An Efficient Parallel K-Means On Multi-Core Processors

Ahmed M. Fahim

Abstract— **Nowadays, all most personal computers have multi-core processors. We try to exploit computational power from the multi-core architecture. We need a new design on existing algorithms and software. In this paper, we propose the parallelization of the well-known k-means clustering algorithm. We employ Parallel for-Loops (parfor) in MATLAB. Where a loop of *n* iterations could run on a cluster of *m* MATLAB workers simultaneously, each worker executes only *n/m* iterations of the loop. Also we apply parallel for on an enhanced version of k-means method. The experimental results demonstrate considerable speedup rate of the proposed parallel k-means clustering method run on a multicore/multiprocessor machine, compared to the serial k-means approach.**

Keywords— **Clustering Algorithms, High Performance Computing, K-means Algorithm, Parallel Computing.**

I. INTRODUCTION

The huge amount of data collected and stored in databases increases the need for effective analysis methods to use the information contained implicitly there. One of the primary data analysis tasks is cluster analysis, intended to help a user understand the natural grouping or structure in a dataset. Therefore, the development of improved clustering algorithms has received much attention. The goal of a clustering algorithm is to group the objects of a database into a set of meaningful subclasses [6]. Clustering is the process of partitioning or grouping a given set of patterns into disjoint clusters. This is done such that patterns in the same cluster are alike, and patterns belonging to two different clusters are different. Clustering has been a widely studied problem in a variety of application domains including data mining and knowledge discovery [4], data compression and vector quantization [7], pattern recognition and pattern classification [2], neural networks, artificial intelligence, and statistics. Many clustering algorithms proposed, but the most widely used one is the k-means method [13].

The popularity of k-means algorithm is due to its linear computational complexity that is O(nkt), where n is the number of data points or objects, k is the number of desired clusters, and t is the number of iterations the algorithm takes for converging to a stable state. The computing process needs improvements to efficiently apply the method to applications with huge number of data objects such as genome data analysis and geographical information systems.

Parallelization is one of the obvious solution to this problem and many researchers have proposed the idea many years ago such as [12] [8] [9] [22] [23] [15]. This paper also focuses on parallelizing k-means algorithm, but we base our study on the multi-core architecture. We implement our extension of the k-means algorithm using parallel MATLAB, where we use parfor loop, part of the parfor body is executed on the MATLAB client (where the parfor is issued) and part is executed in parallel on MATLAB workers working together as a parallel pool. The necessary data on which parfor operates is sent from the client to workers, where most of the computation happens, and the results are sent back to the client and pieced together. Because several MATLAB workers can be computing concurrently on the same loop, a parfor-loop can provide significantly better performance than its analogous for-loop [14].

We are interested in developing k-means algorithm because it is simple and widely used in practice. In addition, it is ideal algorithm for using parallel for in MATLAB. Our contribution in this paper is to significantly reduce time complexity of the serial k-means algorithm by data parallelism. Also we apply parallelism on an enhanced version of k-means. We compared among serial and parallel and enhanced parallel k-means. So we can say that, this paper introduces three different algorithms about k-means method. The enhanced version of k-means is presented in [6]. The rest of the paper is organized as follows. Discussion of related work in parallel k-means is presented in Section ii. Our proposed algorithm; an efficient parallel k-means method is explained in Section iii. Some experimental results are demonstrated in Section iv. The conclusion appears as the last section of the paper.

II. RELATED WORK

A serial k-means algorithm was proposed in 1967 [13] and since then it has gained great interest from data analysts and computer scientists. The algorithm has been applied to variety of applications ranging from medical informatics [10], genome analysis [15], image processing and segmentation [20], [21], to aspect mining in software design [1]. Despite its simplicity and great success, the k-means algorithm is known to degrade when the dataset grows larger in terms of number of objects and dimensions [8], [11]. To obtain acceptable computational speed on huge datasets, most researchers turn to parallelizing scheme.

Li and Fang [12] are among the pioneer groups on studying parallel clustering. They proposed a parallel algorithm on a

***Ahmed M. Fahim**, Computer Science Dept., Faculty of Science, Suez University, (e-mail: ahmmedfahim@yahoo.com). Suez, Egypt.*

Checked at Department of computer science, Faculty of Science and Human Studies, Prince Sattam bin Abdul Aziz University, Aflaj, KSA. +966568994864.

ISSN: 2278 – 7798

single instruction multiple data (SIMD) architecture. Dhillon and Modha [3] proposed a distributed k-means that runs on a multiprocessor environment. Kantabutra and Couch [9] proposed a master-slave single program multiple data (SPMD) approach on a network of workstations to parallel the k-means algorithm. Their experimental results reveal that when on clustering four groups of two-dimensional data the speedup advantage can be obtained when the number of data is larger than 600,000, and their maximum speedup was 2.1, it is very small value. Tian and colleagues [19] proposed the method for initial cluster center selection and the design of parallel k-means algorithm. Stoffel and Belkoniene proposed parallel k-means works on a distributed database, the database was distributed over a network of 32 PCs, their results revealed that as the number of node increase the speedup degrades because of the increase of communication overhead and the variations in the execution times of the different processors [17].

