Assignment Part A

Part I:

We have run our code for three different values of n, i.e N = 1024, 2048, 4096.

Unoptimised Code:

In the unoptimized code, we got to see with perf how the application is running and utilizing our resources.

N=1024,
Input matrix of size 1024
Reference execution time: 2286.64 ms

Performance counter stats for './cmmref data/input_1024.in':

```
        4,595.58 msec task-clock                #    0.998 CPUs utilized
             276      context-switches          #    0.060 K/sec
               2      cpu-migrations            #    0.000 K/sec
           2,699      page-faults               #    0.587 K/sec
  18,19,48,12,353      cycles                    #    3.959 GHz                      (30.66%)
  27,05,98,09,326      instructions              #    1.49  insn per cycle           (38.39%)
   2,41,19,25,688      branches                  #  524.836 M/sec                    (38.57%)
      32,64,541        branch-misses             #    0.14% of all branches          (38.66%)
  14,34,12,75,416      L1-dcache-loads           # 3120.670 M/sec                    (38.75%)
   1,06,86,79,524      L1-dcache-load-misses     #    7.45% of all L1-dcache accesses  (38.76%)
   1,06,47,11,857      LLC-loads                 #  231.682 M/sec                    (30.85%)
     1,37,09,083       LLC-load-misses           #    1.29% of all LL-cache accesses  (30.76%)
  <not supported>      L1-icache-loads
      12,22,208        L1-icache-load-misses                                       (30.67%)
  14,36,29,23,729      dTLB-loads                # 3125.380 M/sec                    (30.58%)
         68,692        dTLB-load-misses          #    0.00% of all dTLB cache accesses  (30.58%)
         11,508        iTLB-loads                #    0.003 M/sec                    (30.58%)
          7,622        iTLB-load-misses          #   66.23% of all iTLB cache accesses  (30.58%)
  <not supported>      L1-dcache-prefetches
  <not supported>      L1-dcache-prefetch-misses

     4.605288573 seconds time elapsed

     4.588244000 seconds user
     0.008035000 seconds sys
```

N=2048,
Input matrix of size 2048
Reference execution time: 63910.7 ms

Performance counter stats for './cmmref data/input_2048.in':

```
     1,30,442.67 msec task-clock                #    1.000 CPUs utilized
```

```
          612      context-switches          #    0.005 K/sec
           55      cpu-migrations            #    0.000 K/sec
       10,381      page-faults               #    0.080 K/sec
5,18,96,67,80,573      cycles                 #    3.979 GHz                     (30.76%)
2,10,94,09,75,436      instructions           #    0.41  insn per cycle          (38.46%)
 18,15,95,50,754      branches                #  139.215 M/sec                   (38.46%)
    1,45,35,344      branch-misses            #    0.08% of all branches         (38.46%)
1,13,24,77,29,533      L1-dcache-loads         #  868.180 M/sec                   (38.47%)
 13,71,94,19,159      L1-dcache-load-misses    #   12.11% of all L1-dcache accesses  (38.47%)
  8,72,86,22,727      LLC-loads                #   66.915 M/sec                   (30.77%)
  6,56,71,48,703      LLC-load-misses          #   75.24% of all LL-cache accesses  (30.77%)
  <not supported>     L1-icache-loads
    1,48,46,294       L1-icache-load-misses                                      (30.77%)
1,13,42,74,45,364      dTLB-loads              #  869.558 M/sec                   (30.77%)
  8,59,73,28,241      dTLB-load-misses         #    7.58% of all dTLB cache accesses  (30.77%)
       1,26,983       iTLB-loads               #    0.973 K/sec                   (30.76%)
       3,79,312       iTLB-load-misses         #  298.71% of all iTLB cache accesses  (30.76%)
  <not supported>     L1-dcache-prefetches
  <not supported>     L1-dcache-prefetch-misses

   130.449578975 seconds time elapsed

   130.415606000 seconds user
     0.027999000 seconds sys
```

