Parallel K-prototypes for Clustering Big Data

Mohamed Aymen Ben HajKacem(✉), Chiheb-Eddine Ben N'cir, and Nadia Essoussi

LARODEC, Université de Tunis, Institut Supérieur de Gestion de Tunis, 41 Avenue de la Liberté, Cité Bouchoucha, 2000 Le Bardo, Tunisia
medaymen.hajkacem@gmail.com, {chiheb.benncir,nadia.essoussi}@isg.rnu.tn

Abstract. Big data clustering has become an important challenge in data mining. Indeed, Big data are often characterized by a huge volume and a variety of attributes namely, numerical and categorical. To deal with these challenges, we propose the parallel k-prototypes method which is based on the Map-Reduce model. This method is able to perform efficient groupings on large-scale and mixed type of data. Experiments realized on huge data sets show the performance of the proposed method in clustering large-scale of mixed data.

Keywords: Big data · K-prototypes · Map-reduce · Mixed data

1 Introduction

Given the exponential growth and availability of data collected from different resources, analyzing these data has become an important challenge referred to as Big data analysis. Big data usually refer to three mains characteristics also called the three Vs [4] which are respectively Volume, Variety and Velocity. Volume refers to the increased quantity of generated data. Variety indicates the many different data types and formats in which data is produced. Velocity refers to the speed at which data should be analyzed. Analyzing Big data usually requires powerful methods and tools since traditional ones are not suitable for processing large and heterogeneous amount of data. For example, existing methods which are used to organize data into groups of similar objects, fail to deal with large-scale of data. The fail is explained by the high computational costs of the existing clustering methods which require unrealistic time to build the grouping.

To obviate this problem, several parallel clustering methods [1,2,6,8,10,11,15] have been proposed in the literature. Most of these methods use the Map-Reduce [3], which is a parallel programming model to process large-scale data sets. For example, Zaho et al. [15] have proposed a parallel k-means based on Map-Reduce. This method first computes locally the centers of clusters in the map phase. Then, the reduce phase updates the global centers. Kim et al. [8] have introduced an implementation of DBSCAN method using Map-Reduce model. This method first partitions the data to find the clusters in each partition via map phase. Then, it merges the clusters from every partition in reduce phase.

Recently, a parallel implementation of fuzzy c-means clustering algorithm using Map-Reduce model is presented in [11]. The proposed method consists in two Map-Reduce jobs. The first one calculates the centroid matrix, and the second one is devoted to generate the membership matrix.

The later discussed methods offer for users an efficient analysis of a huge amount of data through a Map-Reduce model. Nevertheless, these methods can not handle different types of data and are limited to only numerical attributes. Since Big data are also characterized by the variety of attributes' type, including numerical and categorical attributes, we focus our study on the challenge of clustering large amount of mixed data. We propose in this paper the parallel k-prototypes method called PKP, which is based on the Map-Reduce model to cluster large-scale of mixed data. To the best of our knowledge, this is the first work that deals with numerical and categorical large amount of data.

The rest of this paper is organized as follows: Section 2 presents some existing clustering methods for mixed data, with an emphasis on the k-prototypes method. Then, Section 3 describes the Map-Reduce model while Section 4 describes our proposed method. After that, Section 5 presents experiments that we have realized to evaluate the performance of the proposed method. Finally, we conclude this paper in Section 6.

2 Clustering Methods for Mixed Data

Clustering is an important task in data mining that aims to organize data into groups of similar observations. Given that data are often described by different types of attributes such as, numerical and categorical, a pre-processing step is usually required to transform data into a single type since most of proposed clustering methods deal with only numerical or categorical attributes. However, few clustering methods have been proposed to deal with mixed types of data. For instance, Li and Biswas [9] introduced the Similarity-Based Agglomerative Clustering called SBAC, which is a hierarchical agglomerative algorithm for mixed data. Huang [7] proposed k-prototypes method which integrates k-means and k-modes methods to cluster numerical and categorical data. Ji et al. [5] proposed an improved k-prototypes to deal with mixed type of data. This method introduced the concept of the distributed centroid for representing the prototype of categorical attributes in a cluster. Among the later discussed methods, k-prototypes remains the most popular method to deal with mixed data because of its efficiency [5]. In the following, we present the k-prototypes method.

