HW 1

Chen-Tung Chu   cc2396

Matrix-matrix multiplication:

1. An elementary "row-by-column" approach

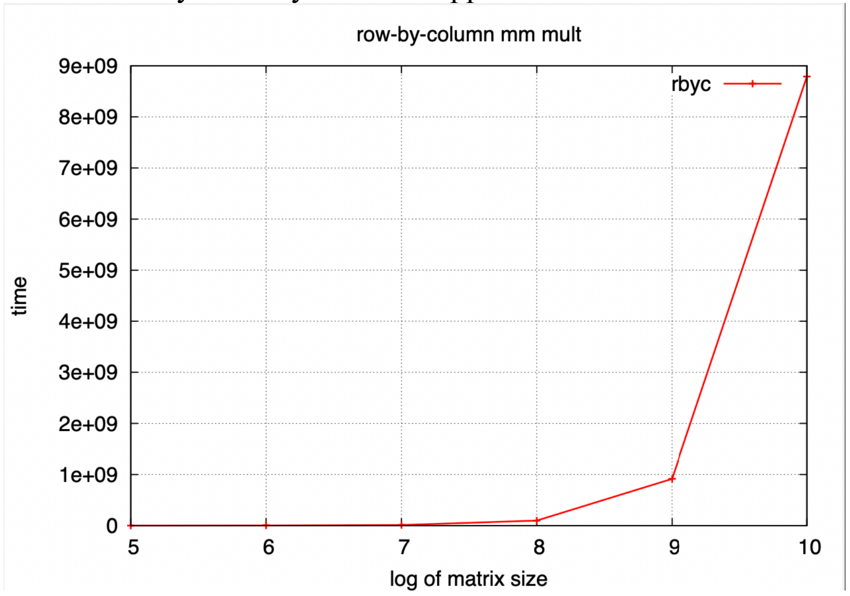

Figure 1. Elementary row-by-column approach

From the result graph we can find that when we doubled the size of the matrix, it meets the requirement of O(n^3) operations for an $n \times n$ matrix. However, it does not always require O(n^3) operations, for example, when the log size of matrix is 9, the calculation time is approximately 1e+09; and when the log size of the matrix is 10, the calculation time is approximately 8.8e+09, which is roughly 8.8 times increase.

2. A blocked approach

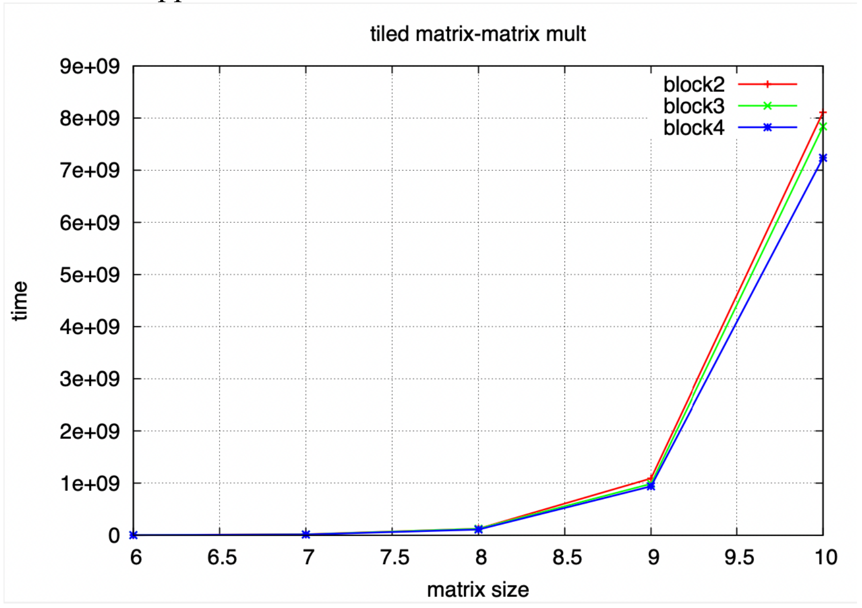

Figure 2. Blocked matrix size with N = 10

For now, by only comparing the time difference between Figure 1 and Figure 2, we can observe that a simply row-by-column multiplication is faster than dividing blocks multiplication before the log size of 9. However, as we divided the matrix into blocks for multiplication, as the sequential blocked matrix-matrix multiplication number grows up, we can see that the computing time is starting to go less, which is as expected. If we divide the whole single matrix into numbers of blocks, we can surely decrease the computing time complexity.

