Volume 5, Issue 9, September 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Enhance the Efficiency of Clustering by Minimizing the Processing Time using Hadoop MapReduce

K. Ramana
M.Tech Scholar, Department of CSE
GMRIT, Rajam, India

A. Venkataramana
Associate Professor, Department of CSE
GMRIT, Rajam, India

***Abstract**— Data is increasing day by day with the Development of Information Technology. Extracting the required information from huge amount of data is a complex and time consuming process. Clustering can be considered the most important unsupervised learning in data mining. K-means clustering is a traditional and popular cluster analysis method in data mining but it is not suitable for large volumes of unstructured data sets. Therefore this paper proposes k-means clustering with Hadoop Map-reducing technique. Hadoop MapReduce is a software framework for easily writing applications which process vast amounts of data in-parallel on large clusters. Experimental results found that for clustering large applications, k-means with Map-reducing approach significantly reduces the processing time when comparing with traditional k-means without map reducing, i.e, processing time for a data sample without map-reducing technique is 50 minutes, whereas for the same data processing time with map-reducing technique is only 30 minutes.*

***Keywords**— Clustering; k-means; TF-IDF; Hadoop; MapReduce.*

I. INTRODUCTION

Clustering is the task of assigning a set of objects into groups (called clusters) so that the objects in the same cluster are more similar (in some sense or another) to each other than to those in other clusters. The Principle of Clustering is Maximizing the Intra class Similarity and Minimizing the Inter class Similarity. The k-means clustering algorithm is known to be efficient in clustering large data sets. This clustering algorithm was developed by MacQueen, and is one of the simplest and the best known unsupervised learning algorithms that solve the well-known clustering problem. The K-Means algorithm aims to partition a set of objects, based on their features, into k clusters, where k is a predefined or user-defined constant. The main idea is to define k centroid, one for each cluster. The centroid of a cluster is formed in such a way that it is closely related (in terms of similarity function; similarity can be measured by using different methods such as cosine similarity, Euclidean distance, Extended Jaccard) to all objects in that cluster. The main Disadvantages of K-means clustering algorithm are Delivers low latency retrieval, Works on relatively small amounts of data. Therefore the traditional k-means is not suitable for large volumes of unstructured data.

Now a day’s every organization maintaining and analyzing large sets of data to discover patterns and other useful information. Big data analytics refers to the process of collecting, organizing and analyzing large sets of data. Big data analytics can help organizations to better understand the information contained within the data and will also help identify the data that is most important to the business and future business decisions. Big data analysts basically want the knowledge that comes from analyzing the data. The exponential growth of data presented challenges to cutting-edge businesses such as Google, Yahoo, Amazon, and Microsoft. They needed to go through terabytes and petabytes of data to figure out which websites were popular, what books were in demand, and what kinds of ads appealed to people. Existing tools were becoming inadequate to process such large data sets. Google was the first to publicize MapReduce [5] a system they had used to scale their data processing needs. Reasons for the growth of Big Data are Increase of storage capacities, Increase of processing power and Availability of data.

Big data can be analyzed with the software tools commonly used as part of advanced analytics disciplines such as predictive analytics, data mining, text analytics and statistical analysis. The semi-structured and unstructured data may not fit well in traditional data warehouses based on relational databases. Furthermore, data warehouses may not be able to handle the processing demands posed by sets of big data that need to be updated frequently or even continually -- for example, real-time data on the performance of mobile applications or of oil and gas pipelines. As a result, many organizations looking to collect, process and analyze big data have turned to a newer class of technologies that includes Hadoop and related tools such as YARN, Map-Reduce, Spark, Hive and Pig as well as NoSQL databases. Those technologies form the core of an open source software framework that supports the processing of large and diverse data sets across clustered systems.

