# Thread Pool Mandelbrot

Philip Nelson

2017 February 4th

## Hypothesis

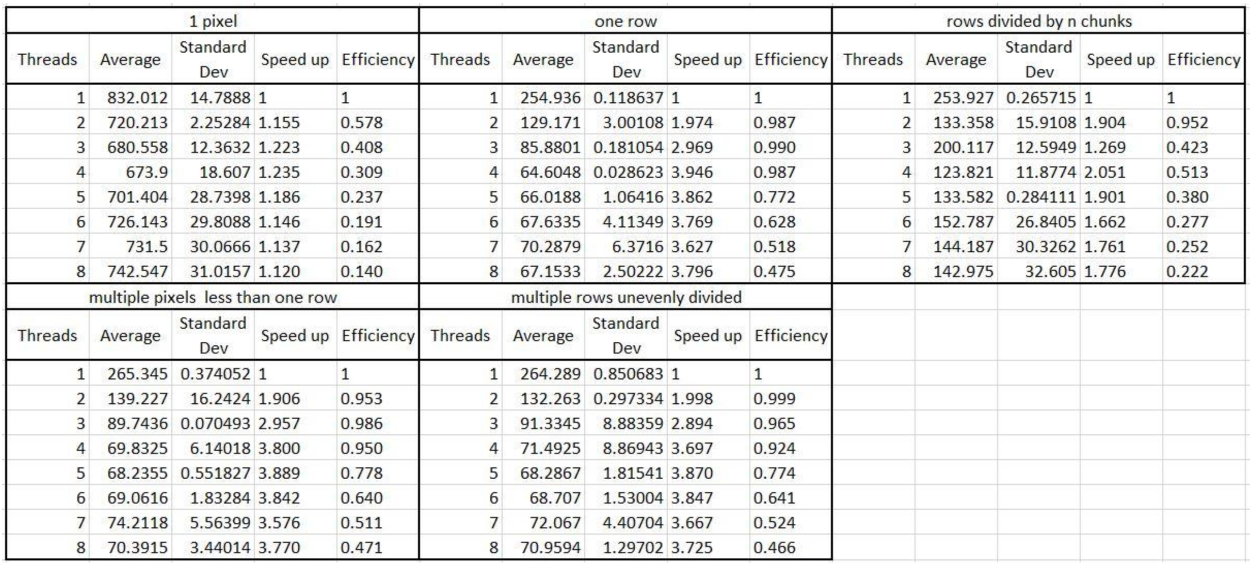
As the program is divided up and pieces are given to different threads to execute, we should see a speed up until the number of threads exceeds the number of cores available on the CPU.

## Process

For a given number of threads, the Mandelbrot set image was rendered 10 times. The average time and standard deviation were calculated for each. The set was rendered with one to nine threads as shown in figure 1

## Findings

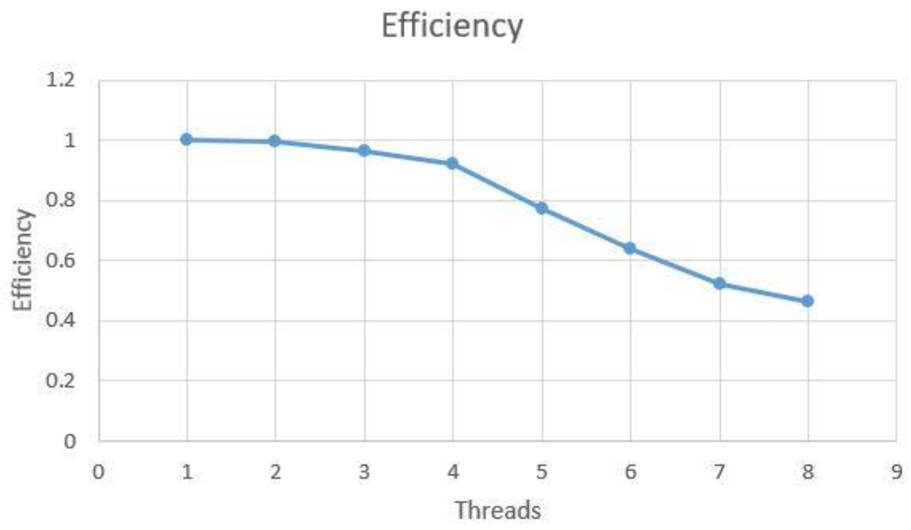
The Mandelbrot set was calculated with a single thread up to eight threads and as more threads were used interesting results were encountered. One thing was that the drop in performance was removed except when dividing by n chunks. Also one row at a time, multiple pixels less than one row and uneven rows had a linear speed up until 5 threads.

