Performance Comparision of Kmeans Clustering in Hadoop Map Reduce

l Vikas Virani
Department of Computer Science
RMIT Univeristy
Melbourne, Australia
virani.vikasr@gmail.com

***Abstract*—Implementation of Kmeans algorithm in Map Reduce framework and performance comparison of Scalability & optimization technique by different size, clusters and node numbers.**

Keywords—Map Reduce, local aggregation, Kmeans clustering.

I. INTRODUCTION

This report aims to explain the implementation of Kmeans clustering algorithm in Hadoop Map Reduce frame work and compare the performance in terms of time by varying number of Hadoop nodes, dataset sizes for different number of clusters.

A detailed explanation of various phases of research like implementation – how Kmeans algorithm is implemented in terms of map reduce context & how it was changed to add optimization for local aggregation, experimentation – analysis on effect of different dataset size, node numbers, clusters on performance as well as scalability of algorithm, results & analysis – evaluation of results by comparing the results obtained from experiments with expected results for each test scenario, comparison with other published studies – how the approach was taken and how it compares to the approach or analysis from other report that are already published with the same dataset, is provided in the report for the implemented approach.

II. METHODOLOGY

A. Implementation

The implementation of map reduce program is divided in 3 packages, mapreduce package which holds Main job configuration file, Mapper & Reducer file and one custom wrapper class; model package which has two class to store DataPoint & Centroid objects; third package is distance which has different classes to calculate different distance measures between Datapoint and Centroids.

KmeanClusteringJob is the main class which is executed when running the program, it first generates data points as per the dataset path passed to argument & then randomly generate K number of centroids based on the value of K passed by user.

As, Kmeans require multiple iterations to converge all cluster, it has multiple iteration of map reduce job till convergence.

Input of mapper is <Centroid, Datapoint> as <Key, Value> pair. For each data point, nearest centroid is calculated from among the K randomly generated centroids. As the local aggregation is implemented in the program, instead of emitting new <Centroid, Datapoint> pair, it is stored in a global associative array (Hash Map) of type <Centroid, DatapointWrapper>, As each Centroid will have multiple data points nearest to it, DatapointWrapper class is created to store an Arraylist of Datapoints nearest to each centroid, instead of storing a single datapoint. After all map tasks are executed, it emits a whole associative array for further processing by reducer.

Reducer will take each centroid & their related list of DatapointWrapper classes & calculate new Centroid based on all Datapoints in the same cluster. These new centroids & data points are written in the output path specified when scheduling a job, these files will be the input for next iteration & this is repeated till convergence.

B. Experimentation

Various experiments were undertaken in terms of size, nodes and number of clusters after implementing this program. Initially, a sample of 50,000 & 100,000 rows were tested. It increases the time to process the records if number of nodes (2 in our case) are same in both case. First one took 6-7 min each iteration (15 iterations to converge), second one took 10 min for each iteration (24 iterations to converge), Almost linear increase in terms of timing.

Next, the dataset was tested on 2, 4 & 6 nodes in Hadoop cluster on a sample data of 50,000 rows to compare the performance. It doesn't affect the time to process the records that much for same number of Clusters. First one took 6-7 min each iteration (15 iterations to converge), second one took 6 min for each iteration (16 iterations to converge), third one took ~5-6 min for each iteration (14 iterations to converge), Almost same results in terms of timing.

Below table shows the detailed comparison of experiments.

Nodes	***Dataset Size***	***K***	***Time (each iteration)***	***Total Time(# of iteration)***
2	**50000 rows**	**2**	**5 min**	**60m (12)**
2	**50000 rows**	**3**	**6 min**	**78m (13)**
2	**50000 rows**	**10**	**6-7 min**	**90m (15)**
4	**100000 rows**	**2**	**9 min**	**189m (21)**

Nodes	*Dataset Size*	*K*	*Time (each iteration)*	*Total Time(# of iteration)*
4	50000 rows	2	5 min	60m (12)
4	50000 rows	3	5 min	60m (12)
6	100000 rows	2	9 min	189m (21)
6	50000 rows	2	5 min	60m (12)
6	50000 rows	3	5 min	60m (12)

This results are just the sample from all combinations that were analyzed. A better visualization of these combinations are added in the figures below. As the whole data took significant amount of time to complete 1 iteration & it requires multiple iterations to converge for Kmeans, it was inefficient & resource consuming to test all combinations on whole dataset so the analysis are carried out on different size of datasets sampled from original one.

C. Results and Analysis

From the experiments performed above, it is observed that as the size of the dataset increases, the time required to process the job increases almost linearly. The experiments were performed on different number of clusters (K) 2, 3, 7 & 10. On a long run, number of clusters doesn't seem to affect that much, it takes almost same time to complete each iteration.

When number of nodes are increased, it doesn't affect the results, it takes almost same time or more time to run each iteration. As the sequence files in which Centroid and DataPoint are stored are same for each iteration, (i.e. output file of reduce will be used as input to next map), it need to be serialized, for this to be done, there should be a single reduce task to write values in file.

Considering this, when above experiments are performed the results are not as expected. It was expected that as the number of nodes will increase it will take less time to execute the whole job, but as opposed to that, it took almost same time to for each iterations while varying the number of nodes for different dataset sizes & different number of clusters.

D. Comprision with Other Published Studies

Reference [1] & [2] is referred when comparing the approach with other published studies. These studies have also done analysis on performance comparison in terms of "Size up", "Scale up" & "Speed up" by various combinations of dataset size, number of nodes etc. These papers are in regards to the same approach in general. In the next part, we'll compare our approach with other published studies using the same data.

