Parallel K-Means

CS6068 Parallel Computing Project Presentation

Fall 2018

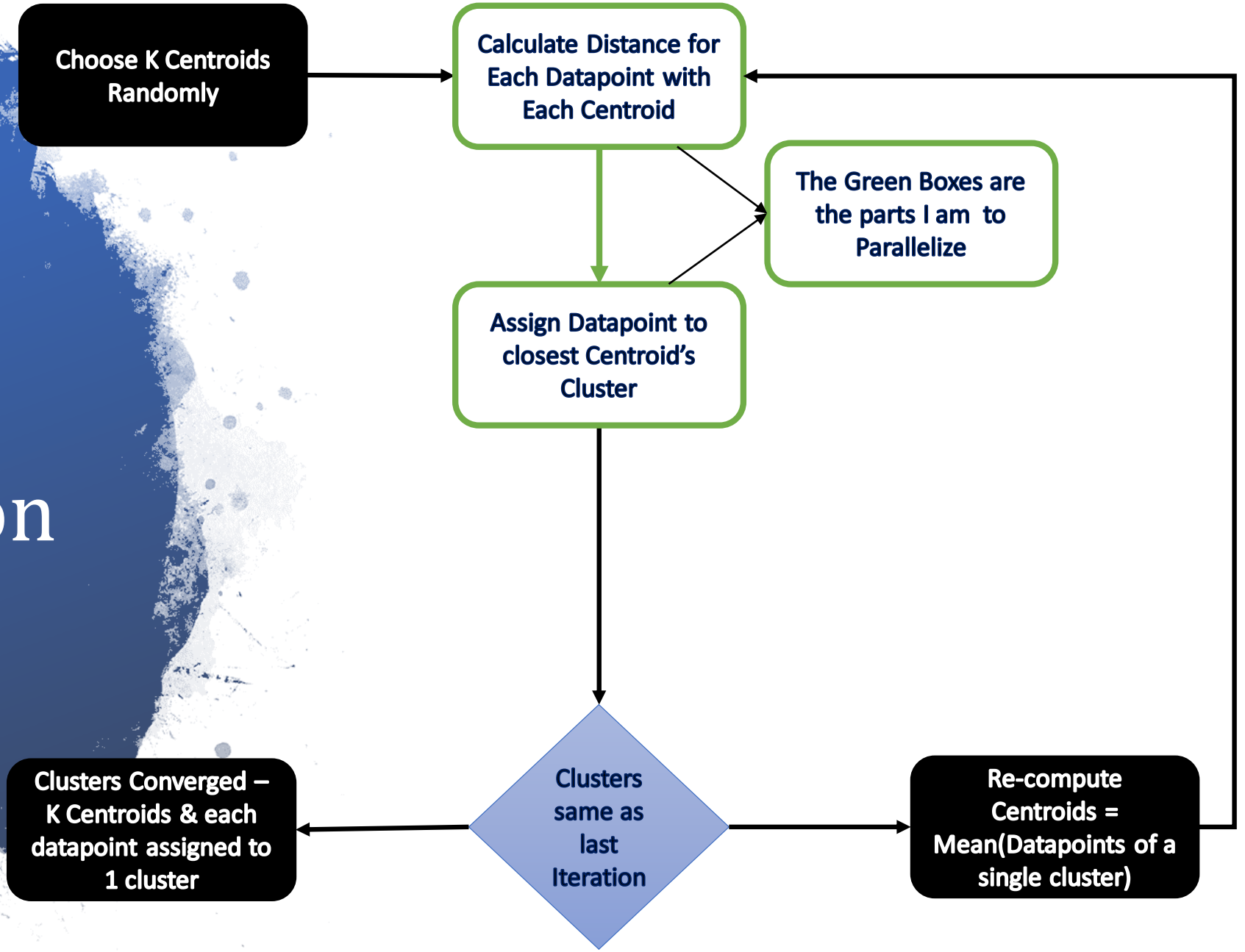
Nithyasri Babu (M12506236)

K-Means Clustering Algorithm

- Unsupervised Learning Algorithm
- Popular for simplicity and correctness
- Computation Intensive Due to Distance Calculations
- Increase in any of the 3 parameters contributing to Time complexity will increase time taken to run the program

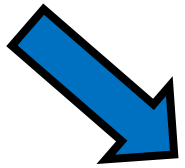
- Time Complexity:
  - O(NKT)
  - N = Number of Data Points
  - K = Number of Clusters to form
  - T = Number of Iterations to final clusters
- Space Complexity:
  - O(((m+k)*n)+n)
  - m = Number of data points
  - k = Number of Clusters
  - n = Number of features