N=4096,

```
Input matrix of size 4096
Reference execution time: 574311 ms

 Performance counter stats for './cmmref data/input_4096.in':

    11,44,275.46 msec task-clock              #    1.000 CPUs utilized
          5,501      context-switches         #    0.005 K/sec
            256      cpu-migrations           #    0.000 K/sec
         41,099      page-faults              #    0.036 K/sec
45,51,06,61,98,212      cycles                #    3.977 GHz                     (30.77%)
16,69,80,17,01,832      instructions          #    0.37  insn per cycle          (38.46%)
 1,41,65,87,77,199      branches              #  123.798 M/sec                   (38.46%)
     6,74,06,734      branch-misses           #    0.05% of all branches         (38.46%)
 9,00,31,21,08,364      L1-dcache-loads         #  786.797 M/sec                   (38.46%)
 1,46,55,09,92,383      L1-dcache-load-misses    #   16.28% of all L1-dcache accesses  (38.46%)
  70,79,74,25,908      LLC-loads                #   61.871 M/sec                   (30.77%)
  69,61,31,98,368      LLC-load-misses          #   98.33% of all LL-cache accesses  (30.77%)
  <not supported>     L1-icache-loads
    12,24,46,694      L1-icache-load-misses                                      (30.77%)
 8,99,96,96,86,505      dTLB-loads              #  786.497 M/sec                   (30.77%)
  68,74,87,23,435      dTLB-load-misses         #    7.64% of all dTLB cache accesses  (30.77%)
       15,63,069       iTLB-loads               #    0.001 M/sec                   (30.77%)
       28,13,848       iTLB-load-misses         #  180.02% of all iTLB cache accesses  (30.77%)
  <not supported>     L1-dcache-prefetches
  <not supported>     L1-dcache-prefetch-misses
```

1144.320582699 seconds time elapsed

1144.189403000 seconds user
0.087996000 seconds sys

Basically we are observing that the L1- Data cache misses are high (7.45%, 12.11%, 16.28%). It is basically due to the access of different blocks of elements in the matrix for the unoptimised code. The unoptimised code is accessing the matrix element that are not currently on the L1 Data cache. Also we can clearly observe that the application takes most of the time in its implementation of checkered matrix multiplication.
Due to the Amdahl's Law, we come to know that if we apply optimisation on the matrix multiplication part the speedup is guaranteed.
Now we apply our optimised code and run the code.

Optimised Code:

Single Thread:
In the single thread, we basically have to improve the algorithm of the unoptimised code. So, in my optimisation, I have implemented the code by calculating the element of the new matrix serially and used the contiguous matrix elements of A and B to calculate so that the D cache misses are less. As the code uses the 1st row of A and 1st columnset of B. Here we see that if we implement B in column major, we have huge advantage because the code will use contiguous access of B.

So in single thread, I have converted B into column major order and calculated the elements of C serially and thus optimising our code.

N=1024,
Input matrix of size 1024
Reference execution time: 0 ms
Single thread execution time: 1332.31 ms

Performance counter stats for './cmmsingle data/input_1024.in':

```
      3,403.17 msec task-clock              #    1.000 CPUs utilized
            12      context-switches        #    0.004 K/sec
             1      cpu-migrations          #    0.000 K/sec
         3,212      page-faults             #    0.944 K/sec
13,54,38,13,577      cycles                  #    3.980 GHz                      (30.67%)
31,58,11,89,631      instructions            #    2.33  insn per cycle           (38.43%)
 3,20,17,15,356      branches                #  940.804 M/sec                    (38.55%)
    25,24,699        branch-misses           #    0.08% of all branches          (38.66%)
15,96,91,39,609      L1-dcache-loads         # 4692.431 M/sec                    (38.78%)
  60,54,54,718       L1-dcache-load-misses   #    3.79% of all L1-dcache accesses  (38.77%)
  53,85,95,687       LLC-loads               #  158.263 M/sec                    (30.89%)
     6,51,034        LLC-load-misses         #    0.12% of all LL-cache accesses  (30.78%)
<not supported>      L1-icache-loads
     8,87,947        L1-icache-load-misses                                      (30.66%)
15,94,19,60,772      dTLB-loads              # 4684.445 M/sec                    (30.56%)
       12,447        dTLB-load-misses        #    0.00% of all dTLB cache accesses  (30.56%)
        7,320        iTLB-loads              #    0.002 M/sec                    (30.56%)
        4,574        iTLB-load-misses        #   62.49% of all iTLB cache accesses  (30.56%)
```

<not supported> L1-dcache-prefetches
<not supported> L1-dcache-prefetch-misses

3.403526695 seconds time elapsed

3.399540000 seconds user
0.003999000 seconds sys

N=2048,

Input matrix of size 2048
Reference execution time: 0 ms
Single thread execution time: 11610.9 ms

Performance counter stats for './cmmsingle data/input_2048.in':

