ISSN: 2321-8169
409 - 413

Document Clustering with Map Reduce using Hadoop Framework

Satish Muppidi*
Department of IT,
GMRIT, Rajam, AP, India
msatishmtech@gmail.com

M. Ramakrishna Murty
Department of CSE
GMRIT, Rajam, AP, India
ramakrishna.malla@gmail.com

Abstract—Big data is a collection of data sets. It is so enormous and complex that it becomes difficult to processes and analyse using normal database management tools or traditional data processing applications. Big data is having many challenges. The main problem of the big data is store and retrieve of the data from the search engines. Document data is also growing rapidly in the eon of internet. Analysing document data is very important for many applications. Document clustering is the one of the important technique to analyse the document data. It has many applications like organizing large document collection, finding similar documents, recommendation system, duplicate content detection, search optimization. This work is motivated by the reorganization of the need for a well efficient retrieve of the data from massive resources of data repository through the search engines. In this work mainly focused on document clustering for collection of documents in efficient manner using with MapReduce.

Keywords—*Document Clustering, Map-Reduce, Hadoop, Document pre-processing*

*****

I. 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. 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. The web documents in each cluster can be pre-processed clustered on Map Reduce.

Now a day's huge data is producing by many social networking websites, e-commerce websites, and many organizations. Analysing this huge data is tedious task for any organization. To analyse the huge data, database management techniques may not be sufficient, so the big data came into existence.

Big data is a collection of data sets that are enormous and complex to manipulate or cross-examine with standard algorithms or techniques. Now a day's Big data is a very popular term used to designate the exponential growth and availability of both structured and unstructured data. Here the question is what big data is and what small data is. For some organizations small data can be a big data for some other organizations.

Big data has challenges can be categorized into 3 challenges. They are 1. Volume: Big volume of data is gathered and growing and growing daily , 2. Variety: Data is all sorts of variety, and the data is not organized data, that is some data is audios, videos, images, text messages, emails, documents, books, log files, public and private records, transactions, so this Big data contains both structured and unstructured data. Big data is really challenging. It's challenging because of the variety of data that it covers – from structured data, such as transactions we made daily or we calculate and store, to unstructured data such as audio files, multimedia presentations and video streams. 3. Velocity: Lot of data is coming at very high speed. The main problem of the big data is store and retrieve of the data from the search engines, recommendation engines, fraud detection etc. Big data is creating the challenges, and the Hadoop is addressing these.

A. *Hadoop*

Hadoop is set of tools that supports running of applications on Big Data. Hadoop addresses the challenges in the Big Data. Apache Hadoop is a software framework that is used to easily write and run applications that process huge amounts of data. Hadoop supports distributed applications, and allows the distributed processing of bulk data sets through the commodity servers. The Apache Hadoop project developed software of open source type for reliable, scalable, efficient, economical and distributed computing. Hadoop implements map-reduce using Hadoop Distributed File System. Hadoop is written with large clusters of computers in mind. The applications runs on the Hadoop need a write-once-read-many access model. The Apache Hadoop software library permits for the disseminated processing of huge data sets through clusters of nodes by simple programming models. Hadoop software library is developed to scale up from single servers up to many machines, each one offering local computation and local storage. The library is designed to sense and handle failures at the application layer, not like in the way of hardware to deliver high availability. So that the data is disseminates a highly- available service on top of the cluster computers. The Hadoop framework has four modules, those are Hadoop Common, Hadoop Distributed File System (HDFS), Hadoop YARN, and Hadoop MapReduce.

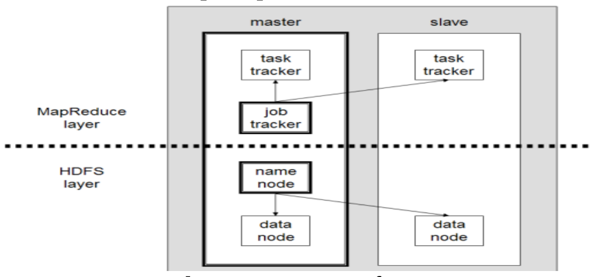


Fig 1: Hadoop Consisting of 2 main parts

The underlying Hadoop Distributed File System (HDFS) utilized by the Hadoop framework is targeted at providing high throughput at the cost of increased latency.

