E0 243: DMM performance optimization

Name: Paras Lohani, S.R.No: 18226

1 Tasks

Optimize the single threaded implementation and write the multi threaded implementation of the diagonal matrix multiplication.

2 System configuration

GPU Server provided by IISc is used to test the changes and for analysing the code.

3 Part A-I

3.1 Analysis

After analysing using **perf** in the server provided, the conclusion was that the cache miss counts are high. To optimize the running time we need to focus on using the whole block of matrix there in the cache before going to the other block (reducing miss count). The command used is written below.

```
perf stat -e cycles:u,instructions:u,branch-misses:u,L1-dcache-load-misses:u,
L1-icache-load-misses:u,dTLB-load-misses:u,dTLB-store-misses:u,iTLB-load-miss
es:u,branch-load-misses:u,L2-load-miss es:u,L2-store-misses:u ./diag_mult dat
a/input_16384.in
```

3.2 Implementation

The processor has a line size of 64 bytes and we are given the matrix of integers which is of 4 bytes. This helps to conclude that the block size will be 64/4 = 16. Therefore the matrix will be divided into blocks of size 16. 16 cache lines with size of 16 integers is replaced in the cache. Therefore when access the elements of matrix such that there are 16*16 hits and then 1 miss. After miss the next block will be added to cache which will be used for next 16 hit.

To decrease the execution time further, the concept of loop unrolling is used where use of loop is reduced and is replaced by separate instructions. It will expose the instruction level parallelism.

3.3 Results

3.3.1 Input size 4096

Reference execution time	=	299.678 ms
Single thread execution time	=	112.052 ms
Speedup	=	2.6744

3.3.2 Input size 8192

Reference execution time	=	1494.35 ms
Single thread execution time	=	480.923 ms
Speedup	=	3.107

3.3.3 Input size 16384

Reference execution time	=	7071.12 ms
Single thread execution time	=	1890.15 ms
Speedup	=	3.741

4 Part A-II

4.1 Analysis

The same concept of single thread implementation will be used as the cache miss is same as of that. Our aim is to reduce the execution time using multiple thread. Assuming there is no limitation on the number of threads to be used.

4.2 Implementation

Here I am using pthread library for creating threads and each thread is given the different blocks created in the matrix. Each thread has separate array where it will save their separate output. These outputs are merged to form the final output. This will handle the problem of synchronization and will also not effect the execution time much. Here the number of thread depends on the matrix size.

4.3 Results

4.3.1 Input size 4096

Reference execution time	=	299.678 ms
Single thread execution time	=	23.861 ms
Speedup	=	12.559

4.3.2 Input size 8192

Reference execution time	=	1494.35 ms
Single thread execution time	=	63.357 ms
Speedup	=	23.586

4.3.3 Input size 16384

Reference execution time	=	7071.12 ms
Single thread execution time	=	203.099 ms
Speedup	=	34.816

5 Figures

```
Input matrix of size 16384
Reference execution time: 8158.52 ms
Single thread execution time: 1993.6 ms
Multi-threaded execution time: 212.287 ms

 Performance counter stats for './diag_mult data/input_16384.in':

     163511368974      cycles:u                                                (44.28%)
     314915168470      instructions:u            #    1.93  insn per cycle     (55.39%)
        233104181      branch-misses:u                                         (55.46%)
       2787876693      L1-dcache-load-misses                                   (55.51%)
          7561951      L1-icache-load-misses                                   (55.61%)
       1080209718      dTLB-load-misses                                        (55.72%)
            67317      iTLB-load-misses                                        (44.54%)
        232708091      branch-load-misses                                      (44.49%)
        164936002      L2-load-misses                                          (44.39%)

     48.177731664 seconds time elapsed
```

Figure 1: Perf Command output

```
g++ main.cpp -o diag_mult -I ./header -lpthread
./diag_mult data/input_4096.in
Input matrix of size 4096
Reference execution time: 299.678 ms
Single thread execution time: 112.052 ms
Multi-threaded execution time: 23.861 ms
./diag_mult data/input_8192.in
Input matrix of size 8192
Reference execution time: 1494.35 ms
Single thread execution time: 480.923 ms
Multi-threaded execution time: 63.357 ms
./diag_mult data/input_16384.in
Input matrix of size 16384
Reference execution time: 7071.12 ms
Single thread execution time: 1890.15 ms
Multi-threaded execution time: 203.099 ms
```

Figure 2: Execution Time Output