Zhang and colleagues [23] presented the parallel k-means with dynamic load balance that used the master/slave model. Their method can gain speedup advantage at the two-dimensional data of size greater than 700,000. Prasad [16] parallelized the k-means algorithm on a distributed memory multi-processors using the message passing scheme. Farivar and colleagues [5] studied parallelism using the graphic coprocessors to reduce energy consumption of the main processor.

Zhao and colleagues [22] proposed parallel k-means method based on map and reduce functions to parallelize the computation across machines. Tirumala Rao and colleagues [18] studied memory mapping performance on multi-core processors of k-means algorithm. They conducted experiments on quad-core and dual-core shared memory architecture using OpenMP and POSIX threads. The speedup on parallel clustering is observable.

A. Serial and Enhanced K-means

The serial k-means algorithm select k initial centers as representatives for clusters and assign each data point p to the closest cluster according to distances among p and all centers, and computes new centers, then start new iteration, to redistribute each data point p to the new centers, and computes new centers and so on until no data point change its cluster or some threshold met. To assign data point p to cluster it computes the distances between this point p and all cluster centers. This is the most time consuming operation. An enhanced k-mean solve this problem depending on the fact that cluster center moves to center of gravity of cluster. i.e center moves toward the highest density region in cluster. So the point in this region will not change its cluster. So it compute the distance between point p and the new center of previous cluster, if the point p become closer to the new center then the point stay in its previous cluster and there is no need to compute its distance from the other(k-1) centers. This improves the execution time of the algorithm. For more details see [6].

In this paper, we apply parallelism on these two versions of k-means algorithm; the experimental results reveal great improvement in execution time of the k-means method.

III. THE PROPOSED ALGORITHM

Given a data set containing n objects, k-means partitions these objects into k groups. Each group is represented by the centroid, or central point, of the cluster. Once cluster means or representatives are selected, data objects are assigned to the nearest centers. The algorithm iteratively finds new better representatives and reassigns data objects until the stable condition has been reached. The stable condition can be observed from cluster assigning that each data object does not change its cluster.

The serial k-means algorithm [13], shown in Fig. 1, takes much computational time on calculating distances between each of n data points and the current k centers. Then iteratively assign each data point to the closest cluster.

```
Serial k-means(DS,k)
    1. Select initial k centers
    2. Repeat
    3. For i=1 to n
    4. Calculates the distances between the current
         point and k centers
    5. Endfor
    6. Assign each point to its nearest cluster
    7. Calculate the new k centers
    8. Until stable k centers reached
```

Fig. 1 Serial K-means Algorithm.

The parallel k-means algorithm concentrates on distance calculation between each point and the k centers, performs these calculations in parallel way. If we have m cores and n data points then each core will approximately calculates the distances between n/m points and k centers. As m increase, the amount of calculation per each core will decrease. Since these calculations are iterated t times until each point stay in its cluster, this parallel paradigm significantly improves running time of the k-means algorithm. The parallel k-means is shown in Fig. 2.

```
Parallel k-means(DS,k)
    1. Select initial k centers
    2. Repeat
    3. parfor i=1 to n
    4. Calculates the distances between the current point
         and k centers
    5. Endparfor
    6. Assign each point to its nearest cluster
    7. Calculate the new k centers
    8. Until stable k centers reached
```

Fig. 2 Parallel K-means Algorithm.

Because the distance between point i and k centers is completely independent of the distance between point j and k centers, so these calculation can be performed in parallel using parallel for in MATLAB. To run code that contains a parallel loop, we first open MATLAB pool. This reserves a collection of MATLAB worker sessions to run loop

ISSN: 2278 – 7798

iterations. The MATLAB pool can consist of MATLAB sessions running on local machine or on a remote cluster. Because the iterations run in parallel in other MATLAB sessions, each iteration must be completely independent of all other iterations. The workers calculating the distance between point *i* and *k* centers might not be the same workers calculating the distance between point *j* and *k* centers. There is no guarantee of sequence [14].

The efficient parallel k-means shown in Fig. 3 concentrates on the idea presented in [6] that decreases the amount of distance calculation since not all points change their clusters in each iteration, and these calculations decrease in each iteration of the main steps of the algorithm. This idea is implemented in step 8 of Fig. 3.

Efficient Parallel k-means(*DS*,*k*)
1. Select initial *k* centers
2. Calculates the distances between the current point and *k* centers
3. Assign each point to its nearest cluster
4. Calculate the new *k* centers
5. Repeat
6. parfor *i*=1 to *n*
7. calculate distance between point *i* and the new center of previous cluster
8. if this distance > the previous distance
   Calculates the distances between the current point and *k* centers
   Else
   Point stay in its cluster, and store the new distance.
   endif
9. Endparfor
10. Assign each point to its nearest cluster
11. Calculate the new *k* centers
12. Until stable *k* centers reached

Fig. 3 Efficient Parallel K-means Algorithm.