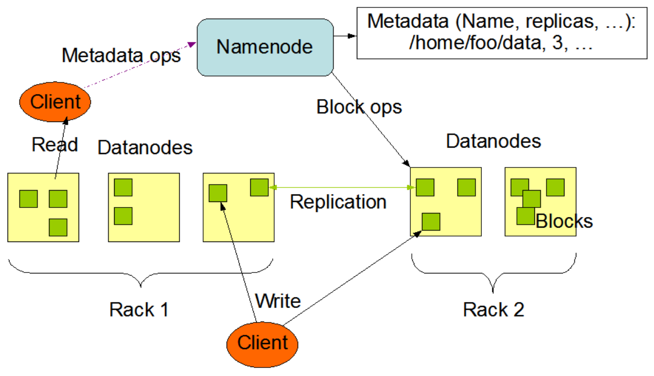
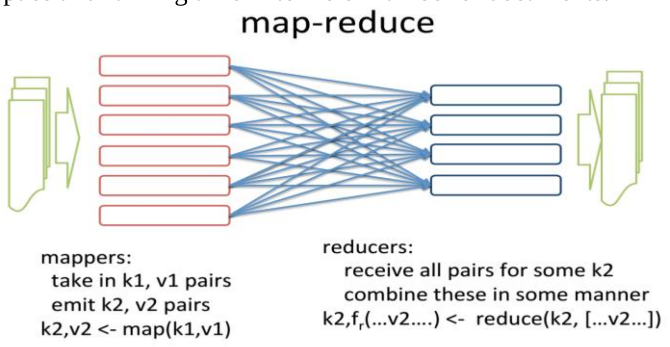


Fig 2: HDFS Architecture

Hadoop is breaking the data into smaller pieces. That's why it is able to deal with the Big data. So after breaking the data into smaller pieces the Hadoop is doing computation in the following way. Hadoop breaks the computation also into smaller pieces and then its sends each slice of computation into each slice of data. The data is broken down into equal pieces, then the computation will finish, then their results are combined together, then this is what is sent back to the application as a combined result. So this approach is called the Map Reduce. This will do the computation into pieces and combined the results and send back to the application.

B. *Map Reduce*

Map/Reduce approach has been popular in order to compute huge volumes of data Map/Reduce motivates to redesign and convert the existing sequential algorithms to Map/Reduce algorithms for big data. So that, this paper presents the document clustering with Map/Reduce using Hadoop framework. The algorithm is to sort data set and to convert it to (key, value) pair to fit with Map/Reduce. The operations of distributing, aggregating, and reducing data in Map/Reduce should cause the bottle-necks. MapReduce is a striking framework because it allows users to decompose the data involved in computing documents similarity into multiplication and summation stages separately in a way that it is well matched to effective disk access across several nodes. The Map-Reduce approach definitely shows the growth in the space and running time in terms of number of documents.


map-reduce
mappers:
take in k1, v1 pairs
emit k2, v2 pairs
k2,v2 <- map(k1,v1)
reducers:
receive all pairs for some k2
combine these in some manner
k2,f,(...v2....) <- reduce(k2, [...v2...])

