ISSN 2350-1022

International Journal of Recent Research in Mathematics Computer Science and Information Technology
Vol. 2, Issue 1, pp: (153-161), Month: April 2015 – September 2015, Available at: www.paperpublications.org

Big Data on Implementation of Many to Many Clustering

Ravi. R[1,] Michael. G[2]

[1]PG Scholar, [2]Assistant Professor
[1,2] Department of C S E , Bharath University, Selaiyur, Chennai - 600 073, India

Abstract: **With development of the information technology, the scale of data is increasing quickly. The massive data poses a great challenge for data processing and classification. In order to classify the data, there were several algorithm proposed to efficiently cluster the data. One among that is the random forest algorithm, which is used for the feature subset selection. The feature selection involves identifying a subset of the most useful features that produces compatible results as the original entire set of features. It is achieved by classifying the given data. The efficiency is calculated based on the time required to find a subset of features, the effectiveness is related to the quality of the subset of features. The existing system deals with fast clustering based feature selection algorithm, which is proven to be powerful, but when the size of the dataset increases rapidly, the current algorithm is found to be less efficient as the clustering of datasets takes quiet more number of time. Hence the new method of implementation is proposed in this project to efficiently cluster the data and persist on the back-end database accordingly to reduce the time. It is achieved by scalable random forest algorithm. The Scalable random forest is implemented using Map Reduce Programming (An implementation of Big Data) to efficiently cluster the data. In works on two phases, the first step deals with the gathering the datasets and persisting on the datastore and the second step deals with the clustering and classification of data. This process is completely implemented using Google App Engine's hadoop platform, which is a widely used open-source implementation of Google's distributed file system using MapReduce framework for scalable distributed computing or cloud computing. MapReduce programming model provides an efficient framework for processing large datasets in an extremely parallel mining. And it comes to being the most popular parallel model for data processing in cloud computing platform. However, designing the traditional machine learning algorithms with MapReduce programming framework is very necessary in dealing with massive datasets.**

Keywords: **Data mining, Hadoop, Map Reduce, Clustering Tree.**

1. INTRODUCTION

Data linkage is one of the important task in data mining. It is of two kinds: one-to –one, one-to-many and many to many. One-to-one data linkage associates one entity with a matching entity in another data set. One-to-many data linkage associates an entity with a group of matching entities from another data set. Many-to-many data linkage associates group of entities from one data set with group of matching entities from another data set. In this paper many-to-many data linkage is implemented with MapReduce framework.

The massive data poses a great challenge for data processing and classification. In order to classify the data, there were several algorithm proposed to efficiently cluster the data. One among that is the random forest algorithm, which is used for the feature subset selection. The feature selection involves identifying a subset of the most useful features that produces compatible results as the original entire set of features. It is achieved by classifying the given data. The efficiency is calculated based on the time required to find a subset of features, the effectiveness is related to the quality of the subset of features. The existing system deals with fast clustering based feature selection algorithm, which is proven to be powerful, but when the size of the dataset increases rapidly, the current algorithm is found to be less efficient as the clustering of datasets takes quiet more number of time.

Hence the new method of implementation is proposed in this project to efficiently cluster the data and persist on the back-end database accordingly to reduce the time. It is achieved by scalable random forest algorithm. The Scalable random forest is implemented using Map Reduce Programming (An implementation of Big Data) to efficiently cluster the data.

It works on two phases, the first step deals with the gathering the datasets and persisting on the datastore and the second step deals with the clustering and classification of data. This process is completely implemented using Google App Engine's hadoop platform, which is a widely used open-source implementation of Google's distributed file system using MapReduce framework for scalable distributed computing or cloud computing.

Hadoop MapReduce is a software framework for distributed processing of large data sets. It is a sub-project of the Apache Hadoop project. The framework takes care of scheduling tasks, monitoring them and re-executing failed tasks. The primary objective of MapReduce is to split the input data set into independent chunks that are processed in a completely parallel manner.