- K-prototypes method

Given a data set X containing n data objects described by m_r numerical attributes and m_c categorical attributes, the aim of k-prototypes [7] is to find k groupings where the following objective function is minimized:

$$J = \sum_{l=1}^{k} \sum_{i=1}^{n} p_{il} d(x_i, Q_l), \tag{1}$$

Where $p_{il} \in \{0, 1\}$ is a binary variable indicating the membership of data object x_i in cluster l, Q_l is the prototype of the cluster l and $d(x_i, Q_l)$ is the dissimilarity measure which is defined as follows:

$$d(x_i, Q_l) = \sum_{r=1}^{m_r} (x_{ir} - q_{lr})^2 + \gamma_l \sum_{c=1}^{m_c} \delta(x_{ic}, q_{lc}), \tag{2}$$

Here x_{ir} represents the values of numeric attributes and x_{ic} represents the values of categorical attributes for each data object x_i. $Q_l = \{q_{l1}, \dots q_{lm}\}$ represents the cluster centers for cluster l, where q_{lr} is the mean of numeric attribute r and cluster l, q_{lc} is the most common value (mode) for categorical attributes c and cluster l. For categorical attributes, δ(p,q)=0 for $p \equiv q$ and $\delta(p, q) = 1$ for $p \neq q$. γ_l is a weight for categorical attributes for each cluster l. The main algorithm of k-prototypes method is described in Algorithm 1.1.

Algorithm 1.1. k-prototypes method

```
Data: Data set X={$x_1 \dots x_n$}; number of clusters k
Result: Cluster centers Q={$Q_1 \dots Q_k$};
begin
    Choose k cluster centers from X
    repeat
        Compute distance between data objects and cluster centers using Equation 2
        Update the cluster centers Q (Save the previous cluster centers as $Q^\wedge$ to analyze the convergence)
    until $Q^\wedge = Q$;
end
```

Although the interesting results shown by the k-prototypes method with small and medium mixed data sets, it fails to deal with large-scale data sets (from millions of instances) [13]. Since, it is necessary to compute the distance between each data object to each center, in each iteration. Here, the distance computation is the time consuming step, especially when the size of data sets increases highly. To overcome this weakness, we propose the parallel k-prototypes method which is based on the Map-Reduce model to handle large-scale of mixed data. We present in the following section the Map-Reduce model.

3 Map-Reduce Model

Map-Reduce [3] is a parallel programming model designed to process large-scale data sets among cluster nodes (i.e machines). It is characterized by the fact that the user is oblivious of the details about the data storage, distribution and replication. This model specifies the computation as two functions namely, map and reduce. Each one has $< key/value >$ pairs as input and output. The map function takes each $< key/value >$ pair and generates a set of intermediate

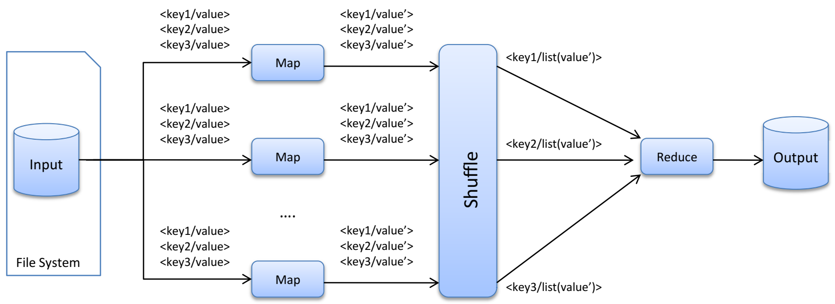

Fig. 1. Map-Reduce flowchart

$< key/value >$ pairs. Then, shuffle phase is devoted to merge all the values associated with the same intermediate key as a list. The reduce function takes this list as input for creating the final values. Fig. 1. illustrates the flowchart of the Map-Reduce model. In a Map-Reduce job, all map and reduce functions are executed in parallel way. All map (resp. reduce) functions are independently run by mapper (resp. reducer) node. The inputs and outputs of a Map-Reduce job are stored in an associated distributed file system that is accessible from any machine of the cluster nodes. The implementation of the Map-Reduce model is available in Hadoop[1]. Hadoop provides a distributed file system named Hadoop Distributed File System, (HDFS) that stores data on the nodes.

