Clustering Categorical-Time Evolving Data from K-Means to Rough Set Theory using Map-Reduce Technique

J Uma Mahesh[1], Harekrishna Allu[2]
[1]*Assistant Professor Department of CSE, Geethanjali College of Engineering and Technology*
[2]*Associate Professor Department of CSE, Geethanjali College of Engineering and Technology*
Cheeryal-501301,Hyderabad, Telangana.

Abstract

Clustering is used to classify related data items under similar group but it fails to achieve well for big data due to massive time complexity of allocating unlabeled data point into proper cluster is big task in the categorical data domain, where real time data changes for every instance so for such scenarios in this paper using sampling and parallelization techniques from k-means to rough set theory by extending Hadoop Map Reduce programming we proposed to label the unlabeled data points. An analysis of projected approach to evaluate its efficiency over many other algorithms using standard data sets for testing and shows that the proposed sampling and parallelization technique can process big data efficiently.

Keywords - *Big Data, Data Mining, Hadoop Map Reduce, k- means clustering, rough set Theory.*

I. INTRODUCTION

We are in the world where gigantic amounts of data are collected and analyzing this data is a critical task. In other words we can say that we are living in the data age in which peta bytes (1000 Terabytes) of data is generated from all needs of business, society, science and engineering, medicine etc., Businesses around the world wide are generating large collection of data sets such as sales transactions, sales promotions, stock trading records, product details, company profiles and their performance and customer feedback [1]. For example large stores online and offline like Walmart, amazon, flip-kart, e-bay are handling millions of transactions per day at various branches across the globe. Scientific and engineering applications are generating peta bytes of data by remote sensing, scientific experiments and engineering observations.

The communication networks carry hundreds of peta bytes of data traffic every day. Even the medical and health industry generate large amounts of data by medical records. Web Searches like Google, social media like Facebook, producing images and videos, blogs. These are the list of various sources that generate endless data in huge amounts. The organization face difficulties to create manipulate and manage the large datasets [2]. Thus extracting useful and valuable information from the huge data is difficult and led to data mining. Data mining makes a large collection of data into useful information referred as knowledge.

Clustering is a method for finding a collection of similar objects from a given data set. The algorithms that are developed for numerical data for clustering may be easy to use in normal conditions but not when it comes to categorical data [3], [4], [5]. Clustering is a challenging issue in categorical domain, where the distance between data points is undefined [1]. It is not easy to find out the class label of unknown data point in categorical domain.

Sampling and parallelization techniques accelerate the clustering [6], [7] and the data points that are not sampled are to be allocated into proper clusters. The data which depends on time called as time evolving data [8], [9]. For example, the buying preferences of customers may vary with time, depending on the current day of the week, availability of alternatives, discounting rate etc. [10] Since data is modified and thus evolve with time, the underlying clusters may also change based on time by the data drifting concept [11], [12]. The clustering time-evolving data in the numerical domain [13], [14] has been explored in the previous literature though not in the categorical domain. Categorical attributes also exist in real data with drifting concepts, for example web logs that record the browsing history of users, stock market details, buying records of customers often evolve with time. It is a challenging problem in the categorical domain therefore to evolve a procedure for precise categorization. Previous methods on clustering categorical data focused on doing clustering on the entire data set and drifting concepts were not taken into consideration. The objective is to propose a framework for performing clustering on the categorical time- evolving data.

One of data analysis techniques, rough sets based methods have been successfully applied in data mining and knowledge discovery during last decades [15], [16], [17] and particularly useful for rule acquisition [18], [19], [20] and feature selection[21], [22], [23]. To our Knowledge, most of the traditional algorithms based on rough sets are the sequential algorithms and corresponding tools only run on a single computer to deal with small data sets. To expand the applications of rough sets in the field of data mining and knowledge discovery from big data, we discuss about rough set based parallel methods for knowledge acquisition in this paper. Based on Map Reduce, we design corresponding parallel algorithm for knowledge acquisition on the basis of the characteristics of the data. The proposed algorithm is implemented on Hadoop platform [24] As a result, a rough set based method for performing clustering on the categorical time evolving data is proposed in this paper. This method find out if there is a drifting concept or not while processing the incoming data. However, in the categorical domain, the above procedure is challenging since the numerical characteristics of clusters are difficult to define. In this paper, a mechanism called rough membership function-based similarity is developed to allocate each unclustered categorical data point into the corresponding proper cluster.

Distributed Computing is a technology aimed at solving computational problems mainly by sharing the computation over a network of interconnected systems. Each individual system connected on the network is called a node and the collection of many nodes that form a network is called a cluster. For the purpose of processing big data, Google developed a software framework called Map Reduce to support large distributed data sets on clusters of computers [25], [26] which is effective to analyze large amounts of data. Implementation of Map Reduce is done through Apache Hadoop which is a software framework that helps constructing the reliable, scalable, distributed systems.

Apache Hadoop [27] is an open source framework that supports distributed computing. It came into existence from Google's Map Reduce and Google File Systems projects. It is a platform that can be used for intense data applications which are processed in a distributed environment. It follows a Map and Reduce programming paradigm where the fragmentation of data is the elementary step and this fragmented data is fed into the distributed network for processing. The processed data is then integrated as a whole. Hadoop [27], [28], [29] also provides a defined file system for the organization of processed data the Hadoop Distributed File System HDFS.

