Parallel Distributed Computing CUDA

Programming Assignment 01

NVIDIA®

CUDA®

D. M. H. Hirosh (UWU/CST/14/0014)

October 27, 2017

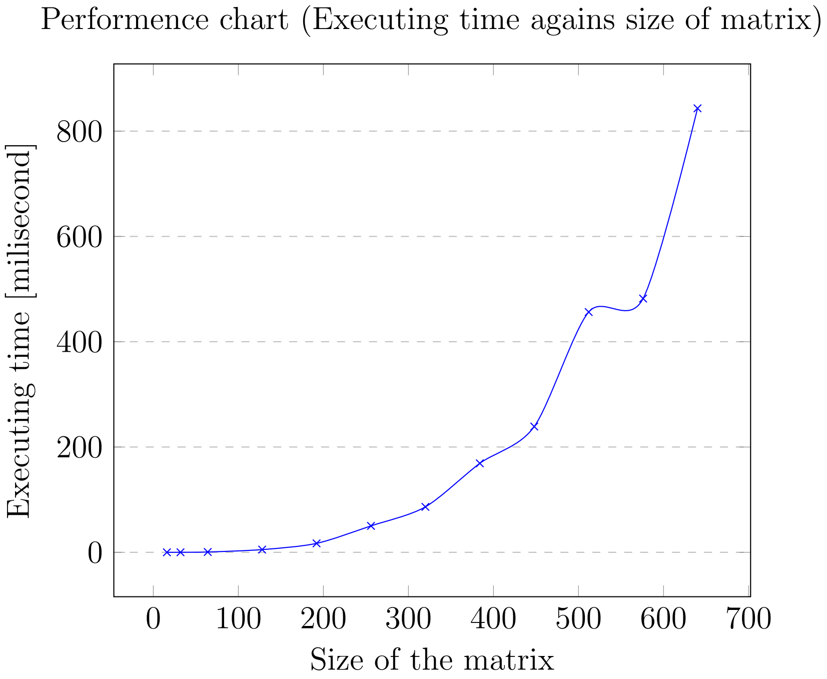
Contents

Chapter 1

Graphs and Tables to compare performance differences agains matrix size

1.1 CPU performance agains matrix size

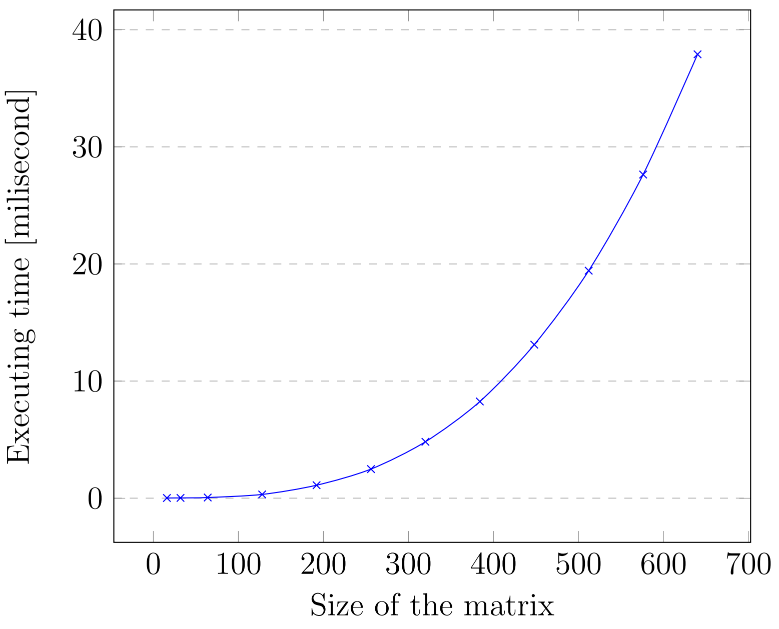
Matrix Size	Executing time
16	0.01078
32	0.08734
64	0.61869
128	5.1997
192	17.01638
256	50.2274
320	86.34207
384	169.03225
448	238.91
512	456.42606
576	481.98096
640	843.3423



1.2 GPU(Global) performance agains matrix size

Matrix Size	Executing time
16	0.01813
32	0.02392
64	0.05775
128	0.32253
192	1.1095
256	2.48709
320	4.8186
384	8.25412
448	13.11031
512	19.42658
576	27.63363
640	37.90049

