IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 7, Issue 2, February 2018

A Non-Overlap based Cluster using Canopy and Parallel K-Means for MapReduce Framework

Summia Parveen[1], Amirjohn BeeBee[2], Thoufiya. M[3], Safana Jasmine[4]

Assistant Professor, Computer Science and Engineering, Dhaanish Ahmed Institute of Technology, Coimbatore, India[1]

Students, Computer Science and Engineering, Dhaanish Ahmed Institute of Technology, Coimbatore, India[2,3,4]

Abstract: There are very big bottlenecks when traditional data mining algorithms deal with large data sets. A novel technique for clustering the large and high dimensional datasets. The main idea is to use an inexpensive and approximate distance measure in order to efficiently partition the data into overlapping subsets which is called as canopies. After we get these canopies the desired clustering is performed by measuring exact distances only between points that occur in a common canopy. Using canopies, large clustering problems that were formerly impossible become practical and efficient. K-Means is typical distance-based clustering algorithm. Here, the canopy clustering algorithm is implemented as an efficient clustering technique by means of knowledge integration. With the study of the canopy clustering the K-Means paradigm of computing, we find is appropriate for the implementation of a clustering algorithm. This paper shows some advantages of canopy cluster to K-Means clustering mechanism and proposes a pre-clustering approach to K-Means Clustering method. Here we use Hadoop's MapReduce program model for K-Means clustering with canopy clustering. The experimental results show that Canopy + K-means algorithm has faster operation speed than K-means algorithm, but both of them show good speed-up ratio under Hadoop environment and Canopy + K-means algorithm is even much better K-means algorithm.

Keywords: Abnormal Event Detection, Outlier Detection, Video Data Stream, Sparse Learning, Dynamic Detection.

I. INTRODUCTION

Various kinds of data come forth with development of information technology and information society. Massive data accumulates in database and data library and it is beyond human's ability to understand it. Data mining technology arises to transfer that understandable to human. Technically, data mining refers to the process to draw out implicit but potentially useful information and knowledge that is unknown to human out of massive, incomplete, noisy, fuzzy, random and seemingly disordered data. Cluster analysis.

Hadoop platform of Apache realizes MapReduce model through Java. Being part of Hadoop, Mahout is a machine learning and data mining library which uses MapReduce program model to realize numbers of algorithms with fine expansion capacity. In reference ([13, 25], research is done concerning Canopy clustering and K-means clustering's MapReduce parallelization respectively. K-means and Canopy have advantages and disadvantages in different manners. Based on this, we here combine them and do clustering case study based on Mahout.

Data clustering is a data mining technique in which the data points are grouped into clusters such that the distance between points in a same cluster is very less and distance between a point in one cluster and another cluster is maximum. Traditional data mining has many clustering algorithms like partitioning methods, hierarchical methods, density-based techniques, etc. Many researchers are involved in parallelizing these clustering algorithms to deal with big data estimation strategy.

MapReduce is a programming model that comes with Hadoop environment. It is a parallel programming model which can be used to deal with large data sets. The MapReduce includes two phases namely, Map and Reduce phases. Users have to write programs as Map and reduce functions and the hadoop environment takes care of parallel execution although the various table text styles are provided. K-means is one of the top ten algorithms in data mining and it is a typical clustering algorithm based on distance. There are a lot of advantages in K-means, for example, it is easier to describe, and it also has a higher efficiency of the time and suitable for large-scale data processing. However, with the deep-going research and the scale of the data is bigger and bigger, K-means gradually exposed some shortcomings, like time complexity of the Serial computing is high and have limitations on its processing capacity.

Hadoop is an open source software. It provides an infrastructure of cloud computing platform which includes HDFS and MapReduce computing frame. Hadoop is current standard platform in cloud computing study and application. Many companies paid attention to it due to its feature of open source, serious computational power and huge storage space.

