International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Study and Analysis on Document Clustering Based on MapReduce in Hadoop using K-Mean Algorithm

Yashika Verma[1], Sumit Kumari[2]

[1]M.Tech Scholar, Computer Science & Engineering, Gurgaon, Haryana, India

[2]Faculty, Computer Science & Engineering, Gurgaon, Haryana, India

Abstract: ***Document clustering is an effective tool to manage information overload. By grouping similar documents together, we enable a human observer to quickly browse large document collections, make it possible to easily grasp the distinct topics and subtopics in them, allow search engines to efficiently query large document collections among many other applications. Hence, it has been widely studied as a part of the broad literature of data clustering. MapReduce is a simplified programming model of distributed parallel computing. It is an important technology of Google, and is commonly used for data-intensive distributed parallel computing. In this paper, we describe how document clustering for large collection can be efficiently implemented with MapReduce. Hadoop implementation provides a convenient and flexible framework for distributed computing on cluster of commodity machines. The design and implementation of direct K-Means and Distributed K-means algorithm on MapReduce is presented.***

Keywords- Hadoop, Mapreduce, Document Clustering, Direct K-Means, Distributed K-Means, Large DataSet

1. Introduction

Document clustering is the use of cluster analysis of textual documents. It has many applications like organizing large document collection, finding similar documents, recommendation system, duplicate content detection, search optimization. Document clustering has been considered for use in a number of different areas of text mining and information retrieval [13]. There are many search engines that are using for information retrieval, but the main challenge in front of the search engine is to present relevant results of the user. Even though there are many knowledge discovery tools to filter, order, classify or cluster their search results exists, still user make extra effects to find the required document. In order to provide solution, combining the entire web mining based data mining techniques [10]. The web documents in each cluster can be pre-processed clustered on Map Reduce.

In this paper we describe our experiences and findings of document clustering based on MapReduce. MapReduce is a framework which programmers only need to specify Map and Reduce functions to make a huge task parallelize and execute on a large cluster of commodity machines. In the document pre-processing stage, we design a new iterative algorithm to calculate tfidf[3] weight on MapReduce[6] in order to evaluate how important a term is to a document in a corpus. Then, a KMeans clustering is implemented on MapReduce to partition all documents into k clusters in which each documents belong to the cluster with the same meaning. Experiments show that our method in approximately linear growth in required running time with increasing corpus size for corpus containing several ten thousand documents.

In this paper, we introduce a MapReduce and the Hadoop distributed clustering algorithm, the design and the implementation of the large-scale data processing model based on MapReduce and Hadoop; and give some experimental results of the distributed clustering based on MapReduce and Hadoop , and discuss the results.

2. Hadoop and Mapreduce

Hadoop is a computational and storage solution for big data problem [9]. Hadoop framework is designed to provide a reliable, shared storage and analysis infrastructure to the user community[4]. The storage portion of the Hadoop framework is provided by a distributed file system solution such as HDFS, [5] while the analysis functionality is presented by MapReduce. Several other components are part of the overall Hadoop solution suite. The MapReduce functionality is designed as a tool for deep data analysis and the transformation of very large data sets. Hadoop enables the users to explore/analyze complex data sets by utilizing customized analysis scripts/commands. [1] In other words, via the customized MapReduce routines, unstructured data sets can be distributed, analyzed, and explored across thousands of shared-nothing processing systems/clusters/nodes. Hadoop's HDFS replicates the data onto multiple nodes to safeguard the environment from any potential data-loss. [5]

A. MapReduce Programming Model

MapReduce programming model was proposed in 2004 by the Google, which is used in processing and generating large data sets implementation [6]. This framework solves many problems, such as data distribution, job scheduling, fault tolerance, machine to machine communication, etc.

i) Mapper

Map function requires the user to handle the input of a pair of key value and produces a group of intermediate key and value pairs. <key,value> consists of two parts, value stands for the data related to the task, key stands for the "group number " of the value . MapReduce combine the intermediate

values with same key and then send them to reduce function [6].
Map algorithm process is described as follows:

Step1. Hadoop and MapReduce framework produce a map task for each Input Split, and each Input Split is generated by the Input Format of job. Each <Key,Value> corresponds to a map task.
Step2. Execute Map task, process the input <key, value> to form a new <key, value>. This process is called "divide into groups". That is, make the correlated values correspond to the same key words. Output key value pairs that do not required the same type of the input key value pairs. A given input value pair can be mapped into 0 or more output pairs.
Step3. Mapper's output is sorted to be allocated to each Reducer. The total number of blocks and the number of job reduce tasks is the same. Users can implement Partitioned interface to control which key is assigned to which Reducer.

ii) Reducer
Reduce function is also provided by the user, which handles the intermediate key pairs and the value set relevant to the intermediate key value. Reduce function mergers these values, to get a small set of values, the process is called "merge ". But this is not simple accumulation. There are complex operations in the process. Reducer makes a group of intermediate values set that associated with the same key smaller [7].