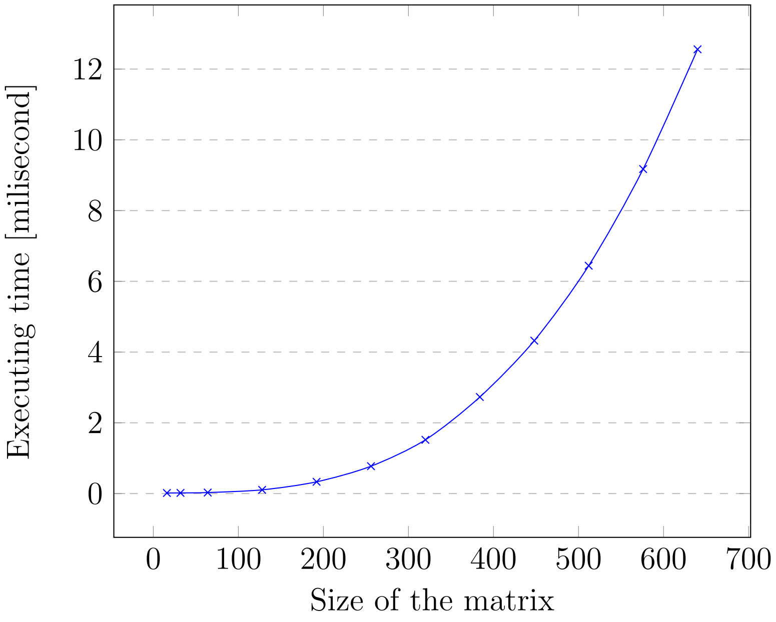
Performence chart (Executing time agains size of matrix)



1.3 GPU(Shared) performance agains matrix size

Matrix Size	Executing time
16	0.01588
32	0.01815
64	0.02836
128	0.10493
192	0.33201
256	0.77322
320	1.51907
384	2.72992
448	4.32268
512	6.44144
576	9.17374
640	12.55953

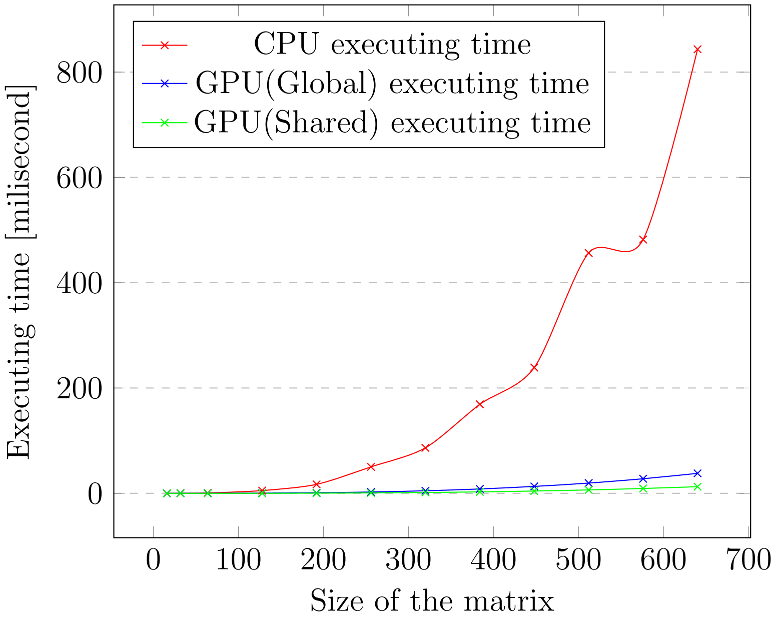
Performence chart (Executing time agains size of matrix)



1.4 CPU and GPU(Global/Shared) performance differences agains size of matrix

Matrix Size	CPU	GPU(Global)	GPU(Shared)
16	0.01078	0.01813	0.01588
32	0.08734	0.02392	0.01815
64	0.61869	0.05775	0.02836
128	5.1997	0.32253	0.10493
192	17.01638	1.1095	0.33201
256	50.2274	2.48709	0.77322
320	86.34207	4.8186	1.51907
384	169.03225	8.25412	2.72992
448	238.91	13.11031	4.32268
512	456.42606	19.42658	6.44144
576	481.98096	27.63363	9.17374
640	843.3423	37.90049	12.55953

