Assignment Report

Mukul Sharma
Sr no. 17935

[PartA-I]

Aim : Optimize the given Diagonal matrix multiplication code.
System Configuration : Intel Core i5 10th gen Quad Core
Tools Used : Perf
Optimization Idea : We can observe using the perf command

```
$ perf stat -e L1-dcache-load-misses:u sleep 1
```

The number of L1 cache data load misses is very high.
Using the following command, we can check L1 cache line size, which is 64 Bytes:

```
getconf -a | grep CACHE
```

In the given algorithm, matrix operations are performed on the entire row of large size. The limitation of this approach is that there will be many data cache misses in a single row elements access.

We can change our approach to access matrices in blocks. We can decompose the big matrix into blocks of size the same as the cache line. When we multiply the small matrices in row major order fashion, the row of size 64B would be copied from main memory to cache line. This strategy will cause less cache misses instead of multiplying the whole matrix at once.

In the new approach, i'm accessing the elements of matA in row order and of matB in column order. The given arrays in the problem i.e. output, matA and matB are of type integer. Size of the integer data type is 4 Bytes. If we make the decomposed matrix of size 16x16, then the size of each row would be of 16*4 = 64 Bytes, same as of cache line size.

Using the perf tool, we can verify the above fact. The cache misses in the original code are high. When we decomposed the matrix in blocks, the cache misses decreased drastically.

L1-dcache-load-misses count in original code	**L1-dcache-load-misses count in optimised code**
3,25,21,07,149	2,49,87,75,835

Execution Time and Speedup Table for both unoptimised and optimised code:
Speed up = Execution time of old code / Execution time of new code

Input Size	**Unoptimized Code Runtime (in ms)**	**Optimised Code Runtime (in ms)**	**Speed Up**
n = 4k	236	96	2.45
n = 8k	1044	406	2.57
n = 16k	6735	1992	3.38

Therefore, the speed up of the optimised code is more than two.

[PartA-II] MULTI THREADING

Aim : Implement and optimize multi-threaded DMM (CPU)

Implementation Idea : To each decomposed matrix of the previous single_thread.h implementation, I have assigned one thread to them in round robin manner. The work of a set of such block matrices is given to one thread and like this all the threads are executed in parallel. To avoid synchronisation problems in the output array, every thread maintains an array of size same as

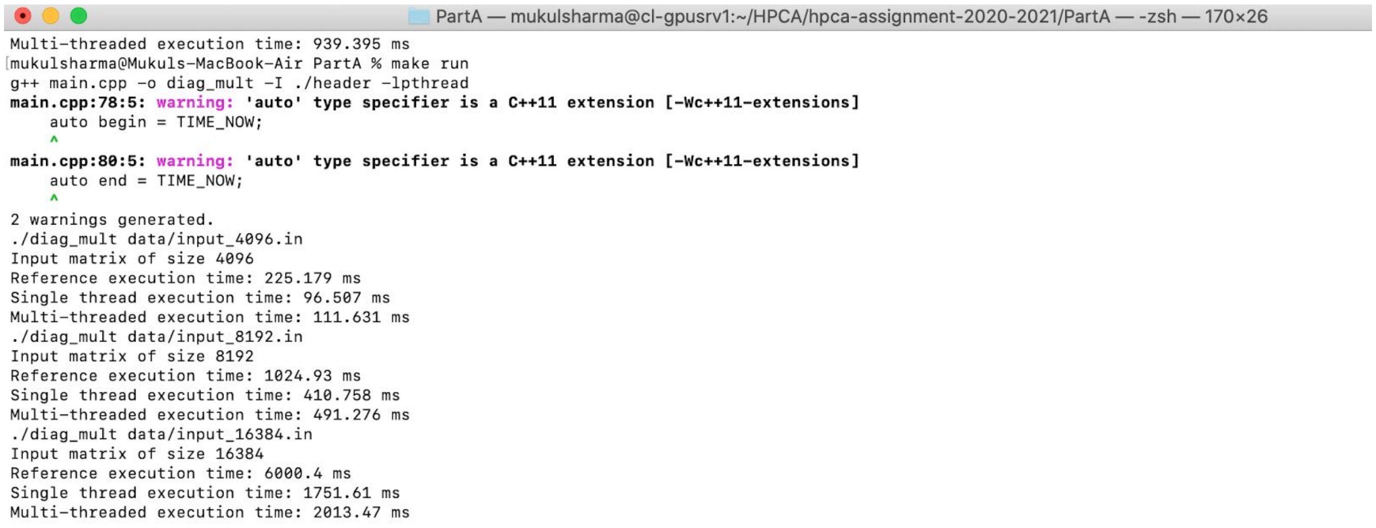
output in which they store their result. After all threads completed their work, all those array values were added in the output array. The execution time using the multithreading decreases by more than half.
Scalability of the implementation: As I increase the number of threads in my implementation of multithreading the execution time decreases by half and speed up increases by a factor of two.

