Map/Reduce Affinity Propagation Clustering Algorithm

Wei-Chih Hung, Chun-Yen Chu, and Yi-Leh Wu
Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan
Email: {d9715010, m10015054}@mail.ntust.edu.tw, ywu@csie.ntust.edu.tw

Cheng-Yuan Tang
Department of Information Management, Huafan University, New Taipei City, Taiwan
Email: cytang@cc.hfu.edu.tw

***Abstract*—The Affinity Propagation (AP) is a clustering algorithm that does not require pre-set K cluster numbers. We improve the original AP to Map/Reduce Affinity Propagation (MRAP) implemented in Hadoop, a distribute cloud environment. The architecture of MRAP is divided to multiple mappers and one reducer in Hadoop. In the experiments, we compare the clustering result of the proposed MRAP with the K-means method. The experiment results support that the proposed MRAP method has good performance in terms of accuracy and Davies–Bouldin index value. Also, by applying the proposed MRAP method can reduce the number of iterations before convergence for the K-means method irrespective to the data dimensions.**

***Index Terms*—affinity propagation, map/reduce, hadoop, K-means, clustering algorithm**

I. Introduction

With the rapid speed of internet development, people get more focus on the Big Data issue. Big Data is a collection of data set so large that it becomes difficult to analyze. One of the common methods that help analyzing data is cluster analysis. The Affinity Propagation (AP) method [1], [2] is a clustering algorithm requires no pre-set number of clusters K. The AP method simultaneously considers all data points as the potential centers. The similarity $s(i,k)$ in AP indicates how well the data point with index k is suited to be the center for data point i. The AP finds the centers by passing two kinds of messages: "availability" and "responsibility". The "availability" $a(i,k)$ which is passed between any two of data points shows the accumulated evidence for how appropriate it would be for point k to be chosen as point i's center. The "responsibility" $r(i,k)$ which is passed between any two of data points shows the accumulated evidence for how well-suited the point *i* to be served as the point *k*. Fig. 1 shows how the availability and the responsibility work among data points.

Based on the original AP, there are many improvements to make AP more efficient. Wang *et al*. [3] extended the single-exemplar model to a multi-exemplar one to create a new Multi-Exemplar Affinity Propagation (MEAP) algorithm which can determine the number of exemplars in each cluster automatically. He *et al.* [4] presented a method which is called "Adaptive Affinity Propagation" to search the range of "preference" that AP needs then find a suitable value which can optimize the result of AP. The "preference" is important in AP to decide the result is good or not. A more suitable preference value makes the clusters assignment more reasonable.

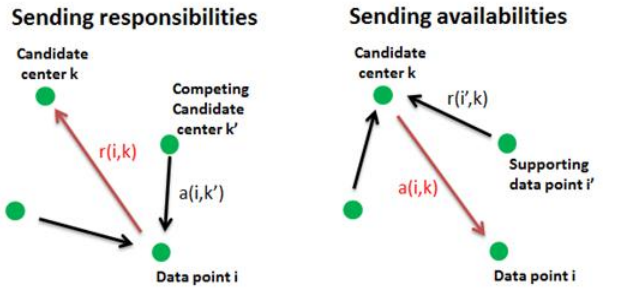

Figure 1. The messages ("responsibility" r(i,k) and "availability" (i,k)) passing between any two points.

To adapt the Map/Reduced model, the proposed system of this work is built on top of the Apache Hadoop. The Apache Hadoop is an open-source software framework of the Map/Reduce model that provides an inexpensive way for processing big data to be seamlessly handled through multiple distributed computers [5]. The Map/Reduce applications are divided into many small fragments (every fragment is called mapper) to work then merge the results of all the small fragments in the reducer. Because the Map/Reduce model is parallel and distributed computing model, it is appropriate for the implementation of scalable clustering algorithms [6]. There are many differences need to be noticed between the Map/Reduce and the serial programming models to promise the Map/Reduce model can handle the reliability and data motion well [7]-[9].