The Hadoop framework takes into account the node failures and is automatically handled by it. This makes Hadoop really flexible and a versatile platform for data intensive applications. The answer to growing volumes of data that demand fast and effective retrieval of information lies in the principles of data mining over a distributed environment such as Hadoop. This not only reduces the time required for completion of the operation but also reduces the individual system requirements for computation of large volumes of data. Starting from the Google File Systems [30] and Map Reduce concept, Hadoop has taken the world of distributed computing to a new level with various versions of Hadoop that are now in existence and also under Research and Development. Few of which include Hive [31], Zookeeper [32], Pig [33].

The remaining part of paper is organized as follows: the literature survey of the proposed work described in Section II. Section III illustrates the methodology of k-means to rough set based methods for knowledge acquisition with Map-Reduce, Section IV shows experimental analysis and Section V illustrates Conclusions.

II. LITERATURE SURVEY

H. Venkateswara Reddy [34] wrote "A Study in Employing Rough Set Based Approach for Clustering on Categorical Time-Evolving Data" it defines that the proportionate increase in the size of the data with increase in space implies that clustering a very large data set becomes difficult and is a time consuming process. After sampling, allocating unlabeled data point into proper cluster is difficult in the categorical domain and in real situations data changes over time. In this paper, we propose and implement a parallel k-means clustering algorithm based on Map Reduce, which is a simple yet powerful parallel programming technique which can scale well and efficiently process large datasets.

Junbo Zhang [35] wrote "Parallel Rough Set Based Knowledge Acquisition Using Map Reduce from Big Data" it defines comprehensive experimental evaluation on large data sets shows that the proposed parallel methods can effectively process big data and future work will focus on unstructured data processing by using rough set theory and Map Reduce techniques.
Y. Swapna [36] wrote "A Frame Work for Clustering Time Evolving Data Using Sliding Window Technique" it defines a new Drifting Concept Detection algorithm that finds the number of outliers that cannot be assigned to any of the cluster. The objective of this algorithm is to compare the distribution of clusters and outliers between the last clustering result and the

current temporal clustering result. The experimental evaluation shows that performing DCD is faster than doing clustering once on the entire data set and DCD can provide high-quality clustering results with correctly detected drifting concepts.

Prajesh P Anchalia [37] wrote "MapReduce Design of K- Means Clustering Algorithm" it defines Cluster is a collection of data members having similar characteristics. Data mining is the process of applying some operations like clustering on raw dataset. The raw data is collected from the different sources is unstructured. Hence, there is a continuous emerge of changing the unstructured data.

Weizhong Zhao [38] wrote "Parallel K-Means Clustering Based on MapReduce" it defines Data clustering has gained significant attention in many applications, such as data mining, document retrieval, image segmentation and pattern classification. The expanding volumes of information evolving by the progress of technology, makes clustering of very huge scale of data a challenging task. In order to deal with that problem, many scientists try to design efficient parallel clustering algorithms.

K. R. Madhavi [39] wrote "Data Labelling Method for Clustering Time-Evolving Categorical Attributes" it defines Rough Set based Data Labelling algorithm, RSDL allocates each unlabeled data item into suitable clusters and identifies occurrence of concept-drift.

Ashish A. Golghate, Shailendra W. Shende [40] wrote "Parallel K-Means Clustering Based on Hadoop and Hama" it defines the drawbacks of Map-Reduce are shortening trust worthiness and fault tolerance, MR jobs does not preserve data in memory. In the distributed file system MR jobs dumps does not read next MR Job. The Parallelism does not suffer this drawbacks, it is alternative to the MR model. Clustering techniques is used for data analysis across all disciplines. K-means is clustering algorithm its simplicity and speed to run on large data set. The performance of K-means is incompetent when large data set will be used. Map-Reduce and Parallelism solve the problem of inefficiency when large data set used.

Rajesh [41] wrote "Implementation and Analysis of Map-Reduce K-Means Clustering Algorithm in Hadoop" it defines that the proposed algorithm can scale well and efficiently process large datasets on commodity hardware and for future work, we can try to compare performance of k-means using map reduce and apache spark.

In this paper, discusses the implementation of the map-reduce K-Means Clustering Algorithm over a distributed environment using Apache Hadoop. K-Means Clustering is one such technique used to provide a structure to unstructured data so that valuable information can be extracted. The design of the k-means algorithm was implemented in the later part of this paper based on a small scale implementation of the K-Means Clustering Algorithm on an experimental setup to assist as a monitor for practical implementations.

III. METHODOLOGY

In this section we will review Cluster Analysis, Rough Sets, K-Means Clustering, Map Reduce paradigm, and K-Means Clustering using Map Reduce.

A. Cluster Analysis

Clustering basically deals with grouping of objects such that each group consists of similar or related objects. The main idea behind clustering is to maximize the intra-cluster similarities and minimize the inter cluster similarities.