DOI 10.17148/IJARCCE.2018.7243

II. RELATED WORK

Hadoop is the latest hot cloud computing framework. The core design of Hadoop is HDFS and MapReduce. HDFS provides storage for massive amounts of data and MapReduce provides calculation. Hadoop distributed file system, short for HDFS. It employs a Master/Slave architecture to manage the file system. One HDFS cluster is composed of one Name Node and more Data Node. Name Node implement the unified management to the whole file system. Data Node is used for storing data information. MapReduce is a programming model for massive data processing and operation. The central idea is to decompose large jobs into many small jobs and gather them after processed. It mainly includes two operations during processing data in parallel: Map and Reduce, means mapping and statute.

With the development of information technology, data volumes processed by many applications will routinely cross the peta-scale threshold, which would in turn increase the computational requirements. Efficient parallel clustering algorithms and implementation techniques are the key to meeting the scalability and performance requirements entailed in such scientific data analyses. So far, several researchers have proposed some parallel clustering algorithms. All these parallel clustering algorithms have the following drawbacks: a) They assume that all objects can reside in main memory at the same time; b) Their parallel systems have provided restricted programming models and used the restrictions to parallelize the computation automatically. Both assumptions are prohibitive for very large datasets with millions of objects. Therefore, dataset oriented parallel clustering algorithms should be developed.

K-means algorithm is one of the most classic, simple and popular method of clustering algorithm. The basic idea is that set K points in the space as the center of the region to cluster, classify the other points to the closest center point. Then, gradually update the value of cluster centers by an iterative method until get a satisfactory result. The principle of K-means algorithm is simple, the bulk of the computing in it comes from the distance calculation between data point and cluster centers and iterative process of new cluster centers. The distance calculation in the algorithm is based on vector. K-means also has some shortcomings, like there is no standard to choose the number K (the number of clustering). Besides, it is difficult to decide the cluster centers which may directly affects the final result.

Though the k-means algorithm is simple and efficient, it starts degrading when number of clusters and data size increases. To avoid this multi restarting k-means can be used. But to work on this more and more staring point are needed which again leads to a bottleneck when large data is used. Many variants have been proposed to improve the performance of k-means algorithm. The methods that are proposed for incremental dataset are noteworthy. Clusters are generated incrementally in these methods. Global k-means and a fast version of global k-means are proposed which arrive at global minimize of a cluster. Along with this a modified version of global k-means is proposed. There are two types of modifications available. Lai et al proposed a fast version of global k-means algorithm which was intended to reduce time complexity. The modification that has been proposed by Adil et al is aimed at reducing the space complexity, i.e. usage of lesser memory.

These variants of K-means are successful in getting better results than traditional algorithm. The enhancement to this K-means algorithm can also be done by some techniques to find out initial centroids. An intelligent method of selecting centroids. First centroid will be selecting by averaging on the data points. The remaining centroids are calculated by finding the points which are far from first calculated centroid. The experimental results in this paper showed better accuracy.

Another existing enhancement to K-means algorithm where the execution is divided into two phases. The output of first phase will form the initial centroids. The input data array is divided into smaller arrays. These sub-arrays represent clusters. In second phase, the cluster sizes vary and after iterative computations final clusters are formed. The basic K-means algorithm has been implemented in MapReduce manner. Here finding distances is in Map phase and combining results in Reduce phase. But again, the traditional method of random selection of initial centroids is adopted in all implementation. Hence in this work we analyse that the proposed method of selecting initial centroids will give a better outcome.

Due to the rich information conveyed by the membership grade matrix, fuzzy clustering has long been widely used in many real-world application domains where well-separated clusters are typically not available. Applications requiring fuzzy clustering abound, ranging from deriving taxonomy for a collection of ambiguous news articles to analyse gigantic Facebook logs and patient stratification. However, with the explosive growth of online big data in recent years, researchers and practitioners come to realize that a single fuzzy clustering might fail with complex data, such as high dimensional text corpora. Hence, the introduction of consensus clustering to fuzzy clustering becomes natural, and fuzzy consensus clustering (FCC) thus emerges as a new research frontier.