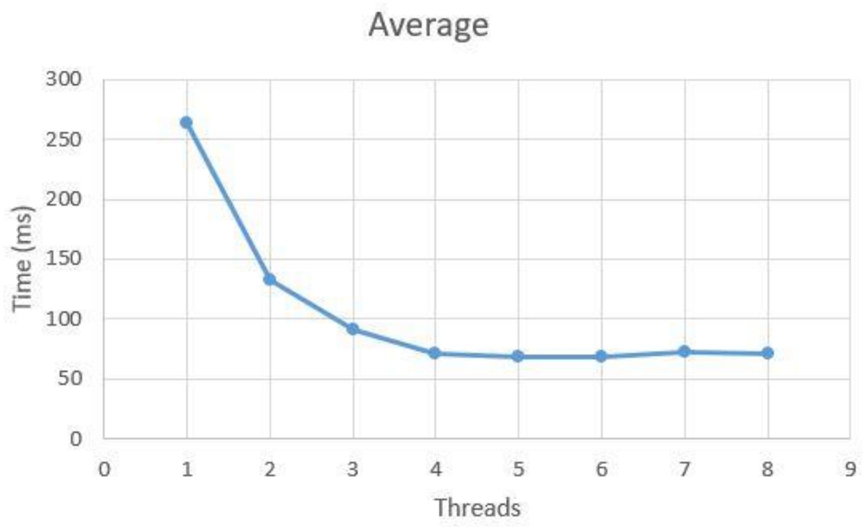


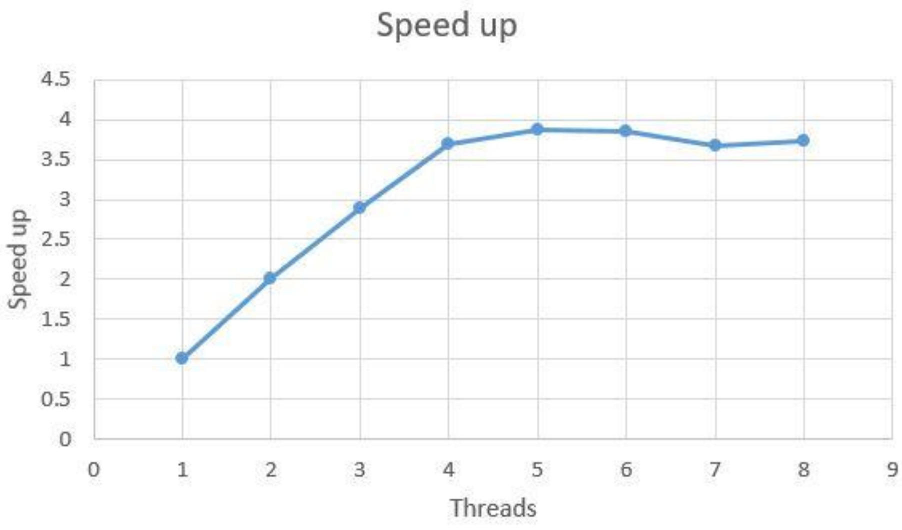
| 1 pixel | | | | | one row | | | | | rows divided by n chunks | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Threads | Average | Standard Dev | Speed up | Efficiency | Threads | Average | Standard Dev | Speed up | Efficiency | Threads | Average | Standard Dev | Speed up | Efficiency |
| 1 | 832.012 | 14.7888 | 1 | 1 | 1 | 254.936 | 0.118637 | 1 | 1 | 1 | 253.927 | 0.265715 | 1 | 1 |
| 2 | 720.213 | 2.25284 | 1.155 | 0.578 | 2 | 129.171 | 3.00108 | 1.974 | 0.987 | 2 | 133.358 | 15.9108 | 1.904 | 0.952 |
| 3 | 680.558 | 12.3632 | 1.223 | 0.408 | 3 | 85.8801 | 0.181054 | 2.969 | 0.990 | 3 | 200.117 | 12.5949 | 1.269 | 0.423 |
| 4 | 673.9 | 18.607 | 1.235 | 0.309 | 4 | 64.6048 | 0.028623 | 3.946 | 0.987 | 4 | 123.821 | 11.8774 | 2.051 | 0.513 |
| 5 | 701.404 | 28.7398 | 1.186 | 0.237 | 5 | 66.0188 | 1.06416 | 3.862 | 0.772 | 5 | 133.582 | 0.284111 | 1.901 | 0.380 |
| 6 | 726.143 | 29.8088 | 1.146 | 0.191 | 6 | 67.6335 | 4.11349 | 3.769 | 0.628 | 6 | 152.787 | 26.8405 | 1.662 | 0.277 |
| 7 | 731.5 | 30.0666 | 1.137 | 0.162 | 7 | 70.2879 | 6.3716 | 3.627 | 0.518 | 7 | 144.187 | 30.3262 | 1.761 | 0.252 |
| 8 | 742.547 | 31.0157 | 1.120 | 0.140 | 8 | 67.1533 | 2.50222 | 3.796 | 0.475 | 8 | 142.975 | 32.605 | 1.776 | 0.222 |

