ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol.2, Special Issue 5, October 2014

A New Hybrid Scalable Parallel Clustering Approach For Large Data Using FCM And FA

Juby Mathew[1], Dr. R Vijayakumar[2]

Research Scholar, Schoolof Computer Science, Mahatma Gandhi University, Kottayam, Kerala, India[1]

Dean, Faculty of Engg., Mahatma Gandhi University, Kottayam, Kerala, India[2]

ABSTRACT: Clustering is an unsupervised learning task where one seeks to identify a finite set of categories termed clusters to describe the data. we try to exploit computational power from the multicore processors. we need a new design on existing algorithms and software. This research work analyzes about the performance of parallel k means algorithm and based on this algorithm ,we propose a new parallel architecture combined with PKM,FCM and FA algorithm. Here, the parallel architecture will be developed by including the process like, splitting the input data, clustering each subset of data and merging to optimal final clustering.Clustering of subset of data FCM used and optimal merging process we apply FA. Firefly-based clustering is a recent method which proves better for optimal clustering finding. The experimental results show that the performance of modified parallel algorithmis better than the parallel k-means algorithm.In order to utilize the intrinsiccapabilities of a multi-core processor the software application must be able to execute tasks in parallel using all available CPUs. To achieve this we can use fork/join method in java programming.

KEYWORDS: Clustering,Parallel k means,Fuzzy c-means, Firefly algorithm

I.INTRODUCTION

Classification as supervised machine learning is the discovery of a predictive learning function that classifies a new unlabelled data object into one of several predefinedclasses . Class descriptions are derived from a collection of labelled objects througha training procedure.Clustering is the task of assigning a set of objects into groups (called clusters) so that the objects in the same cluster are more similar (in some sense or another) to each other than to those in other clusters.A good clustering method will produce high quality clusters with high intra-class similarity - Similar to one another within the same cluster low inter-class similarity - Dissimilar to the objects in other clustersThe goal of a clustering algorithm is to maximize the intra-cluster similarity and minimize the inter-cluster similarity [1][2].

The clustering algorithm finds the centroid of a group ofdata. To determine cluster membership, the algorithm evaluates the distance between a point and the cluster centroids.[1]. The popularity of k-means algorithm is due to its computational complexity that is *O (nkd)*, where *n* is the number of data points or objects, *k* is the number of desired clusters, and *d* 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 [3][4][5].This paper focuses on optimising parallel k-means algorithm using FCM and FA. We implement our extension of the Parallel k-means algorithm using Fork/Join method in JAVA.

Fork/Join parallelism[6]is a style of parallel programming useful for exploiting the parallelism inherent in divide and conquer algorithms,taking the typical form:
if (my portion of the work is small enough)
do the work directly

```
else
{
Split problem into independent parts
Fork new subtasks to solve each part
Join all subtasks
}
```

fork-join executor framework has been created which is responsible for creating one new task object which is again responsible for creating new sub-task object and waiting for sub-task to be completed.Internally it maintains a thread pool and executor assign pending task to this thread pool to complete when one task is waiting for another task to complete. The rest of the paper is organized as follows. Section 2 describes related work. Section 3 presents the proposed parallel algorithm.Section 4 shows experimental results and evaluations. Finally, the conclusions and future work are presented in Section 5.

II.RELATED WORK

Here, we review some of the techniques presented for literature.A serial k-means algorithm was proposed in 1967 and since then it has gained great interest from data analysts and computer scientists.To obtain acceptable computational speed on huge datasets, most researchers turn to parallelizing scheme.Md. Mostofa Ali Patwary et al [7] have presented a scalable parallel OPTICS algorithm(POPTICS) designed using graph algorithmic concepts. To break the data access sequentiality, POPTICS exploits the similaritiesbetween the OPTICS algorithm and PRIM's Minimum SpanningTree algorithm. Additionally, They used the disjoint-set datastructure to achieve a high parallelism for distributed cluster extraction.Using high dimensional datasets containing up to a billionfloating point numbers, they showed scalable speedups of up to 27.5for our OpenMP implementation on a 40-core shared-memory machine,and up to 3,008 for our MPI implementation on a 4,096-coredistributed-memory machine. They also showed that the quality of theresults given by POPTICS is comparable to those given by the classicalOPTICS algorithm.

Li and Fang [3] 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 [8] proposed a distributed k-means that runs on a multiprocessor environment. Kantabutra and Couch [4] 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. 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 [10]. 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.

1. Parallel k means

The pseudo codeof parallel k-means is shown in Algorithm 2.