The Hadoop MapReduce framework sorts the outputs of the maps, which are then input to the reduce tasks. Both the input and the output of the job are stored in a file system. The proposed method with MapReduce links between the entities using a One-Class Clustering Tree. A clustering tree is a tree contains a cluster in each of the leaves instead of a single classification. Each cluster is generalized by a set of rules that is stored in the appropriate leaf. One-class clustering tree is implemented with the help of MapReduce in parallel. Clustering tree assigns each linkage task to mapper class. Master node in the mapper assigns the work to slaves. Slaves in turn completes the work and returns the result to master node. The reducer class combines the intermediate results from mapper class to provide a complete clustering tree.

As clustering tree is implemented in a parallel and distributed environment it reduces the execution time. The objectives of the paper is threefold:

1. To implement one-to-many and many–to-many data linkage

2. To make data linkage for larger datasets more efficient

3. To execute data linkage in a parallel and distributed environment using MapReduce.

2. RELATED WORK

2.1 Map Reduce Overview:

In the beginning of the twenty first century, many computations that are specialized to process large amounts of raw data as crawled documents, web request logs ,etc… These computations were applied by authors and many others at Google. They computed different types of derived data, like inverted indices, summaries of the number of pages per host, various representations of the graph structure of web documents. The input data was usually large in the computations, so it has to be distributed among hundreds or thousands of machines to finish the process in a reasonable time. Although the computations were straightforward, the matters of how to parallelize the computation, distribute the data, and deal with the expected failures ,made the original simple computation to be very complex specially with the huge amount of code to handle these issues. As a result to this complexity a new model was designed to allow expression of the simple computation with hiding the complex details of data distribution, parallelization and fault-tolerance in a library. This model is inspired by the map and 26 reduce primitive in Lisp which is the oldest high-level programming language. The use of this functional abstraction with user specified map and reduce operations enables automatic parallelization and distribution of huge computations, and achieves high performance on large clusters of PCs.

22 Map Reduce Programming Framework:

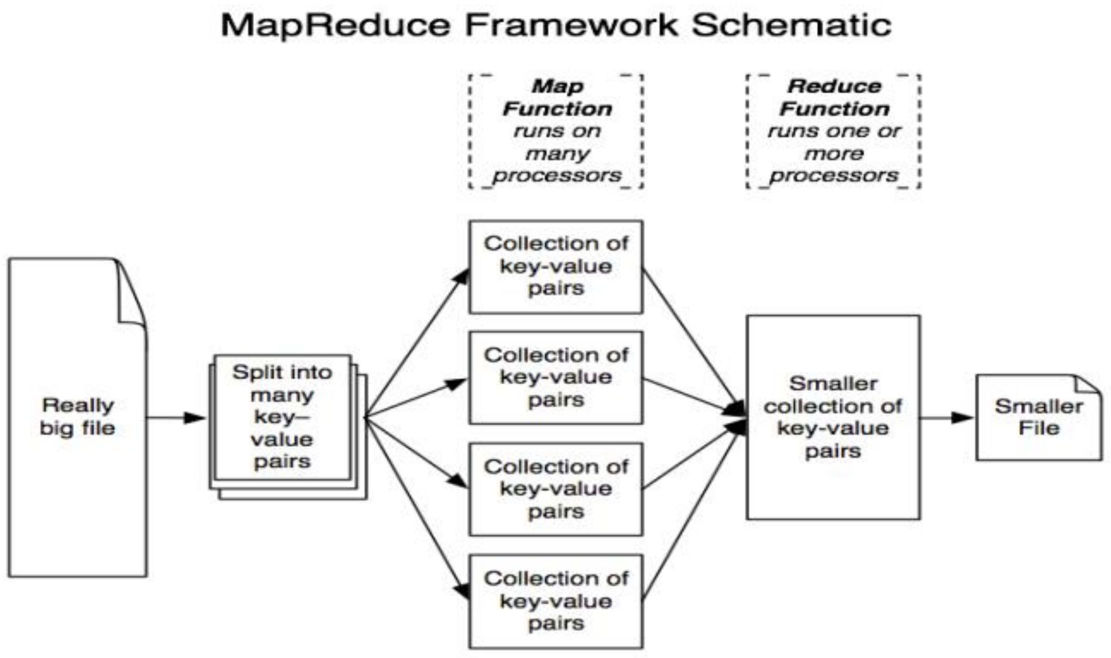
MapReduce is a software framework introduced by Google in2004 to support distributed computing on large data sets onclusters of computers .MapReduce is a programming modelfor processing and generating large data sets. Users specify amap function that processes a key/value pair to generate a setof intermediate key/value pairs and a reduce function thatmerges all intermediate values associated with the same intermediate key