| multiple pixels less than one row | | | | | multiple rows unevenly divided | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Threads | Average | Standard Dev | Speed up | Efficiency | Threads | Average | Standard Dev | Speed up | Efficiency |
| 1 | 265.345 | 0.374052 | 1 | 1 | 1 | 264.289 | 0.850683 | 1 | 1 |
| 2 | 139.227 | 16.2424 | 1.906 | 0.953 | 2 | 132.263 | 0.297334 | 1.998 | 0.999 |
| 3 | 89.7436 | 0.070493 | 2.957 | 0.986 | 3 | 91.3345 | 8.88359 | 2.894 | 0.965 |
| 4 | 69.8325 | 6.14018 | 3.800 | 0.950 | 4 | 71.4925 | 8.86943 | 3.697 | 0.924 |
| 5 | 68.2355 | 0.551827 | 3.889 | 0.778 | 5 | 68.2867 | 1.81541 | 3.870 | 0.774 |
| 6 | 69.0616 | 1.83284 | 3.842 | 0.640 | 6 | 68.707 | 1.53004 | 3.847 | 0.641 |
| 7 | 74.2118 | 5.56399 | 3.576 | 0.511 | 7 | 72.067 | 4.40704 | 3.667 | 0.524 |
| 8 | 70.3915 | 3.44014 | 3.770 | 0.471 | 8 | 70.9594 | 1.29702 | 3.725 | 0.466 |