IV. EXPERIMENTAL RESULTS

We implement these three versions of k-means using MATLAB R2009b language. The code is executed on dell inspiron 1525 Laptop, Intel(R) Core(TM)2 Duo Processor 2.00 GHz, 2 MB cache memory, 2GB RAM, 32-bit Windows 7 Ultimate. We generated 13 synthetic two-dimensional datasets containing form 10000 to 400000 data points with random values grouped into 100 clusters as shown in Fig. 4.

We evaluate performances of the proposed algorithm (Efficient Parallel K-means) on synthetic two-dimensional datasets containing hundred clusters, run concurrently on two lab (session or core). The computational speed of efficient parallel k-means, and parallel k-means as compared to serial k-means is given in Table1 where we set *k* to 100 cluster. Ts refers to execution time of serial k-means, Tp refers to execution time of parallel k-means, Tep refers to execution time of efficient parallel k-means, TDp refers to time difference between serial and parallel k-means, and TDep refers to time difference between serial and efficient parallel k-means. The speedup is calculated according to the following equations.

$$speedup\ 1 = \frac{Ts}{Tp}. \qquad (1)$$

$$speedup\ 2 = \frac{Ts}{Tep}. \qquad (2)$$

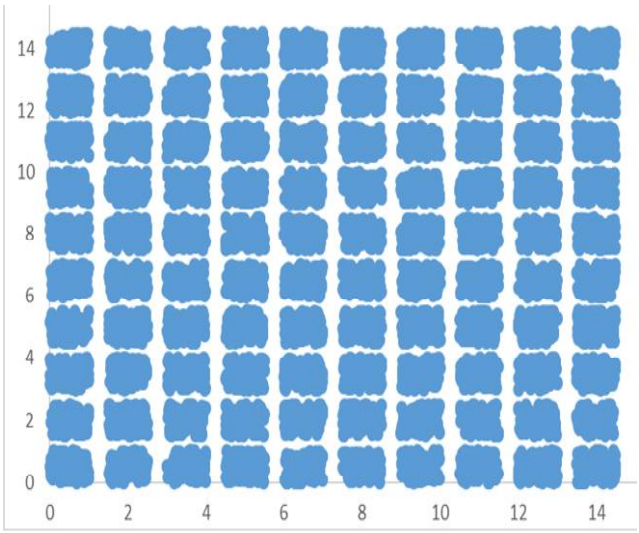

Fig. 4 Synthetic Dataset Containing 100 Clusters.

Linear speedup or ideal speedup is obtained when speed up is equal to number of processors (cores). In our case, linear speedup is equal to two.

If we compare the results shown in Table I with that of [9] and [23] we get better results from our proposed algorithm, because we get speedup form dataset of size 10000 object compared with dataset of size greater than 600000 and 700000 in [9] and [23] respectively.

Running time comparison of parallel, efficient parallel against serial k-means is graphically shown in Fig. 5 where k=100

TABLE I. THE EXECUTION TIME OF SERIAL K-MEANS, PARALLEL K-MEANS, AND EFFICIENT PARALLEL K-MEANS WHERE *K*=100 ON SYNTHETIC DATA SETS.

Dataset size	Ts	Tp	Tep	TDp:	TDep:	Speedup1 Ts/Tp	Speedup2 Ts/Tep
10000	10.80	6.06	3.50	4.74	7.29	1.78	3.08

ISSN: 2278 – 7798

20000	16.01	9.21	6.58	6.80	9.43	1.74	2.43
30000	32.23	18.92	9.30	13.31	22.93	1.70	3.47
40000	43.64	29.81	12.54	13.83	31.10	1.46	3.48
50000	53.77	37.02	17.48	16.75	36.29	1.45	3.08
60000	64.62	39.80	19.02	24.82	45.60	1.62	3.40
70000	77.82	50.13	21.30	27.68	56.52	1.55	3.65
80000	91.96	61.96	25.57	30.00	66.39	1.48	3.60
90000	108.82	77.91	29.99	30.90	78.82	1.40	3.63
100000	83.85	65.15	35.03	18.70	48.82	1.29	2.39
200000	258.68	202.49	149.37	56.20	109.31	1.28	1.73
300000	315.13	224.43	173.31	90.70	141.81	1.40	1.82
400000	616.01	411.71	306.96	204.30	309.05	1.50	2.01

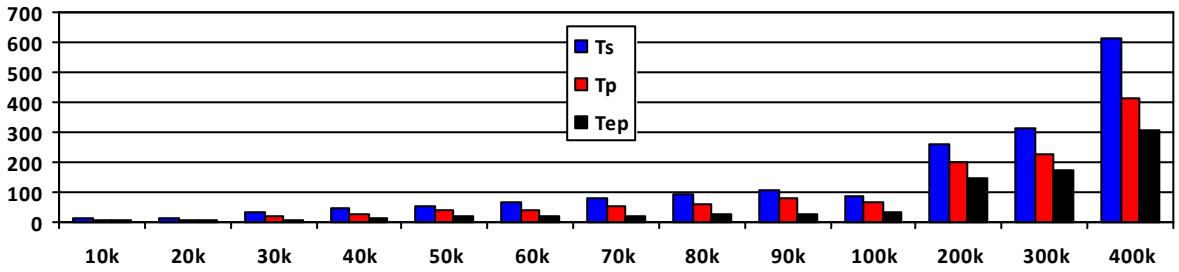

Fig. 5 Running time comparisons of serial, parallel, and enhanced parallel k-means where *k*=100 Clusters on different synthetic data sets.

