Parallel K-Means Algorithm on Agricultural Databases

V.Ramesh[1], K.Ramar[2], S.Babu[3]

[1,3]Assistant Professor,
Department of CSA, SCSVMV University, Kanchipuram, India

[2]Principal, Einstein College of Engineering, Tirunelveli, India

Abstract

A cluster is a collection of data objects that are similar to each other and dissimilar to the data objects in other clusters. K-means algorithm has been used in many clustering work because of the ease of the algorithm. But time complexity of algorithm remains expensive when it applied on large datasets. To improve the time complexity, we implemented parallel k-means algorithm for cluster large dataset. For our study we take agricultural datasets because of limited researches are done in agricultural field.

Keywords*: Clustering,k-means,parallel k-means, agriculture*

1. Introduction

Clustering is grouping input data sets into subsets called 'clusters' within which the elements are somewhat similar. In general, clustering is an unsupervised learning task as very little or no prior knowledge is given except input data sets. The tasks have been used in many fields and therefore various clustering algorithms have been developed. Clustering task is however, computationally expensive as many of the algorithms require iterative or recursive procedures and most of real-life data is high dimensional. Therefore, the parallelization of clustering algorithms is inevitable and various parallel clustering algorithms have been implemented and applied to many applications.

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.

There are different ways to classify parallel computers. One of the more widely used classifications in use since 1972 is called Flynn's taxonomy. Flynn's taxonomy distinguishes multi-processor computer architectures according to how they can be classified along the two independent dimensions of Instruction and Data. Each of these dimension have only one of two possible states, single or multiple. According to Flynn, the matrix given below defines the four possible classifications.

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

A Single Instruction, Single Data (SISD) machine is a traditional sequential computer that provides no parallelism in hardware. Instructions are executed in a serial fashion. One only data stream is processed by the CPU during a given clock cycle. A Multiple Instruction, Single Data (MISD) machine is capable of processing a single data stream using multiple instruction streams simultaneously. A Single Instruction, Multiple Data (SIMD) machine is one in which a single instruction stream has the ability to process multiple data streams simultaneously. A Multiple Instruction, Multiple Data(MIMD) machine is capable of is executing multiple instruction streams, while working on a separate and independent data stream. Given that modern computing machines are either the SIMD or MIMD machines, software developers have the ability to exploit data-level and task level parallelism in software.

Talia[1] identified three main strategies in the parallelism used in data mining algorithms as the following: (1) Independent parallelism where each processor accesses to the whole data to operate but do not communicate each other. (2) Task parallelism where each processor operate different algorithms on the partitioned or on the whole data set. (3) SPMD (Single Program Multiple Data) parallelism where multiple processors execute the same algorithm on different subsets and exchange the partial results to co-operate each other. Most of the parallel clustering algorithms follow the combinations of task and SPMD parallelism with Master – Slave Architecture.

Our study is based on the Single Program Multiple Data (SPMD) model using message-passing which is currently the most prevalent model for computing on distributed memory multiprocessors. Many applications for clustering algorithms, particularly applications in data mining, usually require the algorithms to work on massive data sets with an acceptable speed. For instance, in [2] NASA launches satellites for studying the earth's ecosystem. The Earth Observing System (EOS) is capable of generating about a terabyte of data per day. These terabytes of data will then be used to identify anomalies on earth by a visualization program. A grouping of such data sets could be done by clustering algorithms. The k-means and Apriori data mining algorithm was implemented with the GPUMiner system [3]. The CPUMiner system is a one of the post useful system for data mining and data clustering. The researchers have used three modules of the

GPUMininer. The performances of these algorithms were improved significantly, and the computational speedup is also improved significantly. Judd *et al.*[4] designed and implemented a parallel clustering algorithm for a network of workstations. They used a client-server approach where a block of work assignments were sent to each client process, which then calculates the block partial sum of each cluster and send the results back to the server. The server collects the partial sums from all clients, calculates the new centroids and returns the new centroids to all clients to begin a new iteration. They used parallel virtual machine and message Passing Interface for their implementation.

The main objective of this paper is compare the performance of commonly used classical k-means clustering procedures as well as parallel k-means clustering to realize the advantage of the parallelism of algorithm on agricultural data sets. The present investigation has been taken up to achieve the above mentioned goal.

2. Parallel k-means Algorithm

2.1. k-Means Algorithm

The k-means method has been shown to be effective in producing good clustering results for many practical applications. K-means method is well known for its relatively simple implementation and decent results. However, a direct algorithm of k-means method requires time proportional to the product of number of documents (vectors) and number of clusters per iteration. This is computationally very expensive especially for large datasets.