"Map" step: The master node takes the input, partitions it upinto smaller sub-problems, and distributes them to workernodes. A worker node may do this again in turn, leading to amulti-level tree structure. The worker node processes thesmaller problem, and passes the answer back to its masternode. Map takes one pair of data with a type in one datadomain, and returns a list of pairs in a different domain:

Map (k1, v1) → list (K2, v2)

"Reduce" step: The master node then collects the answers toall the sub-problems and combines them in some way to formthe output – the answer to the problem it was originally tryingto solve.The Reduce function is then applied in parallel to each group,which in turn produces a collection of values in the samedomain:

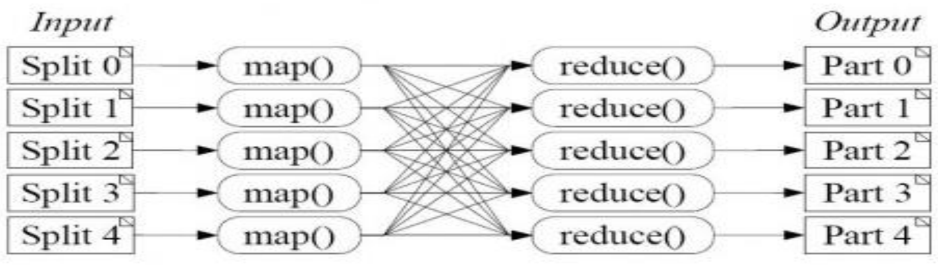
Reduce (K2, list (v2)) → list (v3)


MapReduce Framework Schematic
Map Function runs on many processors
Reduce Function runs one or more processors
Really big file
Split into many key–value pairs
Collection of key-value pairs
Collection of key-value pairs
Collection of key-value pairs
Collection of key-value pairs
Smaller collection of key-value pairs
Smaller File

2.3 Map Reduce Programming Model:

MapReduce is a programming model and a linked implementation for dealing out with many terabytes of data on thousands of machines .Computation obtains a set of input key/value pairs, and generates a set of output key/value pairs, so the user of the MapReduce library states the computation as two functions: Map and Reduce.

In the map reduce function the user gets the data from data sources like lines out of files, rows of a database, etc) and feeds them to the function asan input key/value pair (eg: filename, line). Then it generate a set of intermediate key/value pairs as shown in fig. After that the library combined together these intermediate values related with the same intermediate key, and pass them to the Reduce method which accepts an intermediate key I and a set of values for that key and tries to merges together these values to form a possibly smaller set of values. From practice user can visualize that usually only zero or one final value will be produced per key as presented in figure below



MapReduce Model:

Important thing which should be noticed that all the map( ) functions work in parallel to create different intermediate values from different input data sets, and the same for the reduce( ) functions which run in parallel so each one work on different output key.

Here is a simple example that could explain the functionality of MapReduce Model. In this problem the number of occurrences of each word will be accounted from a large collection of documents. The map function emits each word and an related count of occurrences just like 1 in the simple pseudo-code shown in (Figure 2.8) .On the other hand the reduce function sums together all counts emitted for a certain word. In addition, another code is used by the user to fill in a MapReduce specification object with the names of the input and output files, the user then invokes the MapReduce function, passing it the specification object.

```
map (String key, String value) :
// key : document name
// value : document contents
for each word w in value :
EmitIntermediate(w, “1”);
reduce (String key, Iterator values) :
// key : a word
// value : a list of counts
int result = 0
for each v in values :
result += ParseInt(v)
Emit(AsString(result));
```

As said before when MapReduce process is used, the data must be distributed among the different nodes, so the master program divvies up tasks depending on location of data where it tries to have map() tasks on same node as physical file data, or at least same rack.

Another thing that MapReduce module deals with the expected failures, where the master detects worker failures in the re-executes completed, in-progress map() tasks, and re-executes in-progress reduce() tasks. In addition, master notices particular input key/values that cause crashes in map(), and skips those values on re-execution.

