Scalable Parallel Clustering Approach for Large Data using Possibilistic Fuzzy C-Means Algorithm

Juby Mathew
Dept.of MCA, Amaljyothi College of Engg.
Kanjirapally, Kerala

R Vijayakumar, Ph. D.
Professor and Dean, Faculty of Engg.
Mahatma Gandhi University, Kottayam, Kerala

ABSTRACT
Clustering is an unsupervised learning task where one seeks to identify a finite set of categories termed clusters to describe the data. The proposed system, try to exploit computational power from the multicore processors by modifying the design on existing algorithms and software. However, the existing clustering algorithms either handle different data types with inefficiency in handling large data or handle large data with limitations in considering numeric attributes. Hence, parallel clustering has come into picture to provide crucial contribution towards clustering large data. In this paper a scalable parallel clustering algorithm called Possibilistic Fuzzy C-Means (PFCM) clustering to cluster large data is introduced. In order to harvest the full power of a multi-core processor the software application must be able to execute tasks in parallel utilizing all available CPUs. To achieve this aim, it use fork/join method in java programming. It is the most effective design techniques for obtaining good parallel performance. The experimental analysis will be carried out to evaluate the feasibility of the scalable Possibilistic Fuzzy C-Means (PFCM) clustering approach. The experimental analysis showed that the proposed approach obtained upper head over existing method in terms of accuracy, classification error percentage and time.

Keywords
Clustering, parallel k-means, Fuzzy C-Means, Possibilistic Fuzzy C-Means, Fork/Join

1. INTRODUCTION
Since 40 years ago, clustering, which is one of the renowned data mining techniques, is being extensively studied and applied in numerous applications [1, 2]. Clustering can be defined as a process of allocating data objects into a specific disjoint group, which is called as a cluster; in such a way that the data objects belong to same cluster should be similar to each other, while the data objects of different cluster should be different from each other. Numerous clustering algorithms have been reported in the literature for clustering the subjected data in an efficient way. They can be classified as fuzzy clustering, partitional clustering, hierarchical clustering, artificial neural network - based clustering, statistical clustering algorithms, density-based clustering algorithm, etc. Despite varieties of algorithm classes prevail, partitional clustering algorithms and hierarchical clustering algorithms grab great attention from the researchers. Generally, hierarchically clustering algorithms produce satisfactory level of clustering performance. However, these algorithms do not provide options for reallocation of entities. This may lead to poor classification at the initial stage. Moreover, majority of the hierarchical algorithms consumes high computational time and memory [3].

The proposed method tries to implement the algorithm using Fork/Join method in JAVA. Fork/Join parallelism [4] is among the simplest and most effective design techniques for obtaining good parallel performance. Fork/join algorithms are parallel versions of familiar divide− and−conquer algorithms, taking the typical form:

```
Result solve (Problem problem)
{
If (problem is small)
        directly solve problem
else
{
Split problem into independent parts
Fork new subtasks to solve each part
Join all subtasks
Compose result from sub results
}
}
```

The fork operation starts a new parallel fork/join subtask. The join operation causes the current task not to proceed until the forked subtask has completed. Fork/join algorithms, like other divide−and−conquer algorithms, are nearly always recursive, repeatedly splitting subtasks until they are small enough to solve using simple, short sequential methods. The rest of the paper is organized as follows. Section 2 describes Literature review. Section 3 presents the proposed methodology. Section 4 shows experimental results and evaluations. Finally, the conclusions and future work are presented in Section 5.