In MapReduce framework, the programmer does not need to care about the details of data communication, so <key, value> is the communication interface for the programmer in MapReduce model. The< key,value> can be seen as a "letter", key is the letter's posting address, value is the letter's content. With the same address letters will be delivered to the same place. Programmers only need to set up correctly<key, value>, MapReduce framework can automatically and accurately cluster the values with the same key together.

Reducer algorithm process is described as follows:
Step1. Shuffle, input of Reducer is the output of sorted Mapper. In this stage, MapReduce will assign related block for each Reducer.
Step2. Sort, in this stage, the input of reducer is grouped according to the key (because the output of different mapper may have the same key). The two stages of Shuffle and Sort are synchronized.
Step3. Secondary Sort, if the key grouping rule in the intermediate process is different from its rule before reduce. We can define a Comparator. The comparator is used to group intermediate keys for the second time.
Map tasks and Reduce task is a whole, can't be separated. They should be used together in the program. We call a MapReduce the process as an MR process. In an MR process, Map tasks run in parallel, Reduce tasks run in parallel, Map and Reduce tasks run serially. An MR process and the next MR process run in serial, synchronization between these operations is guaranteed by the MR system, without programmer's involvement.

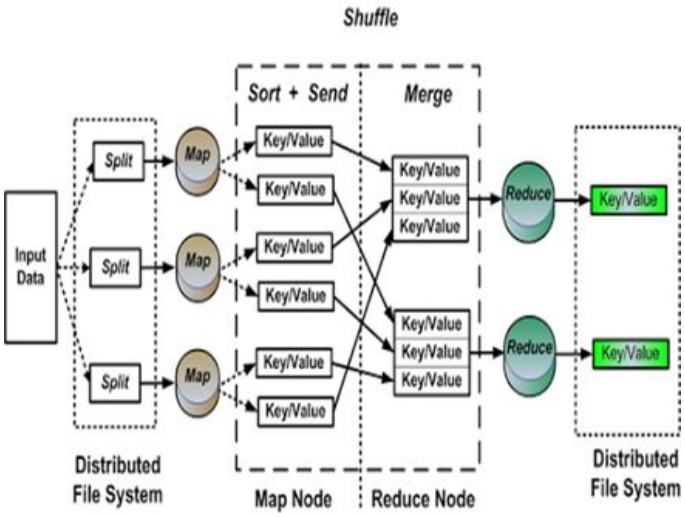


Figure 1: MapReduce Operation Process

3. K-Means Over Mapreduce

In this section, first we describe Simple or Direct K-Means clustering algorithm. Then, we will describe Distributed K-means clustering algorithm both over Hadoop framework.

A. Direct K-means Clustering
K-means document clustering comes under partition technique of clustering where one-level (un-nested) partitioning of the data points is created. If K is the desired number of clusters, then partition approaches typically find all K clusters at once. K-means is based on the idea that a center point can represent a cluster. In particular, for K-means[2] we use the notion of a centroid, which is the mean or median point of a group of points. Basic k-means algorithm is given below.
Input: K: number of cluster, D: Top N documents
Output: K clusters of documents

Algorithm:
Step.1 Generate K centroids C1, C2, …Ck by randomly choosing K documents from D Repeat until there is no change in cluster between two consecutive iterations

Step.2 for each document di in D
for j = 1 to K
Sim(Cj, di) = Find cosine similarity between di and Cj
end for
Assign di to cluster j for which Sim(Cj, di) is maximum
end for

Step.3 Update centroid for each cluster
end loop

Step.4 end K-Means

B. Distributed K-Means Clustering
The iterative computation of k-Means does not directly fit into the MapReduce framework, which mandates a reduce stage following a map stage. However, the computation in each iteration is similar with different cluster centroids and the two phases (assigning points to clusters and calculating the new centroids) in each iteration can be expressed as one

MapReduce job, we use the map tasks to perform the distance computation and point assignment to clusters [11].