In this work, we propose to improve the original Affinity Propagation (AP) method for cluster analysis to the Map/Reduce Affinity Propagation (MRAP) method in

Manuscript received April 16, 2014; revised July 17, 2014.

doi: 10.12720/ijeee.3.4.311-317

Hadoop. The proposed MRAP method is a clustering algorithm for Map/Reduce model and it is scalable on multiple nodes for processing big data.

The main contributions of the proposed MRAP method are:

1. The MRAP method allows partitioning a single cluster analysis job into many small pieces on distributer processing nodes and to make the processing of big data more efficient and more scalable.
2. The MRAP method is self-regulating and thus requires no a priori knowledge of the characteristics of the input data. Hence the MRAP method is suitable for the Big Data environment.
3. The MRAP method can be the initialization of clustering algorithms that require a priori parameter settings such as the Map/Reduce K-means method.

The rest of this paper is organized as follows: In Section 2, we discuss the details of AP method and the proposed MRAP method and other algorithms that are used in experiments. After implementing the MRAP method, we test the result with running time, accuracy, and take the result of MRAP as the pre-cluster for K-means in Section 3. The results show that we get higher accuracy and less processing time than other comparing clustering algorithms in the Apache Mahout. In the end, we conclude this work and discuss the future work in Section 4.

II. Algorithm Design

The AP is the kernel of our proposed algorithm. The AP is a clustering algorithm without the pre-known cluster number and the usual clustering algorithms are quite sensitive with selected initial centers. The AP classifies data by communicating data property between data points. Base on Affinity Propagation, we improve it to design MRAP in the Map/Reduce environment. We divide the original AP into two stages: the mapper stage and the reducer stage [10] for multiple computers. The architecture helps the AP works more efficiently and can handle larger dataset.

In this section, we first introduce how the AP works and discuss how to find a better preference value that the AP required. Second, we discuss how to improve the AP in the Map/Reduce environment and what we try to change in the architecture.

A. Affinity Propagation

In the AP, all points are considered as potential centers. By viewing each data point as a node in a network, the AP recursively transmits real-valued messages along edges of the network until a good set of centers and corresponding clusters emerged.

First, similarity $s(i,k)$ shows how the data point with index *k* is appropriate to be the center for data point *i*. The similarity is set to a negative squared error (Euclidean distance): For data points x_i and x_k:

$$s(i,k) = -\|x_i - x_k\|^2 \quad (1)$$

Rather than requiring the number of clusters pre-specified, AP takes as input a real number $s(k,k)$ for each data point k that are more likely to choose itself as the cluster center. These values $s(k,k)$ are called "preferences". Because all data points are equally suitable as center, the preferences should be set a common value. This common value can be changeable for different number of clusters. From [1], the shared value could be the median of the input similarities (resulting in a moderate number of clusters) or their minimum (resulting in a small number of clusters). Besides employing the method in [1] to decide the preference values, we also employ [4] to measure the preference values to get more accurate clustering results.

There are two kinds of message passing between points in the AP: availability ($a(i,k)$) and responsibility ($r(i,k)$). The availability sent from point *i* to point k shows how appropriate for point *i* to choose point *k* as their cluster center. The responsibility sent from point *i* to point *k* shows how well-suited point *k* is to be served as the cluster center for point *i*. Fig. 1 shows how the availability and the responsibility work between points.

In the first iteration, the availability $a(i,k)$ is set to zero. The responsibilities are computed as shown in (2):

$$\mathrm{r}(i,k) \leftarrow s(i,k) - \max_{k' \text{ s.t. } k' \neq k}\{a(i,k') + s(i,k')\} \quad (2)$$

Afterward, the availability gathers evidence from data points as to whether each candidate center would make a good cluster center as shown in (3):

$$\mathrm{a}(i,k) \leftarrow \min\left\{0, r(k,k) + \sum_{i' \text{ s.t. } i' \notin \{i,k\}} \max\{0, r(i',k)\}\right\} \quad (3)$$

