ICS-E4020 Programming Parallel Computers - Assignment 1

Darshan M S (465687)
darshan.mallenahallishankaralingappa@aalto.fi

April 19, 2015

1 Introduction

In this problem we had to find the median for every pixel component in the image. The brute force approach using sorting provided very bad results. But with the use of quick select algorithm we found that the algorithm improved.

The benchmark values (in secs) obtained when the brute force approach was used is given below:

Size of the image \ Size of the window	1	2	5	10
100	0.003	0.007	0.016	0.035
200	0.005	0.013	0.047	0.144
500	0.026	0.074	0.28	0.907
1000	0.102	0.295	1.119	3.647
2000	0.416	1.17	4.47	14.609

The plot is given below

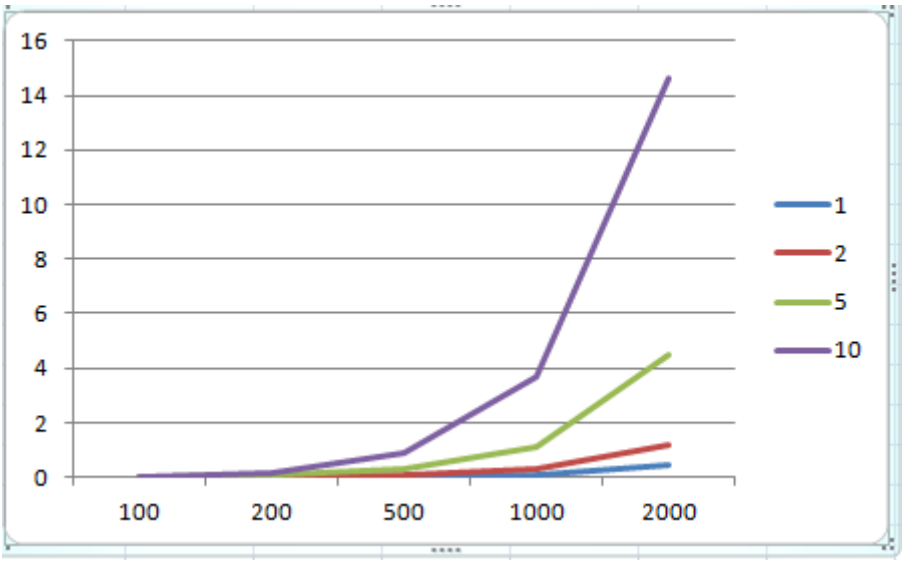

Figure 1: Brute Force Approach - Plot of size of the image vs time required to execute(in secs). Different lines represent different window sizes.

2 Parallel Execution.

The for loops in the program can be executed in parallel using the #pragma omp parallel for command in C++ programming language. One other important thing to note was to not modify the global variables in the parallel threads.

The benchmark time values for number of threads = 1 is same as the brute force approach.

The benchmark values for number of threads = 2 is given below:

Size of the image \ Size of the window	1	2	5	10
100	0.001	0.002	0.007	0.018
200	0.002	0.006	0.024	0.074
500	0.014	0.038	0.147	0.467
1000	0.054	0.152	0.584	1.871
2000	0.217	0.612	2.333	7.5

The values are plotted below:

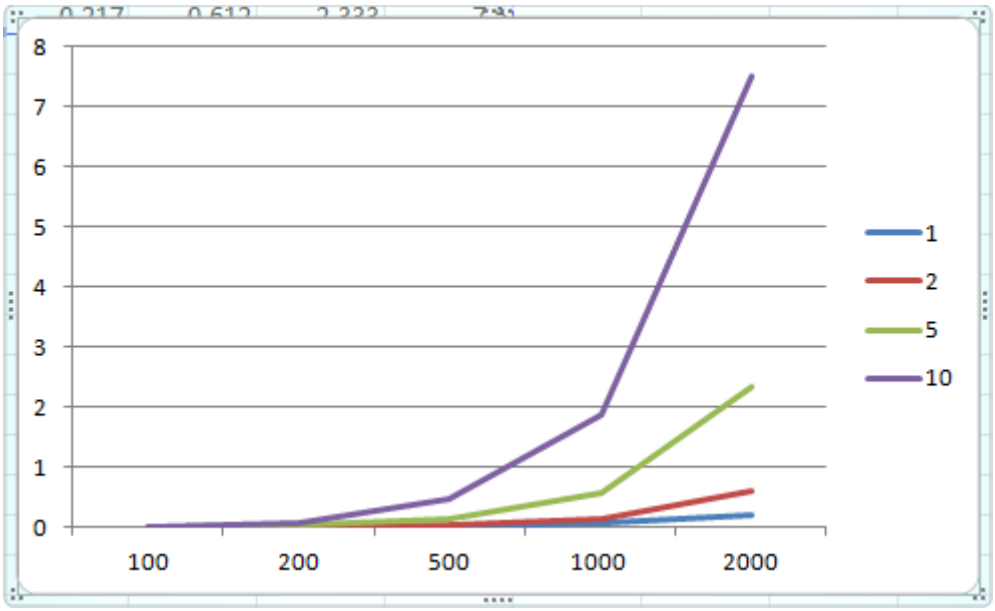

Figure 2: No of threads = 2 - Plot of size of the image vs time required to execute(in secs). Different lines represent different window sizes.