MapReduce for big data clustering. The existing parallel K-means clustering using MapReduce and gave a detailed description for the algorithm, give the first analysis that shows several partitional clustering algorithms in MapReduce. However, not all big data processing problems can be efficient by parallelism; a research shows that partitional clustering algorithm requires exponentially many iterations. Meanwhile, job exponential creation time and time of big data shuffling are hard to swallow especially when data size is huge, so just parallelism is not enough, only by eliminating the partitional clustering algorithms dependence on the iteration can we implement high performance.

DOI 10.17148/IJARCCE.2018.7243

III. PROPOSED APPROACH

The parallelization work of K-means is to work out whether a node belongs to a certain cluster center. Hadoop cluster divides the sample data into parts and stores them in HDFS. Every map task will operate the local data block. The main work of map task is taking out one data node and reading current cluster centers from HDFS, then output identifier of the cluster center through calculation. The combine function integrates the intermediate key-value pair. At last reduce task outputs the result into file system HDFS. Each time K-means completes computing the new cluster centers, reduce operation would send these new centers to distributed file system cache and updates them after each iteration.

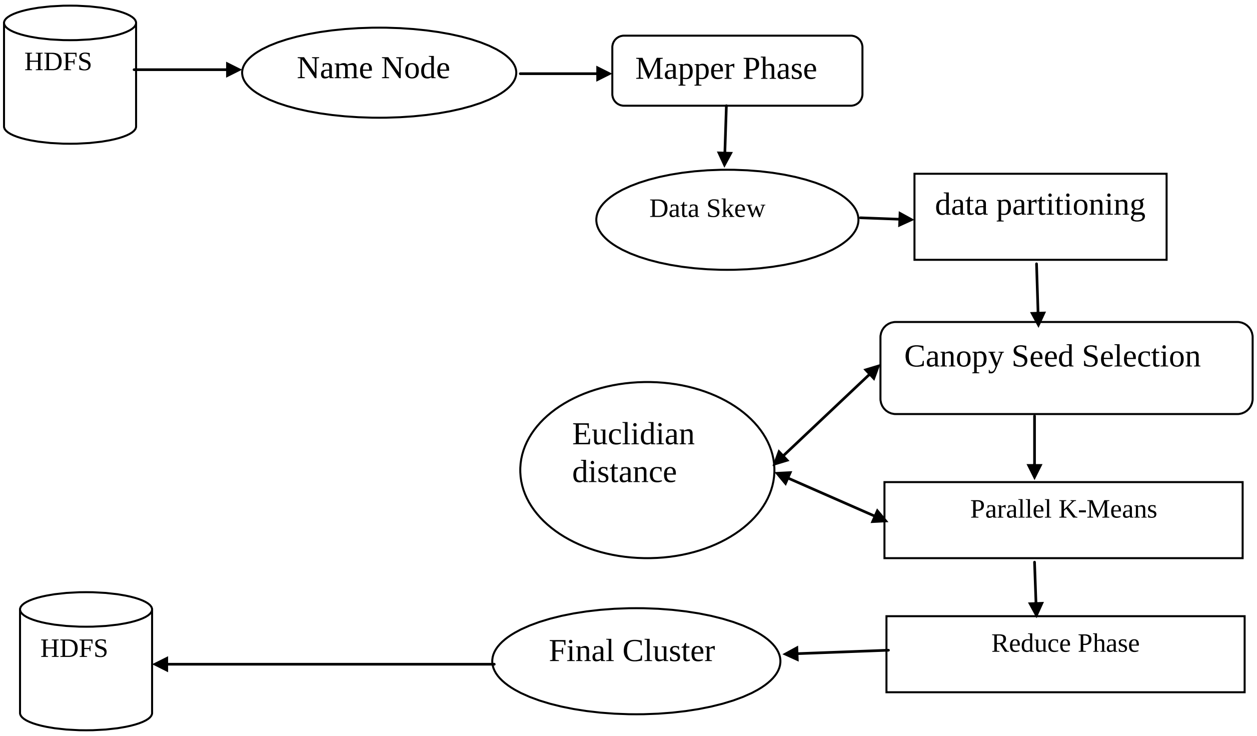

Fig. 1 Proposed architecture diagram

A. Data Skew

In a MapReduce application, the outputs of map tasks are distributed among reduce tasks via hash partitioning (by default). In the map phase, the hash partitioning usually takes a hash function hash key %R to determine the partition number corresponding to each type of key-value pair, where R is the number of reduce tasks. The hash function is usually adequate to evenly distribute the data. However, if the outputs are not evenly distributed, hash partitioning may fail with skewed data. This phenomenon is referred to as partitioning skew. For example, in the Inverted Index application, the hash function may partition intermediate data based on the first letter of a word; reducers processing more popular letters are assigned a disproportionate amount of data. Partitioning skew can occur for the following reasons

- Skewed tuple sizes: The sizes of values in applications vary significantly, which can lead to uneven workload distribution.
- Skewed key frequencies: Some keys occur more frequently in intermediate data, causing reduce tasks that process these popular keys to become overloaded.
- Skewed execution times: Processing a single, large key-value pair may require more time than processing multiple small pairs. Even when the partitioning function perfectly distributes keys across reducers, the execution times of reduce tasks may differ simply because the key groups they are assigned contain significantly more values.

