A Reinforce Parallel K-Means Clustering Using OpenMP: OM K-Means

V Gopinath[1], Ch Yallamandha[2], K Purna Prakash[3], Dr S Krishna Rao[4]
1 Sir C R Reddy College of Engineering,Asst.Prof,velivela.gopi@gmail.com
2 Sir C R Reddy College of Engineering, Asst.Prof, challa.ynaidu@gmail.com
3 Sir C R Reddy College of Engineering,Asst.Prof,prakash.nani@gmail.com
4 Sir C R Reddy College of Engineering,Professor,skrao71@gmail.com

ABSTRACT

In the era of data mining, cluster analysis is very needy technique in engineering applications. So in the jargon of clustering techniques, k means is one among the most effective and efficient algorithm. K means algorithm varies by selecting the initial centriods, distance measures and mean calculations. As the dataset size increases the algorithm performance degrades rapidly. In order to maintain accuracy and time complexity we propose an algorithm that gives accurate results.

The simple K means algorithm accuracy depends upon choosing the initial centriods. In the algorithm analysis, time complexity is a measure of defining an algorithm is efficient or not. So the proposed algorithm maintains the O(nlogn) time complexity. The experimental results reflect that the proposed algorithm works well as if the data set size increases. The proposed algorithm implemented in parallel environment by using OPENMP.

Key Words: Data mining, Data set Clustering, Time Complexity, K-Means and Distance Measures.

1.INTRODUCTION

1.1 Clustering:

The term (or) techniques *Data Mining* has become very popular in recent years. Finding groups of objects such that the objects in a group will be analogous (or related) to one another and different from (or unrelated to) the objects in other groups. [1] [2].

Cluster Analysis is very useful without proper analysis implementation of clustering algorithm will not provide good results. Cluster analysis is very useful to understand group related documents like web browsing, group genes and proteins that have similar functionality or group stocks with related price fluctuations and also reduces the size of very large data sets. Traditionally clustering techniques are usually divided in hierarchical and partitioning and density based clustering.

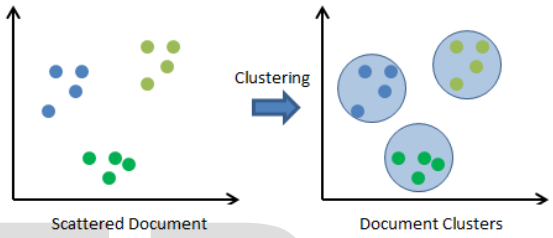

Fig 1.0 Clustering Concept

A very good range of methods have been proposed for solving clustering problems. One of the most popular clustering method is k means clustering algorithm developed [11][15][16] by Mac Queen in 1967. The easiness of k means clustering algorithm made this algorithm used in several fields. The k means clustering algorithm is a partitioning clustering method that separates data into k groups [4][5]. The k-means clustering algorithm is more prominent since its intelligence to cluster massive data rapidly and efficiently. However, k means algorithm is highly precarious in initial cluster centers. Because of the initial cluster centers produced arbitrarily, k means algorithm does not promise to produce the strange clustering results.

1.2 Weka:

The GUI Chooser consists of four interfaces—one for each of the four major Weka applications—and four menus. The buttons can be used to start the following applications:

• Explorer: An environment for exploring the data with WEKA.

• Experimenter: An environment for

performing experiments and conducting statistical tests between learning schemes.

• KnowledgeFlow: This environment supports essentially the same functions as the Explorer but with a drag-and-drop interface. One advantage is that it supports incremental learning.

• Simple CLI: It provides a simple command-line interface that allows direct execution of WEKA commands for operating systems that do not offer their own command line interface.

1.3 OpenMP

The OpenMP API covers only user-directed parallelization, wherein the programmer unambiguously specifies the actions to be taken by the compiler and runtime system in order to execute the program in parallel. OpenMP [18] compliant implementations are not required to ensure for data dependencies, data conflicts, race conditions, or deadlocks, any of which may occur in conforming programs.

The OpenMP API uses the fork and join model of parallel execution. Multiple threads of execution perform tasks defined implicitly or explicitly by OpenMP directives. The OpenMP API is intended to support programs that will execute correctly both as parallel programs and as sequential programs

The rest of this paper is organized as follows. Section 2 gives the related work. Section 3 mainly presents our OM K means algorithm in detail. Section 4 shows our experimental results and performance evaluation. Finally, Section 5 concludes the paper and outlines our future work.

