Exploration of K-Means With Different Multi-Threading Approaches

Jake Najera

CS355-Fall20

4/3/20

In this document, I explore three different implementations of K-Means Clustering, single-threaded, multi-threaded centroid updating, and parallization with workers.

1 The Problem

K-Means Clustering is a method of unsupervised machine learning most commonly used to categorize ‘N’ data samples, using ‘C’ centroids, until reaching a point of convergence or thousands of iterations have been completed. First step of the algorithm is assignment of samples to a centroid, then the second step is updating the coordinates of a centroid. Problem may arise in how long an algorithm takes to complete a given number of iterations. How can multi-threading assist in getting faster completion?

2 Experiment Setup

To generate ‘N’ 2D samples, each sample in the array is randomly generated using a Mersenne-Twister engine to supply doubles in the range [-1000, 1000]. ‘N’ was between [1,000 : 20,000] with a step of 1,000. For each ‘N`, I tested with all 3 algorithms for ‘C’ clusters from [2:10]. Number of threads were the same as number of clusters when applicable. The following compares the algorithms with ‘C’ clusters.

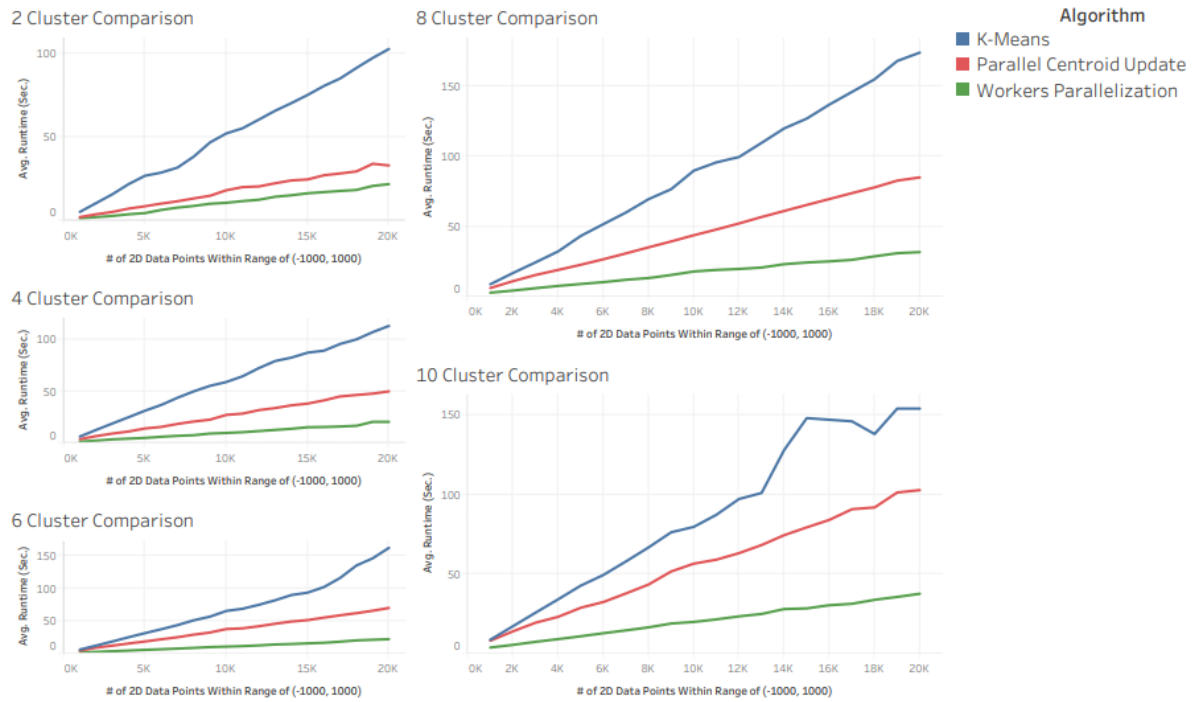

Figure 1: Comparing performance with 'C' clusters

3 Cluster Comparison Graphs' Analysis

We can see across all 5 graphs that regular K-Means is most time intensive of the three, as expected. Parallel centroid updating reduces the time by a little more than half. Workers parallelization reduce that time further being the fastest of all three. It seems that K-Means and Parallel Centroid Update scale noticeably, while Workers Parallelization seemed to only scale a little. One may note that at 10 clusters, K-Means with 10k or more points becomes inconsistent unlike the previous experiments.