Fig. 6 Final clusters of the three algorithms, where *k*=100.

Fig. 6 depicts the final clustering of the three version of k-means algorithm. Also, we experiment the algorithms using three real datasets in the pattern Recognition field.

The first dataset represents the image of English capital letters. The image consists of a large number of black-and-white rectangular pixels display one of the 26 capital letters in the English alphabet. The character images were based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of 20,000 unique stimuli. Each stimulus was converted into 16 primitive numerical attributes.

The second is abalone dataset. This dataset represents physical measurements of abalone. Each abalone is described with 7 attributes.

The third dataset is about wind. This dataset represents measurements about wind from 1/1/1961 to 31/12/1978. This observation of wind described by 15 attributes.

Tables from II to IV represent the execution time in second for all algorithms using different values for *k* in each data set.

TABLE II. THE EXECUTION TIME FOR ABALONE DATA SET FOR K FROM 100 TO 1000.

k	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**
Tep	13.71	20.53	24.99	31.94	39.71	47.99	49.52	78.26	96.97	46.78
Tp	17.26	24.30	32.15	42.48	57.05	62.55	80.36	105.29	131.47	73.74
Ts	25.28	41.00	56.17	74.00	91.10	103.15	120.70	138.05	202.03	112.19
Speedup1	1.46	1.69	1.75	1.74	1.60	1.65	1.50	1.31	1.54	1.52

ISSN: 2278 – 7798

Speedup2	1.84	2.00	2.25	2.32	2.29	2.15	2.44	1.76	2.08	2.40

TABLE III. THE EXECUTION TIME FOR LETTER DATA SET FOR K FROM 10 TO 100.

k	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
Tep	10.00	14.13	20.15	25.70	31.98	39.27	48.84	54.25	63.42	73.16
Tp	10.44	17.59	31.24	45.32	59.20	70.32	82.17	96.58	106.39	109.22
Ts	14.01	26.79	42.79	60.10	80.24	97.87	108.02	135.68	156.75	174.65
Speedup1	1.34	1.52	1.37	1.33	1.36	1.39	1.31	1.40	1.47	1.60
Speedup2	1.40	1.90	2.12	2.34	2.51	2.49	2.21	2.50	2.47	2.39

TABLE IV. THE EXECUTION TIME FOR WIND DATA SET FOR K FROM 100 TO 1000.

k	**20**	**40**	**60**	**80**	**100**	**120**	**140**	**160**
Tep	6.09	10.61	14.09	23.76	24.20	30.17	37.49	44.10
Tp	6.49	11.83	19.34	28.07	37.86	51.85	60.96	67.67
Ts	8.66	17.54	27.55	37.75	51.39	66.99	82.18	93.02
Speedup1	1.33	1.48	1.42	1.35	1.36	1.29	1.35	1.37
Speedup2	1.42	1.65	1.96	1.59	2.12	2.22	2.19	2.11

Figures from 7 to 9 depict the execution time of the three algorithms on the three real data sets.

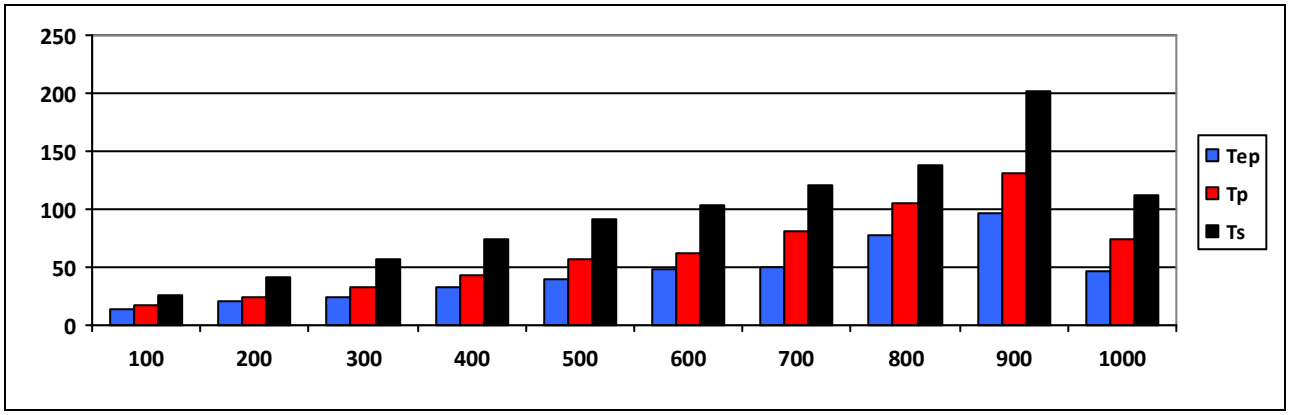

Fig. 7 Running time comparison of serial, parallel, and enhanced parallel k-means where *k* from100 to 1000 Clusters on abalone data set.