2.RELATED WORK

Related work including K-means clustering algorithm, OPENMP and Weka are involved in this section. More precisely, an outline on K-means is summarized in Section A, implementation of enhanced k means is given in Section B and Implementation of openmp is given in Section C.

2.1 K-Means

The most popular and the simplest partition algorithm is K means [5]. K means has a rich and diverse history as it was independently discovered in different scientific fields by Steinhaus (1956), Lloyd (proposed in 1957, published in 1982), Ball and Hall (1965), and MacQueen (1967) [1]. Kmeans is a method of data analysis. The objective is to partition N data objects into K clusters (K<N) such that the objects in the same cluster are as similar as possible and as dissimilar as possible in different clusters.

Algorithm 1: The k-means clustering algorithm

Input:

D = {d1, d2,.....,dn} //set of *n* data items.
k // Number of desired clusters

Output:

A set of *k* clusters.

Steps:

1. Arbitrarily choose *k* data-items from D as initial centroids;
2. Repeat
   2.1 Assign each data item d_i to the cluster which has the closest centroid;
   2.2 Calculate the new mean of each cluster;

Until convergence criterion is met.

2.2 Enhanced K Means

More often, we may have to deal with multi-dimensional data values. Each data point *di* may contain multiple attributes such as *di1, di2,…..dim*, where *m* is the number of attributes or columns in each data value. In such cases we first determine the column with maximum range [6], where range is the difference between the maximum and the minimum element of the column. Initially, from the multidimensional data values, we determine the maximum and minimum element of each column and compute the range of values for each column as the difference between the maximum and minimum values.

Then we identify the attribute (column) having maximum range. The entire set of data values are then sorted in non-decreasing order, using the Heap Sort algorithm [7][12][19], based on the attribute with maximum range. The sorted list of data points are then divided into 'k' equal sets. Finally, the arithmetic means of each of these 'k' sets are computed. These means become the initial centroids of the clusters to be formed.

After determining the initial centroids as described above, the data points are assigned to various clusters by using the same method used in the second phase of the original k means algorithm. Each data point *di* is assigned to the cluster having the closest centroid. Euclidean distance is used as the measure for shaping the distance between the data points and the

centroids. The proposed algorithm is outlined below as Algorithm 2.

Algorithm 2: Enhanced Algorithm for Clustering

Input:
D = {d1, d2,.....,dn} // set of *n* data items.
k // Number of desired clusters.
Output:
A set of *k* clusters.
Steps:
1. For each column of the data set, determine the *range* as the difference between the maximum and the minimum element;
2. Identify the column having the maximum *range*;
3. Sort the entire data set in non-decreasing order based on the column having the maximum *range*;
4. Partition the sorted data set into 'k' equal parts;
5. Determine the arithmetic mean of each part obtained in Step 4 as *c1*, *c2*,….*ck*; Take these mean values as the initial centroids.
6. **Repeat**
   6.1 Assign each data item *di* to the cluster which has the closest centroid;
   6.2 Calculate new mean of each cluster;
   Until convergence criterion is met.

2.3 *OpenMP*

OpenMP is an industry standard API of C/C++ and FORTRAN for shared memory parallel programming. This specification provides a model for parallel programming that is very portable across shared memory architectures from various vendors. Compilers from several vendors support the OpenMP API. The main technique used to parallelize code in OpenMP is the compiler directives. The compiler directives are added to the source code as an indicator to the compiler of the presence of a region to be executed in parallel, along with some bunch of instruction on how that region is to be parallelized. OpenMP is typically used to parallelize the loops by identifying the most time consuming loops. The comparison of serial and parallel programs is given below.

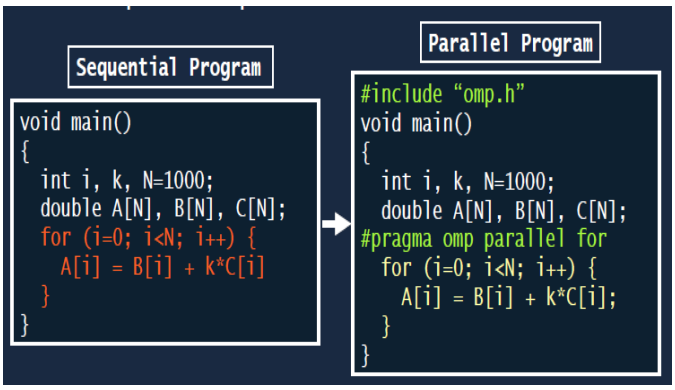

Fig. 2 Example of Parallel program in Openmp

By using constructs the Parallelism is achieved. The common OpenMP constructs are-
1. Parallel Regions
   -# pragma omp parallel
2. Work sharing
   -#pragma omp for, #pragma omp sections
3. Data environment
   -#pragma omp parallel shared/private (...)
4. Synchronization
   -#pragma omp barrier
5. Runtime functions/variables
   -int my_thread_id=omp_get_num_threads (...);
   -omp_set_num_threads (8);

OpenMP has advantages compared to MPI (Message Passing Interface) in data partitioning and communication scheduling. Incremental parallelism is achieved through OpenMP.