4 Parallel k-Prototypes Based on Map-Reduce

To offer for users the possibility to perform clustering of mixed type of large-scale data, we propose the parallel k-prototypes method based on the Map-Reduce model (PKP). As stated before, the most intensive calculation to occur in the k-prototypes method is the calculation of distances, which decreases its performance when dealing with large data sets. Indeed, the distance computations between one object with the cluster centers is independent to the distance computations between another object with the corresponding cluster centers. Thus, the distance computation between different objects and cluster centers can be executed in parallel. In each iteration, the new cluster centers, which are used in the next iteration, should be updated. Hence, this operation will be executed serially.

As shown in Fig. 2., the proposed method mainly consists into three functions: *map function* which performs the assignment of each data object to the nearest cluster, *combine function* which is based on calculating the local cluster centers, and *reduce function* which is devoted to compute the new cluster centers.

[1] http://hadoop.apache.org/

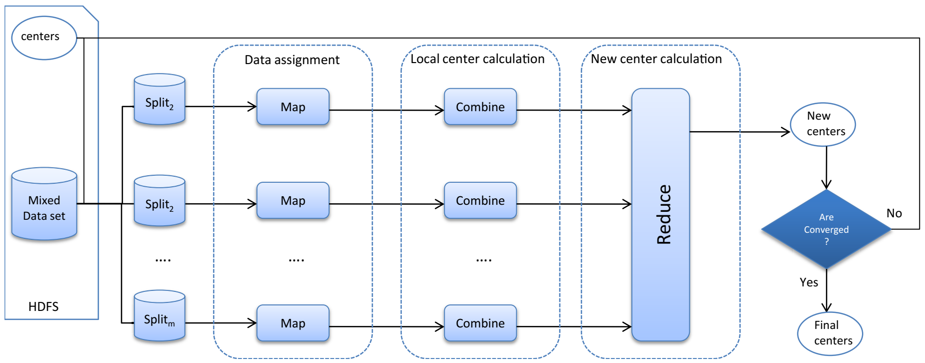

Fig. 2. Parallel k-prototypes method based on Map-Reduce

Suppose an input mixed data set stored in HDFS as sequence file of $< key/value >$ pairs, each of which represents a data object in the data set. The key is an index in bytes and the value represents the data object values. The input data set is composed by h blocks that are accessible from any machine of the cluster nodes. First, the PKP partitions input data set into m splits (i.e $Split_j \dots Split_m$) where m is a user-defined parameter and 1≤ j≤m. Each $Split_j$ is associated to map task j. We can mention that this partitioning process is performed sequentially means that the map task j corresponds to the $Split_j$ data chunk of h/m blocks. Then, the PKP creates a *centers* variable that contains the centers of the cluster, which is stored in HDFS.

4.1 Map Function

When each mapper receives the associated split, it first calculates the distance between each data object and the cluster centers using Equation 2. Given this information, the mapper then assigns the data object to its corresponding nearest cluster. Finally, the map function outputs the intermediate key and value pairs which are composed respectively of the nearest cluster index and the data object values.

Let Compute-Distance(x_i,$center_j$) be a function which returns the distance between data object x_i and $center_j$. The map function is outlined in Algorithm 1.2.

