Documentation

Design steps of parallel algorithm.

Sequential algorithm of complexity $O(n^3)$

```
for(int i=0; i<dims; ++i){
      for(int j=0; j<dims; ++j){
            for(int k=0; k<dims; ++k){
                  C[i*dims+j] += A[i*dims+k]*B[k*dims+j];
            }
      }
}
```

is parallelized using OpenMP directives for loop parallelization.

The command

```
#pragma omp parallel for collapse (x) num_threads(threads)
```

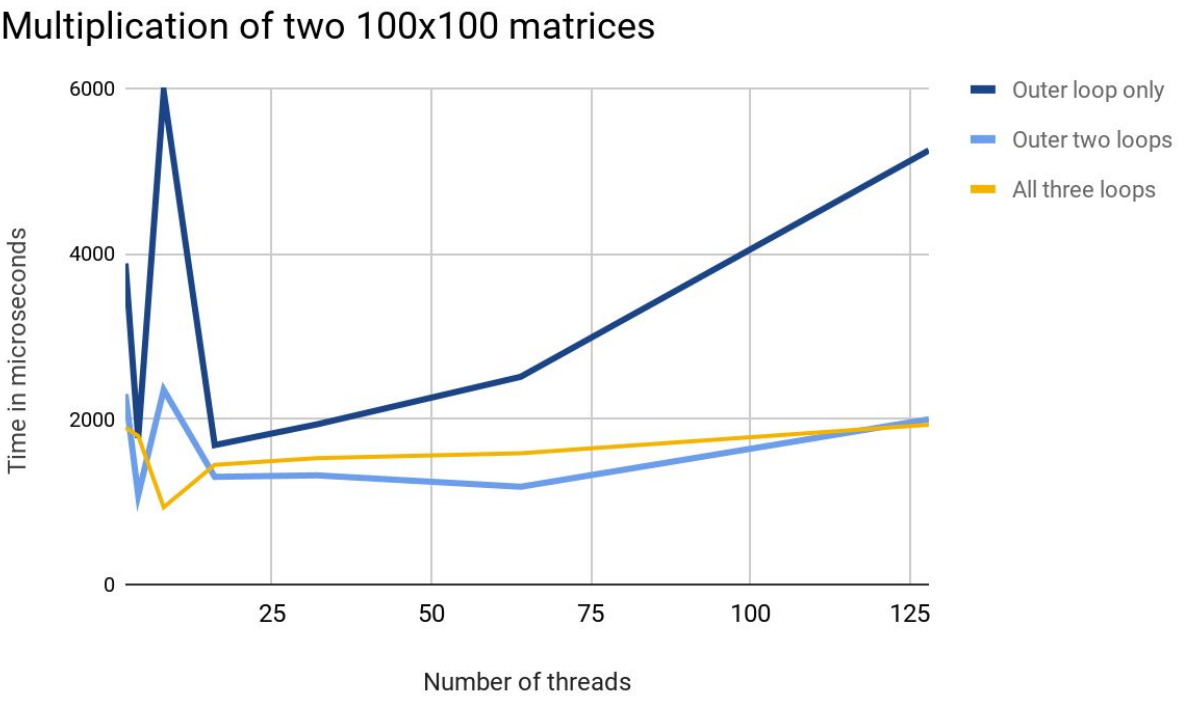
was used for this purpose.

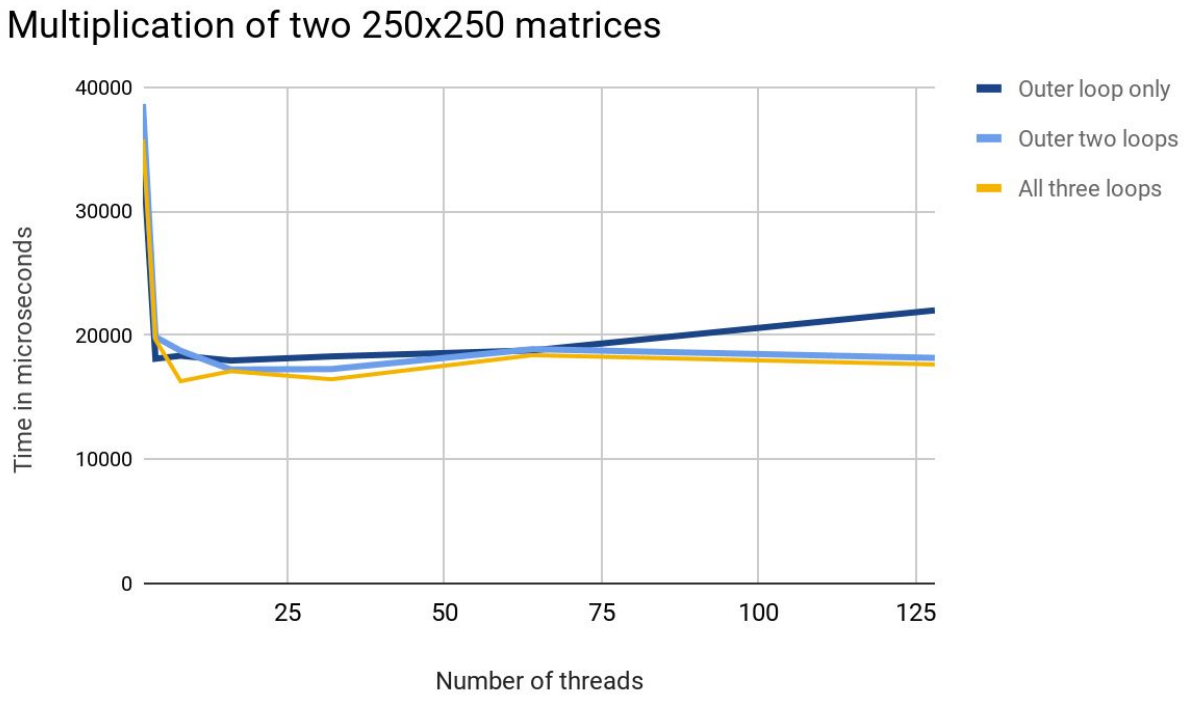
Here, **x** specifies the number of loops to be parallelized.
x when replaced by 1 is equivalent to parallelizing only the outermost loop.
x when replaced by 2 is equivalent to parallelizing the two outer loops.
x when replaced by 3 is equivalent to parallelizing all three loops.
and **threads** specifies the number of threads used for parallelization.