The calculation of the new centroids can be performed by the reduce tasks. Hence, we can iteratively run MapReduce jobs and each MapReduce job performs the computation in each iteration of the k-means algorithm. As the distance computation is the most intensive calculation in k-means algorithm, the computation is done effectively using the MapReduce programming model [2].
Distributed K-Means need to follow [1]:

Step.1 The input data are initially stored in files of roughly equal sizes. The input files contain data points coordinates as a sequence of <key, value>pairs where the coordinates are stored in the value field.

Step.2 To share the centroids which are read and updated by each MapReduce job, we store the centroids in files in HDFS so that they are read by the map tasks for distance computation and are updated by reduce tasks with the new centroids.

Step.3 Hence, the final output of the k-means cluster program is the centroid files after the last iteration.

4. Experimental Environment and Data Set

In this paper, experiments are based on a PC with the following hardware configuration: Intel (R) Core (TM) i5-4210M CPU @1.70GHZ, 1.70GHZ, 4.00GB RAM and 1TB hard disk, 64-bit Operating System. The software environment uses the same configuration: [14] Linux operating system; the distributed computing platform of Hadoopand Java development platform JDK 1.7. The data used in this experiment comes from text classification corpus of 20_newsgroups Data Set. The Distributed version of K-Means is implemented on large Data set. The process of clustering starts with vectorization wherein we do pre-processing of text corpus. It produces vector which is matrix i.e. tf-idf [3] values. We have tested on 20_newsgroups text corpus; which contains 20 subdirectories each with 1000 documents. So, vector file has 20 X 1000 lines i.e. for 20000 documents.

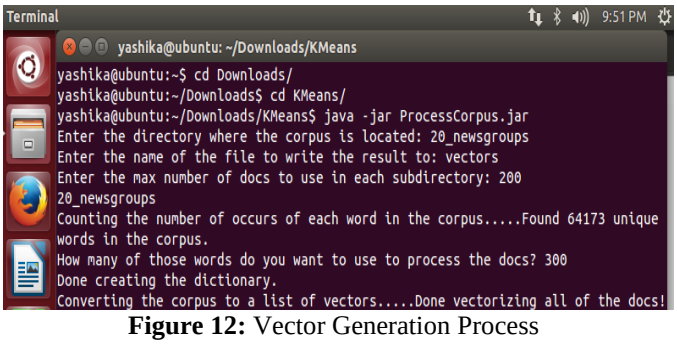


Figure 12: Vector Generation Process

This vector is used to calculate initial set of centroids from the data [8]. This is done by giving vector and number of clusters as input to produce cluster centroids. This initial set of cluster centroids is simply randomly sampled from the "vectors" file. This process is shown in Fig.3.

```
yashika@ubuntu:~/Downloads/KMeans$ java -jar GetCentroids.jar
Enter the data file to select the clusters from: vectors
Enter the name of the file to write the result to: clusters
Enter the number of clusters to select: 10
....Done selecting centroids.
```
Figure 3: Initial Centroids from vectors

After pre-processing the data sets we get some information regarding the unique words, so we made tables for the detailed information about 20_Newsgroups data set.

Table 1: Detailed Information about Data inputs of 20_Newsgroups data set

S. No	Number of Doc in Each Sub-directory	Total Number of Documents	Unique Words Identified	Number of Cluster
1	200	4,000	64,173	10
2	400	8,000	90,567	10
3	600	12,000	11,239	10
4	800	16,000	1,37,943	10
5	1000	20,000	1,53,832	10

If we want to run k-means over hadoop, we need to move these files "vectors" and "cluster centroids" into HDFS file system[12]. This is done using mkdir and copyFromLocal command on Hadoop as shown in Fig. 4& in Fig.5

```
yashika@ubuntu:/usr/local/hadoop$ hadoop dfs -mkdir /vcdir11
Warning: $HADOOP_HOME is deprecated.

yashika@ubuntu:/usr/local/hadoop$ hadoop dfs -mkdir /cldir11
Warning: $HADOOP_HOME is deprecated.
```
Figure 4: Creating Directories in Hadoop

```
yashika@ubuntu:/usr/local/hadoop$ hadoop dfs -copyFromLocal /home/yashika/Downlo
ads/KMeans/vectors /vcdir11
Warning: $HADOOP_HOME is deprecated.

yashika@ubuntu:/usr/local/hadoop$ hadoop dfs -copyFromLocal /home/yashika/Downlo
ads/KMeans/clusters /cldir11
Warning: $HADOOP_HOME is deprecated.
```
Figure 5: Copying Files in Hadoop

