An Evaluation of Hadoop Cluster Efficiency in Document Clustering using Parallel K-means

Tanvir Habib Sardar[1], Zahid Ansari[2] , Amina Khatun[3]

[2] *Assistant Professor, Dept. of CSE, P.A.College of Engineering*

[2] *Professor, Dept. of CSE and Dean (Research), P.A.College of Engineering*

[3] *Tata Consultancy Services Ltd, Kolkata*

[1]*tanvir.cs@pace.edu.in*

[2]*zahid_cs@ pace.edu.in*

[3] *aminak77@gmail.com*

Abstract— **One of the significant data mining techniques is clustering. Due to digitalization and globalization of each work space, large datasets are being generated rapidly. Such large dataset clustering is a challenge for traditional sequential clustering algorithms as it requires large execution time to cluster such datasets. Distributed parallel architectures and algorithms are thus helpful to achieve performance and scalability requirement of clustering large datasets. In this study, we design and experiment a parallel k-means algorithm using MapReduce programming model and compared the result with sequential k-means for clustering varying size of document dataset. The result demonstrates that proposed k-means obtains higher performance and outperformed sequential k-means while clustering documents.**

Keywords— Clustering, Data Mining, MapReduce, Hadoop, Distributed Computing.

I. INTRODUCTION

This era of digitalization has covered almost all field of work such as commerce, engineering, scientific laboratories, research, education, healthcare, agriculture etc. The digital datasets are glowing at a very fast rate in our well connected globalized world. It requires implementing proper strategies and efficient techniques to analyse such huge datasets [1]. Data mining is a well-known phrase in analysing and extracting interesting patterns and knowledge from the datasets [2]. One of the major data mining techniques is clustering. Clustering is a method for partitions a dataset into different groups in such a way that different groups possess objects of dissimilar type [3]. Thus clustering is a classification technique which assigns similar data objects into groups [4]. Clustering algorithms are used in a wide variety of working domains in order to analyse different types of datasets [5] [6] [7] [8]. Document clustering is a method where documents are clustered based on frequency of occurrence of words in text dataset.

Due to its simplicity, k-means is a popular and widely used clustering algorithm [9].

The K-means algorithm works as follows [10]: Initially k-means takes an integer input from user which determines the number of clusters to be generated from an input dataset. Depends upon this integer input value, let we denote as k, k number of data points are randomly selected from the dataset and considered as initial cluster centroids. A centroid is a data point, either imaginary or a real object, which represent the mean centre point of a particular group or cluster. Iteratively each data objects are then placed into a cluster after calculating least distance between the objects and centroid. One of the popular and widely used distance calculation measures is Euclidean distance measure [11]. Therefore, after each iteration the centroids recalculation takes place by calculating mean value of the cluster. Newly calculated centroid value is then input to the next iteration and so on. The iterations finishes when new centroid value becomes same or some convergence criteria met. Considering t is the number of iterations, k is the number of clusters and n is the number of data objects in the dataset, the time complexity of the algorithm is $O(nkt)$.

It is found in the literature that traditional clustering algorithms become ineffective in clustering large datasets [12]. Although k-means is a fast and simple clustering algorithm, the rapid growth in data repositories requires an improvement on traditional k-means in order to cluster large datasets efficiently [9]. Distributed frameworks provide a computing environment where a number of interconnected machines share the entire task of processing an algorithm. Each computing machines connected through a computer network are known as a node and the entire computing framework is known as a cluster. Apache Hadoop is open source distributed computing architecture which is designed to process large datasets efficiently. It used MapReduce programming model [13]. Hadoop has two main parts

- *Processing part:* Hadoop recognize and execute a program when developed using MapReduce paradigm. Thus, k-means algorithm can be executed over Hadoop platform by modifying it using MapReduce programming model.
- *Storage Part:* Hadoop Distributed File System (HDFS) is a distributed file system used by Hadoop for storage management.