Hadoop is a platform that provides both distributed storage and computational capabilities. It is an open source software project that enables the distributed processing of large data sets across clusters of commodity servers. It is designed to scale up from a single server to thousands of machines, with a very high degree of fault tolerance. Hadoop is a distributed master-slave architecture that consists of Hadoop distributed file system (HDFS) [3],[5]for storage and Map-Reduce for computational capabilities. Rather than relying on high-end hardware, the resiliency of these clusters comes from the software's ability to detect and handle failures at the application layer. In a "normal" relational database, data is found and analyzed using queries, based on the industry-standard Structured Query Language (SQL). Non-relational databases use queries, too; they're just not constrained to use only SQL, but can use other query languages to pull information out of data stores. Hadoop is more of a data warehousing system - so it needs a system like Map-Reduce to actually process the data. Hadoop can handle all types of data from disparate systems: structured, unstructured, log files, pictures, audio files, communications records, email. Even when different types of data have been stored in unrelated systems, you can dump it all into your Hadoop cluster with no prior need for a schema. In other words, you don't need to know how you intend to query your data before you store it.

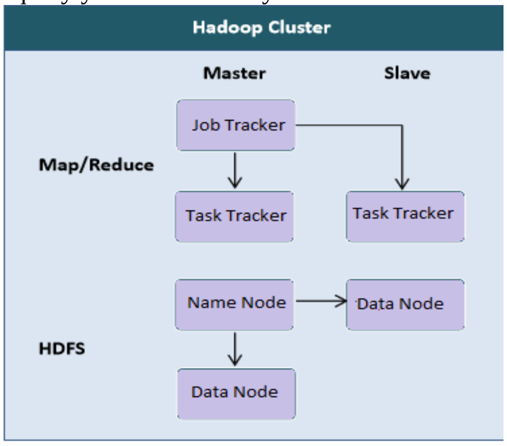


Fig 1: Hadoop Cluster

II. LITERATURE REVIEW

Following are the early implementations of k-means along with the MapReduce framework in order to implement the algorithm in parallel manner: Weizhong Zha et.al [1]. proposed a fast parallel k -means clustering algorithm based on MapReduce, which has been widely embraced by both academia and industry. They use speedup, scaleup and sizeup to evaluate the performances of proposed algorithm. P. ZHOU et.al[2]. design and realize a parallel K- Means algorithm based on MapReduce framework with Large-Scale Data Sets. They use Vector space model ,It is the most widely used in information retrieval. This model uses feature entries and their weights to express the document information. In VSM method DFR and TFIDF are calculated. Sandip A. Ganage et.al [3].presented the acceleration of a widely used data clustering algorithm using Hadoop & MapReduce framework, in the context of heterogeneous computing devices like CPUs and GPUs and implemented k-means using OpenCL heterogeneous computing platform with the help of Hadoop-MapReduce framework. OpenCL is a heterogeneous computing platform and one of the widely used for GPU Computing. Xianfeng Yang et.al [4]. Achieved Speedup Comparison of the CLARA1,CLARA2,PAM, and Newman Algorithms with the help of Hadoop MapReduce Technique. Kamble Ashwini et.al [5].Presented a paper on Improve the Performance of Mapreduce on hadoop. They analyzing the process of Map tasks in Hadoop ,the serial execution of data transmission and data processing is discovered. TamerElsayed et.al [6]. presented a MapReduce algorithm for computing pairwise document similarity in large document collections.They implemented the symmetric variant of Okapi- BM25 as the similarity function and used the AQUAINT-2 collection of newswire text. JianWan et.al [7].Design and Implemented of Distributed Document Clustering Based on MapReduce. They Calculated tfidf and implementation of K-Means for Document Clustering. ZHAO Weizhong [8].Researchs on Parallel K-means Algorithm Design Based on Hadoop Platform. Previous implementations of K-Means algorithm by using hadoop mapreduce with small amount of data like 50mb (50000 documents). This paper proposed a new method which divides the original algorithm into two different functions „Map" and "Reduce" which are executed separately one after another. Large amount of data (450MB i.e 150000 documents) can be given to the K-Means as input to Processing in less time. However such input requires a framework to handle the input, like a File system. This paper proposes an idea of using a distributed file system which handles the input and output data very efficiently.

