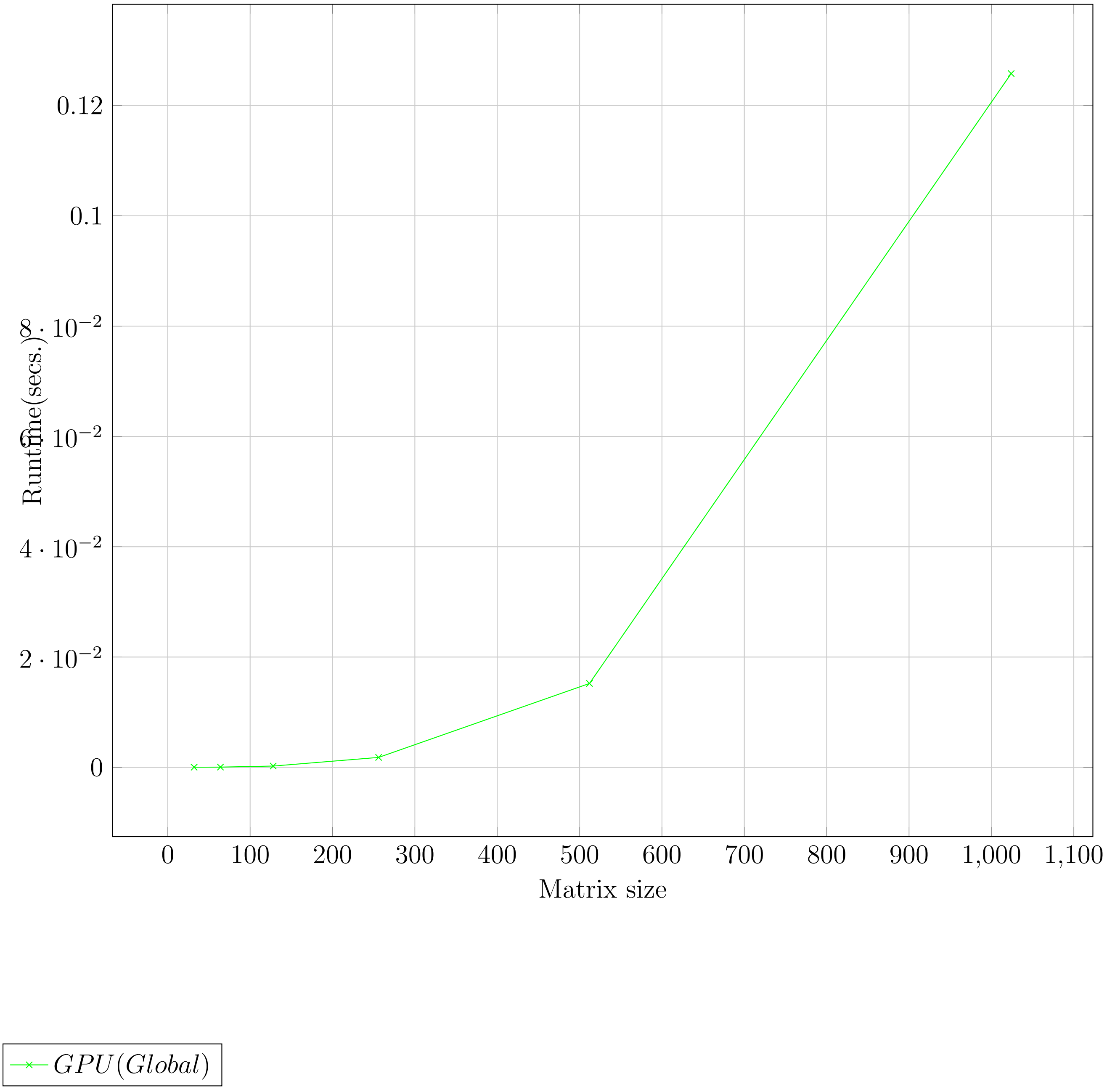
2 CPU Program runtime

This program is executed in normal CPU and x, y values are extracted from the average runtime of each different matrix sizes.