Map Phase: After all map tasks are completed, all key-value pairs are sorted by partition number. Inside the partition, all key-value pairs are sorted following the key order. When dealing with large-scale datasets, the output data generated by each map task usually occupy a large amount of memory, which is spilled to the local disk. All spilled files are then merged and written to the disk after all map tasks are completed. Throughout the process of spilling and merging, the index corresponding to each partition is established by the map tasks. When reading data, it can speed up the task of obtaining subsequent data for the reduce partitions.

DOI 10.17148/IJARCCE.2018.7243

- Once all map tasks are complete, the output is written to the local disk. The Task Tracker uses heartbeat information to send messages to a Job Tracker stating that the task has been completed.
- The Job Tracker maintains a map task completion message queue for each MapReduce job. When the Task Tracker runs a reduce task asks for a completion message for the map task, the Job Tracker removes the message from the queue and delivers it to the corresponding Task Tracker.
- In the same MapReduce job, a reduce task gets a completion message for the map task from its Task-Tracker. The runtime information of the map task is extracted from the completion message, including map task number, and information concerning execution nodes. Using this information, the reduce task establishes an HTTP connection with the execution node and requests the metadata information output of the map task.
- Based on the request number of a map task, the Task Tracker reads the corresponding index file of the map outputs from the local file system and sends it to the corresponding reduce task.
- The reduce task merges the virtual partitions of the same index number from different index files. It then aggregates the data of each virtual partition that has the same type of key-value pairs.

Repartitioning: The repartitioning process divides the collected virtual partitions into new partitions of the same number as reduce tasks. The data size of the biggest partition can be minimized after repartitioning process. It can also reduce the processing time needed for the maximum partition, thereby speeding up the completion of the entire reduce phase and increasing the rate of completed jobs as well as system throughput

B. Canopy seed point selection

It is not hard for traditional cluster algorithm to solve general application problems (basically small data), but it becomes difficult when it comes to large data. Canopy is an algorithm developed from traditional cluster algorithm. Canopy algorithm divides cluster into two steps. Step one: divide data into overlapped subset called canopy by using a simple and quick distance calculation method; step two: calculate distance of all data vectors of the same canopy in step one by suing an accurate and strict distance calculation method. This differs from previous means of cluster in its using of two distance calculation methods. Meanwhile, it reduces calculation as it only needs to calculate data vector of the overlapped part. The advantage of Canopy lies in its speed of obtaining cluster, as it only needs to traverse data once to get the result. This, however, is also its disadvantage. The algorithm cannot give accurate cluster result, but optimum cluster quantity. It doesn't need to specify cluster K in advance like K-means.

