# Study and Analysis of Distributed Document Clustering Based on MapReduce in Hadoop

**Suman Devi[1] Dr. Suresh Kumar[2]**
[1]Student [2]Professor
[1,2]Faculty of Engineering & Technology Manav Rachna International University Faridabad, India

*Abstract*— 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. Cluster analysis is the most important data mining methods. Efficient parallel algorithms and frameworks are the key to meeting the scalability and performance requirements entailed in such scientific data analysis. 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.

***Key words:*** Hadoop; MapReduce, Document Clustering, Distributed Document Clustering, Large Data Sets

## I. INTRODUCTION

With the rapid development of the Internet, huge volumes of documents need to be processed in a short time. Research on web mining focuses on scalable method applicable to mass documents. Storage and computing of mass documents data in a distributed system is an alternative method. In distributed computing, a problem is divided into many tasks, each of which is solved by one computer. However, many problems such as task scheduling, fault tolerance and inter-machine communication are very tricky for programmers with little experience with parallel and distributed system.

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[4] weight on MapReduce[9] 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 belongs to the cluster with the same meaning.

More importantly, we find that ignoring the terms with the highest document frequencies can not only speed up our algorithm on MapReduce, but also improve the precision of document clustering slightly. 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.

## II. MAPREDUCE AND HADOOP

### A. Hadoop Overview

When data sets go beyond a single storage capacity, it is necessary to distribute them to multiple independent computers. Trans-computer network storage file management system is called distributed file system. A typical Hadoop distributed file system contains thousands of servers, each server stores partial data of file system. HDFS[8] cluster configuration is simple. It just needs more servers and some simple configuration to improve the Hadoop cluster computing power, storage capacity and IO bandwidth. In addition HDFS achieves reliable data replication, fast fault detection and automatic recovery, etc.

### B. MapReduce Overview

In a distributed data storage, when parallel processing the data, we need to consider much, such as synchronization, concurrency, load balancing and other details of the underlying system. It makes the simple calculation become very complex. MapReduce programming model was proposed in 2004 by the Google, which is used in processing and generating large data sets implementation. This framework solves many problems, such as data distribution, job scheduling, fault tolerance, machine to machine communication, etc.

MapReduce is applied in Google's Web search. Programmers need to write many programs for the specific purpose to deal with the massive data distributed and stored in the server cluster, such as crawled documents, web request logs, etc., in order to get the results of different data, such as inverted indices, web document, different views, worms collected the number of pages for each host a summary of a given date within the collection of the most common queries and so on.

### C. MapReduce Programming Model

MapReduce programming model[6], by map and reduce function realize the Mapper and Reducer interfaces. They form the core of task.

*1) 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.

Map algorithm process is described as follows:

1) 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.

 

2) 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.
3) 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.

*2) 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.

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. <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.
- Step 3: 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.

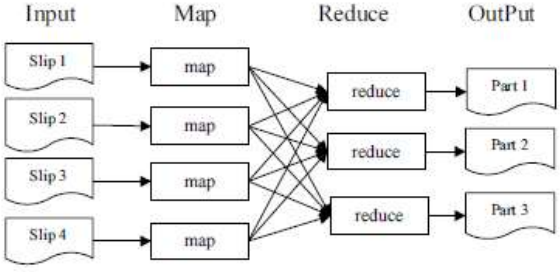


Fig. 1: MapReduce Operation Process

## III. K-MEANS OVER HADOOP

In this section, we describe K-means document clustering. Then we will cover distributed K-means using 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[5] 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*

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 .

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.
Distributed K-Means need to follow:

- 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.

 

## IV. EXPERIMENTAL ENVIRONMENT AND DATA SET

In this paper, experiments are based on a PC with the following hardware configuration: Intel (R) Core (TM) i5-3210M CPU @2.50GHZ, 2.50GHZ, 8.00GB RAM and 1TB hard disk, 64-bit Operating System. The software environment uses the same configuration: Linux operating system; the distributed computing platform of Hadoop; and Java development platform JDK 1.7. The data used in this experiment comes from text classification corpus of 20_newsgroups Data Set and Webdata Data Set which is a HTML type corpus.