2. LITERATURE REVIEW
Clustering is a task of assigning a set of objects into groups called clusters. In general the clustering algorithms can be classified into two categories. One is hard clustering; another one is soft (fuzzy) clustering. Hard clustering, the data's are divided into distinct clusters, where each data element belongs to exactly one cluster. In soft clustering, data elements belong to more than one cluster, and associated with each element is a set of membership levels. To obtain acceptable computational speed on huge datasets, most researchers turn to parallelizing scheme. Here, reviewed some of the techniques presented for literature. Md. Mostofa Ali Patwary et al [5] have presented a scalable parallel OPTICS algorithm (POPTICS) designed using graph algorithmic concepts. To break the data access sequentiality, POPTICS exploits the similarities between the OPTICS algorithm and PRIM's Minimum Spanning Tree algorithm. Li and Fang [6] are among the pioneer groups on studying parallel clustering. They proposed a parallel algorithm on a single instruction multiple data (SIMD) architecture. Dhillon and Modha [7] proposed a distributed k-means that runs on a multiprocessor environment. Kantabutra and Couch [8] proposed a master-slave single program multiple data (SPMD) approach on a network of workstations to parallel the k-means algorithm. Tian and colleagues [9] proposed the method for initial cluster center selection and the design of parallel k-means algorithm. Prasad [11] parallelized the k-means algorithm on a distributed memory multi-processors using the message passing scheme. Farivar and colleagues [12] studied parallelism using the graphic coprocessors to reduce energy

consumption of the main processor. Inderjit S et al. [13] presented a parallel implementation of the k-means clustering algorithm based on the message passing model. Jiabin Deng et al., [14] proposed an improved fuzzy clustering-text clustering method based on the fuzzy C-Means clustering algorithm and the edit distance algorithm, however, FCM is sensitive to noise and outliers because of its constraint of probabilistic type. A possibilistic approach called possibilistic c-means (PCM) was proposed by Krishnapuram and Keller to solving these problems of FCM [10]. PCM successfully solves the noise sensitivity problem of FCM. In the meanwhile, some new problems are also brought about by the PCM clustering model. PCM is sensitive to the initializations , it tends to generate coincident clusters and the clustering results of PCM heavily depend on the parameter of its clustering model and so on. Possibilistic C-Means (PCM) has been shown to be advantageous over Fuzzy C-Means (FCM) in noisy environments, it has been reported that the PCM has an undesirable tendency to produce coincident clusters. This approach combines the partitioning property of the FCM with the robust noise insensibility of the PCM. In 1997, Pal et al., [15] proposed the fuzzy-possibilistic C-Means (FPCM) technique and algorithm that generated both membership and typicality values when clustering unlabeled data. FPCM constrains the typicality values so that the sum over all data points of typicality's to a cluster is one. For large data sets the row sum constraint produces unrealistic typicality values. In this approach, a new model is presented called Possibilistic-Fuzzy C-Means (PFCM) model. PFCM produces memberships and possibilities simultaneously, along with the usual point prototypes or cluster centers for each cluster. PFCM is a hybridization of possibilistic c-means (PCM) and fuzzy c-means (FCM) that often avoids various problems of PCM, FCM and FPCM. PFCM produces memberships and possibilities concurrently, along with the usual point prototypes or cluster centers for each cluster.

3. METHODOLOGY

The aim of the proposed method is to cluster a large dataset efficiently. Here a scalable parallel clustering algorithm is used to overcome the problem in clustering large dataset with high dimension. The clustering is the Possibilistic Fuzzy C-Means (PFCM) clustering algorithm which is applied to the each randomly divided set of input data. Then finally the resultant cluster is obtained at the output. The fig.1 shown below represents the architecture of the proposed scalable parallel clustering algorithm. The initial stage of the proposed method is dividing the input large dataset in to n number of dataset according to the n number of cores available in the system.

