OS2 CS3523

Name: Swapnil Bag

Roll no: ES22BTECH11034

---

**Aim**: To develop a multithreaded program to calculate the square of the matrix using pthread implementation.

## Low level design

- First the input matrix and output matrix (C_chunk and C_mixed to store answer) is globally defined to have a max size of 4096
- A structure th_param is created that stores the matrix size N, the thread value i (0,1,2….) the chunk size p and the max number of threads K
- In main function value of N,K and matrix is read through command line argument
- Worker thread arrays are created of size K (threads_1 for chunk and threads_2 for mixed, thread_3 for diagonal methods)
- For computing the matrix using mixed and chunk method, a for loop is initially run which creates K threads using pthread_create and passing parameters as the address of worker threads, NULL default attribute, function name and th_param structure that gets dereferenced in the called function. A second for loop is run for the resulting threads and consolidate results

- **Functions:**
    a) ***Square_chunk*** : chunk method is used. It is a void function that accepts a void type argument. The argument is typecasted back into th_param structure type. It contains N,K and thread number (i) value from which the start and end index of rows that have to be computed are determined. *Thread_i* populates *row i*p +1 to (i+1)*p* and finally gets stored in C_chunk matrix

    b) ***square_mixed***: It has same function signature as square_chunk. thread_0 is responsible for rows 0, k,2k… thread_1 1,k+1,2k+1….. in steps of k. Final output stored in C_mixed matrix

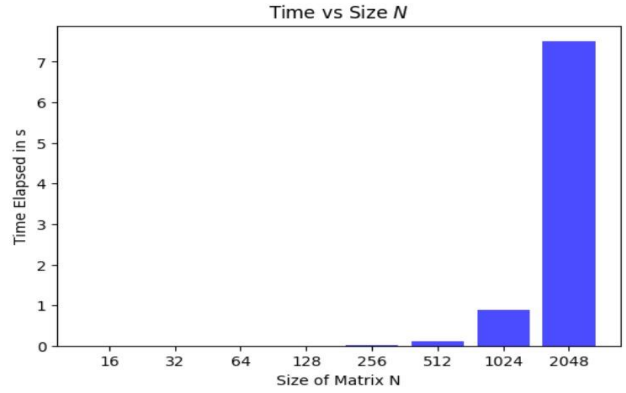
    c) ***square_diagonal***: Same function signature as above. This method splits the work among threads by assigning it to a diagonal identified by (id = x_coord-y_coord ) and computes from topleft to bottomright. A given threads selects a starting diagonal and increments in steps of K (similar to mixed method but here works on diagonals). id value ranges from N-1 to –(N-1) and totally there are 2N-1 diagonals. ith thread starts from diagonal id N-1-i to –(N-1) in steps of K. If id is positive (x_coord > y_coord) it means it's in lower triangular half and if negative its in upper triangular half. Based on this way the diagonals of matrix is computed by a given thread.(refer to code for better clarity)

**Input**: N, K and the matrix A using an input file

**Output**: out.txt, it contains time taken by 3 methods and resulting matrix computed by these 3 methods

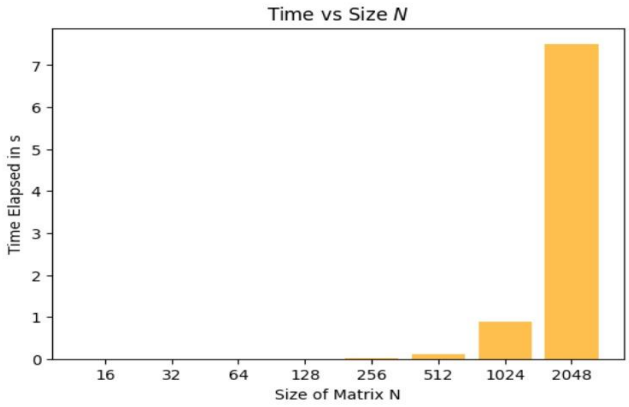
## Time vs. Size, N:

Chunk method



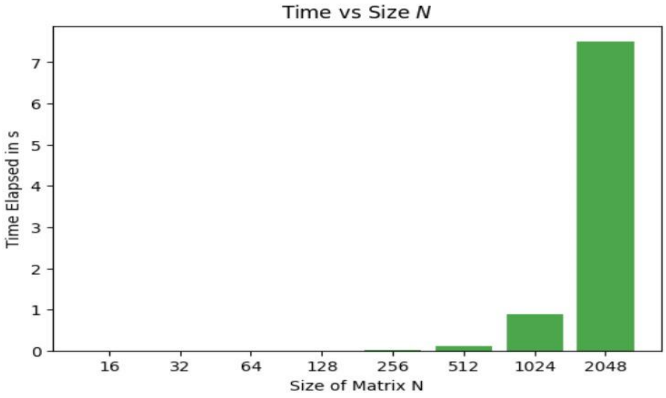

| Size N | Time (s) |
|---|---|
| 16 | 0.00049 |
| 32 | 0.00056 |
| 64 | 0.00087 |
| 128 | 0.00192 |
| 256 | 0.01162 |
| 512 | 0.10272 |
| 1024 | 0.89380 |
| 2048 | 7.51000 |