Algorithm 1. 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 hasequal size
3. for each P,
4. Create a new process
5.Send C to the created process for calculatingdistances and assigning cluster members
6. Receive cluster members of K clusters from Pprocesses
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

2. Parallel Fuzzy c means

The parallel FCM cluster analysis procedure is described by the following sequence:
Step 1: Splits the data set equally among the available processors so that each one receives N/ p records, where N is the number of records and p is the number of processes
Step 2: Compute the geometrical center of its local data and communicate this center to all processors, so that every processor can compute the geometrical center of the entire database.
Step 3: Sets initial centers and broadcasts them, so that all processors have the same clusters' centers values at the beginning of the FCM looping.
Step 4: Until convergence is achieved compute the distances from each record in the local dataset to all clusters' centers; update the partition matrix and calculate new clusters' centers.
Step 5. If the range of number of clusters is covered, stops, otherwise returns to Step3.
The procedure described above is computed for each number of clusters in the cluster analysis, so that the procedure is repeated as many times as the desired range of numbers of clusters[20]

3. Firefly algorithm

The firefly algorithm (FA) is a metaheuristic algorithm, inspired by the flashing behaviour of fireflies. The primary purpose for a firefly's flash is to act as a signal system to attract other fireflies.[14]
a.All fireflies are unisexual, so that one firefly will be attracted to all other fireflies regardless of their sex;
b.Attractiveness is proportional to their brightness, and for any two fireflies, the less bright one will be attracted by the brighter one.
c.If there are no fireflies brighter than a given firefly, it will move randomly.
The brightness should be associated with the objective function.
A popular partitional clustering algorithm—k-means clustering, is essentially a function minimization technique. The main drawback of k-means algorithm is that it converges to local minima from the starting position of the search [15].To overcome this problem we can apply firefly algorithm.
Assume that there exists a swarm of n agents (fireflies) and Xi represents a solution for a firefly i, whereas f (Xi) denotes its fitness value[20]. Here the brightness I of a firefly is selected to reflect its current position x of its fitness value f (x) [16].

$$I_i = f(X_i),\ 1 \le i \le n. \qquad (1)$$

Firefly attractiveness is proportional to the light intensity seen by adjacent fireflies [16]. Each firefly has its distinctive attractiveness β which implies howstrong it attracts other members of the swarm. However, the attractiveness β is relative, it will vary with the distance r_{ij} between two fireflies i and j at locations x_i and x_j respectively, is given as

$$r_{ij} = \|x_i - xj\|. \qquad (2)$$

The degree of attractiveness of a firefly is determined by

$$\beta(r) = \beta_0 e{-}^{\gamma\ r2} \qquad (3)$$

where β_0 is the attractiveness at r = 0 and γ is the light absorption coefficient at the source.
The movement of a firefly i at location x_i attracted to another more attractive (brighter) firefly j at location x_j is determined by

$$x_i(t+1) = xi(t) + \beta_0 e^{-\gamma r2}(x_j - x_i). \qquad (4)$$

Algorithm 2: firefly algorithm

```
Initialize algorithm parametes.
Define light absorption co efficient γ
Define maxGenerations
While(t< maxGenerations)
for i = 1 to no.of fireflies
for j = 1 to no.of fireflies
If ($I_j$< $I_i$)
```

```
Move firefly i toward j using Eq. (4)
end if
Evaluate new solutions and update lightintensity using Eq. (1)
end for j
end for i
Rank the fireflies and find the current best
End while
```

III.PROPOSED WORK

Proposed Algorithm

In the proposed algorithm,the clustering has been two steps,first step initialize fireflies with random values. The objective function, which must be minimum, is Euclidean distance. The mechanism of firefly algorithm must do till predefine iteration.In the second step, Sets initial centers and broadcasts them, so that all processors have the same clusters' centers values at the beginning of the FCM looping and it will assign the position of best firefly. The proposed hydride parallel clustering algorithm can be summarized as the pseudo code.

The proposed algorithm concentrates on distance calculation between each point and the k centers, performs these calculations in parallel and optimized way. If we have m cores and n data points then each core will approximately calculate the distances between n/m points and k centers. As m increase, the amount of calculation per each core will decrease.

