Threaded Mandelbrot

Philip Nelson

2017 January 28

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 50 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 nine threads and as more threads were used unexpected results were encountered. First, when three threads were used there was a drop in performance. Second, the best speed ups occurred when eight and nine threads were used figure 4.

Conclusion

I believe that the slower times seen while using three threads was due to the second thread having to handle the middle third of the set which falls mostly inside the Mandelbrot set. This would make the second thread do much more work than the other two. As for obtaining better speed up with eight and nine threads I do not know why this would happen. Possibly there could be an optimization occurring in the CPU taking advantage of idle clock cycles to do more work with more threads.

Data

Threads	Standard Dev	Average	Speed up	Efficiency
1	10.002	256.806	1	1
2	7.673	130.195	1.972	0.986
3	17.766	203.471	1.262	0.421
4	4.818	123.230	2.084	0.521
5	19.118	146.119	1.758	0.352
6	9.092	125.876	2.040	0.340
7	17.979	131.576	1.952	0.279
8	18.547	117.370	2.188	0.274
9	14.582	113.100	2.271	0.252

Figure 1: Data

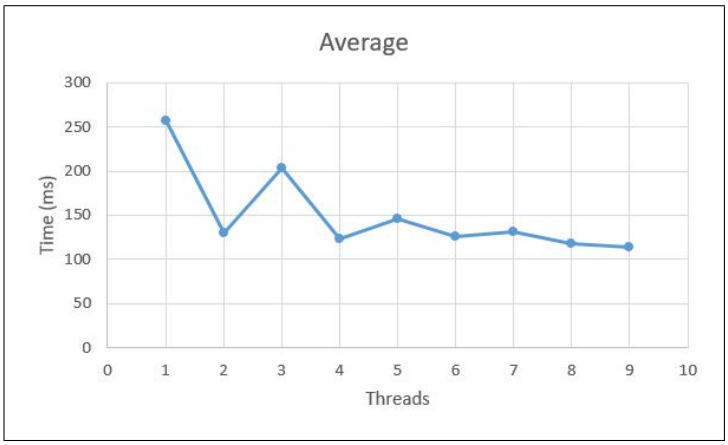

Figure 2: Average Time vs. Threads

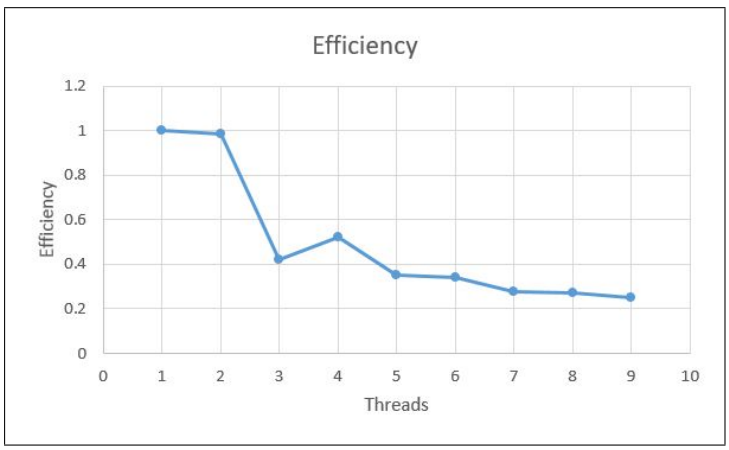

Figure 3: Efficiency vs. Threads

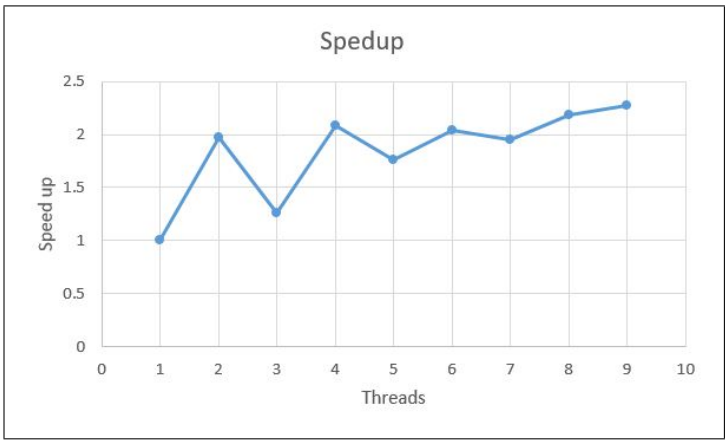

Figure 4: Speed Up vs. Threads