Reduced Matrix Multiplication Report

```
1. Srinjoy Mukherjee(SrNo: 21152)
2. Adesh Adikane(SrNo: 21027)
```

The link for the github repository: Reduced Matrix Multiplication

Part A

In this part of the assignment, we are supposed to implement single threaded and multi-threaded Reduced Matrix Multiplication(RMM) . Naive implementation is given for reference.Improving the performance of the operation can be achieved using many ways. In the previous assignment , we explored improvement by using blocked matrix multiplication and loop interchange.We observed significanr improvement using those techniques. In this assigment we use vector instructions to exploit data parallelism. We make a avx vector if 256 bits/4 integers and do the addition and multiplication operations with the corresponding vectors of the other matrix.We also use loop interchange to utilise locality of reference.We observe significant improvement over naive implementation of RMM.The reason for the improvement is using data parallelism, SIMD instructions, where a single instruction is fetched and decoded but executed on multiple elements.In multi-threaded implementation we use 8 threads . Each thread is responsible for calculating RMM for (N/8) number of rows of resultant matrix where N is the size of resultant matrix.

Observations

We make the graphs of the different counters we used to compare the performance. We compare the performance on the following parameters:

1. Time
2. L1 Read Miss
3. LL Read Miss
4. LL Write Miss
5. TLB Miss
6. Page Faults

Observation Table

IMPLEMENTATION	SIZE	TIME	L1 READ MISS	LL READ MISS	LL WRITE MISS	TLB MISS	PAGE FAULT
REFERENCE	16	0.107	37	0	0	1	0
SINGLE-THREADED	16	0.08	43	0	0	0	0
MULTI-THREADED	16	0.831	1460	64	10	179	16
REFERENCE	4096	986309	30981154829	18278461380	311788	17350628081	0
SINGLE-THREADED	4096	258387	11289280138	5115642215	179371	6813772779	40917
MULTI-THREADED	4096	42577.9	1830	392	91	373	30
REFERENCE	8192	8539590	2.96909E+11	1.33618E+11	1529653	1.39737E+11	0

IMPLEMENTATION	SIZE	TIME	L1 READ MISS	LL READ MISS	LL WRITE MISS	TLB MISS	PAGE FAULT
SINGLE-THREADED	8192	2549420	1.1181E+11	52095504886	3581961	55496798295	32768
MULTI-THREADED	8192	421384	1667	429	79	420	38
REFERENCE	16384	69689700	3.06613E+12	1.85347E+12	8174646	1.12464E+12	8337199
SINGLE-THREADED	16384	23426700	9.64299E+11	5.60175E+11	16979372	4.64353E+11	419786
MULTI-THREADED	16384	3457470	1805	594	108	495	42

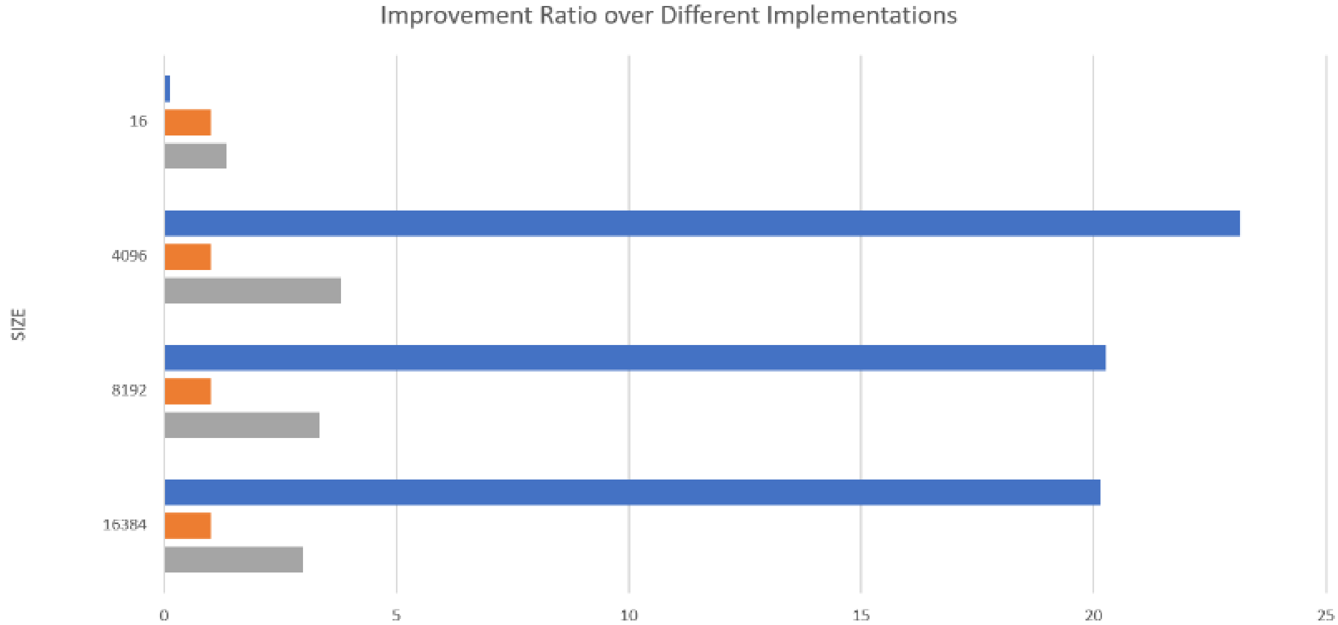
Looking at the tale, we observe that the numbers are too high, so for ease of plotting them on the graph , we simply take the performance improvement ratio. For each parameter it is calculated by: (Reference Value/Improved Value).