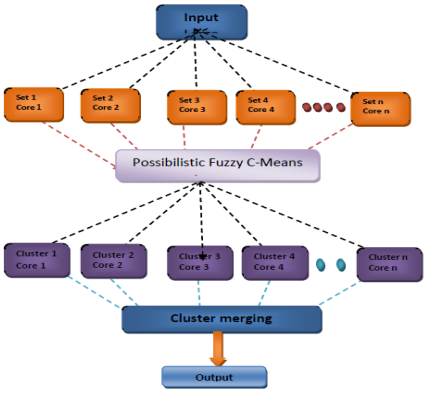

Fig. 1. Parallel architecture of the proposed algorithm

3.1 Partitioning the Input Large Dataset

Let the input be the large dataset with a size of $M \times N$. In this processing, input large dataset using Possibilistic fuzzy c-means clustering algorithm is difficult. So dividing the input dataset randomly in to small subsets of data with equal size will make system better. So further in this proposed system the input large data set is divided into N number of subset, based on number of cores available in the system, $S = \{S_1, S_2, S_3 \ldots\ldots\ldots S_N\}$, where N is the total number of sets with equal size. Here the each subset of data is clustered in to clusters using a standard and efficient clustering algorithm called Possibilistic Fuzzy C-Means (PFCM). Programmatically used in fork method in Java. The each single data subset S consist of a vector of d measurements, where $X = (x_1, x_2, x_3, \ldots\ldots x_d)$. The attribute of an individual data set is represented as x_i and d represents the dimensionality of the vector. The Possibilistic Fuzzy C-Means (PFCM) is applied to the each subset of dataset for clustering the input dataset $n \times d$ in to k-clusters.

Possibilistic Fuzzy C-Means (PFCM) clustering method [29] is applied to divided subset of data. The PFCM is one of the most efficient parallel clustering methods.

Let the unlabelled data set is $S = \{S_1, S_2, S_3 \ldots\ldots\ldots S_N\}$ which is further clustered in to a group of k-clusters using PFCM clustering method. This proposed PFCM is based on the minimization of the objective function given below,

$$\min_{(U,T,V)} \left\{ J_{m,\eta}(U,T,V;X) = \sum_{k=1}^{n}\sum_{i=1}^{c}\left(au_{ik}^{m} + bt_{ik}^{n}\right) \times \|x_k - v_i\|_A^2 + \sum_{i=1}^{c}\gamma_i \sum_{k=1}^{n}(1-t_{ik})^{n} \right\}$$

Subject to the constraints, $\sum_{i=1}^{c} u_{ik} = 1 \quad \forall k$, and $0 \le u_{ik}, t_{ik} \le 1$.

Here $a > 0, b > 0, m > 1, \eta > 1,$ where m is any real number greater than 1, u_{ik} is the degree of membership of x_i in the cluster j, x_i is the i^{th} of d-dimensional measured data, v_i is the d-dimension center of the cluster, and ||*|| is any norm expressing the similarity between any measured data and the center,where, $D_{ikA} = \|X_k - v_i\|A$ and $\sum_{k=1}^{n} t_{ik} = 1 \quad \forall i$

The PFCM clustering or partitioning is carried out through an iterative optimization of the objective function shown above, with the update of membership u_{ik} and the cluster centers v_i by,

$$u_{ik} = \left(\sum_{j=1}^{c} \left(\frac{D_{ikA}}{D_{jkA}} \right)^{2/(m-1)} \right)^{-1}, 1 \le i \le c; 1 \le k \le n$$

$$t_{ik} = \frac{1}{1 + \left(\frac{b}{\gamma_i} D_{ikA}^2 \right)^{1/(\eta-1)}}, 1 \le i \le c; 1 \le k \le n$$

,

$$v_i = \frac{\sum_{k=1}^{n} \left(au_{ik}^{m} + bt_{ik}^{\eta} \right) X_k}{\sum_{k=1}^{n} \left(au_{ik}^{m} + bt_{ik}^{\eta} \right)}, 1 \le i \le c.$$

This iteration will stop when $\max_{ik} \left\{ \left| u_{ik}^{(k+1)} - u_{ik}^{(k)} \right| \right\} < \varepsilon$

Where, ε is a termination criteria between 0 and 1, whereas k are the iteration steps. This procedure converges to local minimum of $J_{m,\eta}$.

The PFCM clustering algorithm contains various steps;
Algorithm 1:

Step 1: Initialize $U = [u_{ik}]matrix, U^{(0)}$

Step 2: At k step: calculate the centers vectors $C^{(k)} = [v_i]withU^{(k)}$

$$v_i = \frac{\sum_{k=1}^{n}\left(u_{ik}^{m} + t_{ik}^{\eta}\right)x_k}{\sum_{k=1}^{n}\left(u_{ik}^{m} + t_{ik}^{\eta}\right)}, 1 \le i \le c.$$

$$U^{(k)}, U^{(k+1)}$$

Step 3: Update

$$u_{ik} = \left(\sum_{j=1}^{c}\left(\frac{D_{ikA}}{D_{jkA}}\right)^{2/(m-1)}\right)^{-1}$$

Step 4: If $\left\|U^{(k)} - U^{(k+1)}\right\| < \varepsilon,$ then stop; otherwise return to step 2.