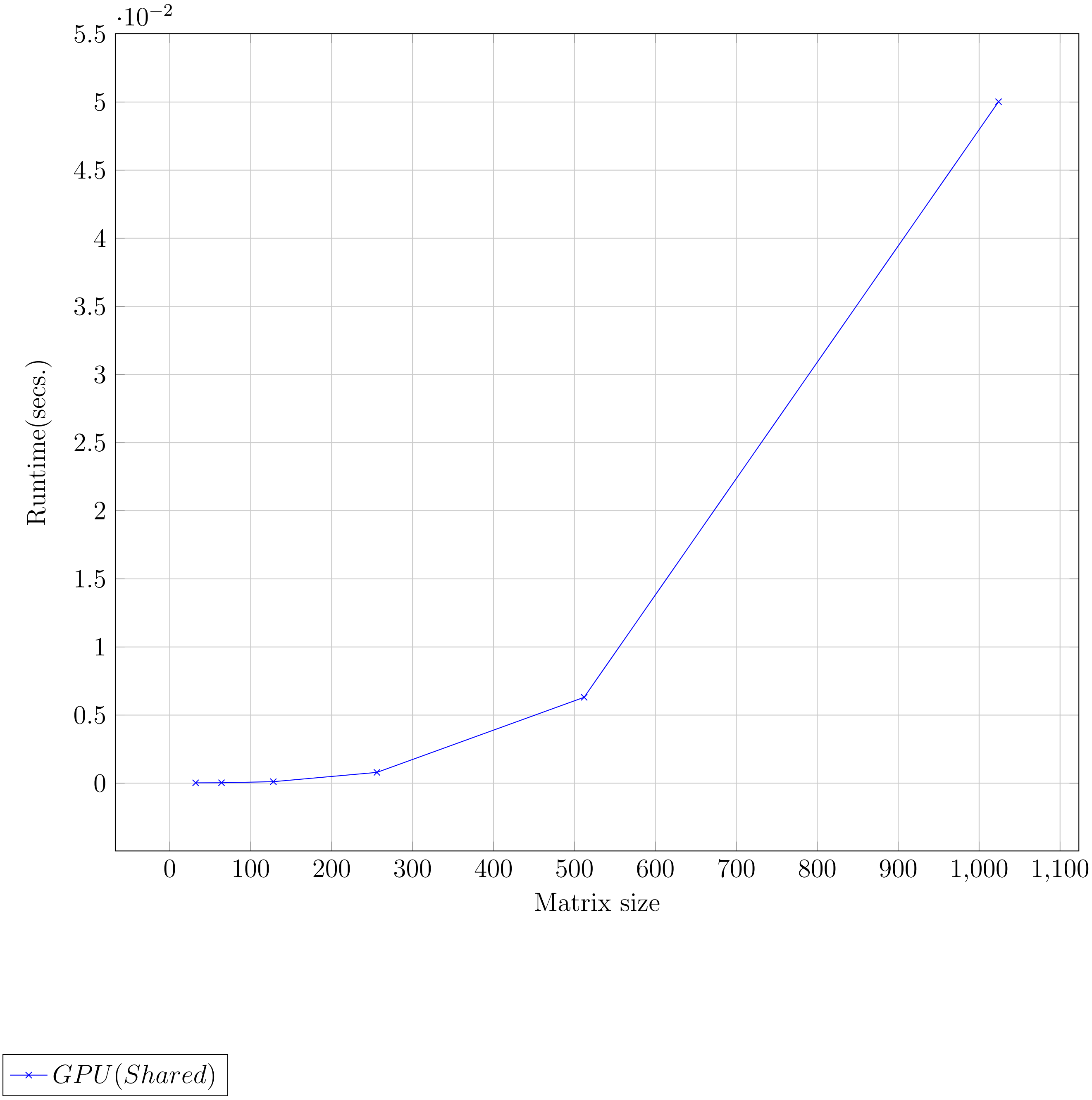
3 GPU Global Memory Program(CUDA) runtime

This program is executed in Nvdia GPU (Global memory used) and x, y values are extracted from the average runtime of each different matrix sizes.