3.PROPOSED ALGORITHM

The proposed work involves a novel approach of introducing parallel concept to the enhanced k-means algorithm. The parallelism is achieved by using OpenMP API.

The Proposed algorithm is designed by identifying the most time consuming loop of the older algorithm. And then introduce parallel OpenMP constructs by #pragma omp parallel construct. The time complexity of the algorithm is remained unaltered, moreover when the dataset size increases rapidly then the proposed algorithm works efficiently.

Algorithm: OM K-Means Algorithm for Clustering

Input:
D = {d1, d2...dn} // set of *n* data items.
K // Number of desired clusters.

Output:
A set of *k* clusters.

Steps:
1. Master process reads the data set and then partition the data set and sends to all other slave process.
2. For each slave process do the steps 3-8,
3. For each column of the data set, find out the *range* as the difference between the maximum and the minimum element.
4. Identify the column having the maximum *range*;
5. Sort the entire data set in anti-decreasing order based on the column having the maximum *range*.
6. Partition the sorted data set into 'k' equal parts.
7. Establish the arithmetic mean of each part obtained in Step 4 as *c1, c2,….ck*; Take these mean values as the initial centroids
8. **Repeat** // #pragma omp parallel
   8.1 Assign each data item *di* to the cluster which has the closest centroid;
   8.2 Calculate new mean of each cluster;
   Until convergence criterion is met.
9. Master process gathers the all cluster results by slave processes.
10. Mater process declares the clustering results.

The architecture of proposed algorithm is illustrated in the figure 3.

The proposed algorithm is a parallel algorithm. So there involves master and slave processes. At first master Process loads the dataset and perform partition of data set and sends evenly to the slaves. Now the effect and advantage of parallelism came to known. Each slave process performs initial centroids first and then assigns each data point di to the cluster until convergence criteria met. If convergence criteria met among the each slave process the clustering result is sends to master. The master process displays the final clustering result. Each data point *di* may contain multiple attributes such as *di1, di2…dim*, where *m* is the number of attributes or columns in each data point value.

In such cases we first identify the column with maximum range [7], where range is the difference between the maximum and the minimum element in the column. Initially, from the multidimensional data values, we determine the maximum and minimum element of each column and compute the range of values for each column as the difference between the maximum and minimum values.