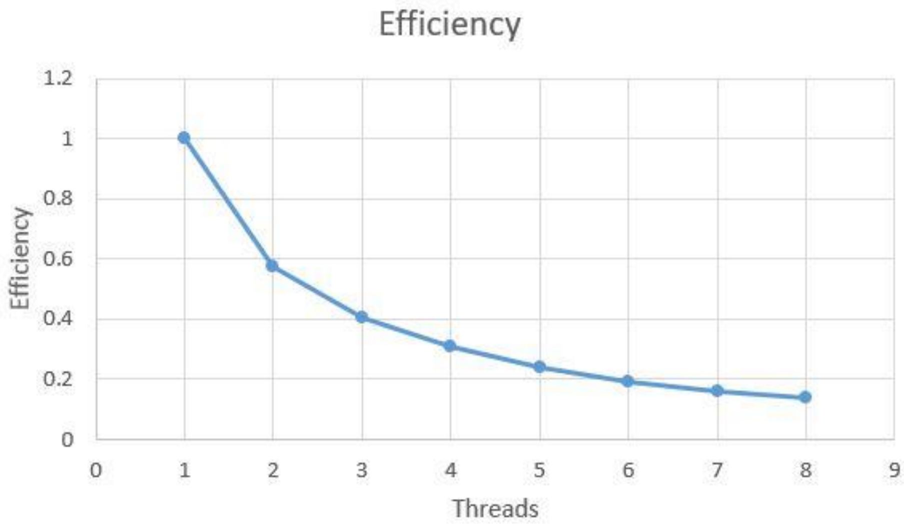
Uneven Rows

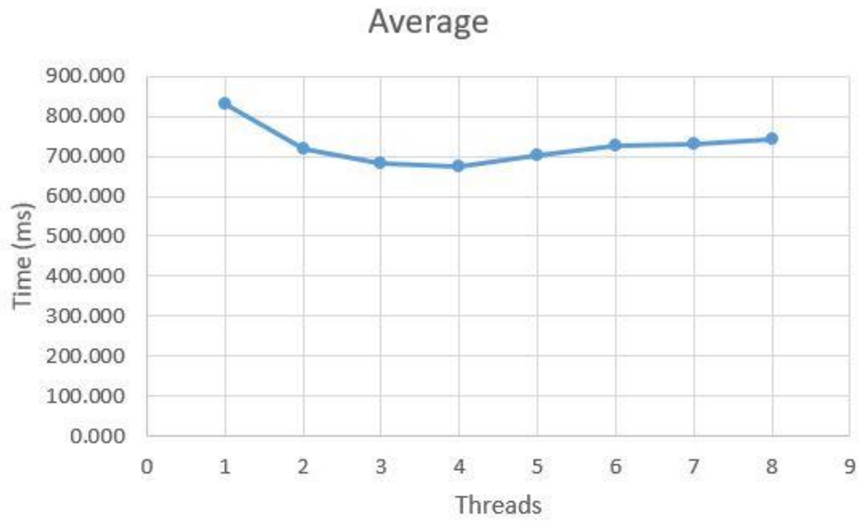

Efficiency
Efficiency
1.2
1
0.8
0.6
0.4
0.2
0
0
1
2
3
4
5
6
7
8
9
Threads

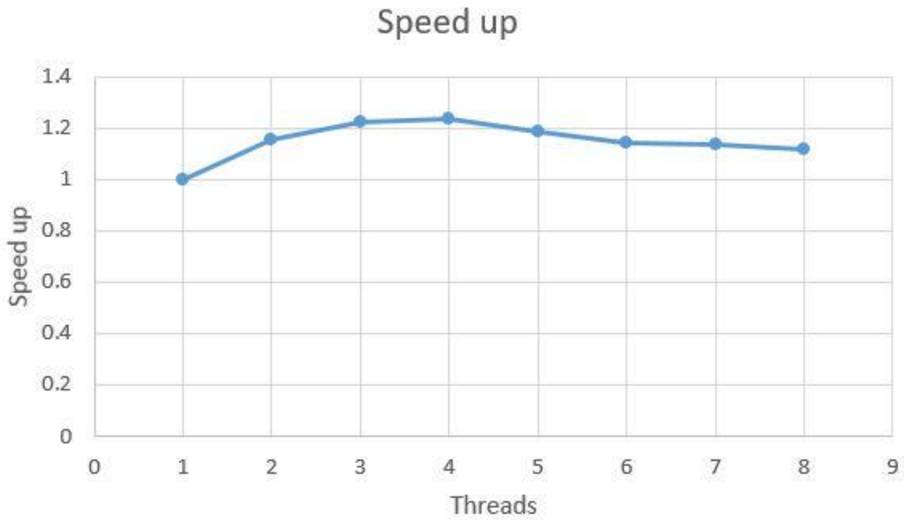

Average
Time (ms)
300
250
200
150
100
50
0
0
1
2
3
4
5
6
7
8
9
Threads


Speed up
Speed up
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
0
1
2
3
4
5
6
7
8
9
Threads

