Analysis of Clustering Techniques on Big Data

Prachi P. Surwade
ME Computer (Student) Department of Computer Engineering
K. K. Wagh Institute of Engineering Education & Research Nashik, S.P.P.U

S.S.Banait
Assistant Professor
Department of Computer Engineering
K. K. Wagh Institute of Engineering Education & Research Nashik, S.P.P.U

ABSTRACT

In this In today's era data generated by scientific applications and corporate environment has grown rapidly not only in size but also in variety. This data collected is of huge amount and there is a difficulty in collecting and analyzing such big data. Data mining is the technique in which useful information and hidden relationship among data is extracted, but the traditional data mining approaches cannot be directly used for big data due to their inherent complexity. Cluster analysis is used to classify similar objects under same group. It is one of the most important data mining methods. However, it fails to perform well for big data due to huge time complexity. For such scenarios parallelization is a better approach. Map reduce is a popular programming model which enables parallel processing in a distributed environment. In this paper to propose system for analyze the performance of two clustering techniques on big dataset. The goal of this paper is to find better clustering technique between K-Medoid and BIRCH clustering by applying on real life large dataset.

Keywords

Big data, Clustering, Hadoop, Map-Rreduce, K-Medoid(KM), BIRCH.

1. INTRODUCTION

Data is increasing overwhelmingly during the past decades, and "big data" is becoming more and more popular. Big data doesn't only refer to massive data, but also a series of techniques which turn a flood of data into valuable information.

Big data is defined as "datasets whose size is beyond the ability of typical database software tools to capture, store, manage, and analyze" by McKinsey Global Institute in 2011[2]. John R. define big data as "Any amount of data that's too big to be handled by one computer"

Clustering techniques is commonly used technique and it is mainly used for analysis of data. Clustering process combining group of items and grouping can be done by creating the most similar item of one group (cluster) and most non similar items in the other group. This process is similar to classification. Clustering is an unsupervised way of learning or it is an unsupervised classification of data items, (feature vectors (FV), patterns, or observations) into clusters (groups).

Clustering is found the clusters (or grouping) of data in a set of unlabelled data. In artificial intelligence (AI) data clustering technique is major assignment. Human being can easily identify the cluster in low dimension with less number of records, but in case of computer it is extremely hard to instruct the computer to find such a relationship. When the dimensions of the data increases human has difficulties in finding the interesting patterns of the data. To find an interesting pattern in exploratory data analysis or to extract the information from the data is the objective of exploratory data analysis.

At the starting of new era information has grownup speedily in sizes as well as in varieties also. This data collected is of huge amount and it has some degree of difficulty when it comes to collecting and analyzing data as big data. Data mining method used to extracts useful information and unseen correlation between data. Huge amount of data stored in datasets are quickly growing due to rapid technological growth (progress). Therefore some applications require the storage space and repossession of complex multimedia items that may be represented by high dimensional FV. Very difficult task is to obtain more valuable information concealed in some databases therefore to remove these difficulty Clustering analysis used. It is best techniques which are applied on large data sets for analyzing purpose. Moreover, the (input) parameters choice like number nearest neighbors, Kn amount of clusters and some other factor in algorithms create the clustering is challenging process. One of the very effective ways of dealing with these data is to categorize or assemble that data into a set of classes. Now a day clustering methods emerge as another influential meta-learning tool for correctly analyze the big volume of data created by some new applications.

In this work we are using two clustering algorithms as K-Medoid clustering algorithm from Partitioning Clustering category and BIRCH clustering method form Hierarchical Clustering category.

Big data analytics is often associated with cloud computing because the analysis of large data sets in real-time requires a platform like Hadoop to store large data sets across a distributed cluster and Map-Reduce to coordinate, combine and process data from multiple sources[3].

In this paper to organized some sections as: section2 gives the analysis of related work of clustering approaches and Map-Reduce related survey. Section3 gives the methodology. Section4 describes the dataset details. Section5 explains the experimental setup. Section6 explains the system architecture. Section7 shows experimental result and last sections8 conclude the paper.

2. LITERATURE SURVEY

All Data Clustering is the most popular aspect to derive the classes or cluster, desired groups of pattern and concept. It is very difficult to find the clustering or grouping of the data and finding interesting patterns about which nothing is known already. This also leads to the general problem of how to find out the knowledge of the data for mining and exploration. Although the literature on clustering is huge there has been relatively little attention paid to the problem of finding different clustering techniques. There are five categories of clustering as given below.

- Partition Based Clustering (PBC)
- Hierarchical Based Clustering (HBC)
- Density-Based Clustering (DBC)
- Grid-Based Clustering (GBC)
- Model-Based Clustering (MBC)

T. Zhang, R. Ramakrishna [5], suggested a BIRCH is an HBC algorithm. This algorithm is categorized agglomerative approach. For Big databases this algorithm is useful. It is especially suitable for very large databases. This algorithm has been planned for minimizing number of I/O operations. BIRCH algorithm is dynamically and incrementally clusters the incoming data items (objects) and then they try to create a greatest quality of clusters with in some limited assets (as time constraints and available memory). In this algorithm memory data structure uses called as clustering feature tree (CF-tree).

First clustering method is BIRCH. It is firstly developed for handle noise. Disadvantage of this method is it only handles numeric data.

S. Guha , R. Rastogiand K. Shim [6], present CUREs agglomerative HBC method. In this algorithm random data samples are 1st partition. After that every created partition is then partly cluster. Next final pass for creation of final clusters to preclustred an data from each partitions by eliminating an outliers. In each step of this algorithm, closest pair of each clusters medoid (centroids) of two clusters is combining (merged). This method only combines representative objects (centroid) and uses single linkage method for selecting more than one centroid from each cluster. Disadvantage of this method is combined inter-connectivity of points in two different clusters information is not considered. To overcome this problem Chameleon algorithm [6] is implemented.