MapReduce programming model is briefed below [14]:

978-1-5090-6480-9/17/$31.00©2017 IEEE

MapReduce takes a set of <key, value> pair as an input and produces a resulting <key, value> pair after processing. This job is accomplished by two procedures: Map and Reduce. The dataset is first transformed to <key, value> pairs by HDFS keeping the content of each line of the dataset as value and file offset of each line as key. Mapper procedures takes these <key, value> pairs as input and produces intermediate <key, value> pairs for a particular split of the dataset parallel at each node. These intermediate pairs are then sum up by the reducer to get the final result. For a real world example, let we have to count total number of bank currency in each denomination. This currency is gathered in numerous containers. Traditional way to count is a person will open a bag at a time and count the numbers. In MapReduce model the containers will be distributed to few people (mappers) and separately they will count the number of denomination for the allotted containers. After finishing the work the count for each container will be returned to you (reducer) for aggregation. A simple model for MapReduce processing is shown in fig. [1].

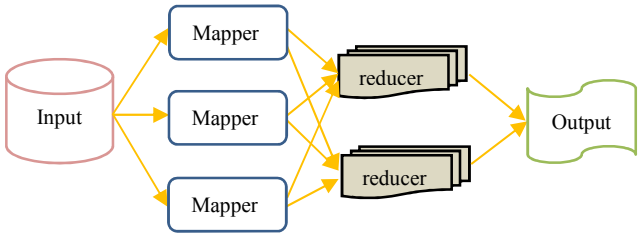

Fig. 1. MapReduce Data Processing Model

HDFS manages the storage part of Hadoop framework [15]. HDFS can be incorporated in commodity hardware. It has much similarity with existing distributed file systems but it has few advantages over other distributed file system also [16]:

- HDFS provides automatic data recovery even if a node fails while execution.
- It works efficiently in commodity hardware.
- It automatically manages large dataset effectively.
- HDFS provides high throughput in accessing diverse datasets.

HDFS is designed on master-slave architecture. Hadoop cluster is created using a master node and few slaves. A dataset is processed from master node which splits the original datasets into pieces and assigns those pieces to the slave nodes. The final result is then gathered from slaves to the master.

In our previous work [4], we have modified k-means using MapReduce on a simple dataset consist of 2-D points. In our first work on Hadoop [17], we evaluated the efficiency gain of MapReduce programs by processing varied sized datasets in parallel. In this work, we proposed a parallel k-means using MapReduce model so as to execute it on top of Hadoop platform for clustering document datasets efficiently. We have varied number of nodes in Hadoop cluster for experimentations and observed the differences in execution time. We have also compared our result with traditional k-means execution. The paper is organized as follows: section II provides the methodology of our proposed algorithm, and section III demonstrate the result obtained by execution of proposed algorithm. Section IV concludes our work.

II. METHODOLOGY

We have modified the traditional k-means algorithm in MapReduce model in order to execute it over Hadoop. The key objective of this work is to derive the advancement in efficiency using proposed parallel K-means.

K-means can operate on numeric dataset. Hence the document dataset has to be transformed before input to k-means. We have transformed the dataset using vector space model. Vector space model normalized the dataset into numeric quantity depends upon the occurrence of words in text files. This transformed dataset then input to HDFS. A text file consists of n terms (words). Vector space model calculates weights for each terms based on its occurrence and represent the file in a vector of numeric data weights w1, w2,…, wn. The relation between documents is determined by the distance between the vectors. This transformed dataset is input to the HDFS and the number of cluster k is taken as an input from the user. The number of centroids k is then randomly chosen from the dataset.

It is problematic to determine which part of k-means should be implemented using Map procedure and which part using Reduce in MapReduce model. It is observed that the distance calculation in k-means is iterative operation and calculation of new centroid is serial operation. Following the above stated mechanism of k-means, parallel k-means using MapReduce programming model is designed. The mapper performs the iterative job of calculating distance between data vectors to centroids and the reducer performs the serial job of calculating new centroids. The entire scenario of proposed parallel k-means is shown in the fig. [2].