4.2 Combine Function

In this function, we combine the intermediate data produced from each map task, in order to reduce the amount of data transferred across to the reducers. To this end, we compute a partial information about the cluster centers denoted by the local cluster centers. First, we sum the values of the numerical attributes of the data objects assigned to the same cluster. Second, we compute the frequencies of different values of categorical attributes relative to the data objects for each

Algorithm 1.2. Map function

```
Data: $< key/value >$; where $key$: position and $value$: data object values, centers
Result: $< key_1/value_1 >$; where $key_1$: nearest cluster index and $value_1$: data object
        values
begin
    minDis$\leftarrow$ 0
    index$\leftarrow$ 0
    $x_i \leftarrow value$
    foreach $center_j \in centers$ do
        dis= Compute-Distance($x_i, center_j$)
    if $dis < minDis$ then
        minDis $\leftarrow$ dis
        index $\leftarrow$ j
    $key_1 \leftarrow index$
    $value_1 \leftarrow x_i$
end
```

cluster. Third, we record the number of data objects assigned in the same cluster. As each combine function finishes its processing, the intermediate data which consists into the local cluster centers, are forwarded to a single reduce task.

Let Dom[j]=$\{a_j^1, a_j^2, \ldots a_j^t\}$ be the domain of categorical attribute j. Let $Sum_i[j]$ be the sum of values of numeric attribute j for the data objects assigned in cluster i, $Freq_i[j, a_j^k]$ be the frequency of value a^k of categorical attribute j for the data objects assigned in cluster i, and Num_i be the number of data objects of cluster i. The combine function is outlined in Algorithm 1.3.

4.3 Reduce Function

In this function, the local cluster centers are merged in order to compute the new cluster centers. So, we first sum the numeric values and the total number of data objects assigned to the same cluster. Then, we compute the global frequencies for different values of categorical attributes relative to the data objects. Given these information, we can compute both the mean and mode value of the new center. Once the new centers are obtained, the PKP moves to the next iteration until the convergence. It is important to mention that the convergence is achieved when the cluster centers become stable for two consecutive iterations.

Let $Newcenter_i$ be the new center values of the cluster i and Highest-Freq(Freq[j,.]) be a function which returns the most common value (i.e mode) of the categorical attribute j from Freq[j,.] variable. The reduce function is outlined in Algorithm 1.4.

5 Experiments and Results

In this section, we evaluate the performance of PKP using speedup and scaleup measures [14]. We run the experiments on the Amazon Elastic MapReduce

Algorithm 1.3. Combine function

```
Data: $< key/V >$; where $key$: cluster index and V: list of data objects assigned in
      the same cluster
Result: $< key_1/value_1 >$; where $key_1$: cluster index and $value_1$: local cluster center
begin
    $Sum_{key} \leftarrow \emptyset$
    $Freq_{key} \leftarrow \emptyset$
    $Num_{key} \leftarrow 0$
    while $V.HasNext()$ do
        % Get a data object
        $x_i \leftarrow$ V.next()
        for $j \leftarrow 1 \dots m_r$ do
            $Sum_{key}[j] \leftarrow Sum_{key}[j] + x_{ij}$
        for $j \leftarrow m_r + 1 \dots m_c$ do
            for $k \leftarrow 1 \dots t$ do
                if $x_{ij} = a^k$ then
                    $Freq_{key}[j, a_j^k] \leftarrow Freq_{key}[j, a_j^k] + 1$
        $Num_{key} \leftarrow Num_{key} + 1$
    $key_1 \leftarrow key$
    $value_1 \leftarrow Sum_{key} \cup Freq_{key} \cup Num_{key}$
end
```

(Amazon EMR)[2] which is a web service for processing huge amounts of data by exploiting the parallelism on a cluster of machines, each of which has single core 2.6 GHz cores and 1 GB of memory.

For the following evaluations, we utilize three data sets generated from KDD CUP 1999[3] named Data set1, Data set2, Data set3, which are listed in Table 1.