E.-H. Han, V. Kumar and G. Karypis [7], suggested CHAMELEON as HBC agglomerative method which can discover dynamic modeling. This algorithm work on two phases: 1st divides (partition) data items into a sub clusters then using graph partitioning method continually merging sub-clusters and then to get a final clusters. This algorithm is mainly used to find clusters densities their diverse shapes and sizes in 2D (two dimensional) space. To obtain the arbitrary shape clusters and arbitrary densities of clusters Chameleon method uses an dynamic model. The algorithm is applicable for the applications whose data volume is large. Disadvantage of CHAMELEON only is that it is well-known for low dimensional data space.

K. Shim, S. Guha and R. Rastogi [8], discussed ROCK HBC technique. This agglomerative HBC technique uses a idea of links [9]. It is suitable for managing big volume data sets. In ROCK technique similarity of clustering depends on the points of diverse clusters which are nearest in ordinary can measures clusters similarity. This algorithm is employees a link not a distance for merging clusters. Also they discussed the next version of the ROCK algorithm is QROCK algorithm used for clustering a categorical data. QROCK is quicker than

QROCK[9].

J. Han and R. T. Ng [10], proposed CLARANS algorithm. CLARANS technique supports to randomized search. It is not used any predefined configuration. This algorithm not more affect on incrementing dimensionality of database. This technique does not require distance function. It supports point as well as polygonal objects.

Other conventional clustering technique is the KMeans technique [11]. In this algorithm assigns every data object to the clusters. Those points are neighboring to center called centroid. Centroid is nothing but the arithmetic average of all the data points in the clusters. The simplicity and speed is major advantage of this algorithm. It also runs on large datasets. The disadvantage of this method is that it does not create the same result when it execute at every time.

Although the K-Medoids (KM) algorithm is quite popular among other authors [5], [6], [7], [8], [9], [10] and [11].The KM techniques is correlated to K-Means and mode shift algorithm. The KM method partition the dataset into a partitions or clusters. It reduces the space among cluster center and data points in the clusters. Unlike K-Mean method, KM method chooses data object as medoids (centroids). KM technique provides correct visualization of the data, especially when there are more than two attributes involved in the clustering process [12]. Comparatively it is more robust to noise and outliers than K-Means, because K-Mean reduces average pair wise dissimilarities. KM technique is clusters the dataset of n data points into Kn clusters. Medoids referred as the most centrally located data point in the cluster. There are different compensation of KM technique also for execution it is simple

Hadoop provides an open source framework for cloud computing, as well as a distributed file system. Hadoop supports the Map-Reduce model, which was introduced by Google as a method of solving a class of petascale problems with large clusters of inexpensive machines. The model is based on two distinct steps for an application.

- **Map**: An initial ingestion and transformation step, in which individual input records can be processed in parallel.
- **Reduce**: an aggregation or summarization step, in which all associated records, must be processed together by a single entity [3].

The core concept of Map-Reduce in Hadoop is that input may be split into logical chunks, and each chunk may be initially processed independently, by a map task. The results of these individual processing chunks can be physically partitioned into distinct sets, which are then sorted. Each sorted chunk is passed to a reduce task [4].

3. METHODOLOGY

Cluster analysis clusters a data points. To clusters a data points based on based information obtained in the data objects and the relationships among data objects. The objective is that data points within group (region) correlated to each other and not related to data points in other region. If similarity in one of the region is more and dissimilarity among the other regions is more, it is Better or more distinct the clustering.

3.1 The K-Medoids(KM) Clustering Algorithm

Items with a very huge value considerably distort the data distribution therefore K-Means technique is responsive to noise [11]. KM technique choose a medoid, many times it is centrally placed in clusters, instead of selecting the average mean value of data point in a cluster as a reference point. Therefore, the partitioning technique applies depend on theory of reducing the summation of the dissimilarities among every data points and their related medoids. It creates the basic idea of the KM method. Idea of KM technique is to discover Kn numbers of clusters in given dataset which having n data points then by 1st randomly searching the medoids or a reference points for every cluster. Then each remaining data object is compare with medoids (reference point) and after that clusters a data object with centroid. KM technique does not consider the average mean of data points in each clusters. KM algorithm execution is given as follows:

I/P: Let Kn be the number of clusters And dataset as DD having n objects

O/P: Group of Kn numbers of cluster specified by the user.

Algorithmic steps:

Steps of execution of KM algorithm as given below-

1) Initially arbitrarily choose Kn number clusters as the Md data objects as the medoid.
2) Then combine (associate) every data objects from dataset to the neighboring medoid.
3) Apply for loop to every medoid as Md:
   - i. Again for condition apply on non medoid data object as Od
   - ii. Interchange Md and Od and calculate the average costing of the new computation.
4) Choose the computation having the minimum costing.
5) Reiterate step 2nd up to 4th till you found that in medoids has no change occurs.

3.2 The BIRCH Clustering

Summary of BIRCH clustering algorithm is, it execute mainly four phase. We explain the actual task of each of these phases. Phase1 scan all the data and in given amount of memory initially CF-tree is build. That CF- tree tries to reproduce the clustering related information of dataset within the memory limit, with packed data objects merged as fine sub clusters. Phase2 is optional. In phase 2 from initial CF-tree scan all the leaf entries and then to reconstruct the smaller CF-tree and removing outliers in the process. To cluster every leaf entries semi-global or global methods are used in phase3. Later than completion of phase3, obtain a group of cluster which captures the main distributions pattern in the given available data. Then for correct the inaccuracy in data and for filter the cluster or further use to calculate the total cost in phase4. BIRCH algorithm can work in any given space of memory.

4. DATASET DETAILS

This section gives an overview of the dataset which will be used for experiments. We used categorical attribute dataset .There are two datasets are generated.

i) Small Dataset - 14 MB (10^6 records)

ii) Big Dataset - 400 MB Extra Big Datasets Will Depend on Performance on system. As algorithm is working Dataset contains single dimension floating data in range of 1-100 to define boundaries.