The data set may have objects with more than attributes. The classification is done by selecting the appropriate attribute and relate to a carefully selected reference and this is solely dependent on the field that concerns the user. Classification therefore plays a more definitive role in establishing a relation among the various items in semi or unstructured data set.

Cluster analysis is a broad subject and hence there are abundant clustering algorithms available to group data sets. Very common methods of clustering involve computing distance, density and interval or a particular statistical distribution. Depending on the requirements and data sets we apply the appropriate clustering algorithm to extract data from them. Clustering has a broad spectrum and the methods of clustering on the basis of their implementation can be grouped into

Connectivity Technique
Example: Hierarchical Clustering
Centroid Technique
Example: K-Means Clustering
Distribution Technique
Example: Expectation Maximization
Density Technique
Example: DBSCAN
Subspace Technique Example: Co-Clustering
Advantages of Data Clustering
i) Provides a quick and meaningful overview of data.
ii) Improves efficiency of data mining by

combining data with similar characteristics so that a generalization can be derived for each cluster and hence processing is done batch wise rather than individually.

iii) Gives a good understanding of the unusual similarities that may occur once the clustering is complete.

iv) Provides a really good base for nearest neighboring and ordination of deeper relations.

B. Rough Sets

Given a pair K = (U, R), where U is a non-empty finite set called the universe, and R □ U x U is an equivalence on U.

The pair K = (U, R) is called an approximation space. The equivalence relation R partitions the set U into several disjoint subsets. This partition of the universe forms a quotient set

induced by R, denoted by U/R. If two elements, x, y □ U, are indistinguishable under R, we say x and y belong to the same

equivalence class. The equivalence class including x is denoted by □*x*□R.

An approximation space K = (U, R) is characterized by an information system S = (U, A, V, f), where

U is a non-empty finite set of objects, called a universe.

A is a non-empty finite set of attributes.

V equals to ∪ V_a, V_a is a domain of the attribute a(a□ A).

f is an information function U x A → V, such that f(x, a)□ V_a for every x□ U, a□ A.

Specifically, S = (U, A, V, f) is called a decision table if

= C □ D, where C is a set of condition attributes and D is a decision, C □ D = Ø.

A rough set definition for a given class, C is approximated by two sets – a lower approximation of C and an upper approximation of C. The lower approximation of C consists of all the data tuples that are based on the knowledge of the attributes, are certain to belong to C without ambiguity. The upper approximation of C consists of all the tuples that are based on the knowledge of the attributes, cannot be described as not belonging to C. Where each rectangular region represents an equivalence class. Decision rules can be generated for each class. Typically, a decision table is used to represent the rules.

C. K-Means Clustering

K-Means Clustering is a method used to classify semi structured or unstructured data sets. This is one of the most commonly and effective methods to classify data because of its simplicity and ability to handle voluminous data sets.

It accepts the number of clusters and the initial set of centroids as parameters. The distance of each item in the data set is calculated with each of the centroids of the respective cluster. The item is then assigned to the cluster with which the distance of the item is the least. The centroid of the cluster to which the item was assigned is recalculated.

One of the most important and commonly used methods for grouping the items of a data set using K-Means Clustering is calculating the distance of the point from the chosen mean.

This distance is usually the Euclidean Distance though there are other such distance calculating techniques in existence. This is the most common metric for comparison of points.

Suppose there the two points are defined as X = (p1(x),p2(x),p3(x)……..) and Y = (q1(y),q2(y),q3(y)…..). The distance is calculated by the formula given by

D(X, Y) =

$$\sqrt{(p_1(X)-p_2(Y))_2+(p_2(X)-p_2(Y))_2+....)}$$

$$= \sqrt{(\sum^{n}(p_j(X) - p_j(Y))2)}$$

The next important parameter is the cluster centroid. The point whose coordinates corresponds to the mean of the coordinates of all the points in the cluster.

The data set may or better said will have certain items that may not be related to any cluster and hence cannot be classified under them, such points are referred to as outliers and more often than not correspond to the extremes of the data set depending on whether their values or extremely high or low.

The main objective of the algorithm is to obtain a minimal squared difference between the centroid of the cluster and the item in the dataset.

$$Xi(j) \square Cj\ 2 \Big|$$

Where X_i is the value of the item and C_j is the value of the centroid of the cluster.

The Algorithm is discussed below:

Algorithm: k-means. The k-means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

Input:

K: the number of clusters,
D: a data set containing n objects.
Output: A set of k clusters.
Method:

- Arbitrarily choose k objects from D as the initial cluster centers;
- **Repeat**
- (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;

Update the cluster means, that is, calculate the mean value of the objects for each cluster;

- **Until** no change;

Pseudo Code:

Pseudo code for k-means can be given as follows:

```
Input: E = { e1,e2,e3,………,en } set of entities to be
    clustered
K (number of clusters) MaxItems (limit of iterations)
Output: C = {c1,c2,c3,……,ck} set of cluster centroids
    L = {l(e) | e = 1,2,3,…………,n} set of cluster
    labels of E for each ci ε C do
Ci <- ej e E do (e.g. random selection) End
For each ei e Edo

L(ei)<- argminDistance(ei,Cj)j e {1………k} End

Changed <- false; Iter<- 0;
Repeat
For each Ci e C do
UpdateCluster(Ci);
End
For each ei e Edo
minDist<- argminDistance(ei,Cj) j e {1,………k}
    ifminDist != l(ei) then
l(ei) <- minDist; changed<- true; end

end iter++; until

changed = true &iter<= maxIters;
```

