A Multicore Approach for DBCAN Data Clustering Using KD Tree

Ms. Alka Shrivastava
alkashriwastava@gmail.com
Bharti College of Engineering & Technology, Durg

Mr. Suman Swarnkar
sumanswarnkar17@gmail.com
Bharti College of Engineering & Technology, Durg

***Abstract*— The issue of recognizing classes of data points is challenging when the clusters are of distinctive size, shape and density. Most of these issues turn out to be much more critical when the data is of high dimensionality and when it incorporates outliers and noise. In earlier implementation of DBSCAN algorithms some bottlenecks that need to be resolved, i.e. presence of noise in input dataset affects the algorithm performance and increase in outlier, existing parallel implementation of DBSCAN algorithms are based master slave which imposes I/O overhead henceforth there is need of better implementation of DBSCAN. In this paper we have proposed distributed DBSCAN implementation by means of multicore programming, further we have used k-dimension tree as data structure.**

***Keywords*—KD,Eps,DCPGS**

I. Introduction

Data clustering is a data mining practice that assemblies data into evocative subclasses, termed as clusters, such that it curtails the intra-differences and get the most out of inter-differences of these subclasses. Renowned algorithms for data clustering include DBSCAN, K-medoids, Kmeans, BIRCH, WaveCluster and STING. These algorithms have been castoff in numerousmethodical areas for example satellite image segmentation, noise sieving and outlier detection, unsupervised text document clustering, and clustering of bioinformatics data. Earlier data clustering algorithms have been unevenlyclassified into four classes: partitioning-based, hierarchybased, grid-based, and density-based.

DBSCAN (Density Based Spatial Clustering of Applications with Noise) is a density centered clustering algorithm. The concept behind DBSCAN algorithm is that for every data point in a cluster, the vicinity within a specified radius (eps) has to comprise at least a minimum number of points (minpts), i.e. the density of the neighborhood has to outstrip some threshold.

DBSCAN algorithm was projected by Martin Ester and others in 1996. DBSCAN algorithm entails only two factors: Eps & MinPts. Eps frights with a randompreliminary point that has not been stayed. This point's neighborhood is salvaged, and if it comprisesadequately many points, a cluster is underway. MinPts is the minutest number of points needs to form a cluster. Eps is aprevalent parameter for DBSCAN algorithm but shrewd the value of Eps is time consuming and difficult. For calculating Eps.

DBSCAN (D, eps, MinPts)

```
C = 0
for each unvisited point p in dataset D
mark P as visited
NeighborPts = regionQuery (P, eps)
If sizeof(NeighborPts) < MinPts
mark P as NOISE
else
C= next cluster
expandCluster (P, NeighborPts, C, eps, MinPts)
add P to cluster C
for each point P’ in NeighborPts
if P’ is not visited
mark P’ as visited
NeighborPts’ = regionQuery (P’, eps)
If sizeof (NeighborPts’) >= MinPts
NeighborPts = Neighborpts joined with NeighborPts’
if P’ is not yet member of any cluster
add P’ to cluster C
regionQuery (P, eps)
return all points within P’s eps-neighbourhood
```

KD Tree (K Dimensional Trees)

In this paper proposed algorithm uses KD Tree data structure while implementing multicore DBSCAN. Splitting planes perpendicular to Coordinate Axes.Useful in Nearest Neighbor Search.KD tree reduces the Overall Time Complexity to O(log n) which has been used in many clustering algorithms and other domains.

Input:
Insert: (30,40), (5,25), (10,12), (70,70), (50,30), (35,45)