Flow Chart for Sequential Implementation of K-Means



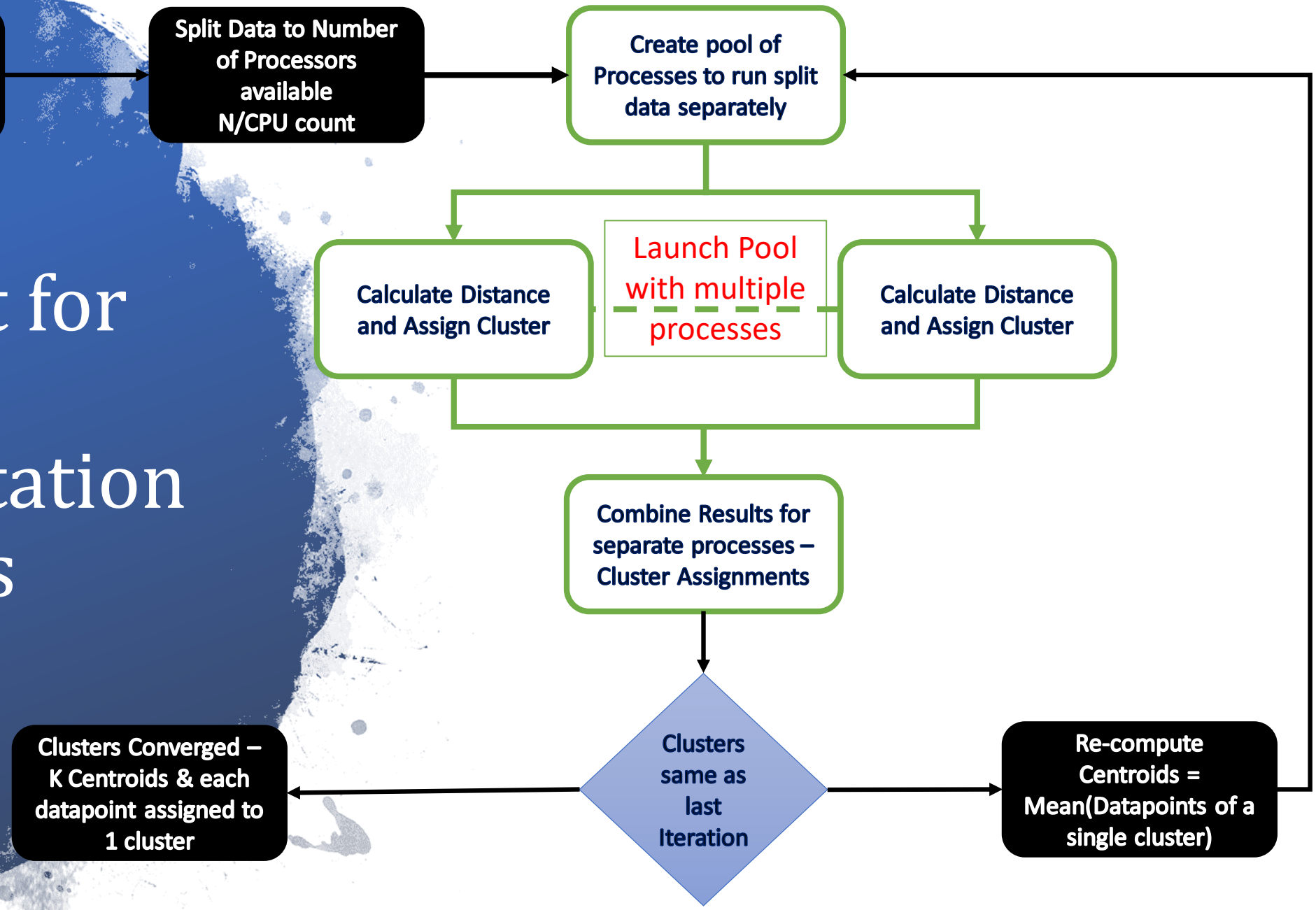
Sequential K-Means Implementation

- Steps:
  - Pick K centroids randomly
  - Compute Distance between each data point with k centroids
  - Assign Cluster based on the minimum distance from the centroid
  - If Clusters are not the same as previous iteration:
    - Recompute Centroids with new cluster Assignments
    - Repeat from Step 2
  - If no, End Process



Parallel Distance Computation

Flow Chart for Parallel Implementation of K-Means



Implementation Specifics

- Python
- Using the Multiprocessing module, in-built with python
- Pool() will launch given number of processes with find_cluster function on multiple processors