III. HADOOP MAPREDUCE

With the rapid development of the Information Technology, huge volumes of data need to be processed in a short time. Clustering is the process of organizing objects into groups whose members are similar in some way. K-means clustering is a traditional and popular cluster analysis method in data mining. However traditional data mining Algorithms face many challenges while dealing with high dimensional data. To overcome these problems this we proposes an efficient and advanced clustering algorithm called parallel k-means clustering by using Hadoop MapReduce [3],[7]. Hadoop MapReduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.

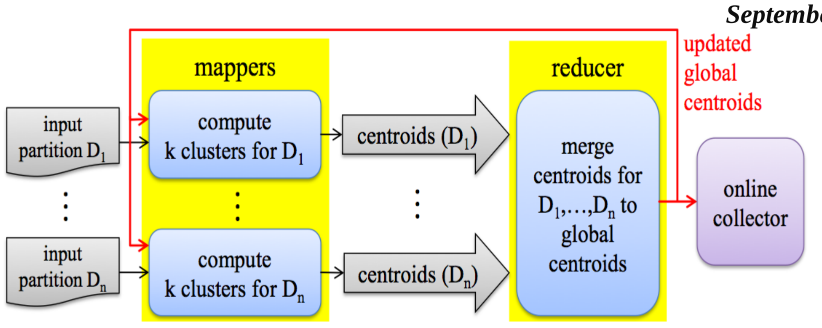


Fig 2: Map-Reduce Execution

A MapReduce job usually splits the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically both the input and the output of the job are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks. The MapReduce framework [6] operates exclusively on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value>pairs as the output.

IV. METHODOLOGY

Before performing clustering on hadoop mapreduce, document pre-processing is needed i.e divide a input data set into subsets. In this process we should Calculate TF-IDF , It's stands for term frequency-inverse document frequency, is a numerical statistics which reflects how important a word is to a document in a collection or corpus, it is the most common weighting method used to describe documents in the Vector Space Model, particularly on IR problems. The number of times a term occurs in a document is called its term frequency. We can calculate the term frequency for a word as the ratio of number of times the word occurs in the document to the total number of words in the document. The inverse document frequency is a measure of whether the term is common or rare across all documents. The map function calculated the distance between each document and each cluster centre after that reads the input data and calculates the distance to each center. For every document, map function produces an output pair with: <key(cluster id) ,value (coordinates of documents)>. We can use a combine function to reduce the size before sending it to Reduce. The Combine function calculates the average of the coordinates for each cluster id, along with the number of documents. All data of the same current cluster are sent to a single reducer. The reduce function, compute new cluster center coordinates. Its output is written to the cluster file, and contains: iteration number, cluster id, number of documents assigned to the cluster. finally the new cluster coordinates are compared to the original ones. If the criterion function converges, the program end, and we have found the clusters.