2.4 MapReduce Algorithms:

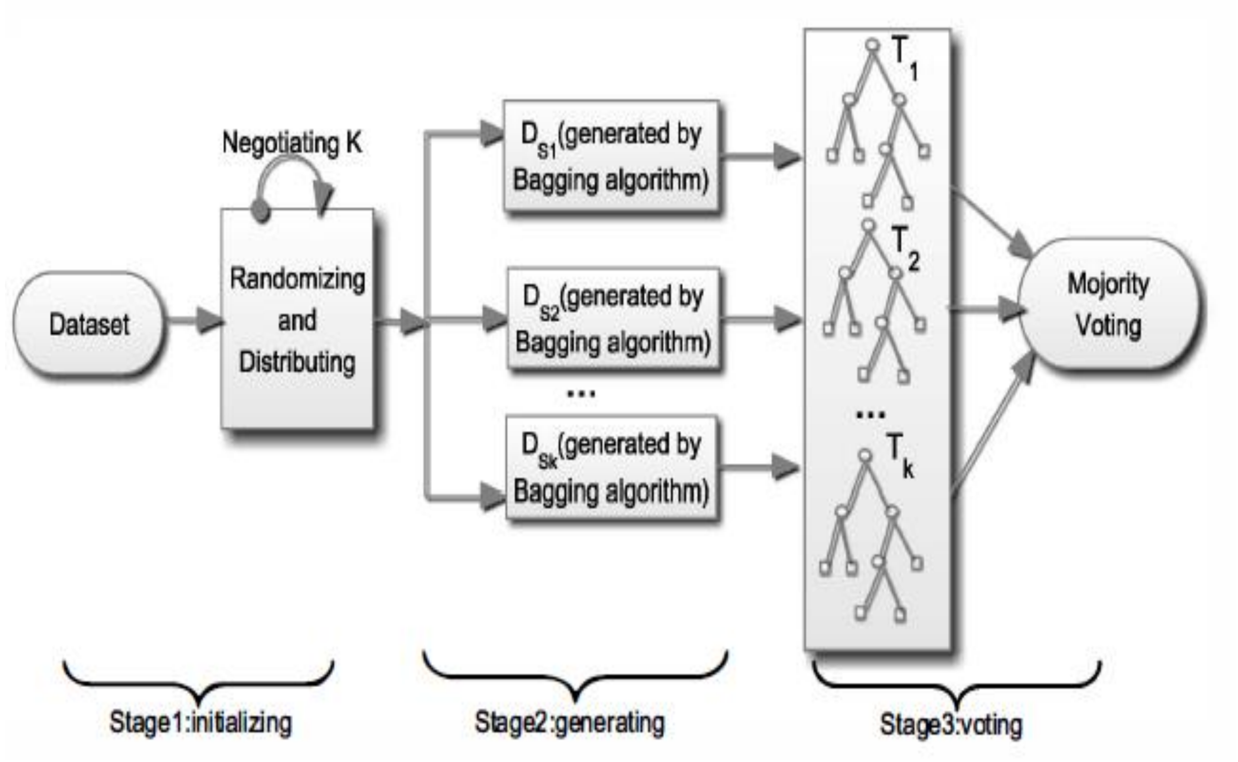
There are many problems that can be mapped to a MapReduce program, such as: sorting, searching, indexing and classification. These programs must fit the features of the MapReduce algorithm. For any MapReduce algorithm, processing data must go through a map phase and a reduce phase. With consideration that the output of the map phase as the input to the reduce phase. In this project we use classification algorithm i.e Scalable random forest algorithm.

2.4.1 Existing Random Forest Algorithm:

Random forest is an ensemble classifier that consists of many decision trees. The method to build the random forest is bagging. Given the independent variable X, Random Forest Classification is an ensemble classification model which combined of K decision tree classifiers hl(X),hiX), ... hdX).Each of the classifier decision trees votes for one of the classifications and the winner is the final classification results. The main work step of Random Forest Classification: Firstly, select K samples from the original training dataset using bagging method randomly . Then, the K samples will be the training set for growing K trees accordingly to achieve the K classification results .Finally, the K classifiers vote to elect the optimal classification with majority.