The algorithm uses a fast distance measure and two distance thresholds (T_1 and T_2, $T_1 > T_2$). It starts from a cluster including several points and an empty Canopy list and then iterates the data, in which process generates Canopy. In each round of iteration, it moves a point away from the cluster and add Canopy which takes this point as its center to the list. Then the rest points in the cluster are traversed. For each point, it can calculate distance between the point and center of each Canopy. If the distance is smaller than T_1, it would be added into this Canopy. If the distance is smaller than T_2, then the point would be moved out of the cluster, so new Canopy will not be built in following circulations. Repeat the above process till the cluster becomes empty.

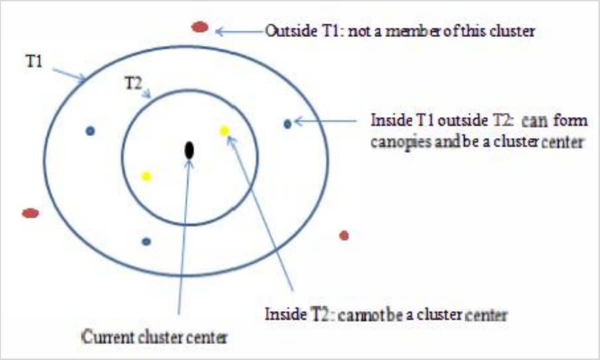
Outside T1: not a member of this cluster
T1
T2
Inside T1 outside T2: can form canopies and be a cluster center
Inside T2: cannot be a cluster center
Current cluster center

Fig.1 Canopy Clustering

The parallel design of Canopy algorithm is divided into two processes, the map process and the Reduce process. The main task of the map process is to deal with the data objects in this node according to the idea of Canopy algorithm. For each map node need to enter the threshold of $T_1$and T_2, the formation of canopy center set. Because the canopy center set generated by the map process is local. The number of data objects on which the canopy center set is much smaller than the number of data objects on the map node. Therefore, the number of canopy centers on all map nodes is also much smaller than the number of original data sets. Create distance matrix from a point X_j to each of the cluster centers considering the Euclidean distance between the point and the cluster center using the formula:

$$d_{ij} = \sqrt{\sum \left(X_j - C_j\right)^2} \quad (1)$$

d_{ij} = Euclidian distance between the Jh data point and the ith cluster center. The membership matrix is created using

$$\mu_i(x_j) = \frac{\left(\frac{1}{d_{ij}}\right)^{\frac{1}{m-1}}}{\sum_{k=1}^{p}\left(\frac{1}{d_{kj}}\right)^{\frac{1}{m-1}}} \quad (2)$$

Algorithm of Map:

1. The centroid points obtained from Canopy generation is considered as key and vector point as value.
2. Calculate the Euclidian distance between centroid point and the data point using (1).
3. Compute the membership value of each data point using (2).
4. Create the membership matrix.
5. Clusters are generated using the nearest centroid and the data points assigned to that particular cluster.
6. Maintains the detail about which data point is in which cluster

Algorithm of Reduce:

1. Recalculates the centroid for each cluster.

The threshold value of the reduce process is slightly larger than that of the map process. In order to reduce the number of reduce process, only need a reduce node pair from all map, the node to get the canopy center of the collection of the merger

C. Parallel K-Means Cluster

Parallel K-means algorithm is the most well-known and commonly used clustering method. It takes the input parameter, k, and partitions a set of n objects into k clusters so that the resulting intra-cluster similarity is high whereas the inter cluster similarity is low. Cluster similarity is measured according to the mean value of the objects in the cluster, which can be regarded as the cluster's "center of gravity". The algorithm proceeds as follows: Firstly, it randomly selects k objects from the whole objects which represent initial cluster centers. Each remaining object is assigned to the cluster to which it is the most similar, based on the distance between the object and the cluster center. The new mean for each cluster is then calculated. This process iterates until the criterion function converges.