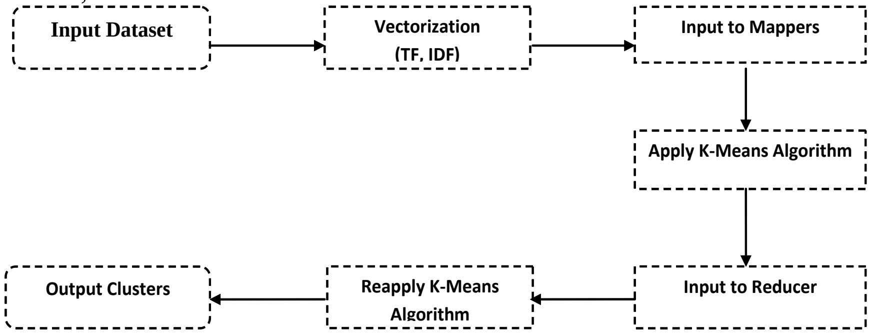
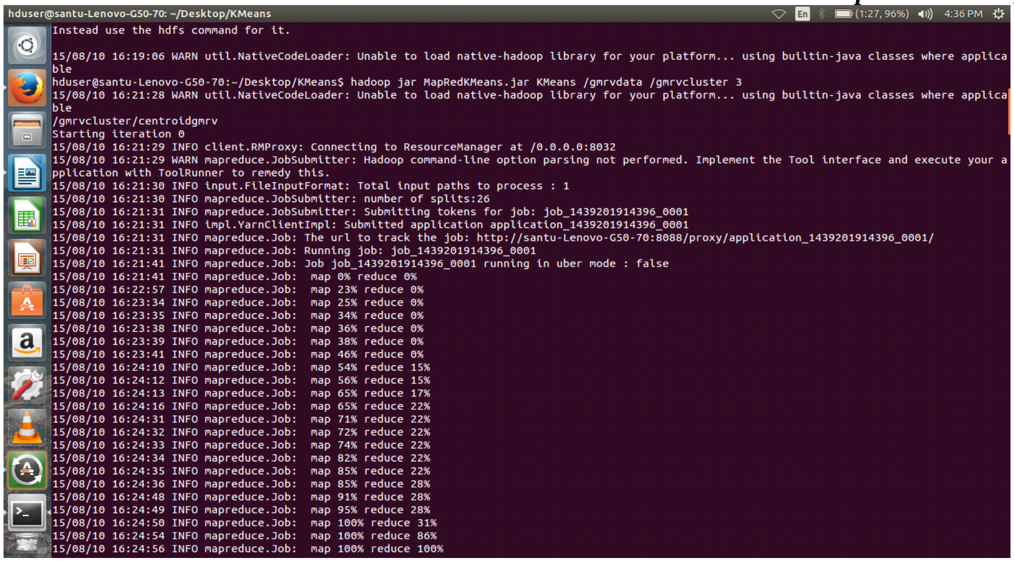


Fig 3: Map-Reduce Procedure

Input Data Set for This Project is NSF (National Science Foundation) Data .The data set consists of 1,50000 abstracts (Text Documents) describing NSF awards for basic research, bag-of-word data files extracted from the abstracts, and a list of words used for indexing the bag-of-word data. The bag-of-word data consists of 2-column and 3-column files. The bag-of-word data was produced by automatically processing the abstracts with a text analyzer called NSFABST, built using Visual Text. While most fields of the output are very accurate, the authors were not extracted from the Investigator: field with 100% accuracy, due to wide variability in that field.

V. RESULTS

This Project improves the efficiency of clustering with respect to time and scaleup. Scaleup evaluates the ability of the algorithm to grow both the system and the dataset size. The data used in this experiment is taken from text classification corpus of NSF. The corpus is the text documents on different awarded topics. We choose a number of documents to cluster and verify the effectiveness and feasibility of the algorithm. The objective of this paper is to reduce the execution time by using K-Means map-reduce algorithm on hadoop with single node cluster. In this paper the results have been presented by analyzing different sizes of data with and without mapreduce techniques. The screen shots of the map-reduce execution process is shown in Fig 4.