Algorithm 3:Modified parallel k means

```
1. Initialize a population of fireflies
2. Define light absorption coefficient $\gamma$
3. Find number of cores P
4. Partition data to P subgroups
5. for each P
6.          for i=1 to no.of fireflies
for j=1:i
If ($I_j$>$I_i$) then
Move firefly i towards j in all d dimensions
End if
End for j
End for i
7. Receive cluster members of k cluster from P Process
8. Recalculate new centroid using eq.5
```

$$z_j = \frac{\sum_{i=1}^{n} \mu_{ij}^{m} o_i}{\sum_{i=1}^{n} \mu_{ij}^{m}} \qquad (5)$$

```
9. If k stablecentroid reached
10. Else go to step 4
```

Our approach consists of parallelizing the first phase of each step, where we calculate the nearest centroid to each point. The correctness of the algorithm is guaranteed, to the extent that it produces the same output as the original Parallel k-means algorithm. Our implementation performs the same steps as the original Parallel k-means algorithm in parallel without changing the semantics or logic of the original implementation.

The Fork/Join Framework enhances multithreaded programming in two ways. First, it simplifies the creation and use of multiple threads. Second, it automatically makes use of multiple processors**.**

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 ExecutorService.[17].

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.[6]
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());
}
```

Fork/join parallelism is implemented by means of a fixed pool of worker threads. We set the number of worker threads in the fork/join pool to a maximum of four, which is the number of cores in a node. 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.

IV.EXPERIMENTAL RESULTS

We implemented the proposed algorithm, parallel k-means andmodifiedalgorithm 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.

We have implemented proposed algorithm on two-dimensional dataset, four clusters. The computational speed of proposed algorithm as compared with parallel k-means is given in Table1, *Tp* refers to execution time of parallel k-means, *Tpm* refers to execution time of parallel k-means modified and *Sp* are the time difference obtained from modified algorithm and existing algorithm.If we compare the results shown in Table 1 with that of [4] and [18] we get better results from our proposed algorithm.Running time comparison of modified parallel against parallel k-means is graphically shown in Figure 2.The CPU usage shows that all processing power is completely utilized while the Fork/Join method runs with 4 clusters hence the application gets more resource and process faster.
Fig 1 shows that Task Manager Performance view of the proposedalgorithm is running:

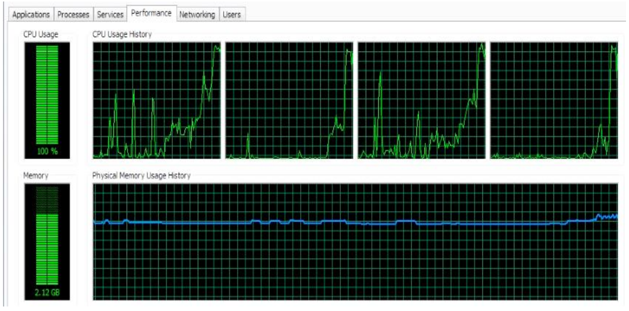

Fig.1:Task manger Performance view

Table1: The execution time of Parallel k means and Modified Parallel k means

Dataset Size N	Exec.Sec PKM (Tp)	Exec.Sec MPKM (Tpm)	Time Diff(Sp) MPKM
100K	3.12	3.12	0
200K	6.56	6.55	0.01
300K	10.8	10.6	0.2
400K	12.9	12.9	0
500K	15.92	15.9	0.02
600K	19.9	18.7	1.2
700K	24.2	23.2	1
800K	27.9	26.4	1.5
900K	29.76	28.5	1.26
1000K	31.45	30.1	1.35

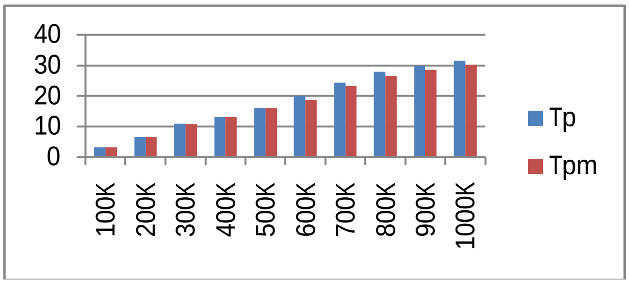

Fig.2 Running time Comparison k=4 clusters

V. PERFORMANCE ANALYSIS

Performance of the proposed algorithm are evaluated by classification error percentage(CEP) and classification efficiency. In paper [19],Senthilnath applied FA for clustering data objects into groups according to the values of their attributes.

Classification Error Percentage (CEP)

The classification of each pattern is done by assigning it tothe 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

$$\text{CEP}=\frac{\text{m}}{\text{n}} * 100 \qquad (6)$$

Classification efficiency

Classification efficiency is obtained using both the training(75%)and test data(25%). The efficiency is indicated bythe percentage classification which tells us how many samples belonging to a particular class have been correctly classified**.** Thepercentage classification (μi) for the class ci

$$\mu i = \frac{qii}{\sum_{j=1}^{n} qji} \ (7)$$