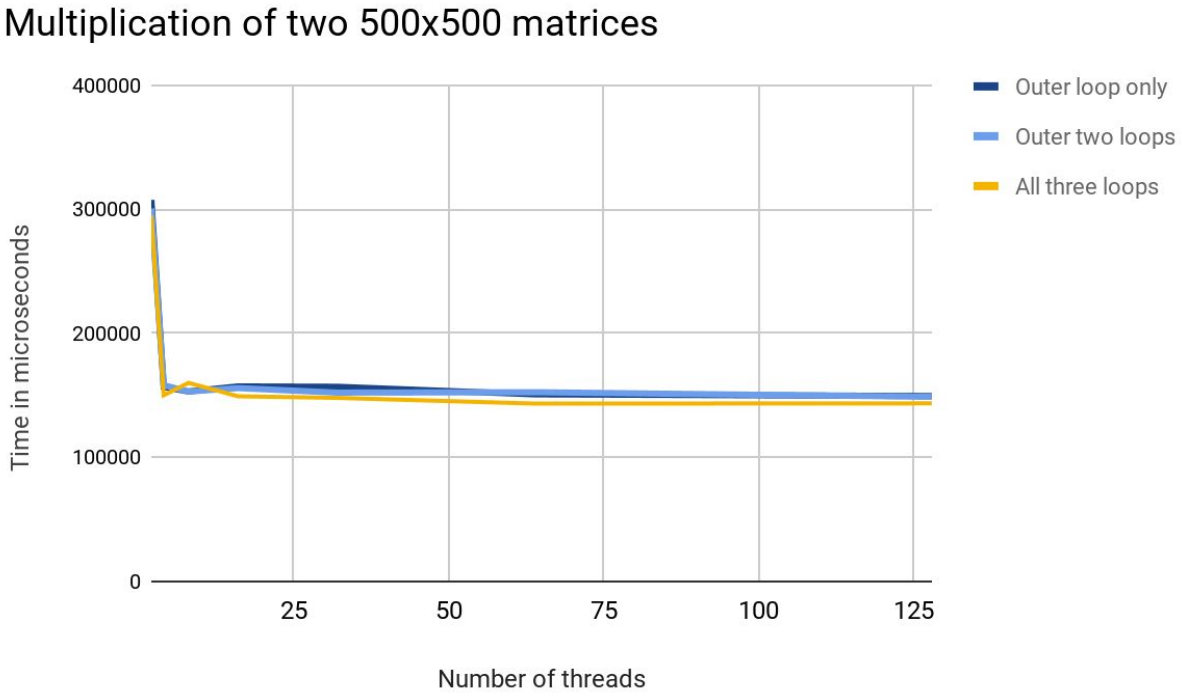
Note: Single dimension array is used to represent a 2D array as contiguous memory allocation leads to more efficient cache operations.

Note: All three cases of loop parallelization have been handled in a single file using separate functions.

Performance graphs







As can be seen from the above graphs, the optimal number of threads is 4. For more than 4 threads, performance either degrades due to thread overhead or only slightly improves.

Execution

The command

```
gcc -g -Wall -fopenmp -o matrix_mul matrix_mul.c
```

is used to compile the program.

and

```
./matrix_mul
```

is used to execute it. No command line arguments are taken.

```
Enter matrix dimensions
```

```
100
Enter number of threads
2
```

When prompted enter a single positive integer for number of dimensions (**dims**) and number of threads (**threads**)

The program will generate two matrices with randomly generated elements, both of size **dims**x**dims**.

These two matrices will be written to the file “**generated_input.txt**”

The matrices are multiplied using the three options specified above: outer loop parallelized, two outer loops parallelized, and all three loops parallelized. The number of threads is specified by **threads**.

The output of all three cases is written to “**output.txt**”.