In k-means algorithm, the most intensive calculation to occur is the calculation of distances. In each iteration, it would require a total of (nk) distance computations where n is the number of objects and k is the number of clusters being created. It is obviously that the distance computations between one object with the centers is irrelevant to the distance computations between other objects with the corresponding centers. Therefore, distance computations between different objects with centers can be parallel executed. In each iteration, the new centers, which are used in the next iteration, should be updated. Hence the iterative procedures must be executed serially.

PK-Means algorithm needs one kind of MapReduce job. The map function performs the procedure of assigning each sample to the closest center while the reduce function performs the procedure of updating the new centers. In order to decrease the cost of network communication, a combiner function is developed to deal with partial combination of the intermediate values with the same key within the same map task.

The steps for Map function are as follows.
Step-1: Split the data into chunks
Step-2: select randomly generated initial centroids
Step-3: Assign each data point to a cluster center by calculating minimum distance.
Step-4: Generate <key, value> pairs, where key is the instance number and value are the cluster to which is initially assigned.

The reducer function has following steps.
Step-1: collect mapped data from mappers
Step-2: compute average vector in each cluster
Step-3: update the cluster center with this new average value.
Step-4: repeat the procedure until no more changes are possible.
Step-5: finally generate the <kev, value> pairs where key is instance number and value is cluster to which it is finally assigned.

Map-function the input dataset is stored on HDFS as a sequence file of <key, value> pairs, each of which represents a record in the dataset. The key is the offset in bytes of this record to the start point of the data file, and the value is a string of the content of this record. The dataset is split and globally broadcast to all mappers. Consequently, the distance computations are parallel executed. For each map task, PK-Means construct a global variant centers which is an array containing the information about centers of the clusters. Given the information, a mapper can compute the closest center point for each sample. The intermediate values are then composed of two parts: the index of the closest center point and the sample information.

DOI 10.17148/IJARCCE.2018.7243

Combine-function. After each map task, we apply a combiner to combine the intermediate data of the same map task. Since the intermediate data is stored in local disk of the host, the procedure cannot consume the communication cost. In the combine function, we partially sum the values of the points assigned to the same cluster. In order to calculate the mean value of the objects for each cluster, we should record the number of samples in the same cluster in the same map task.

Reduce-function. The input of the reduce function is the data obtained from the combine function of each host. As described in the combine function, the data includes partial sum of the samples in the same cluster and the sample number. In reduce function, we can sum all the samples and compute the total number of samples assigned to the same cluster. Therefore, we can get the new centers which are used for next iteration.

IV. EXPERIMENTAL SETUP

The evaluate the performance of our proposed algorithm with respect to speedup, scaleup and size up. Performance experiments were run on a cluster of computers, each of which has two 2.8 GHz cores and 4GB of memory. Hadoop version 1.2.1 and Java 1.7 are used as the MapReduce system for all experiments.

