ICS-4020 Programming Parallel Computers
Week 1

Kristian Hartikainen (222956)
kristian.hartikainen@aalto.fi

April 19, 2015

1 Introduction

I implemented the median filtering function by using simple C code, with one addittional C++ function to calculate the actual medians. The task itself is pretty straight forward, and the code is commented such that it should be sufficient as a documentation.

I ran the benchmarks for the task on the classroom computer 'ruokki'. The simple (reasonably efficient) solution for the mf1 runs in $\mathcal{O}(nk^2)$ time, and for example for 1000*1000 pixel image with window width 10 runs in 3.837 seconds. The output for the benchmarks are listed on table 1 and the results plotted in figure 1.

For the mf2, I used simple OpenMP pragmas to parallelize the for loops. Other than adding the parallel for pragmas around the for loops, I only needed to move around the variable declarations such that they will be correctly isolated between the threads. Again the code should be commented enough to be self documentary. The benchmarks for the mf2 solution running on 1, 2, 4 and 8 cores can be found from the tables 2, 3, 4, 5 respectively. The plots for these benchmark are shown in the figures 2 and 3.

For example The 2000*2000 pixel image with window width 10 runs in 17.151 seconds with 1 core, 8.695 seconds with 2 cores, 4.594 seconds with 4 cores and 3.220 seconds with 8 cores. That is, with 8 cores the algorithm runs about 5.33 times (17.151s/3.220s) faster than with a single core. Similarily, the 1000*1000 pixel image with window width 10 runs about 5.27 times (4.279 / 0.811) faster on 8 cores compared to a single core.

2 Results

nx	ny	k	time
100	100	1	0.003
100	100	2	0.007
100	100	5	0.027
100	100	10	0.043
200	200	1	0.005
200	200	2	0.013
200	200	5	0.048
200	200	10	0.151
500	500	1	0.030
500	500	2	0.078
500	500	5	0.296
500	500	10	0.954
1000	1000	1	0.120
1000	1000	2	0.313
1000	1000	5	1.186
1000	1000	10	3.837
2000	2000	1	0.478
2000	2000	2	1.252
2000	2000	5	4.751
2000	2000	10	15.425

Table 1: Benchmarks for the exercise mf1

nx	ny	k	time
100	100	1	0.012
100	100	2	0.017
100	100	5	0.023
100	100	10	0.041
200	200	1	0.021
200	200	2	0.029
200	200	5	0.063
200	200	10	0.168
500	500	1	0.128
500	500	2	0.179
500	500	5	0.390
500	500	10	1.064
1000	1000	1	0.510
1000	1000	2	0.716
1000	1000	5	1.562
1000	1000	10	4.279
2000	2000	1	2.044
2000	2000	2	2.874
2000	2000	5	6.250
2000	2000	10	17.151

Table 2: Benchmarks for the exercise mf2 running on 1 core

nx	ny	k	t
100	100	1	0.006
100	100	2	0.009
100	100	5	0.017
100	100	10	0.021
200	200	1	0.012
200	200	2	0.016
200	200	5	0.033
200	200	10	0.086
500	500	1	0.072
500	500	2	0.099
500	500	5	0.202
500	500	10	0.542
1000	1000	1	0.288
1000	1000	2	0.401
1000	1000	5	0.806
1000	1000	10	2.166
2000	2000	1	1.155
2000	2000	2	1.562
2000	2000	5	3.227
2000	2000	10	8.695

Table 3: Benchmarks for the exercise mf2 running on 2 cores

nx	ny	k	time
100	100	1	0.003
100	100	2	0.003
100	100	5	0.005
100	100	10	0.012
200	200	1	0.007
200	200	2	0.009
200	200	5	0.019
200	200	10	0.061
500	500	1	0.060
500	500	2	0.082
500	500	5	0.148
500	500	10	0.288
1000	1000	1	0.172
1000	1000	2	0.299
1000	1000	5	0.599
1000	1000	10	1.540
2000	2000	1	0.632
2000	2000	2	0.849
2000	2000	5	1.716
2000	2000	10	4.594

Table 4: Benchmarks for the exercise mf2 running on 4 cores

nx	ny	k	time
100	100	1	0.004
100	100	2	0.002
100	100	5	0.004
100	100	10	0.009
200	200	1	0.006
200	200	2	0.007
200	200	5	0.016
200	200	10	0.055
500	500	1	0.050
500	500	2	0.064
500	500	5	0.082
500	500	10	0.224
1000	1000	1	0.130
1000	1000	2	0.176
1000	1000	5	0.331
1000	1000	10	0.811
2000	2000	1	0.530
2000	2000	2	0.650
2000	2000	5	1.236
2000	2000	10	3.220