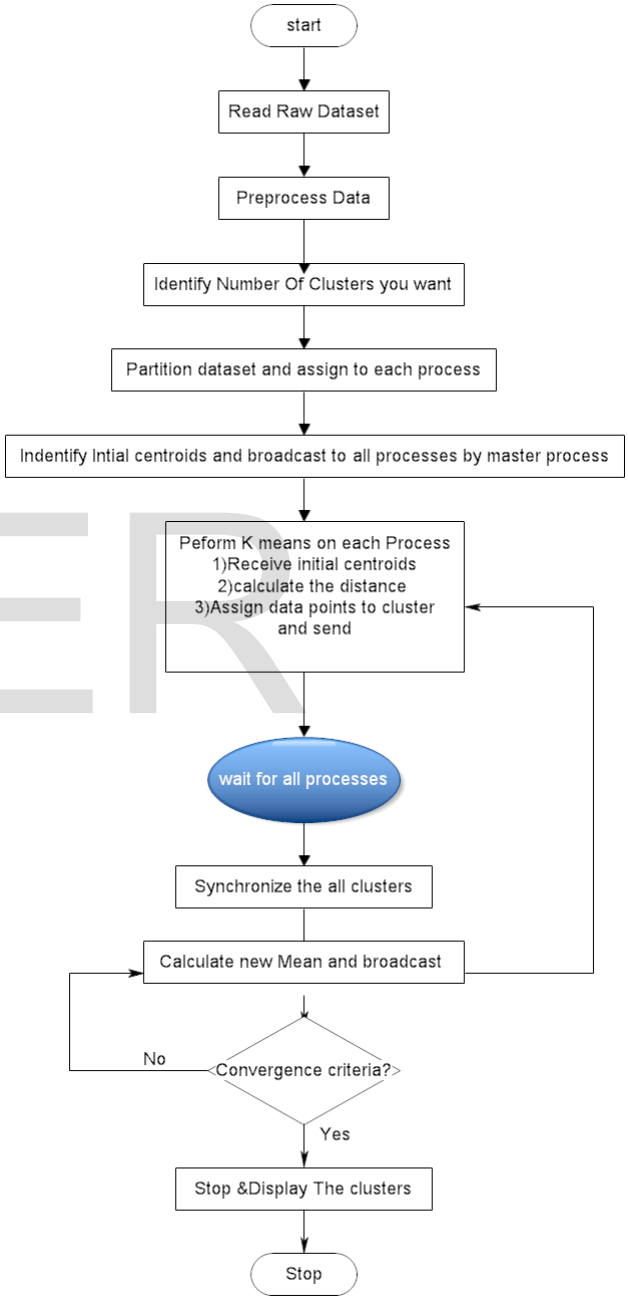

Fig. 3 System architecture of OM K mean

Then we identify the attribute (column) having maximum range. The entire set of data values are then sorted in increasing order, using the quick Sort algorithm [6], based on the attribute with maximum range. The sorted list of data points are then divided into 'k' equal sets. Finally, the

arithmetic means of each of these 'k' sets are computed. These means become the initial centroids of the clusters to be formed.

4.EXPERIMENTS

In this section, we first give the experimental environment in Section A. Second, we give the description of experimental data sets in Section B. Last, we give the experimental results and analysis in Section C.

4.1 Experimental Environment

The hardware platform in this paper uses a PC with the configuration: Intel Pentium processor, 2GB RAM, 300GB hard disk; the software environment uses the following configuration: the operation system is Windows XP Professional Service Pack 3, the parallel and distributed environment is the Windows version of OPENMP4.0, Java development platform is the JDK 1.6; Network environment is 100M- LAN.

In terms of above mentioned platform, eclipse SDK 4.4.2 is used to develop procedures. Considering the fairness of comparison, the configuration of OPENMP parallel development platform is based on open resource project Eclipse in Windows, and the experimental platform has a C/C++ compiler based on GNU.

4.2 Data Sets:

All experimental data sets are selected from the UCI Machine Learning Dataset Repository [8]. The information of all data sets is illustrated as shown in Table 1. In this table, eight testing data sets are listed corresponding to the number of instances. As the number of instances increases, the space consumption of data sets also increases, denoted as size.

Table1. List of datasets and their sizes

Code Number of Datasets	Name of the Datasets(.arff)	Size	Number of Instances
1	Vowel	90KB	990
2	Segment	298KB	2310
3	Daily and Sports Activity	1MB	9120
4	Gas Sensor Array Drift	1.5MB	13910
5	KEGG Metabolic Reaction Network	6.4MB	65554
6	Diabetes	10MB	100000
7	Individual Household Electric Power Consumption	207.6MB	2075259
8	Heterogeneity Activity Recognition	1.2GB	43930257

4.3. Experimental Results

In our experiments, the time cost is the key performance. The efficiency and clustering time are calculated respectively. In simple k means, enhanced k-means and OM K means. To reflect the fairness and authenticity of the proposed algorithms, the number of processes is 1 in OM K means. Table 2 reports the results of the above mentioned two algorithms, including the accuracy and time taken to form clusters.

Table2. Accuracy comparison of three algorithms

Code Number of Datasets	Accuracy of Simple k-means (%)	Accuracy of Enhanced k means	Accuracy of OM K means
1	85	89	89.3
2	73	76	78.2
3	62	69	76.8
4	56.6	65	72.8
5	51.5	69	71.5
6	45	50	70.6
7	30	52	69.5
8	32	46	68.5

Table 3.Time taken Comparison of Algorithms

Code Number of Datasets	Time taken for K means(ms)	Time taken for Enhanced K	Time taken for OM K means(ms)

		means(ms)	
1	41	35	32
2	52	42	37
3	68	55	41
4	72	62	49
5	79	65	56
6	83	69	59
7	94	70	62
8	110	82	68

5.CONCLUSION

We proposed parallel k-means partitioning clustering algorithm to improve running time of the algorithm as well as accuracy. As the dataset size increases the efficiency of k-means algorithm degrades very rapidly. The time complexity and accuracy is depends on choosing the initial centroids. The way we choose the initial centroids, the way the efficient clusters are formed.