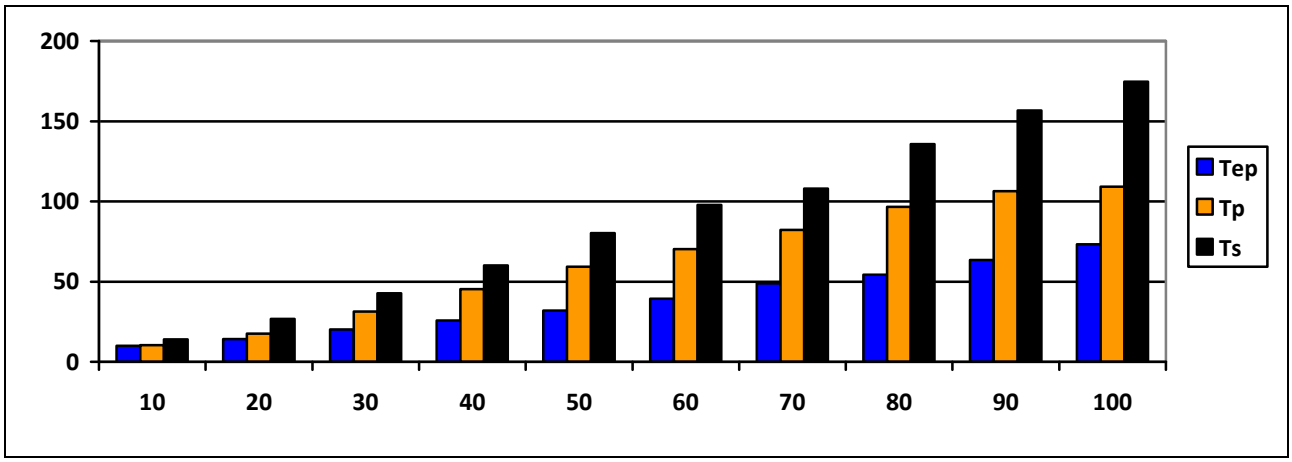

Fig. 8 Running time comparison of serial, parallel, and enhanced parallel k-means where *k* from10 to 100 Clusters on letter data set.

ISSN: 2278 – 7798

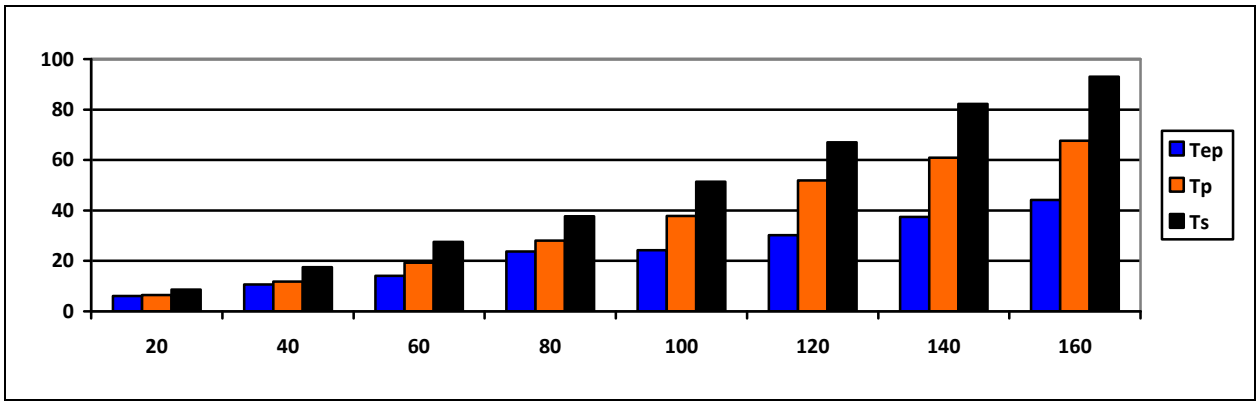

Fig. 9 Running time comparison of serial, parallel, and enhanced parallel k-means where *k* from20 to 160 Clusters on wind data set.

Figures from 10 to 12 show the speedup comparison between parallel and enhanced parallel k-means. These Figures show the great improvement in execution time of the k-means when implemented in efficient parallel method. Efficient parallel method is always superior. It is very fast method in producing the final result. Fig. 13 shows speedup comparison between parallel and enhanced parallel k-means on different synthetic data sets.