Improvement Ratio Table

IMPLEMENTATION	SIZE	TIME
REFERENCE	16	1
SINGLE-THREADED	16	1.3375
MULTI-THREADED	16	0.128
REFERENCE	4096	1
SINGLE-THREADED	4096	3.817
MULTI-THREADED	4096	23.16
REFERENCE	8192	1
SINGLE-THREADED	8192	3.349
MULTI-THREADED	8192	20.265
REFERENCE	16384	1
SINGLE-THREADED	16384	2.9747
MULTI-THREADED	16384	20.15

As we look into the table , we observe that for very small matrix sizes, the performance for naive implementation is better than single and multi-threaded.This is because creating the vector variables from data in matrix takes significant percentage of entire execution time . As we keep increasing the size , the usage of vector instructions becomes more prominent.Among the three implementations, the multi-threaded performed the best as expected. It performs the best beause firstly, it uses loop interchange utilising locality of reference, secondly vector instructions to exploit data parallelism and finally multi-threading which divides the tasks among multiple threads and which can be processed parallelly.

Charts

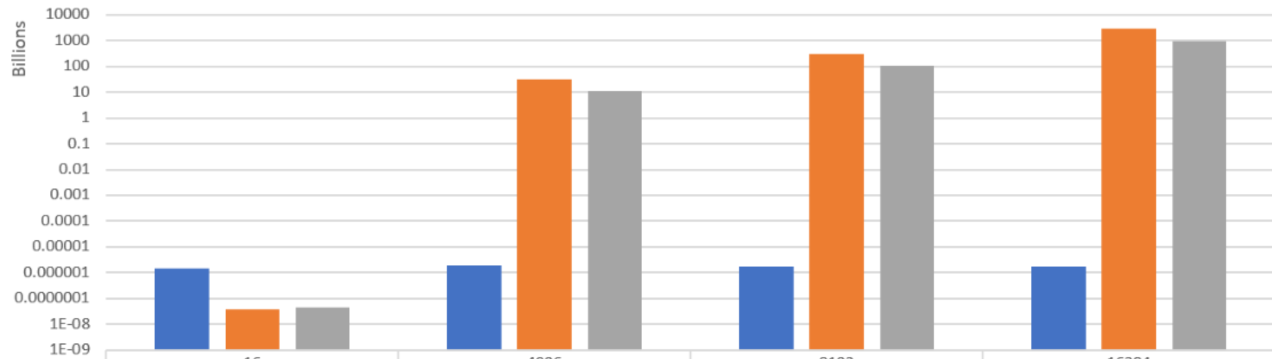


	16384	8192	4096	16
MULTI-THREADED	20.15	20.265	23.16	0.128
REFERENCE	1	1	1	1
SINGLE-THREADED	2.9747	3.349	3.817	1.3375

TIME

MULTI-THREADED REFERENCE SINGLE-THREADED

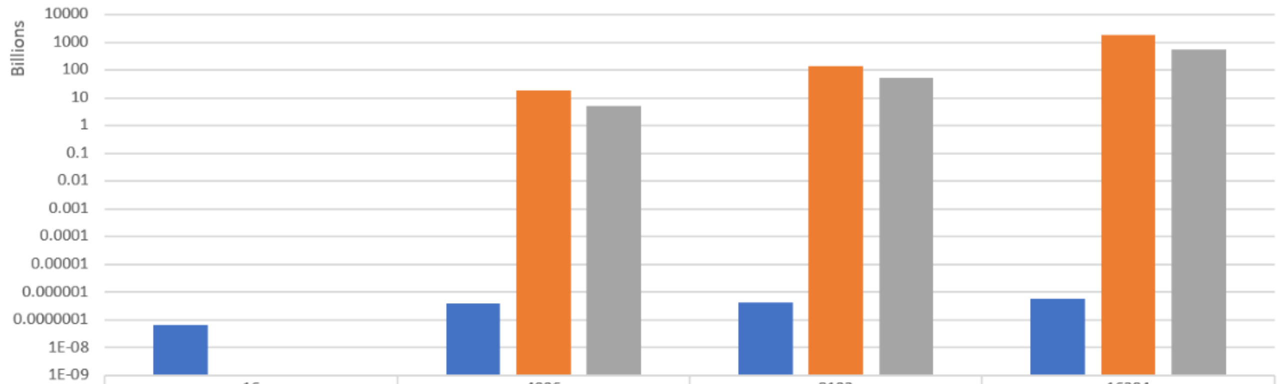
All of these charts are in logarithm scale.



	16	4096	8192	16384
MULTI-THREADED	1460	1830	1667	1805
REFERENCE	37	30981154829	2.96909E+11	3.06613E+12
SINGLE-THREADED	43	11289280138	1.1181E+11	9.64299E+11

IMPLEMENTATION

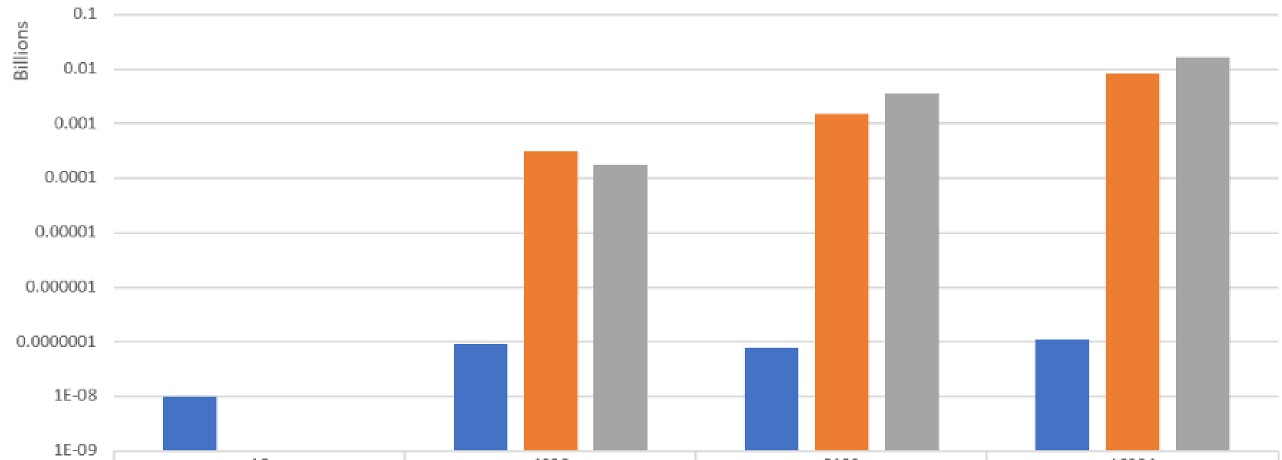
MULTI-THREADED REFERENCE SINGLE-THREADED



	16	4096	8192	16384
MULTI-THREADED	64	392	429	594
REFERENCE	0	18278461380	1.33618E+11	1.85347E+12
SINGLE-THREADED	0	5115642215	52095504886	5.60175E+11

IMPLEMENTATION

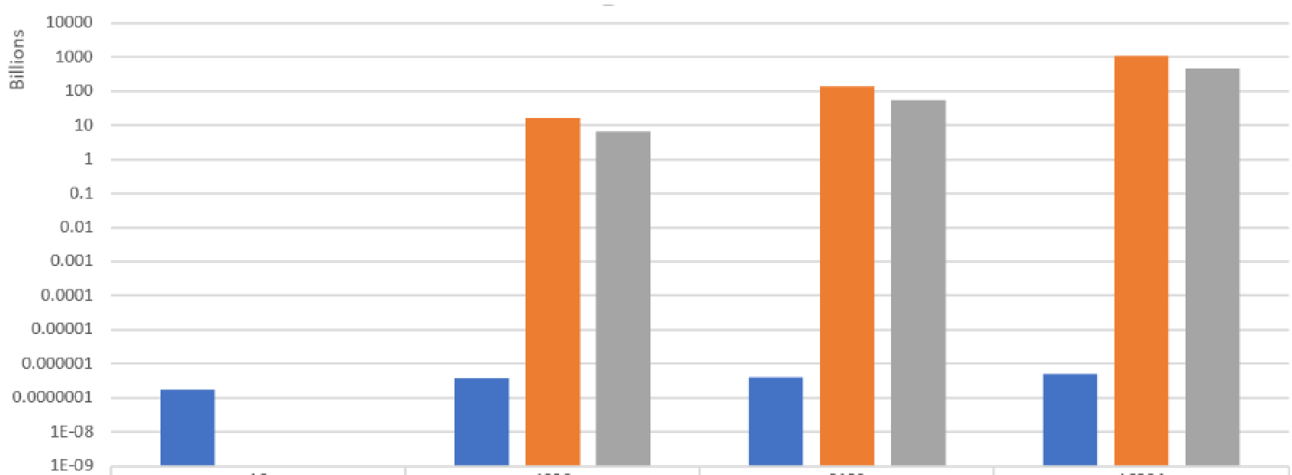
MULTI-THREADED REFERENCE SINGLE-THREADED



