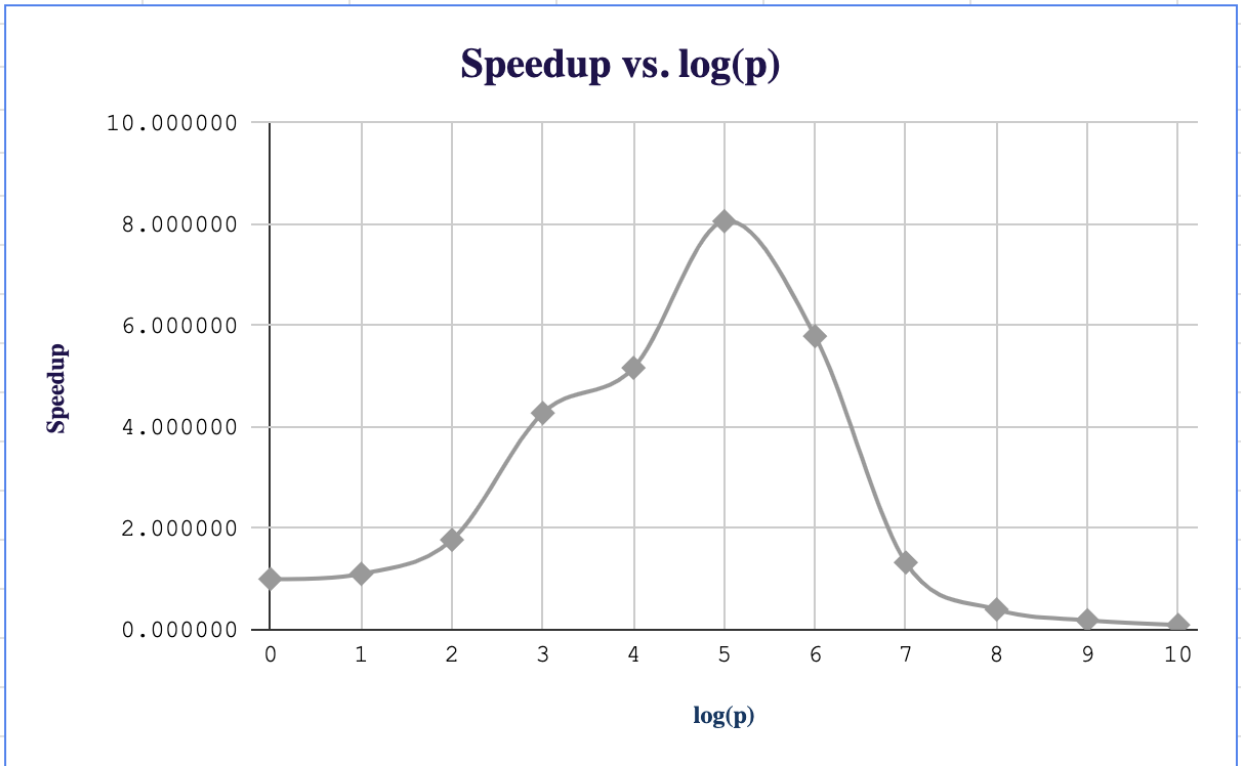
I developed a shared memory code that utilizes OpenMP to implement Strassen's recursive technique for matrix multiplication in parallel. This approach involves breaking down the matrix into equally sized blocks and calculating the multiplication values to reduce the time complexity of the matrix.