78,660.82 msec task-clock # 1.000 CPUs utilized
1,243 context-switches # 0.016 K/sec
32 cpu-migrations # 0.000 K/sec
12,428 page-faults # 0.158 K/sec
3,11,15,14,42,360 cycles # 3.956 GHz (30.77%)
2,47,48,04,48,755 instructions # 0.80 insn per cycle (38.46%)
24,61,80,98,786 branches # 312.965 M/sec (38.46%)
1,12,69,958 branch-misses # 0.05% of all branches (38.46%)
1,26,08,72,69,103 L1-dcache-loads # 1602.923 M/sec (38.46%)
7,41,79,02,528 L1-dcache-load-misses # 5.88% of all L1-dcache accesses (38.46%)
4,37,48,65,564 LLC-loads # 55.617 M/sec (30.77%)
3,30,11,39,766 LLC-load-misses # 75.46% of all LL-cache accesses (30.77%)
<not supported> L1-icache-loads
1,28,83,895 L1-icache-load-misses (30.77%)
1,26,27,61,08,045 dTLB-loads # 1605.324 M/sec (30.77%)
4,30,51,07,034 dTLB-load-misses # 3.41% of all dTLB cache accesses (30.77%)
1,64,268 iTLB-loads # 0.002 M/sec (30.76%)
2,25,261 iTLB-load-misses # 137.13% of all iTLB cache accesses (30.76%)
<not supported> L1-dcache-prefetches
<not supported> L1-dcache-prefetch-misses

78.692315373 seconds time elapsed

78.617746000 seconds user
0.043989000 seconds sys

N=4096,

Input matrix of size 4096
Reference execution time: 0 ms
Single thread execution time: 93343.7 ms

Performance counter stats for './cmmsingle data/input_4096.in':

95,034.11 msec task-clock # 1.000 CPUs utilized
503 context-switches # 0.005 K/sec
94 cpu-migrations # 0.001 K/sec
41,100 page-faults # 0.432 K/sec
3,64,44,44,38,543 cycles # 3.835 GHz (30.76%)

```
11,34,75,89,79,815      instructions              #    3.11  insn per cycle          (38.46%)
 1,24,08,31,34,633      branches                  # 1305.669 M/sec                    (38.46%)
      2,51,16,153       branch-misses             #    0.02% of all branches         (38.46%)
 5,55,97,63,41,051      L1-dcache-loads           # 5850.282 M/sec                    (38.47%)
   6,63,75,64,728       L1-dcache-load-misses     #    1.19% of all L1-dcache accesses  (38.47%)
      8,93,24,585       LLC-loads                 #    0.940 M/sec                    (30.77%)
      7,48,11,698       LLC-load-misses           #   83.75% of all LL-cache accesses  (30.77%)
  <not supported>       L1-icache-loads
      2,06,78,699       L1-icache-load-misses                                        (30.77%)
 5,55,88,71,99,791      dTLB-loads                # 5849.344 M/sec                    (30.77%)
      7,62,14,693       dTLB-load-misses          #    0.01% of all dTLB cache accesses  (30.77%)
        4,90,406        iTLB-loads                #    0.005 M/sec                    (30.76%)
        3,45,595        iTLB-load-misses          #   70.47% of all iTLB cache accesses  (30.76%)
  <not supported>       L1-dcache-prefetches
  <not supported>       L1-dcache-prefetch-misses

     95.038998524 seconds time elapsed

     94.963109000 seconds user
      0.071999000 seconds sys
```

Multi-thread:

In the multi-thread implementation, I have basically used the implementation of single thread and divided the rows into some threads and calculated the matrix multiplication parallelly.
In the testing, we can see that for no of threads increasing from 1 to 2, the speedup is likely to be nearly 2x times. And if we make it 4, the speedup is almost 4x times for ideal but here close to 3. So in this way, we can show the scalability of our implementation.

N=1024,

```
Input matrix of size 1024
Reference execution time: 0 ms
Multi-threaded execution time: 353.456 ms

 Performance counter stats for './cmmmulti data/input_1024.in':

         3,409.69 msec task-clock                #    1.447 CPUs utilized
               15      context-switches          #    0.004 K/sec
                0      cpu-migrations            #    0.000 K/sec
            3,223      page-faults               #    0.945 K/sec
   13,23,97,48,137      cycles                    #    3.883 GHz                      (31.46%)
   31,39,48,17,133      instructions              #    2.37  insn per cycle           (39.23%)
    3,19,97,26,565      branches                  #  938.422 M/sec                    (39.16%)
       25,32,269        branch-misses             #    0.08% of all branches         (38.96%)
   15,90,49,26,072      L1-dcache-loads           # 4664.626 M/sec                    (38.61%)
      61,85,87,509      L1-dcache-load-misses     #    3.89% of all L1-dcache accesses  (38.15%)
      58,18,50,079      LLC-loads                 #  170.646 M/sec                    (29.96%)
       1,53,815         LLC-load-misses           #    0.03% of all LL-cache accesses  (30.14%)
  <not supported>       L1-icache-loads
      10,54,519         L1-icache-load-misses                                        (31.54%)
   15,46,11,14,058      dTLB-loads                # 4534.464 M/sec                    (31.75%)
          36,496        dTLB-load-misses          #    0.00% of all dTLB cache accesses  (31.81%)
           6,854        iTLB-loads                #    0.002 M/sec                    (31.64%)
```

```
         6,721      iTLB-load-misses          #   98.06% of all iTLB cache accesses  (31.49%)
   <not supported>      L1-dcache-prefetches
   <not supported>      L1-dcache-prefetch-misses

       2.356414789 seconds time elapsed

       3.390231000 seconds user
       0.019989000 seconds sys
```