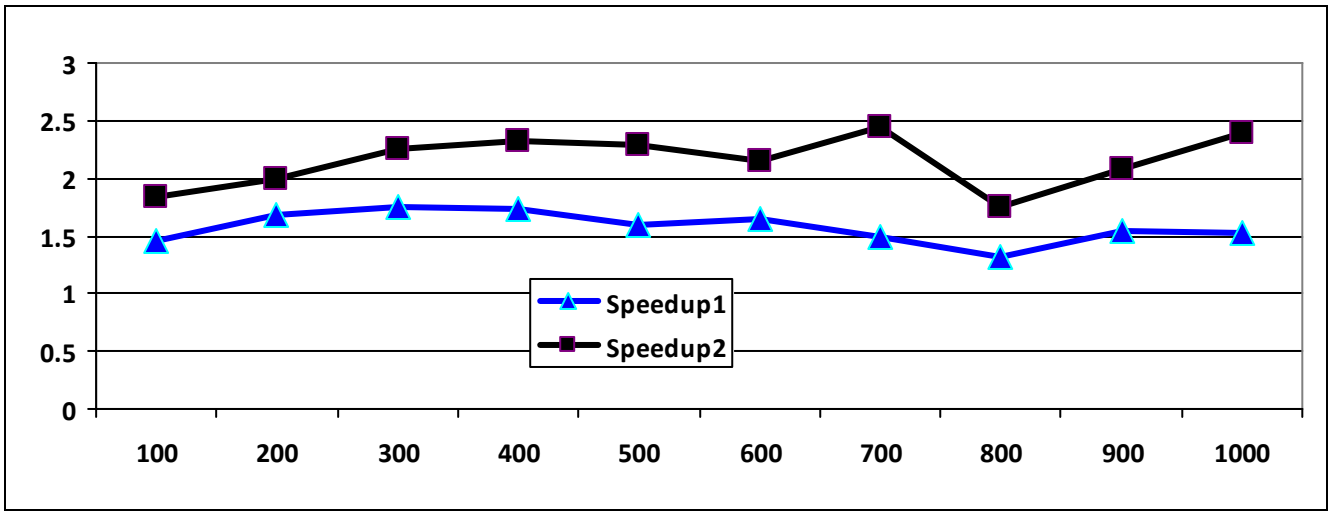

Fig. 10 Speedup comparison of parallel, and enhanced parallel k-means where k from10 to 1000 Clusters on abalone data set.

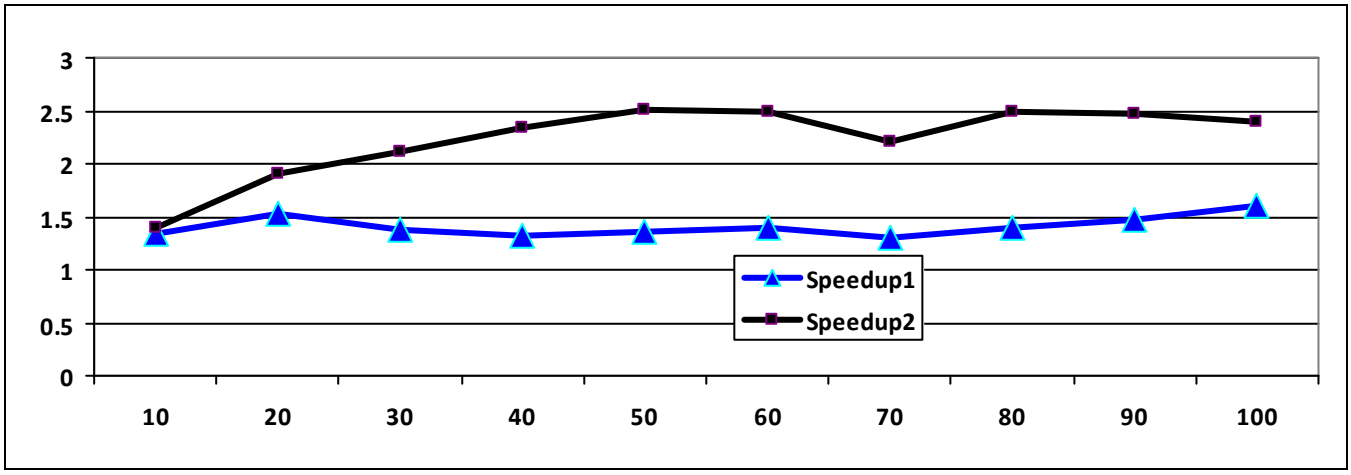

Fig. 11 Speedup comparison of serial, parallel, and enhanced parallel k-means where k from10 to 100 Clusters on letter data set.