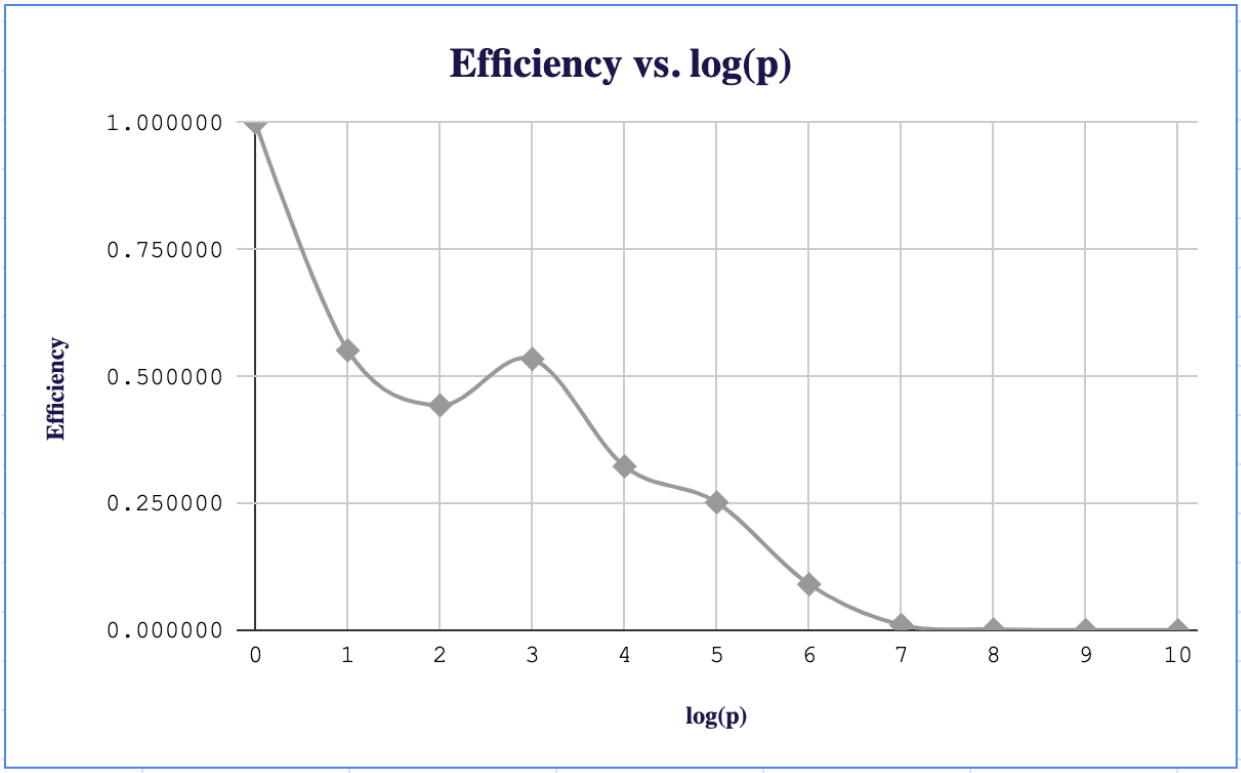
Design Choices:
To achieve this, the algorithm computes values from M1 to M7 for each step. As these values are independent of each other, they can be parallelized. However, they must be calculated before updating the final value of C for that particular step.
Initially, the parallel technique was expected to always perform better than the serial algorithm. However, this is only sometimes the case. When the values of k and k' are close to each other, the serial technique performs well. This could be attributed to the overhead computations involved in the parallel method, which increases the overall execution time of the program.

Experiment Results:

Processes	Execution Time	Matrix Size(k)	Threshold(k-k')	log(p)	k'
1	0.962501	10	3	0	7
2	0.872956	10	3	1	7
4	0.543090	10	3	2	7
8	0.225073	10	3	3	7
16	0.186291	10	3	4	7
32	0.119402	10	3	5	7
64	0.166143	10	3	6	7
128	0.726246	10	3	7	7
256	2.402134	10	3	8	7
512	5.129315	10	3	9	7
1024	10.137017	10	3	10	7

Graphs from the experiment:

