Assignment 2: Efficient Large Matrix Multiplication in OpenMP

Problem Statement

To develop an efficient large matrix multiplication algorithm in OpenMP.

Program Design:

The program creates two random matrices of dimensions starting from 200 x 200 through 2000 x 2000, increasing 200 dimensions in each iteration, using the function randomSquareMatrix(). The rand() function is used to create random numbers between 1 and 10, to fill the matrices A and B. The # pragma omp parallel for statement does the loop parallelizing so that we can initialize the matrix much faster and efficiently. Then a zero matrix is created of the same dimensions as the random matrix, for storing the results of the multiplication, using the zeroSquareMatrix() function. These matrices are multiplied by block multiplication with and without pragma directives.[1]

Block multiplication is the method of dividing matrices into blocks in such a way that the product of each block can be taken care of. The blocks are then stored in the secondary memory and their products are calculated. This method is faster than the traditional naïve method of matrix multiplication. The BlockMultiply() function performs normal block multiplication.

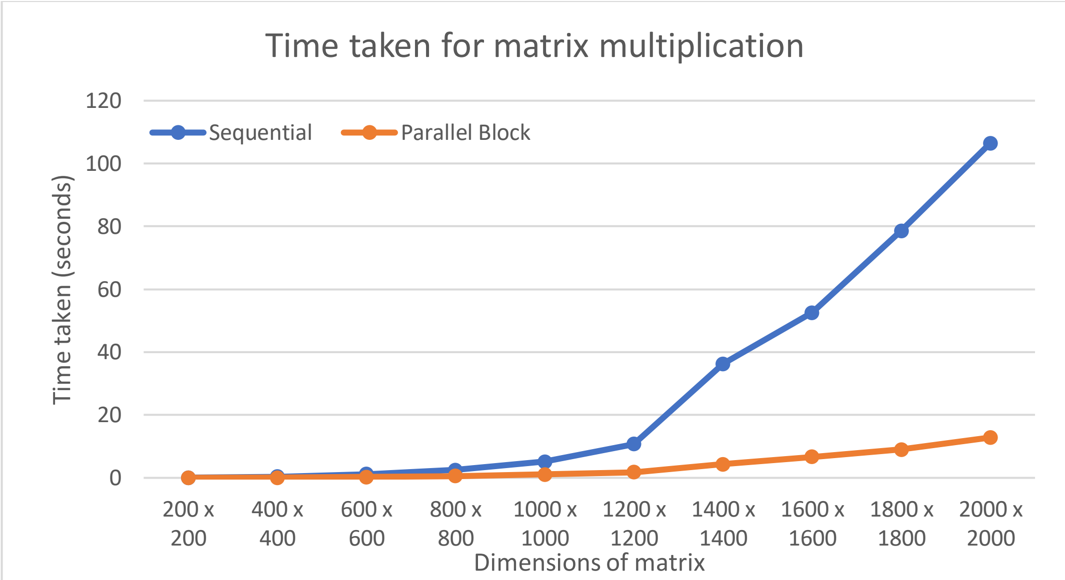
Parallel block multiplication is performed using the BlockMultiplyPragma() function [2]. The function uses the pragma omp parallel directive to parallelize and fork the blocks. In the #pragma omp parallel shared(matrixA, matrixB, matrixC, size, chunk) private(i, j, k, jj, kk, tmp) pragma, matrixA, matrixB, matrixC, size, and chunk are variables shared among the threads whereas I, j, k, jj, kk, and tmp are private variables each specific to each thread. The number of blocks depends on the L1 cache of the computer. This was run on a computer with a 384kB L1 cache. The optimal block size was found to be 16 by trial and error.

The #pragma omp for schedule (static, chunk) directive is used for scheduling the iterations of the loop. Static scheduling is used and, the chunk size is 1 which is the default. OpenMP divides the iterations into approximately equal size and distributes the chunks to threads in a circular fashion.