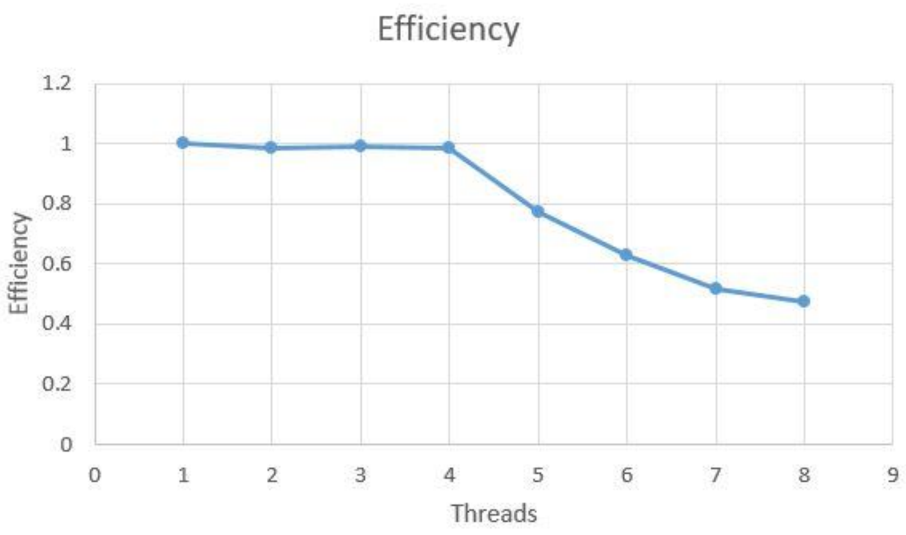
One Pixel

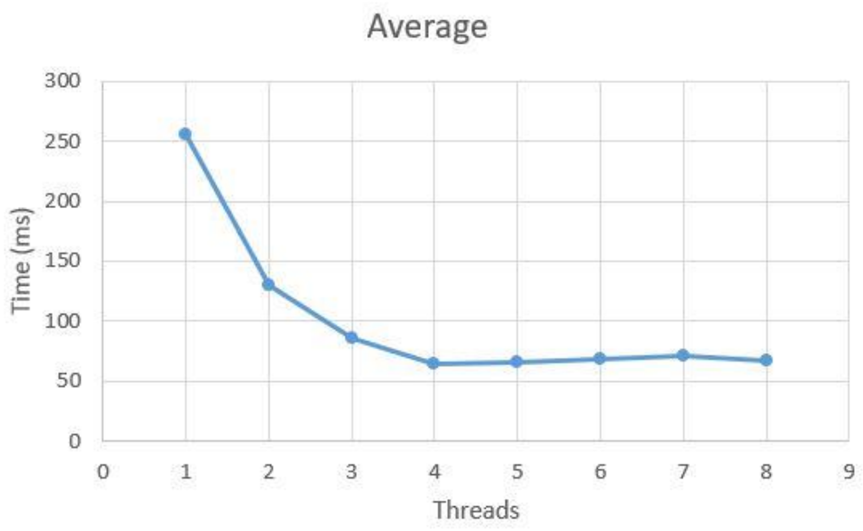

Efficiency
1.2
1
0.8
0.6
0.4
0.2
0
Efficiency
0
1
2
3
4
5
6
7
8
9
Threads

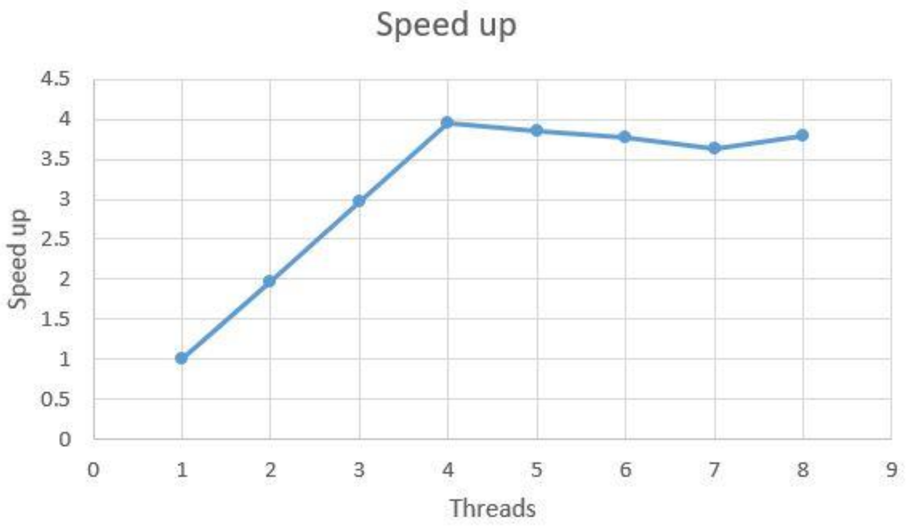

Average
900.000
800.000
700.000
600.000
500.000
400.000
300.000
200.000
100.000
0.000
Time (ms)
0
1
2
3
4
5
6
7
8
9
Threads