Mixed method




| Size N | Time (s) |
|---|---|
| 16 | 0.00046 |
| 32 | 0.00051 |
| 64 | 0.00091 |
| 128 | 0.00186 |
| 256 | 0.01509 |
| 512 | 0.09230 |
| 1024 | 0.94290 |
| 2048 | 7.93000 |

Diagonal method

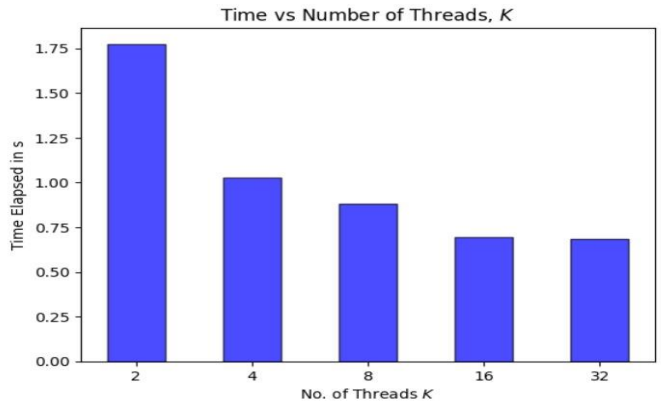



| Size N | Time (s) |
|---|---|
| 16 | 0.000370 |
| 32 | 0.000523 |
| 64 | 0.000638 |
| 128 | 0.002060 |
| 256 | 0.011700 |
| 512 | 0.090700 |
| 1024 | 0.812900 |
| 2048 | 7.420000 |

Here all 3 methods have the profile. As expected, as the size of N increases the time taken increases. For N=2048 it takes around 7s. The lower values for N= 16.. cannot be seen as it is extremely small compared to 7s . From the tables the diagonal method is slightly faster than chunks and mixed methods as each thread handles a specific diagonal. However, the draw back of Diagonal method is that if N is comparable to K then there is unequal distribution of work in the threads and the initial threads does less work. Do note the time taken highly depends the number of cores on the machine and how OS schedules the threads.

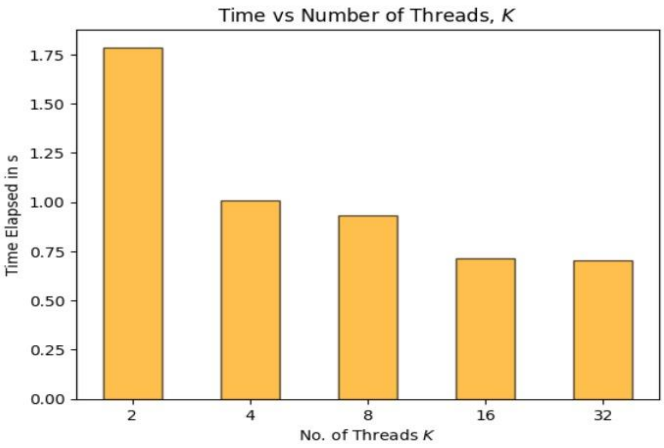
## Time vs. Number of threads, K:

Chunk method



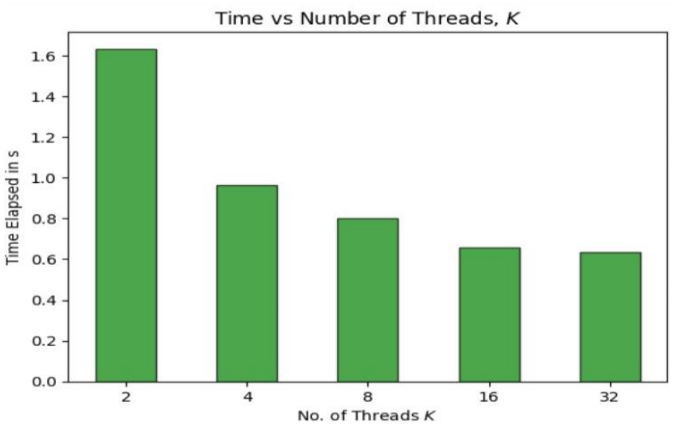

| No.of Threads | Time (s) |
|---|---|
| 2 | 1.775 |
| 4 | 1.025 |
| 8 | 0.881 |
| 16 | 0.694 |
| 32 | 0.682 |

Mixed method




| No.of Threads | Time (s) |
|---|---|
| 2 | 1.7862 |
| 4 | 1.0070 |
| 8 | 0.9300 |
| 16 | 0.7160 |
| 32 | 0.7032 |

Diagonal Method




| No.of Threads | Time (s) |
|---|---|
| 2 | 1.6340 |
| 4 | 0.9644 |
| 8 | 0.8000 |
| 16 | 0.6560 |
| 32 | 0.6332 |

Here the 3 graphs have a similar profile. As the value of K increases for constant N, the time taken reduces as more threads perform the calculations parallelly. Here again the diagonal method is slightly faster than the rest. In certain cases, if the number of cores is less than the number of threads then the time increases as there is overhead incurred in context switches. Do note the time taken highly depends the number of cores on the machine and how OS schedules the threads.