2.4.2 Scalable Random Forest Algorithm:

Random Forest classification algorithm based on MapReduce-SMRF algorithm. The algorithm structure of the SMRF are showed as Fig. 1, The algorithm is divided into three stages: initializing, generating and voting. In the stage initializing, SMRF algorithm generates a file descriptor for the dataset to describe the attributions, meanwhile, In order to make good use of the computing resources of computer clusters or cloud computing environments. When the SMRF algorithm starting, it will negotiate the trees number of the forest with the controller automatically. The recommended values of K which are depending on the system performance are stored in a Hashlist. The recommended values are based on the recommendation of Breiman.L and experts experiences. cBreiman's recommendations are to pick a large number of trees, as well as the square root of the number of variables of the subsets.

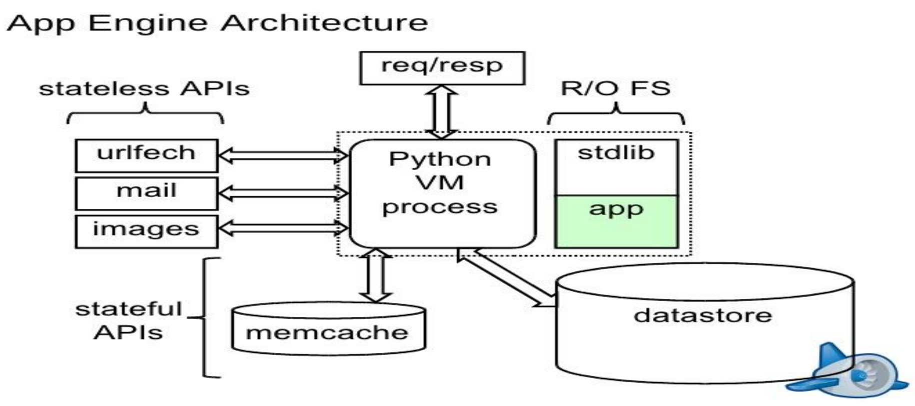

Negotiating K
Dataset
Randomizing and Distributing
D_{S1}(generated by Bagging algorithm)
D_{S2}(generated by Bagging algorithm)
...
D_{Sk}(generated by Bagging algorithm)
T_1
T_2
...
T_k
Mojority Voting
Stage1:initializing
Stage2:generating
Stage3:voting

Scalable Random Forest Algorithm approach

In the stage generating, SMRF algorithm based on MapReduce programming model will divide the original dataset into many subsets randomly. SMRF perform the job by using MapReduce distributed file system named HDFS, meanwhile, the bootstrap samples which are for growing the random forest will be generated from the training dataset by Bagging algorithm. At the same time the attributions will be selected randomly by Bagging algorithm too. In the MapReduce processing, the key of the decision tree is n which represents the n'th decision tree, and the value is the class of each root node of the trees. Demanding on the classification, it is necessary to describe the classification attributions of the training dataset. And the attributions are leaf, real-valued, categorical etc. Each of the decision tree is slitting at the node which has the maximum Information Gain. In the state generating all the decision trees will have grown over at the distributed computing nodes differently. In the stage voting , each of the trees gives a classification, and all the decision trees vote for the results. The forest decides the classification with the majority vote. When it comes to the prediction, each of the K trees will give a prediction value and return an average value of them at last. In the whole SMRF algorithm procedure, The map function selects the training dataset and the attributions randomly from the original dataset to build the decision trees. The reduce function is used to merge all the decision trees votes and generates the last classified results.

2.5 Google App Engine:

Google App Engine is a service for hosting Web applications. It’s often presented as an example eof a platform-as-a-service (PaaS) cloud computing solution because Web application developer scan use it to create their applications using a set of Google tools. In contrast to other cloud services, such as the Amazon Elastic Compute Cloud (EC2), Google App Engine requires that application developers use only a limited set of supported programming languages together with a small set of APIs and frameworks that are mostly dedicated to Web applications. The applications are executed in a secure hosting environment running on the computing infrastructure that Google provides and manages. The infrastructure offers load balancing and auto scaling capabilities — that is, it automatically adjusts the number of application instances (virtual machines) to the request rate.