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 [4] 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. One more HTML corpus webdata has been taken for the experiment This process is illustrated in Fig.2.

```
hduser@ubuntu:~/Downloads/KMeans/KMeans$ java -jar ProcessCorpus.jar
Enter the directory where the corpus is located: 20_newsgroups
Enter the name of the file to write the result to: vectors
Enter the max number of docs to use in each subdirectory: 100
20_newsgroups
Counting the number of occurs of each word in the corpus...Found 40570 unique wo
rds in the corpus.
```

Fig.2. Process to generate Vectors

This vector is used to calculate initial set of centroids from the data. 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.

```
hduser@ubuntu:~/Downloads/KMeans/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: 20
..Done selecting centroids.
```

Fig.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 and Webdata data set.

| S. No | Number of Doc in Each Sub-directory | Total Number of Documents | Unique Words Identified | Number of Cluster |
|---|---|---|---|---|
| 1 | 250 | 5000 | 69,996 | 20 |
| 2 | 500 | 10,000 | 1,11,203 | 20 |
| 3 | 750 | 15,000 | 1,35,162 | 20 |
| 4 | 1000 | 20,000 | 1,53,832 | 20 |

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 | 10 | 100 | 11,410 | 10 |
| 2 | 20 | 200 | 16,957 | 10 |
| 3 | 30 | 300 | 20,024 | 10 |

Table 2: Detailed Information about Data inputs Webdata data set

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

```
hduser@ubuntu:~/Downloads/hadoop$ bin/hadoop dfs -mkdir /data
Warning: $HADOOP_HOME is deprecated.

hduser@ubuntu:~/Downloads/hadoop$ bin/hadoop dfs -mkdir /clusters
Warning: $HADOOP_HOME is deprecated.

hduser@ubuntu:~/Downloads/hadoop$ bin/hadoop dfs -copyFromLocal /home/hduser/Dow
nloads/KMeans/vectors /data
Warning: $HADOOP_HOME is deprecated.

hduser@ubuntu:~/Downloads/hadoop$ bin/hadoop dfs -copyFromLocal /home/hduser/Dow
nloads/KMeans/clusters /clusters
Warning: $HADOOP_HOME is deprecated.
```

Fig.4. Copy data from local to HDFS

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 /data /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. Similarly, We check for the html corpus webdata. It is given below in Fig.5 which shows 20 clusters numbered as cluster 0 to cluster 19 of 20_Newsgroups Data Set.


Fig.5.Document's Distribution in various cluster

### A. *Experiment 1:*

Comparing Execution time of 20_Newsgroups dataset Provide by K-Means algorithm on stand-alone computer and Multi Node Computer(3 nodes), 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 | 5000 (18.26 MB) | 09:26 | 02:01 |
| 2 | 10,000 (39.96 MB) | 19:14 | 05:39 |

 

| | | | |
|---|---|---|---|
| 3 | 15,000 (55.51 MB) | 20:02 | 06:23 |
| 4 | 20,000 (80.26 MB) | 27:36 | 11:28 |

Table 3: Comparing Execution time of Webdata dataset

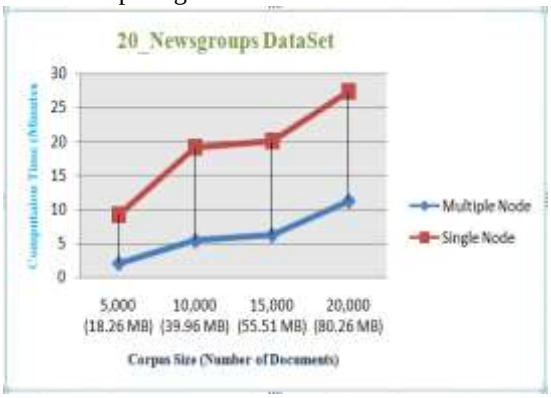


Fig. 6: Comparison of Computation time (20_Newsgroups) on Stand-alone and Multinode System