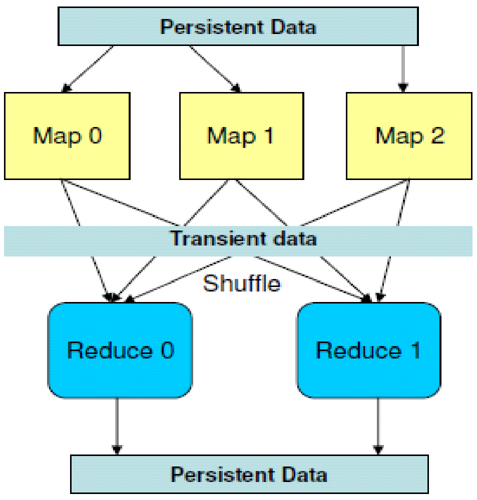


Fig 3: Map Reduce Data Flow

Map/Reduce is an algorithm used in Artificial Intelligence as functional programming. Solve the problems to analyse huge volumes of data set in distributed computing environment. Map/Reduce programming platform is implemented in the Apache Hadoop project that develops open-source software for reliable, scalable, economical, efficient and distributed computing. Users normally stores data rows in labelled tables. A data row has a category of key and a number of columns. The table stored lightly, so that rows in the same table can have different columns. Hadoop can compose hundreds of nodes that process and compute peta- or tera-bytes of data working together. The map and reduce functions run on distributed nodes in parallel. Each map operation can be processed independently on each node and all the operations can be performed in parallel.

Map/Reduce does not guarantee to increase the performance even though we add more nodes because there is a problem for distributing, aggregating, and reducing the large data set among nodes against computing powers of additional machines. The Map Reduce algorithm is for efficiently computing pair wise document similarity in large document collections. In addition to offering specific benefits for a number of real-world tasks, that could be useful for a broad range of text analysis problems.

II. Document Clustering

Before document clustering pre-processing is the one of the viral method to get efficient results. In the following described pre-processing method.

A. *Stop word elimination*

Stop words are words which are filtered out prior to, or after, processing of natural language data. A stop word is a commonly used word in our daily life, that a search engine has been programmed to ignore, both when searching and when retrieving them as a result of a search query. The major work is to identify the mostly weighted words are called as keywords for the documents that reduce the dimensions of the matrix. Stop word elimination is done based on ASCII values of each letter without considering the case (either lower case or upper case) and sum the each letter corresponding ASCII value for every word and generate the number. Assign number to corresponding word, and keep them in sorted order.

B. *Stemming*

Stemming is the process for reducing modified words to their stem, base or root forms generally a written word form. Stem is a part of a word like ing, er, etc., The term is used with slightly different meanings. An algorithm for removing derivations endings and inflectional in order to reduce word forms to a common stem. In this stemming algorithm the suffixes and prefixes were eliminated according to the conditions by which the stemming procedure was applied.

C. *Clustering*

Clustering is a group of similar objects. Cluster is a collection of data objects, that are contains all similar objects, and these objects are dissimilar to the other clusters objects. A cluster of data objects can be treated collectively as one group and so may be considered as a form of data compression. Objects within the cluster have the high similarity in comparison to one another but are dissimilar to objects in the other clusters. Clustering principle:" Maximizing the intra class similarity and minimizing the inter class similarity".

III. MAP REDUCE

Map reduce is data parallel paradigm with message passing. It is a pipelined procedure with two phases, that is map phase and the reduce phase. It's a higher level abstraction, programmers' needs to specify what mapper needs to do and reducer needs to do. Map-Reduce is the software framework for writing the programs easily which processes huge amounts of data in parallel on large clusters of commodity hardware in a fault-tolerant, reliable manner. A Map-Reduce job usually splits the input data into independent portions which are processed by the map tasks in a parallel manner. The framework sorts the outputs coming from the maps, and gives to the reduce tasks as input. Naturally the input and the output of the jobs are stored in the file system. The Map-Reduce operates completely on <key, value> pairs, that is, the framework sees the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as output of the job. Map function is applied to the every input key-value pair and it generates intermediate key-value pairs. These intermediate <key, value> pairs are sorted and grouped by the key. Reduce function is applied to sorted and grouped intermediate key values. Reduce function emits the resultant key-values.

A. *Map/Reduce Algorithm:*

Mapper:

1. Reads each transaction of input file and generates the data set of the items: (<V1>, <V2>, …, <Vn>) where < Vn>: (vn1, vn2,.. vnm)
2. Sort all data set <Vn> and generates sorted data set <Un>: (<U1>, <U2>, …, <Un>) where < Un>: (un1, un2,.. unm)
3. Loop While <Un> has the next element; note: each list Un is handled individually.
4. End Loop While
5. Data set is created as input of Reducer: (key, <value>) = (ynl, <number of occurrences>)

Reducer:

1. Read (ynl, <number of occurrences>) data from multiple nodes
2. Add the values for ynl to have (ynl, total number of occurrences). The reducer is to accumulate the number of values per key. Thus, its time complexity is O(v) where v is the number of values per key.