2.6 Task Queue:

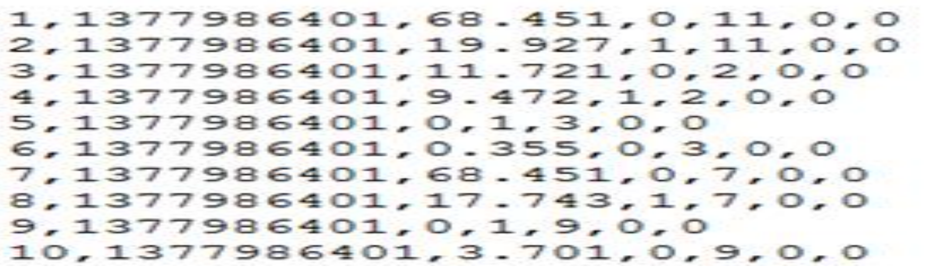
Work focuses on Task Queue, which is particularly interesting for computational applications. Task Queue lets you execute small jobs asynchronously in the background of HTTP request processing. By dispatching tasks to multiple App Servers according to the current load, Task Queue becomes a perfect tool to provide scalability. However, using Task Queue and the Google App Engine's execution environment presents several difficulties. Task execution must be idempotent, because Google guarantees that each task will be executed at least one time, which might lead, in exceptional circumstances, to some tasks re executing. The execution model doesn't ensure any kind of task concurrency control, and Google doesn't provide APIs for inter task communication. It's therefore impossible to specify when and on which application server a given task should be executed. The only way to provide data flow between tasks is to store and read data from the Big table database.

2.7 Extension Possibilities:

Although Google App Engine mainly hosts Web applications that handle many short (interactive)requests, there are use cases in which longer background processing is required. In addition to the Task Queue API there are other including the Pipeline API and MapReduce. The Pipeline API allows developers to create workflow-based applications in which larger computations are split into tasks (pipelines) that are connected using data dependencies. The pipelines can be used to represent not only processing tasks but also human interactions using Web requests, email, and so on. The Pipeline API implements the MapReduce framework on top of App Engine. Users must define the map() function, which will be iterated by the framework over entries in the input dataset, which might include data store objects or lines in blobstore, or be provided by a custom reader. The tasks in both the Pipeline and MapReduce APIs follow the same constraints and quotas as the ones in Task Queue, which means that these APIs can be used at no cost in the same way as framework for parameter study applications.

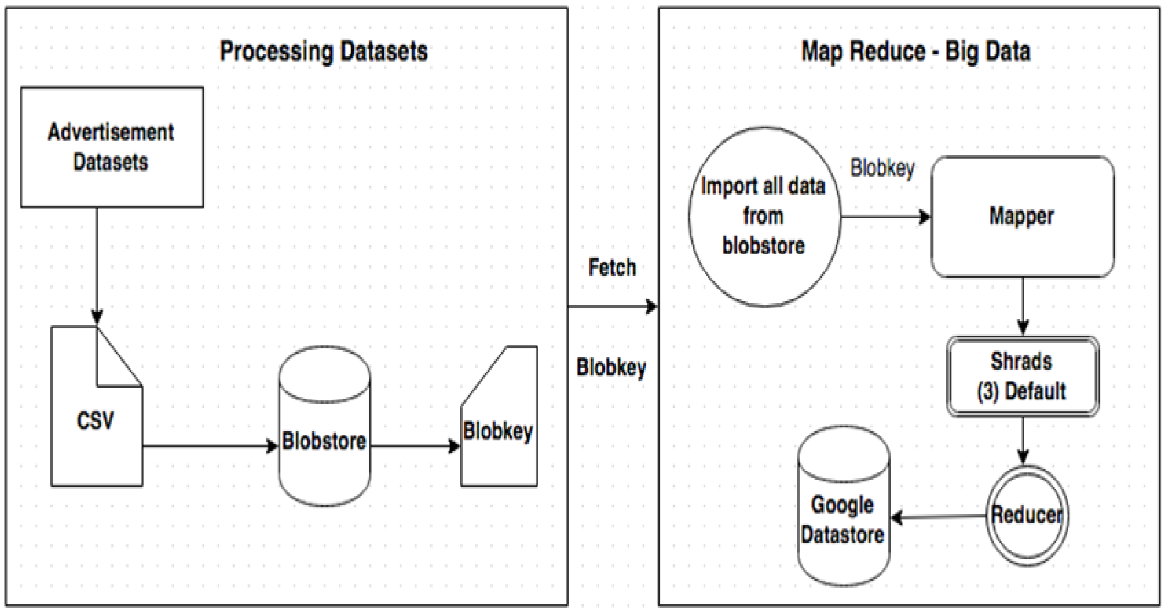
2.8 Data Storage Management:

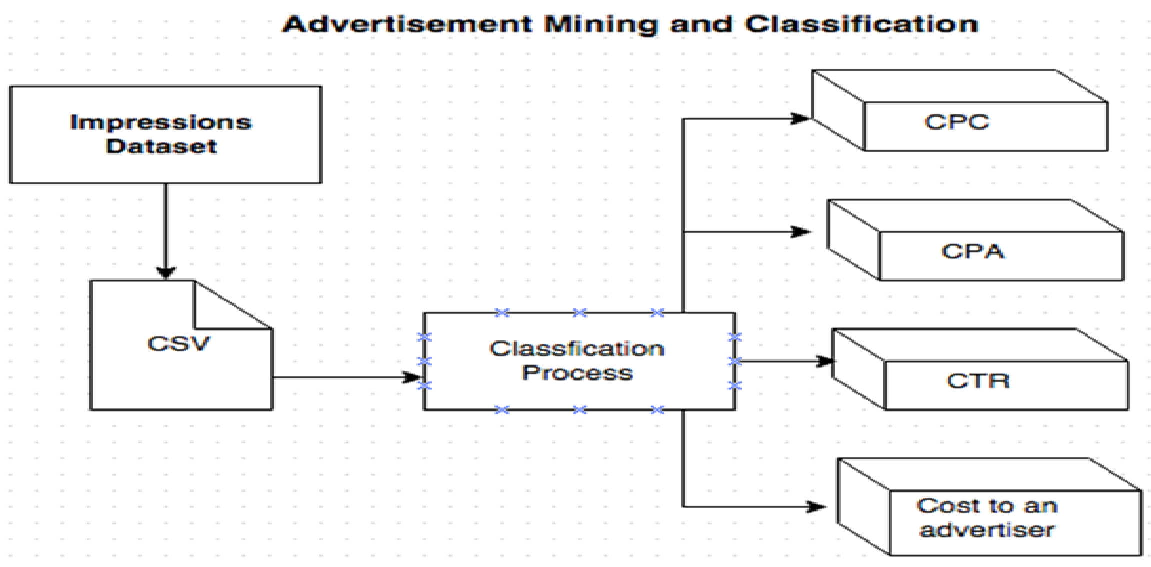
Data is stored in CSV format. A comma-separated value (CSV) file stores tabular data (numbers and text) in plain-text form. Plain text means that the file is a sequence of characters, readable as textual or numerical material without much processing, usually opposed to formatted text. A CSV file consists of any number off records, separated by line breaks of some kind; each record consists of fields, separated by some other character or string, most commonly a literal comma or tab or new line. Usually, all records have an identical sequence of fields. CSV formats are not limited to a particular character set. They work just as well with Unicode as with ASCII.



```
1,1377986401,68.451,0,11,0,0
2,1377986401,19.927,1,11,0,0
3,1377986401,11.721,0,2,0,0
4,1377986401,9.472,1,2,0,0
5,1377986401,0,1,3,0,0
6,1377986401,0.355,0,3,0,0
7,1377986401,68.451,0,7,0,0
8,1377986401,17.743,1,7,0,0
9,1377986401,0,1,9,0,0
10,1377986401,3.701,0,9,0,0
```

3. PROPOSED ARCHITECTURE





The advertise dataset Is taken in to consideration in order to effectively apply the mining and clustering technique to classify the advertisements to different categories based on several factors like click rate, cost of advertising etc.

The main work towards the step of Algorithm is: Firstly, the samples from the original dataset is selected. Then, the samples will be the training set for growing K trees accordingly to achieve the K classification results . Before that, the classification is relied on a mathematical calculation which is depending on the other parameters of the dataset.

The final classification of datasets is done on as CPC, CPA, CTR, Cost of Advertisement.

4. MODULES

4.1 Login – Authentication:

Authentication is used to make the application much secured and allow only the authorized person to use the application. Google Data stores is used to store the user information, and the same is applied during the authentication process.

4.2 Split data into Subset:

This process is technically done by using map reduce algorithm. The Mapper is used for splitting as the data set is huge in quantity. For Splitting, Map Reduce uses the concept called InputSplit. InputSplit represents the data to be processed by an Individual Mapper.