ISSN: 2278 – 7798

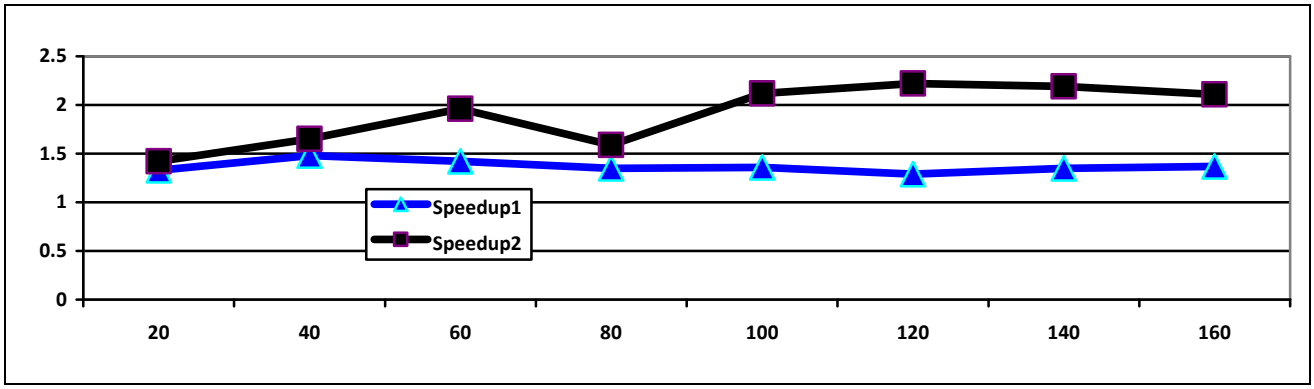

Fig. 12 Speedup comparison of serial, parallel, and enhanced parallel k-means where k from20 to 160 Clusters on wind data set.

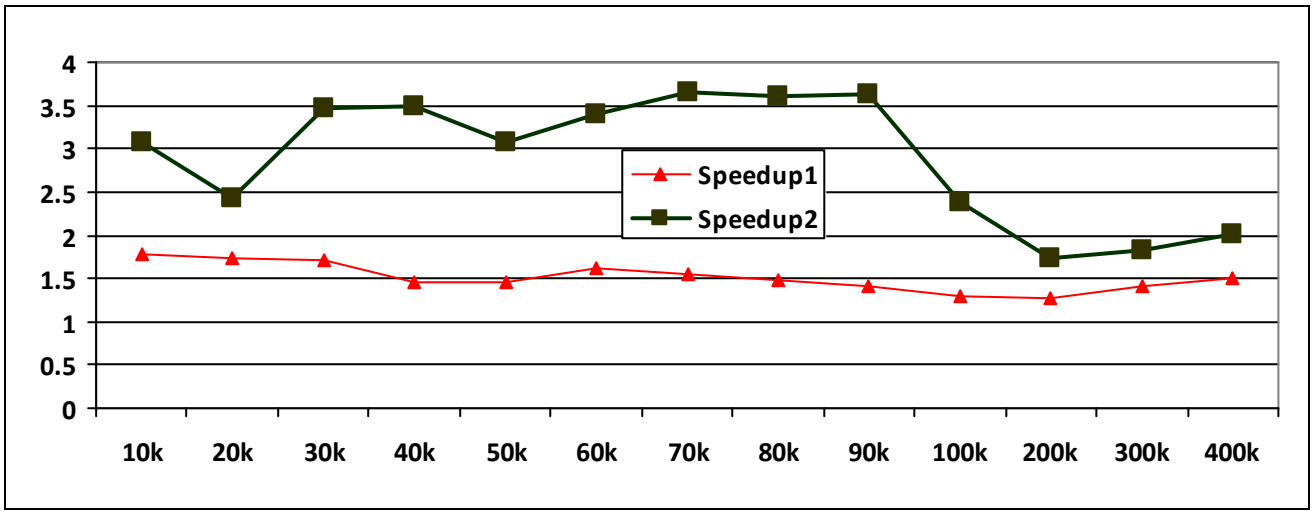

Fig. 13 Speedup comparison of serial, parallel, and enhanced parallel k-means where k =100 Clusters on different synthetic data sets with sizes as shown on horizontal axis.

V.CONCLUSIONS

The k-means algorithm is simple but it performs intensive calculation on computing distances between data points and cluster central points. For the dataset with n data points and k clusters, each iteration of k-means requires as much as $(n \times k)$ computations.

Fortunately, the distance computation of one data point does not interfere with the computation of other points. Therefore, k-means clustering is a good candidate for parallelism.

In this paper, we propose the design and implementation of efficient parallel k-means algorithm: we paralyzed the k-means method by using parallel for that run distance computation concurrently on multi-cores machine. The parallel programming model used in our implementation is based on parallel MATLAB programming.

The experimental results reveal that the proposed parallel method considerably speedups the computation time. Our future work will focus on the real applications. We will test our algorithm with a genome dataset.

ACKNOWLEDGEMENT

First of all thanks for Allah, then thanks for my wife, and colleagues encouraging me doing scientific research, my deep thanks for prince Sattam bin Abdulaziz University allowed me to use the Saudi Digital Library (SDL) and get the most recent papers in field.