For $k = i$, the responsibility $r(k,k)$ is set to the input preference that point k be chosen as the center, $s(k,k)$, minus the largest of the similarities between point *i* and all other candidate centers. But the self-availability ($\mathrm{a}(k,k)$) is updated separately:

$$\mathrm{a}(k,k) \leftarrow \sum_{i' \text{ s.t. } i' \neq k} \max\{0, r(i',k)\} \quad (4)$$

When updating the messages, it is important to avoid numerical oscillations. Each message is set to be λ multiply the value from the previous iteration plus $1-\lambda$ multiply the prescribed update value, as shown in (5). The default damping value λ is 0.5,

$$updated\ message \leftarrow updatedValue * (1-\lambda) + oldValue * \lambda \quad (5)$$

where *updatedValue* means updated availability and responsibility; *oldValue* means availability and responsibility from previous iteration.

In later iterations, when some points are effectively assigned to other centers, their availabilities will drop below zero. For availability, only the positive portions of incoming responsibilities are added, because it is only necessary for a good center to represent some data points well, regardless of how poorly it represents other data points. There are three terminal conditions for AP: (1) a fixed number of iterations are passed (2) the changes of

messages fall below a threshold value and (3) the local decisions stay constant for some number of iterations. Fig. 2 shows how the AP works step by step.

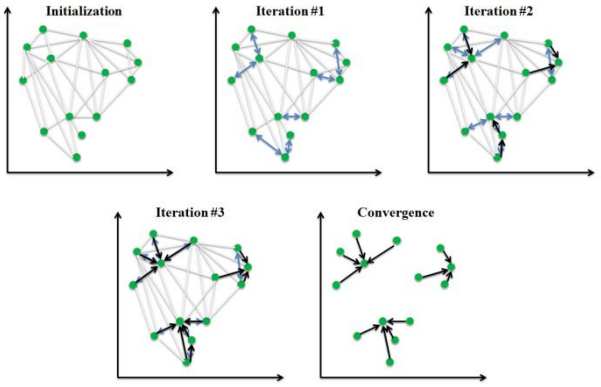

Figure 2. The progressing of the affinity Propagation.

For each point, combining the availabilities and responsibilities can identify its own center. For point *i*, the value of k that maximizes $a(i,k)+r(i,k)$ either identifies point *i* as a center if *k* = *i*, or identifies the data point that is the center for point *i*.

Each iteration of the AP consists of (i) updating all responsibilities given the availabilities, (ii) updating all availabilities given the responsibilities, and (iii) combining availabilities and responsibilities to monitor the center decisions and to decide termination or not.

Fig. 2 illustrates the AP for two-dimensional data points, where negative Euclidean distance (squared error) was used to measure similarity matrix. Through the exchange of availability and responsibility, shown as the blue lines during iteration, each point tries to decide its belonging to different clusters. The black arrow shows the final decision of clustering. The terminal condition is that the value of availability and responsibility does not vary through iteration or the floating values are below the threshold after a fixed number of iterations. The black arrow directed from point *i* to point k shows point *i* belongs to cluster with center point *k*.

B. Map/Reduce Affinity Propagation

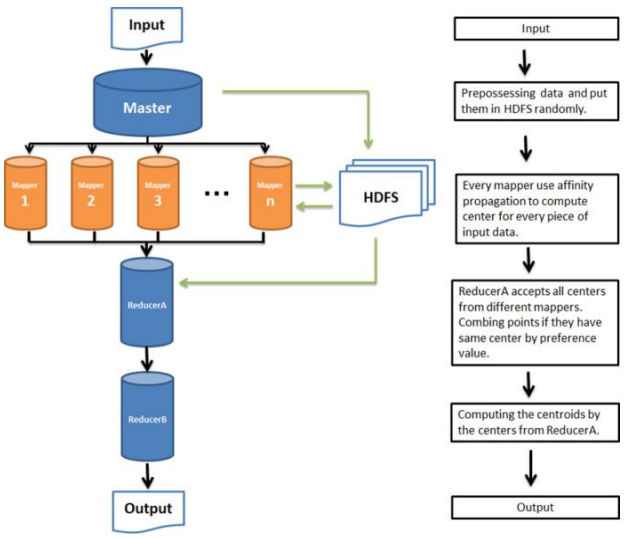

Figure 3. The framework of Map/Reduce affinity propagation.