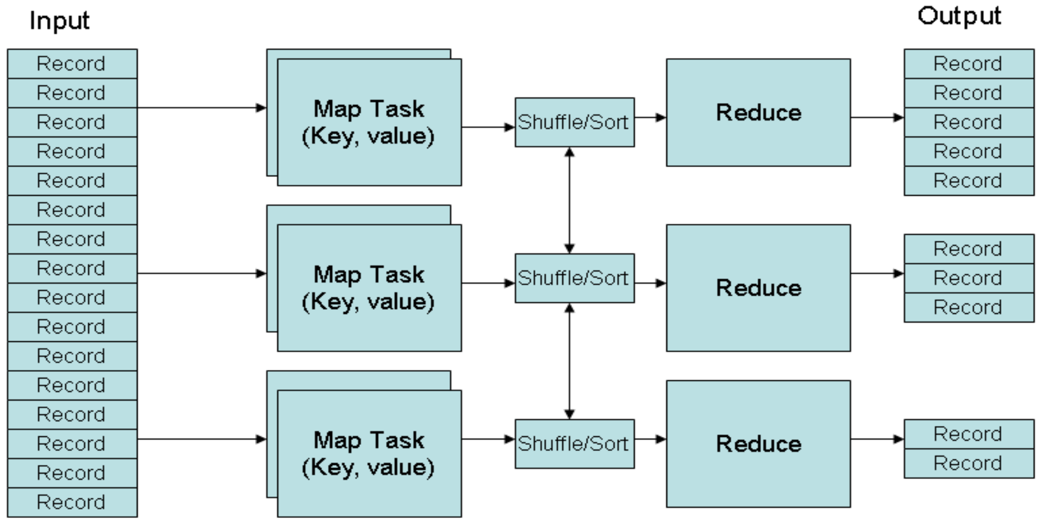


Fig 4: Map Reduce Architecture

B. *K-means Clustering:*

K-Means clustering choose k initial points and mark each as a center point for one of the k sets. Then for every item in the total data set it marks which of the k sets it is closest to. It then finds the average center of each set, by averaging the points which are closest to the set. With the new set of centers (centroid), it repeats the algorithm until convergence has been reached.

The implementation of document clustering on MapReduce accepts two input directories: one is the documents directory with the output of calculating term frequencies, and one is centers directory with k initial document centers. The k initial document centers are chosen from the records of documents directory. Note that the k-line document data have the same terms as fewer as possible. In each of the iteration MapReduce framework will partition the input files of document directory into a set of M splits, and then these splits are processed in parallel by M Map functions.