As seen in earlier, MapReduce programming framework needs a mapper program and reducer program. These files are bundled in MapRedKMeans.jar. So, next, is to run MapReduce program using command "hadoop jar MapRedKMeans.jar KMeans /vectors /clusters 1". Here, /data and /clusters are HDFS directories storing vectors and initial clusters. "1" indicates number of iteration. If we give last parameter as "3", it will run 3 iterations of the KMeans algorithm. This means that three separate MapReduce jobs will be run in sequence. The centroids produced at the end of iteration 1 will be put into the HDFS directory "/clusters1", those from the end of iteration 2 in "/clusters2", and those from the end of iteration 3 in "/clusters3". When it completes, the results can be examined by running: "java -jar GetDistribution.jar". This will count how many of each of the 20 types of newsgroups was put into each cluster. It is given below from Fig.6 to Fig.8 which shows 20 clusters numbered as cluster 0 to cluster 9 of 20_Newsgroups Data Set.[16]

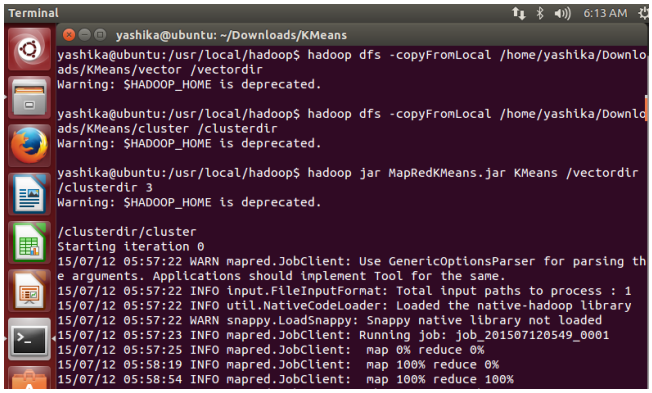


Figure 6: MapRedKMeans Process

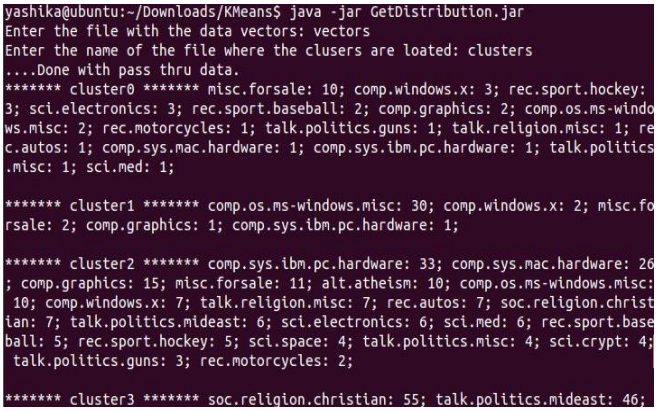


Figure 7: GetDistribution Process

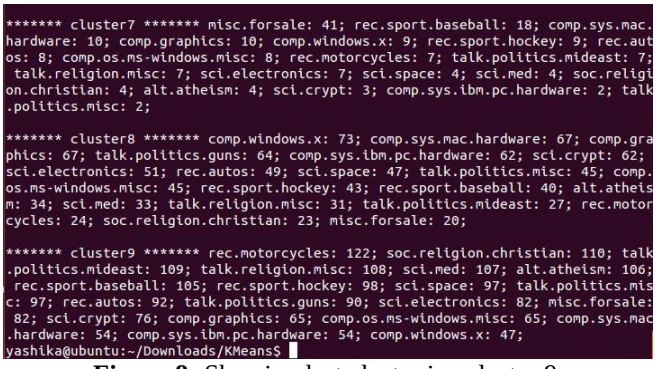
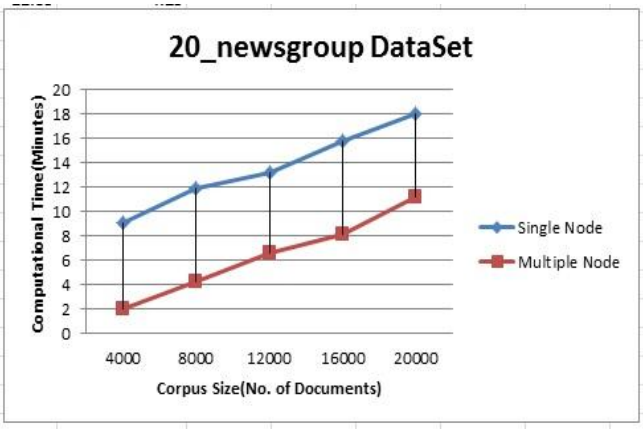


Figure 8: Showing last cluster i.e. cluster 9

Experiment: Comparing Execution time of 20_Newsgroups dataset Provide by K-Means algorithm on stand-alone computer and Multi Node Computers(3 Nodes) [15], based on Map Reduce and Hadoop for the number of documents.