Speed up
1.4
1.2
1
0.8
0.6
0.4
0.2
0
Speed up
0
1
2
3
4
5
6
7
8
9
Threads

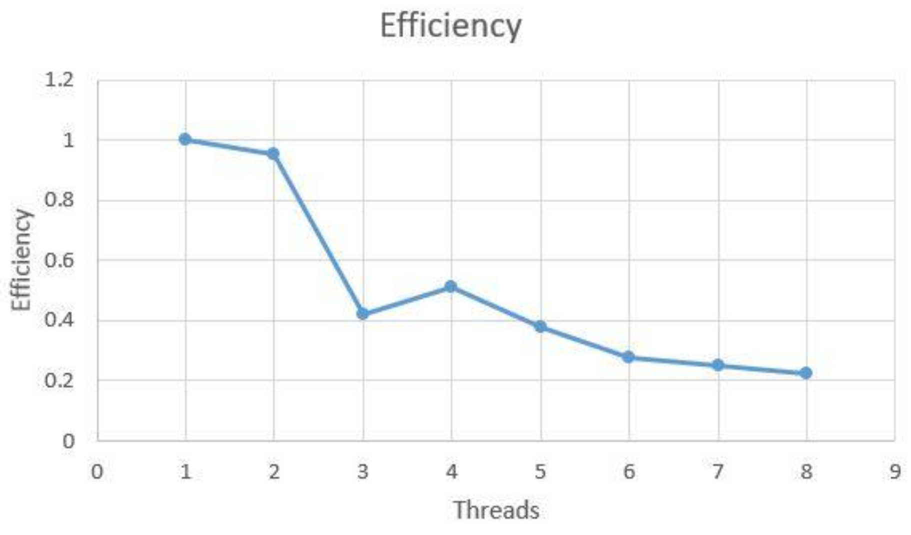
One Row

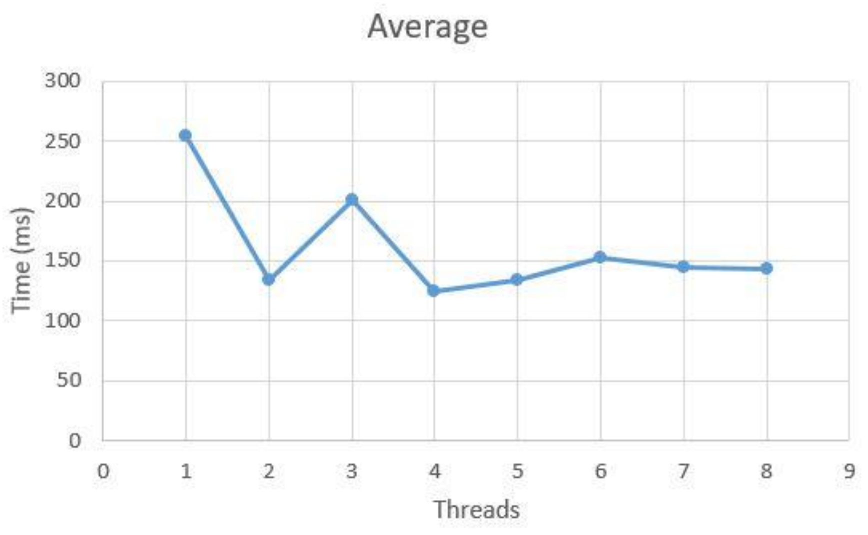

Efficiency
1.2
1
0.8
0.6
0.4
0.2
0
Efficiency
0
1
2
3
4
5
6
7