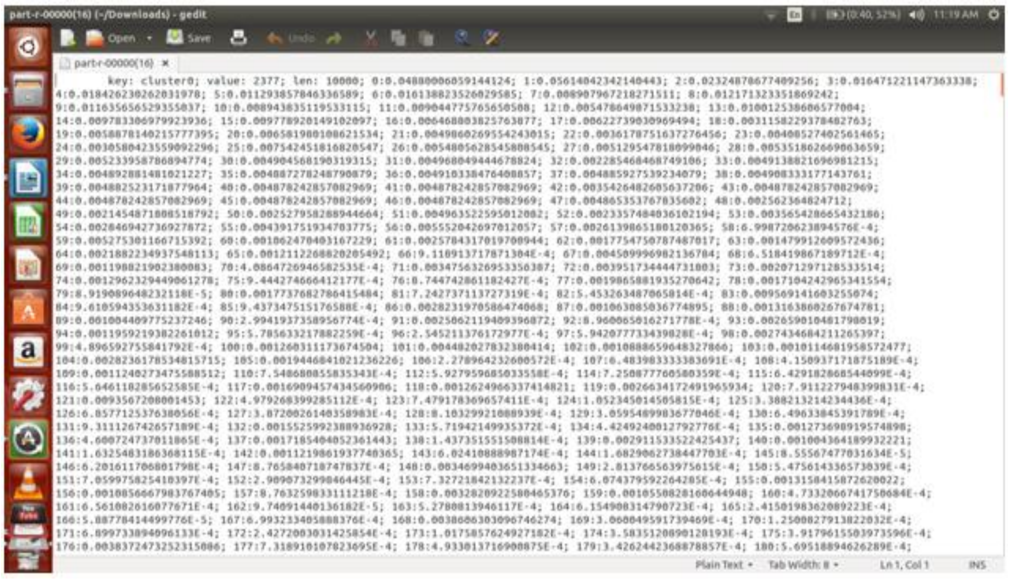


```
hduser@santu-Lenovo-G50-70: ~/Desktop/KMeans
Instead use the hdfs command for it.

15/08/10 16:19:06 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applica
ble
hduser@santu-Lenovo-G50-70:~/Desktop/KMeans$ hadoop jar MapRedKMeans.jar KMeans /gmrvdata /gmrvcluster 3
15/08/10 16:21:28 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applica
ble
/gmrvcluster/centroidgmrv
Starting iteration 0
15/08/10 16:21:29 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
15/08/10 16:21:29 WARN mapreduce.JobSubmitter: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your a
pplication with ToolRunner to remedy this.
15/08/10 16:21:30 INFO input.FileInputFormat: Total input paths to process : 1
15/08/10 16:21:30 INFO mapreduce.JobSubmitter: number of splits:26
15/08/10 16:21:31 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1439201914396_0001
15/08/10 16:21:31 INFO impl.YarnClientImpl: Submitted application application_1439201914396_0001
15/08/10 16:21:31 INFO mapreduce.Job: The url to track the job: http://santu-Lenovo-G50-70:8088/proxy/application_1439201914396_0001/
15/08/10 16:21:31 INFO mapreduce.Job: Running job: job_1439201914396_0001
15/08/10 16:21:41 INFO mapreduce.Job: Job job_1439201914396_0001 running in uber mode : false
15/08/10 16:21:41 INFO mapreduce.Job:  map 0% reduce 0%
15/08/10 16:22:57 INFO mapreduce.Job:  map 23% reduce 0%
15/08/10 16:23:34 INFO mapreduce.Job:  map 25% reduce 0%
15/08/10 16:23:35 INFO mapreduce.Job:  map 34% reduce 0%
15/08/10 16:23:38 INFO mapreduce.Job:  map 36% reduce 0%
15/08/10 16:23:39 INFO mapreduce.Job:  map 38% reduce 0%
15/08/10 16:23:41 INFO mapreduce.Job:  map 46% reduce 0%
15/08/10 16:24:10 INFO mapreduce.Job:  map 54% reduce 15%
15/08/10 16:24:12 INFO mapreduce.Job:  map 56% reduce 15%
15/08/10 16:24:13 INFO mapreduce.Job:  map 65% reduce 17%
15/08/10 16:24:16 INFO mapreduce.Job:  map 65% reduce 22%
15/08/10 16:24:31 INFO mapreduce.Job:  map 71% reduce 22%
15/08/10 16:24:32 INFO mapreduce.Job:  map 72% reduce 22%
15/08/10 16:24:33 INFO mapreduce.Job:  map 74% reduce 22%
15/08/10 16:24:34 INFO mapreduce.Job:  map 82% reduce 22%
15/08/10 16:24:35 INFO mapreduce.Job:  map 85% reduce 22%
15/08/10 16:24:36 INFO mapreduce.Job:  map 85% reduce 28%
15/08/10 16:24:48 INFO mapreduce.Job:  map 91% reduce 28%
15/08/10 16:24:49 INFO mapreduce.Job:  map 95% reduce 28%
15/08/10 16:24:50 INFO mapreduce.Job:  map 100% reduce 31%
15/08/10 16:24:54 INFO mapreduce.Job:  map 100% reduce 86%
15/08/10 16:24:56 INFO mapreduce.Job:  map 100% reduce 100%
```

