CSCE 735 Fall 2022 Major Project Report
Parallelizing Strassen's Matrix-Multiplication Algorithm

Name: Sai Namith Garapati
UIN: 832001176

Contents

Strategy Chosen for Implementation:

A shared memory code is developed in C++ using OpenMPI to perform the parallel implementation of Strassen's Divide and Conquer algorithm to compute matrix multiplications. Matrices of size n x n where $n = 2^k$ for k = 2,4,6,8,10 are inputted allowing the user also to vary the terminal matrix size n x n where $n = 2^k/2^{k'}$ for values of k' taken respectively based on k. The code is compiled and executed successfully for all the above-mentioned combinations of inputs distributed on multiple threads varying from 1,2,4,16,64,256,1024.

Compilation and Execution Steps:

All the coding parts and the open MPI function calls needed for implementing the matrix multiplication using the Strassen method are mentioned in the majorproject_pc. cpp file

Steps followed:

- Prior to compiling and executing the code, please load the Intel software stack using the following command

```
module load intel
```

- To compile the OpenMPI code, use the following command to generate an executable file named majorproject_pc.exe

```
icc –qopenmp –o majorproject_pc.exe majorproject_pc.cpp
```

- To execute the program, use the following command which produces results on the terminal

  ./majorproject_pc.exe <k> <k'> <num_threads>

Where an input matrix of size n x n for $n = 2^k$, k is given as an argument. The size of the terminal matrix is equal to $2^k/2^{k'}$, and k' is given as an argument. Here the argument num_threads refers to the number of threads that are employed for execution.

e.g. `./majorproject_pc.exe 8 2 10`

- If you want to execute multiple commands in one go. Create a job file named starting with grace_job. <jobfile_name> and execute the following command which produces results for multiple commands in an output file.

```
sbatch grace_job.openmp_strassen p
```

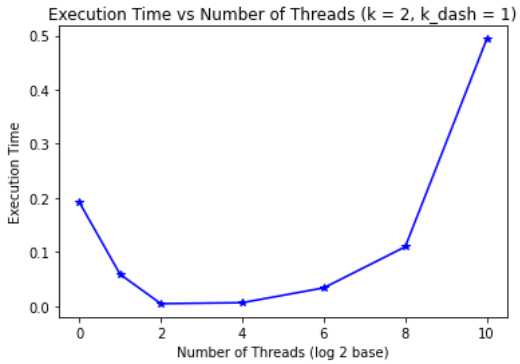
Execution time results, Speedup and Efficiency plots for different values of k and k':

Execution time results are shown for various combinations of k and k'. Execution time, Speedup and Efficiency plots are also obtained from all the results.

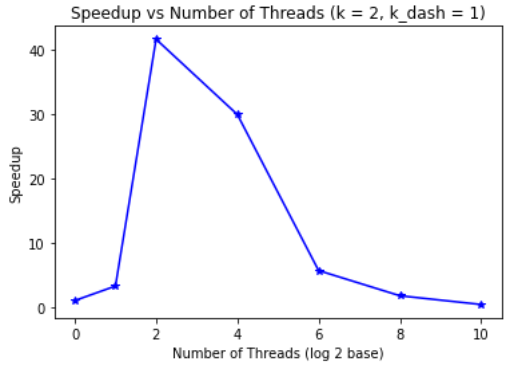
For k = 2, k' = 1 (4 x 4 matrix):

```
[namith03@grace1 ~]$ cat stra_2_1
Correct: k=2, Size of Matrix = 4 X 4, base= 1, number of threads = 1, Execution time for strassen =   0.1919 sec
Correct: k=2, Size of Matrix = 4 X 4, base= 1, number of threads = 2, Execution time for strassen =   0.0589 sec
Correct: k=2, Size of Matrix = 4 X 4, base= 1, number of threads = 4, Execution time for strassen =   0.0046 sec
Correct: k=2, Size of Matrix = 4 X 4, base= 1, number of threads = 16, Execution time for strassen =   0.0064 sec
Correct: k=2, Size of Matrix = 4 X 4, base= 1, number of threads = 64, Execution time for strassen =   0.0338 sec
Correct: k=2, Size of Matrix = 4 X 4, base= 1, number of threads = 256, Execution time for strassen =   0.1097 sec
Correct: k=2, Size of Matrix = 4 X 4, base= 1, number of threads = 1024, Execution time for strassen =   0.4933 sec
```

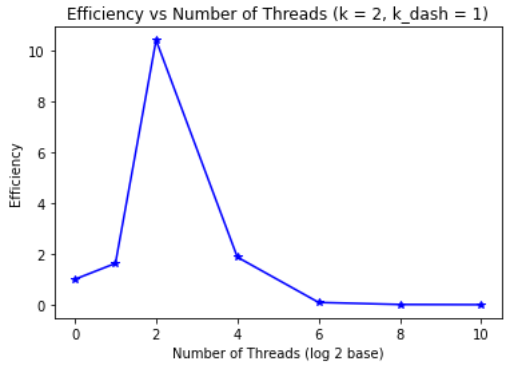
Execution time vs Number of Threads Plot:



Speedup Vs Number of Threads plot:



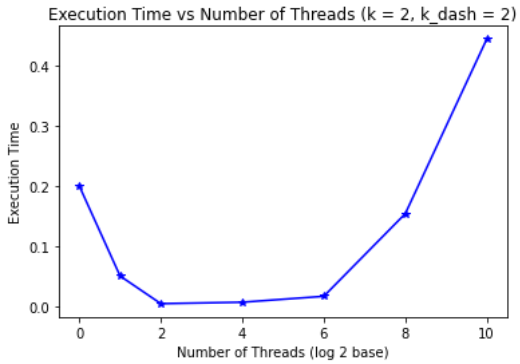
Efficiency vs Number of Threads Plot:



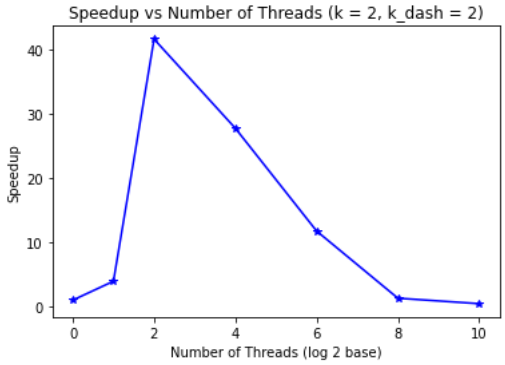
For k = 2, k' = 2 (4 x 4 matrix):

```
[namith03@grace1 ~]$ cat stra_2_2
Correct: k=2, Size of Matrix = 4 X 4, base= 2, number of threads = 1, Execution time for strassen =   0.1998 sec
Correct: k=2, Size of Matrix = 4 X 4, base= 2, number of threads = 2, Execution time for strassen =   0.0507 sec
Correct: k=2, Size of Matrix = 4 X 4, base= 2, number of threads = 4, Execution time for strassen =   0.0048 sec
Correct: k=2, Size of Matrix = 4 X 4, base= 2, number of threads = 16, Execution time for strassen =   0.0072 sec
Correct: k=2, Size of Matrix = 4 X 4, base= 2, number of threads = 64, Execution time for strassen =   0.0170 sec
Correct: k=2, Size of Matrix = 4 X 4, base= 2, number of threads = 256, Execution time for strassen =   0.1540 sec
Correct: k=2, Size of Matrix = 4 X 4, base= 2, number of threads = 1024, Execution time for strassen =   0.4436 sec
```

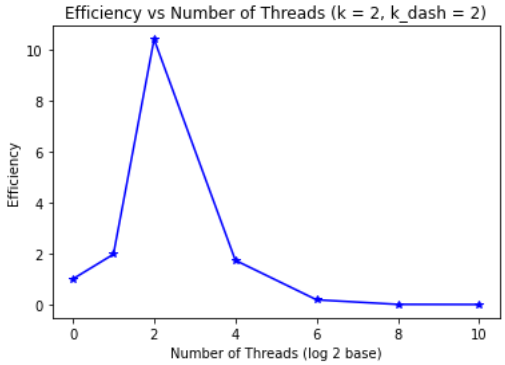
Execution Time Vs Number of Threads Plot:



Speedup Vs Number of Threads Plot:



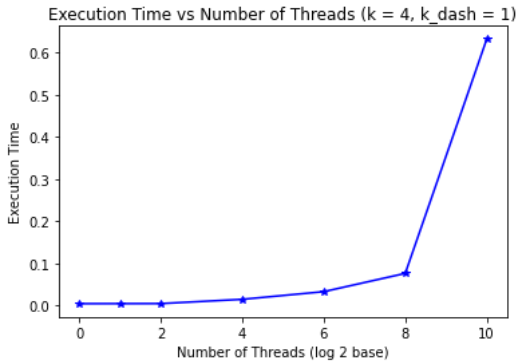
Efficiency vs Number of Threads Plot:



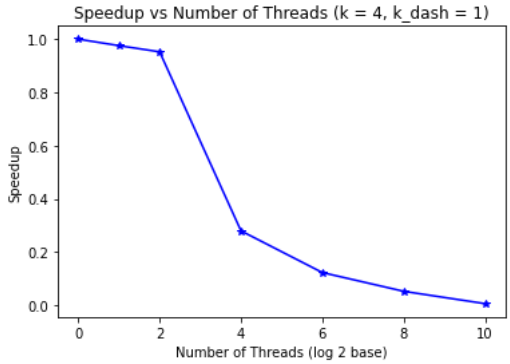
For k = 4, k' = 1 (16 x 16 matrix):

```
[namith03@grace1 ~]$ cat stra_4_1
Correct: k=4, Size of Matrix = 16 X 16, base= 1, number of threads = 1, Execution time for strassen =   0.0040 sec
Correct: k=4, Size of Matrix = 16 X 16, base= 1, number of threads = 2, Execution time for strassen =   0.0041 sec
Correct: k=4, Size of Matrix = 16 X 16, base= 1, number of threads = 4, Execution time for strassen =   0.0042 sec
Correct: k=4, Size of Matrix = 16 X 16, base= 1, number of threads = 16, Execution time for strassen =   0.0143 sec
Correct: k=4, Size of Matrix = 16 X 16, base= 1, number of threads = 64, Execution time for strassen =   0.0325 sec
Correct: k=4, Size of Matrix = 16 X 16, base= 1, number of threads = 256, Execution time for strassen =   0.0759 sec
```

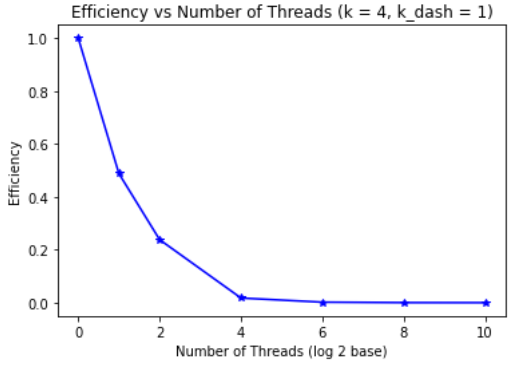
Execution Time vs Number of threads Plot:



Speedup vs Number of Threads Plot:



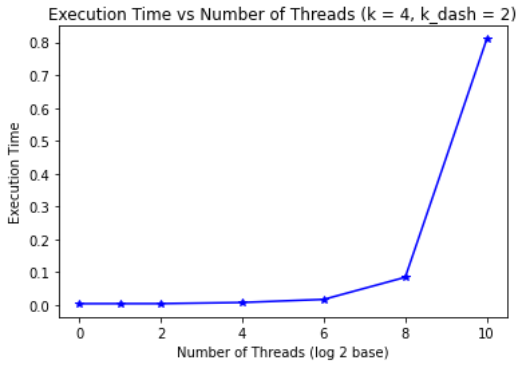
Efficiency vs Number of Threads Plot:



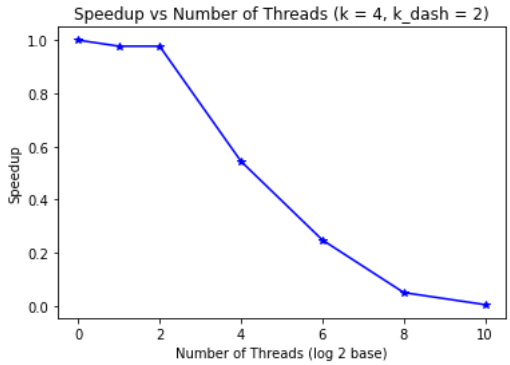
For k = 4, k' = 2 (16 x 16 matrix):

```
[namith03@grace1 ~]$ cat stra_4_2
Correct: k=4, Size of Matrix = 16 X 16, base= 2, number of threads = 1, Execution time for strassen =   0.0043 sec
Correct: k=4, Size of Matrix = 16 X 16, base= 2, number of threads = 2, Execution time for strassen =   0.0044 sec
Correct: k=4, Size of Matrix = 16 X 16, base= 2, number of threads = 4, Execution time for strassen =   0.0044 sec
Correct: k=4, Size of Matrix = 16 X 16, base= 2, number of threads = 16, Execution time for strassen =   0.0079 sec
Correct: k=4, Size of Matrix = 16 X 16, base= 2, number of threads = 64, Execution time for strassen =   0.0173 sec
Correct: k=4, Size of Matrix = 16 X 16, base= 2, number of threads = 256, Execution time for strassen =   0.0847 sec
Correct: k=4, Size of Matrix = 16 X 16, base= 2, number of threads = 1024, Execution time for strassen =   0.8098 sec
```

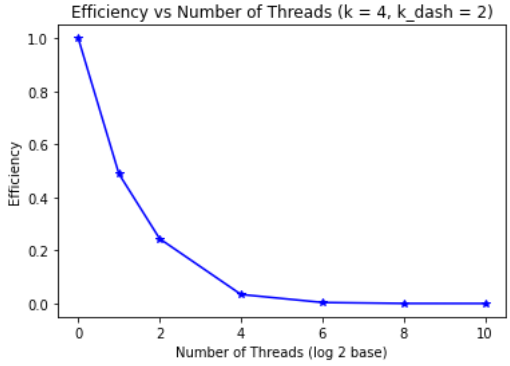
Execution Time vs Number of Threads:



Speedup vs Number of Threads:



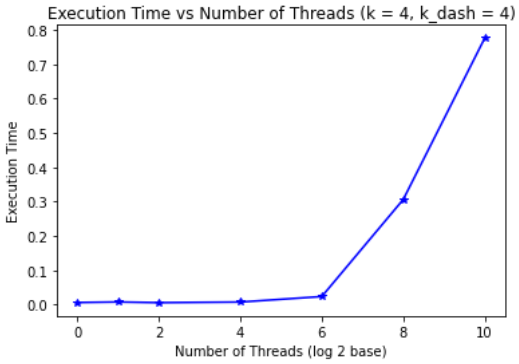
Efficiency vs Number of Threads:



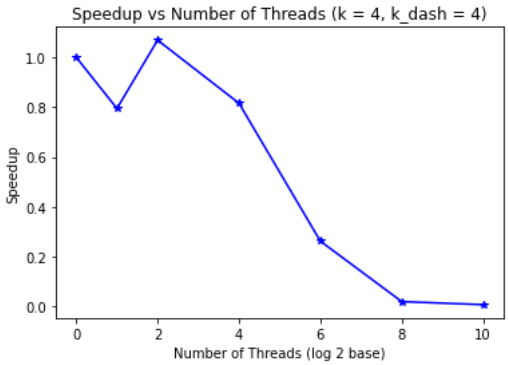
For k = 4, k' = 4 (16 x 16 matrix):

```
[namith03@grace1 ~]$ cat stra_4_4
Correct: k=4, Size of Matrix = 16 X 16, base= 4, number of threads = 1, Execution time for strassen =   0.0062 sec
Correct: k=4, Size of Matrix = 16 X 16, base= 4, number of threads = 2, Execution time for strassen =   0.0078 sec
Correct: k=4, Size of Matrix = 16 X 16, base= 4, number of threads = 4, Execution time for strassen =   0.0058 sec
Correct: k=4, Size of Matrix = 16 X 16, base= 4, number of threads = 16, Execution time for strassen =   0.0076 sec
Correct: k=4, Size of Matrix = 16 X 16, base= 4, number of threads = 64, Execution time for strassen =   0.0236 sec
Correct: k=4, Size of Matrix = 16 X 16, base= 4, number of threads = 256, Execution time for strassen =   0.3063 sec
Correct: k=4, Size of Matrix = 16 X 16, base= 4, number of threads = 1024, Execution time for strassen =   0.7762 sec
```