*B. Experiment 2:*

Comparing Execution time of Webdata dataset Provide by K-Means algorithm on stand-alone computer and Multi Node(3 Nodes) Computer, based on Map Reduce and Hadoop for the number of documents.

| S. No. | Corpus Size (Number of Documents) | Computation Time (Minutes) | |
|---|---|---|---|
| | | Single Node | Multiple Node |
| 1 | 100 (374.61KB) | 6.41 | 0.12 |
| 2 | 200 (752.77 KB) | 6.48 | 0.18 |
| 3 | 300 (1000.77KB) | 6.54 | 0.34 |

Table 4: Comparing Execution time of Webdata dataset

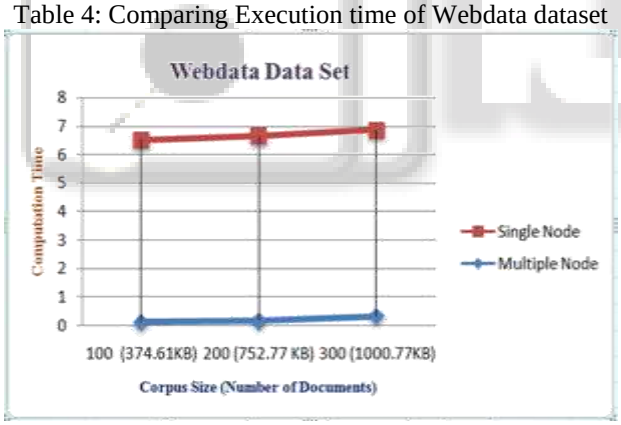


Fig. 7: Comparing Execution time of Webdata dataset

## V. 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] 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.

[2] Weizhong Zhao, Huifang Ma, and Qing He:" Parallel K-Means Clustering Based on MapReduce". Springer-Verlag Berlin Heidelberg, LNCS 5931, pp. 674–679, 2009

[3] Varad Meru, "Data Clustering using MapReduce: A Look at Various Clustering Algorithms Implemented with MapReduce Paradigm".

[4] Jian Wan, Wenming Yu, and Xianghua Xu: " Design and Implement of Distributed Document Clustering Based on MapReduce". ISCSCT '09, DEC.2009, pp. 278-280.

[5] 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.

[6] 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

[7] Brandeis University, Networks and Distributed Computing, "Introduction to Hadoop," [online], available http://www.cs.brandeis.edu/~cs147a/lab/hadoop-intro/

[8] Yahoo Developer Network, "Module 2: The Hadoop Distributed File System,",[online],availableat http://developer.yahoo.com/hadoop/tutorial/module2.html

[9] Lars Vogel, Version 0.4, "MapReduce Introduction – Tutorial," [online], available at http://www.vogella.com/tutorials/MapReduce/article.html, Oct. 2010

[10] Diana MacLean, "A Very Brief Introduction to MapReduce," [online], available at http://hci.stanford.edu/courses/cs448g/a2/files/map_reduce_tutorial.pdf, 2011

[11] 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/

[12] "Install Apache Hadoop 2.6.0 in Ubuntu (Single node setup)", [online], available at http://pingax.com/install-hadoop2-6-0-on-ubuntu/.

[13] "Install Apache Hadoop 2.6.0 in Ubuntu (Multi node/Cluster setup)", [online], available at http://pingax.com/install-apache-hadoop-ubuntu-cluster-setup/

[14] WordCount example, [online], available at http://wiki.apache.org/hadoop/WordCount

[15] Project Gutenberg, [online], available at http://www.gutenberg.org/ebooks

[16] K-Means over Hadoop, [online], available at http://cmj4.web.rice.edu/MapRedKMeans.html

 