International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Design and Implementation of K-Means and Hierarchical Document Clustering on Hadoop

Y. K. Patil[1], Prof. V. S. Nandedkar[2]

[1]M.E student PVPIT Pune, Maharashtra, India

[2]HOD IT dept. PVPIT, Pune, Maharashtra, India

Abstract: ***Document clustering is one of the important areas in data mining. Hadoop is being used by the Yahoo, Google, Face book and Twitter business companies for implementing real time applications. Email, social media blog, movie review comments, books are used for document clustering. This paper focuses on the document clustering using Hadoop. Hadoop is the new technology used for parallel computing of documents. The computing time complexity in Hadoop for document clustering is less as compared to JAVA based implementations. In this paper, authors have proposed the design and implementation of Tf-Idf, K-means and Hierarchical clustering algorithms on Hadoop.***

Keywords: Hadoop, Tf-Idf, Cosine Similarity, K-means and Hierarchical clustering

1. Introduction

The large amount of texts is available on internet, from huge corpus text need to be processed within a short period. We can implement document clustering using programming language with parallel execution but in this approach some issues are like fault tolerance, inter processor communication and task scheduling. Research on distributed document clustering based on MapReduce has been done [1]. Storing huge data and retrieving the text of a document in a distributed system is an alternative method [2]. In distributed environment the task is divided and scheduled to appropriate system. In this paper we describe the document clustering based on Hadoop uses the MapReduce procedure for implementing k-means and Hierarchical clustering.

Hadoop provide distributed environment with Hadoop distributed file system (HDFS) and MapReduce function. HDFS is used to store the large data and data is store on single or multiple nodes. Where one node is called master node and all other are data (Workers) nodes the master node manages the file system namespace. This information is stored persistently on the local disk in the form of two files the namespace image and the edit log It maintains the file system tree and the metadata for all the files and directories in the tree. The namenode also knows the datanodes on which all the blocks for a given file are located, however, it does not store block locations persistently, since this information is reconstructed from datanodes when the system starts. MapReduce is framework where the documents are processed with Map and Reduce function to achieve parallelism on huge Data set. MapReduce function works just like divide and merge strategy. In the proposed system input is different documents and the output is clustered and tree based (Hierarchical) clustered. In document pre-processing phase the Tf-Idf is determined on MapReduce. This Tf-Idf is important to identify the document from the corpus. In this paper initially the system architecture of proposed system is briefly explained. Then the Algorithm for implementation of K-means and hierarchical clustering is presented. Finally concluded that how the proposed system is good for document clustering.

2. Related Work

Document Clustering is the task of grouping a set of documents in such a way that objects in the same group are more similar to each other than to those in other groups clusters Basically there are two main approaches in document clustering that are Partition clustering & Hierarchical clustering[3][4]. The authors Jiawei Han, Micheline Kamber, they have presented various clustering Techniques for Data Mining. They have published how to use K-means clustering algorithm to cluster large data set in various disjoint clusters [5]. The author Benjamin C.M. Fung, Ke Wang, Martin Ester, presented Hierarchical Document Clustering Using Frequent Item sets. They have focused on how to manage the documents in hierarchical clusters [6]. The author Tian Xia, presented a research work on "An Improvement to TF-IDF: Term Distribution based Term Weight Algorithm", this paper talks about how to find Tf-Idf weight for document clustering [7]. The authors Tom White Shvachko, Hairong Kuang presented a paper on Hadoop: The Definitive Guide. Authors have discussed the working of Map and Reduce methods in Hadoop system [8]. The Hadoop Distributed File System paper focused on the Hadoop implementation using single node implementation as well as multimode implementation details [9]. Our approach effectively combines the K-means and Hierarchical clustering algorithm.

3. System Architecture

The Figure 1 is proposed system architecture. In Figure 1the input to system is pdf documents. The Hadoop which uses the MapReduce function for parallel computing of documents. The parallel processing reduces the time complexity. In this architecture documents are parallel executed to find Tf-Idf. Before giving the input to our system we need to perform Preprocessing and Text processing of collected documents and need to find Tf-Idf as well as cosine similarity.