Fig 4: Map-Reduce Execution Process

The MapReduce framework operates exclusively on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value>pairs as the output. The screen shot of cluster output on Hadoop browser (HDFS) is shown in Fig 5.



```
part-r-00000(16) (~/Downloads) - gedit
        key: cluster0; value: 2377; len: 10000; 0:0.04880006059144124; 1:0.05614042342140443; 2:0.02324870677409256; 3:0.01647122114736333 8;
4:0.01842623026203197 8; 5:0.011293857846336589; 6:0.01613882352602958 5; 7:0.00890796721827151 1; 8:0.01217132335186924 2;
9:0.011635656529355037; 10:0.008943835119533115; 11:0.009044775765650508; 12:0.005478649871533238; 13:0.010012538606577004;
14:0.009783306979923936; 15:0.009778920149102097; 16:0.006468803825763877; 17:0.006227390309694 94; 18:0.003115822937840276 3;
19:0.005887814021577739 5; 20:0.006581980108621534; 21:0.004986026955424301 5; 22:0.003617875163727645 6; 23:0.004005274025614 65;
24:0.003058042355909229 6; 25:0.007542451816820547; 26:0.005480562854580854 5; 27:0.005129547810099046; 28:0.005351862669063659;
29:0.005233958786894774; 30:0.004904568190319315; 31:0.004968049444678824; 32:0.002285468468749106; 33:0.004913882169698121 5;
34:0.004892881481021227; 35:0.004887278248790879; 36:0.004910338476408857; 37:0.004885927539234079; 38:0.004908333177143761;
39:0.004882523171877964; 40:0.004878242857082969; 41:0.004878242857082969; 42:0.003542648260563720 6; 43:0.004878242857082969;
44:0.004878242857082969; 45:0.004878242857082969; 46:0.004878242857082969; 47:0.004865353767835602; 48:0.002562364824712;
49:0.002145487100851879 2; 50:0.002527958288944664; 51:0.004963522595012082; 52:0.002335748403610219 4; 53:0.003565428665432186;
54:0.002846942736927872; 55:0.004391751934703775; 56:0.005552042697012057; 57:0.002613986518012036 5; 58:6.998720623894576E-4;
59:0.005275301166715392; 60:0.001062470403167229; 61:0.002578431701970094 4; 62:0.001775475078748701 7; 63:0.001479912609572436;
64:0.002188223493754811 3; 65:0.001211226882020549 2; 66:9.110913717871304E-4; 67:0.004509996982136784; 68:6.518419867189712E-4;
69:0.001198821902380083; 70:4.086472694658253 5E-4; 71:0.003475632695335638 7; 72:0.003951734444731003; 73:0.002071297128533514;
74:0.001296232944906127 8; 75:9.444274666412177E-4; 76:8.744742861182427E-4; 77:0.001986588193527064 2; 78:0.001710424296534155 4;
79:8.919089648232118E-5; 80:0.001773768278641548 4; 81:7.242737113727319E-4; 82:5.453263487065814E-4; 83:0.009569141603255074;
84:9.610594353631182E-4; 85:9.437347515176588E-4; 86:0.002823197058647406 8; 87:0.001063085036774895; 88:0.001316386026767478 1;
89:0.001004409775237246; 90:2.994193735895677 4E-4; 91:0.002506211940939687 2; 92:8.960065016271778E-4; 93:0.002659010481798019;
94:0.001195921938226101 2; 95:5.785633217882259E-4; 96:2.545211376172977E-4; 97:5.942077733439828E-4; 98:0.002743468421126539 7;
