Using Parallel Computing to Performing the Cluster Analysis(K-means)

CS566 – Parallel Computing

Rajarajan Loganathan
Hood College
Department of Computer Science
Frederick, MD, USA

Abstract-**Clustering is one of the most popular methods for exploratory data analysis, which is prevalent in many disciplines such as image segmentation, bioinformatics, pattern recognition and statistics etc. The most famous clustering algorithm is K-means because of its easy implementation, simplicity, efficiency and empirical success. However, the real-world applications produce huge volumes of data, thus, how to efficiently handle of these data in an important mining task has been a challenging and significant issue. In addition, MPI (Message Passing Interface) as a programming model of message passing presents high performances, scalability and portability. Motivated by this, a parallel K-means clustering algorithm with MPI, called M-Kmeans, is proposed in this paper. The algorithm enables applying the clustering algorithm effectively in the parallel environment. Experimental study demonstrates that M-Kmeans is relatively stable and portable, and it performs with low overhead of time on large volumes of data sets.**

Keywords – Kmean Clustering model, MPI, parallel processing

I.Introduction:

Data clustering is a data exploration technique that allows objects with similar characteristics to be grouped together in order to facilitate their further processing. Data clustering has many engineering applications including the identification of part families for cellular manufacture. The K-means algorithm is a popular data clustering algorithm. To use it requires the number of clusters in the data to be pre-specified. Finding the appropriate number of clusters for a given data set is generally a trial-and-error process made more difficult by the subjective nature of deciding what constitutes 'correct' clustering . This paper proposes a method based on information obtained during the K-means clustering operation itself to select the number of clusters, K. The method employs an objective evaluation measure to suggest suitable values for K, thus avoiding the need for trial and error. The remainder of the paper consists of five sections. Performing K-Mean algorithm using parallel processing or computing. Parallel data mining algorithms have been recently considered for tasks such as association rules and classification, see, for example, Agrawal and Shafer [6], Chattratichat et al. [7], Cheung and Xiao [8], Han, Karypis, and Kumar [9].

II.Parallel k-means :

Our parallel algorithm design 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; we now briefly review this model.

III.Message-Passing Model of Parallel Computing:

We assume that we have P processors each with a local memory. We also assume that these processors are connected using a communication network. We do not assume a specific interconnection topology for the communication network, but only assume that it is generally cheaper for a processor to access its own local memory than to communicate with another processor.

The message-passing model posits a set of processes that have only local memory but are able to communicate with other processes by sending and receiving messages. It is a defining feature of the message-passing model that data transfers from the local memory of one process to the local memory of another process require operations to be performed by both processes.

MPI, the Message Passing Interface, is a standardized, portable, and widely available message-passing system designed by a group of researchers from academia and industry [1, 2]. MPI is robust, efficient, and simple-to-use from JAVA.

From a programmer's perspective, parallel computing using MPI appears as follows. The programmer writes a single program in JAVA compiles it, and links it using the MPI library jar file. The resulting object code is

Sequential Algorithms	Parallel Algorithm
1: 2: 3: MSE = LargeNumber; 4: 5: Select initial cluster centroids $\{m_j\}_j^k = 1$; 6: 7: 8: do { 9: OldMSE = MSE; 10: MSE' = 0; 11: for j = 1 to k 12: m'_j = 0; n'_j= 0; 13: endfor; 14: for i = 1 to n 15: for j = 1 to k 16: compute squared Euclidean distance $d^2(X_{i+}m_j)$; 17: endfor; 18: find the closest centroid mℓ to X_i; 19: m'_ℓ= m'_ℓ + X_i ; n'_ℓ= n'_ℓ+ 1; 20: MSE' = MSE' + $d^2(X_{i+}m_j)$; 21: endfor; 22: for j = 1 to k 23: 24: 25: nj = $max(n'_j, 1)$; $mj = m'_j$/n; 26: endfor; 27: MSE = MSE' ; 28:} while (MSE < OldMSE)	1: P = MPI Comm size (); 2: μ = MPI Comm rank (); 3: MSE = LargeNumber; 4: if (μ = 0) 5: Select initial cluster centroids $\{m_j\}_j^k = 1$; 6: endif; 7: MPI Bcast ($\{m_j\}_j^k = 1,0$) 8: do { 9: OldMSE = MSE; 10: MSE' = 0; 11: for j = 1 to k 12: m'_j = 0; n'_j = 0; 13: endfor; 14: for i = μ ∗ (n/P) + 1 to (μ + 1) ∗ (n/P) 15: for j = 1 to k 16: compute squared Euclidean distance $d^2(X_{i+}m_j)$; 17: endfor; 18: find the closest centroid m_ℓ to X_i; 19: m'_ℓ= m'_ℓ + X_i ; n'_ℓ= n'_ℓ+ 1; 20: MSE' = MSE' + $d^2(X_{i+}m_j)$; 21: endfor; 22: for j = 1 to k 23: MPI_Allreduce (n'_j, n_j , MPI_SUM); 24: MPI_Allreduce (m'_j , m_j , MPI_SUM); 25: n_j = max(n_j , 1); $m_j = m_j/n_j$; 26: endfor; 27: MPI_Allreduce (MSE' , MSE, MPI_SUM);