S. No.	Corpus Size (Number of Documents)	Computation Time	
		Single Node	Multiple Node
1	4000	09:05	02:01
2	8,000	11.85	04.25
3	12,000	13.10	06:56
4	16,000	15.74	08.10
5	20,000	17.92	11.20

Comparison of Computation time (20_Newsgroups) on Stand-alone and Multinode System



5. Conclusion & Future Scope

The paper has introduced a mass documents clustering in a distributed system Hadoop. Experiments show the scalability of our method in processing mass data. The contributions of our work lie in both design and implement K-Means algorithm on MapReduce. We believe that our work provides an example of a programming paradigm that could be useful for a broad range of text analysis problems. So further we propose to implement a K-Mean which can automatically decide the number of clusters, number of iterations based on the type of data inputted. Finally, we always pay attention to the alternative approaches to similar problems based on MapReduce. Hadoop provides unprecedented opportunities for researchers to handle real-world problems at scale.

References

[1] Weizhong Zhao, Huifang Ma, and Qing He:" Parallel K-Means Clustering Based on MapReduce". Springer-Verlag Berlin Heidelberg, LNCS 5931, pp. 674–679, 2009
[2] Varad Meru, "Data Clustering using MapReduce: A Look at Various Clustering Algorithms Implemented with MapReduce Paradigm".
[3] Jian Wan, Wenming Yu, and Xianghua Xu: "Design and Implement of Distributed Document Clustering Based on MapReduce". ISCSCT '09, DEC.2009, pp. 278-280.
[4] Brandeis University, Networks and Distributed Computing, "Introduction to Hadoop," [online], available http://www.cs.brandeis.edu/~cs147a/lab/hadoop-intro/
[5] Yahoo Developer Network, "Module 2: The Hadoop Distributed FileSystem,",[online],availableathttp://developer.yahoo.com/hadoop/tutorial/module2.html
[6] Lars Vogel, Version 0.4, "MapReduce Introduction – Tutorial," [online], available at http://www.vogella.com/tutorials/MapReduce/article.html, Oct. 2010
[7] Diana MacLean, "A Very Brief Introduction to MapReduce," [online], available at http://hci.stanford.edu/courses/cs448g/a2/files/map_reduce_tutorial.pdf, 2011
[8] Ping ZHOU, Jingsheng LEI, Wenjun YE:"Large-Scale Data Sets Clustering Based on MapReduce and Hadoop". Journal of Computational Information Systems 7: 16 (2011) 5956-5963

[9] Aditya B. Patel, Manashvi Birla, Ushma Nair, "Addressing Big Data Problem Using Hadoop and Map Reduce",6-8 Dec. 2012
[10] Sandip A Ganage, Dr. R.C. Thool, Heshsham Abdul Basit:"Heterogeneous Computing Based K-Means Clustering Using Hadoop-MapReduce Framework", International Journal of Advanced Research in Computer Science and Software Engineering, June 2013, ISSN:2277 128X.
[11] Ashish A. Golghate, 2 Shailendra W. Shende:" Parallel K-Means Clustering Based on Hadoop and Hama". International Journal of Computing and Technology, Volume 1, Issue 3, April 2014 ISSN : 2348 - 6090.
[12] Michael G. Noll, Applied Research, Big Data, Distributed Systems, Open Source, "Running Hadoop on Ubuntu Linux (Single-Node Cluster)", [online], available at http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/
[13] Satish Muppidi, M.Ramkrishna Murty, "Document Clustering with MapReduce using Hadoop Framework", Vol. 3, Issue 1, January 2015.
[14] "Install Apache Hadoop 2.6.0 in Ubuntu (Single node setup)", [online], available at http://pingax.com/install-hadoop2-6-0-on-ubuntu/.
[15] "Install Apache Hadoop 2.6.0 in Ubuntu (Multi node/Cluster setup)", [online], available at http://pingax.com/install-apache-hadoop-ubuntu-cluster-setup/
[16] WordCount example, [online], available at http://wiki.apache.org/hadoop/WordCount