The algorithm is an iterative procedure and requires that the number of clusters k be given a priory. Suppose that the k initial cluster centers are given, and then the algorithm follows the steps as below :

1. Compute the Euclidean distance from each of the documents to each cluster center. A document is associated with a cluster such that its distance from that cluster is the smallest among all such distances.
2. After this assignment or association of all the documents with one of k clusters is done, each cluster center is recomputed so as to reflect the true mean of its constituent documents.
3. Steps 1 and 2 are repeated until the convergence is achieved.

Suppose there are n data points or documents, X_1, X_2,… X_n are given such that each one of them belongs to R^d . The problem of finding the minimum variance clustering of this dataset into k clusters is that of finding the k points $\{m_j\}_{j=1}^{k}$ in R^d such that

$$(1/n) * \Sigma(\min_j d^2(X_i , m_j)), \text{ for } i = 1 \text{ to } n$$

is minimized, where $d(X_i , m_j)$ denotes the Euclidean distance between X_i and m_j. The reason for popularity of k-means algorithm is ease of interpretation, simplicity of implementation, scalability, speed of convergence and adaptability to sparse data. The k-means algorithm can simply be explained as follows :

1. Phase Initialization

Select a set of k starting points $\{m_j\}_{j=1}^{k}$ in R^d . This selection may be done in a random manner or making use of some heuristic.

2. Phase Distance Calculation

For each data point X_i $1 \le i \le n$, compute its Euclidean distance to each cluster m_j $1 \le j \le k$ and then find the closest cluster centroid.

3. Phase Centroid Recalculation

For each $1 \le j \le k$, recompute cluster centroid m_j as the average of data points assigned to it.

4. Convergence Condition

Repeat steps 2 and 3 until convergence.

2.2. parallel k-means Algorithm

In contrast, the parallel k-means algorithm was developed for cluster analysis of very large data sets. The parallel algorithm can scale a problem size upto O(K) times the size of the problem on a single machine[6]. The implementation was done in the Java using Message Passing Interface(MPI) for distributed memory parallelism. The dataset to be clustered is divided among the available system. The initial centroid is selected by one system and is broad cast to all the system. Every process operates only on its chunk of the dataset, carries out the distance calculation of points from the centroids and assigns it to the closest centroid. Each process also calculates partial sum along each dimension of the points in each cluster for its chunk for the centroid calculation. At the end of the iteration, a global reduction operation is carried out, after each process obtains the information, to calculate the new cluster centroids. Iterations are carried out until convergence, after which the cluster assignments are written to an output file. A simple life cycle of the program can be listed as :

1. Master reads the data and divides it into N portions and sends one of them to each slave.
2. Master randomly initializes the centroids.
3. Master sends all of the centroids to all of the slaves.
4. Each slave receives a portion of dataset along with the centroids from master.
5. Each slave calculates the cluster membership of the data points assigned to it, and sends the results back to the master.
6. Master calculates new cluster centers by taking the means.
7. If converged, the process terminates, otherwise the master once again send the new centroids to the slaves for to calculate cluster membership for the new centroids.

3. Applications

Performance of the parallel k-means algorithm and implementation was evaluated using soil datasets which was collected from the department of Agriculture, Government of Tamil Nadu. By using parallel k-means clustering, the soil of Tamil Nadu was clustered according to their soil characteristics. The soil characteristics such as pH, EC, available nutrients like N, P, K, Zn, Fe, Mn, Cu, B and other features like Soil textures, and Lime status are considered as attributes for clustering.

A tool was developed using Java for the implementation of parallel k-means clustering. ServerKmeans is a jar file to do the master process. ClientKmeans is a jar file to run in client/slave systems. (Fig.1). ClientKmeans is installed and in running condition in all clients.

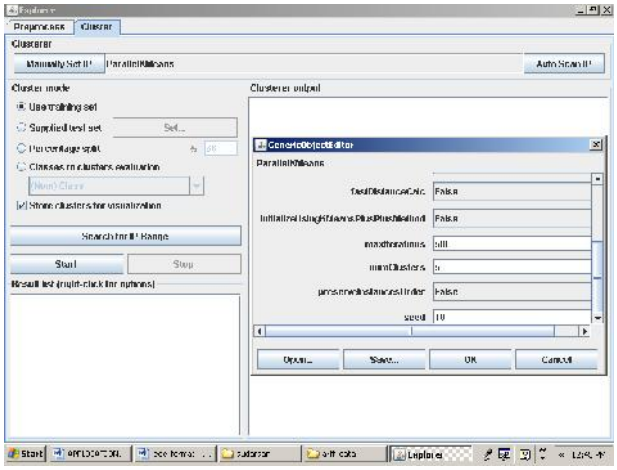


Fig.-1 ServerKmeans accepts Number of cluster points

ServerKmeans accepts the IP numbers of nodes connected with the master(Fig. 2).