We improve the AP in Map/Reduce environment to use multiple computers' computing abilities more efficiently to handle larger data. Multiple mappers and two reducers are needed in this framework. The number of mappers is scalable. More mappers indicate that the data size for each mapper is smaller, but more mappers will make the communication between mappers more frequently. The framework of the proposed Map/Reduce Affinity Propagation (MRAP) is shown in Fig. 3.

First, the input data is preprocessed for initial similarity and preference values. The input data can be processed by multiple mappers at the same time. We divide the input data to small pieces randomly and stored both the source input data and the divided input data in the Hadoop Distributed File System (HDFS). The suitable preference value range can be decided through the method described by [4]:

Step1. Initialize $s(k,k)$ to zero.

$$s(k,k)=0 \tag{6}$$

Step2. Compute the maximal preference.

$$P_{max}=max\{s(i,k)\} \tag{7}$$

Step3. Compute the minimal preference.

$$\text{dpsim1}=\max_{j}\left\{\sum_{j} s(i,j)\right\} \tag{8}$$

$$\text{dpsim2}=\max_{i\neq j}\left\{\sum_{k}\max\{s(i,k),s(j,k)\}\right\} \tag{9}$$

Step4. Compute the minimal preference.

$$P_{min}=dpsim1-dpsim2 \tag{10}$$

The range of preference value is from P_{min} to P_{max}.

In the mapping stage, every mapper will get different divided data from the HDFS and will perform the AP on their own data. During the AP, each mapper uses its own similarity matrix and preference values. After the AP, each mapper gets the cluster centers and cluster points for its own data. Each mapper has its own key to transmit its result to the next reducing stage. If point *i*'s center is point *k*, the transmitted data tuple is <*key*, *i*, *k*>.

In the Map/Reduce environment, all parameters transmit between a mapper and a reducer (or a mapper and another mapper, or a reducer and another reducer) have to be the form of <*key*, *value*>. The Map/Reduce unit will collect the values with the same key and process the values at the same time.

There are two reasons that each mapper should perform the AP independently with its own data: (1) in the AP, there are about hundreds of iterations needed. If we exchange the information for every iteration, a huge communication cost is expected in the map/reduce environment. (2) Every mapper is isolated in processing, which is restricted by the architecture to transfer data among mappers.

The reduce stage is composed of two parts: classify points and calculate the center points. Reducer A

classifies the clustered points and sends the result to reducer B to calculate the centers of clusters.

In reducer A, clustered points and their own centers are collected from the results of mappers. If the distance between two centers is less than the threshold, the points that are originally in two clusters will be combined into one cluster. Through experiments, the recommended threshold value is set to preference*0.5. The threshold value can be adjusted with different input data sets. The result is that points from different mappers with neighboring centers will be classified into the same cluster. The points that are far away from other data points will be classified as isolated points. There are two conditions that a point will be classified as isolated points: (1) if a cluster have few points and (2) the cluster's center is very far away from other clusters. In the end of reducer A, the points that are in the same cluster will be sent to reducer B by the same key.

In reducer B, there may be multiple centers in a cluster. To decide the true centers of clusters, we employ the centroid of centers from reducer A. If there are K centers from reducer A, the centroid is decided as in (11).

$$\text{Centroid} = \frac{C_1+C_2+C_3\ldots+C_k}{K} \quad (11)$$

After combining clusters, the centers of clusters are decided and the points of those combined clusters will be renumbered. Different cluster centers and their own clustered points are set with unique keys and then output. The output results are sets of clustered points and their corresponding cluster centers.

