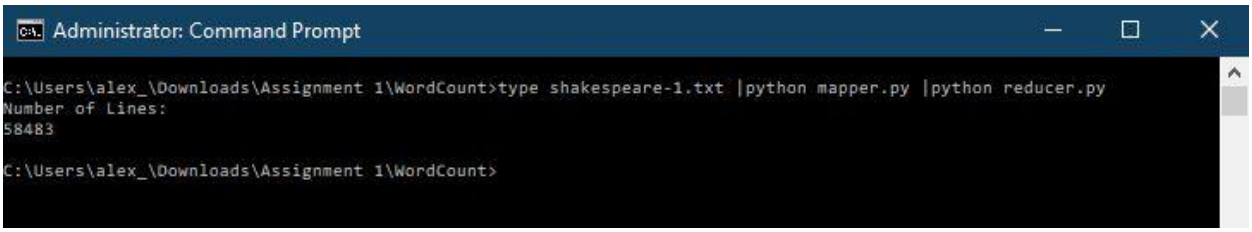
MIE1628: Cloud-Based Data Analytics
Alexander Cheng – 1001634298 – Assignment 1

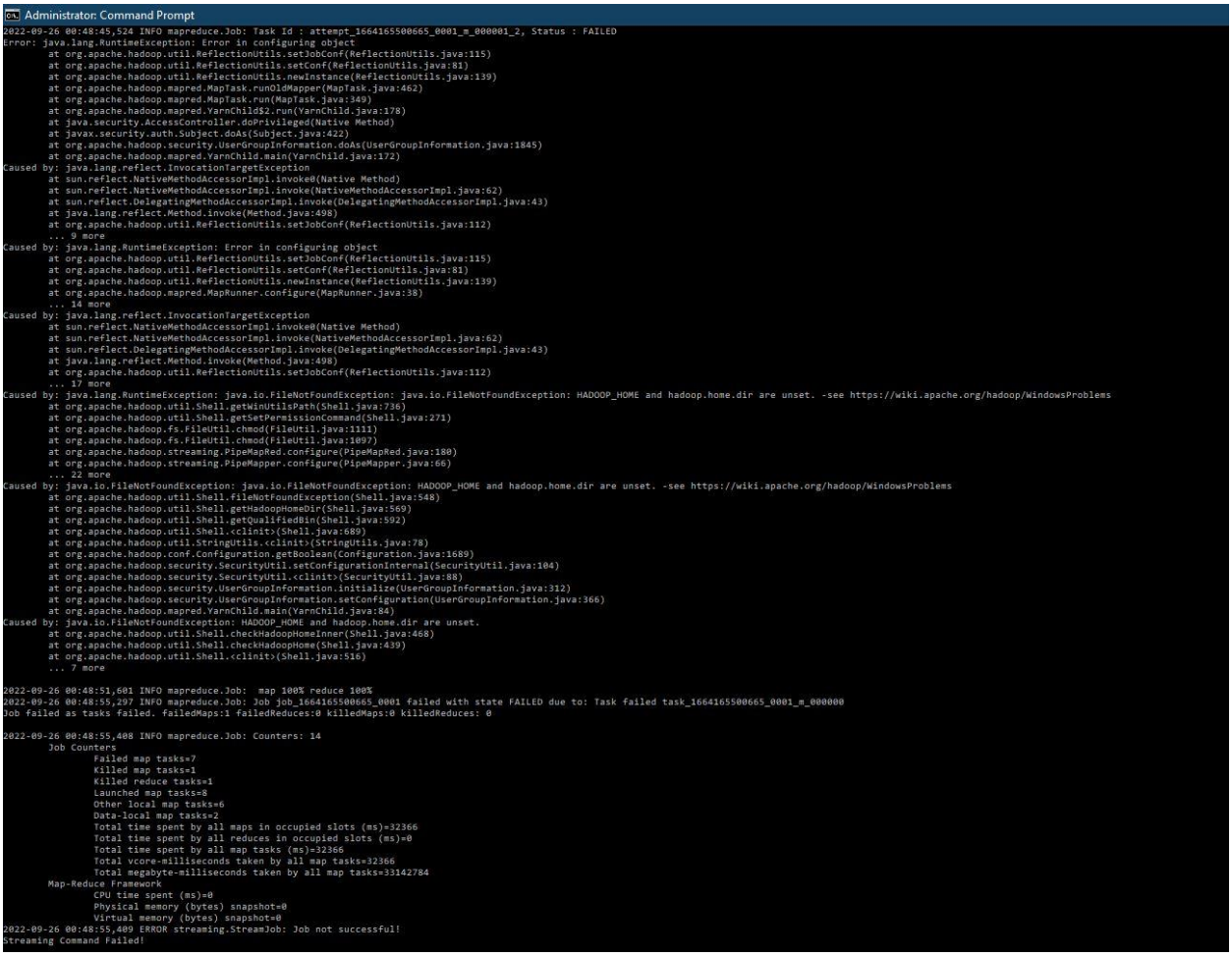
MIE1628 Assignment 1: MapReduce

Question 1: Implement a Map Reduce Function program for counting the number of lines in a document. Use ‘shakespeare.txt’ file, download it from Quercus. Please submit output files with code.

Output from Command Prompt:



Output from Hadoop: Issue with Hadoop, do not have enough time to troubleshoot it.

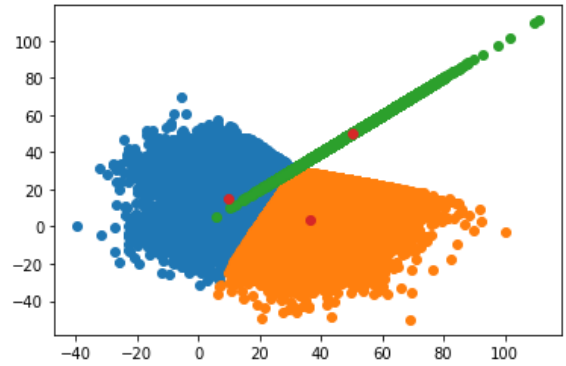


MIE1628: Cloud-Based Data Analytics
Alexander Cheng – 1001634298 – Assignment 1

Question 2: Apply K-Means Clustering on MapReduce using k=3 and k=6 clusters on the given dataset, list the cluster labels or centroids, the number of iterations for convergence or use maximum iterations = 10 and time/duration.