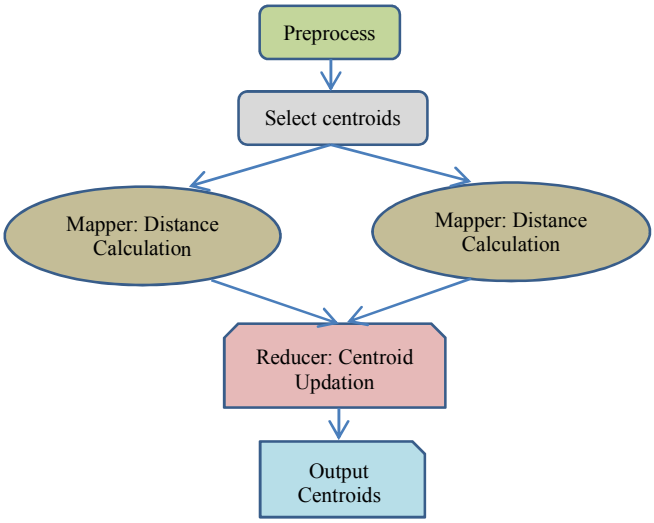

Fig. 2. The Stages of Parallel k-means

The algorithm for mapper and reducer for proposed parallel k-means is provided below.

Algorithm for Mapper:

1. LOAD input_split, cluster_file

2. FOR all $o_i \in$ input_spit and $c_j \in$ cluster_file do
   Similarity=dis (O_i, Cj), 1<i<n; 1<j<k
3. IF (minimumDist>similarity) then
   Centroid=c
4. END IF
5. Issue (Centroid, o_i)
6. END FOR
7. i=i+1

The input of transformed vector space dataset is spitted and allocated to individual mappers in different nodes along with a copy of initially selected k centroids. This is represented in the algorithm as input_split and cluster_file in line 1. O_i is the objects in the input_split, 1<i<n; and C_j is the centroids in cluster_file, 1<j<k. Iterative parallel distance calculation, centroid allocation and outputting the intermediate pair is shown in 2-6.

Algorithm for Reducer:

OutputPair =Input (<Key, Value> pairs from pairs
Sum =0
TotObj =0
MapperCentroid= {NULL}
ReducerCentroid= {NULL}
Let η= <key, value> pair outputted from mappers
1. FOR all η ϵ OutputPair do
   MapperCentroid ← η.key
   DataObject ← η.value
2. END FOR
3. FOR all $o_i \in$ DataObject do
   Sum = sum+ o_i
   TotObj = TotObj + 1
4. END FOR
5. ReducerCentroid = (Sum / TotObj)
6. Issue (MapperCentroid, ReducerCentroid)

In reducer the input is the intermediate <key, value> pairs generated by mapper. Key is the centroid and value is the objects associated to the cluster. In 1-2, it shows that the key obtained from mapper is kept aside and in 3-5 the centroid recalculation is obtained. The resulting updated centroid is issues with the old centroid value in 6.

III. Result and Discussion

The experiments of proposed k-means are carried out using varied Hadoop cluster sizes to analyze the efficiency of distributed clustering using Hadoop and the performance is compared with traditional k-means. Each node of Hadoop cluster is made up with commodity computers with Intel Core2 Due processors, 8GB RAM, 80GB HDD and connected with a 100MBps LAN network. Hadoop version 2.7.2 is installed on Ubuntu 14.04 operating system. The input documents dataset is a newsgroup dataset containing 500 MB and 1GB unstructured text files. The dataset has 20 folders, each containing different categories of news such as politics, game, religion etc. This dataset is received from rice.edu. But due to small size of the dataset, many text files from different sources of online newspapers are combined so as to make it large in size.

The variation in Hadoop cluster size provides a framework in comparing performance gain of Hadoop cluster and proposed algorithm. The traditional k-means is also executed in a standalone computer having same configuration of a node our Hadoop cluster.

Table 1 provides the summary of execution time observed while clustering 500MB and 1GB document dataset using traditional sequential k-means and our proposed parallel k-means using 3-node, 5-node, 8-node and 10-node Hadoop clusters. It is clearly observed that the traditional k-means takes 649 seconds (10 minutes and 49 seconds) to cluster 1GB document dataset. The same task can be achieved by proposed parallel k-means in 6:43 minutes, 6:05 minutes, 3 minutes, and 2:21 minutes if executed over 3-node, 5-node, 8-node and 10-node Hadoop clusters respectively. Thus, proposed k-means when executed over 10-node Hadoop cluster is 4.6 time efficient than traditional k-means. Similarly, the execution time is also shown for clustering 500MB dataset in table 1.

Table 1. Execution time in second

Data Size	Technique	Sequential	3-node	5-node	8-node	10-node
500MB	Execution Time	520	300	176	135	110
	Ratio	4.7	2.72	1.6	1.22	1
1GB	Execution Time	649	403	365	180	141
	Ratio	4.6	2.86	2.6	1.27	1

To achieve and analyze performance gain using proposed k-means, few experiments are conducted which provides us 2 major observations.