N=2048,

```
Input matrix of size 2048
Reference execution time: 0 ms
Multi-threaded execution time: 3697.09 ms

 Performance counter stats for './cmmmulti data/input_2048.in':

        78,760.88 msec task-clock                #    1.158 CPUs utilized
           4,225      context-switches          #    0.054 K/sec
              41      cpu-migrations            #    0.001 K/sec
          12,440      page-faults               #    0.158 K/sec
 2,99,57,20,28,964      cycles                    #    3.804 GHz                      (30.81%)
 2,47,67,49,79,479      instructions              #    0.83  insn per cycle           (38.50%)
  24,69,31,44,278      branches                  #  313.520 M/sec                    (38.51%)
     1,14,23,995      branch-misses             #    0.05% of all branches          (38.49%)
 1,26,37,97,49,111      L1-dcache-loads           # 1604.601 M/sec                    (38.50%)
   7,41,69,35,545      L1-dcache-load-misses     #    5.87% of all L1-dcache accesses  (38.50%)
   4,34,57,46,691      LLC-loads                 #   55.176 M/sec                    (30.78%)
   2,77,84,61,799      LLC-load-misses           #   63.94% of all LL-cache accesses  (30.76%)
   <not supported>      L1-icache-loads
     1,63,02,304      L1-icache-load-misses                                         (30.76%)
 1,25,52,57,24,215      dTLB-loads                # 1593.757 M/sec                    (30.75%)
   4,30,87,45,277      dTLB-load-misses          #    3.43% of all dTLB cache accesses  (30.75%)
        1,37,471      iTLB-loads                #    0.002 M/sec                    (30.77%)
        2,83,741      iTLB-load-misses          #  206.40% of all iTLB cache accesses  (30.80%)
   <not supported>      L1-dcache-prefetches
   <not supported>      L1-dcache-prefetch-misses

      67.988130952 seconds time elapsed

      78.680092000 seconds user
       0.084004000 seconds sys
```

N=4096,

```
Input matrix of size 4096
Reference execution time: 0 ms
Multi-threaded execution time: 30505.6 ms

 Performance counter stats for './cmmsingle data/input_4096.in':

     1,20,454.29 msec task-clock                #    3.740 CPUs utilized
           2,021      context-switches          #    0.017 K/sec
             131      cpu-migrations            #    0.001 K/sec
          41,109      page-faults               #    0.341 K/sec
```

```
 3,85,44,97,32,089      cycles                    #    3.200 GHz                    (30.79%)
11,34,81,01,68,136      instructions              #    2.94  insn per cycle         (38.49%)
 1,24,11,79,91,978      branches                  # 1030.416 M/sec                  (38.48%)
      2,75,08,455       branch-misses             #    0.02% of all branches        (38.47%)
 5,56,22,24,20,375      L1-dcache-loads           # 4617.705 M/sec                  (38.46%)
    6,75,98,86,320      L1-dcache-load-misses     #    1.22% of all L1-dcache accesses  (38.45%)
      8,57,63,385       LLC-loads                 #    0.712 M/sec                  (30.75%)
      6,40,38,576       LLC-load-misses           #   74.67% of all LL-cache accesses  (30.77%)
   <not supported>      L1-icache-loads
      3,52,10,479       L1-icache-load-misses                                      (30.77%)
 5,54,78,96,79,265      dTLB-loads                # 4605.811 M/sec                  (30.79%)
      6,85,03,870       dTLB-load-misses          #    0.01% of all dTLB cache accesses  (30.79%)
        7,76,594        iTLB-loads                #    0.006 M/sec                  (30.79%)
        4,74,956        iTLB-load-misses          #   61.16% of all iTLB cache accesses  (30.80%)
   <not supported>      L1-dcache-prefetches
   <not supported>      L1-dcache-prefetch-misses

      32.209103747 seconds time elapsed

     120.316759000 seconds user
       0.139972000 seconds sys
```

Conclusion:

So the speedup obtained in single threaded is: ~5.5

And speedup obtained in multi threaded of 4 threads: ~17.28 (From single to multi speed up is 3.14)