I tried implementing it on Java, but since I am not familiar with Java I switched to Python. I had trouble implementing MapReduce on Python using the command prompt. I used Google Colab and wrote what can be implemented into either a Mapper or a Reducer function for the command prompt and Hadoop.

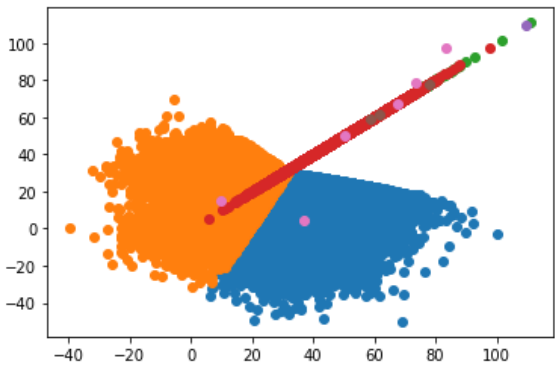
```
Rep #:
5
```




```
Figure
0.7108865278558043
9.97745489524424,15.07609305231439
36.70047392731942,4.193885451642344
50.211208291340625,50.211385617595106
Time:  47.65373420200012
```

Output for K= 3 Clusters

```
Rep #:
10
```




```
Figure
162.6453815300516
36.79606946221731,4.330883984814583
9.981918505281383,15.06574906824593
69.23819920088187,69.53771824605555
50.1930469992544,50.19322600582505
109.54380257893274,219.08760515786548
66.0328343543542,85.62385275600116
Time:  179.17172948999996
```