4.3 Data Parallel Processing (Mapper):

In this module, we implement or configure the map reduce job for dividing the input data in to different clusters and share it among multiple nodes and process in parallel.

4.4 Combine Intermediate Result (Reducer):

In this module, we combine the intermediate results from all individual map reduce job, that we allocated on the previous module. After combing, the negotiation of K value for feature subset selection is processed.

4.5 Collect AD Impressions Datasets:

- Cost-per click is important because it is the number that is going to determine the financial success of your paid search advertising campaign.
- CPA advertising tracks the person clicking on an ad and determines if that person then also creates a desired transaction on the destination site.

4.6 Classification process (using Map Reduce):

- The advent of revolution in technology and internet has caused increase in marketing platform for advertising companies.
- The classification is done in order to categorize the advertisements based on different factors that includes, CPA, CPC. Etc.
- There are various models for determining the cost of advertising. They are Cost per Impressions (CPM), Cost per Click (CPC) and Cost per Action (CPA).
- CPM denotes Cost per 1000 Impressions (frequency of the ad display). If the advertisement is shown 2000 times the cost will be equal to 2 CPM price.

5. CONCLUSION

The proposed methodology is designed based on MapReduce model and it is likely to work mostly in computer clusters or cloud computing environments. Internally the proposed mechanism uses the MapReduce framework that uses the shell script to fetch the number of computing nodes and negotiates with the distributed computing environments to decide the parameters K. Technically, the K Values are stored in Hash List in order to apply the Many-Many Clustering in a sequential order. The Data Clustering is tested with 29567 rows of sample data sets. It is experimented in the distributed environment with changed computing nodes. Observations are made by giving various shrads values and increasing the datasets. The more decision trees will bring the better classification results. The proposed framework's parallel performances are mainly depend on the MapReduce model and the model has already been proved of good Scalability. From all the results we evaluate the many to many clustering performances and prove that to have good scalable performance.

ISSN 2350-1022

International Journal of Recent Research in Mathematics Computer Science and Information Technology
Vol. 2, Issue 1, pp: (153-161), Month: April 2015 – September 2015, Available at: www.paperpublications.org

REFERENCES

[1] Ma'ayan Dror, Asaf Shabtai, Lior Rokach, Yuval Elovici "OCCT: A One-Class Clustering Tree for Implementing One-to-Many Data Linkage", IEEE transactions on Knowledge and Data engineering, Vol. 26, No. 3, March 2014

[2] C. Li, Y. Zhang, and X. Li, "OcVFDT: One-Class Very Fast Decision Tree for One-Class Classification of Data Streams," Proc. Third Int'l Workshop Knowledge Discovery from Sensor Data, pp. 79- 86, 2009.

[3] Anne-Laure Boulesteix, Silke Janitza J ochen Kruppa, Inke R. K¨onig "Overview of Random Forest Methodology and Practical Guidance with Emphasis on Computational Biology and Bioinformatics " pre-review version of a manuscript accepted for publication in WIREs Data Mining & Knowledge Discovery , July 25th 2012

[4] Chen, W Y; et al. (2011). "Parallel Spectral Clustering in Distributed Systems". IEEE Trans. Pattern Anal. Mach. Intell., 568-586.

[5] Jiawei Hanl, Yanheng Liul, Xin Sunl "A Scalable Random Forest Algorithm Based on MapReduce" presented at the IEEE Summer Power Meeting , 2013 IEEE

[6] Aditya B. Patel, Manashvi Birla, Ushma Nair "Addressing Big Data Problem Using Hadoop and Map Reduce" presented at NIRMA university international conference on engineering NUiCONE-2012, 06-08December, 2012.

[7] Jyoti Nandimath , Ankur Patil, Ekata Banerjee, Pratima Kakade "Big Data Analysis Using Apache Hadoop" presented at IEEE IRI 2013, August 14-16, 2013, San Francisco, California, USA

[8] J. Dean and S. Ghemawat, "MapReduce: simplified data processingon large clusters," Commun. ACM, vol. 51, no. I, pp. 107-113, 2008.

[9] Apache Software Foundation. Official apache hadoop website, http://hadoop.apache.org/, Aug, 2012.