	16	4096	8192	16384
MULTI-THREADED	10	91	79	108
REFERENCE	0	311788	1529653	8174646
SINGLE-THREADED	0	179371	3581961	16979372

IMPLEMENTATION

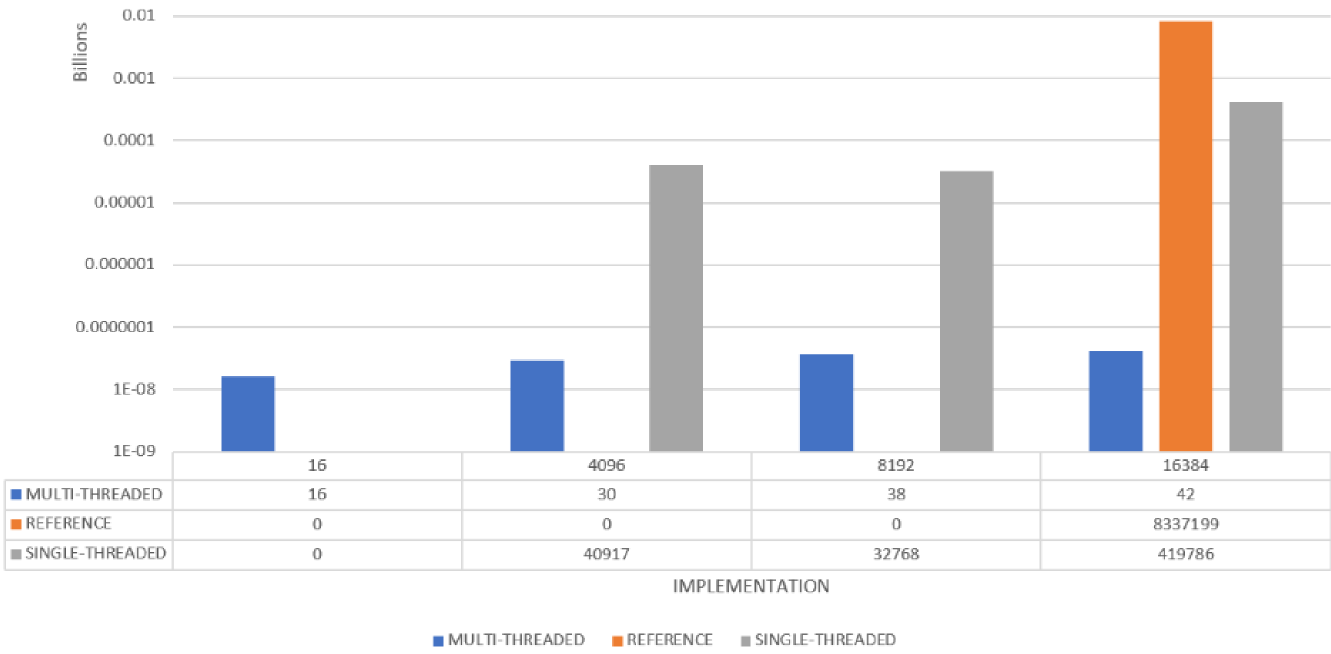
MULTI-THREADED REFERENCE SINGLE-THREADED



	16	4096	8192	16384
MULTI-THREADED	179	373	420	495
REFERENCE	1	17350628081	1.39737E+11	1.12464E+12
SINGLE-THREADED	0	6813772779	55496798295	4.64353E+11

IMPLEMENTATION

MULTI-THREADED REFERENCE SINGLE-THREADED



	16	4096	8192	16384
MULTI-THREADED	16	30	38	42
REFERENCE	0	0	0	8337199
SINGLE-THREADED	0	40917	32768	419786

Part B