D. *MapReduce Paradigm*

Map Reduce is a programming paradigm used for computation of large datasets. A standard Map Reduce process computes terabytes or even peta bytes of data on interconnected systems forming a cluster of nodes. Map Reduce implementation splits the huge data into chunks that are independently fed to the nodes so the number and size of each chunk of data is dependent on the number of nodes connected to the network. The programmer designs a Map function that uses a (key, value) pair for computation. The Map function results in the creation of another set of data in form of (key, value) pair which is known as the intermediate data set. The programmer also designs a Reduce function that combines value elements of the (key, value) paired intermediate data set having the same intermediate key [42].

Map and Reduce steps are separate and distinct and complete freedom is given to the programmer to design them. Each of the Map and Reduce steps are performed in parallel on pairs of (key, value) data members. Thereby the program is segmented into two distinct and well defined stages namely Map and Reduce. The Map stage involves execution of a function on a given data set in the form of (key, value) and generates the intermediate data set. The generated intermediate data set is then organized for the implementation of the Reduce operation. Data transfer takes place between the Map and Reduce functions. The Reduce function compiles all the data sets bearing the particular key and this process is repeated for all the various key values. The final output produced by the Reduce call is also a dataset of (key, value) pairs. An important thing to note is that the execution of the Reduce function is possible only after the Mapping process is complete.

Each Map Reduce Framework has a solo Job Tracker and multiple task trackers. Each node connected to the network has the right to behave as a slave Task Tracker. The issues like division of data to various nodes, task scheduling, node failures, task failure management, communication of nodes, monitoring the task progress is all taken care by the master node. The data used as input and output data is stored in the HDFS file-system.

E. *K-Means Clustering using MapReduce*

The first step in designing the Map Reduce routines for K- means is to define and handle the input and output of the implementation. The input is given as a <key, value> pair, where 'key' is the cluster center and 'value' is the serializable implementation of vector in the data set.

The prerequisite to implement the Map and Reduce routines is to have two files one that houses the clusters with their centroids and the other that houses the vectors to be clustered.

Once the set of initial set of clusters and chosen centroids is defined and the data vectors that are to be clustered properly organized in two files then the clustering of data using K- Means clustering technique can be accomplished by following the algorithm to design the Map and Reduce routines for K- Means Clustering.

The initial set of centers is stored in the input directory of HDFS prior to Map routine call and they form the 'key' field in the <key, value> pair. The instructions required to compute the distance between the given data set and cluster center fed as a <key, value> pair is coded in the Mapper routine. The Mapper is structured in such a way that it computes the distance between the vector value and each of the cluster centers mentioned in the cluster set and simultaneously keeping track of the cluster to which the given vector is closest. Once the computation of distances is complete the vector should be assigned to the nearest cluster.

Once Mapper is invoked the given vector is assigned to the cluster that it is closest related to. After the assignment is done the centroid of that particular cluster is recalculated. The recalculation is done by the Reduce routine and also it restructures the cluster to prevent creations of clusters with extreme sizes i.e. cluster having too less data vectors or a cluster having too many data vectors. Finally, once the centroid of the given cluster is updated, the new set of vectors and clusters is re-written to the disk and is ready for the next iteration.

After understanding of what the input, output and functionality of the Map and Reduce routines we design the Map and Reduce classes by following the algorithm discussed below.

Algorithm 1: Mapper for K-Means Clustering

```
procedure KMEANMAPDESIGN LOAD Cluster file
fp = Mapclusterfile Create two list listnew = listold
CALL read (Mapclusterfile) newfp = MapCluster()
dv = 0
Assign correct centroid read(dv) calculatecenteroid
dv =  minCenter() CALL KmeansReduce()
end procedure = 0
```

Algorithm 2: Reducer for K-Means Clustering

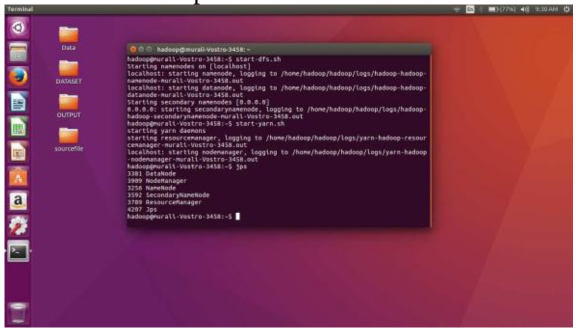
```
procedureKMEANREDUCEDESIGN NEW
    ListofClusters
COMBINE resultant clusters from MAP CLASS
    ifcluster size too high or too low thenRESIZE the
    cluster
CMax = findMaxSize(ListofClusters)
Cmin = findMinSize(ListofClusters)
ifCmax> 1/20 totalSize then Resize(cluster) WRITE
    cluster FILE to output DIRECTORY.
```