- Data Partitions created for cpu_count
- Load is shared by all CPU Processors
- Faster than sequential for large inputs

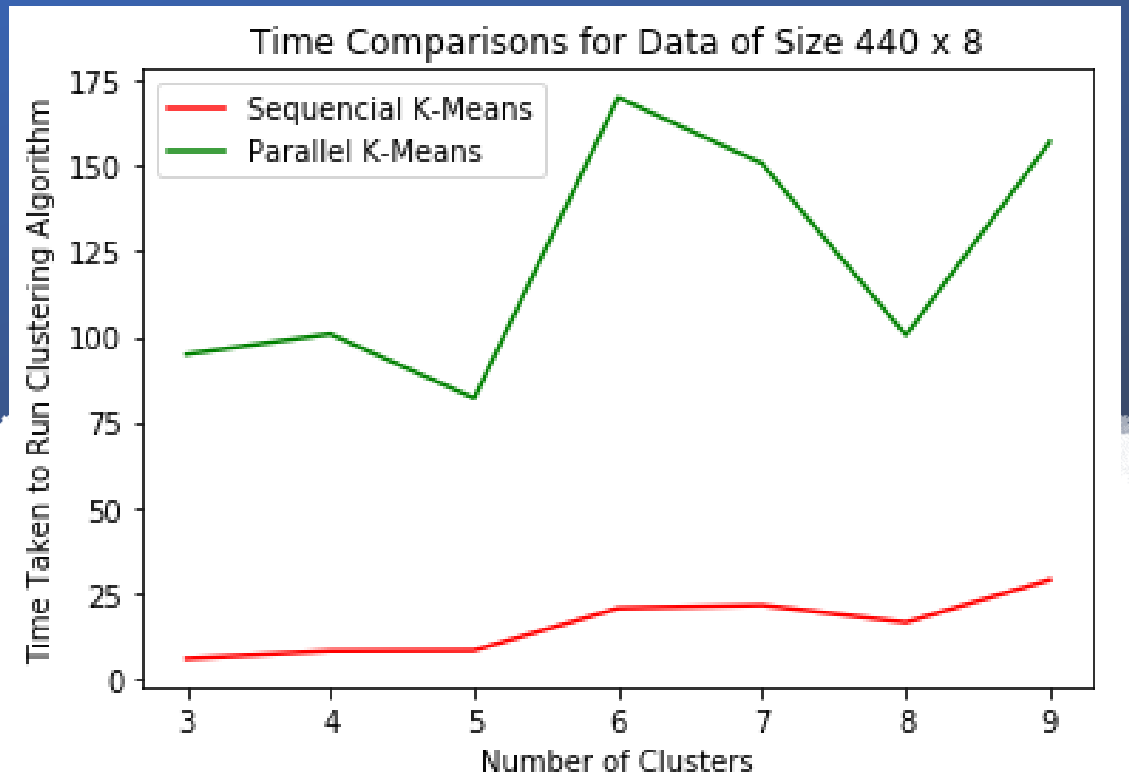
System & Testing

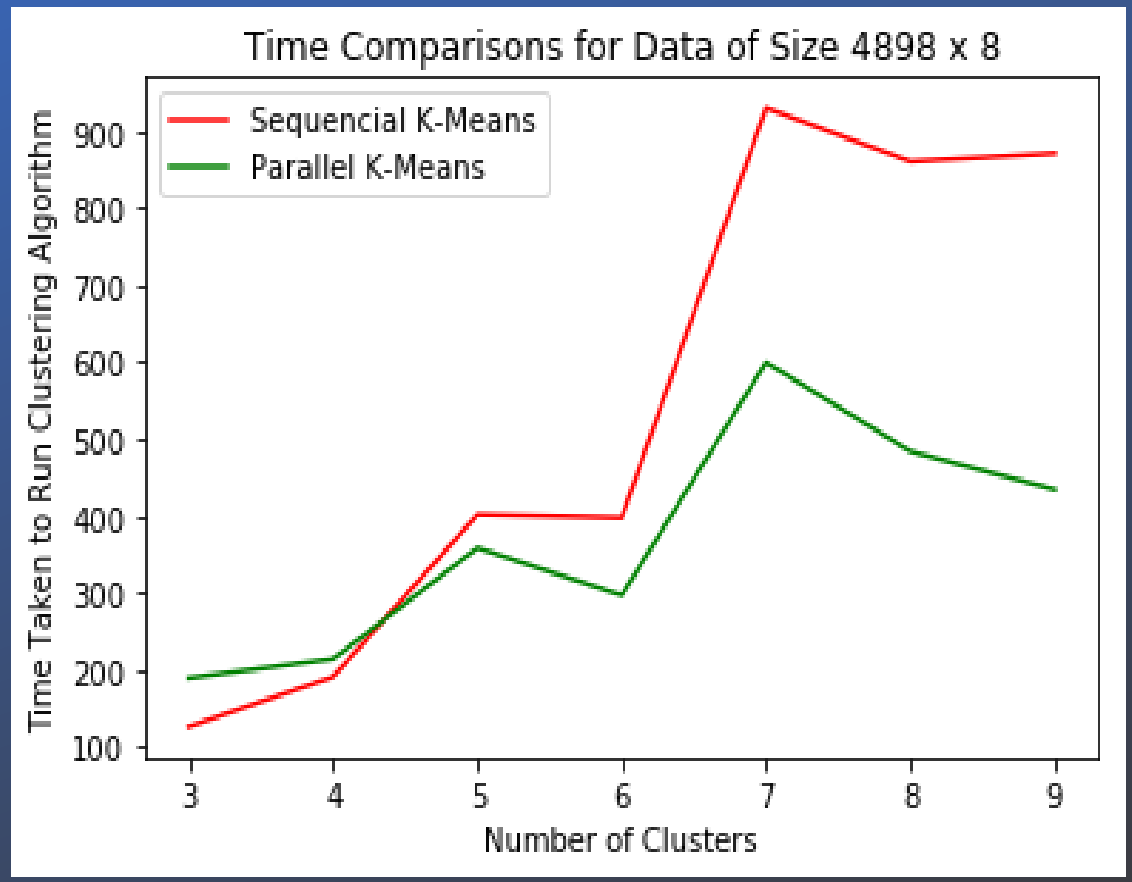
- Intel® Xeon® W-2123 CPU @ 3.60GHz Processor
- 16.0 RAM
- 64-Bit Windows Operating System

- Wholesale Customer Data: 440 Data points × 8 Attributes
- Wine Quality Data: 4898 Data points x 12 Attribute