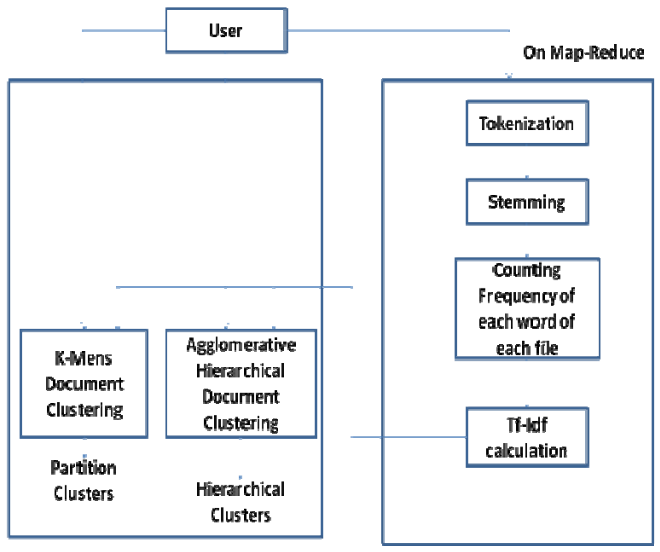

User
On Map-Reduce
Tokenization
Stemming
Counting Frequency of each word of each file
Tf-Idf calculation
K-Mens Document Clustering
Agglomerative Hierarchical Document Clustering
Partition Clusters
Hierarchical Clusters

Figure 1: Architecture of proposed System

A. Preprocessing Phase:

A.1 Parser Phase:

1. Parse Pdf file and store Pdf stream in random access file create using COS Document constructor. PDFParser class is used to parse the Pdf documents. This class needs a FileInputStream and File as parameter to parse a dot pdf file.

```
PDFParser parser = new PDFParser (new FileInputStream (new File ()));
COSDocumentcosdoc = parser.getDocument ();
```

2. Extract Text from random access file. In this process PDFTextStripper constructor class is used.

```
PDFTextStripper Stripper = new PDFTextStripper ();
PDDocument Doc = new PDDocument (cosdoc);
String parsedText = Stripper.getText (Doc);
```

3. Write parsed text in newly created text file

A.2 Text Processing

1) Read input string form text file and tokenize it e.g. String line = “Data mining is the process of knowledge discovery”, in this string we need to find tokens. Tokens are Data, mining, is, the, process, of, knowledge, discovery, but ‘is’, ’the’, ’of’ are stop words so these words need to be removed as well as some of the words are converted to its root form e.g. mining is mine
2) Read tokenized words one by one and removes punctuations from input line such as comma, dot etc e.g. Input token: discovery, Output token: discovery Input token: patterns. Output token: patterns.
3) Stemming: convert each word to its root word by removing ing, es, ed etc. of each words in each text file, and then find occurrence of each word in each text file and in whole corpus.

B. To determine Tf-Idf and Cosine Similarity

The Tf-Idf weight and cosine similarity are calculated as given in eq.1 and eq.2. Tf-Idf (Term Inverse Document Frequency) a mechanism for calculating the effect of terms that occur so frequently in corpus. The cosine similarity is used to find how the documents are closely similar to each other in terms to do cluster. The idea of determining the Tf-Idf and cosine are given in [9].

$$Tf-Idf=\log(\frac{n}{df_t}) \qquad (1)$$

Where,
Tf-Idf=Inverse document frequency of term
n=Number of documents in corpus
df_t=Frequency of a term t in corpus

$$\cos(x,y)=\frac{\sum_{i=0}^{n}x_i y_i}{\sqrt{\sum_{i=0}^{n}x_i^2}\sqrt{\sum_{i=0}^{n}y_i^2}} \qquad (2)$$

Where,
x_i=is the Tf-Idf weight of ith term in first document
y_i=is the Tf-Idf weight of ith term in second document

4. K-Means and Hierarchical Agglomerative Algorithm

Algorithm 1: K-Means Documents Clustering Algorithm

```
Input: Set of pdf documents. K-number of cluster,
Step 1: Identifying unique words from the given input document
Step 2: Generation of input vector using TF-IDF weighting
Step 3: Selection of similarity measure for generating similarity matrix. Here in this project cosine similarity is used.
Step 4: Specifying the value of k i.e. number of clusters.
Step 5: Randomly select k documents and place one of k selected documents in each cluster.
Step 6: Place the documents in cluster based on similarity between documents and the documents present in the clusters.
Step 7: Compute centroids for each cluster.
Step 8: Again by using similarity measures, find the similarity between the centroids and the input documents.
Step 9: Now place the documents in the clusters based on similarity between documents and the centroids of clusters.
Step 10: After placing all the documents in the clusters, compare the precious iteration clusters with current iteration clusters.
Step 11: If all the clusters contains same documents in previous and current iteration then terminate the algorithm here and hence found the clusters
Step 12: Else repeat through step-7
Output: The entire cluster contains the same document
```

Algorithm 2: Hierarchical Documents Agglomerative Clustering Algorithm

Input: 'n' number of pdf documents.

```
1. Create 'n' folders.
2. foreach (document) do
 {
 Put in individual cluster.
 }
 end foreach
3. foreach document of each folder do
 {
 Map (document)
 Calculate TF-IDF weight value
 }
 end foreach

 4. foreach calculated document do
 {
 Reduce (document)
 if (Tf-Idf matches with other documents)
 {
 Merge clusters
 }
 end if
 }
 end foreach

5. Finally merged all disjoint clusters in a root cluster.
```

Output: Hierarchical Agglomerative clustered documents.