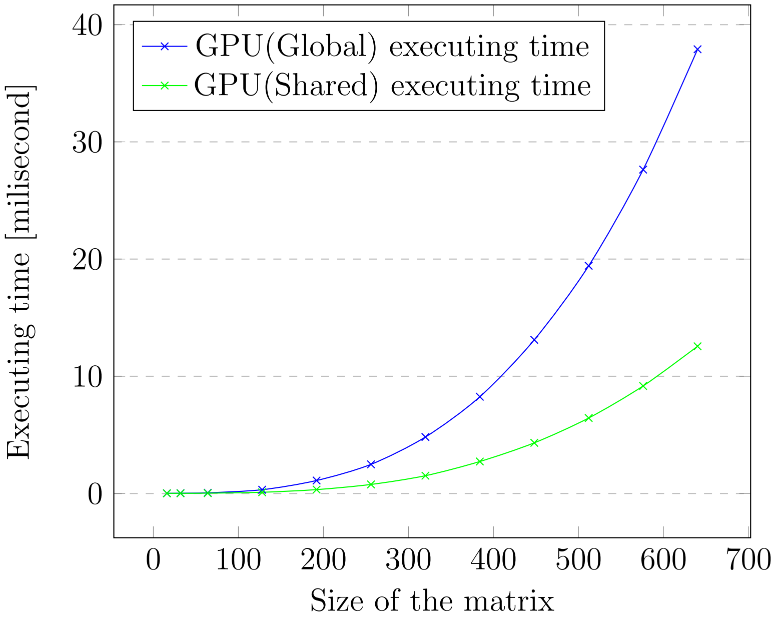
Performence chart (Executing time agains size of matrix)



1.5 GPU(Global/Shared) performance differences agains size of matrix

Matrix Size	GPU(Global)	GPU(Shared)
16	0.01813	0.01588
32	0.02392	0.01815
64	0.05775	0.02836
128	0.32253	0.10493
192	1.1095	0.33201
256	2.48709	0.77322
320	4.8186	1.51907
384	8.25412	2.72992
448	13.11031	4.32268
512	19.42658	6.44144
576	27.63363	9.17374
640	37.90049	12.55953

Performence chart (Executing time agains size of matrix)

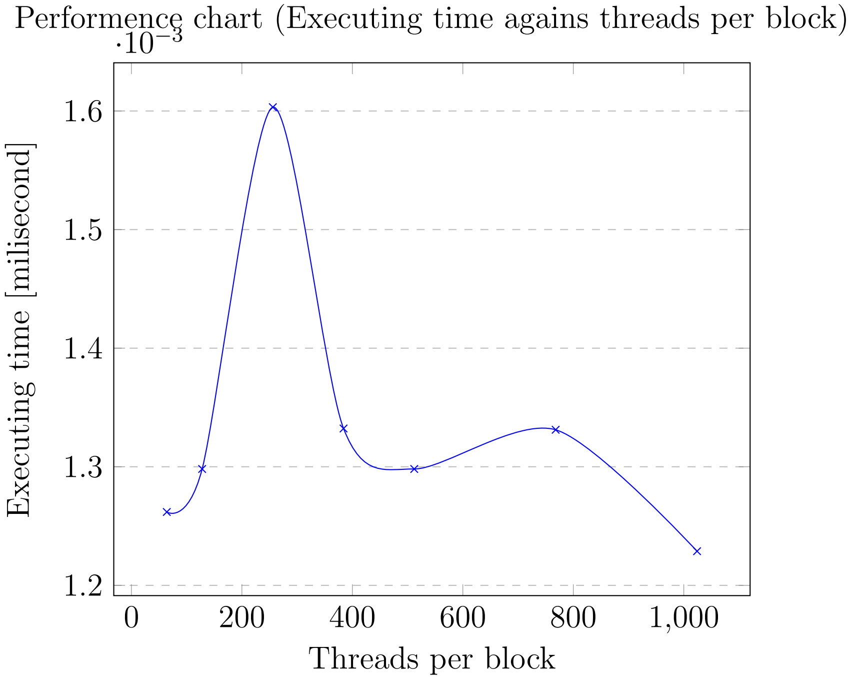


Chapter 2

Graphs and Tables to compare performance differences agains threads per block

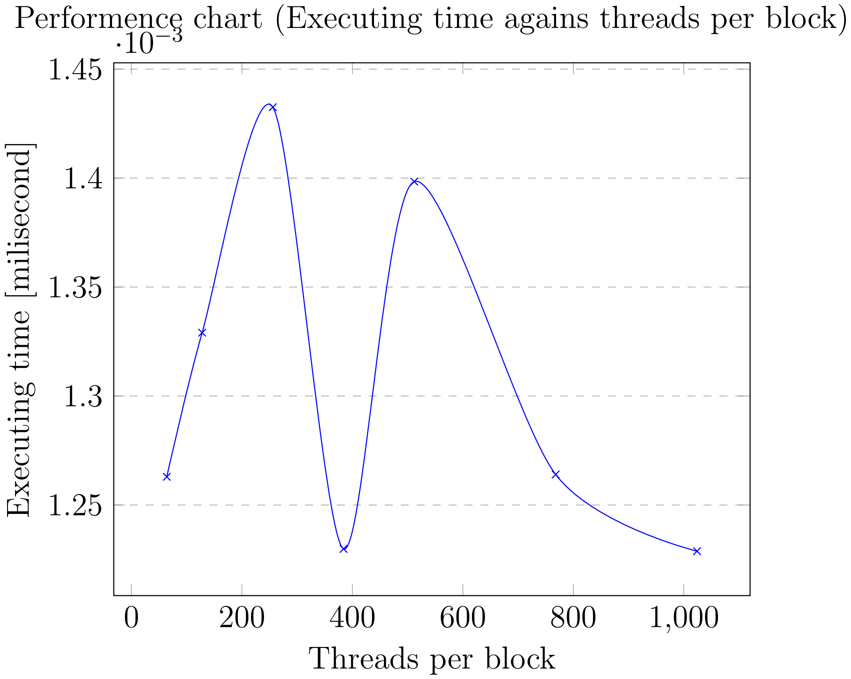
2.1 GPU(Global) performance agains threads per block