Output for K=6 Clusters

First step is to import the initial centroid positions text file and convert it into an array. We then make clusters based on which data points are closest to which clusters and assign each data point the cluster label. The new centroids are calculated by taking the average of all the data points that belong in the cluster label. We then edit the text file to those new centroid positions and loop the map and reducer function. There are two stopping criteria used, either the clusters converge in which the clusters do not change much between each iteration, or a maximum number of iterations (10) has been reached. The time for each calculation has also been reached, we can observe that having 6 clusters takes significantly more time to complete compared to the program with only 3 clusters. We can see that as the number of clusters goes up, it will take exponentially more time to run in which we need help of Hadoop to run in parallel.

Question 3: Explain advantages and disadvantages of using K-Means Clustering with MapReduce:

Advantages:

- MapReduce separates the task into two tasks – Map and Reduce that can be used iteratively and in parallel when multiple clusters are required.
- The reducer function does the shuffling of input pairs with the key values
- Computationally much faster

Disadvantages:

- Hard to optimize hyperparameter k in MapReduce
- Iterative process of MapReduce is hard to implement
- MapReduce requires the input data format in the same format

Question 4: Can we reduce the number of distance comparison by applying Canopy Selection? Which distance metric should we use for the canopy selection and why?

We can reduce the number of distance comparisons by applying Canopy Selection. Canopy Selection allows us to significantly reduce the number of distance comparisons by filtering out the number of calculations, we would only have to calculate the distances between the points that are within the same canopy in which a canopy is when we divide the data into overlapping subsets.

This is significantly less calculations than calculating the distance between all the data points of the dataset. The Canopy Selection can be ran using a cheap and approximate similarity measure, such as the inverted index, which is commonly used as a cheap distance metric and provides high accuracy with lower computational cost when a small number of features are sufficient to build canopies.

Question 5: Is it possible to implement Canopy Selection on MapReduce? If yes, then explain in words, how would you implement it?

It is possible to implement Canopy Selection on MapReduce. We can implement it by sending the list of data points as an input for the Mapper function. We can also set the threshold values T1 and T2 inside the

mapper with T1>T2. We can use a while loop and randomly select a data point in which we find the distance between that data point and the rest of the data points. The points within the distance of T1 is set as a Canopy, and the points with the distance less than T2 will be grouped together with the data point to form a key-value pair in which those data points are removed from the data set list. The while loop is run again to form a new cluster until there are no more data points inside the list. The output of the mapper will be the cluster location and the key value pairs (data points that belong to which cluster).

Question 6: Is it possible to combine the Canopy Selection with K-Means on MapReduce? If yes, then explain in words, how would you do that?

It is possible to combine the Canopy Selection with K-Means on MapReduce. The Mapper function will take the list of data points and the number of clusters (k) with initial centroid positions as an input value, the initial centroid positions can be randomly selected as one of the data points. We can also set the threshold values T1 and T2 inside the mapper with T1>T2. We can use a while loop and randomly set an initial centroid which we find the distance between that centroid and the rest of the data points. The points within the distance of T2 is set as a Canopy, and the points with the distance less than T1 will be grouped together with the data point to form a cluster in which those data points are removed from the data set list and the number of clusters k goes up. The while loop is run again to form a new cluster until there are no more data points inside the list or if the number of clusters k has been met. If the list isn't empty after k clusters, the distance between the remaining data points and the selected centroids will be calculated and they will be assigned to the closest centroid. The output of the mapper will be the centroid locations and the data points that belong to each cluster. The reducer will take the intermediate key/value pairs as an input and the new centroid for each cluster will be calculated. The list of centroids will also be updated and sent as a loop for MapReduce and will repeat until the stopping criteria is met (such as no data points changes, number of iterations passed, etc.).