Table 1. The description of data sets

Data set	**No. of data objects**	**No. of numerical attributes**	**No. of categorical attributes**	**No. of clusters**
Data set1	$4 * 10^6$	37	4	23
Data set2	$8 * 10^6$	37	4	23
Data set3	$16 * 10^6$	37	4	23

To evaluate the speedup, we keep the size of the data set constant and increase the number of the nodes in the system. Speedup given by the larger system with m nodes is defined as follows [14]:

$$Speedup(m) = \frac{T_1}{T_m}, \tag{3}$$

[2] http://aws.amazon.com/elasticmapreduce/
[3] http://kdd.ics.uci.edu/

Algorithm 1.4. Reduce function

```
Data: $< key/V >$; where $key$: cluster index and V: list of local cluster centers
Result: $< key_1/value_1 >$; where $key_1$: cluster index and $value_1$: new center values
begin
    $Sum_{key} \leftarrow \emptyset$
    $Freq_{key} \leftarrow \emptyset$
    $Num_{key} \leftarrow 0$
    $Newcenter_{key} \leftarrow 0$
    while $V.HasNext()$ do
        % Get a local cluster center
        $x_i \leftarrow$ V.next()
        for $j \leftarrow 1 \ldots m_r$ do
            $Sum_{key}[j] \leftarrow Sum_{key}[j] + x_i.Sum_{key}[j]$
        for $j \leftarrow m_r + 1 \ldots m_c$ do
            for $k \leftarrow 1 \ldots t$ do
                $Freq_{key}[j, a_j^k] \leftarrow Freq_{key}[j, a_j^k] + x_i.Freq_{key}[j, a_j^k]$
        $Num_{key} \leftarrow Num_{key} + x_i.Num_{key}$
    for $j \leftarrow 1 \ldots m_r$ do
        $Newcenter_{key}[j] \leftarrow Sum_{key}[j]/Num_{key}$
    for $j \leftarrow m_r + 1 \ldots m_c$ do
        $Newcenter_{key}[j] \leftarrow$ Highest-Freq($Freq_{key}$[j,.])
    $key_1 \leftarrow key$
    $value_1 \leftarrow Newcenter_{key}$
end
```

Where T_1 is the execution time on one node and T_m is the execution time on m nodes. The efficient parallel method gives linear speedup. A system with m times the number of nodes yields a speedup of m. Meanwhile, linear speedup is difficult to achieve since the cost of network communication increases with the number of nodes becomes large. We perform the speedup evaluation on the three data sets. The number of nodes varied from 1 to 4. Fig. 3.(a) shows the speedup results. As the size of the data set increases, the speedup of PKP becomes approximately linear. From Fig. 3.(a), we can conclude that the PKP handles large amount of mixed types data efficiently.

Scaleup is defined as the ability of a m-times larger system to perform a m-times larger job in the same execution time [14].

$$Scaleup(DS, m) = \frac{T_{DS_1}}{T_{DS_m}}, \tag{4}$$

Where DS is the data set, T_{DS_1} is the execution time for DS on one node and T_{DS_m} is the execution time for m*DS on m nodes. To show how well PKP can treat large-scale of mixed data sets when more nodes are available, we have realized scaleup experiments where we increase the size of the data sets in direct proportion to the number of machines in the cluster nodes. Roughly speaking, the data sets' size of $4*10^6$, $8*10^6$ and $16*10^6$ instances are executed on 1,2 and 4 machines respectively. Fig. 3.(b) shows the performance results of the data sets. Apparently, the PKP scales well when dealing with large-scale of data.