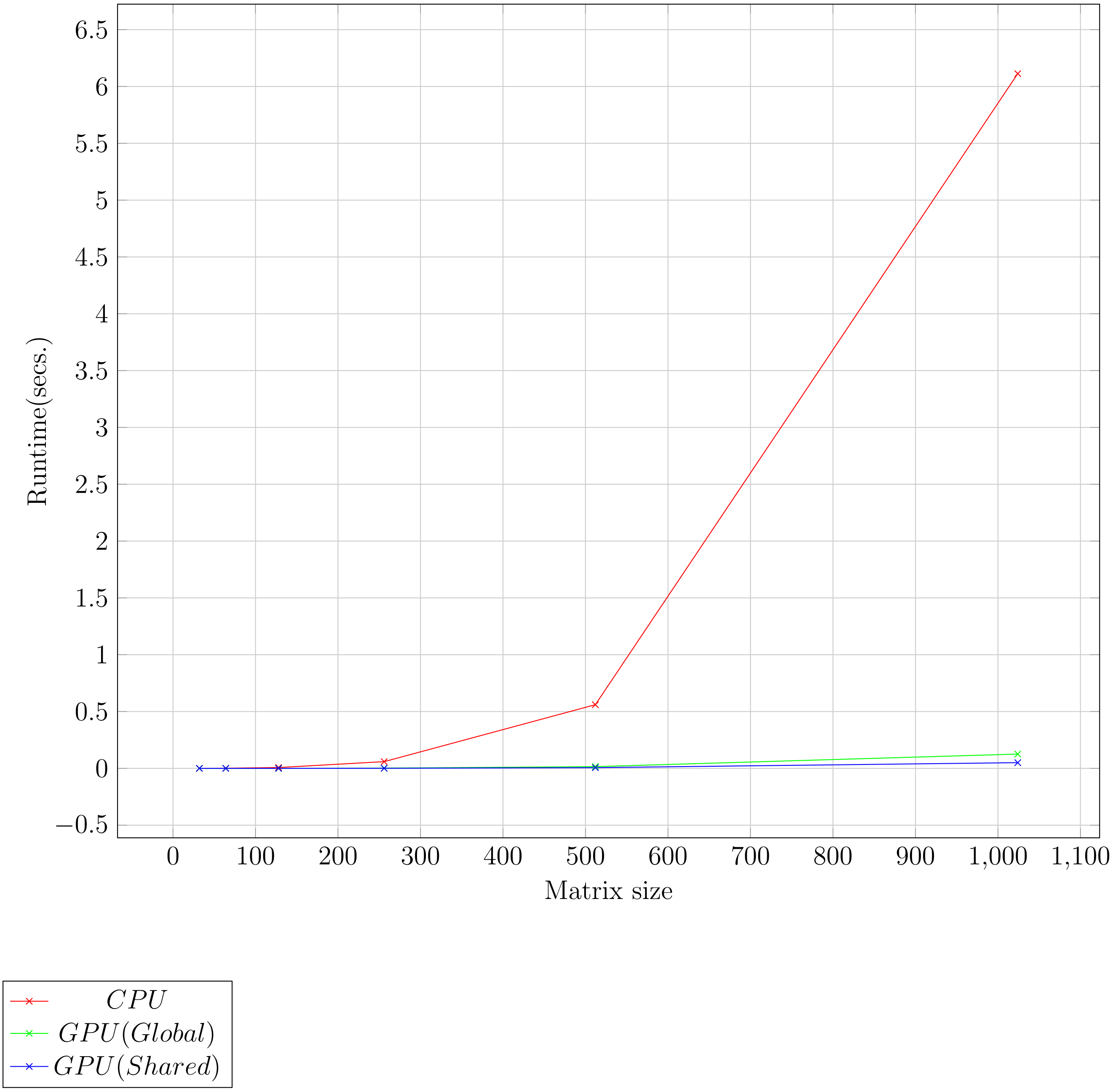
4 GPU Shared Memory Program(CUDA) runtime

This program is executed in Nvidia GPU(Shared memory) and x, y values are extracted from the average runtime of each different matrix sizes.