Finally for subset of input data, a group of K-clusters is obtained after applying the PFCM clustering method.

3.2 Parallel k-means

The pseudo code of parallel k-means is shown in Algorithm 2.
Algorithm 2. Parallel k-means (PKM)
Input: a set of data points and the number of clusters, K
Output: K-centroids and members of each cluster
Steps
1. Set initial global centroid C = <C1, C2, …, CK>
2. Partition data to P subgroups, each subgroup has equal size
3. for each P,
4. Create a new process
5. Send C to the created process for calculating distances and assigning cluster members
6. Receive cluster members of K clusters from P processes
7. Recalculate new centroid C‟‟
8. If difference(C, C‟‟)
9. Then set C to be C‟‟ and go back to step 2
10. Else stop and return C as well as cluster members
Fork/Join Framework adds two main classes to the java.util.concurrent package:

- ForkJoinPool
- ForkJoinTask

The execution of ForkJoinTask takes place within a ForkJoinPool, which manages the execution of the tasks. ForkJoinTask objects support the creation of subtasks and waiting for the subtasks to complete. Advantage of the ForkJoinPool, is that it can 'steal' work. It means that it allows one thread that has finished a task to immediately execute another task with much less overhead than the Executor Service. This work stealing enables efficient load balancing. The number of worker threads in a fork/join pool is generally upper-bounded by the number of cores in the system. Work stealing automatically corrects unequal distribution of work without central coordination [16].
The RecursiveAction and RecursiveTask are the only two direct, known subclasses of ForkJoinTask. The only difference between these two classes is that the RecursiveAction does not return a value while RecursiveTask does have a return value and returns an object of specified type.

ForkJoinTask objects feature two specific methods:
The fork () method allows a new ForkJoinTask to be launched from an existing one. In turn, the join () method allows a ForkJoinTask to wait for the completion of another one. A ForkJoinTask instance is very light weight when compared to a normal Java thread. [4]
Following code can be used to perform parallel operations.

```
public class FJoin extends RecursiveTask<Integer>
{
public static ArrayList<ArrayList<String>> input=new ArrayList<ArrayList<String>();
private static final int size=130;
public FJoin(int data,int start,int end)
{
this.data=data;
this.start=start;
this.end=end;
}
public FJoin(int data)
{
this(data,0,data);
}
@override
protected Integer compute()
{
final int length=end-start;
if(length<size)
{
return computeDirectly();
}
final int split=length/2;
final FJoin first=new FJoin(data,start,start+split);
first.fork();
final FJoin second=new FJoin(data,start+split,end);
return Math.max(second.compute(),first.join());
}
```

Following code can be used to find accuracy

```
public class Accuracy
{
 public static void  findaccuracy (ArrayList <ArrayList
<Integer>> cluster,ArrayList<Integer> cls)
   {
     ArrayList<Integer> uni=new ArrayList<Integer>();
     for(int i=0;i<cls.size();i++)
     if(!uni.contains(cls.get(i)))
     uni.add(cls.get(i));
     int sum=0,totsum=0;
     for(int i=0;i<cluster.size();i++)
     {
        int a[]=new int[uni.size()];
        for(int j=0;j<a.length;j++)
        a[j]=0;
        for(int j=0;j<cluster.get(i).size();j++)
        {
          int v=cls.get(cluster.get(i).get(j));
          a[v-1]=a[v-1]+1;
        }
        System.out.println(a[0]+"  "+a[1]);
        int v=a[0];
        totsum=totsum+v;
        for(int j=1;j<a.length;j++)
        {totsum=totsum+a[j];
         if(v<a[j])
          v=a[j];
        }
       sum=sum+v;
     }
```

```
        System.out.println("Accuracy ::
        "+((double)sum/(double)totsum));
    }
}
```

Fork/join parallelism is implemented by means of a fixed pool of worker threads. This is achieved by setting the number of worker threads in the fork/join pool to a maximum of four, which is the number of cores in a node in the system. Each worker thread can execute one task at a time. Tasks waiting to be executed are stored in a queue, which is owned by a particular worker thread. Currently executing tasks can dynamically generate (i.e. fork) new tasks, which are then enqueued for subsequent execution.