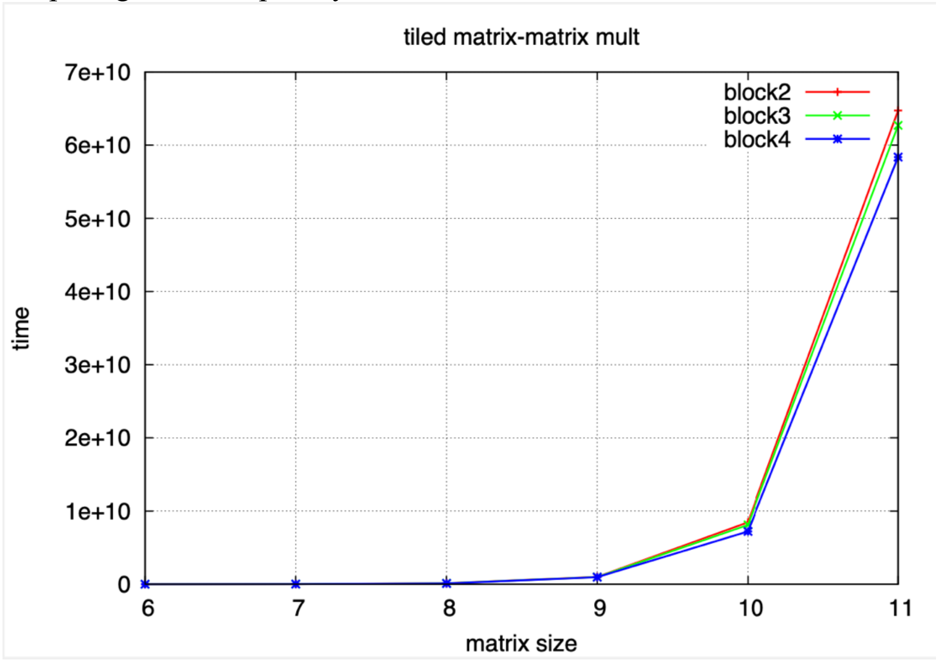

Figure 3. Blocked matrix size with N = 11

However, Figure 3 shows that as we increase the block size of the matrix, the computing time does not decrease as we copulated, it increases quite significantly.