5. EXPERIMENTAL SETUP

5.1 Hardware Requirement

- 16 GB RAM
- 8 Virtual cores
- 100 Mbps network box preferences

5.2 Software Requirement

- CentOS v6.6+
- Java v7+
- Hadoop2.7 in distributed mode
- HDFS
- Map-Reduce

6. SYSTEM ARCHITECTURE

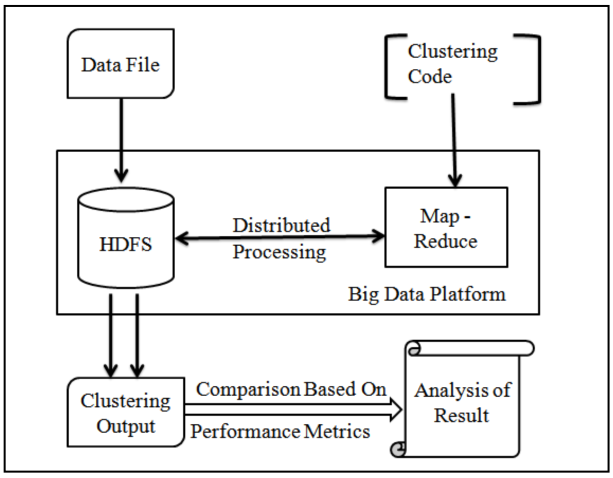


Fig .1 Execution of Clustering Techniques on Big Data Platform

Figure 1 show the system architecture of Analysis of clustering techniques on big data. In this architecture Big Data platform is used and on this platform data is to be processed. This architecture has two inputs as “Data file” and “Clustering code”. To process this input files one of the Big data platform is used. After that to decide which type of data is to be process i.e. to decide Data file and how that data is to process i.e. to decide clustering code. Here we use only one data file and two clustering code as KM and BIRCH clustering code. After processing on clustering code we get an output. Then on the basis of output we have to analyze an clustering technique. System architecture main platform save the dataset as text file or in hive. Firstly to store dataset on HDFS then split dataset using Mapper and Reducer or using Spark technology. Then we do a basic analysis of KM and BIRCH clustering techniques to process on Big data platform. In short we will follow do the following steps we will follow the below steps

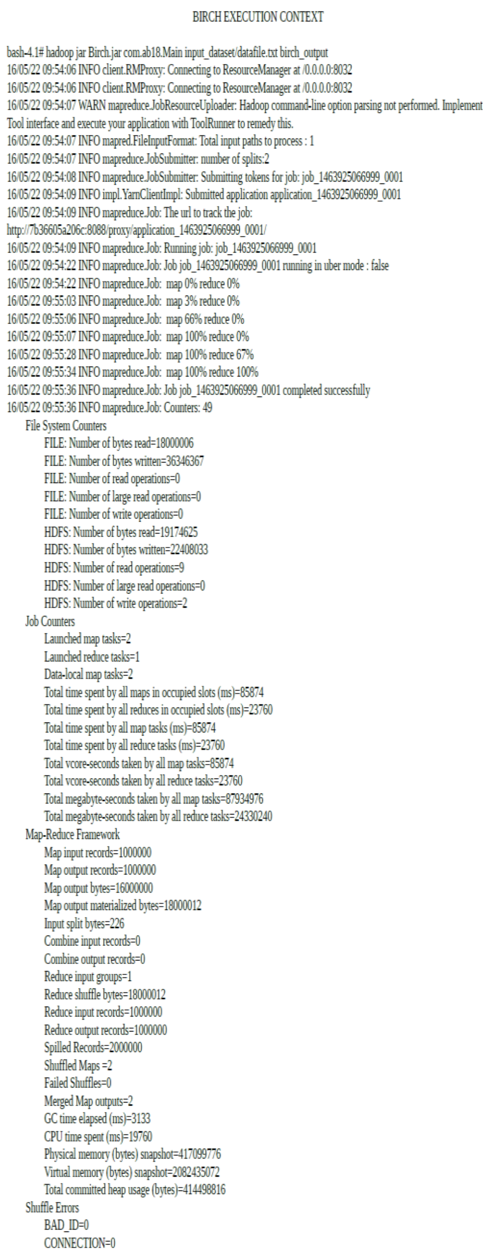
1. Load the given dataset into HDFS directory with name datafile.txt
2. For KM, get first n cluster records from datafile.txt and add to centroid.txt. This would help in setting up the bootstrapping process for KM
3. Load the binaries of KM & BIRCH into Hadoop Cluster
4. Execute it using command like below hadoop jar Birch.jar com.ab18.Main input_dataset/datafile.txt

birch_output hadoop jar KMedoid.jar KMedoid input_dataset kmedoid_output

5. Gather important stats like execution timings, resources used.

7. EXPERIMENTAL RESULT

This section shows the implementation of KM and BIRCH a on Big dataset as shown in following figure 2, 3, 4, 5, 6, 7 and 8.