Input Size	**Single Thread Execution**	**No. of Threads = 16**	**No. of Threads = 32**	**No. of Thread = 64**
n = 4k	96	111	60	34
n = 8k	406	491	272	182
n = 16k	1992	2013	1352	813

We can see that , as the number of threads increases from 32 to 64, the execution time decreases by half. This shows the scalability of the implementation.
for n = 16k,
Speed up = Execution time with 32 threads/ Execution time with 64 threads.
= 1352/813 = **1.66** (In ideal case, it can be 2).

Screenshots of the result in next page:

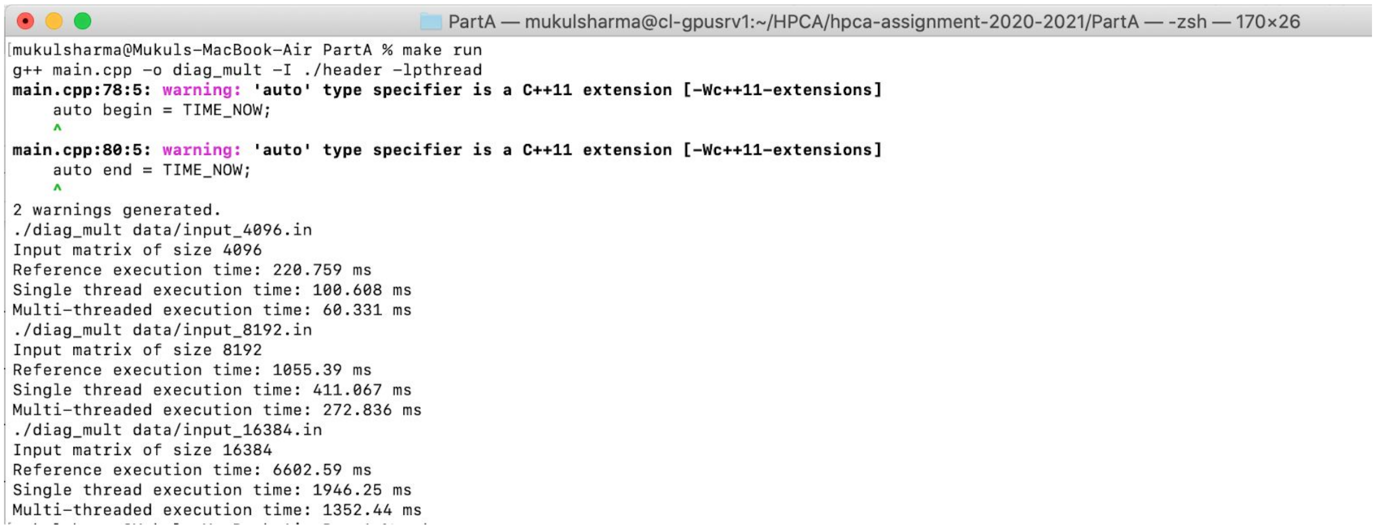
No. of Threads = 16;