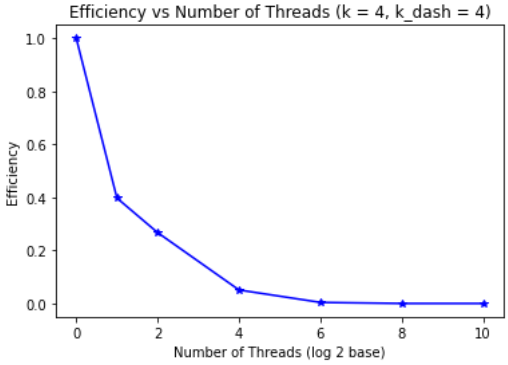
Execution Time Vs Number of Threads:



Speedup Vs Number of Threads:



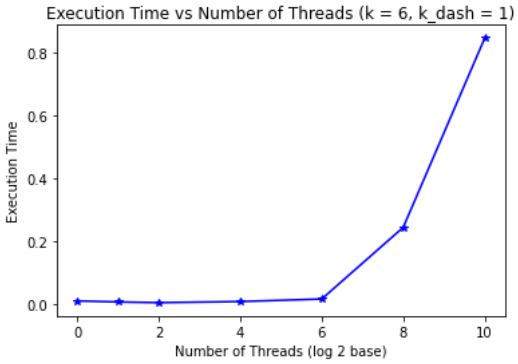
Efficiency Vs Number of Threads:



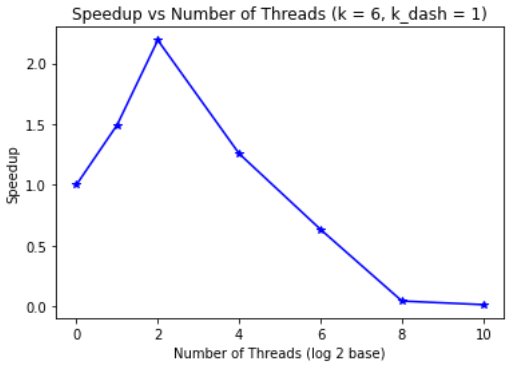
k = 6, k' = 1 (64x 64 matrix):

```
[namith03@grace1 ~]$ cat stra_6_1
Correct: k=6, Size of Matrix = 64 X 64, base= 1, number of threads = 1, Execution time for strassen =   0.0103 sec
Correct: k=6, Size of Matrix = 64 X 64, base= 1, number of threads = 2, Execution time for strassen =   0.0069 sec
Correct: k=6, Size of Matrix = 64 X 64, base= 1, number of threads = 4, Execution time for strassen =   0.0047 sec
Correct: k=6, Size of Matrix = 64 X 64, base= 1, number of threads = 16, Execution time for strassen =   0.0082 sec
Correct: k=6, Size of Matrix = 64 X 64, base= 1, number of threads = 64, Execution time for strassen =   0.0163 sec
Correct: k=6, Size of Matrix = 64 X 64, base= 1, number of threads = 256, Execution time for strassen =   0.2441 sec
Correct: k=6, Size of Matrix = 64 X 64, base= 1, number of threads = 1024, Execution time for strassen =   0.8453 sec
```

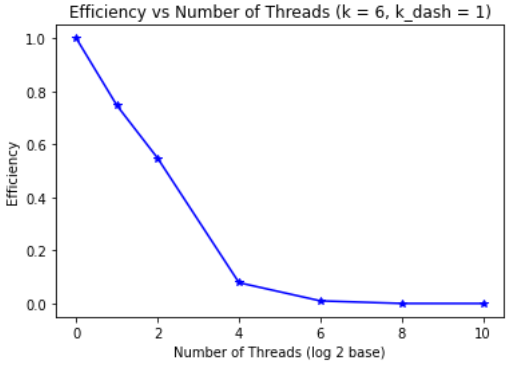
Execution Time Vs Number of Threads:



Speedup Vs Number of Threads:



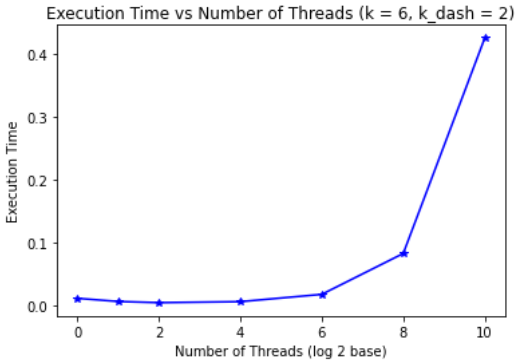
Efficiency Vs Number of Threads:



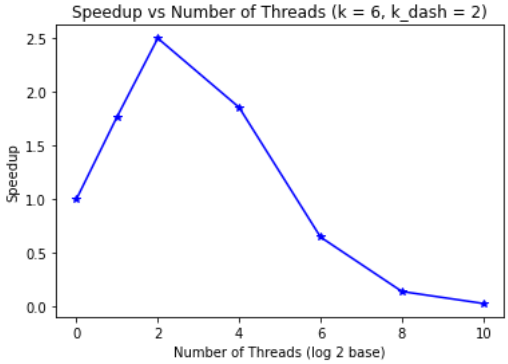
k = 6, k' = 2 (64x 64 matrix):

```
[namith03@grace1 ~]$ cat stra_6_2
Correct: k=6, Size of Matrix = 64 X 64, base= 2, number of threads = 1, Execution time for strassen =   0.0115 sec
Correct: k=6, Size of Matrix = 64 X 64, base= 2, number of threads = 2, Execution time for strassen =   0.0065 sec
Correct: k=6, Size of Matrix = 64 X 64, base= 2, number of threads = 4, Execution time for strassen =   0.0046 sec
Correct: k=6, Size of Matrix = 64 X 64, base= 2, number of threads = 16, Execution time for strassen =   0.0062 sec
Correct: k=6, Size of Matrix = 64 X 64, base= 2, number of threads = 64, Execution time for strassen =   0.0178 sec
Correct: k=6, Size of Matrix = 64 X 64, base= 2, number of threads = 256, Execution time for strassen =   0.0828 sec
Correct: k=6, Size of Matrix = 64 X 64, base= 2, number of threads = 1024, Execution time for strassen =   0.4253 sec
```

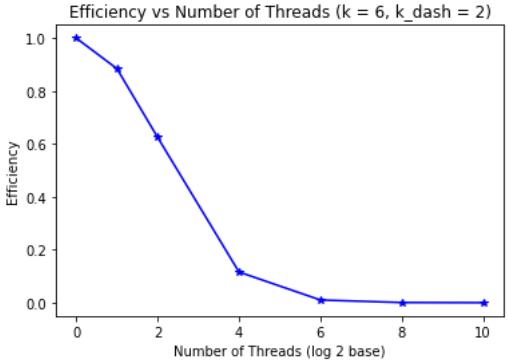
Execution Time vs Number of Threads Plot:



Speedup vs Number of Threads Plot:



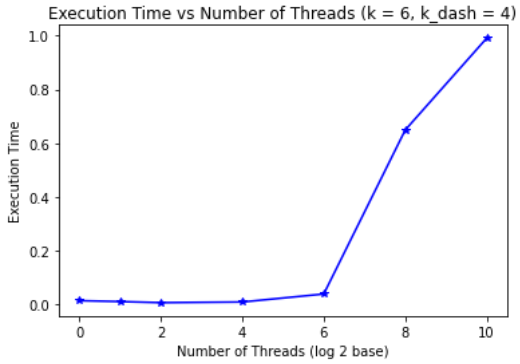
Efficiency vs Number of Threads Plot:



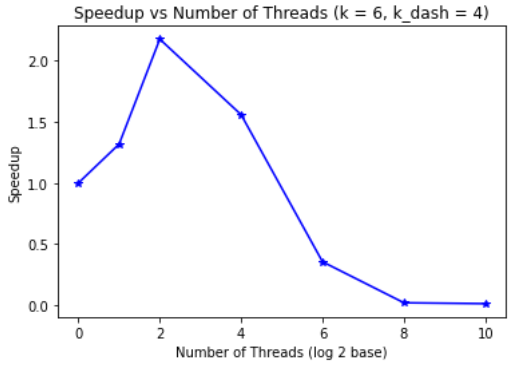
k = 6, k' = 4 (64x 64 matrix):

```
[namith03@grace1 ~]$ cat stra_6_4
Correct: k=6, Size of Matrix = 64 X 64, base= 4, number of threads = 1, Execution time for strassen =   0.0137 sec
Correct: k=6, Size of Matrix = 64 X 64, base= 4, number of threads = 2, Execution time for strassen =   0.0104 sec
Correct: k=6, Size of Matrix = 64 X 64, base= 4, number of threads = 4, Execution time for strassen =   0.0063 sec
Correct: k=6, Size of Matrix = 64 X 64, base= 4, number of threads = 16, Execution time for strassen =   0.0088 sec
Correct: k=6, Size of Matrix = 64 X 64, base= 4, number of threads = 64, Execution time for strassen =   0.0385 sec
Correct: k=6, Size of Matrix = 64 X 64, base= 4, number of threads = 256, Execution time for strassen =   0.6490 sec
Correct: k=6, Size of Matrix = 64 X 64, base= 4, number of threads = 1024, Execution time for strassen =   0.9908 sec
```

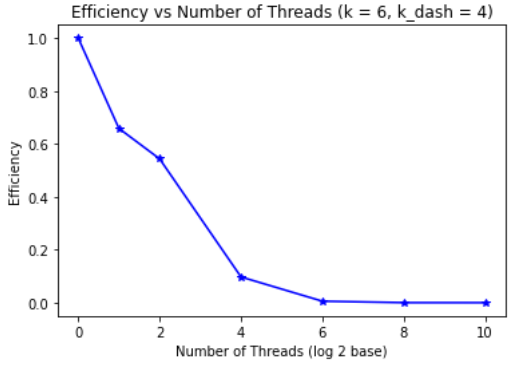
Execution Time vs Number of Threads Plot:



Speedup vs Number of Threads Plot:



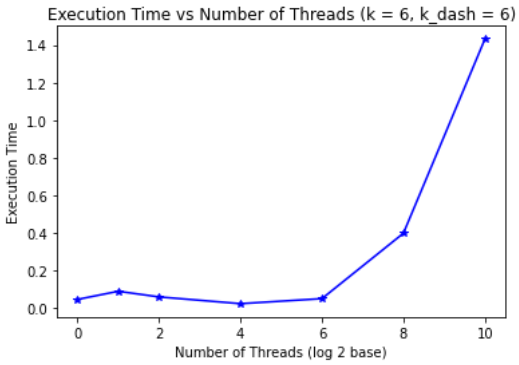
Efficiency vs Number of Threads Plot:



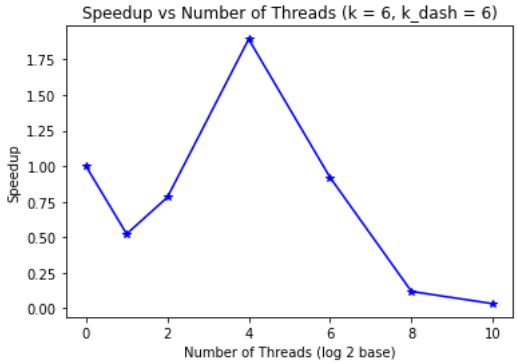
k = 6, k' = 6 (64x 64 matrix):

```
[namith03@grace1 ~]$ cat stra_6_6
Correct: k=6, Size of Matrix = 64 X 64, base= 6, number of threads = 1, Execution time for strassen =   0.0477 sec
Correct: k=6, Size of Matrix = 64 X 64, base= 6, number of threads = 2, Execution time for strassen =   0.0911 sec
Correct: k=6, Size of Matrix = 64 X 64, base= 6, number of threads = 4, Execution time for strassen =   0.0611 sec
Correct: k=6, Size of Matrix = 64 X 64, base= 6, number of threads = 16, Execution time for strassen =   0.0252 sec
Correct: k=6, Size of Matrix = 64 X 64, base= 6, number of threads = 64, Execution time for strassen =   0.0518 sec
Correct: k=6, Size of Matrix = 64 X 64, base= 6, number of threads = 256, Execution time for strassen =   0.3995 sec
Correct: k=6, Size of Matrix = 64 X 64, base= 6, number of threads = 1024, Execution time for strassen =   1.4314 sec
```

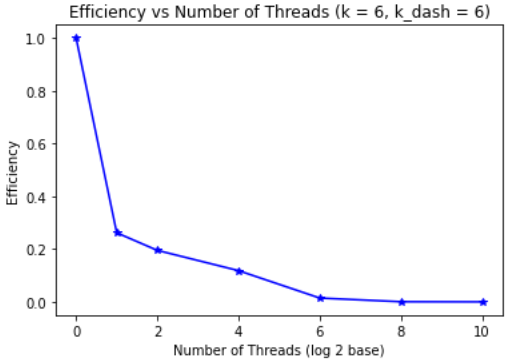
Execution Time vs Number of Threads Plot:



Speedup vs Number of Threads Plot:



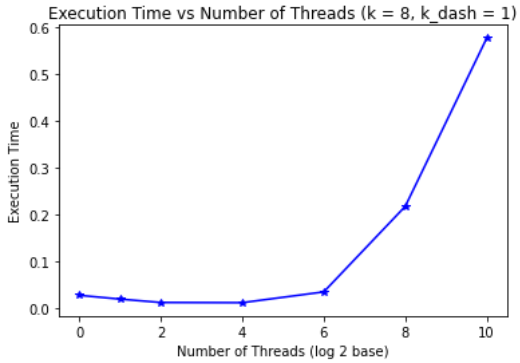
Efficiency vs Number of Threads Plot:



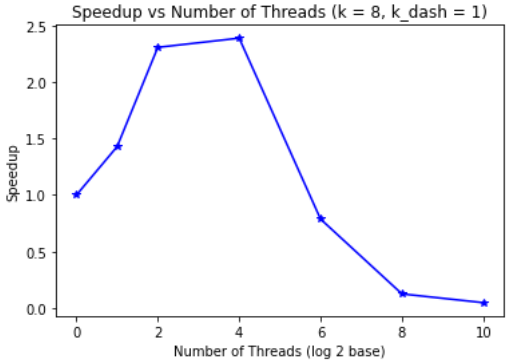
k = 8, k' = 1 (256x 256 matrix):

```
[namith03@grace1 ~]$ cat stra_8_1
Correct: k=8, Size of Matrix = 256 X 256, base= 1, number of threads = 1, Execution time for strassen =   0.0270 sec
Correct: k=8, Size of Matrix = 256 X 256, base= 1, number of threads = 2, Execution time for strassen =   0.0189 sec
Correct: k=8, Size of Matrix = 256 X 256, base= 1, number of threads = 4, Execution time for strassen =   0.0117 sec
Correct: k=8, Size of Matrix = 256 X 256, base= 1, number of threads = 16, Execution time for strassen =   0.0113 sec
Correct: k=8, Size of Matrix = 256 X 256, base= 1, number of threads = 64, Execution time for strassen =   0.0343 sec
Correct: k=8, Size of Matrix = 256 X 256, base= 1, number of threads = 256, Execution time for strassen =   0.2171 sec
Correct: k=8, Size of Matrix = 256 X 256, base= 1, number of threads = 1024, Execution time for strassen =   0.5767 sec
```

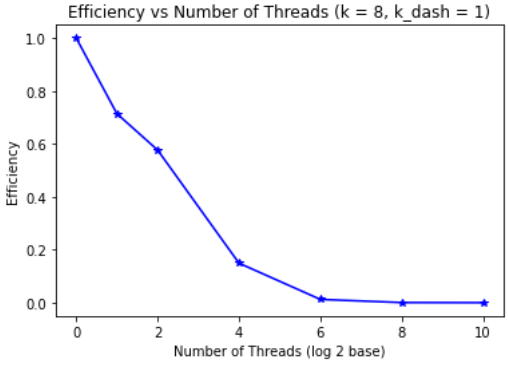
Execution Time Vs Number of Threads Plot:



Speedup Vs Number of Threads Plot:



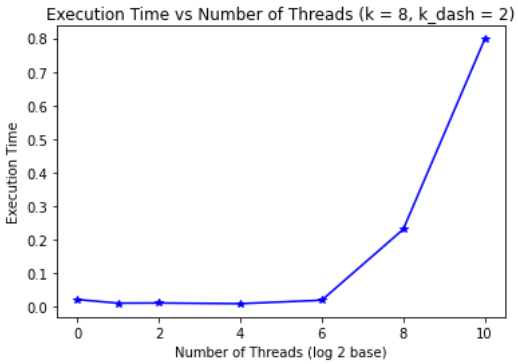
Efficiency vs Number of Threads Plot:



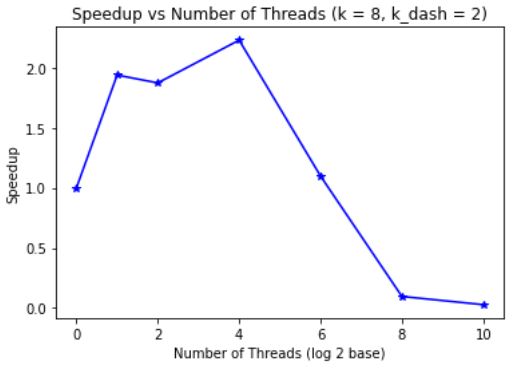
k = 8, k' = 2 (256x 256 matrix):

```
[namith03@grace1 ~]$ cat stra_8_2
Correct: k=8, Size of Matrix = 256 X 256, base= 2, number of threads = 1, Execution time for strassen =   0.0223 sec
Correct: k=8, Size of Matrix = 256 X 256, base= 2, number of threads = 2, Execution time for strassen =   0.0115 sec
Correct: k=8, Size of Matrix = 256 X 256, base= 2, number of threads = 4, Execution time for strassen =   0.0119 sec
Correct: k=8, Size of Matrix = 256 X 256, base= 2, number of threads = 16, Execution time for strassen =   0.0100 sec
Correct: k=8, Size of Matrix = 256 X 256, base= 2, number of threads = 64, Execution time for strassen =   0.0203 sec
Correct: k=8, Size of Matrix = 256 X 256, base= 2, number of threads = 256, Execution time for strassen =   0.2314 sec
Correct: k=8, Size of Matrix = 256 X 256, base= 2, number of threads = 1024, Execution time for strassen =   0.7975 sec
```

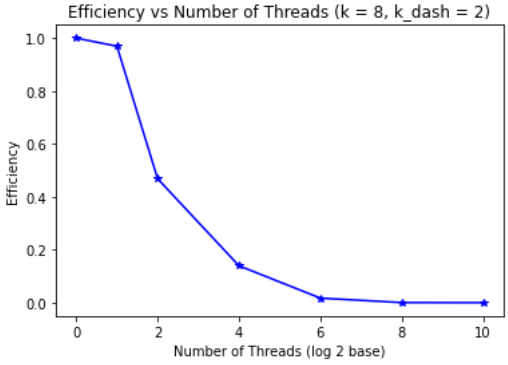
Execution Time vs Number of Threads:



Speedup vs Number of Threads:



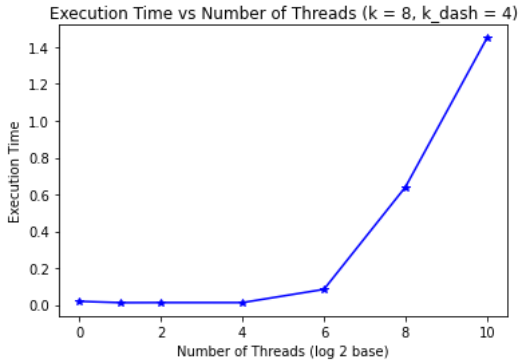
Efficiency vs Number of Threads:



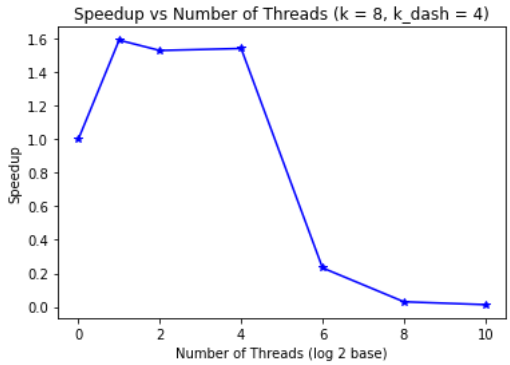
k = 8, k' = 4 (256x 256 matrix):

```
[namith03@grace1 ~]$ cat stra_8_4
Correct: k=8, Size of Matrix = 256 X 256, base= 4, number of threads = 1, Execution time for strassen =   0.0197 sec
Correct: k=8, Size of Matrix = 256 X 256, base= 4, number of threads = 2, Execution time for strassen =   0.0124 sec
Correct: k=8, Size of Matrix = 256 X 256, base= 4, number of threads = 4, Execution time for strassen =   0.0129 sec
Correct: k=8, Size of Matrix = 256 X 256, base= 4, number of threads = 16, Execution time for strassen =   0.0128 sec
Correct: k=8, Size of Matrix = 256 X 256, base= 4, number of threads = 64, Execution time for strassen =   0.0847 sec
Correct: k=8, Size of Matrix = 256 X 256, base= 4, number of threads = 256, Execution time for strassen =   0.6403 sec
Correct: k=8, Size of Matrix = 256 X 256, base= 4, number of threads = 1024, Execution time for strassen =   1.4489 sec
```

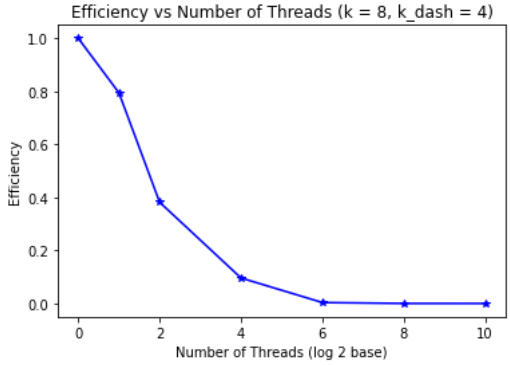
Execution Time vs Number of Threads:



Speedup Vs Number of Threads:



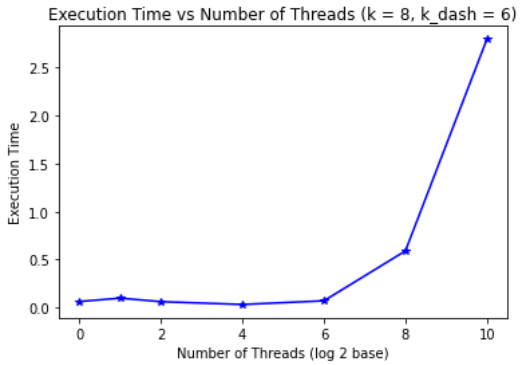
Efficiency vs Number of Threads:



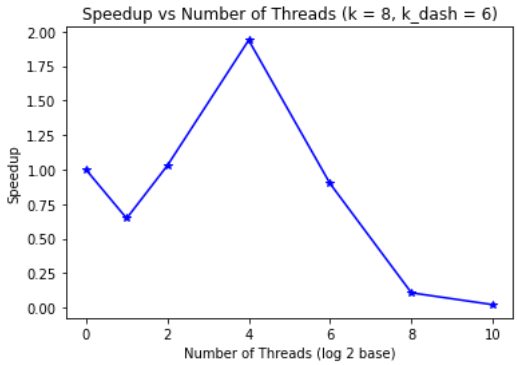
k = 8, k' = 6 (256x 256 matrix):

```
[namith03@grace1 ~]$ cat stra_8_6
Correct: k=8, Size of Matrix = 256 X 256, base= 6, number of threads = 1, Execution time for strassen =   0.0641 sec
Correct: k=8, Size of Matrix = 256 X 256, base= 6, number of threads = 2, Execution time for strassen =   0.0990 sec
Correct: k=8, Size of Matrix = 256 X 256, base= 6, number of threads = 4, Execution time for strassen =   0.0623 sec
Correct: k=8, Size of Matrix = 256 X 256, base= 6, number of threads = 16, Execution time for strassen =   0.0331 sec
Correct: k=8, Size of Matrix = 256 X 256, base= 6, number of threads = 64, Execution time for strassen =   0.0713 sec
Correct: k=8, Size of Matrix = 256 X 256, base= 6, number of threads = 256, Execution time for strassen =   0.5902 sec
Correct: k=8, Size of Matrix = 256 X 256, base= 6, number of threads = 1024, Execution time for strassen =   2.7929 sec
```

