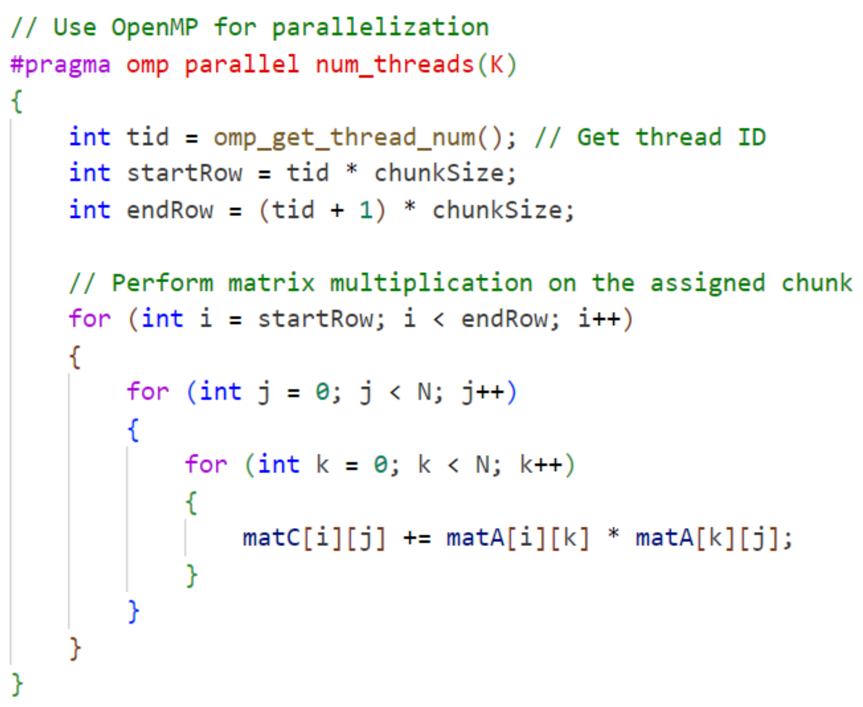
OS II - Assignment 1 - Report (cs23mtech11026)

- I have used the OpenMP library for parallelization.
- I used a Python code to generate the random matrix between 1 and 10 numbers. That generated random matrix is stored in inp.txt format.
- Further, add N and K manually in that inp.txt as required.
- The output of the matrix multiplication is stored in out.txt (file has the resultant matrix and time take.)
- I have zipped following files
    1. Code for both methods
    2. Python code for generating random matrix
    3. Sample inp.txt and out.txt
    4. This Report

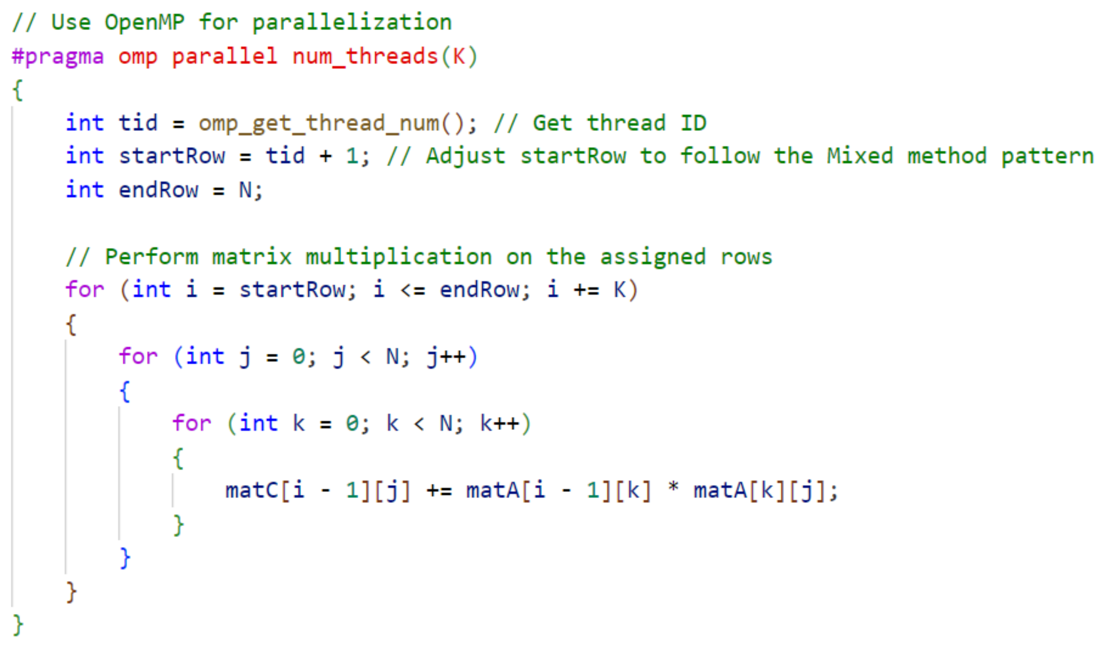
1. **Implementation of the chunk method:**
   Activating the thread function before going into loop for the no. of rows in the resultant matrix. The loop runs for the chunk size (rows / no. of threads). It can be seen in the code snippet given below.



```
// Use OpenMP for parallelization
#pragma omp parallel num_threads(K)
{
    int tid = omp_get_thread_num(); // Get thread ID
    int startRow = tid * chunkSize;
    int endRow = (tid + 1) * chunkSize;

    // Perform matrix multiplication on the assigned chunk
    for (int i = startRow; i < endRow; i++)
    {
        for (int j = 0; j < N; j++)
        {
            for (int k = 0; k < N; k++)
            {
                matC[i][j] += matA[i][k] * matA[k][j];
            }
        }
    }
}
```