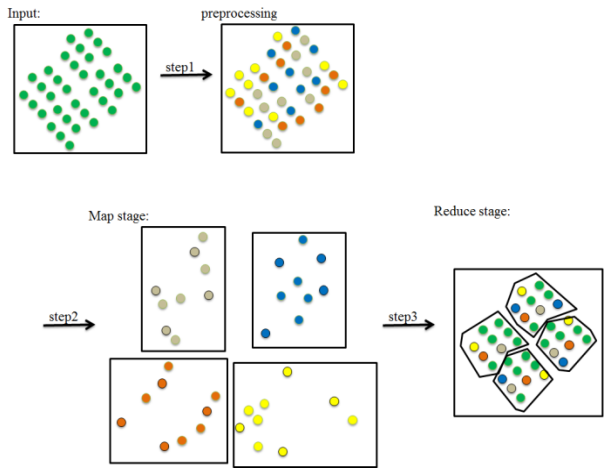

Figure 4. The processing stages of Map/Reduce affinity propagation.

Fig. 4 shows how the MRAP works: before the input to be processed, the MRAP divides data randomly and computes the similarity in the preprocessing stage. In the map stage, every mapper gets a part of divided data and performs AP independently. In the map stage of Fig. 4, color dots with black outlines denote the cluster centers after affinity propagation. Then the mappers deliver their own clustering results to the next stage. The reducer combines the clusters from the map stage and calculates the new cluster centers after combining neighboring clusters. During the process of calculating the new cluster centers, the reducer removes points that are far away from any of the clusters. In the end, the MRAP outputs the details of clusters including the final cluster centers and every point in each cluster.

III. Experiments

In the following experiments, the proposed MRAP is compared with the Map/Reduce K-means algorithm, which is included in the Apache Mahout Classification package. The experiments are divides into four sections: (1) running time, (2) accuracy, and (3) using the clustering results of the MRAP as the initial points for the K-means algorithm and compare to other initial methods for the K-means algorithm. We chose two datasets (iris and wine quality) from the UC Irvine Machine Learning Repository [11] and The Yale Face Database [12].Experiment environments as follow:

- Master (CPU: Intel® Core™2 Quad Processor Q6600 (8M Cache, 2.40 GHz, 1066 MHz FSB); RAM: DDR2-800 2G * 2)
  Node1for NameNode and JobTracker
- Slave (CPU: Intel® Core™2 Quad Processor Q6600 (8M Cache, 2.40 GHz, 1066 MHz FSB) ; RAM: DDR2-800 2G * 2)
  Node2~9 for DataNode and TaskTracker

1) Running time test

In this experiment, we focus on the processing time with different data dimensions and the number of data points is fixed to 5000. We chose the Canopy method as the comparing algorithm because similar to the MRAP, the a priori knowledge of the number of clusters is not required for the Canopy method.

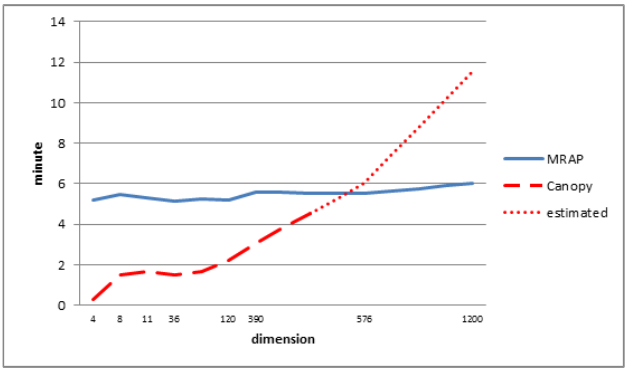

Figure 5. The result of running time experiment.

The dimension of input data varies from 4 to 1200. By employing the MRAP, the running time is about 5.5 minutes and does not increase when the dimensionality of data increases. Comparing to the Canopy method, the running time increases almost linearly when the dimensionality increases. While applying the Canopy method, we start getting the "Java heap error" when the dimension is increased to about 400. An as a result, the experiment cannot be completed beyond 400 dimensions. To try to conquer the error, we increase the RAM for each salve node from 1G to 4G, but the error persists disregarding the additional memory space. Fig. 5 shows the running time of the MRAP and the Canopy methods. The solid lines show the actual observed data, the dotted

line is an interpolation because the experiments cannot be completed with "Java heap error".