4. EXPERIMENTAL RESULTS AND DISCUSSION

The experimental result of the proposed approach is furnished in this section. The experimental evaluation is conducted in order to evaluate the proposed approach. The method is implemented by using JAVA language. The code is executed on dell inspiron N4030 Laptop, Intel(R) Core(TM) i5 Processor 2.67 GHz, 2 MB cache memory, 3GB RAM, 64-bit Windows 7 Home and NetBeans IDE 8.0.

4.1 Dataset Description

In this proposed method two kinds of datasets are used for evaluation. The Iris data set consists of three varieties of flowers—setosa, virginica and versicolor. There are 150 instances and 4 attributes that make up the 3 classes.

The Second data set contains thyroid data set. The Thyroid data set is based on the diagnosis of thyroid whether it is hyper or hypofunction. The data set contains 215 patterns, 5 attributes and 3 classes.

4.2. Evaluation Metrics

The proposed scalable parallel clustering algorithm uses mainly three evaluation matrices, the clustering accuracy, Classification Error Percentage (CEP) and computation time.

4.2.1. Clustering Accuracy

The clustering accuracy is measured by counting the number of correctly assigned documents and dividing by N, which is given the equation below

$$Accuracy = \frac{1}{N}\sum_{k}\max_{j}\left|B_k \cap C_j\right|$$

Where, B_k is the set of clusters $B = \{B_1, B_2, B_3 \ldots\ldots\ldots B_k\}$ and C_j is the set of classes $C = \{C_1, C_2 . C_3, \ldots\ldots C_k\}$. It interpret B_k as the set of documents in B_k and C_j as the set of documents in C_j.

Table 1: Accuracy of Iris Dataset

Final Clusters	PKM	PFCM
2	65.2	66.7
3	72.5	78.6
4	80.3	88.9
5	83.7	90.5
6	85.1	92.3

In the figure 2, shows the accuracy of the proposed approach by varying the clusters of the parallel clustering system. The final clusters are varied from 2 to 6. The analysis from the Table 1 shows that the proposed approach has upper hand over the existing method. In the case of proposed approach, the accuracy increases as the number of final cluster increases.

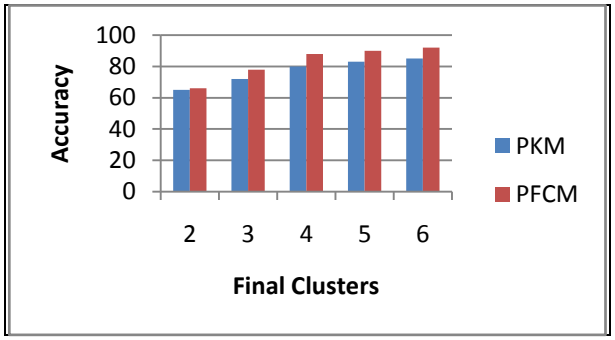

Fig.2.Iris Dataset

The maximum accuracy attained by the proposed approach is 92%, which is very better figure as compared to the existing method, for which the maximum accuracy obtained is 85%. All these experiments are conducted in same environment.

Table 2: Accuracy of Thyroid Dataset

Final Clusters	PKM	PFCM
2	58.1	60.8
3	65.6	68.2
4	68.4	78.1
5	75.6	85.4
6	80.8	91.7

In the figure 3, it presented the accuracy of the proposed approach by varying the final clusters of the parallel clustering system in thyroid dataset. The final clusters are varied from 2 to 6. The analysis from the Table 2 shows that the proposed approach has upper hand over the existing method. In the case of proposed approach, the accuracy attained by the proposed approach is very better as compared to the existing method.

