A NOVEL APPROACH TOWARDS PARALLEL K-MEANS

Gopinath Velivela, DS Bhupal Naik
Department of Computer Science & Engineering, Vignan University, Guntur, Andhra Pradesh, India
velivela.gopi@gmail.com, dsbupal@gmail.com

ABSTRACT
Clustering analysis plays an important role in scientific research and non scientific applications. K-means algorithm is a rapidly used partitioned method in clustering. As the dataset's scale increases gradually, it is very hard to use K-means to deal with big data. A parallel Technique is incorporated into clustering method. For Enhancing the accuracy and efficiency of clustering process, enhanced k means is paralyzed to improve the execution time and without affecting the accuracy and time complexity O (n log n).The proposed algorithm enhances the efficiency of enhanced k-means by introducing parallelism concept. Parallelism is achieved through OpenMP.

INDEX TERMS– *Clustering, K-means, Enhanced K-means, Parallelism concept.*

1. INTRODUCTION

The Introduction consists of the three sections. Section A involves brief introduction to the cluster analysis. In section B
Brief knowledge of k-means algorithm is discussed. The parallel environment is discussed under section C.

A. Clustering in data mining

Large amount of data is being collected every day in many business and science areas [1]. This data needs to be analyzed in order to find interesting information from it, and one of the most important analyzing methods is data clustering.
Clustering is one of the most important data mining tools which help data analyzers to understand the natural grouping of attributes in the data [2]. Cluster analysis is used in many field such as data mining [3], pattern recognition and pattern classification, data compression, machine learning, image processing and analysis and bioinformatics.
Data clustering is a method in which a cluster of objects is made that are somehow similar in distinctive. The process for checking the similarity is implementation dependent. K-means clustering algorithm is one of the most powerful efficient methods of discovering clusters.

B. K-means clustering

One of the most popular clustering methods is k-means clustering algorithm. In this algorithm mostly Euclidean distance is used to find distance between data points and centroids [4].The Euclidean distance between two multi-dimensional data points *X = (x1, x2, x3... xm)* and *Y = (y1, y2, y3... ym)* is described as follows:

$$\mathrm{d(X,Y)}=\sqrt{(x_1-y_1)^2+(x_2-y_2)^2+..+(x_m-y_m)^2}$$

Algorithm: The k-means clustering algorithm

Input: *D = {d1, d2, d3... di... dn }* // Set of n Data points.
K // Number of desired clusters
Output: A set of *k* clusters.
Steps:
1. Arbitrarily choose *k* data points from *D* as initial
Centroids;
2. **Repeat**
Assign every point d_i to the cluster which has the
Closest centroid;
Calculate & identify the new mean for each cluster;
Until convergence criteria is met.

Figure 1. K-means algorithm

The computational time complexity of the k-means technique is *O (nkl),* where *n* is the total number of data points in the dataset, *k* is the number of clusters needed and *l* is the number of iterations. So, the computational complexity of the k-means algorithm is rely on the number of data points, number of iterations and number of clusters .

C. *Parallel environment*

Parallel computing is simultaneous use of multiple compute resources to solve a computational problem. In parallel computing a problem is broken into discrete parts that can be solved concurrently and each part is further broken down to a series of instructions. The instructions from each part execute simultaneously on different CPUs. According to Flynn, the matrix given below defines the four possible classifications.

Table 1. Flynn's classification

SISD (Single Instruction, Single Data)	**SIMD** (Single Instruction Multiple Data)
MISD (Multiple Instruction, Single Data)	**MIMD** (Multiple instruction, Multiple Data)

Many scholars focus on the studies in the fields of parallel and distributed clustering. More specifically, (1) Rasmussen et al. suggested that the parallel processing using an array processor like the DAP (Distributed Array Processor) can provide significant speedups over serial processing for the hierarchic agglomerative cluster analysis of large data sets [5]. (2) Olson et al. considered parallel algorithms for hierarchical clustering using several inter cluster distance metrics and parallel computer architectures [6]. (3) Zhao et al. proposed a fast parallel K-means clustering algorithm based on Map Reduce. The proposed algorithm can scale well and efficiently to process large datasets on commodity hardware [7].

2. RELATED WORK

Related work includes enhanced K-means clustering algorithm, OpenMP. More precisely, an outline on enhanced K-means is summarized in Section *A*, and the brief introduction of OpenMP is given in Section *B*.

A. Enhanced K-means clustering

The basic idea of this algorithm is to determine the initial centroids of the clusters in a heuristic manner, so as to ensure that the centroids are chosen in accordance with the distribution of data. The method involves sorting the input data set and partition the sorted data set into 'k' number of sets where 'k' is the number of clusters to be formed. Mean values of each of these sets are taken as the initial centroids.

Algorithm: Enhanced 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. For each and every column of the data set D, identify the
Range as the difference of 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
6.2 Assign each data item *di* to the cluster which has the closest centroid;
6.3 Calculate new mean of each cluster;
Until convergence criterion is met.