Algorithm 3: Implementing KMeans Function

```
procedureKMEANS FUNCTION
ifInitial Iteration thenLOAD cluster file from
    DIRECTORY
elseREAD cluster file from previous iteration Create
    new JOB
SET MAPPER  to map class defined
SET REDUCER to reduce class define paths for output
    directory
SUBMIT JOB
```

IV. EXPERIMENTAL RESULTS

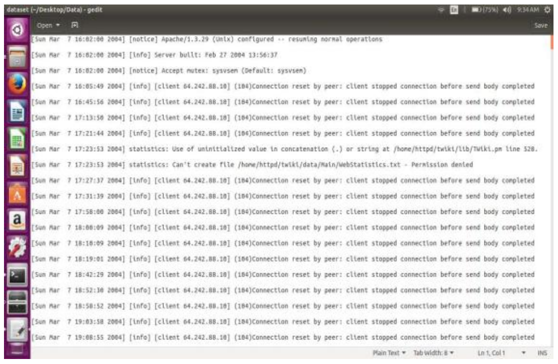
In this section, we only evaluate the performance of the proposed parallel methods. All experiments run on the Apache Hadoop platform. Hadoop version 2.0 and Java 1.8 are used as Map Reduce system. Figure 1 shows all services which are started in a hadoop.



```
hadoop@murali-Vostro-3458: ~
hadoop@murali-Vostro-3458:~$ start-dfs.sh
Starting namenodes on [localhost]
localhost: starting namenode, logging to /home/hadoop/hadoop/logs/hadoop-hadoop-
namenode-murali-Vostro-3458.out
localhost: starting datanode, logging to /home/hadoop/hadoop/logs/hadoop-hadoop-
datanode-murali-Vostro-3458.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/hadoop/hadoop/logs/hadoop-
hadoop-secondarynamenode-murali-Vostro-3458.out
hadoop@murali-Vostro-3458:~$ start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /home/hadoop/hadoop/logs/yarn-hadoop-resour
cemanager-murali-Vostro-3458.out
localhost: starting nodemanager, logging to /home/hadoop/hadoop/logs/yarn-hadoop
-nodemanager-murali-Vostro-3458.out
hadoop@murali-Vostro-3458:~$ jps
3381 DataNode
3909 NodeManager
3256 NameNode
3592 SecondaryNameNode
3789 ResourceManager
4207 Jps
hadoop@murali-Vostro-3458:~$
```

Figure 1. Hadoop all Services Started

We utilize the large data set, as shown in the figure 2 which consists of approximately one million records. Each record consists of 1 decision attribute and 35 condition attributes, where 5 are categorical and 30 are numeric. Since out method can only deal with categorical attributes, we discretize the 30 numeric attributes firstly. In addition, three data sets are generated by means of Hadoop distributed file system. The data set, statistics are split into 64, 128 and 256 blocks, respectively when we upload these data to HDFS.



```
[Sun Mar  7 16:02:00 2004] [notice] Apache/1.3.29 (Unix) configured -- resuming normal operations
[Sun Mar  7 16:02:00 2004] [info] Server built: Feb 27 2004 13:56:37
[Sun Mar  7 16:02:00 2004] [notice] Accept mutex: sysvsem (Default: sysvsem)
[Sun Mar  7 16:05:49 2004] [info] [client 64.242.88.10] (104)Connection reset by peer: client stopped connection before send body completed
[Sun Mar  7 16:45:56 2004] [info] [client 64.242.88.10] (104)Connection reset by peer: client stopped connection before send body completed
[Sun Mar  7 17:13:50 2004] [info] [client 64.242.88.10] (104)Connection reset by peer: client stopped connection before send body completed
[Sun Mar  7 17:21:44 2004] [info] [client 64.242.88.10] (104)Connection reset by peer: client stopped connection before send body completed
[Sun Mar  7 17:23:53 2004] statistics: Use of uninitialized value in concatenation (.) or string at /home/httpd/twiki/lib/TWiki.pm line 528.
[Sun Mar  7 17:23:53 2004] statistics: Can't create file /home/httpd/twiki/data/Main/WebStatistics.txt - Permission denied
[Sun Mar  7 17:27:37 2004] [info] [client 64.242.88.10] (104)Connection reset by peer: client stopped connection before send body completed
[Sun Mar  7 17:31:39 2004] [info] [client 64.242.88.10] (104)Connection reset by peer: client stopped connection before send body completed
[Sun Mar  7 17:58:00 2004] [info] [client 64.242.88.10] (104)Connection reset by peer: client stopped connection before send body completed
[Sun Mar  7 18:00:09 2004] [info] [client 64.242.88.10] (104)Connection reset by peer: client stopped connection before send body completed
[Sun Mar  7 18:10:09 2004] [info] [client 64.242.88.10] (104)Connection reset by peer: client stopped connection before send body completed
[Sun Mar  7 18:19:01 2004] [info] [client 64.242.88.10] (104)Connection reset by peer: client stopped connection before send body completed
[Sun Mar  7 18:42:29 2004] [info] [client 64.242.88.10] (104)Connection reset by peer: client stopped connection before send body completed
[Sun Mar  7 18:52:30 2004] [info] [client 64.242.88.10] (104)Connection reset by peer: client stopped connection before send body completed
[Sun Mar  7 18:58:52 2004] [info] [client 64.242.88.10] (104)Connection reset by peer: client stopped connection before send body completed
[Sun Mar  7 19:03:58 2004] [info] [client 64.242.88.10] (104)Connection reset by peer: client stopped connection before send body completed
[Sun Mar  7 19:08:55 2004] [info] [client 64.242.88.10] (104)Connection reset by peer: client stopped connection before send body completed
```