Fig.-2 Accepts IP number of clients

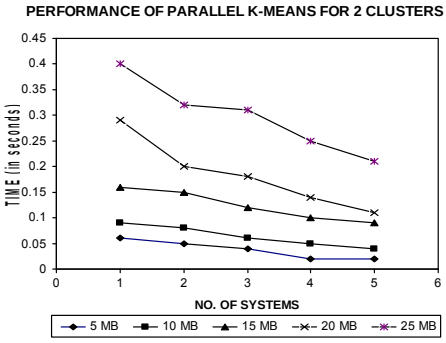
After getting IP address of the each slaves, the master sends the partitioned data and centroids to the clients.

4. Results and Discussion

We run the program in Pentium I5 with 2 GB ROM available in SCSVMV University, Kanchipuram Computer Centre. We used maximum of ten systems for our study. We tested our tool with some UCI datasets also. The performance of the parallel k-means algorithms was tested proved having a better performance compared with sequential k-means. Initially the number of clusters required is set as two. The time taken to cluster the different sizes of data is given in Table 1.

Table 1. Time complexity for the two clusters

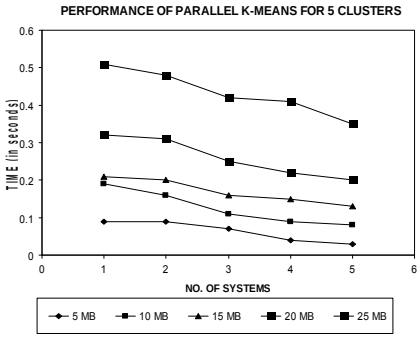
No. of Slaves / Data Size	1	2	3	4	5
5 MB	0.06	0.05	0.04	0.02	0.02
10 MB	0.09	0.08	0.06	0.05	0.04
15 MB	0.16	0.15	0.12	0.1	0.09
20 MB	0.29	0.2	0.18	0.14	.011
25 MB	0.4	0.32	0.31	0.25	0.21



The number of required clusters is also increased to five. The time taken to cluster the different sizes of data for the given cluster=5 is given in Table 2.

Table 2 Time complexity for the five clusters

No. of Slaves / Data Size	1	2	3	4	5
5 MB	0.09	0.09	0.07	0.04	0.03
10 MB	0.19	0.16	0.11	0.09	0.08
15 MB	0.21	0.2	0.16	0.15	0.13
20 MB	0.32	0.31	0.25	0.22	0.2
25 MB	0.51	0.48	0.42	0.41	0.35



We can conclude that our parallel k-means algorithm achieves more efficiency and time complexity than the normal sequential k-means algorithm. The processing speed is also disturbed with the bandwidth of the network available in the centre. However, our tool can be used to cluster the very large data sets in any field.

Acknowledgements

We would like to thank the Department of Computer Science & Engineering and Computer Science & Applications at SCSVMV University, Kanchipuram for allowing us to use the computer facilities for our experiments.

References

[1] Domenico Talia, Parallelism in Knowledge Discovery Techniques, PARA'02 Proceedings of the 6th International Conference on Applied Parallel Computing and Advanced Scientific Computing, pages 127-138, London,U.K

[2] K.Assiter, K.P. Lentz, A. Couch, and C.Currey, "Locating Anomalies in Large Data Sets", Society for Computer Simulation Military, Government, and Aerospace Simulation, April 5, 1998, pp. 218-223

[3] Wenbin Fang, Ka Keung Lau, Mian Lu, Xiangye Xiao, Chi Kit Lam, Philip Yang Yang, Bingsheng He, Quong Luo, Pedro V.Sander, and Ke Yang, "Parallel Data Mining on Graphics Processors", Technical Report HKUSTCS0807 (Oct.2008).

[4] D.Judd, P.McKinley, A. Jain, Larger-scale parallel data clustering, Pattern Analysis and Machine Intelligenc,e IEEE Transactions on 20 (8) (1998) 871-876.

[5] Sanpawat Kantabutra and Alva L.Couch, Parallel k-means Clustering Algorithm on NOWs, Technical Journal, Vol.1, No.6, January – Feburary 2000.

[6] Wooyoung Kim – Parallel Clustering Algorithms : Survey, Spring, 2009.