- *Observation 1:* 500MB dataset is experimented using traditional k-means and then proposed k-means is experimented using 3-node, 5-node, 8-node and 10-node Hadoop clusters. Traditional k-means took 8:40 minutes (520 seconds) to cluster the dataset whereas proposed k-means in 3-node cluster execution took 5 minutes, 5-node cluster took 2:56 minutes, 8-node cluster took 2:15 minutes and finally 10-node cluster took 1:50 minutes to cluster the dataset. Thus, it is clearly observed that as the number of nodes in the cluster increases the execution time of the clustering job decreases.
- *Observation 2:* 1GB dataset is experimented using traditional k-means and then proposed k-means is experimented using 3-node, 5-node, 8-node and 10-node Hadoop clusters. Traditional k-means took 10:49minutes (649 seconds) to cluster the dataset whereas proposed k-means in 3-node cluster execution took 6:43 minutes, 5-node cluster took 6:05minutes, 8-node cluster took 3 minutes and finally 10-node cluster took 2:21 minutes to cluster the dataset. Thus, it is clearly observed that for 1 GB dataset the execution time decreases as the Hadoop cluster size increases.

The result in execution time specified above is studied so as to analyse proposed k-mean's performance and efficiency. We have taken execution time obtained from 1GB dataset as a unit for evaluating observation 1. Similarly, we have taken execution time obtained from 500MB dataset as a unit for evaluating observation 2. The ratio is then calculated

by dividing the execution time obtained from different Hadoop cluster and traditional execution and shown in table 1. It is shown in table 1 that the execution of proposed k-means in 10-node Hadoop cluster is 1.22, 1.6, 2.72 and 4.7 times faster respectively in comparison of 8-node, 5-node, 3-node and traditional execution of k-means for 500 MB dataset. The execution of proposed k-means in 10-node Hadoop cluster is 1.27, 2.6, 2.86 and 4.6 times faster respectively in comparison of 8-node, 5-node, 3-node and traditional execution of k-means for 1GB dataset.

The efficiency increase is found true in Hadoop clusters but the efficiency gain is not smooth. Smoothness in efficiency means that if we draw a line graph on the execution time then the line in the graph should be straight. This is happened in Hadoop clusters due overheads the nodes of a cluster encompasses. The examples of such overheads are processing overhead of OS processes and basic software tools (like antivirus, firewall tools etc.), priority changes of different system processes, transfer of data requires overhead among nodes, replication of data for proving fault tolerance and checksum in HDFS etc.

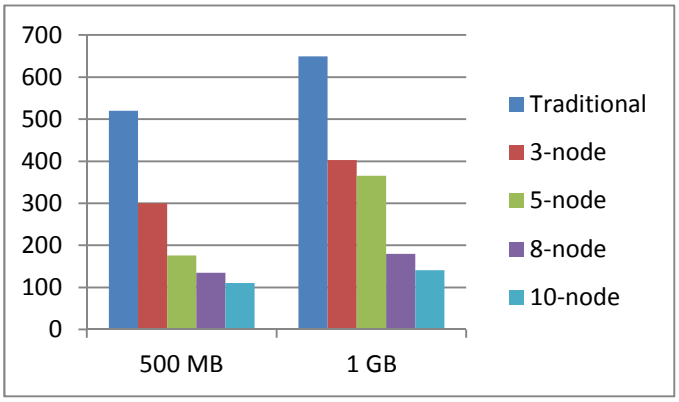

Fig.3. overall combined scenario of experiments and observations

The Relationship between execution time with respect to dataset sizes, traditional k-means execution and proposed k-means execution over different Hadoop cluster sizes are depicted into the bar chart in fig. [3]. The dataset is represented in this bar chart in the horizontal line: 500MB and 1GB while Y axis represents execution time in seconds.

IV. Conclusions

Large datasets are being generated due to globalization and digitalization of each field of work. To process such large datasets, traditional data clustering algorithms are being replaced by parallel implementation. Hadoop is a widely used distributed framework which process algorithms developed in MapReduce model. K-means algorithm is parallelized using MapReduce and run over a variety of Hadoop clusters with 2 different sized datasets. The execution time obtained from parallel k-means is compared with traditional k-means. The experimental results are recorded and displayed using tabular form and using bar graphs.