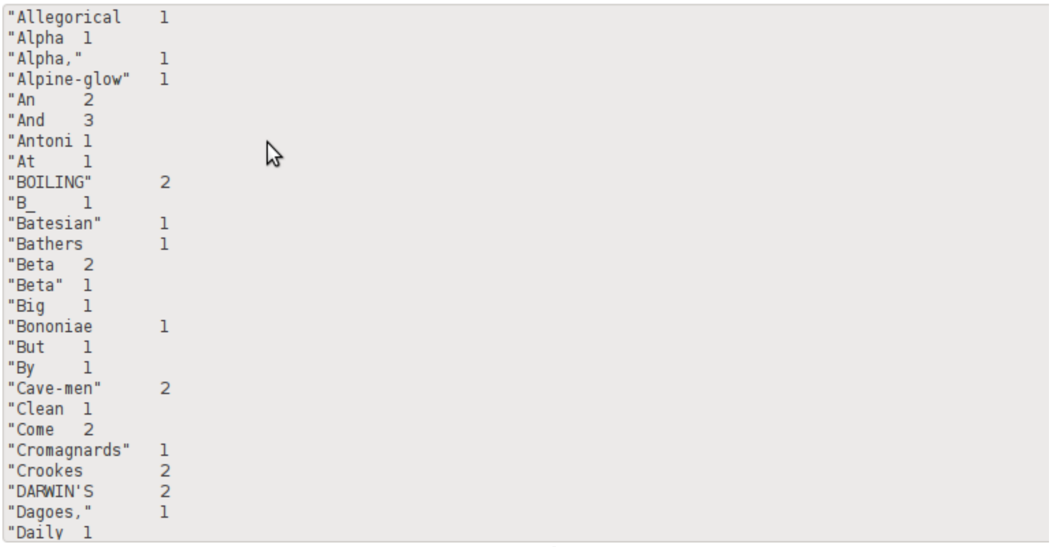
Map functions should determine which of the current set of k document centers; the document is closest to and emits a record containing the entire document's data and its chosen k-center. The Reduce function receives a k-center and all documents which are bound to this k-center. It should calculate a new k-center, and put the new k-center in centers directory. To evaluate the distance between any two documents, we use the cosine similarity metric of term frequency, and use arithmetic average to calculate the new k center. Note that the contents in document directory will not change during the process.

Document clustering for large collection can be efficiently implemented with MapReduce. Initially we perform the pre-processing on the dataset. In the document pre-processing stage, we design a new iterative algorithm to calculate tfidf weight on MapReduce in order to evaluate how important a term is to a document in a corpus. Then, a K-Means clustering is implemented on MapReduce to partition all documents into k clusters in which each documents belongs to the cluster with the same meaning. The map and reduce functions run on distributed nodes in parallel. Each map operation can be processed independently on each node and all the operations can be performed in parallel. In map function, the master node makes parts the input into sub problems, and distributes those to worker nodes. Mapper reads the input data and creates a list of items for each transaction. For each transaction, its time complexity is O(n) where n is the number of items for a transaction. In reduce function, the master node takes the results to all the sub-problems and combines them to get the output.

IV. RESULTS AND DISCUSSIONS

In our experiments we used Hadoop version 0.20.2 an open source Java implementation of Map Reduce running on a cluster. First the documents stop words will be removed. After the stop word removal, then stemming process has been done. So the each document the stop word removal and stemming process has been done. Then by using the tfidf algorithm the word count has taken from each document. Here we have taken the large documents; the size of the document is more than 100 MB.

We have taken 7 large documents for clustering, the following are the screenshots of the clustering documents, and the 7 documents will be clustered into 3 groups.



```
"Allegorical	1
"Alpha	1
"Alpha,"	1
"Alpine-glow"	1
"An	2
"And	3
"Antoni	1
"At	1
"BOILING"	2
"B_	1
"Batesian"	1
"Bathers	1
"Beta	2
"Beta"	1
"Big	1
"Bononiae	1
"But	1
"By	1
"Cave-men"	2
"Clean	1
"Come	2
"Cromagnards"	1
"Crookes	2
"DARWIN'S	2
"Dagoes,"	1
"Daily	1
```

Fig 5: Documents word count

In the following screen we can see that the input to the algorithm is 7 documents.

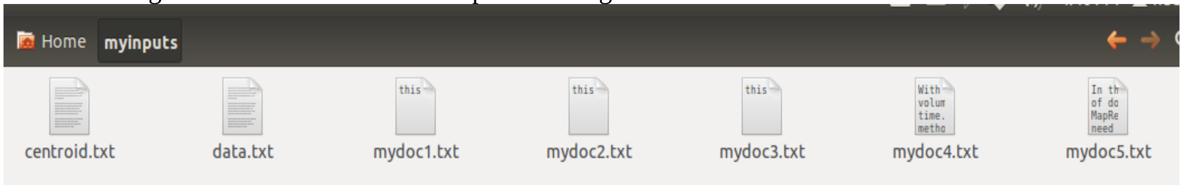


Fig 6: Input to the algorithm

Then the following screen showing that the 7 documents in the input folder will be clustered into cluster0, cluster1 and cluster2. That means here 3 clusters created and the 7 documents will be clustered into 3 clusters. The documents centroid.txt, mydoc5.txt, mydoc1.txt, mydoc3.txt documents entered into cluster0, the documents data.txt entered into cluster1, mydoc2.txt, mydoc4.txt documents entered into cluster2.