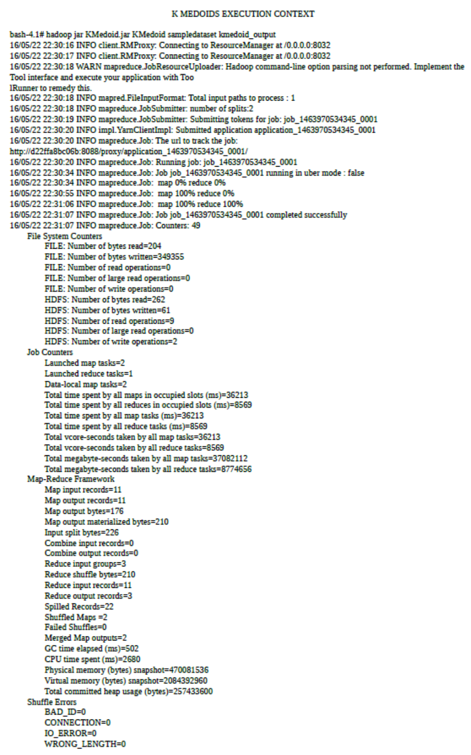


```
BIRCH EXECUTION CONTEXT

bash-4.1# hadoop jar Birch.jar com.ab18.Main input_dataset/datafile.txt birch_output
16/05/22 09:54:06 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
16/05/22 09:54:06 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
16/05/22 09:54:07 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement
Tool interface and execute your application with ToolRunner to remedy this.
16/05/22 09:54:07 INFO mapred.FileInputFormat: Total input paths to process : 1
16/05/22 09:54:07 INFO mapreduce.JobSubmitter: number of splits:2
16/05/22 09:54:08 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1463925066999_0001
16/05/22 09:54:09 INFO impl.YarnClientImpl: Submitted application application_1463925066999_0001
16/05/22 09:54:09 INFO mapreduce.Job: The url to track the job:
http://7b36605a206c:8088/proxy/application_1463925066999_0001/
16/05/22 09:54:09 INFO mapreduce.Job: Running job: job_1463925066999_0001
16/05/22 09:54:22 INFO mapreduce.Job: Job job_1463925066999_0001 running in uber mode : false
16/05/22 09:54:22 INFO mapreduce.Job:  map 0% reduce 0%
16/05/22 09:55:03 INFO mapreduce.Job:  map 3% reduce 0%
16/05/22 09:55:06 INFO mapreduce.Job:  map 66% reduce 0%
16/05/22 09:55:07 INFO mapreduce.Job:  map 100% reduce 0%
16/05/22 09:55:28 INFO mapreduce.Job:  map 100% reduce 67%
16/05/22 09:55:34 INFO mapreduce.Job:  map 100% reduce 100%
16/05/22 09:55:36 INFO mapreduce.Job: Job job_1463925066999_0001 completed successfully
16/05/22 09:55:36 INFO mapreduce.Job: Counters: 49
    File System Counters
        FILE: Number of bytes read=18000006
        FILE: Number of bytes written=36346367
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=19174625
        HDFS: Number of bytes written=22408033
        HDFS: Number of read operations=9
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=2
    Job Counters
        Launched map tasks=2
        Launched reduce tasks=1
        Data-local map tasks=2
        Total time spent by all maps in occupied slots (ms)=85874
        Total time spent by all reduces in occupied slots (ms)=23760
        Total time spent by all map tasks (ms)=85874
        Total time spent by all reduce tasks (ms)=23760
        Total vcore-seconds taken by all map tasks=85874
        Total vcore-seconds taken by all reduce tasks=23760
        Total megabyte-seconds taken by all map tasks=87934976
        Total megabyte-seconds taken by all reduce tasks=24330240
    Map-Reduce Framework
        Map input records=1000000
        Map output records=1000000
        Map output bytes=16000000
        Map output materialized bytes=18000012
        Input split bytes=226
        Combine input records=0
        Combine output records=0
        Reduce input groups=1
        Reduce shuffle bytes=18000012
        Reduce input records=1000000
        Reduce output records=1000000
        Spilled Records=2000000
        Shuffled Maps =2
        Failed Shuffles=0
        Merged Map outputs=2
        GC time elapsed (ms)=3133
        CPU time spent (ms)=19760
        Physical memory (bytes) snapshot=417099776
        Virtual memory (bytes) snapshot=2082435072
        Total committed heap usage (bytes)=414498816
    Shuffle Errors
        BAD_ID=0
        CONNECTION=0
```

Fig .2

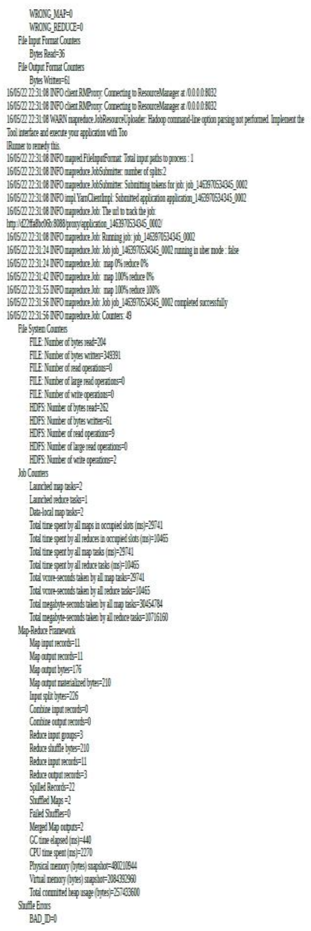
```
        IO_ERROR=0
        WRONG_LENGTH=0
        WRONG_MAP=0
        WRONG_REDUCE=0
    File Input Format Counters
        Bytes Read=19174399
    File Output Format Counters
        Bytes Written=22408033
bash-4.1# hdfs dfs -ls
Found 3 items
drwxr-xr-x   - root supergroup          0 2016-05-22 09:55 birch_output1463925246245
drwxr-xr-x   - root supergroup          0 2015-07-22 11:17 input
drwxr-xr-x   - root supergroup          0 2016-05-22 09:53 input_dataset

bash-4.1# hdfs dfs -cat birch_output1463925246245/part-00000 | head -n 10
27      46.54505081046233
1       42.627554744703794
93      57.0117119065816
53      12.712629680160966
58      7.9672750802387045
39      4.056485452307578
43      32.20255263998891
15      2.5754185718203115
5       3.428837736153479
117     71.71320027809205
```