2) Accuracy test

The Canopy method as the initialization of the Map/Reduce K-means algorithm is chosen for the comparison study here. The accuracy which includes the precision rate and the recall rate of the MRAP and the Canopy method is presented in this section.

The dimension of the Iris dataset is 4. The T1 and T2 values of the Canopy method are set to the optimal values through many trials. As shown in Table I, the MRAP and the Canopy initialed Map/Reduce K-means method produce similar accuracy when the dimensionality is low.

TABLE I. ACCURACY OF IRIS DATASET

	Map/Reduce Affinity Propagation		K-means (Canopy initial)	
	Precision	Recall	Precision	Recall
Cluster1	0.69	0.98	0.723077	0.94
Cluster2	1	1	1	1
Cluster3	0.97	0.56	0.914286	0.64
Average	0.8867	0.84667	0.87667	0.86

The dimension of the Wine quality dataset is 11. The T1 and T2 values of the Canopy method are not set to the optimal values because it is hard to decide the suitable T1, T2 when the dimensionality is increasing. When the T1 and T2 values deviate from the prospective ranges, data points will be assigned to only specific clusters, as shown in Table II. The result is that the precision rate and the recall rate decrease significantly. The MRAP method produces more stable clustering output than the Canopy initialed Map/Reduce K-means when the data dimensionality increases.

TABLE II. ACCURACY OF WINE QUALITY DATASET

	Map/Reduce Affinity Propagation		K-means (Canopy initial)	
	Precision	Recall	Precision	Recall
Cluster1	0.75	0.1472	0.0615	1
Cluster2	0.4665	0.1584	0.0745	0.1123
Cluster3	0.4474	0.2253	0	0
Cluster4	0.1823	0.3643	0	0
Cluster5	0.0543	0.4629	0	0
Average	0.466	0.267	0.026	0.22

To sum up, the proposed MRAP method and the Canopy initialed Map/Reduce K-means method have similar clustering accuracy when the data dimensionality is low and with simpler data distributions. When the data dimensionality increases, the data distributions tend to be more complex. However, the prospective T1 and T2 parameter values in the Canopy method are increasingly more difficult to find as the dimensionality increases. Even though we can get fair T1 and T2 values for the Canopy initialed Map/Reduce K-means method, the clustering accuracy is lower than the proposed MRAP method.

3) K-means initialization test

The following experiments try to decide which initialization algorithm can reduce the number of iterations before convergence for the Map/Reduce K-means method provided by the Apache Mahout package. Three algorithms are chosen in this experiment: the proposed MRAP method, the Canopy method, and the random seed method, where the latter two methods are provided by the Apache Mahout clustering package. Each method is repeated 10 times for different datasets in the experiments.

The Fig. 6 shows that the numbers of iterations before convergence for the Map/Reduce K-means method initialized by the Canopy method is generally less than initialized by the MRAP method and the random seed method. But the average clustering precision of the Canopy method (0.026) is much lower than the MRAP method (0.466), as discussed in the previous section.

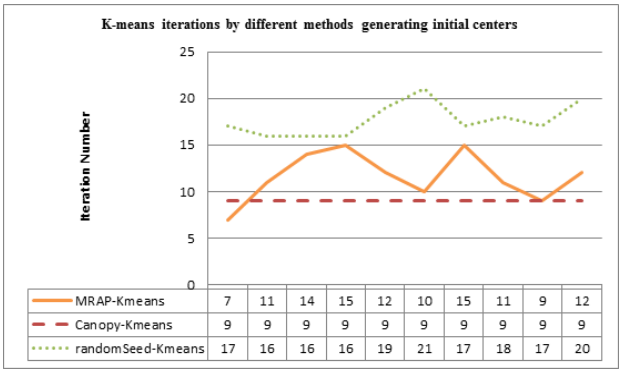

Figure 6. The number of K-means iterations for wine quality dataset.