	28:} while (MSE < OldMSE)

loaded in the local memory of every processor taking part in the computation; thus creating P "parallel" processes. Each process is assigned a unique identifier between 0 and P − 1. Depending on its processor identifier, each process may follow a distinct execution path through the same code. These processes may communicate with each other by calling appropriate routines in the MPI library which encapsulates the details of communications between various processors.

Table 1 gives a glossary of various MPI routines which we use in our parallel version of k-means in Figure 2. Next, we discuss the design of the proposed parallel algorithm

MPI_Comm_size()	Returns the number of processes
MPI_Comm_rank()	Returns the process identifier for the calling process
MPI_Allreduce(A, B, MPI_SUM)	Sums all the local copies of "A" in all the processes (reduction operation) and places the result in "B" on all of the processes (broadcast operation)
MPI_Wtime()	Returns the number of seconds since some fixed, arbitrary point of time in the past
MPI_Bcast(message, root)	Broadcasts "message" from a process with identifier "root" to all of the processes
MPI_Send()	Sending the values
MPI_Recieve()	Receive the values

IV.Parallel Algorithm Design:

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.

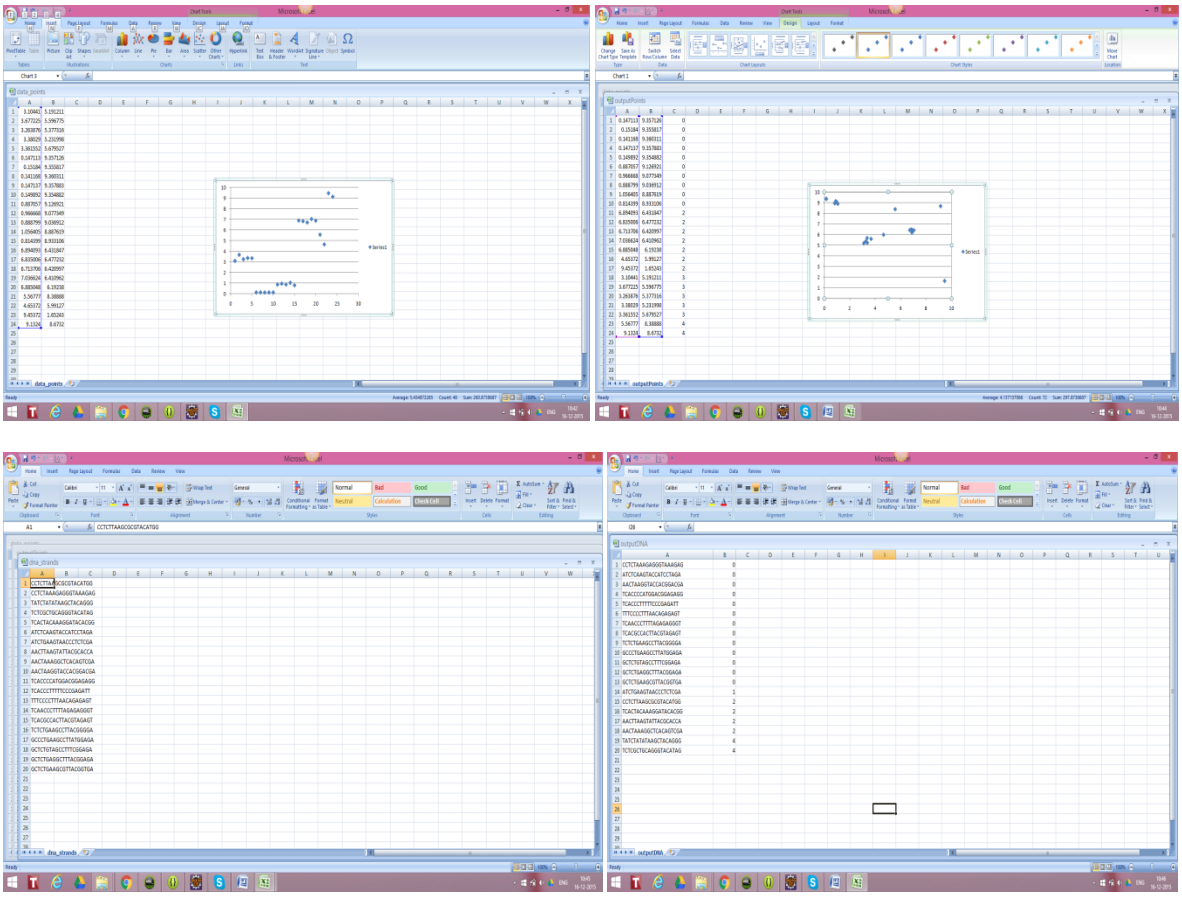
4. Each slave receives a portion of dataset along with the centroids from master.