Fig.3



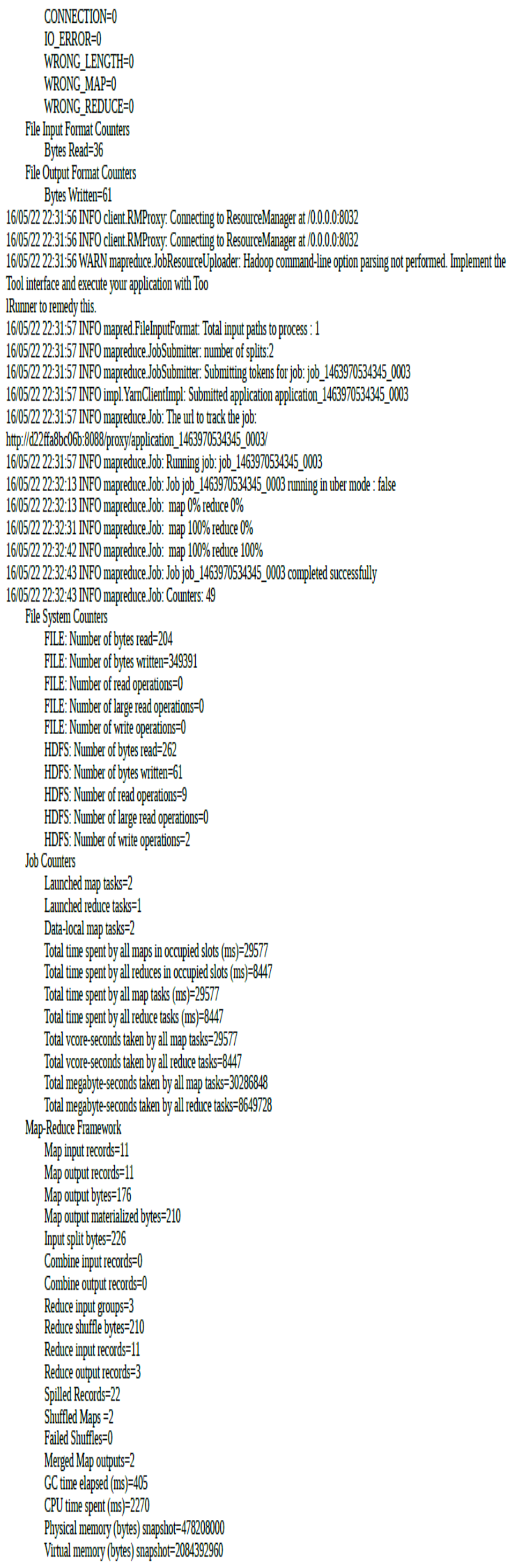
```
K MEDOIDS EXECUTION CONTEXT

bash-4.1# hadoop jar KMedoid.jar KMedoid sampledataset kmedoid_output
16/05/22 22:30:16 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
16/05/22 22:30:17 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
16/05/22 22:30:18 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the
Tool interface and execute your application with Too
lRunner to remedy this.
16/05/22 22:30:18 INFO mapred.FileInputFormat: Total input paths to process : 1
16/05/22 22:30:18 INFO mapreduce.JobSubmitter: number of splits:2
16/05/22 22:30:19 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1463970534345_0001
16/05/22 22:30:20 INFO impl.YarnClientImpl: Submitted application application_1463970534345_0001
16/05/22 22:30:20 INFO mapreduce.Job: The url to track the job:
http://d22ffa8bc06b:8088/proxy/application_1463970534345_0001/
16/05/22 22:30:20 INFO mapreduce.Job: Running job: job_1463970534345_0001
16/05/22 22:30:34 INFO mapreduce.Job: Job job_1463970534345_0001 running in uber mode : false
16/05/22 22:30:34 INFO mapreduce.Job:  map 0% reduce 0%
16/05/22 22:30:55 INFO mapreduce.Job:  map 100% reduce 0%
16/05/22 22:31:06 INFO mapreduce.Job:  map 100% reduce 100%
16/05/22 22:31:07 INFO mapreduce.Job: Job job_1463970534345_0001 completed successfully
16/05/22 22:31:07 INFO mapreduce.Job: Counters: 49
    File System Counters
        FILE: Number of bytes read=204
        FILE: Number of bytes written=349355
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=262
        HDFS: Number of bytes written=61
        HDFS: Number of read operations=9
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=2
    Job Counters
        Launched map tasks=2
        Launched reduce tasks=1
        Data-local map tasks=2
        Total time spent by all maps in occupied slots (ms)=36213
        Total time spent by all reduces in occupied slots (ms)=8569
        Total time spent by all map tasks (ms)=36213
        Total time spent by all reduce tasks (ms)=8569
        Total vcore-seconds taken by all map tasks=36213
        Total vcore-seconds taken by all reduce tasks=8569
        Total megabyte-seconds taken by all map tasks=37082112
        Total megabyte-seconds taken by all reduce tasks=8774656
    Map-Reduce Framework
        Map input records=11
        Map output records=11
        Map output bytes=176
        Map output materialized bytes=210
        Input split bytes=226
        Combine input records=0
        Combine output records=0
        Reduce input groups=3
        Reduce shuffle bytes=210
        Reduce input records=11
        Reduce output records=3
        Spilled Records=22
        Shuffled Maps =2
        Failed Shuffles=0
        Merged Map outputs=2
        GC time elapsed (ms)=502
        CPU time spent (ms)=2680
        Physical memory (bytes) snapshot=470081536
        Virtual memory (bytes) snapshot=2084392960
        Total committed heap usage (bytes)=257433600
    Shuffle Errors
        BAD_ID=0
        CONNECTION=0
        IO_ERROR=0
        WRONG_LENGTH=0
```