Figure 2. Data Set

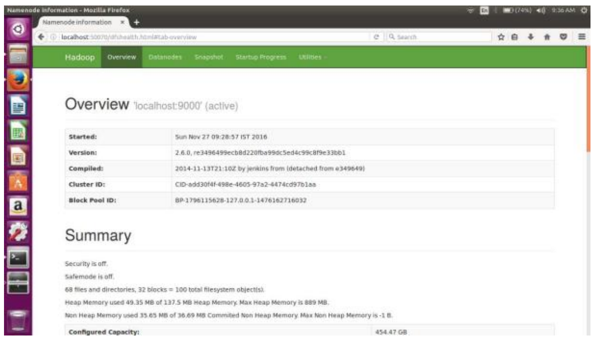
Our experiments are conducted on large clusters of compute machines. The operating system in these machines was Linux Ubuntu 16.4 LTS. We executed the aforementioned three kinds of knowledge acquisition methods on Hadoop Map Reduce system. Further, we examine the purity ratio of the proposed parallel methods.

To measure the purity ratio, we keep the data set constant and increase the number of cores in the system. Purity ratio given by the larger system is defined by the following formula:

$$\text{Purity ratio } (\alpha) = \frac{Tl}{Tp}$$

Where p is the number of cores, Tl is the execution timeon single core, Tp is the execution time on p cores.

Figure 3 shows the name node information on the Hadoop system.



Overview 'localhost:9000' (active)

Started:	Sun Nov 27 09:28:57 IST 2016
Version:	2.6.0, re3496499ecb8d220fba99dc5ed4c99c8f9e33bb1
Compiled:	2014-11-13T21:10Z by jenkins from (detached from e349649)
Cluster ID:	CID-add30f4f-498e-4605-97a2-4474cd97b1aa
Block Pool ID:	BP-1796115628-127.0.0.1-1476162716032

Summary

Security is off.

Safemode is off.

68 files and directories, 32 blocks = 100 total filesystem object(s).

Heap Memory used 49.35 MB of 137.5 MB Heap Memory. Max Heap Memory is 889 MB.

Non Heap Memory used 35.65 MB of 36.69 MB Commited Non Heap Memory. Max Non Heap Memory is -1 B.

Configured Capacity:	454.47 GB

Figure 3. Name Node Information

We perform the purity ratio evaluation on data sets with quite different sizes and structures. The number of processes varied from 1 to 64. Table 1 and figure 4 show the purity ratio for over all data sets for 2, 4, 8, 16, 32 and 64 processing cores. As the result shows, the proposed parallel method has a very good purity ratio performance. Therefore, the proposed parallel methods can treat big data efficiently.

Table 1: Purity Ratio Between Various K-Means

Data set	Number of Cores					
	2	4	8	16	32	64
K-means	1.98	3.84	7.49	13.61	25.35	27.92
K-means with map reduce	1.97	3.79	7.13	12.69	21.01	23.57
K-means through RST using map reduce	1.96	3.61	5.86	10.00	15.67	20.12

Figure.4 shows purity ratio for map reduce execution process we performed the execution time on data set with quite different sizes and structures. The number of processes varied from 1 to 64. Table 1 and Figure 4 show the purity ratios for over all data sets for 2, 4, 8, 16, 32 and 64 processing cores. As the result shows, the proposed parallel method has a very good speedup performance. K-means through RST using map reduce has a lower speedup curve, because the execution time of it is too small. As the size of the data set increases, the purity ratio performs better. Therefore, the proposed parallel methods can treat big data efficiently.

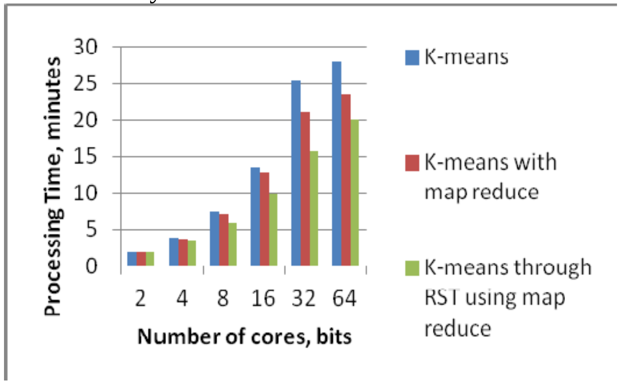
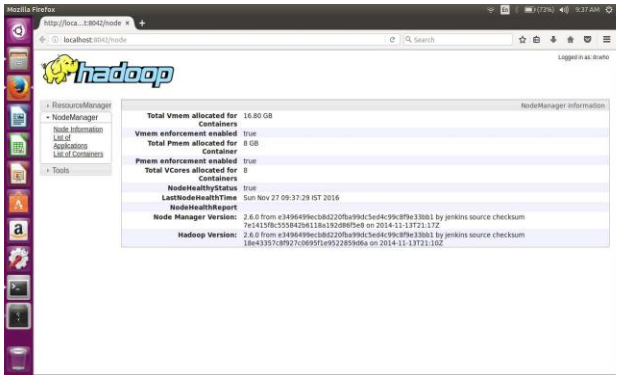


Figure 4. Purity Ratio For Map Reduce Execution Processes The Following Figure 5 Shows The Hadoop Node Manager .