Here we present the Hierarchical Agglomerative clustering algorithm where the input is pdf documents. The documents are parallely processed using MapReduce function to determine the Tf-Idf. The 'n' initial clusters are chosen from the corpus and each cluster is kept in a separate folders. The details procedure for clustering is given in Algorithm 2.

5. Math

Step 1:

$$Tf - Idf = \log(\frac{n}{df_t}) \;\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \;(1)$$

Where,
Tf-Idf=Inverse document frequency of term
n=Number of documents in corpus
df_t=Frequency of a term t in corpus

Step 2:

$$\cos(x, y) = \frac{\sum_{i=0}^{n} x_i y_i}{\sqrt{\sum_{i=0}^{n} x_i^2}\sqrt{\sum_{i=0}^{n} y_i^2}} \;\ldots\ldots\ldots\; (2)$$

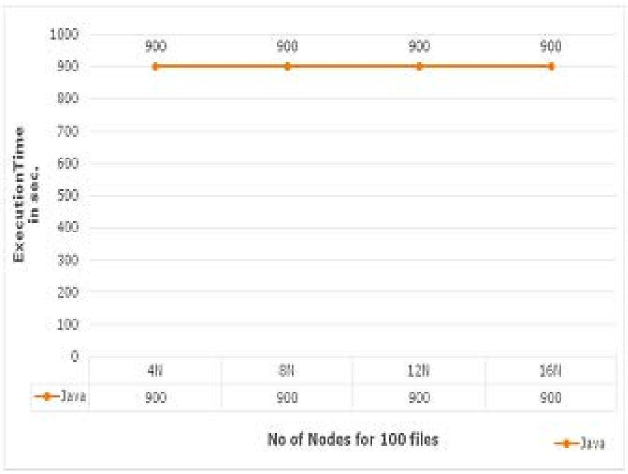
Where,
x_i=is the Tf-Idf weight of ith term in first document.
y_i=is the Tf-Idf weight of ith term in second document

6. Experiment and Result

In this proposed system n-number of text files are given as input and here tokenization (i.e. removing stop words), stemming (i.e. converting each word of each files to its root words), and finding Tf-Idf values of each word of each files are carried out on the Hadoop. Once this text pre-processing has completed then K-means algorithm is applied which provide partitioned clusters or applying Hierarchical algorithm is applied which will give Hierarchical clusters.

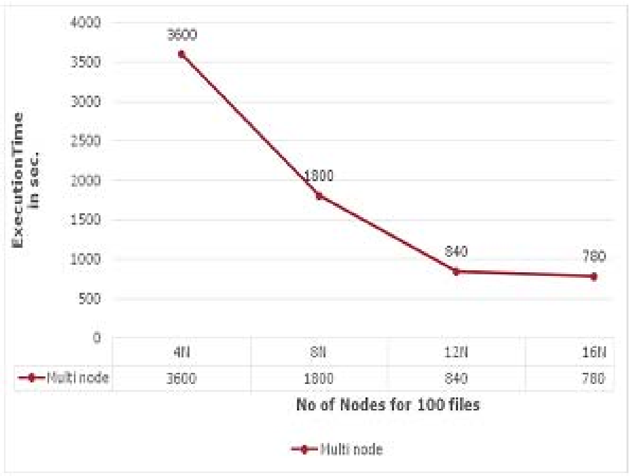
In experimentation, project executed on Hadoop version 1.0.3 running on a multi core machine (one is the master, also act as a slave) having Linux on machine.

In Experimentation, I am giving 100 files for clustering over existing system then I found that , Execution time require to cluster 100 files is remains constant even if increase the number node. Its result is showing in following graph.