The Fig. 7 shows the numbers of iterations before convergence for the Map/Reduce K-means method with the Iris datasets. Fig. 7 shows that the MRAP initialed Map/Reduce K-means can help to reduce the numbers of iterations before convergence significantly. The random seed method is generally requires higher number of iterations.

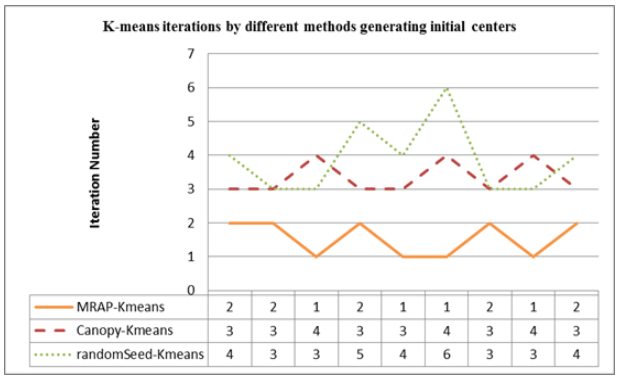

Figure 7. The number of K-means iterations for iris dataset.

Fig. 8 shows the numbers of iterations before convergence for the Map/Reduce K-means method with the Face dataset, which is 1200 in dimensions. In Section 3.1, we discuss that the Canopy initialed Map/Reduce K-means method cannot work in high dimensional data space. So in this experiment we only compare the proposed MRAP method with the random seed method. The results shown in Fig. 8 suggest that the proposed MRAP method can still reduce the number of iterations for the Map/Reduce K-means method in extreme high

dimensional data space. From the experiments in this section, the proposed MRAP method can reduce the number of iterations before convergence for the Map/Reduce K-means method irrespective to the data dimensions.

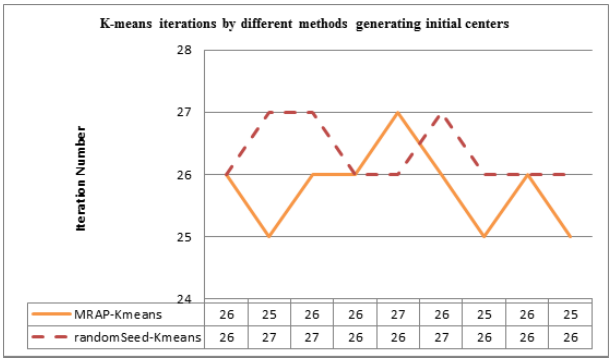

Figure 8. The number of K-means iterations for face image dataset.

After the experiments, we identify the advantages and disadvantages of the proposed MRAP method:

- Advantages:

1) The clustering process can be spread to multiple nodes and the processing time does not increase when the dimensionality of data set increases.

2) MRAP's precision and recall is higher than the Map/Reduce K-means. It is more obvious when the dimension of data set increase.

3) Using MRAP for the initialization of K-means can make the convergence of K-means better than the other initializations provided by Apache Mahout.

4) The reason that Map/Reduce K-means does not have a good cluster resulting is there is no good initial method for Map/Reduce K-means. (Canopy's result extremely depends on the sets of <T1, T2>, but there is no a reasonable way to find them.) If MRAP is set to the initial method of Map/Reduce K-means, it can improve the precision and the number of convergence iteration.

- Disadvantages:

The clustering result will not good obviously when dimension of data set keeps increasing.

IV. Conclusion

We proposed the Map/Reduce Affinity Propagation (MRAP) method implemented on Hadoop. The proposed MRAP requires multiple mappers and two reducers and can partition the job to multiple nodes to make data clustering more efficient. The MRAP is scalable by one or multiple nodes. Because of the Map/Reduce architecture, the MRAP can process large data set with good performance.

In our system architecture, the MRAP can accept large data set and process them in constant time. During the process, the MRAP can decide the appropriate number of clusters automatically. In the experiments, we observe that the MRAP have lower processing time, higher precision and recall rate, and can help the K-means algorithm to converge faster than other initialization techniques of K-means algorithm such as Canopy.