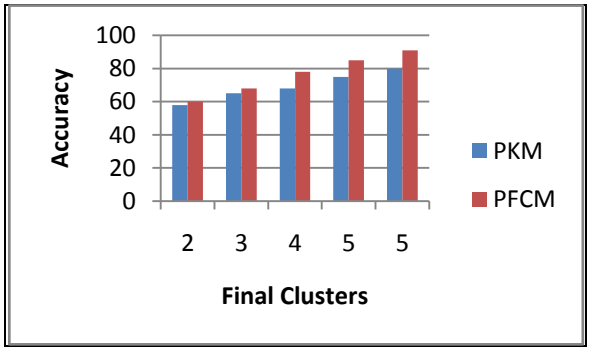

Fig. 3.Thyroid Dataset

4.2.2. Classification Error Percentage (CEP)

One of the most important characteristics of a clustering method is the ability of it in deceasing clustering error. The given data set, 75% of the data set is randomly selected to obtain the cluster centers using Algorithm 1. In this way to obtain the cluster centers for all the classes. The remaining 25% of data set is used (called test data set) to obtain the classification error percentage (CEP).The classification of each pattern is done by assigning it to the class whose distance is closest to the center of the clusters. Then, the classified output is compared with the desired output and if they are not exactly the same, the pattern is separated as misclassified. Let **n** be the total number of elements in the dataset and **m** be the number of elements misclassified after finding out the cluster center using the above algorithms. Then classification error percentage is given by [17]

CEP= $\frac{m}{n} * 100$

Here use two dataset for calculating CEP. Table 3 shows that classification error percentage of two data set like Iris and Thyroid.

Table 3: Classification error percentage

	PKM	PFCM
Iris	0.34	0
Thyroid	1.23	0.12

From the training data set the knowledge in the form of cluster centers is obtained using the algorithm 1.For these cluster centers the testing data sets are applied and the CEP values are obtained. As can be seen from Table 3, the PFCM outperforms the existing method.

4.2.3. Time Comparison

The time analysis based on the data size is shown in the figure 4 and figure 5. The computation time which is measured based on the starting and ending time of the program. From this analysis found that the proposed approach and existing approach is clearly different from each other. PFCM algorithm enables improving the time performance of clustering on large scale data sets and Java F/J is quite appropriate for the parallel computing environment.

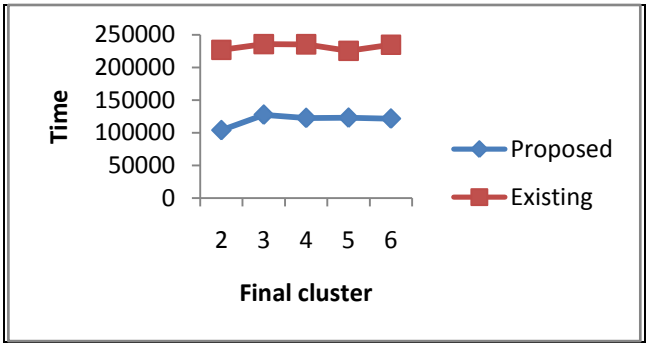

Fig.4.Iris Dataset

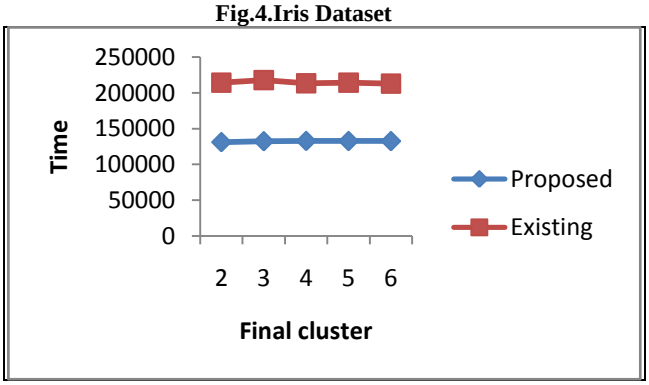

Fig. 5.Thyroid Dataset

As considering all scenarios, assess that proposed algorithm is efficient in compared with parallel k means algorithm.