99:4.896592755841792E-4; 100:0.001260311173674504; 101:0.004482027832380414; 102:0.001088865964832786 0; 103:0.001011468195857247 7;
104:0.002823617853481571 5; 105:0.001944684102123622 6; 106:2.278964232600572E-4; 107:6.483983333383691E-4; 108:4.150937171875189E-4;
109:0.001124027347558851 2; 110:7.540680055835343E-4; 112:5.927959685033558E-4; 114:7.250877760580359E-4; 115:6.429182868544099E-4;
116:5.646118285652585E-4; 117:0.001690945743456090 6; 118:0.001262496633741482 1; 119:0.002663417249196593 4; 120:7.911227948399831E-4;
121:0.009356720800145 3; 122:4.979268399285112E-4; 123:7.479178369657411E-4; 124:1.052345014505815E-4; 125:3.388213214234436E-4;
126:6.857712537638056E-4; 127:3.872002614035898 3E-4; 128:8.103299210889 39E-4; 129:3.059548998367704 6E-4; 130:6.496338453917 89E-4;
131:9.311126742657189E-4; 132:0.001552599238893692 8; 133:5.719421499353 72E-4; 134:4.424924001279277 6E-4; 135:0.001273698919574898;
136:4.600724737011865E-4; 137:0.001718540405236144 3; 138:1.437351551508814E-4; 139:0.002911533522425437; 140:0.001004364189932221;
141:1.632548318636811 5E-4; 142:0.001121986193774036 5; 143:6.024108889871 74E-4; 144:1.682906273844770 3E-4; 145:8.555674770316 34E-5;
146:6.201611706801798E-4; 147:8.765840718747837E-4; 148:0.003469940365133466 3; 149:2.813766563975615E-4; 150:5.475614336573039E-4;
151:7.059975825410397E-4; 152:2.909073299846445E-4; 153:7.327218421322 37E-4; 154:6.074379592264285E-4; 155:0.001315841587262002 2;
156:0.001085666798376740 5; 157:8.763259833111218E-4; 158:0.003282092258046537 6; 159:0.001055082816064494 8; 160:4.733206674175068 4E-4;
161:6.561082616077671E-4; 162:9.740914401361 82E-5; 163:5.278081394611 7E-4; 164:6.154908314790723E-4; 165:2.415019836208922 3E-4;
166:5.887784144997 76E-5; 167:6.993233405888376E-4; 168:0.003808630309674627 4; 169:3.060049591739469E-4; 170:1.250082791382203 2E-4;
171:6.899733894096133E-4; 172:2.427200303142585 4E-4; 173:1.017585762492718 2E-4; 174:3.583512089012019 3E-4; 175:3.917961550397359 6E-4;
176:0.003837247325231500 6; 177:7.318910107823695E-4; 178:4.933013716900875E-4; 179:3.426244236887885 7E-4; 180:5.695188946262 89E-4;
                                                                    Plain Text ▾   Tab Width: 8 ▾      Ln 1, Col 1      INS
```

Fig 5: Cluster output on Hadoop Browser (HDFS)

Time taken for Map-Reduce Execution Process for iteration1 is CPU Time spent 178010 ms for Iteration 2 CPU time spent 774940ms and for Iteration 3 CPU time spent 792150ms. finally total CPU time spent for K-means map-reduce is 1745 seconds i.e 29.08 minutes nearly 30 minutes. Map-reducing approach significantly reduces the processing time for large data sets when comparing with traditional k-means without map reducing i.e, processing time for 450 MB data without map-reducing technique is 50 minutes, whereas for the same data processing time with map-reducing technique is only 30 minutes. Experimental results are shown in figure 6.