PartA — mukulsharma@cl-gpusrv1:~/HPCA/hpca-assignment-2020-2021/PartA — -zsh — 170×26

```
Multi-threaded execution time: 939.395 ms
mukulsharma@Mukuls-MacBook-Air PartA % make run
g++ main.cpp -o diag_mult -I ./header -lpthread
main.cpp:78:5: warning: 'auto' type specifier is a C++11 extension [-Wc++11-extensions]
    auto begin = TIME_NOW;
    ^
main.cpp:80:5: warning: 'auto' type specifier is a C++11 extension [-Wc++11-extensions]
    auto end = TIME_NOW;
    ^
2 warnings generated.
./diag_mult data/input_4096.in
Input matrix of size 4096
Reference execution time: 225.179 ms
Single thread execution time: 96.507 ms
Multi-threaded execution time: 111.631 ms
./diag_mult data/input_8192.in
Input matrix of size 8192
Reference execution time: 1024.93 ms
Single thread execution time: 410.758 ms
Multi-threaded execution time: 491.276 ms
./diag_mult data/input_16384.in
Input matrix of size 16384
Reference execution time: 6000.4 ms
Single thread execution time: 1751.61 ms
Multi-threaded execution time: 2013.47 ms
```

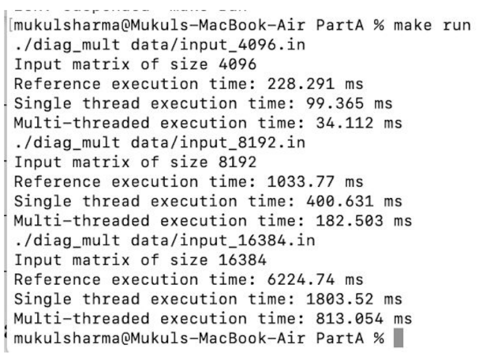
No. of threads = 32;



PartA — mukulsharma@cl-gpusrv1:~/HPCA/hpca-assignment-2020-2021/PartA — -zsh — 170×26

```
mukulsharma@Mukuls-MacBook-Air PartA % make run
g++ main.cpp -o diag_mult -I ./header -lpthread
main.cpp:78:5: warning: 'auto' type specifier is a C++11 extension [-Wc++11-extensions]
    auto begin = TIME_NOW;
    ^
main.cpp:80:5: warning: 'auto' type specifier is a C++11 extension [-Wc++11-extensions]
    auto end = TIME_NOW;
    ^
2 warnings generated.
./diag_mult data/input_4096.in
Input matrix of size 4096
Reference execution time: 220.759 ms
Single thread execution time: 100.608 ms
Multi-threaded execution time: 60.331 ms
./diag_mult data/input_8192.in
Input matrix of size 8192
Reference execution time: 1055.39 ms
Single thread execution time: 411.067 ms
Multi-threaded execution time: 272.836 ms
./diag_mult data/input_16384.in
Input matrix of size 16384
Reference execution time: 6602.59 ms
Single thread execution time: 1946.25 ms
Multi-threaded execution time: 1352.44 ms
```

No. of Threads = 64



```
mukulsharma@Mukuls-MacBook-Air PartA % make run
./diag_mult data/input_4096.in
Input matrix of size 4096
Reference execution time: 228.291 ms
Single thread execution time: 99.365 ms
Multi-threaded execution time: 34.112 ms
./diag_mult data/input_8192.in
Input matrix of size 8192
Reference execution time: 1033.77 ms
Single thread execution time: 400.631 ms
Multi-threaded execution time: 182.503 ms
./diag_mult data/input_16384.in
Input matrix of size 16384
Reference execution time: 6224.74 ms
Single thread execution time: 1803.52 ms
Multi-threaded execution time: 813.054 ms
mukulsharma@Mukuls-MacBook-Air PartA %
```