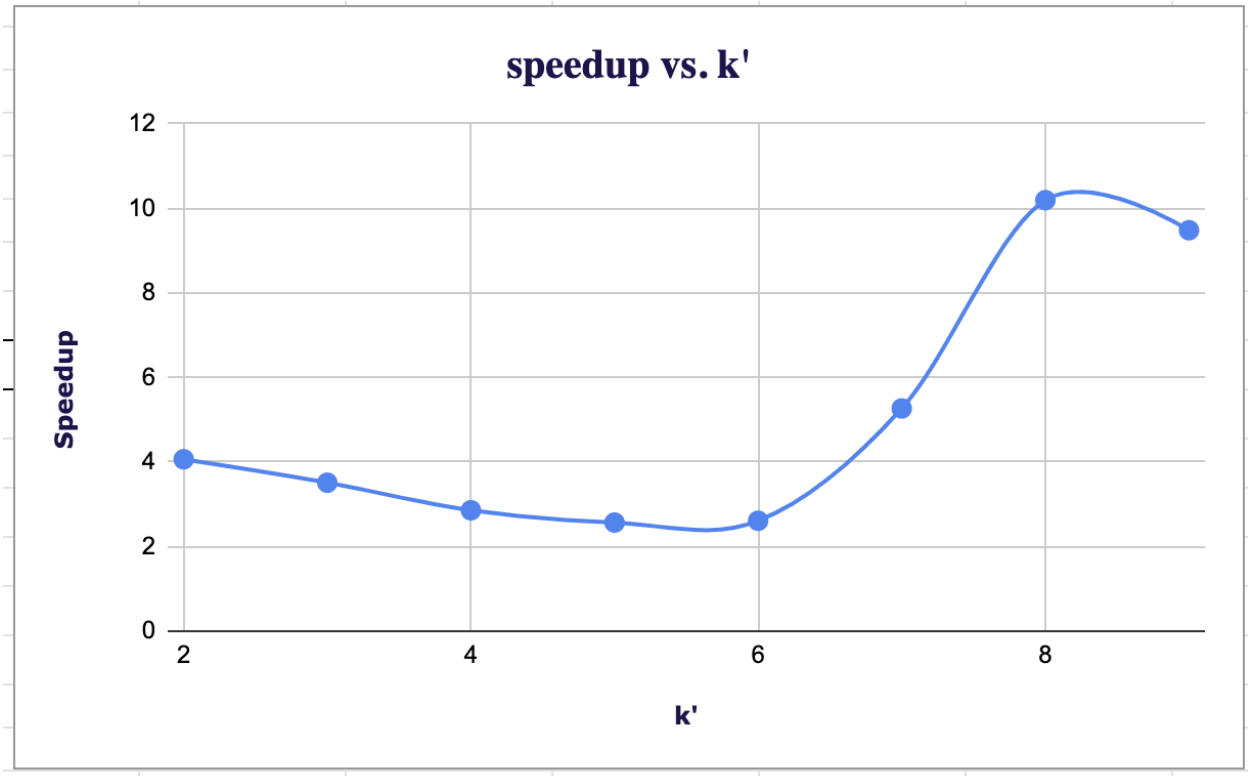

Speedup vs. log(p)
10.000000
8.000000
6.000000
4.000000
2.000000
0.000000
Speedup
0
1
2
3
4
5
6
7
8
9
10
log(p)