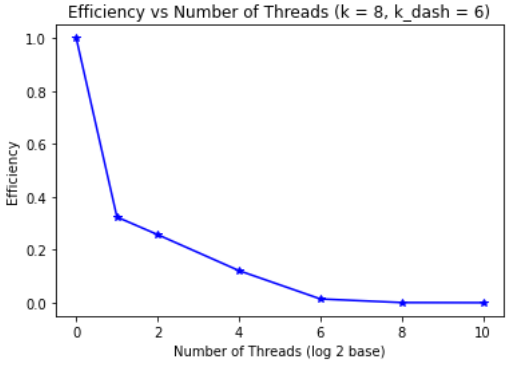
Execution Time Vs Number of Threads:



Speedup Vs Number of Threads:



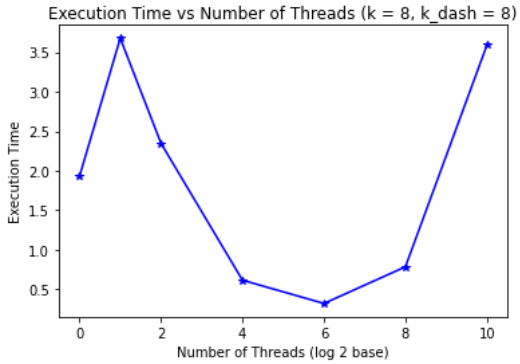
Efficiency Vs Number of Threads:



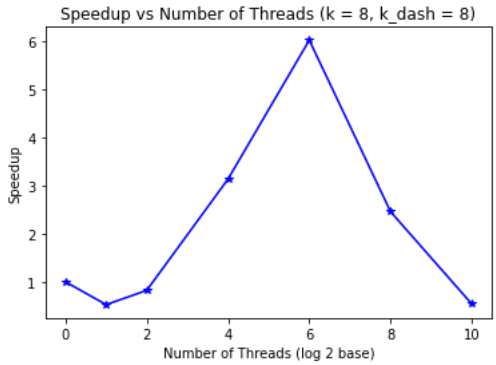
k = 8, k' = 8 (256x 256 matrix):

```
[namith03@grace1 ~]$ cat stra_8_8
Correct: k=8, Size of Matrix = 256 X 256, base= 8, number of threads = 1, Execution time for strassen =   1.9301 sec
Correct: k=8, Size of Matrix = 256 X 256, base= 8, number of threads = 2, Execution time for strassen =   3.6763 sec
Correct: k=8, Size of Matrix = 256 X 256, base= 8, number of threads = 4, Execution time for strassen =   2.3472 sec
Correct: k=8, Size of Matrix = 256 X 256, base= 8, number of threads = 16, Execution time for strassen =   0.6149 sec
Correct: k=8, Size of Matrix = 256 X 256, base= 8, number of threads = 64, Execution time for strassen =   0.3200 sec
Correct: k=8, Size of Matrix = 256 X 256, base= 8, number of threads = 256, Execution time for strassen =   0.7844 sec
Correct: k=8, Size of Matrix = 256 X 256, base= 8, number of threads = 1024, Execution time for strassen =   3.5890 sec
```

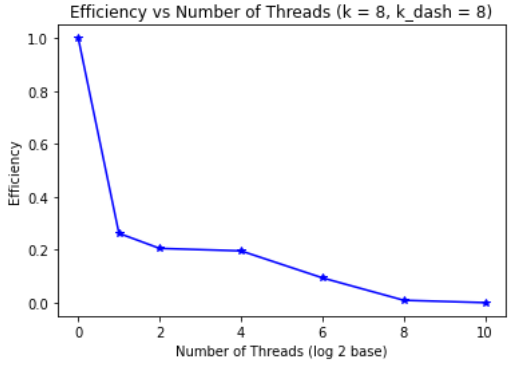
Execution Time vs Number of Threads plot:



Speedup vs Number of Threads plot:



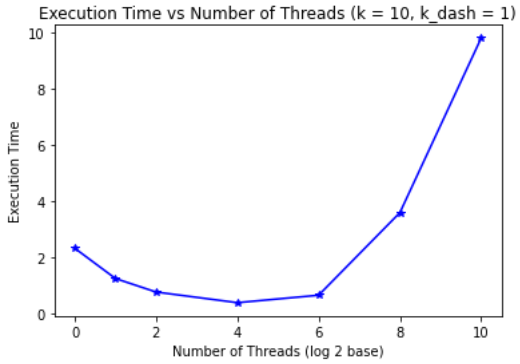
Efficiency vs Number of Threads plot:



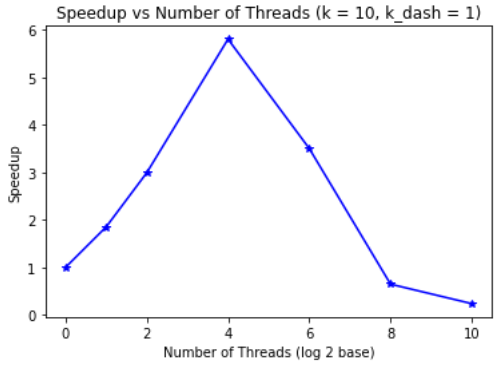
k = 10, k' = 1 (1024 x 1024 matrix):

```
[namith03@grace1 ~]$ cat stra_10_1
Correct: k=10, Size of Matrix = 1024 X 1024, base= 1, number of threads = 1, Execution time for strassen =   2.3305 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 1, number of threads = 2, Execution time for strassen =   1.2582 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 1, number of threads = 4, Execution time for strassen =   0.7786 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 1, number of threads = 16, Execution time for strassen =   0.4014 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 1, number of threads = 64, Execution time for strassen =   0.6652 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 1, number of threads = 256, Execution time for strassen =   3.5955 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 1, number of threads = 1024, Execution time for strassen =   9.7880 sec
```

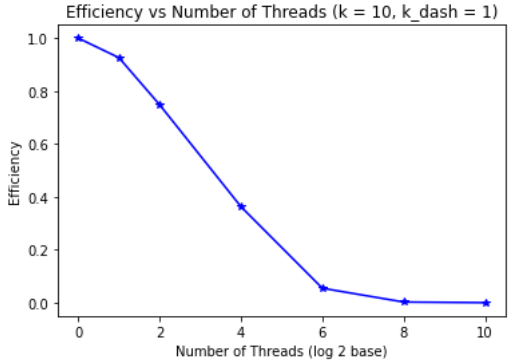
Execution Time vs Number of Threads Plot:



Speedup vs Number of Threads Plot:



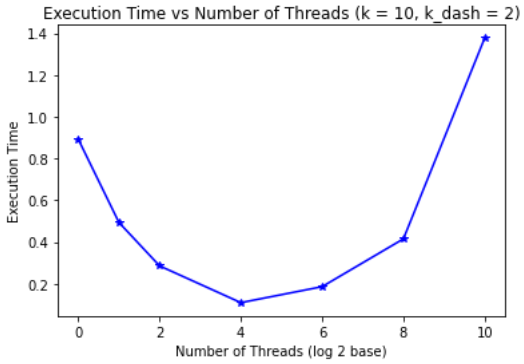
Efficiency vs Number of Threads Plot:



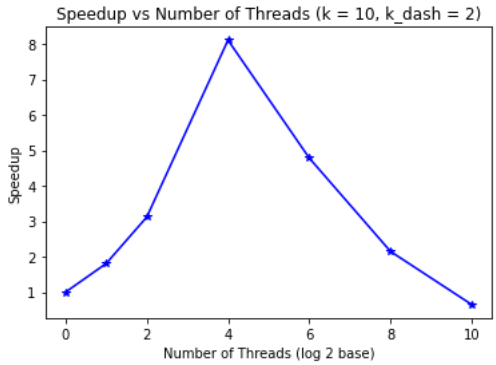
k = 10, k' = 2 (1024 x 1024 matrix):

```
[namith03@grace1 ~]$ cat stra_10_2
Correct: k=10, Size of Matrix = 1024 X 1024, base= 2, number of threads = 1, Execution time for strassen =   0.8918 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 2, number of threads = 2, Execution time for strassen =   0.4938 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 2, number of threads = 4, Execution time for strassen =   0.2861 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 2, number of threads = 16, Execution time for strassen =   0.1100 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 2, number of threads = 64, Execution time for strassen =   0.1865 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 2, number of threads = 256, Execution time for strassen =   0.4152 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 2, number of threads = 1024, Execution time for strassen =   1.3761 sec
```

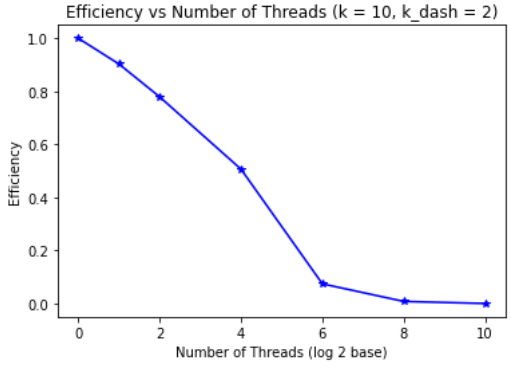
Execution Time Vs Number of Threads Plot:



Speedup Vs Number of Threads Plot:



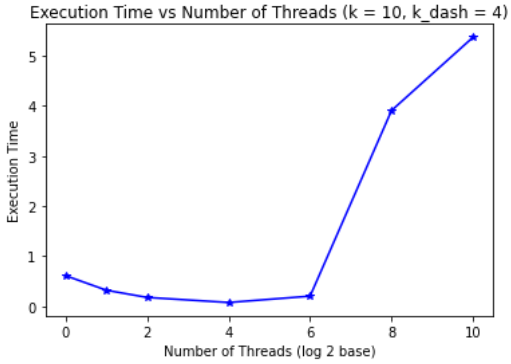
Efficiency Vs Number of Threads Plot:



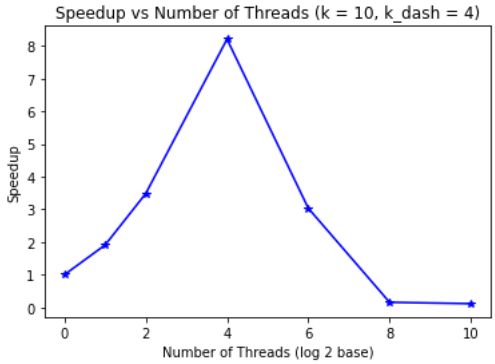
k = 10, k' = 4 (1024 x 1024 matrix):

```
[namith03@grace1 ~]$ cat stra_10_4
Correct: k=10, Size of Matrix = 1024 X 1024, base= 4, number of threads = 1, Execution time for strassen =   0.6091 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 4, number of threads = 2, Execution time for strassen =   0.3191 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 4, number of threads = 4, Execution time for strassen =   0.1749 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 4, number of threads = 16, Execution time for strassen =   0.0742 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 4, number of threads = 64, Execution time for strassen =   0.2008 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 4, number of threads = 256, Execution time for strassen =   3.9063 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 4, number of threads = 1024, Execution time for strassen =   5.3680 sec
```

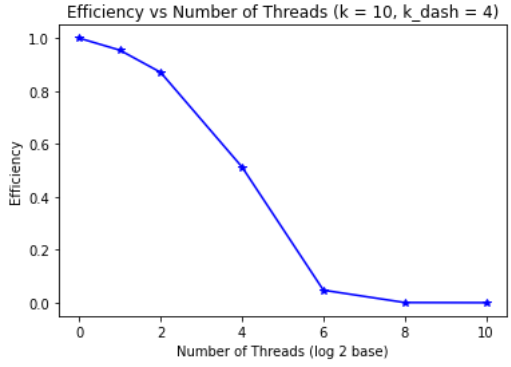
Execution Time Vs Number of Threads Plot:



SpeedUp Vs Number of Threads Plot:



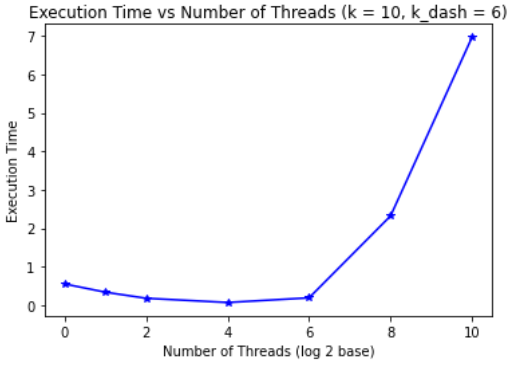
Efficiency Vs Number of Threads Plot:



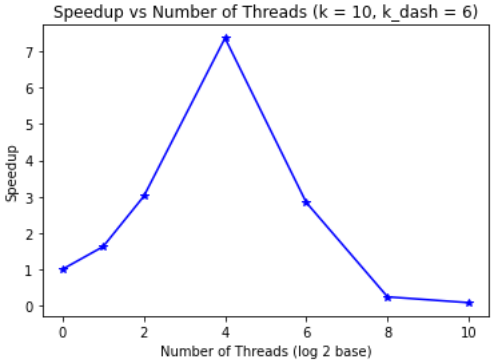
k = 10, k' = 6 (1024 x 1024 matrix):

```
[namith03@grace1 ~]$ cat stra_10_6
Correct: k=10, Size of Matrix = 1024 X 1024, base= 6, number of threads = 1, Execution time for strassen =   0.5562 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 6, number of threads = 2, Execution time for strassen =   0.3426 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 6, number of threads = 4, Execution time for strassen =   0.1849 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 6, number of threads = 16, Execution time for strassen =   0.0755 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 6, number of threads = 64, Execution time for strassen =   0.1959 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 6, number of threads = 256, Execution time for strassen =   2.3217 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 6, number of threads = 1024, Execution time for strassen =   6.9595 sec
```

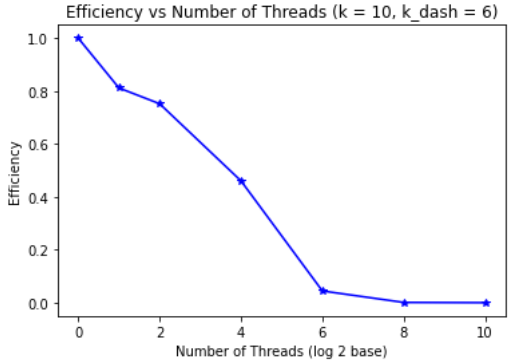
Execution Time Vs Number of Threads Plot:



Speedup Vs Number of Threads Plot:



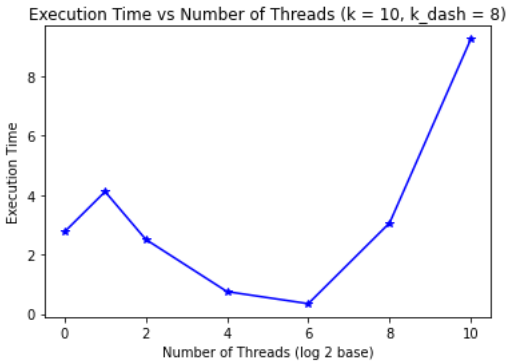
Efficiency vs Number of Threads Plot:



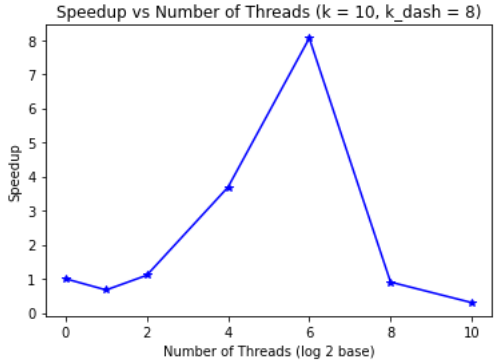
k = 10, k' = 8 (1024 x 1024 matrix):

```
[namith03@grace1 ~]$ cat stra_10_8
Correct: k=10, Size of Matrix = 1024 X 1024, base= 8, number of threads = 1, Execution time for strassen =   2.7567 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 8, number of threads = 2, Execution time for strassen =   4.1125 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 8, number of threads = 4, Execution time for strassen =   2.5049 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 8, number of threads = 16, Execution time for strassen =   0.7480 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 8, number of threads = 64, Execution time for strassen =   0.3422 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 8, number of threads = 256, Execution time for strassen =   3.0571 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 8, number of threads = 1024, Execution time for strassen =   9.2440 sec
```

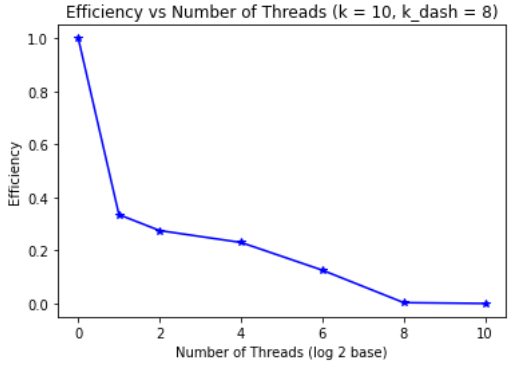
Execution Time Vs Number of Threads Plot:



Speedup Vs Number of Threads Plot:



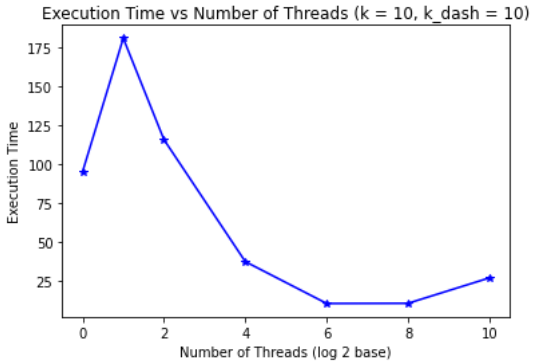
Efficiency vs Number of Threads Plot:



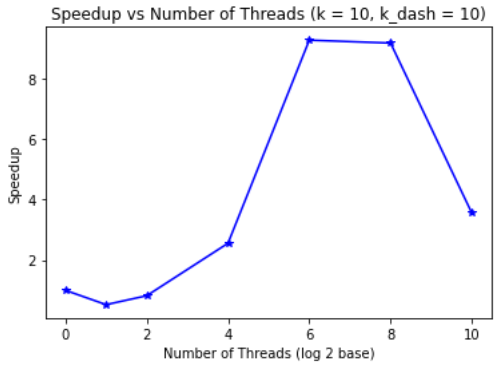
k = 10, k' = 10 (1024 x 1024 matrix):

```
[namith03@grace1 ~]$ cat stra_10_10
Correct: k=10, Size of Matrix = 1024 X 1024, base= 10, number of threads = 1, Execution time for strassen =  95.0358 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 10, number of threads = 2, Execution time for strassen = 180.8197 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 10, number of threads = 4, Execution time for strassen = 115.6494 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 10, number of threads = 16, Execution time for strassen =  37.1610 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 10, number of threads = 64, Execution time for strassen =  10.2457 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 10, number of threads = 256, Execution time for strassen =  10.3534 sec
Correct: k=10, Size of Matrix = 1024 X 1024, base= 10, number of threads = 1024, Execution time for strassen =  26.7182 sec
```

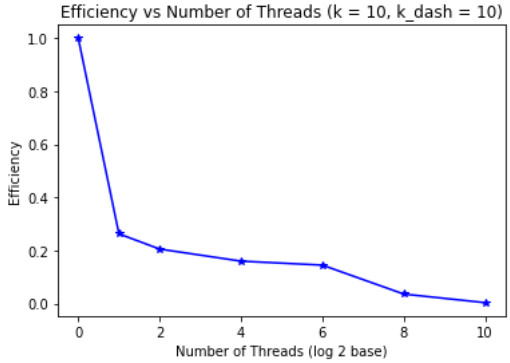
Execution Time Vs Number of Threads Plot:



Speedup Vs Number of Threads Plot:



Efficiency vs Number of Threads Plot:



Analysis and Insights regarding the parallel performance of the code:

It can be observed that for all values of k = 2 and respective k' values, the best speedup and efficiency is observed using 4 threads. For k = 4 and following k' values, the best speedup is obtained using 4 threads. k = 6 with k' = 1,2,4 also showed similar trends as k = 4 for speedup , but for k ' = 6, best speedup is achieved using 16 threads. The best speedup is obtained using 16 threads for all values of k' for k = 8. Finally, for k = 10, the best speedup is observed using 16 threads for k' = 1,2,4,6 whereas the best speedup for k' = 8,10 is observed using 64 threads.

The worst speedup and execution time is observed using 256 and 1024 threads in most of the combinations of k and k' respectively.

From the above observations, we can conclude that as the size of the matrix and terminal matrix grows, more threads are required to achieve a good speedup. But beyond a certain number of threads, the overhead associated with scheduling them results in a significant drop in speedup.

Effect of Terminal matrix size:

Here, the size of the terminal matrix is $n/2^{k'}$ where n is the size of the input matrix and k' is the level after which we terminate the recursion. After terminating the recursion, we compute naïve matrix multiplications for the terminal matrices.

If k' = 2, it means that we terminate recursion after the first level. For e.g., for k = 6 and k'= 2, recursion is terminated after the 2^{nd} level and terminal matrices of size 4 x 4 onwards are computed with the naïve method of matrix multiplication.

From the results, it can be observed that

(Comparing the results using a number of threads as 4 for a fair comparison)

- For k = 2 i.e. input matrix size of 4 x 4, at k ' = 1 which refers to terminal matrix size of 2 x 2, we observe the best performance
- For k = 4 i.e. input matrix size of 16 x 16, at k ' = 2 which refers to terminal matrix size of 4 x 4, we observe the best performance.
- For k = 6 i.e. input matrix size of 64 x 64, at k ' = 2 which refers to terminal matrix size of 16 x 16, we observe the best performance.
- For k = 8 i.e. input matrix size of 256 x 256, at k ' = 4 which refers to terminal matrix size of 16 x 16, we observe the best performance.
- For k = 10 i.e. input matrix size of 1024 x 1024, at k ' = 6 which refers to terminal matrix size of 16 x 16, we observe the best performance.
- It can be observed that for k = k', it is showing the worst performance in comparison with other values of k'.

From the above results, it can be observed that Strassen's algorithm shows the best results at approximately k = k'/2 for k = 2,4,6,8,10 and the worst performance at k = k' for the experimented values of k.

For considerably large values of k, when k = k', the naïve multiplication method is computed at the last level which occurs after a lot of recursion calls and at this point there is no use in taking advantage of the terminal matrices because it does not terminate until the last moment.

It is very much ideal to terminate at the size of the terminal matrix where the naïve matrix multiplication does not take significant time and can save the execution time incurred by additional recursion calls. This happens at k' = k/2 for the input sizes considered of k = 2,4,6,8, 10 and the terminal matrix of size $2^{k/2}$ x $2^{k/2}$ in our case considered does not take much execution time for the naïve method compared to the overhead caused by recursion calls. But this cannot be taken as a standard, because for immense matrices, it wouldn't be ideal to terminate to half size and compute naïve multiplication because the naïve method might prove costly for the terminal matrices of a bigger size.

Design Choices:

C++ and OpenMPI are chosen for implementing Strassen because of having the best familiarity and ease of implementation with the respective languages.

The design choice of code that must be parallelized and the part that should be run in serial is made because of the following reasons.

Code that is parallelized:

- In computing the result for Strassen, the calculations for M1, M2, M3, M4, M5, M6, and M7 are expensive and can also be done independently. Since they do not have any dependency, all the functions that calculate M1 through M7 can be run parallelly.

Code that runs in serial:

- There is a function that allocates memory to the submatrices which will be storing the result of addition and subtraction. These matrices are created for every recursion call, hence this should run in serial.

- The function that creates a memory for M1 through M7 is also run serially.