5. CONCLUSION

In this paper, proposed the design and implementation of scalable PFCM algorithm. Here used fork and join model for the Java programming concurrently on multi-cores machine. Fork/join method overcomes deficiencies of multithreaded execution. Large data clustering plays crucial and related processes in various domains. However, most of the clustering algorithms are compatible only with small data. The solution to address large scale clustering problem is exploiting parallel algorithm. The proposed approach is designed to address mainly for the difficulty to cluster large data sets. The proposed approach used a PFCM algorithm to handle the large data set. The proposed method is compared with the performance of the existing parallel k-means clustering algorithm. The performance analysis and experimental result showed that proposed method provide better result. Finally, solution utilizes maximum hardware capabilities of multi-core systems for faster execution by processing multiple tasks in parallel. Also the experimental analysis showed that the proposed approach obtained upper head over existing method in terms of accuracy and time. In future work it will apply different cores in different cluster size.

6. REFERENCES

[1] JinchaoJi , Wei Pang, Chunguang Zhou, Xiao Han, Zhe Wang, “A fuzzy k-prototype clustering algorithm for mixed numeric and categorical data”, journal of Knowledge-Based Systems, vol. 30, pp. 129-135, 2012

[2] Swagatam Das, Ajith Abraham, Amit Konar, "Automatic Clustering Using an Improved Differential Evolution Algorithm", IEEE Transactions on Systems, Man, and Cybernetics—Systems And Humans, Vol. 38, No. 1, 2008

[3] HesamIzakian, Ajith Abraham, Vaclav Snasel, "Fuzzy Clustering Using Hybrid Fuzzy c-means and Fuzzy Particle Swarm Optimization", World Congress on Nature and Biologically Inspired Computing (NaBIC 2009), India, IEEE Press, pp. 1690-1694, 2009.

[4] Doug Lea, A Java Fork/Join Framework, State University of New York at Oswego,www.developer.com

[5] D. Mostofa Ali Patwary,Diana Palsetia1, Ankit Agrawal1,Wei-keng Liao1, Fredrik Manne2, AlokChoudhary, " Scalable Parallel OPTICS Data Clustering Using Graph Algorithmic Techniques", International Conference for High Performance Computing, Networking, Storage and Analysis, ACM, No. 49, 2013

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

[7] Dhillon and Modha D., “A Data-Clustering Algorithm on Distributed Memory Multiprocessors”, Proceedings of ACM Workshop on Large Scale Parallel KDD Systems, 1999, pp. 47-56.

[8] Kantabutra S. and Couch A., “Parallel k-means clustering algorithm on NOWs”, Technical Journal NECTEC, 2000, Vol.1,No.6

[9] Tian J., Zhu L., Zhang S., and Liu L., “Improvement and parallelism of k-means clustering algorithm”, Tsignhua Science and Technology, 2005

[10] R. Krishnapuram and J. M. Keller, “A possibilistic approach to clustering,” IEEE Transactions on Fuzzy Systems, vol. 1, p.10.1109/91.227387, 1993.

[11] Prasad, “Parallelization of k-means clustering algorithm”, Project Report, University of Colorado, 2007.

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

[13] Inderjit S. Dhillon and Dharmendra S. Modha, “A Data-Clustering Algorithm On Distributed Memory

Multiprocessors", Proceedings of KDD Workshop High Performance Knowledge Discovery, pp. 245-260, 1999.

[14] Jiabin Deng, JuanLi Hu, Hehua Chi and Juebo Wu, "An Improved Fuzzy Clustering Method for Text Mining", Second International Conference on Networks Security Wireless Communications and Trusted Computing (NSWCTC), Vol. 1, Pp. 65–69, 2010.

[15] Pal N.R, Pal K, Keller J.M. and Bezdek J.C, "A Possibilistic Fuzzy c-Means Clustering Algorithm", IEEE Transactions on Fuzzy Systems, Vol. 13, No. 4, Pp. 517–530,

[16] Robert D Blumofe, The University of Texas at Austin, Scheduling Multithreaded Computations by Work stealing.

[17] J Senthilnath, S.N.Omkar, Swarm and Evolutionary Computation 1(2011)164-171.