Total Vmem allocated for Containers	16.80 GB
Vmem enforcement enabled	true
Total Pmem allocated for Container	8 GB
Pmem enforcement enabled	true
Total VCores allocated for Containers	8
NodeHealthyStatus	true
LastNodeHealthTime	Sun Nov 27 09:37:29 IST 2016
NodeHealthReport	
Node Manager Version:	2.6.0 from e3496499ecb8d220fba99dc5ed4c99c8f9e33bb1 by jenkins source checksum 7e1415f8c555842b6118a192d86f5e8 on 2014-11-13T21:17Z
Hadoop Version:	2.6.0 from e3496499ecb8d220fba99dc5ed4c99c8f9e33bb1 by jenkins source checksum 18e43357c8f927c0695f1e9522859d6a on 2014-11-13T21:10Z

Figure 5. Node Manager

V. CONCLUSIONS

Data mining from big data has been a new challenge in recent years. Traditional rough sets based methods for knowledge discovery fail to deal with the enlarging data in applications. Comprehensive experimental results on the data sets demonstrated that the proposed methods could effectively process large data sets in data mining. Our future research work will focus on unstructured data processing by using rough set theory and Map Reduce techniques.

REFERENCES

[1] J.Han, M.Kamber, Data Mining: Concepts and Techniques, 2nd Edition, Morgan Kaufman, San Francisco, 2006.

[2] A. Malcom Marshall 1, Dr. S.Gunasekaran PG scholar(M.E), 2 prof & Head,1, "A Survey on Job and Task Scheduling in Big Data", Dept. of Computer Science & Engineering Coimbatore Institute of Engineering & Technology, Coimbatore, India.

[3] Anil K.Jain & richard C. Dubes. "Algorithms for clustering Data", Prentice-Hall International, 1988.

[4] Jain A K MN Murthy and PJ Flyn, "Data Clustering: A Review", ACM Computing Survey, 1999.

[5] Kaufman L, P.Rousseuw, "Finding Groups in Data – An Introduction to Cluster Analysis", Wiley Series in probability and Math. Sciences, 1990.

[6] Bradley, P.S.Usama Fayyad, and cory Reina, "Scaling Clustering algorithms to Large data bases", Fourth International Conference on Knowledge Discovery and Data Mining, 1998.

[7] Joydeep Ghosh. Scalable Clustering methods for data mining. In Nong Ye, editor, "Hand book of Data Mining", chapter 10, pp.247-277. Lawrence Ealbaum Assoc, 2003.

[8] Sudipto Guha, Adam Meyerson, Nina Mishra,Rajeev Motwani, and Liasen O"Callaghan, "Clustering data streams Theory and practice", IEEE Transactions on Knowledge and Data Engineering, pp.515-528, 2003.

[9] Gibson, D.,Kleinberg J.M and Raghavan, P "Clustering Categorical Data An Approach Based on Dynamical Systems", VLDB pp.3-4, pp.222-236, 2000.

[10] Michael R.Anderberg, "Cluster Analysis for applications", Academic press, 1973.

[11] Chen H.L, M.-S.Chen and S-U Chen Lin "Frame work for clustering Concept – Drifting Categorical data", IEEE Transaction Knowledge and Data Engineering v21 no5, 2009.

[12] Klinkenberg. R, Using Labeled and unlabeled data to learn drifting Concepts", IJCAI-01 workshop on Learning from Temporal and Spatial Data, pp. 16-24, 2001.

[13] Aggarwal, C, Han, J., Wang, J. and Yu P, "A Framework for Clustering Evolving Data Streams", Very Large Data Bases(VLDB), 2003.

[14] Aggarwal, C, Wolf, J.L., Yu, P.S. Procopiuc, C. and Park, J.S. "Fast-Algorithms for projected Clustering.", ACM SIGMOD'99, pp.61-72, 1999.

[15] J.W.Grzymala - Busse and W.Ziarko. Data mining and rough set theory. Commun. ACM, 43(4):108-109, Apr 2000.

[16] Z.Pawlak, J. Grzymala – Busse, R. Slowinski, and W. Ziarko. Rough Sets, Commun. ACM, 38(11):88-95, Nov. 1995.

[17] W. Ziarko. Discovery through rough Set theory. Commun. ACM, 42(11):54-57, Nov. 1999.