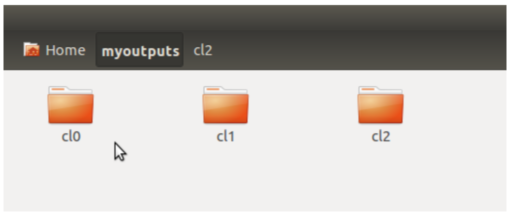


Fig 7: three clusters created

V. Conclusion and future work

The main focus in this work is documents clustering using Hadoop framework. In this work stopword elimination and stemming methods are used to pre-process the input documents and find keywords of every document and make them into vector space model for document clustering. We used iterative algorithm to calculate tfidf weight on MapReduce in order to evaluate important a term is to a document in a corpus. In the process of document clustering used K-means algorithm procedure with distributed environment. We used a MapReduce algorithm for efficiently computing pair wise document similarity in large document collections. The map and reduce functions run on distributed nodes in parallel. Each map operation can be processed independently on each node and all the operations can be performed in parallel.

Hadoop with Map/Reduce motivates the needs to propose new algorithms for the existing applications that have had algorithms for sequential computation. In the future work we need to improve the scalability of the algorithm and also cluster the document with different versions of the K-means algorithm like incremental, bisecting and fuzzy c-means for efficient results of document clustering.

References

[1] Rui Máximo Esteves, Thomas Hacker, Chunming Rong, “A new approach for accurate distributed cluster analysis for Big Data: competitive K-Means” Int. J. Big Data Intelligence, Vol. 1, Nos. 1/2, 2014

[2] M.RamakrishnaMurty, JVRMurty, Prasad Reddy PVGD, “A dimensionality reduced text data clustering with prediction of optimal number of clusters” International journal of Applied Research on information technology and computing Vol 2 Issue 2, 2011.

[3] K. Grolinger, M. Hayes, W. Higashino, A. L'Heureux, D. S. Allison, M. A. M. Capretz, “Challenges for MapReduce in Big Data”, toappear in the Proc. of the IEEE 10th 2014 World Congress on Services (SERVICES 2014), June 27-July 2, 1014, Alaska, USA.

[4] Rachita Sony Krotha and Satish Muppidi, “Performance Evaluation of K-Means Algorithm and Enhanced Mid-point based K-Means Algorithm on Mining Frequent Patterns”, International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE) Vol 3 Issue 10, October 2013.

[5] Tamer Elsayed and Jimmy Lin and Douglas W. Oard, “Pairwise Document Similarity in Large Collection with Map Reduce”, Proceedings of ACL-08, June 2008

[6] Tom white, “Hadoop the definitive guide”.

[7] WoohyuKim and Coord, “MapReduce Debates and Schema-Free”, March 2010.

[8] Jian Wan and Wenming Yu and Xianghua Xu, “Design and Implement of Distributed Document Clustering Based on Map Reduce”, ISCSCT, December 2009

[9] Jongwook Woo, Yuhang xu , “Market Basket Analysis Algorithm with Map/Reduce of Cloud Computing”.

[10] Apache Hadoop Project,, http://hadoop.apache.org/

[11] Apache HBase, http://hbase.apache.org/

[12] Michael Steinbach, George Karypis, and Vipin Kumar “A Comparison of Document Clustering Techniques”

[13] Jimmy Lin and Chris Dyer, “Data-Intensive Text Processing with MapReduce”, Morgan & Claypool Publishers, 2010.

[14] Jongwook Woo, “Market Basket Analysis Example in Hadoop”, http://dalcloudcomputing.blogspot.com/2011/03/market-basket-analysis-example-in.html, March 2011