The proposed k-means outperformed traditional k-means even for small cluster size of 3-nodes. The larger the Hadoop cluster is the better the efficiency of clustering obtained. It is also found that Hadoop processing speed is not even and constant in terms of speed up of clustering with respect to Hadoop cluster size.

References

[1] Lovele Sharma, Vivek Srivastava, “Performance Enhancement of Information Retrieval via Artificial Intelligence”, International Journal of Scientific Research in Science, Engineering and Technology, Volume 3 | Issue 1 | January-February – 2017.

[2] Bansal, Sharma and Goel, “Improved K-mean Clustering Algorithm for Prediction Analysis using Classification Technique in Data Mining”, International Journal of Computer Applications (0975 – 8887) Volume 157 – No 6, January 2017.

[3] Satish Chaurasiya and Ratish Agrawal, “K-Means Clustering With Initial Centroids Based On Difference Operator”, International Journal of Innovative Research in Computer and Communication Engineering, Vol. 5, Issue 1, January 2017.

[4] Tanvir Habib Sardar, Ahmed Rimaz Faizabadi, Zahid Ansari, “An Evaluation of MapReduce Framework in Cluster Analysis”, IEEE International Conference On Intelligent Computing, Instrumentation And Control Technologies (ICICICT-2017), Held at Vimaj Jyothi Engineering College, Kerala, 6-7th July, 2017.

[5] A. A. Abbasi and M. Younis, “A survey on clustering algorithms for wireless sensor networks," Comput. Communication, vol. 30, nos. 14_15 pp. 2826_2841, Oct. 2007.

[6] C. Aggarwal and C. Zhai, ``A survey of text clustering algorithms”, in Mining Text Data. New York, NY, USA: Springer-Verlag, 2012, pp. 77_128.

[7] J. Brank, M. Grobelnik, and D. Mladenić, ``Asurvey of ontology evaluation techniques," in Proc. Conf. Data Mining DataWarehouses (SiKDD), 2005.

[8] R. Xu and D. Wunsch, “Survey of clustering algorithms," IEEE Trans Neural Netw., vol. 16, no. 3, pp. 645_678, May 2005.

[9] Zhang, Jing, et al. “A parallel clustering algorithm with mpi-mkmeans." J Computers 8.1 (2013): 10-17.

[10] Zulfadhilah, Prayudi and Riadi, “Cyber Profiling using Log Analysis and K-Means Clustering”, International Journal of Advanced Computer Science and Applications, Vol. 7, No. 7, 2016.

[11] Bawane, Vinod S, and Sandesha M. Kale. "Clustering Algorithms in MapReduce: A Review", International Journal of Computer Applications, 0975 – 8887.

[12] Ezhilan, Vignesh Kumar, Arvind Kumar, Afzal Khan, “Implementation of Optimised K-Means Clustering on Hadoop Platform”, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 6, Issue 2, February 2016.

[13] Chaturbhuj and Chaudhary, “Improved K-means clustering on Hadoop”, International Journal on Recent and Innovation Trends in Computing and Communication, Volume: 4 Issue: 4, pages 601 – 604, April 2016.

[14] Shweta Mishra, Vivek Badhe, “Improved Map Reduce K Mean Clustering Algorithm for Hadoop Architecture”, International Journal Of Engineering And Computer Science, Volume 5 Issues 7 July 2016, Page No. 17144-17147.

[15] Raghu Garg and Himanshu Aggarwal, “Big Data Analytics Recommendation Solutions for Crop Disease using Hive and Hadoop Platform”, Indian Journal of Science and Technology, 9(9), 2016.

[16] Cheng et al., “Research On HDFS-based Web Server Cluster”, 2011 International Conference on E-Business and E-Government (ICEE), 2011.

[17] Tanvir Habib Sardar, Ahmed Rimaz Faizabadi, Zahid Ansari, “An Analysis of Data Processing using MapReduce Paradigm on the Hadoop Framework”, International Conference on Emerging Trends in Science and Engineering (ICETS-2017), held at Coorg Institute of Technology, Karnataka, 11-12th May, 2017.