[18] K.Kaneiwa. A Rough Set approach to mining Connections from information Systems. In proceedings of the 2010 ACM Symposium on Applied Computing, SAC'10, pages 990-996, New York, NY, USA, 2010. ACM.

[19] S.Tsumoto. Automated extraction of medical expert system rules from Clinical databases based on rough Set theory. Information Sciences, 112(1-4):67-84, Dec. 1998.

[20] Y. Leung, W – Z, Wu , and W. –X. Zhang. Knowledge acquisition in incomplete information systems: A rough set approach. European Journal of Operational Research, 168(1):164-180, Jan.2006.

[21] Q. Hu, W. Pedrycz, D. Yu, and J. Lang. Selecting discrete and continuous features based on neighborhood decision error minimization. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 40(1): 137-150, feb. 2010.

[22] Q. Hu, Z. Xie, and D. Yu. Hybrid attribute reduction based on a novel fuzzy-rough model and information granulation. Pattern Recognition, 40(12): 3509-3521, Dec. 2007.

[23] Y. Qian, J. Liang, W. Pedrycz, and C. Dang. Positive approximation: An accelerator for attribute reduction in rough set theory. Artificial Intelligence, 174(9-10):597- 618, June 2010.

[24] Hadoop: Open source implementation of MapReduce, <http: //hadoop.apache.org/mapreduce/>.

[25] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. In Proceedings of the 6th conference on Symposium on Operating Systems Design & Implementation – Volume 6, OSDI'04, pages 10-10, Berkeley, CA, USA, 2004. USENIX Association.

[26] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. Commun. ACM, 51(1): 107-113, Jan. 2008.

[27] Apache Hadoop. http://hadoop.apache.org/

[28] J. Venner, Pro Hadoop. Apress, June 22, 2009.

[29] T. White, Hadoop: The Definitive Guide. O'Reilly Media, Yahoo! Press, June 5, 2009.

[30] S. Ghemawat, H. Gobioff, S. Leung. "The Google file system," In Proc. Of ACM Symposium on Operating Systems Principles, Lake George, NY, Oct 2003, pp 29-43.

[31] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, R. Murthy, "Hive- A Warehousing Solution Over a Map-Reduce Framework." In Proc. Of Very Large Data Bases, vol.2 no.2, August 2009, pp. 1626-1629.

[32] F.P. Junqueira, B. C. Reed. "The life and times of a zookeeper." In Proc. Of the 28th ACM Symposium on Principles of Distributed Computing, Calgary, AB, Canada, August 10-12, 2009.

[33] A. Gates, O. Natkovich, S. Chopra, P. Kamath, S. Narayanam, C.Olston, B. Reed, S. Srinivasan, U. Srivastava. "Building a High-Level Dataflow System on top of MapReduce: The Pig Experience," In Proc. of Very Large Data Bases, vol 2 no. 2, 2009, pp. 1414-1425.

[34] H. Venkateshwara Reddy, S. Viswanadha Raju,A Study in

Employing Rough Set Based Approach for Clustering on Categorical Time-Evolving Data, IOSR Journal of Computer Engineering (IOSRJCE) ISSN: 2278-0661 Volume 3. Issue 5 (July – Aug. 2012), PP 44-51 www.iosrjournals.org.

[35] Junbo Zhang, Tianrui Li, Yi Pan, "Parallel Rough Set Based Knowledge Acquisition Using MapReduce from Big Data", BigMine 12, August 12, 2012 Beijing, China ACM 978-1-4503-1547-0/12/08.

[36] Y. Swapna, S. Ravi Sankar, "A Framework for clustering Time Evolving Data Using Sliding Window Technique", Vol. 3, Issue 3, Oct-Dec(2012), pp. 377-383, IJCET.

[37] Prajesh P Anchalia, Anjan K Koundinya, Srinath N K, "MapReduce Design of K-Means Clustering Algorithm", IEEE, 2013.

[38] Weizhong Zhao, Huifang Ma, and Qing He, "Parallel K-Means Clustering Based on MapReduce" The Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences Graduate University of Chinese Academy of Sciences.

[39] K.R. Madhavi, Dr. A. V. Babu & Dr. A.A. Rao, "Data Labelling Method for Clustering Time Evolving Categorical Attributes, "Global Journal of Computer Science and Technology: C Software and Data Engineering, vol. 15, issue 5 version 1.0 , 2015.

[40] Ashish A. Golghate, Shailendra W. Shende, "Parallel K-Means Clustering Based on Hadoop and Hama", Department of Information Technology, Yeshwantrao Chavan College of Engineering, Nagpur, Maharashtra, India.

[41] K.V.N.Rajesh, Ravuri Daniel, P.Prudhvi kiran, T. Madhuri Priyadarshani, "Implementation and Analysis of Map-Reduce K-Means Clustering Algorithm in Hadoop", Research Article, Vol 6, Issue No.5, IJESC.

[42] Anjan K Koundinya, Srinath N K, A K Sharma, Kiran Kumar, Madhu M N and Kiran U Shanbagh, "Map/Reduce Design and Implementation of Apriori Algorithm for handling Voluminous Data-Sets, Advanced Computing: An International Journal (ACIJ), Vol.3, No.6, Nov. 2012.