Performance Analysis

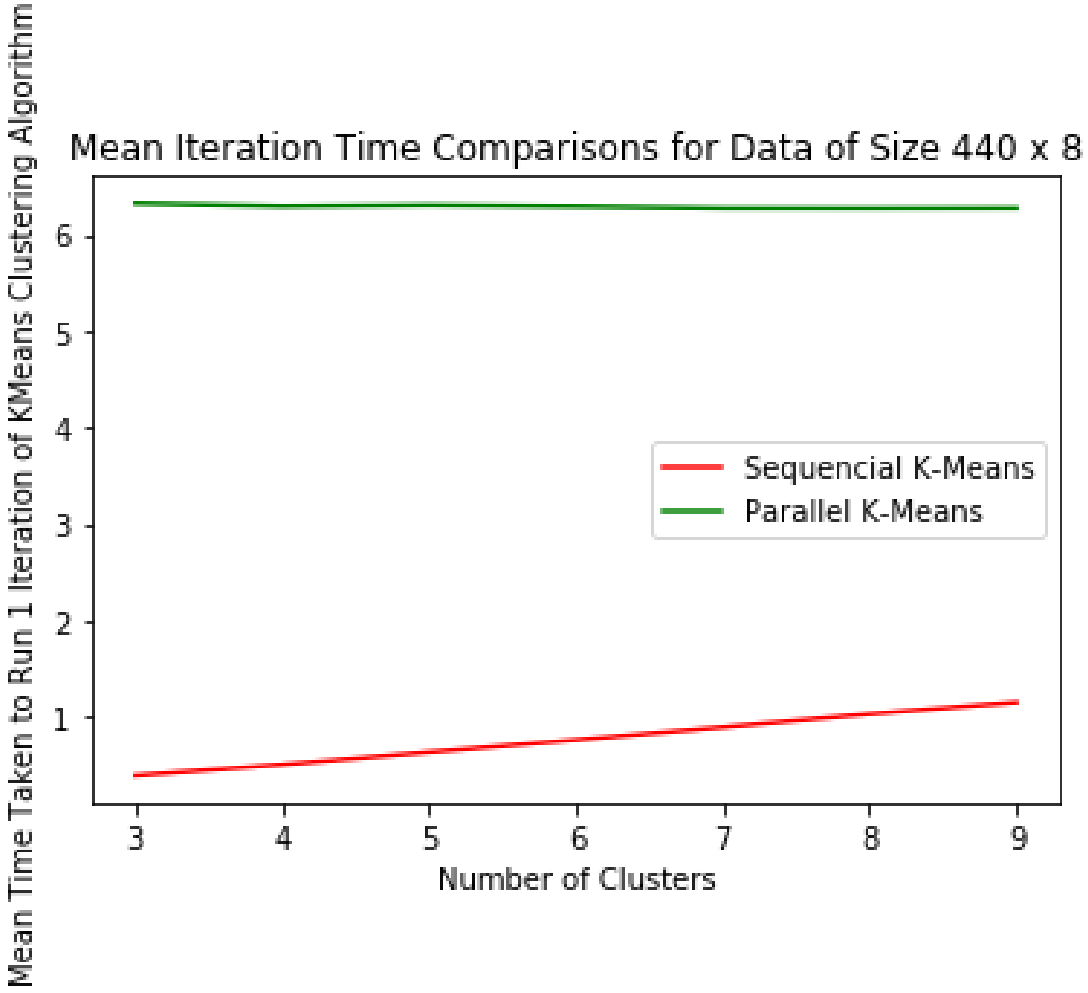
Wholesale Customer Data
440 x 8

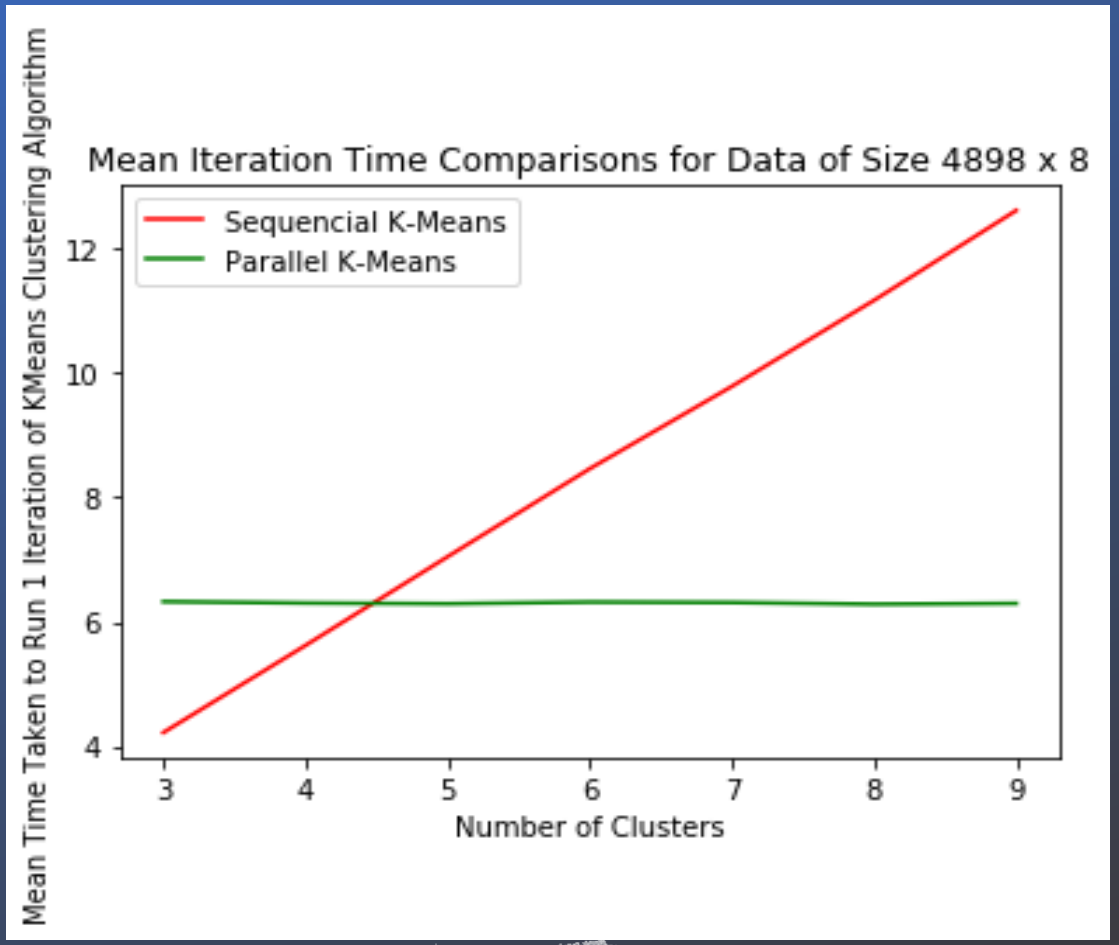




Wine Quality Data
4898 x 8

Wholesale Customer Data
440 x 8





Wine Quality Data
4898 x 8

CPU Utilization = 100 %