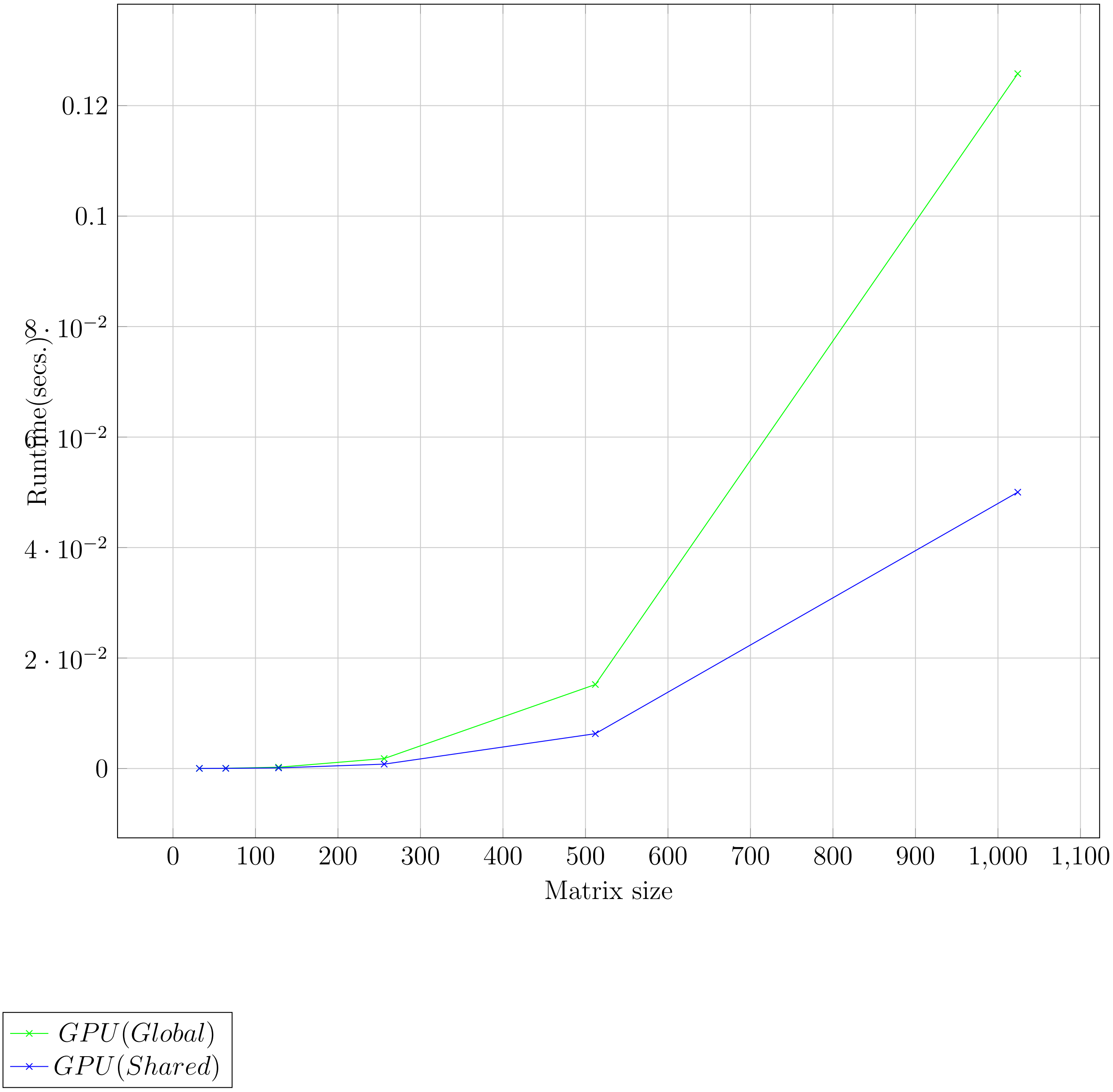
5 CPU vs GPU Performance Comparison

Previous plots are compared within same graph. According to the plots it is clear that GPU execution is giving very good performance than the CPU.