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.

V.Sample Input and Output:



VI.Results and Discussion:

As data The Simulation was performed on the Computer.

Processor: Intel i7 8 cores 2.5 GHz

RAM: 16 GB

System: Windows 8

VII.Conclusion:

As data clustering has attracted a significant amount of research attention, many clustering algorithms have been proposed in the past decades. However, enlarging data in applications makes clustering of very large scale of data a challenging task. In this paper, we propose a fast parallel k-means clustering algorithm based on the above, which has been widely embraced by both academia and industry. We use speedup, scaleup and sizeup to evaluate the performances of our proposed algorithm. The results show that the

proposed algorithm can process large datasets on commodity hardware effectively.

VIII.Reference:

[1].D.Fati,italy in Clustering data mining on multi-cores
http://grid.cs.gsu.edu/~wkim/index_files/papers/scalable_parallel_datamining.pdf

[2].Wooyoung king,Parallel Clustering algorithms
http://grid.cs.gsu.edu/~wkim/index_files/SurveyParallelClustering.html#dc_review

[3].Data clustering Review ,Ohio state University
http://grid.cs.gsu.edu/~wkim/index_files/papers/dataclustering_review.pdf

[4].parallel K-mean clustering Algorithm
http://grid.cs.gsu.edu/~wkim/index_files/papers/parallel_kmeans_now.pdf

[5].Selection of k in k-means clustering ,university of cardiff ,UK
https://www.ee.columbia.edu/~dpwe/papers/PhamDN05-kmeans.pdf

[6]. Agrawal, R., Shafer, J.C.: Parallel mining of association rules: Design, implementation, and experience. IEEE Trans. Knowledge and Data Eng. 8 (1996) 962–969

[7]. Chattratichat, J., Darlington, J., Ghanem, M., Guo, Y., H¨uning, H., K¨ohler, M., Sutiwaraphun, J., To, H.W., Yang, D.: Large scale data mining: Challenges and responses. In Pregibon, D., Uthurusamy, R., eds.: Proceedings Third International Conference on Knowledge Discovery and Data Mining, Newport Beach, CA, AAAI Press (1997) 61–64

[8]. Cheung, D.W., Xiao, Y.: Effect of data distribution in parallel mining of associations. Data Mining and Knowledge Discovery (1999) to appear.

[9]. Han, E.H., Karypis, G., Kumar, V.: Scalable parallel data mining for association rules. In: SIGMOD Record: Proceedings of the 1997 ACM-SIGMOD Conference on Management of Data, Tucson, AZ, USA. (1997) 277–288

[10] Gropp, W., Lusk, E., Skjellum, A.: Using MPI: Portable Parallel Programming with the Message Passing Interface. The MIT Press, Cambridge, MA (1996).

[11]. Snir, M., Otto, S.W., Huss-Lederman, S., Walker, D.W., Dongarra, J.: MPI: The Complete Reference. The MIT Press, Cambridge, MA (1997).