We can see that the time has decreases by a factor of 2 when compared to the brute force approach.

The benchmark values for number of threads = 4 is given below:

Size of the image \ Size of the window	1	2	5	10
100	0.001	0.002	0.006	0.017
200	0.002	0.005	0.017	0.05
500	0.011	0.029	0.091	0.263
1000	0.04	0.103	0.33	0.994
2000	0.114	0.325	1.272	3.973

The values are plotted below:

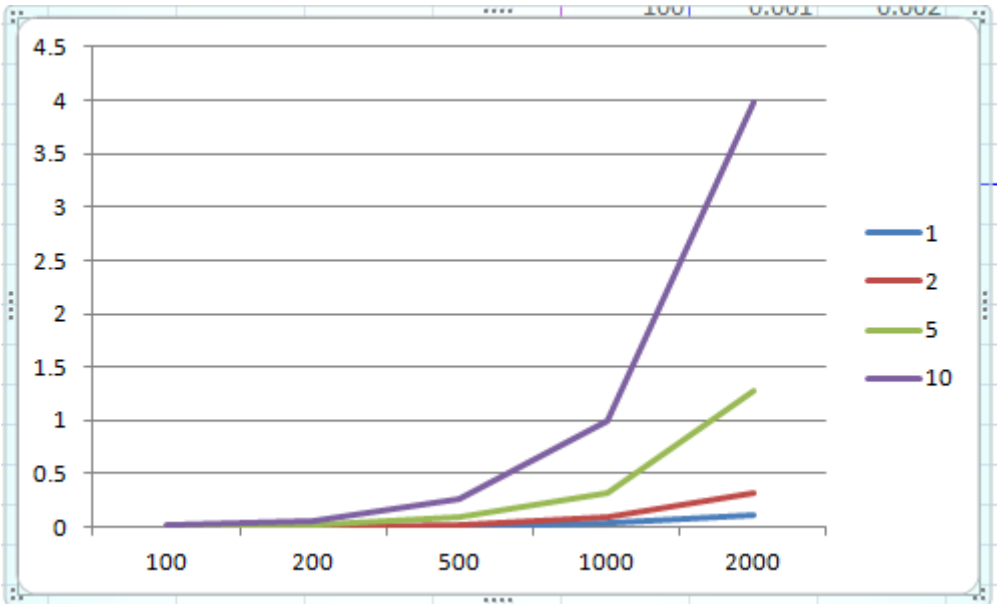

Figure 3: No of threads = 4 - Plot of size of the image vs time required to execute(in secs). Different lines represent different window sizes.

We can see that the time has decreases almost by a factor of 4 on an average when compared to the brute force approach.

The benchmark values for number of threads = 8 is given below:

Size of the image \ Size of the window	1	2	5	10
100	0.001	0.002	0.006	0.015
200	0.003	0.006	0.019	0.027
500	0.006	0.014	0.055	0.167
1000	0.022	0.056	0.209	0.66
2000	0.083	0.222	0.834	2.643

The values are plotted below:

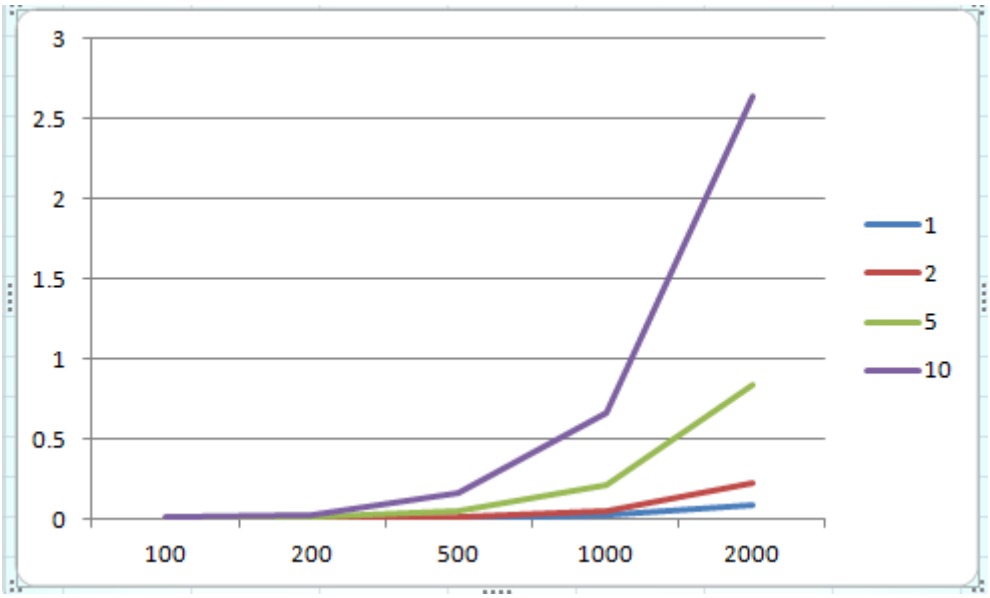

Figure 4: No of threads = 8 - Plot of size of the image vs time required to execute(in secs). Different lines represent different window sizes.

We can see that the time has decreases almost by a factor of 8 on an average when compared to the brute force approach.

We can clearly see that for larger data sets, parallelizing the operations can really improve the performance.