8
9
Threads

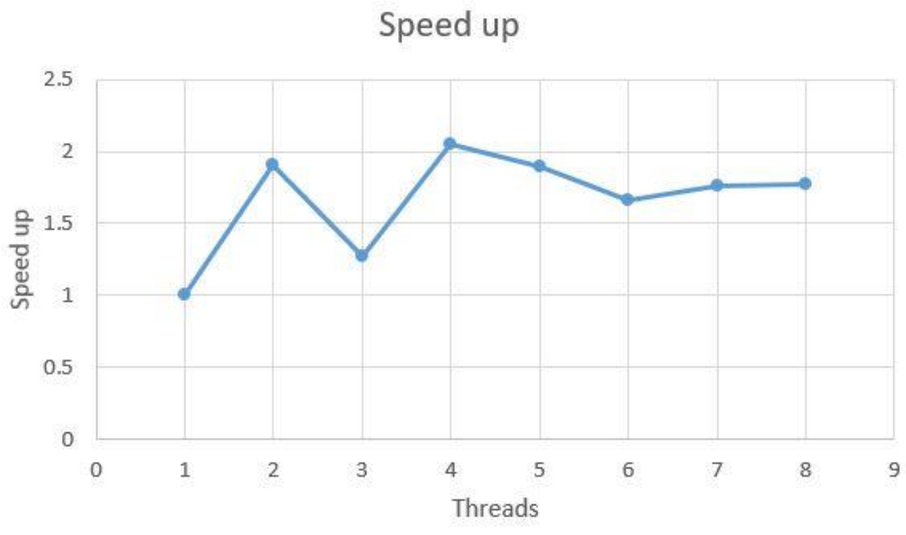

Average
300
250
200
150
100
50
0
Time (ms)
0
1
2
3
4
5
6
7
8
9
Threads


Speed up
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
Speed up
0
1
2
3
4
5
6
7
8
9
Threads