To overcome the difficulties in the cluster analysis the paper suggests the best option. So to improve the efficiency and accuracy of the clustering process, the proposed algorithm has taken initial centriods in a preferable manner and to handle with big data problems parallelism concept is introduced in the proposed algorithm.

ACKNOWLEDGMENT

The efforts were guided and supported relatively by Management & Principal of Sir C R Reddy College of Engineering, Eluru, A.P, and India.

REFERENCES

[1] Micheline Kamber, Datamining:conecepts and techniques,2nd edi, daine cerra publication text book.

[2] Margaret H Dunham, Data *Mining-Introductory and Advanced Concepts*, Pearson Education, 2006.

[3] F. Yuan, Z. H. Meng, H. X. Zhangz, C. R. Dong, " A New Algorithm to Get the Initial Centroids," proceedings of the 3rd International Conference on Machine Learning and Cybernetics, pp. 26-29, August 2004.

[4] Koheri Arai and Ali Ridho Barakbah, "Hierarchical K-means: analgorithm for Centroids initialization for k-means," department of information science and Electrical Engineering Politechnique in Surabaya, Faculty of Science and Engineering, Saga University,Vol. 36, No.1, 2007

[5] S. Kantabutra, A. Couch, "Parallel K-means Clustering Algorithm on NOWs," Technical Journal, vol. 1, no. 6, 2000.

[6] Pang-Ning Tan, Michael Steinback and Vipin Kumar, Introduction to Data Mining, Pearson Education, 2007

[7] T H Cormen, C E Leiserson, R L Rivest and C Stein, Introduction to Algorithms, Second Edition, MIT Press, 2001.

[8] Merz C and Murphy P, UCI Repository of Machine Learning Databases, Available: ftp://ftp.ics.uci.edu/pub/machine-learningdatabases

[9] J. Zhang, G. Wu, H. Li, X. Hu, X. Wu, "A 2-Tier Clustering Algorithm with Map-Reduce," in: Proceedings of the 5th ChinaGrid Annual Conference (ChinaGrid'10), pp. 160-166, July 2010.

[10] K. A. Abdul Nazeer and M. P. Sebastian, "Improving the accuracy and efficiency of the k-means clustering algorithm," in International Conference on Data Mining and Knowledge Engineering (ICDMKE),Proceedings of the World Congress on Engineering (WCE-2009), Vol 1, July 2009, London, UK

[11] W. Zhao, H. Ma, Q. He, "Parallel K-Means Clustering Based on Map Reduce," in: Cloud Computing, vol. 5931, pp. 674-679, 2009.

[12] C. Blake, E. Keogh, C. Merz, "UCI Repository of machine learning databases," Irvine: Department of Information and Computer Science, University of California, 1998.

[13] E. Frank, M. Hall, G. Holmes, R. Kirkby, B. Pfahringer, I. Witten, L. Trigg, "Weka-A Machine Learning Workbench for Data Mining, "Data Mining and Knowledge Discovery Handbook, pp. 1269-1277,2010.

[14] E. Lusk, N. Doss, A. Skjellum, "A High-Performance, Portable Implementation of the MPI Message Passing Interface," Parallel Computing, vol. 22, pp. 789-828, 1996.

[15] Yuan F, Meng Z. H, Zhang H. X and Dong C. R, "A New Algorithm to Get the Initial Centroids," Proc. of the 3rd International Conference on Machine

Learning and Cybernetics, pages 26–29, August 2004.

[16] Chen Zhang and Shixiong Xia, " K-means Clustering Algorithm with Improved Initial center," in Second International Workshop on Knowledge Discovery and Data Mining (WKDD), pp. 790-792, 2009.

[17] S. Deelers and S. Auwatanamongkol, "Enhancing K-Means Algorithm with Initial Cluster Centers Derived from Data Partitioning along the Data Axis with the Highest Variance," International Journal of Computer Science, Vol. 2, Number 4.

[18] X. Wu, V. Kumar, Ross, J. Ghosh, Q. Yang, H. Motoda, G. Mclachlan, A. Ng, B. Liu, P. Yu, Z.-H. Zhou, M. Steinbach, D. Hand, and D. Steinberg, "Top 10 algorithms in data mining," Knowledge and Information Systems, vol. 14, no. 1, pp. 1-37,January 2008.

[19] I. Witten, E. Frank, L. Triqq, M. Hall, G. Holmes, S. Cunninqham, "Weka: Practical Machine Learning Tools and Techniques with Java Implementations," in: Proceedings of ICONIP/ ANZIIS/ ANNES99 Future Directions for Intelligent Systems and Information Sciences, 1999.