Table 5: Benchmarks for the exercise mf2 running on 8 cores

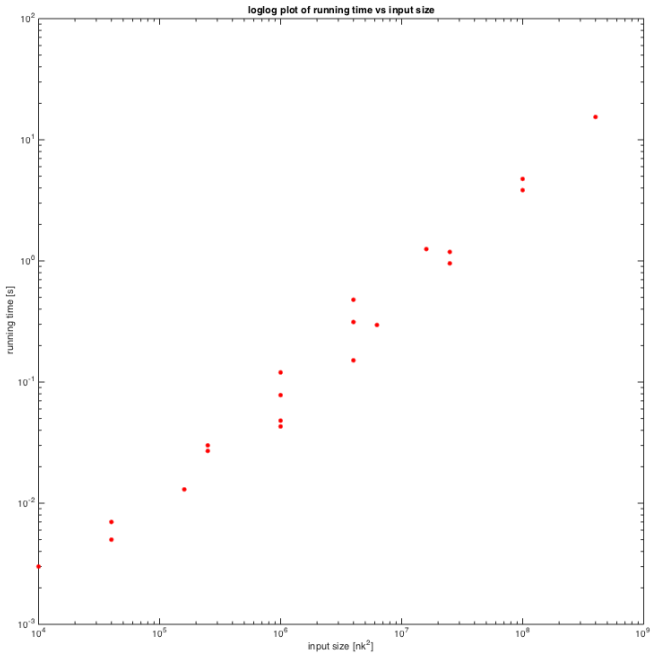

Figure 1: loglog plot of the running time vs the input size

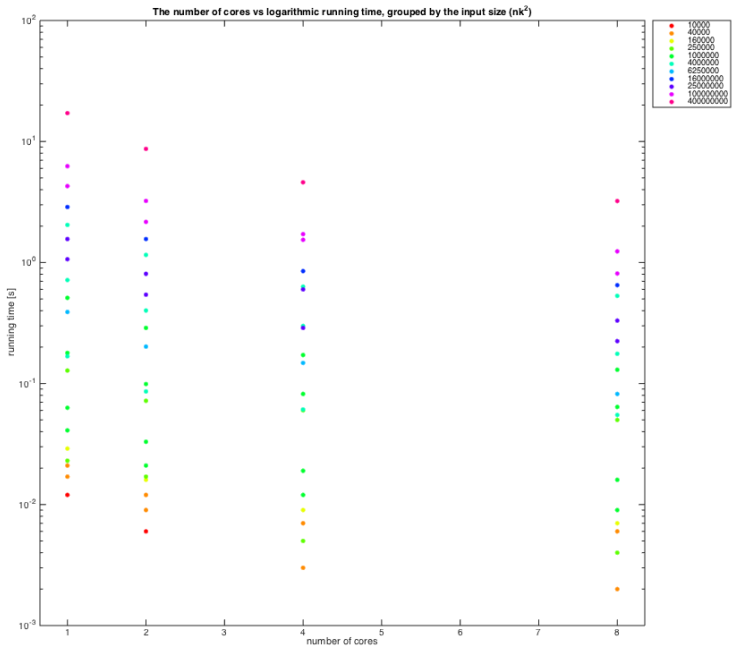

Figure 2: The number of the cores vs logarithmic running time, grouped by the input size

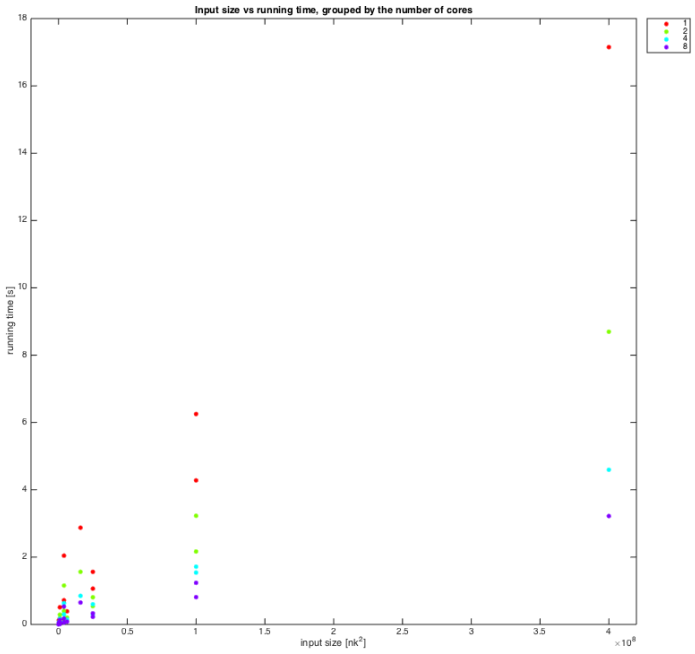

Figure 3: Input size vs running time, grouped by the number of the cores