Fig. 4



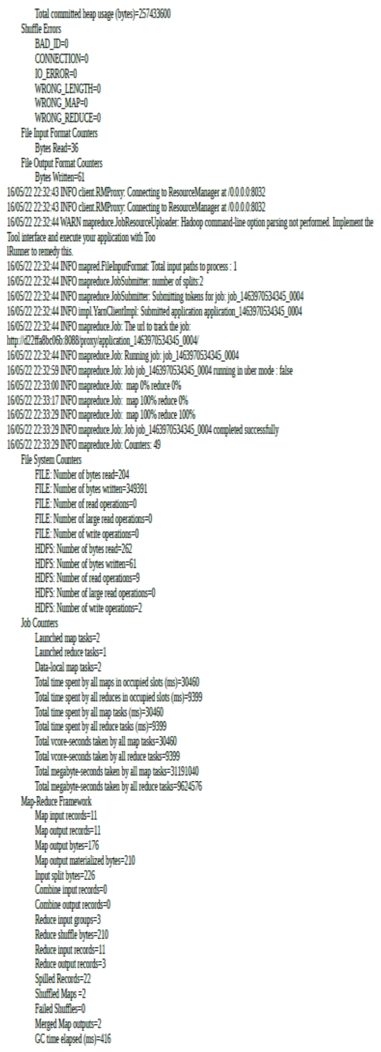
```
        WRONG_MAP=0
        WRONG_REDUCE=0
    File Input Format Counters
        Bytes Read=36
    File Output Format Counters
        Bytes Written=61
16/05/22 22:31:08 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
16/05/22 22:31:08 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
16/05/22 22:31:08 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the
Tool interface and execute your application with Too
lRunner to remedy this.
16/05/22 22:31:08 INFO mapred.FileInputFormat: Total input paths to process : 1
16/05/22 22:31:08 INFO mapreduce.JobSubmitter: number of splits:2
16/05/22 22:31:08 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1463970534345_0002
16/05/22 22:31:08 INFO impl.YarnClientImpl: Submitted application application_1463970534345_0002
16/05/22 22:31:08 INFO mapreduce.Job: The url to track the job:
http://d22ffa8bc06b:8088/proxy/application_1463970534345_0002/
16/05/22 22:31:08 INFO mapreduce.Job: Running job: job_1463970534345_0002
16/05/22 22:31:24 INFO mapreduce.Job: Job job_1463970534345_0002 running in uber mode : false
16/05/22 22:31:24 INFO mapreduce.Job:  map 0% reduce 0%
16/05/22 22:31:42 INFO mapreduce.Job:  map 100% reduce 0%
16/05/22 22:31:55 INFO mapreduce.Job:  map 100% reduce 100%
16/05/22 22:31:56 INFO mapreduce.Job: Job job_1463970534345_0002 completed successfully
16/05/22 22:31:56 INFO mapreduce.Job: Counters: 49
    File System Counters
        FILE: Number of bytes read=204
        FILE: Number of bytes written=349391
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=262
        HDFS: Number of bytes written=61
        HDFS: Number of read operations=9
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=2
    Job Counters
        Launched map tasks=2
        Launched reduce tasks=1
        Data-local map tasks=2
        Total time spent by all maps in occupied slots (ms)=29741
        Total time spent by all reduces in occupied slots (ms)=10465
        Total time spent by all map tasks (ms)=29741
        Total time spent by all reduce tasks (ms)=10465
        Total vcore-seconds taken by all map tasks=29741
        Total vcore-seconds taken by all reduce tasks=10465
        Total megabyte-seconds taken by all map tasks=30454784
        Total megabyte-seconds taken by all reduce tasks=10716160
    Map-Reduce Framework
        Map input records=11
        Map output records=11
        Map output bytes=176
        Map output materialized bytes=210
        Input split bytes=226
        Combine input records=0
        Combine output records=0
        Reduce input groups=3
        Reduce shuffle bytes=210
        Reduce input records=11
        Reduce output records=3
        Spilled Records=22
        Shuffled Maps =2
        Failed Shuffles=0
        Merged Map outputs=2
        GC time elapsed (ms)=440
        CPU time spent (ms)=2270
        Physical memory (bytes) snapshot=480210944
        Virtual memory (bytes) snapshot=2084392960
        Total committed heap usage (bytes)=257433600
    Shuffle Errors
        BAD_ID=0
```