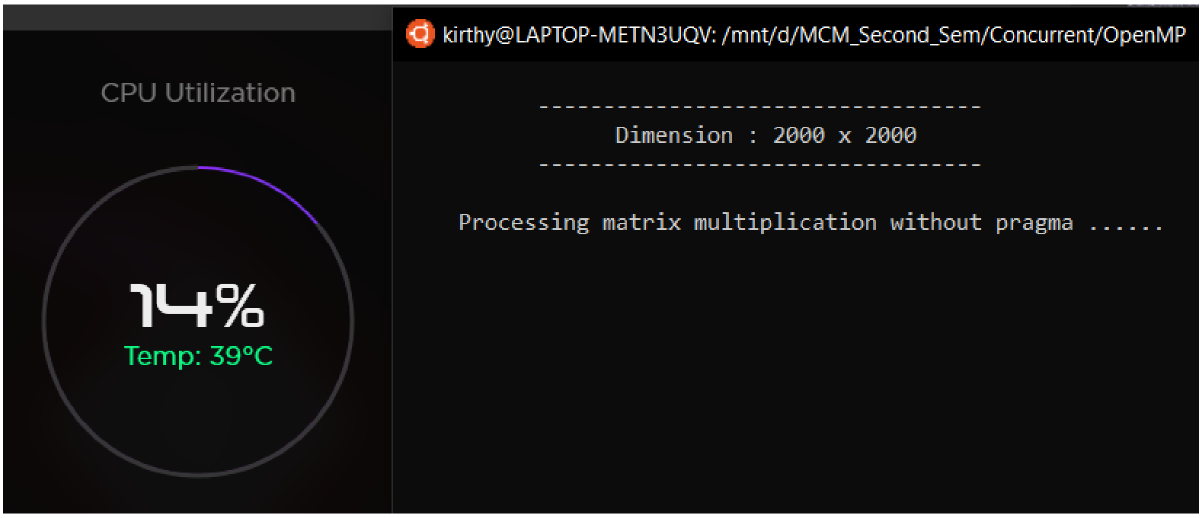
Program Efficiency:

Block matrix multiplication is faster than the traditional naïve matrix multiplication as the matrix is divided into blocks and secondary memory is used. The pragma directive parallelizes the iterations, making the program faster and more efficient.

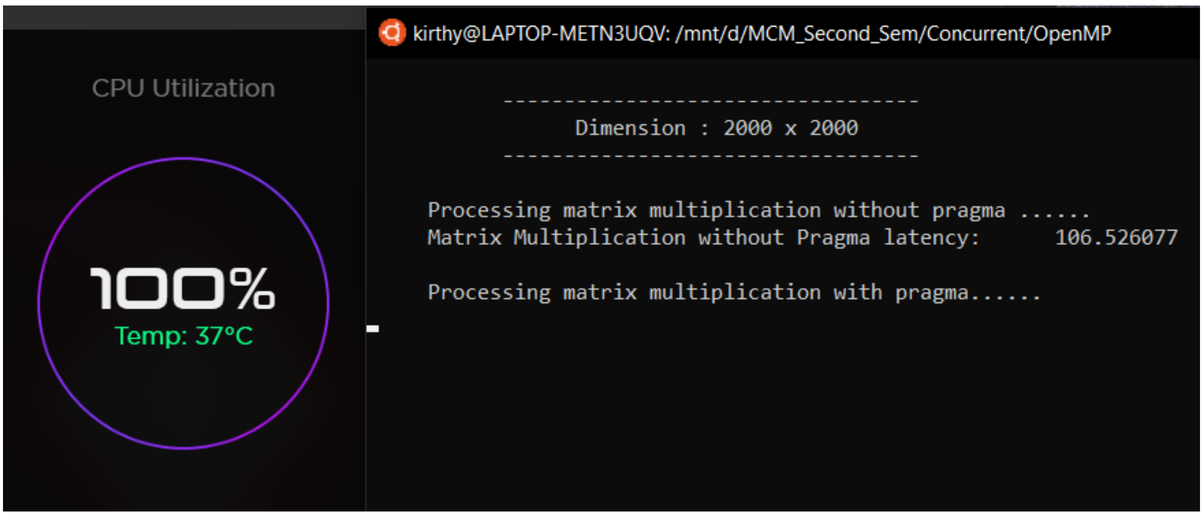
The time taken for normal block multiplication increases exponentially with an increase in dimensions while the time taken for parallel block multiplication increases very slightly as can be observed from the graph below.