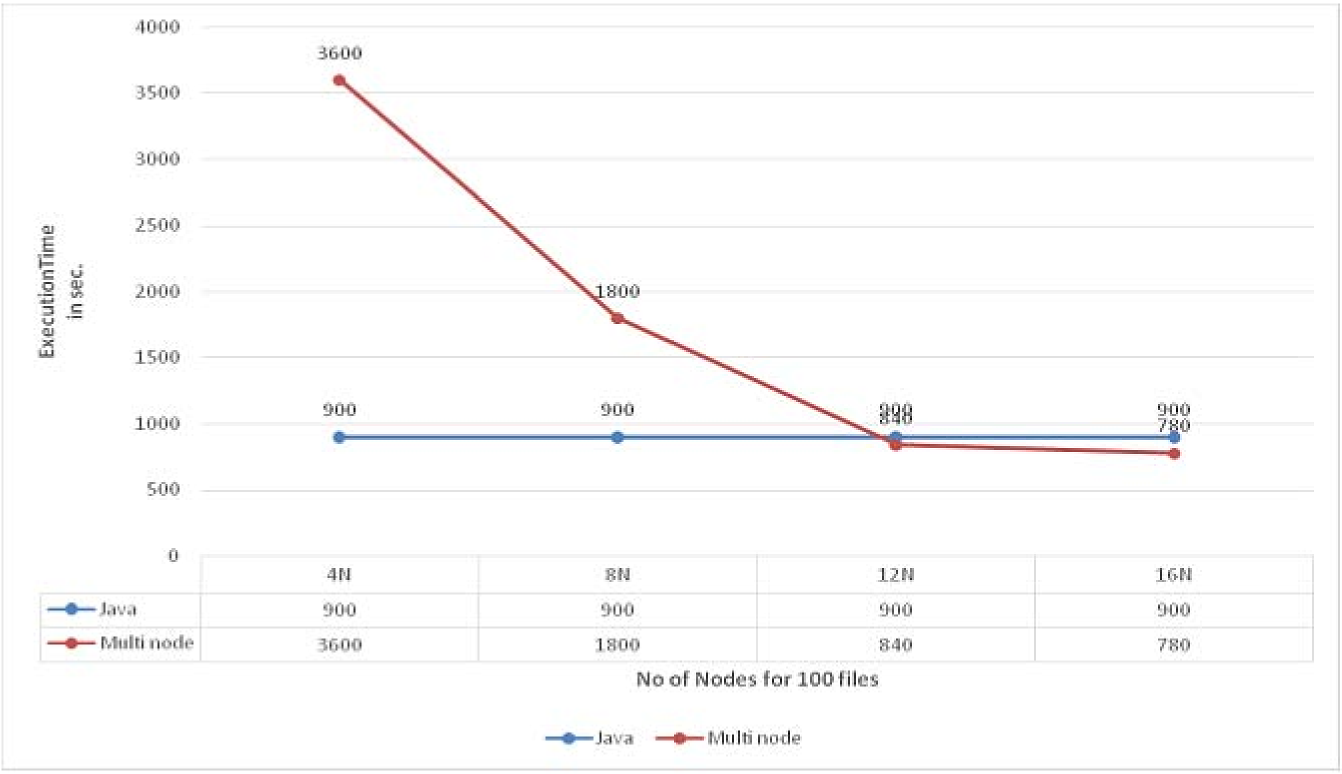
Graph 1: Result of Existing System

In Experimentation, I am giving 100 files for clustering over proposed system then I found that , Execution time require to cluster 100 files is exponentially decreases with increase in number of node on corpus containing . Its result is showing in following graph.



Graph 2: Result of Proposed System

Comparing execution time between Java and Hadoop concludes that execution time in Hadoop exponentially decreases with increase in number of node on corpus containing n no. of documents whereas in java execution time cannot be varied.



	4N	8N	12N	16N
Java	900	900	900	900
Multi node	3600	1800	840	780

Graph 3: Hadoop varying Execution Time Compare to java

C. Outcome of K-Means Document clustering

For K-means Document clustering algorithm , Input given as Tf-Idf weight of each word of each file and number of cluster want to be form as a four then four cluster are created at first step and randomly it select four document from input an place each in separate cluster and remaining document are assign to cluster on the basis of similarity and calculate centroid of each cluster and redistribute document on the basis of similarity to centriod this process perform recursively till no change is occur and finally we got the following cluster.