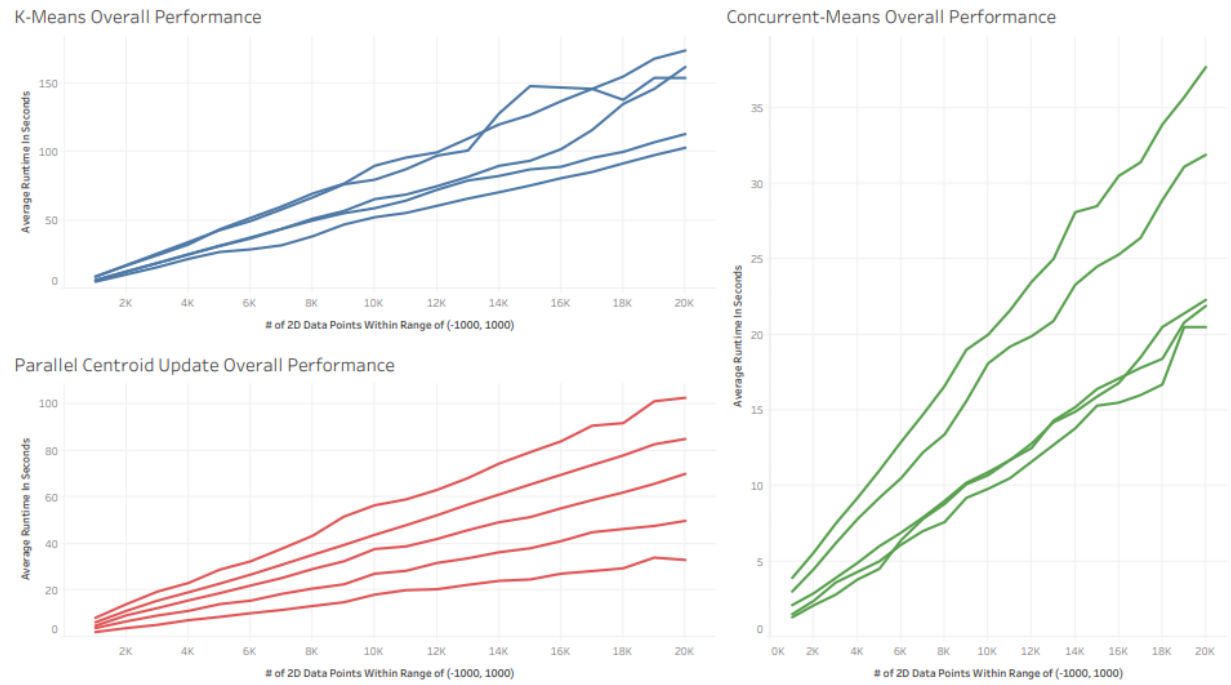

Figure 2: Comparison between algorithms, note difference in the Y-axis

4 Algorithm Comparison Graphs' Analysis

K-Means overall is a slow algorithm, what a surprise. It scales in slowness as more clusters are used. Not consistent, by observing the lines/trends. Slowest time was at 160+ seconds.

Parallel Centroid Update is an improvement upon the base algorithm by performing step two multi-threadedly. We can see that the algorithm is consistent in scaling slowdown. Slowest time was at around 100 seconds.

Concurrent-Means is the ultimate improvement upon the base algorithm by threading step one and two by dividing the work amongst threads. With 2,4,6 clusters they tend to float somewhere in the same realm of times. Note that these times were calculated with the number of workers being equal to the number of clusters, so speedup would be the most optimal with each experiment if working at optimal number of threads.