However, there are still several problems need to be solved in the future work. First, if it is possible to transfer data between mappers during the cluster processing, it can make the MRAP more efficient. The Map/Reduce 2.0 [13] architecture under developing will allow information transfer between mappers and hence can further improve the performance of the proposed MRAP in the future. Second, for the threshold of combining the clusters from mappers, we can try to automatically decide the threshold by clusters' variance, which may help to improve the accuracy of the MRAP.

Acknowledgment

This work was partially supported by the National Science Council, Taiwan, under the Grants No. NSC102-2221-E-011-134, NSC102-2221-E-211-012.

References

[1] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, pp. 972-976, Feb. 2007.
[2] I. E. Givoni and B. J. Frey, "A binary variable model for affinity propagation," *Neural Computation*, vol. 21, no. 6, pp. 1589-1600, Jun. 2009.
[3] C. D. Wang, J. H. Lai, C. Y. Suen, and J. Y. Zhu, "Multi-Exemplar affinity propagation," *Pattern Analysis and Machine Intelligence, IEEE Transactions*, Jan. 2013.
[4] Y. C. He, Q. C. Chen, X. L. Wang, R. F. Xu, X. H. Bai, and X. J. Meng, "An adaptive affinity propagation document clustering," in *Proc. 7th International Conference on Informatics and Systems*, March 2010, pp. 1-7.
[5] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Communications of the ACM-50th Anniversary Issue: 1958-2008*, vol. 51, no. 1, pp. 107-113, Jan. 2008.
[6] Z. Q. Xu and D. W. Zhao, "Research on clustering algorithm for massive data based on hadoop platform," *Computer Science and Service System*, pp. 43-45, Aug. 2012.
[7] M. A. Bhandarkar, "MapReduce programming with apache Hadoop," in *Proc. International Parallel and Distributed Processing Symposium/International Parallel Processing Symposium-IPDPS(IPPS)*, April 2010, pp. 19-23.
[8] Y. Lai, "An efficient data mining framework on hadoop using java persistence API," *Computer and Information Technology*, Jul. 2010.
[9] M. Maurya and S. Mahajan, "Performance analysis of MapReduce programs on Hadoop cluster," *Information and Communication Technologies (WICT)*, pp. 505-510, 2012.
[10] D. L. Olson, *Advanced Data Mining Techniques*, 1st ed. Feb. 2008.
[11] (Mar. 1st, 2013). UCI Machine Learning Repository. [Online]. Available: http://archive.ics.uci.edu/ml/
[12] (Mar. 1st, 2013). The Yale Face Database. [Online]. Available: http://cvc.yale.edu/projects/yalefaces/yalefaces.html
[13] A. Radwan. (May 1st, 2013). MapReduce 2.0 in Apache Hadoop 0.23. [Online]. Available: http://blog.cloudera.com/blog/2012/02/mapreduce-2-0-in-hadoop-0-23/

Wei-Chih Hung received the B.B.A. and M.I.M. degrees in information management form Hua-Fan University, Taiwan, in 2006 and 2008 respectively. Currently, he is a graduate student in Department of Computer Science and Information Engineering at the National Taiwan University of Science and Technology, Taiwan.

Chun-Yen Chu He received his Master's degrees in Computer Science and Information Engineering at the National Taiwan University of Science and Technology.

Yi-Leh Wu is an Assistant Professor of the Department of Computer Science and Information Engineering at the National Taiwan University of Science and Technology, Taiwan. Prior to joining the faculty, he led research and software development projects full-time as an engineering manager at VIMA Technologies Inc. He received his doctoral degree in Computer Science from the University of California, Santa Barbara.

Cheng-Yuan Tang is an Associate Professor of the Department of Information Management of Hua-Fan University, Taiwan. He received the Ph.D. degree in computer science and information engineering from National Chiao Tung University of Taiwan. His research interests include computer vision, image processing, medical image, visual surveillance, pattern recognition, and computer graphics.