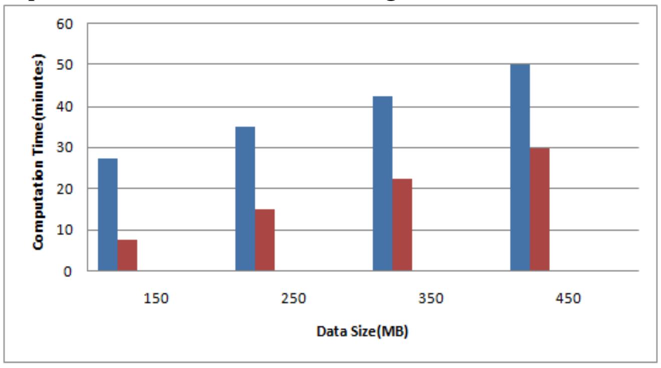


Fig 6: Variation in processing time between K-means and Mapreduce K-means

VI. CONCLUSION

Information retrieval techniques are widely popular in most of the search engines to efficiently organize and retrieve information systems. Most of the data in internet is in the format of unstructured and semi structured. Currently clustering techniques are used to organize and group the similar data objects to retrieve search results faster. TF-IDF is used for k-means algorithm with supportive similarity measure. As the data is enormously increasing day by day, elastic resources are required to store and compute. Hadoop framework supports to store and compute big data in parallel and distributed platform with the help of HDFS and Map reduce. This project exhibits k-means map reducing approach for clustering text documents. The results obtained by executing map reduce k-means algorithm on single node cluster shows that the performance of the algorithm increases as the text corpus increases. Map-reducing approach significantly reduces the processing time for large data sets when comparing with traditional k-means without map reducing i.e, processing time for 450 MB data without map-reducing technique is 50 minutes, whereas for the same data processing time with map-reducing technique is only 30 minutes.

REFERENCES

[1] Parallel K-Means Clustering Based on MapReduce By Weizhong Zha, Huifang Ma and Qing He, Springer-Verlag Berlin Heidelberg 2009, pp. 674–679.

[2] Large-Scale Data Sets Clustering Based on MapReduce and Hadoop By P. ZHOU, J. LEI and W. YE,Journal of Computational Information Systems, vol. 7, no. 16, pp. 5956- 5963, 2011 .

[3] Heterogeneous Computing Based K-Means Clustering Using Hadoop-MapReduce Framework By Sandip A. Ganage, Dr. R. C. Thool and Heshsham Abdul Basit,International Journal of Advanced Research in Computer Science and Software Engineering,Volume 3, Issue 6, June 2013.

[4] A New Data Mining Algorithm based on MapReduce and Hadoop By Xianfeng Yang and Liming Lian , International Journal of Signal Processing, Image Processing and Pattern Recognition 2014, pp.131-142.

[5] A Brief on MapReduce Performance By Kamble Ashwini and Kanawade Bhavana, IJIRAE 2014

[6] Pairwise Document Similarity in Large Collections with MapReduce By TamerElsayed,JimmyLin and Douglas W.Oard,2008.

[7] Design and Implement of Distributed Document Clustering Based on MapReduce By JianWan, Wenming Yu and Xianghua Xu, ISCSCT 2009, pp. 278-280.

[8] Research on Parallel K-means Algorithm Design Based on Hadoop Platform By ZHAO Weizhong, MA Huifang, FU Yanxiang, and SHI Zhongzhi, Computer Science,2011.38(10):166- 176.

[9] Unique Distance Measure Approach for K-means Clustering Algorithm By WK. Daniel PUN and ABM shawkat ALI, TENCON 2007 IEEE Region 10 Conference, pp. 1-4.

[10] Parallel K-means Algorithm Based on constrained information By YU Yuecheng, WANG Jiandong, ZHENG Guanshengand CHEN Bin Journal Of Southeast University, 2011.41(3):505-508.