TABLE I COMPARISON OF RUNTIME OF TWO ALGORITHMS UNDER NO OF CLUSTER

Algorithm	**No of cluster**					
	1	2	3	4	5	6
Fuzzy Consensus Clustering	43	67	86	90	121	137
Canopy+PK-Means	23	34	45	76	83	95

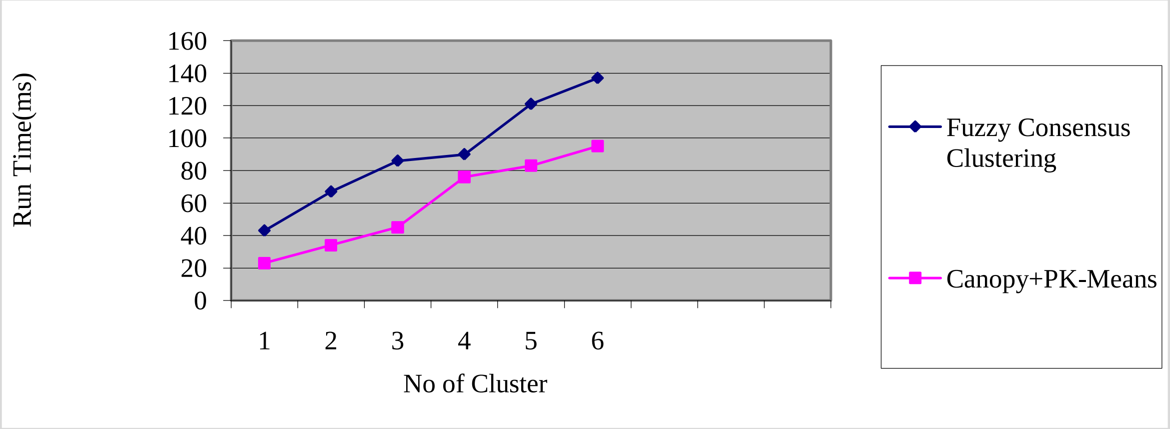

Fig. 2 Comparison of map/reduce runtime

TABLE II VALUE SILHOUETTE COEFFICIENT FOR FUZZY CONSENSUS CLUSTERING AND CANOPY+P K-MEANS

Algorithm	**Silhouette coefficient**				
	0.2	0.4	0.6	0.8	1
Fuzzy Consensus Clustering	0.34	0.45	0.51	0.64	0.74
Canopy+PK-Means	0.46	0.57	0.69	0.75	0.89

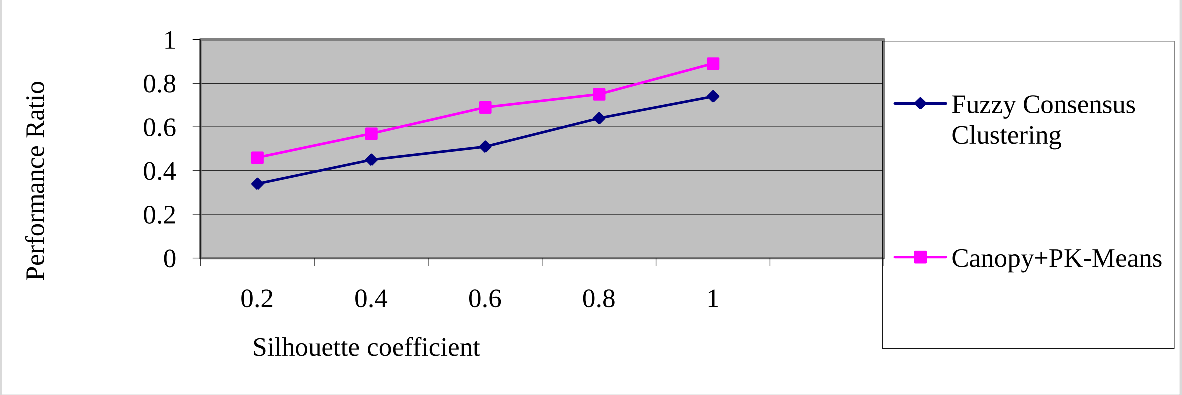

Fig. 3 value of silhouette coefficient or traditional Fuzzy Consensus Clustering and canopy+parallel k-means

DOI 10.17148/IJARCCE.2018.7243

The performed the speedup evaluation on datasets with different sizes and systems. The number of computers varied from 1 to 4. The size of the dataset increases from 1GB to 4GB. The speedup for different datasets. As the result shows, canopy-based PK-Means has a very good speedup performance. Specifically, as the size of the dataset increases, the speedup performs better.