Fig.5



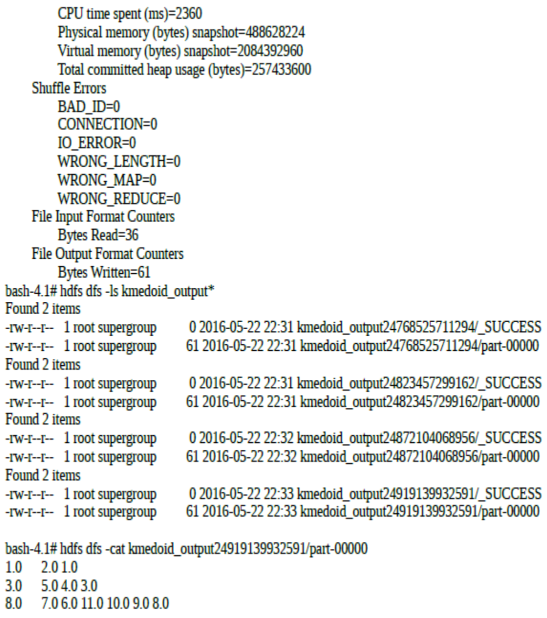
```
        CONNECTION=0
        IO_ERROR=0
        WRONG_LENGTH=0
        WRONG_MAP=0
        WRONG_REDUCE=0
    File Input Format Counters
        Bytes Read=36
    File Output Format Counters
        Bytes Written=61
16/05/22 22:31:56 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
16/05/22 22:31:56 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
16/05/22 22:31:56 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the
Tool interface and execute your application with Too
lRunner to remedy this.
16/05/22 22:31:57 INFO mapred.FileInputFormat: Total input paths to process : 1
16/05/22 22:31:57 INFO mapreduce.JobSubmitter: number of splits:2
16/05/22 22:31:57 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1463970534345_0003
16/05/22 22:31:57 INFO impl.YarnClientImpl: Submitted application application_1463970534345_0003
16/05/22 22:31:57 INFO mapreduce.Job: The url to track the job:
http://d22ffa8bc06b:8088/proxy/application_1463970534345_0003/
16/05/22 22:31:57 INFO mapreduce.Job: Running job: job_1463970534345_0003
16/05/22 22:32:13 INFO mapreduce.Job: Job job_1463970534345_0003 running in uber mode : false
16/05/22 22:32:13 INFO mapreduce.Job:  map 0% reduce 0%
16/05/22 22:32:31 INFO mapreduce.Job:  map 100% reduce 0%
16/05/22 22:32:42 INFO mapreduce.Job:  map 100% reduce 100%
16/05/22 22:32:43 INFO mapreduce.Job: Job job_1463970534345_0003 completed successfully
16/05/22 22:32:43 INFO mapreduce.Job: Counters: 49
    File System Counters
        FILE: Number of bytes read=204
        FILE: Number of bytes written=349391
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=262
        HDFS: Number of bytes written=61
        HDFS: Number of read operations=9
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=2
    Job Counters
        Launched map tasks=2
        Launched reduce tasks=1
        Data-local map tasks=2
        Total time spent by all maps in occupied slots (ms)=29577
        Total time spent by all reduces in occupied slots (ms)=8447
        Total time spent by all map tasks (ms)=29577
        Total time spent by all reduce tasks (ms)=8447
        Total vcore-seconds taken by all map tasks=29577
        Total vcore-seconds taken by all reduce tasks=8447
        Total megabyte-seconds taken by all map tasks=30286848
        Total megabyte-seconds taken by all reduce tasks=8649728
    Map-Reduce Framework
        Map input records=11
        Map output records=11
        Map output bytes=176
        Map output materialized bytes=210
        Input split bytes=226
        Combine input records=0
        Combine output records=0
        Reduce input groups=3
        Reduce shuffle bytes=210
        Reduce input records=11
        Reduce output records=3
        Spilled Records=22
        Shuffled Maps =2
        Failed Shuffles=0
        Merged Map outputs=2
        GC time elapsed (ms)=405
        CPU time spent (ms)=2270
        Physical memory (bytes) snapshot=478208000
        Virtual memory (bytes) snapshot=2084392960
```

Fig.6



```
          Total committed heap usage (bytes)=257433600
     Shuffle Errors
          BAD_ID=0
          CONNECTION=0
          IO_ERROR=0
          WRONG_LENGTH=0
          WRONG_MAP=0
          WRONG_REDUCE=0
     File Input Format Counters
          Bytes Read=36
     File Output Format Counters
          Bytes Written=61
16/05/22 22:32:43 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
16/05/22 22:32:43 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
16/05/22 22:32:44 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the
Tool interface and execute your application with Too
lRunner to remedy this.
16/05/22 22:32:44 INFO mapred.FileInputFormat: Total input paths to process : 1
16/05/22 22:32:44 INFO mapreduce.JobSubmitter: number of splits:2
16/05/22 22:32:44 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1463970534345_0004
16/05/22 22:32:44 INFO impl.YarnClientImpl: Submitted application application_1463970534345_0004
16/05/22 22:32:44 INFO mapreduce.Job: The url to track the job:
http://d22ffa8bc06b:8088/proxy/application_1463970534345_0004/
16/05/22 22:32:44 INFO mapreduce.Job: Running job: job_1463970534345_0004
16/05/22 22:32:59 INFO mapreduce.Job: Job job_1463970534345_0004 running in uber mode : false
16/05/22 22:33:00 INFO mapreduce.Job:  map 0% reduce 0%
16/05/22 22:33:17 INFO mapreduce.Job:  map 100% reduce 0%
16/05/22 22:33:29 INFO mapreduce.Job:  map 100% reduce 100%
16/05/22 22:33:29 INFO mapreduce.Job: Job job_1463970534345_0004 completed successfully
16/05/22 22:33:29 INFO mapreduce.Job: Counters: 49
     File System Counters
          FILE: Number of bytes read=204
          FILE: Number of bytes written=349391
          FILE: Number of read operations=0
          FILE: Number of large read operations=0
          FILE: Number of write operations=0
          HDFS: Number of bytes read=262
          HDFS: Number of bytes written=61
          HDFS: Number of read operations=9
          HDFS: Number of large read operations=0
          HDFS: Number of write operations=2
     Job Counters
          Launched map tasks=2
          Launched reduce tasks=1
          Data-local map tasks=2
          Total time spent by all maps in occupied slots (ms)=30460
          Total time spent by all reduces in occupied slots (ms)=9399
          Total time spent by all map tasks (ms)=30460
          Total time spent by all reduce tasks (ms)=9399
          Total vcore-seconds taken by all map tasks=30460
          Total vcore-seconds taken by all reduce tasks=9399
          Total megabyte-seconds taken by all map tasks=31191040
          Total megabyte-seconds taken by all reduce tasks=9624576
     Map-Reduce Framework
          Map input records=11
          Map output records=11
          Map output bytes=176
          Map output materialized bytes=210
          Input split bytes=226
          Combine input records=0
          Combine output records=0
          Reduce input groups=3
          Reduce shuffle bytes=210
          Reduce input records=11
          Reduce output records=3
          Spilled Records=22
          Shuffled Maps =2
          Failed Shuffles=0
          Merged Map outputs=2
          GC time elapsed (ms)=416
```