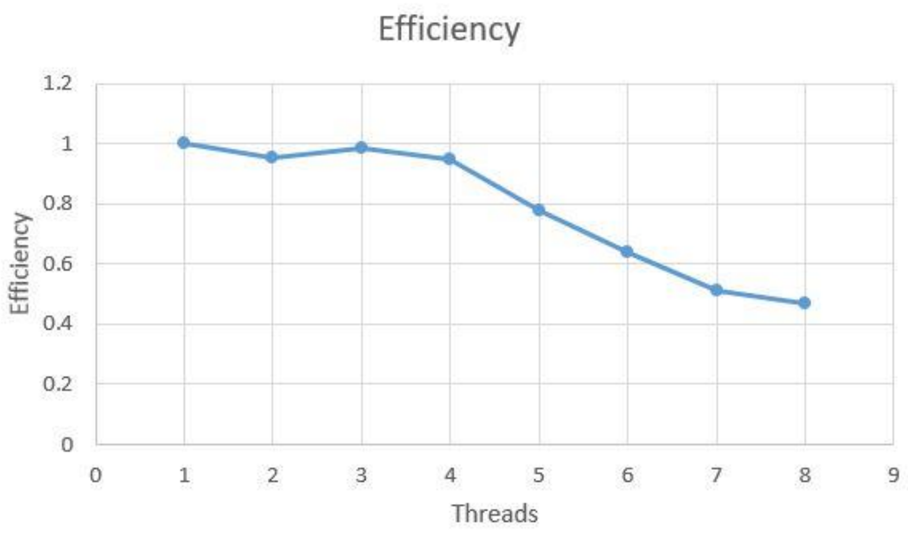
Rows Divided by n Chunks

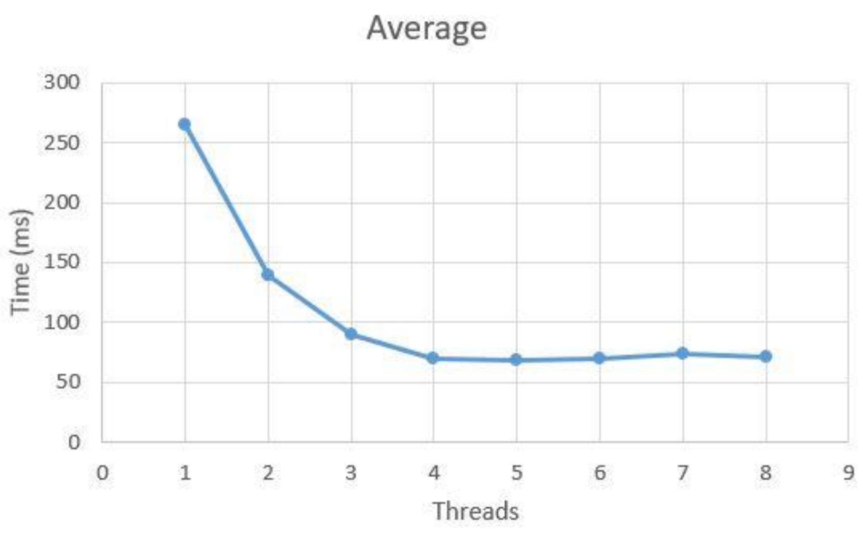


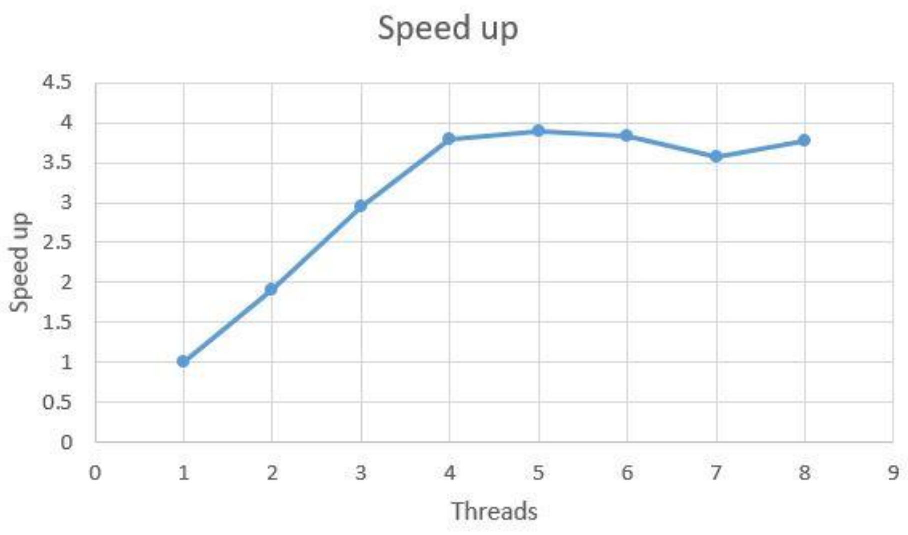




Multiple pixels less than one row


Efficiency
Efficiency
Threads


Average
Time (ms)
Threads


Speed up
Speed up
Threads