Figure 2. Enhanced k-means algorithm

Unlike the original k-means algorithm in which the initial centroids are selected randomly, the proposed algorithm determines the initial centroids in a more meaningful way, in accordance with the distribution of data. Consequently, the algorithm converges much faster than the original k-means algorithm. Moreover, since the method for
Determining the initial centroids is based on the technique of
Sorting, this phase requires less time compared to other similar approaches.

B. 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 portable across shared memory architectures from various vendors. Compilers from several vendors support the OpenMP API. The main technique used to parallelise 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.

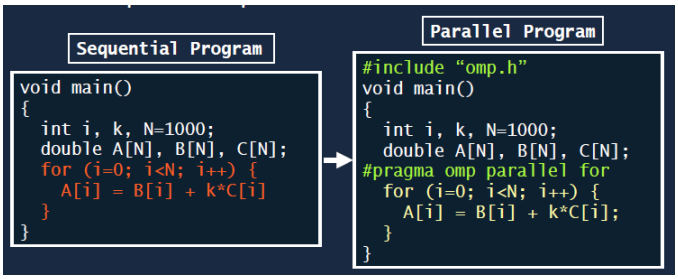

Figure 3. Example of OpenMP

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 WORK

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.

Parallel K-means Algorithm

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 partition the data set and sends to all slave process.
2. For each slave process install the steps 3-8,
3. For each column of the data set, determine 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 non-decreasing order based on the column having the maximum *range.*
6. Partition the sorted data set into 'k' equal parts.
7. Determine 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
   8.2 Assign each data item *di* to the cluster which has the closest centroid;
   8.3 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.

Figure 4. Exalt Parallel K-means algorithm

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 ccentroids first and then assigns each data point d_i 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 [8], 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. 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 [9], 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.*

A. Experimental environment

The hardware platform in this paper uses a PC with the configuration: Intel Xeon 5110 dual-core processor, 3GB RAM, 320GB hard drive; 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 OpenMP standard, Java development platform is the JDK 1.6; Network environment is 100M- LAN.

In terms of aforementioned platform, my eclipse SDK 8.6 is used to develop procedures. Considering the fairness of comparison, the configuration of OpenMP parallel development platform is based on open resource project My Eclipse in Windows, and the experimental platform has a C/C++ compiler based on MinGW (Minimalist GNU for Windows).

B. Data sets

All experimental data sets are selected from the UCI Machine Learning Dataset Repository [10]. The information of all data sets is illustrated as shown in Table 2. In this table, seven 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.

Table 2. Dataset report

Dataset Number	Name of dataset(.arff)	Size(KB)	Number of instances
1	Glass	18	214
2	Diabetes	37	768
3	Ionosphere	79	351
4	Soybean	199	683
5	Supermarket	1979	4627

C. Experimental results and analysis

In our experiments, the time cost is the key performance. The I/O time and clustering time are calculated respectively. In enhanced k-means and exalt parallel k-means. To reflect the fairness and authenticity of the proposed algorithms, the number of processes is 1 in MKmeans.

Table 3 reports the results of the aforementioned two algorithms, including the accuracy and time taken to form clusters.

Table 3. Comparison of algorithms

	Enhaced K-means		Exalt Parallel k-means	
Dataset Name	Accuracy (%)	Time taken (ms)	Accuracy (%)	Time taken (ms)
Glass	80.03	41	80.03	35
Diabetes	76.25	56	74.76	56
Ionosphere	86	60	96	52
Soybean	58.23	75	82.34	54
Super Market	78.34	88	80.45	72

5. CONCLUSIONS

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 faculty of Computer Science and Engineering department, Vignan University, Vadlamudi, A.P, India.

REFERENCES

[1] Sanjay Goil, Harasha Nagesh, Alok Choudhary, "MAFIA: Efficient and Scalable Subspace Clustering for Very Large Data Sets", 1999
[2] U.M. Fayyad, G Piatesky –Shapiro, P.Smyth, and R.Uthuusamy. "Advances in data mining and knowledge discovery. MIT Press", 1994
[3] M. Eirinaki and M. Vazirgiannis, "Web Mining for Web Personalization," ACM Transactions on Internet Technology (TOIT), vol. 3, no. 1, pp. 1-27, 2003
[4] 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
[5] E. Rasmussen, P. Willett, "Efficiency of hierarchic agglomerative clustering using the ICL distributed array processor," Journal of Documentation, vol. 45, March 1989.
[6] C. Olson, "Parallel algorithms for hierarchical clustering," Parallel Computing, vol. 21, no. 8, pp. 1313-1325, August 1995.
[7] W. Zhao, H. Ma, Q. He, "Parallel K-Means Clustering Based on Map Reduce," in: Cloud Computing, vol. 5931, pp. 674-679, 2009.
[8] Pang-Ning Tan, Michael Steinback and Vipin Kumar, Introduction to Data Mining, Pearson Education, 2007.
[9] T H Cormen, C E Leiserson, R L Rivest and C Stein, Introduction to Algorithms, Second Edition, MIT Press, 2001.
[10] Merz C and Murphy P, UCI Repository of Machine Learning Databases, Available: ftp://ftp.ics.uci.edu/pub/machine-learningdatabases