Fig.7



```
          CPU time spent (ms)=2360
          Physical memory (bytes) snapshot=488628224
          Virtual memory (bytes) snapshot=2084392960
          Total committed heap usage (bytes)=257433600
     Shuffle Errors
          BAD_ID=0
          CONNECTION=0
          IO_ERROR=0
          WRONG_LENGTH=0
          WRONG_MAP=0
          WRONG_REDUCE=0
     File Input Format Counters
          Bytes Read=36
     File Output Format Counters
          Bytes Written=61
bash-4.1# hdfs dfs -ls kmedoid_output*
Found 2 items
-rw-r--r--   1 root supergroup        0 2016-05-22 22:31 kmedoid_output24768525711294/_SUCCESS
-rw-r--r--   1 root supergroup       61 2016-05-22 22:31 kmedoid_output24768525711294/part-00000
Found 2 items
-rw-r--r--   1 root supergroup        0 2016-05-22 22:31 kmedoid_output24823457299162/_SUCCESS
-rw-r--r--   1 root supergroup       61 2016-05-22 22:31 kmedoid_output24823457299162/part-00000
Found 2 items
-rw-r--r--   1 root supergroup        0 2016-05-22 22:32 kmedoid_output24872104068956/_SUCCESS
-rw-r--r--   1 root supergroup       61 2016-05-22 22:32 kmedoid_output24872104068956/part-00000
Found 2 items
-rw-r--r--   1 root supergroup        0 2016-05-22 22:33 kmedoid_output24919139932591/_SUCCESS
-rw-r--r--   1 root supergroup       61 2016-05-22 22:33 kmedoid_output24919139932591/part-00000

bash-4.1# hdfs dfs -cat kmedoid_output24919139932591/part-00000
1.0     2.0 1.0
3.0     5.0 4.0 3.0
8.0     7.0 6.0 11.0 10.0 9.0 8.0
```

Fig.8

Table 1.Experimental Analysis of Clustering Technique

Performance Measure	**KM**	**BIRCH**	**Analysis**
Execution Time	> BIRCH	< KM	Variable with respect to dataset and computing
Memory Usage	> BIRCH	< KM	Variable with respect to dataset and computing power

8. CONCLUSION

Cluster analysis is the data mining techniques becomes progressively more and more popular and there is a continuous development of the algorithms used in the process. It allows us to cluster data based on common characteristics, similarities in situations, when there are none pre-defined rules for doing so. These rules are being discovered during the execution of the clustering algorithm

This paper is mainly focus on two clustering algorithms, KM and BIRCH clustering algorithm. KM algorithm number of clusters k is determined before the first iteration and is constant during the execution of the algorithm. However, the algorithm does not calculate centroids like k-means, but medoids (mc) instead, which are considered the representative objects for each cluster. BIRCH clustering method is used for very large dataset. This method is used for reducing the number of Input/output operations. It can produce the best quality cluster in available memory and time constraint.

In this work we analyze the KM and BIRCH technique on the Big data set. After analyzing we conclude that BIRCH technique is better than KM technique.

9. ACKNOWLEDGMENTS

I would like to thanks to my project guide, Prof. S. S. Banait, Associate professor in computer department, and his immense support to complete this paper in effective way.

10. REFERENCES

[1] Fahad Adil, Najlaa Alshatri, Zahir Tari, Abdullah Alamri, Ibrahim Khalil, Albert Y. Zomaya, Sebti Foufou, and Abdelaziz Bouras, A Survey of Clustering Algorithms for Big Data: Taxonomy and Empirical Analysis , on Emerging Topics on Computing, IEEE, 11 June 2014Ding, W. and Marchionini, G. 1997 A Study on Video Browsing Strategies. Technical Report. University of Maryland at College Park.

[2] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, et al, “Big data: the next frontier for innovation, competition and productivity,” McKinsey Global Institute, June 2011. Available: http://www.mckinsey.com/Insights/MGI/Research/Technology_and_Innovation/Big_data_The_next_frontier_for_innovikiation. Forman, G. 2003. An extensive empirical study of feature selection metrics for text classification. J. Mach. Learn. Res. 3 (Mar. 2003), 1289-1305.

[3] D. Jiang, B. C. Ooi, L. Shi, and S. Wu. The Performance of Map-Reduce: An In-depth Study. PVLDB, 3(1), 2010.

[4] Hadoop-Map-Reduce-Tutoral. http://hadoop.apache.org/common/docs/r0.20.2/mapred_tutorial.html

[5] T. Zhang, R. Ramakrishnan, and M. Livny, ”BIRCH: An effcient data clustering method for very large databases”, in Proc. ACM SIGMOD Rec., Jun. 1996, vol. 25, no. 2, pp. 103-114.

[6] S. Guha, R. Rastogi, and K. Shim,”Cure: An effcient clustering algorithm for large databases”,in Proc. ACMSIGMOD Rec., Jun. 1998, vol. 27, no. 2, pp. 73-84.

[7] G. Karypis, E.-H. Han, and V. Kumar, ”Chameleon: Hierarchical clustering using dynamic modelling”, IEEE Comput., vol. 32, no. 8, pp. 68-75, Aug. 1999.

[8] S. Guha, R. Rastogi, and K. Shim,”Rock: A robust clustering algorithm for categorical attributes”, Inform. Syst., vol. 25, no. 5, pp. 345-366, 2000.

[9] M. Dutta, A. Kakoti Mahanta and A.K. Pujari, QROCK: A quick version of the ROCK algorithm for clustering of categorical data, Pattern Recognition Letters, 26 (2005), 2364-2373.

[10] R. T. Ng and J. Han, ”CLARANS: A method for clustering objects for spatial data mining”, IEEE Trans. Knowl. Data Eng. (TKDE), vol. 14, no. 5, pp. 1003-1016, Sep./Oct. 2002.

[11] ALSABTI K., RANKA S., SINGH V., ”An Efficient k-means Clus- tering Algorithm”, Proc. First Workshop High Performance Data Mining, 1998.

[12] CHU S. C., RODDICK J. F., CHEN T. Y., PAN J. S., Efficient search approaches for k-medoids-based algorithms, TENCON 02,Proceed- ings, 2002 IEEE Region 10 Conference on Computers, Communi- cations, Control and Power Engineering, 2002.

Hae-Sang Park, Chy Hyuck Jun,A Simple And Fast Algorithm For K-Medoid clustering, Department of Industrial and Manage- ment Engineering, POSTECH, San 31 Hyoja-dong, Pohang 790-784, South Korea.