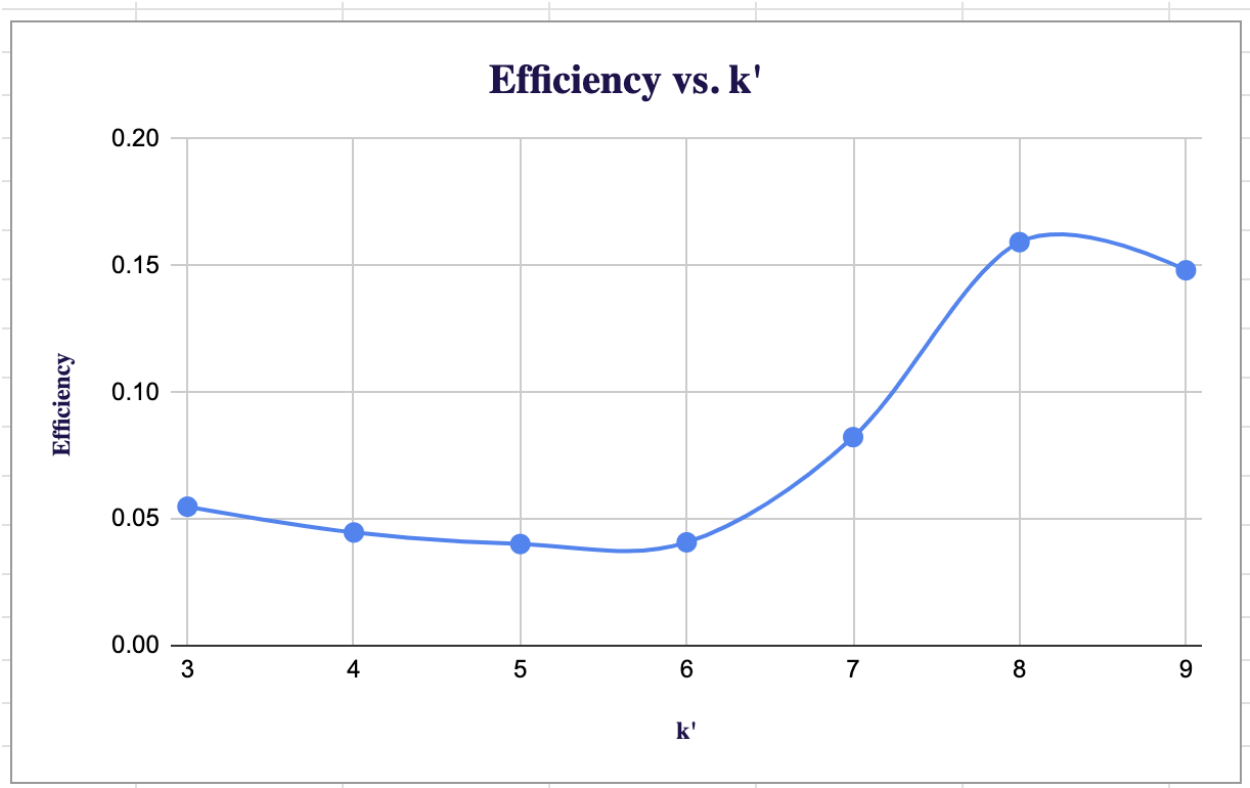

Efficiency vs. log(p)
1.000000
0.750000
0.500000
0.250000
0.000000
Efficiency
0
1
2
3
4
5
6
7
8
9
10
log(p)

Experiment with changing k':

Processes	Execution Time	Matrix Size(k)	Threshold(k-k')	k'
128	0.39691	10	8	2
128	0.771541	10	7	3
128	1.50881	10	6	4
128	0.620181	10	5	5
128	1.100746	10	4	6
128	0.517671	10	3	7
128	1.094301	10	2	8
128	2.108224	10	1	9
1	0.872634	10	8	2
1	0.701465	10	7	3
1	0.507932	10	6	4
1	0.480728	10	5	5
1	0.576571	10	4	6
1	0.980045	10	3	7
1	3.026044	10	2	8
1	16.60088	10	1	9


Execution Time vs. k'
Execution Time
2.0
1.5
1.0
0.5
0.0
2
4
6
8
k'


speedup vs. k'
Speedup
12
10
8
6
4
2
0
2
4
6
8
k'



As a matrix grows in size, its execution time also increases. When the threshold is increased, more serial algorithms are utilized, leading to a general increase in execution time, with a marginal improvement compared to the performance of the serial technique. However, the speedup increases exponentially with the size of the matrix, while the efficiency exhibits no significant trend.
The variations in performance between the two strategies are due to the number of levels and the changing matrix size. Proper selection of k and k' is crucial for achieving the best outcomes.

Compilation steps :
We used OpenMP to parallelize the algo.
1. Load the intel module
    module load intel
2. Compile the code :
    icc -qopenmp -o final_pro.exe final_pro.cpp
3. Now execute different commands for different k, k’ and threads
./major.exe 10 7 16
k = 10 k’ = 7
Threads = 16