To measure the speedup, we kept the dataset constant and increase the number of computers in the system. The perfect parallel algorithm demonstrates linear speedup: a system with m times the number of computers yields a speedup of m. However, linear speedup is difficult to achieve because the communication cost increases with the number of clusters becomes large.

V. Conclusion

The partition the data into overlapping subsets which is called as canopies. After we get these canopies the desired clustering is performed by measuring exact distances only between points that occur in a common canopy. Using canopies, large clustering problems that were formerly impossible become practical and efficient. K-Means is typical distance-based clustering algorithm. Here, the canopy clustering algorithm is implemented as an efficient clustering technique by means of knowledge integration. With the study of the canopy clustering the K-Means paradigm of computing, we find is appropriate for the implementation of a clustering algorithm. This paper shows some advantages of canopy cluster to K-Means clustering mechanism and proposes a pre-clustering approach to K-Means Clustering method. Here we use Hadoop's MapReduce program model for K-Means clustering with canopy clustering. The experimental results show that Canopy + K-means algorithm has faster operation speed than K-means algorithm, but both of them show good speed-up ratio under Hadoop environment and Canopy + K-means algorithm is even much better K-means algorithm

References

[1] Xindong Wu, Xingquan Zhu, Gong-Qing Wu, and Wei Ding, "Data mining with big data", IEEE Transactions on Knowledge and Data Engineering, vol 26 (1), 2014, pp-97-107.

[2] Rasmussen, E.M., Willett, P.: Efficiency of Hierarchical Agglomerative Clustering Using the ICL Distributed Array Processor. Journal of Documentation 45(1), 1989, 1-4.

[3] A.M.Bagirov, "Modified global k-means algorithm for sum-of-squares clustering problem", Elsevier-Pattern Recognition, vol.41, 2008, pp. 3192–3199.

[4] A. Likas, M. Vlassis, J. Verbeek, "The global k-means clustering algorithm", Elsevier-Pattern Recognition, vol.36, 2003, pp.451–461.

[5] A.M. Bagirov, K. Mardaneh, "Modified global k-means algorithm for clustering in gene expression datasets", Proceedings of the AI Workshop on Intelligent Systems of Bioinformatics, Hobart, Australia, 2006, pp. 23–28.

[6] J.Z.C. Lai, T.J. Huang, "Fast global k-means clustering using cluster membership and inequality", Elsevier - Pattern Recognition, vol 43 (3), 2010, pp.731–737.

[7] Adil M. Bagirov, Julien.Ugon, Dean. Webb, "Fast modified global kmeans algorithm for incremental cluster construction", Elsevier-Pattern Recognition, vol.44, 2011, pp.866-876.

[8] Anand M. Baswade1, Prakash S. Nalwade," Selection of Initial Centroids for k-Means Algorithm", International journal of Computer Science and Mobile Computing (IJCSMC), Vol. 2 (7), 2013, pp.161 – 164.

[9] Azhar Rauf, Sheeba, Saeed Mahfooz, Shah Khusro and Huma Javed,, "Enhanced K-Mean Clustering Algorithm to Reduce Number of Iterations and Time Complexity", Middle-East Journal of Scientific Research, vol.12 (7), 2012, pp.959-963.

[10] Weighing Zhao, Huifang Ma and Qing He1, "Parallel K-Means Clustering Based on MapReduce", Springer-Verlag Berlin Heidelberg, 2009, pp.674-679.

[11] S. Aranganayagi, K. Thangavel, "Clustering Categorical Data using Sihouette Coefficient as a Relocating Measure", Proceedings of International Conference on Computational Intelligence and Multimedia Applications, Sivakasi, India, 2007, pp.13-17.

DOI 10.17148/IJARCCE.2018.7243