Threads per block	Executing time
64	0.00126
128	0.0013
256	0.0016
384	0.00133
512	0.0013
768	0.00133
1,024	0.00123



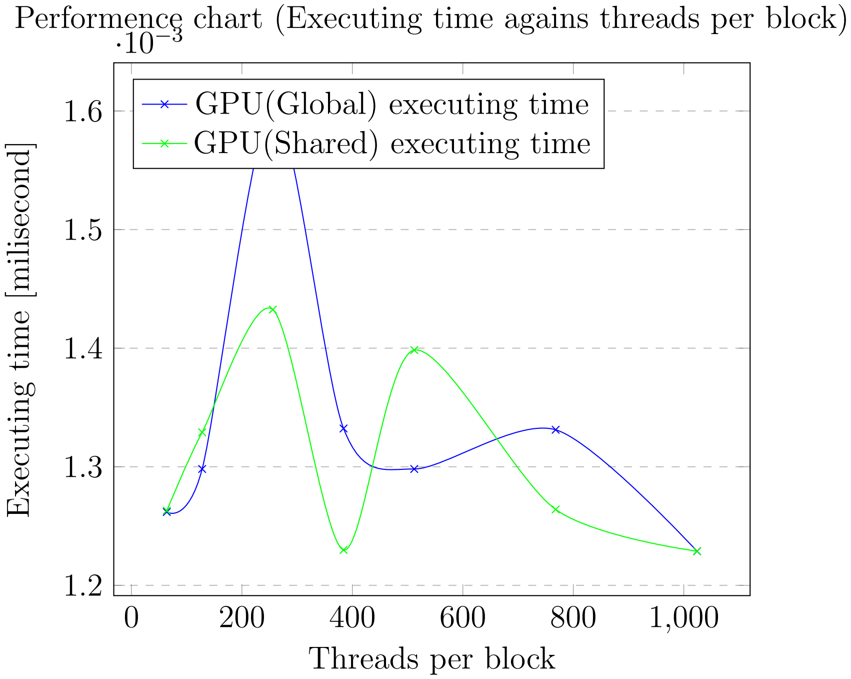
2.2 GPU(Shared) performance agains threads per block

Threads per block	Executing time
64	0.00126
128	0.00133
256	0.00143
384	0.00123
512	0.0014
768	0.00126
1,024	0.00123



2.3 GPU(Global/Shared) performance differences agains threads per block

Threads per block	GPU(Global)	GPU(Shared)
64	0.00126	0.00126
128	0.0013	0.00133
256	0.0016	0.00143
384	0.00133	0.00123
512	0.0013	0.0014
768	0.00133	0.00126
1,024	0.00123	0.00123



Chapter 3

Small descriptions about graphs

3.1 Compare performance differences agains matrix size

The graph and tables represented in "Chapter 1" was calculated matrix multiplication average execute time agains matrix size. As user inputed data, every average execution time was calculated doing 30 iterations with constant block size (for gpu calculation) was 16. And user defined matrix sizes are 16 32 64 128 192 256 320 384 448 512 576 640.

3.2 Compare performance differences agains threads per block

The graph and tables represented in "Chapter 2" was calculated matrix multiplication average execute time agains block size. As user inputed data, every average execution time was calculated doing 30 iterations with constant matrix size was 896. And user defined block sizes are 64 128 256 384 512 768 1024.

Chapter 4

Conclusions

The CPU and GPU are two very different computing devices, and are meant to handle different types of computation. CPUs have fewer cores that can each handle more work per core and while the GPU has thousands of lightweight cores, making them good for smaller computations that need to be repeated often.

According to above graphs, You can performance calculation like matrix maltiplication with less time using GPU base programs, And data decomposition use for this matrix multiplication, so Shared memory is the best method for like these decompositions

Bibliography

[1] Matrix Multiplication in CUDA
https://github.com/lzhengchun/matrix-cuda

[2] ShareLaTex
https://www.sharelatex.com/

[3] UNIX / LINUX Tutorial (TutorialsPoint)
http://www.tutorialspoint.com/unix/