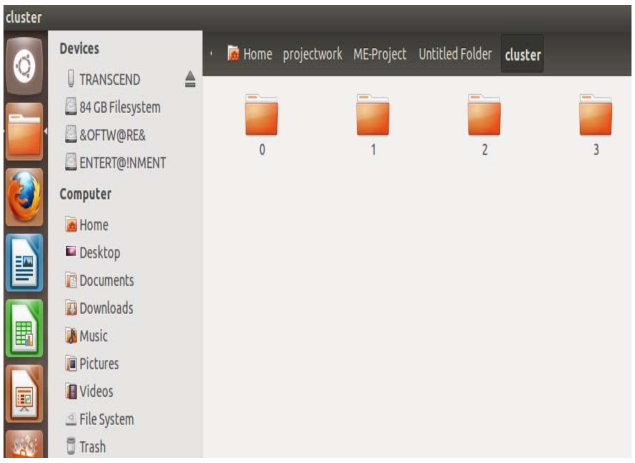


Figure 2: Partitioned Clusters

B. Outcomes of Agglomerative Hierarchical Document Clustering

For Agglomerative hierarchical clustering input given as a Tf-Idf weight of each word of each file and at first step it create number of cluster equals to number of input document and place each document in separate cluster and in next step it compares the content of one cluster to content of all other cluster if similarity is found then merge this two cluster in one cluster this process is repeated till content of all cluster in previous iteration and next iteration is same then finally merge all cluster in one cluster, which shown in following figure.

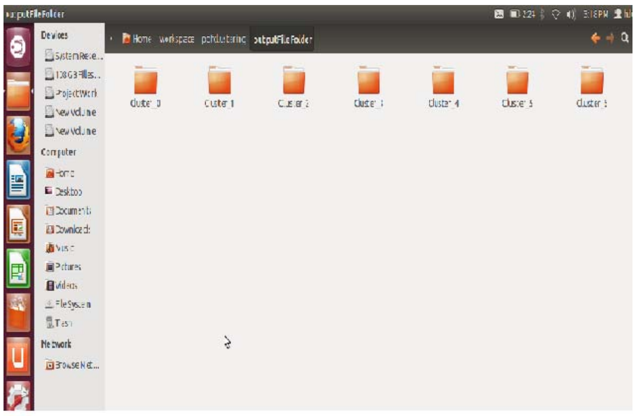


Figure 4: Hierarchical clusters

7. Conclusion and Future Scope

The paper has introduced a document clustering based on Hadoop distributed system. The System will find Tf-Idf value and cosine similarity on MapReduce and it produce partition and hierarchical clusters and its execution time is exponentially decreases with increase in number of node on Hadoop. Our work can be extended further for email document clustering, face book comments and movie review comments clustering. Hadoop has been provided a platform to implement the real world problem and to reduce the computing time complexity.

8. Acknowledgement

I would like to give my sincere gratitude to our guide Prof. V.S. Nandedkar who encouraged and guided me throughout this paper. I am especially grateful H.O.D Prof. Y. B. Gurav for their valuable guidance and encouragement. I also thanks to PG CO-ORDINATOR, Prof. N.D. Kale for their valuable guidance and for allowing me to use the college resources and facilities provided by them. Last but not the least, many thanks and deep regards to Principal sir for their support and encouragement.

References

[1] conf Jian Wan, Wenming Yu and Xianghua Xu," Design and Implementation of Distributed Document Clustering Based on MapReduce," in proc. 2nd ISCST., 2013, pp. 278-

[2] Jingxuan Li Bo shao, Tao Li and Mitsunori Ogihara, "Hierarchical Co- Clustering: A New Way to Oraganize the Music Data," IEEE Trans. vol.14, no. 2, pp. 471-481, 2012

[3] Hierarchical co-clustering- A new way to organize the music data, Published by jingxuam Li,Bo Sh. IEEE Trans 2010 .

[4] Eui-Hong Han and George Karypis "Centroid-based Document Classification: Analysis & Experimental Result", 2001.

[5] Jiawei Han, MichelineKamber,"Data Mining Concepts and Techniques", Second Edition, Morgan Kaufman Publishers, 2006 Elsevier In.

[6] Benjamin C.M. Fung, Ke Wang, Martin Ester, "Hierarchical Document Clustering Using Frequent Itemsets",by SIAM, pp. 59-70,2003.

[7] Tian Xia, "An Improvement TF-IDF:Termlished by Journal Distribution based Term Weight Algorithm", published by Journal of SW Engg. Vol. 6, No. 3, March 2011.

[8] Tom White, "Hadoop: The Definitive Guide", First Edition, Published by Reilly Media, June 2009, in United States of America.ao,Tao Li , IEEE transaction,2012.

[9] Konstantin Shvachko, HairongKuang, "The Hadoop Distributed File System", Published by IEEE, 2010.

[10] Chu, C.-T., S.K. Kim, and Y.-A. Lin, Mapreduce for machine learning on multicore, in In Proceedings of Neural Information Processing Systems Conference (NIPS) 2007.