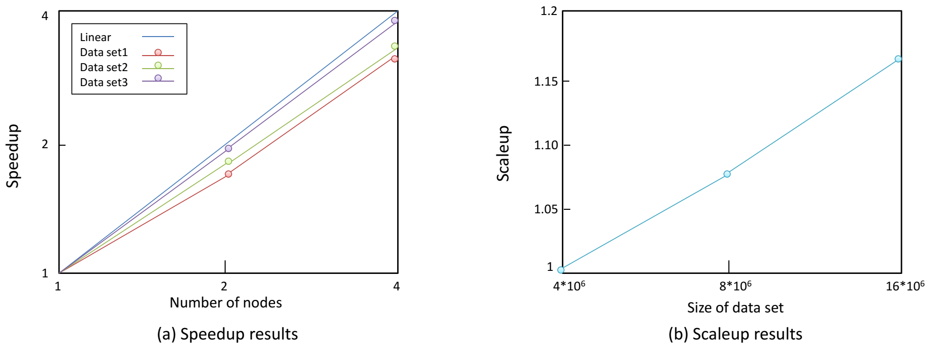

Fig. 3. Experimental evaluation

6 Conclusion

Since traditional clustering methods can not deal with Big data, there is a need for scalable solutions. Furthermore, Big data are often characterized by the mixed type of attributes such as numerical and categorical. This paper have investigated the parallelization of k-prototypes method using Map-Reduce model to cluster large-scale of mixed data. We have used speedup and scaleup measures to evaluate the performances of the proposed method. Experimental results on large data sets show the performance of our method. A proper initialization of k-prototypes method is crucial for obtaining a good final solution. Thus, it should be interesting to parallelize the initialization step of k-prototypes using Map-reduce model, in order to create an initial set of cluster centers that is probably close to the optimum solution.

References

1. Bahmani, B., Moseley, B., Vattani, A., Kumar, R., Vassilvitskii, S.: Scalable k-means++. Proceedings of the VLDB Endowment **5**(7), 622–633 (2012)
2. Cui, X., Zhu, P., Yang, X., Li, K., Ji, C.: Optimized big data K-means clustering using MapReduce. The Journal of Supercomputing **70**(3), 1249–1259 (2014)
3. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Communications of the ACM **51**(1), 107–113 (2008)
4. Gorodetsky, V.: Big data: opportunities, challenges and solutions. In: Ermolayev, V., Mayr, H.C., Nikitchenko, M., Spivakovsky, A., Zholtkevych, G. (eds.) ICTERI 2014. CCIS, vol. 469, pp. 3–22. Springer, Heidelberg (2014)

5. Ji, J., Bai, T., Zhou, C., Ma, C., Wang, Z.: An improved k-prototypes clustering algorithm for mixed numeric and categorical data. Neurocomputing **120**, 590–596 (2013)
6. Hadian, A., Shahrivari, S.: High performance parallel k-means clustering for disk-resident datasets on multi-core CPUs. The Journal of Supercomputing **69**(2), 845–863 (2014)
7. Huang, Z.: Clustering large data sets with mixed numeric and categorical values. In: Proceedings of the 1st Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 21–34 (1997)
8. Kim, Y., Shim, K., Kim, M.S., Lee, J.S.: DBCURE-MR: An efficient density-based clustering algorithm for large data using MapReduce. Information Systems **42**, 15–35 (2014)
9. Li, C., Biswas, G.: Unsupervised learning with mixed numeric and nominal data. Knowledge and Data Engineering **14**(4), 673–690 (2002)
10. Li, Q., Wang, P., Wang, W., Hu, H., Li, Z., Li, J.: An efficient K-means clustering algorithm on mapreduce. In: Bhowmick, S.S., Dyreson, C.E., Jensen, C.S., Lee, M.L., Muliantara, A., Thalheim, B. (eds.) DASFAA 2014, Part I. LNCS, vol. 8421, pp. 357–371. Springer, Heidelberg (2014)
11. Ludwig, S.A.: MapReduce-based fuzzy c-means clustering algorithm: implementation and scalability. International Journal of Machine Learning and Cybernetics, 1–12 (2015)
12. MacQueen, J.: Some methods for classification and analysis of multivariate observations. Proceedings of the fifth Berkeley symposium on mathematical statistics and probability **14**(1), 281–297 (1967)
13. Vattani, A.: K-means requires exponentially many iterations even in the plane. Discrete Computational Geometry **45**(4), 596–616 (2011)
14. Xu, X., Jger, J., Kriegel, H.P.: A fast parallel clustering algorithm for large spatial databases. High Performance Data Mining, 263–290 (2002)
15. Zhao, W., Ma, H., He, Q.: Parallel k-means clustering based on mapreduce. Cloud Computing, 674–679 (2009)