REFERENCES

[1] G. Czibula, G. Cojocar, and I. Czibula, "Identifying Crosscutting Concerns using Partitional Clustering", WSEAS Transactions on Computers 8(2): pp. 386-395, 2009.

[2] R.O. Duda, P. E. Hart, "Pattern Classification and Scene Analysis". John Wiley & Sons, New York. 1973

[3] D. Dhillon and Modha, "A Data-Clustering Algorithm on Distributed Memory Multiprocessors", Proceedings of ACM SIGKDD Workshop on LargeScale Parallel KDD Systems, pp. 47-56, 1999.

[4] U.M. Fayyad, Piatetsky-Shapiro G., Smyth P., Uthurusamy R., "Advances in Knowledge Discovery and Data Mining" AAAI/MIT Press, 1996.

[5] R. Farivar, D. Rebolledo, E. Chan, and R. Campbell, "A Parallel Implementation of k-means Clustering on GPUs", Proceedings of International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA), pp. 340-345, 2008.

[6] A. M. Fahim, A. M. Salem, F. A. Torkey, M. A. Ramadan, "An efficient enhanced k-means clustering algorithm". Journal of Zhejiang University SCIENCE A, 7(10): pp. 1626-1633, 2006.

[7] A. Gersho, R. M. Gray, "Vector Quantization and Signal Compression". Kluwer Academic, Boston, 1992.

[8] M. Joshi, "Parallel k-means algorithm on distributed memory multiprocessors", Technical Report, University of Minnesota, pp. 1-12, 2003.

[9] Kantabutra S. and Couch A., "Parallel k-means clustering algorithm on NOWs", NECTEC Technical Journal, 1(6): pp. 243-248, 2000.

ISSN: 2278 – 7798

[10] N. Kerdprasop and K. Kerdprasop, "Knowledge Induction from Medical Databases with Higher-order Programming", WSEAS Transactions on Information Science and Applications, 6(10): pp. 1719-1728, 2009.

[11] K. Kerdprasop, N. Kerdprasop, and P. Sattayatham, "Weighted k-means for density-biased clustering", Lecture Notes in Computer Science, Vol.3589: pp. 488-497, Data Warehousing and Knowledge Discovery (DaWaK), August 2005,.

[12] X. Li and Z. Fang, "Parallel clustering algorithms", Parallel Computing, 11(3): pp. 275-290, 1989.

[13] J. MacQueen, "Some methods for classification and analysis of multivariate observations", Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281-297, 1967.

[14] MathWorks, Parallel Computing Toolbox™ User's Guide_R2011b,[Online]._Available: http://www.mathworks.com/help/distcomp/getting-started-with-parfor.html?s_tid=doc_12b

[15] F. Othman, R. Abdullah, N. Abdul Rashid, and R. Abdul Salam, "Parallel k-means clustering algorithm on DNA dataset", Proceedings of the 5th International Conference on Parallel and Distributed Computing: Applications and Technologies (PDCAT), pp. 248-251, 2004.

[16] Prasad, "Parallelization of k-means clustering algorithm", Project Report, University of Colorado, pp. 1-6, 2007.

Ahmed Mahmoud Fahim born in 1976, lives in Egypt, work at Faculty of Science, Suez University, Egypt, interested in data mining, get PhD in Computer Science in 2010 from Faculty of Science, Menofiya University, Egypt.

[17] Stoffel Kilian and Belkoniene Abdelkader, "Parallel k/h-Means Clustering for Large Data Sets", Euro-Par'99, LNCS 1685, pp. 1451-1454, 1999.

[18] S. Tirumala Rao, E. Prasad, and Venkateswarlu N., "A critical performance study of memory mapping on multi-core processors: An experiment with k-means algorithm with large data mining data sets", International Journal of Computer Applications, 1(9): pp. 1–8, 2010.

[19] J. Tian, L. Zhu, S. Zhang, and L. Liu, "Improvement and parallelism of k-means clustering algorithm", Tsignhua Science and Technology, 10(3): pp. 277-281, 2005.

[20] H. Wang, J. Zhao, H. Li, and J. Wang, "Parallel clustering algorithms for image processing on multicore CPUs", Proceedings of International Conference on Computer Science and Software Engineering (CSSE), pp. 450-53, 2008.

[21] Z. Ye, H. Mohamadian, S. Pang, and S. Iyengar, "Contrast enhancement and clustering segmentation of gray level images with quantitative information evaluation", WSEAS Transactions on Information Science and Applications, 5(2): pp. 181-188, 2008.

[22] W. Zhao, H. Ma, and Q. He, "Parallel k-means clustering based on MapReduce", Proceedings of the First International Conference on Cloud Computiong (CloudCom), pp. 674-679, 2009.

[23] Y. Zhang, Z. Xiong, J. Mao, and L. Ou, "The study of parallel k-means algorithm", Proceedings of the 6th World Congress on Intelligent Control and Automation, pp. 5868-5871, 2006.

ISSN: 2278 – 7798