Reference [3] & [4] is referred when comparing the studies published using the same data. These studies try to find distributed profitable area and Traffic flow & density of NYC Taxi Data respectively using clustering.

These studies are majorly focused on application related analysis. Our approach was to do an analysis on performance & scalability of algorithm, for which we have used NYC Taxi data as our data sample.

Our approach was combined analysis from both the above two types of published studies, we implemented Kmeans clustering in map reduce & then performed in-depth analysis on the algorithm's scalability in terms of size of data, what is the performance of algorithm when size is increased, what are the observation if number of nodes on which this program run increases. Then we compared the actual results with our expected results of what should happen, which were explained in Results and Analysis section.

III. Future Work

For the future work, Instead of having same sequence file of Centroid for each iteration, we can use a new sequence file for each iteration which is serialized among multiple reduce task in the same iteration to improve the performance. Currently, as there is a single Centroid file, number of reduce task is 1 only because one reduce task may delete the centroids written by other reduce task in same iteration, which can improve the performance significantly.

Also, the time taken depends on the values of attributes that were considered as datapoints, the time taken or above studies were based on pickup & drop off location codes, instead when location co-ordinates are used, it converged faster. So, choice of feature to select for clustering can also improve the performance.

References.

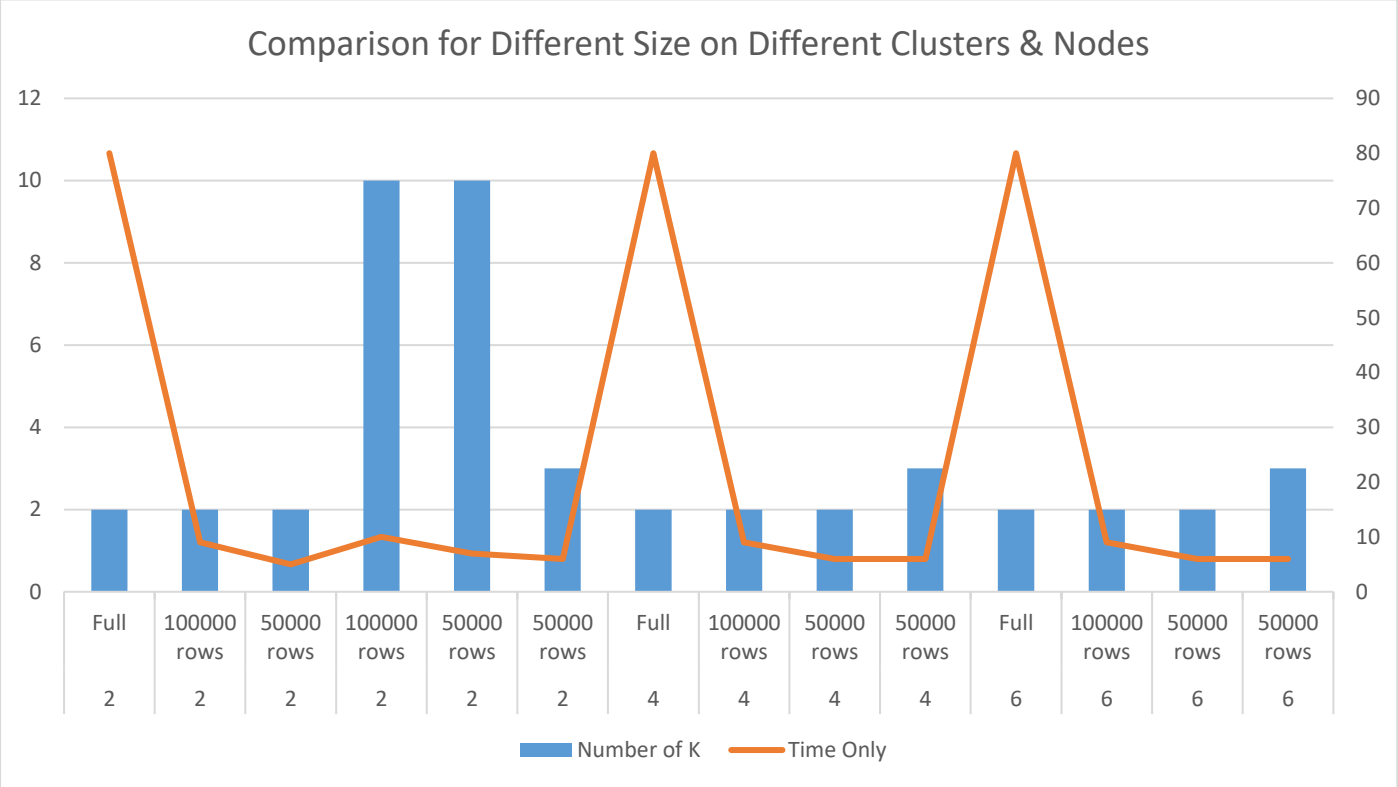
[1] Weizhong Zhao1,2, Huifang Ma1,2, and Qing He1, "Parallel K-Means Clustering Based on MapReduce" The Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Graduate University of Chinese Academy of Sciences

[2] https://stanford.edu/~rezab/classes/cme323/S16/projects_reports/bodoia.pdf

[3] https://static1.squarespace.com/static/560e334de4b035d279a77d57/t/5c49445a03ce64a98708ba9c/1548305545814/Thesis.pdf

[4] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5677113/

Below are the graphs visualized for some combinations of analysis which has the dataset size of – Full, 100000 rows and 50000 rows; Number of K as 2, 3 and 10; node numbers as 2, 4, and 6



Below chart is visualized by keeping number of nodes 2 constant & then varying different dataset size & cluster numbers to compare how it performs on each combination.