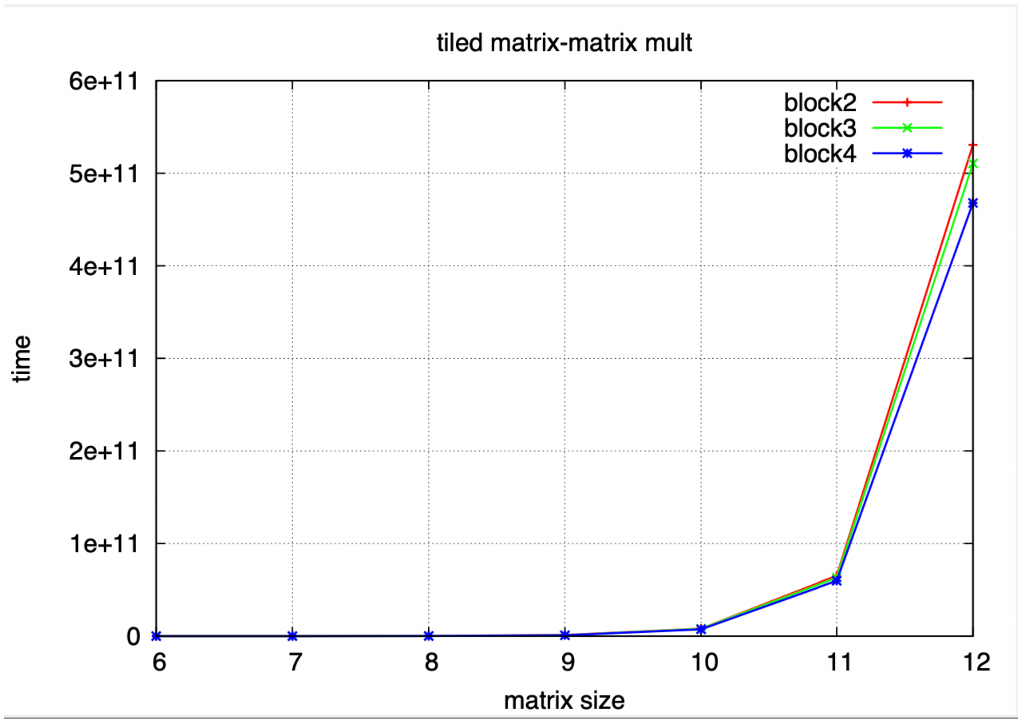

Figure 4. Blocked matrix size with N = 12

When we increase the size of N to 12, it is more obvious that the operating time is way more than a single matrix-matrix multiplication.

Discussion:

1. Matrix-matrix multiplication requires O(n^3) floating point operations for an $n \times n$ matrix. Does the timing increase 8-fold when the matrix size is doubled?

- At some point, the matrix-matrix multiplication does follow O(n^3) time complexity, but at some points as the log N size goes bigger, it may differ from the expected value, whether larger or smaller.

2. How does the performance of the sequential row-by-column compare to the sequential blocked matrix-matrix multiplication?

- Ideally, the sequential blocked matrix-matrix multiplication is faster than row-by-column multiplication, but as the size of the blocked matrix increases, the performance does not as we expected.

3. How does the performance vary with different dimensions of tiles? What is the speed-up (or slow down) of the blocked version over the row by column version?

- When the tile size is small, the average multiplication time is smaller than the regular row-by-column multiplication, we can discuss the performance as follows:
  - Row-by-column:
    - Load b(:, j), j = 1, …, n, n times each for total n^3 loads
    - Load a(i, :), i = 1, …, n, n one time each for total n^2 loads
    - Load and store c(i, j) once for total $2n^2$ slow memory ops

        - # slow memory ops $K = n^3 + 3n^2$, #flops $N = 2n^3$
        - $q = \frac{N}{K} \approx 2$
    - Blocked matrix:
        - Load A(i, k) and B(k, j) p^3 time for total of $2p^3b^2 = 2pn^2$ loads
        - Load and store each C(i, j) once for total of $2n^2$ memory ops
        - # memory ops is $K = 2n^2(p+1)$, # flops $N = 2n^3$
        - $q = \frac{N}{K} = \frac{2n^3}{(2p+2)n^2} \approx \frac{n}{p} = b$

4. Normalize your measurements by n^3 to see how times per a single flop change when the dimensions change.

- From my rbyc.csv, I made up a table for my performance on each size of array:

Array size	2^5	2^6	2^7	2^8	2^9	2^{10}
time	1.652+E05	1.296E+06	1.091E+07	8.758E+07	9.088E+08	7.393E+09

For each normalization with my array size to n^3 I can see how many times per a single flop difference, below is the normalization table of single flops:

N^3 array	2^{15}	2^{18}	2^{21}	2^{24}	2^{27}	2^{30}
Times per flop	5.041	4.944	5.202	5.220	6.771	6.885

Reference:
1. https://penny-xu.github.io/blog/tiled-matrix-multiplication
2. https://www.cs.cornell.edu/~bindel/class/cs5220-s10/slides/lec03.pdf