Another observation is the CPU utilization. Normal block multiplication though very slow uses less than 15% CPU.



Whereas parallel block multiplication while very fast, takes 100% of CPU.



From the above, it can be concluded that parallel block multiplication is faster and much more efficient. Below are the times (in seconds) for normal block multiplication and parallel block multiplication for dimensions from 200 to 2000.

```
Select kirthy@LAPTOP-METN3UQV: /mnt/d/MCM_Second_Sem/Concurrent/OpenMP
        ----------------------------------
              Dimension : 200 x 200
        ----------------------------------

   Processing matrix multiplication without pragma ......
   Matrix Multiplication without Pragma latency:      0.079646

   Processing matrix multiplication with pragma......
   Matrix Multiplication with Pragma latency:         0.018877

        ----------------------------------
              Dimension : 400 x 400
        ----------------------------------

   Processing matrix multiplication without pragma ......
   Matrix Multiplication without Pragma latency:      0.399463

   Processing matrix multiplication with pragma......
   Matrix Multiplication with Pragma latency:         0.080884

        ----------------------------------
              Dimension : 600 x 600
        ----------------------------------

   Processing matrix multiplication without pragma ......
   Matrix Multiplication without Pragma latency:      1.177054

   Processing matrix multiplication with pragma......
   Matrix Multiplication with Pragma latency:         0.247088

        ----------------------------------
              Dimension : 800 x 800
        ----------------------------------

   Processing matrix multiplication without pragma ......
   Matrix Multiplication without Pragma latency:      2.533425

   Processing matrix multiplication with pragma......
   Matrix Multiplication with Pragma latency:         0.537815

        ----------------------------------
              Dimension : 1000 x 1000
        ----------------------------------

   Processing matrix multiplication without pragma ......
   Matrix Multiplication without Pragma latency:      5.168971

   Processing matrix multiplication with pragma......
   Matrix Multiplication with Pragma latency:         1.086140
```

```
Select kirthy@LAPTOP-METN3UQV: /mnt/d/MCM_Second_Sem/Concurrent/OpenMP

        ----------------------------------
              Dimension : 1200 x 1200
        ----------------------------------

   Processing matrix multiplication without pragma ......
   Matrix Multiplication without Pragma latency:      10.786072

   Processing matrix multiplication with pragma......
   Matrix Multiplication with Pragma latency:         1.797924

        ----------------------------------
              Dimension : 1400 x 1400
        ----------------------------------

   Processing matrix multiplication without pragma ......
   Matrix Multiplication without Pragma latency:      36.327751

   Processing matrix multiplication with pragma......
   Matrix Multiplication with Pragma latency:         4.372088

        ----------------------------------
              Dimension : 1600 x 1600
        ----------------------------------

   Processing matrix multiplication without pragma ......
   Matrix Multiplication without Pragma latency:      52.522961

   Processing matrix multiplication with pragma......
   Matrix Multiplication with Pragma latency:         6.630903

        ----------------------------------
              Dimension : 1800 x 1800
        ----------------------------------

   Processing matrix multiplication without pragma ......
   Matrix Multiplication without Pragma latency:      78.643311

   Processing matrix multiplication with pragma......
   Matrix Multiplication with Pragma latency:         8.978637

        ----------------------------------
              Dimension : 2000 x 2000
        ----------------------------------

   Processing matrix multiplication without pragma ......
   Matrix Multiplication without Pragma latency:      106.526077

   Processing matrix multiplication with pragma......
   Matrix Multiplication with Pragma latency:         12.836622
```

References:

[1] https://github.com/roshanmadhushanka/Parallel-Matrix-Multiply- program structure

[2] https://github.com/dmitrydonchenko/Block-Matrix-Multiplication-OpenMP/blob/master/block_matrix/Source.cpp - block multiplication