where **q**$\mathbf{q_{ii}}$ is the number of correctly classified samples and **n** is the number of samples for the class **ci** in the data set**.**

We use two dataset for calculating CEP and efficiency for the new algorithm. The Credit data set consists of 690 patterns,15 input features and the output has 4 classes.The glassdata set contains 4 types of glass, with 214patterns.Table 2 shows that classification error percentage of two data set like Credit and glass using eq.6..Table 3 shows that Classification efficiency of two data sets calculated by eq.7

Table 2:Classification error percentage

	PKM	MPKM
Credit	15.8	8.7
Glass	22.3	10.12

Table 3:Classification efficiency

	PKM(%)	MPKM(%)
Credit	75	95
Glass	88	97

The above results shows that Proposed algorithm is efficient in compared with parallel k means algorithm.

V.CONCLUSION

Finally, we conclude that modified parallel k-means algorithm should be improved in terms of its computation time and efficiency.we have used fork and join model for the Java programming concurrently on multi-cores machine. Fork/join method overcomes deficiencies of multithreaded execution.The experimental results reveal that the new parallel method considerably speedups the computation time. Finally, our solution utilizes maximum hardware capabilities of multi-core systems for faster execution by processing multiple tasks in parallel. In a multi-coremachine, our algorithm is orders of magnitude faster than parallel k-means clustering algorithms. In our future work it will apply different cores in different cluster size. From the results obtained we conclude that proposed algorithm is efficient and optimezed for generating cluster centers.

REFERENCES

[1]. L.Hall, B.Ozyrt, “Clustering with a Genetically Optimized Approach”, IEEE Transactions on Evolutionary computation, 1999, Vol.3
[2]. M.P Sebastian, K.A Abdul Nazeer, “Improving the Accuracy and Effiency of the K-means Clustering Algorithm”, WCE 2009, London
[3]. Li X. and Fang Z., “Parallel clustering algorithms”, Parallel Computing, 1989, 11(3): pp. 275-290.
[4]. Kantabutra S. and Couch A., “Parallel k-means clustering algorithm on NOWs”, Technical Journal NECTEC, 2000, Vol.1,No.6
[5]. Zhang Y., Xiong Z., Mao J.,“The study of parallel k-means algorithm”, Proceedings of the 6th World Congress on Intelligent Control and Automation, 2006.
[6]. Doug Lea, A Java Fork/Join Framework, State University of New York at Oswego,
www.developer.com
[7] d. Mostofa Ali Patwary,Diana Palsetia1, Ankit Agrawal1,Wei-keng Liao1, Fredrik Manne2, AlokChoudhary, " Scalable Parallel OPTICS Data ClusteringUsing Graph Algorithmic Techniques", International Conference for High Performance Computing, Networking, Storage and Analysis, ACM, No. 49, 2013
[8]. 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.
[9]. Tian J., Zhu L., Zhang S., and Liu L., “Improvement and parallelism of k-means clustering algorithm”, Tsignhua Science and Technology, 2005.
[10].Stoffel Kilian and Belkoniene Abdelkader, “Parallel k/h-Means Clustering for Large Data Sets”, Euro-Par'99, LNCS 1685, 1999, pp. 1451-1454.
[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]M. R. Senapati. "Local linear wavelet neural network based breast tumor classification using firefly algorithm", Neural Computing and Applications,2012
[15]. S.Z. Selim, M.A. Ismail, K-means type algorithms: a generalized convergence theorem and characterization of local optimality, IEEE Transactions on Pattern Analysis and Machine Intelligence 6 (1984) 81–87.
[16].X.S.Yang, Nature-Inspired Metaheuristic Algorithms, Luniver Press, 2008.
[17]. Robert D Blumofe,The University of Texas at Austin, Scheduling Multithreaded Computations by Work stealing
[18]Fahim Ahmed M, Parallel Implementation of K-Means on Multi-Core Processors, ISSN 1512-1232, Computer Science and Telecommunications 2014.
[19]. J. Senthilnath, S.N. Omkar, V. Mani,
Clustering using firefly algorithm: Performance study, Swarm and Evolutionary Computation 1 (2011) 164–171.
[20]. Jie Lio, Chao-Hsien Chu, Wang and Yungeng. An In-depth analysis of fuzzy c-means clustering for cellular manufacturing. (FSKD’08), 2008.