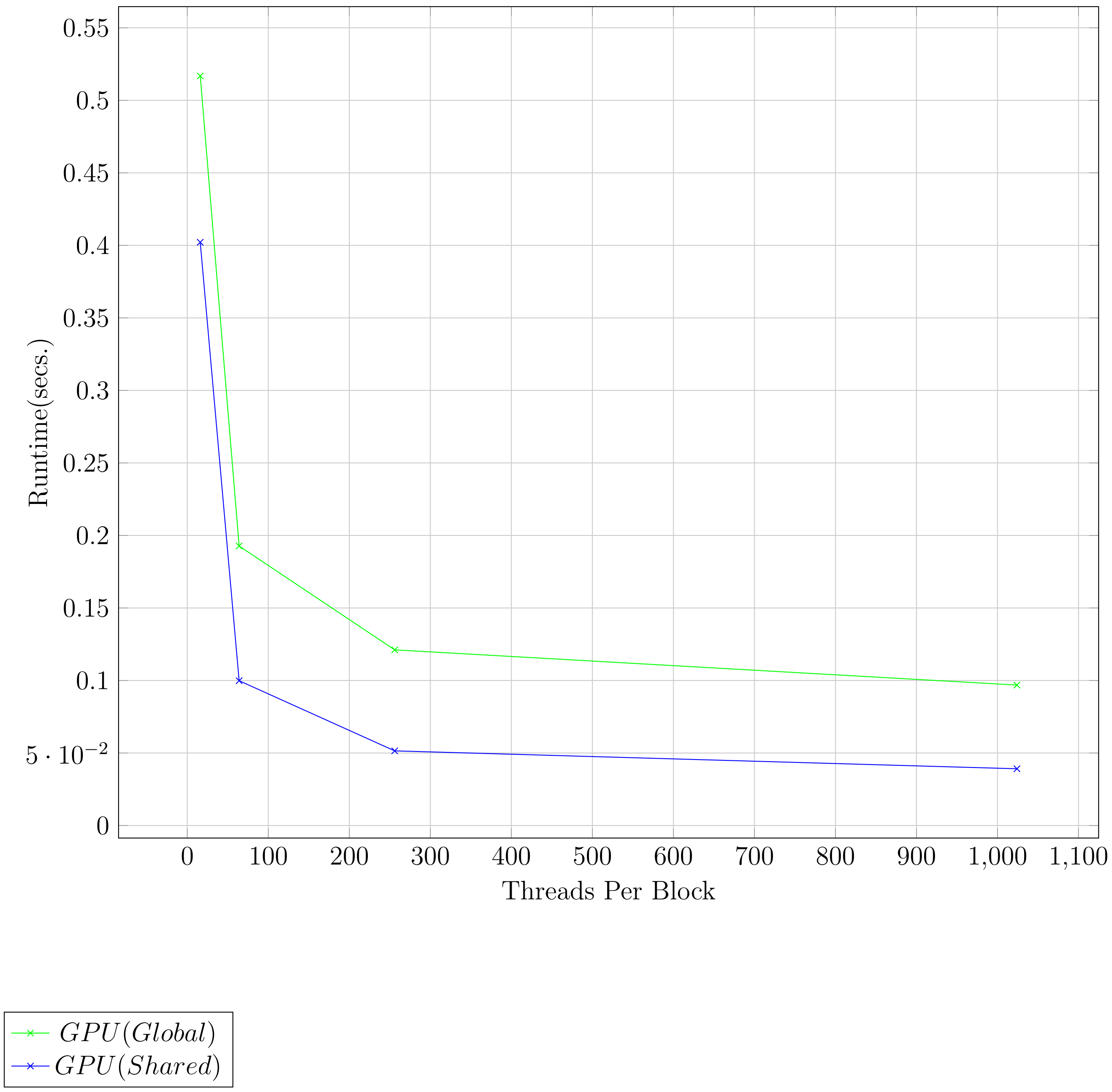
6 Shared Memory vs Global memory of GPU

This graph compares the efficiency between shared memory and global memory of the Nvidia GPU. According to the graph the Shared memory program gives good performance comparing to Global memory program.



This graph compares the efficiency between shared memory and global memory of GPU with block size(threads per block).

Fixed Matrix size - 1024 x1024



7 Conclusion

- Using first graphs it is clear that GPU execution of matrix multiplication program is giving very good performance. Thus CPU execution gives a plot similar to cubic curve since we used naive $O(n^3)$ algorithm here. Further CPU program can be optimized using several algorithms(Eg :- Solvay Strassen Algorithm).
- Shared memory decreases the execution time of the program because shared memory is onchip memory unlike global memory.
- In conclusion Shared memory approach is the most efficient method for matrix multiplication computation among all program types.
- Thus If the Block size is being increased in shared memory approach there is apparent performance gain.

Following table compares the efficiency of matrix muliplication for
1024 x1024 matrix.

Approach	**Time(secs.)**
CPU	6.1128976000
GPU (Global Memory)	.1257838000
GPU (Shared Memory)	.0500242000

8 References

- https://www.tug.org/twg/mactex/tutorials/ltxprimer-1.0.pdf.
- https://gist.github.com/LeCoupa/122b12050f5fb267e75f
- https://tex.stackexchange.com/questions