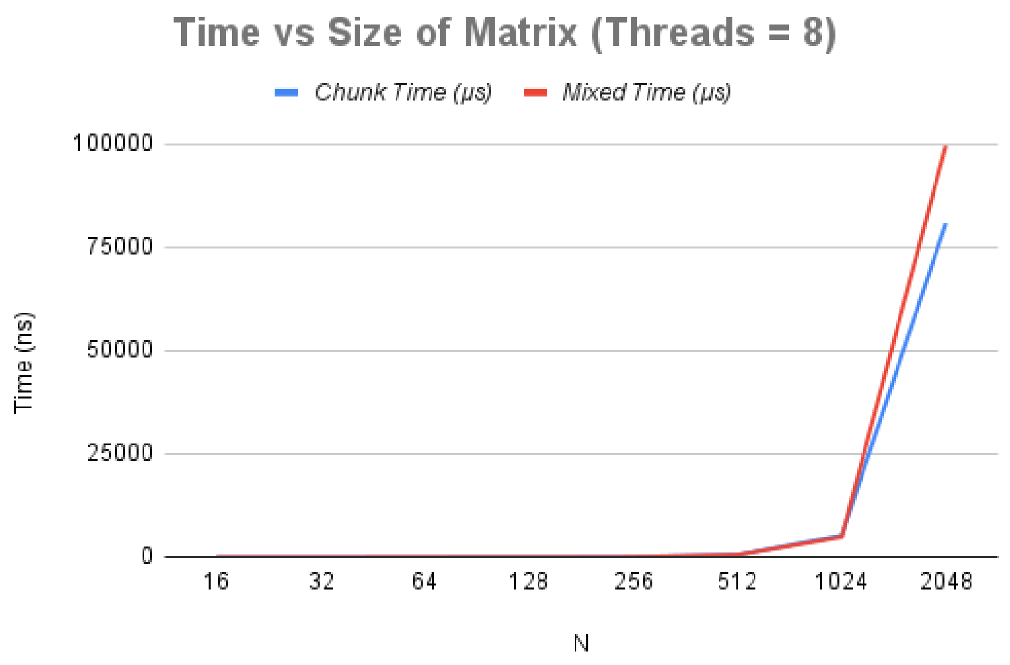
2. **Implementing the Mixed method:**
   Here just modifying the end Row and increment of the outer loop. Making it to run like as mentioned in the instruction of mixed method. The code snippet as follows.



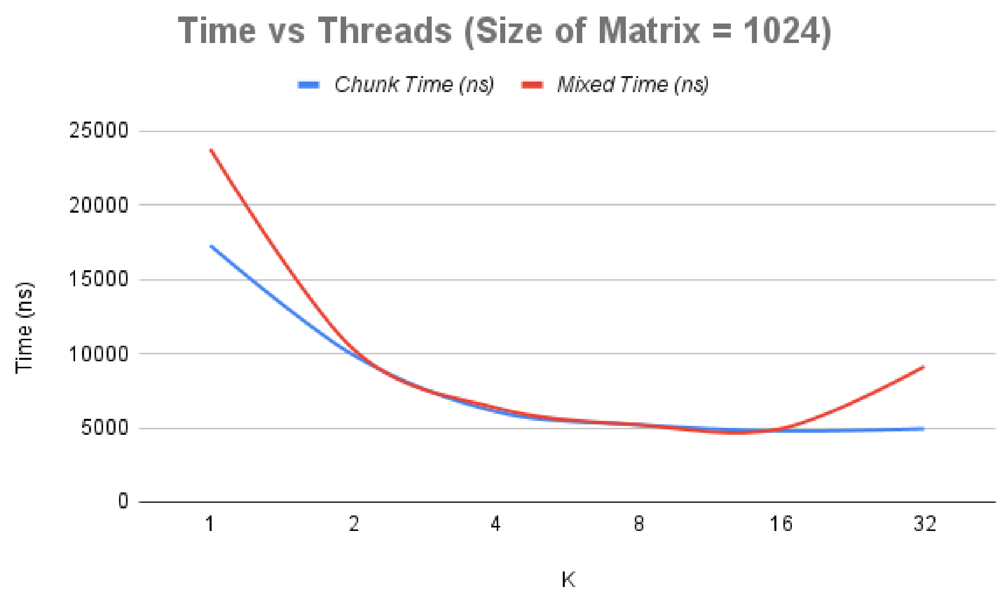
```
// Use OpenMP for parallelization
#pragma omp parallel num_threads(K)
{
    int tid = omp_get_thread_num(); // Get thread ID
    int startRow = tid + 1; // Adjust startRow to follow the Mixed method pattern
    int endRow = N;

    // Perform matrix multiplication on the assigned rows
    for (int i = startRow; i <= endRow; i += K)
    {
        for (int j = 0; j < N; j++)
        {
            for (int k = 0; k < N; k++)
            {
                matC[i - 1][j] += matA[i - 1][k] * matA[k][j];
            }
        }
    }
}
```

The performance of both methods is shown by the below tow plots



For the matrix size till 512, there is a negligible difference in the time taken to compute the resultant matrix.



Now, by keeping the matrix size constant and changing threads, the performance in both methods has significant differences at the lower and higher end of no. of threads. Overall , the chunk method is more efficient the Mixed method.