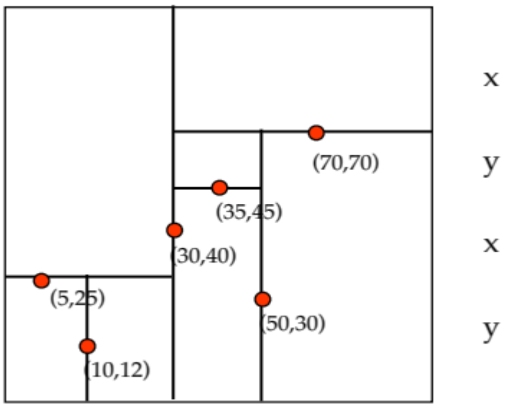


Fig.-1 Splitting planes perpendicular to Coordinate Axes

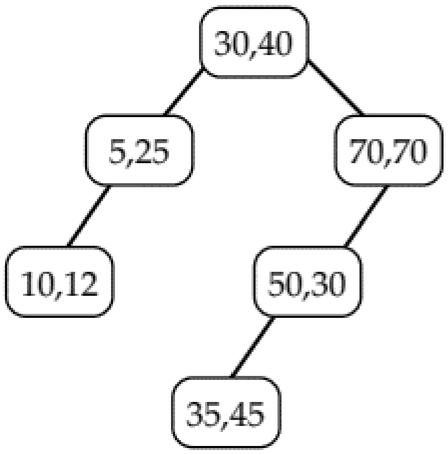


Fig.-2 kD Tree Constructed

Further in this paper in section-II we have discussed different literature and given tabular association among different literature, in section-IIIwe will discuss advantage and disadvantages of different implementation, in section IV proposed methodology discussed, in section V implementation section at last we will conclude our research.

II. Literature Survey

I this paper we have gone through different literature some of them are briefed below:

Xiangliang Zhang et. al. said that proposed algorithm STRAP aims at clustering data streams with evolving data distributions. STRAP confronts the arriving items to the current AP model, storing the outliers in a reservoir and monitoring the ratio of outliers using the PH change point detection test. Upon triggering the PH test, the clustering model is rebuilt from the current one and the reservoir using WAP \. The key issue here was to build the change indicator, monitored by the PH test, in order to preserve the computational cost vs accuracy tradeoff. In this paper, we monitored the ratio of outliers over a sliding window and adapted the change detection threshold in real-time. The proposed approach STRAP was theoretically analyzed on guaranteeing acceptable distortion loss when exemplars slightly drift from the already selected ones, on consuming small amount of memory with little variation, and on requiring acceptable computing time that depends on the complexity of underlying distribution. The performance of STRAP in clustering quality and efficiency is empirically validated on KDD'99 benchmark problem and the URLs stream [IEEE 2013].

Jieming Shi et. al. studied for the first time the problem of Densitybased Clustering Places in Geo-Social Networks (DCPGS). Our clustering model extends the density-based clustering paradigm to consider both the spatial and social distances between places. We defined a new measure for the social distance between places, considering the social ties between users that visit them. Also said that our measure is shown to be more effective and efficient to compute, compared more complex ones based on node-to-node graph proximity. We analyzed the effectiveness of DCPGS via case studies and demonstrated that DCPGS can discover clusters with interesting properties (i.e., barrier-based splitting, spatially loose clusters, clusters with fuzzy boundaries), which cannot be found by merely using spatial clustering. Besides, we designed two evaluation measures to quantitatively evaluate the social quality of clusters detected by DCPGS or competitors, called social entropy and community score, which also confirm that DCPGS is more effective than alternative approaches.[IEEE 2017].

Kamran Khan et. al. said that Data Mining is all about data analysis techniques. It is useful for extracting hidden and interesting patterns from large datasets. Clustering techniques are important when it comes to extracting knowledge from large amount of spatial data collected from various applications including GIS, satellite images, X-ray crystallography, remote sensing and environmental assessment and planning etc. To extract useful pattern from these complex data sources several popular spatial data clustering techniques have been proposed. DBSCAN (Density Based Spatial Clustering of Applications with Noise) is a pioneer density based algorithm. It can discover clusters of any arbitrary shape and size in databases containing even noise and outliers. DBSCAN however are known to have a number of problems such as: (a) it requires user's input to specify parameter values for executing the algorithm; (b) it is prone to dilemma in deciding meaningful clusters from datasets with varying densities; (c) and it incurs certain computational complexity. Many researchers attempted to enhance the basic DBSCAN algorithm, in order to overcome these drawbacks, such as VDBSCAN, FDBSCAN, DD_DBSCAN, and IDBSCAN. In this study, we survey over different variations of DBSCAN algorithms that were proposed so far. These variations are critically evaluated and their limitations are also listed [IEEE 2014].

Chetan Dharni et. al. Analyzed that the varied density clusters and time complexity are the main problems which has been improved in many DBSCAN variations. The time complexity is reduced to O(N) with the help of indexing in DBSCAN algorithm. The problem of varied density and increasing dimensionality effect the performance of DBSCAN algorithm. Still it needs more improvement [IJCST 2013].

Fang Huang et. al. said that Density-based spatial clustering of applications with noise (DBSCAN) is a density-based clustering algorithm that has the characteristics of being able to discover clusters of any shape, effectively distinguishing noise points and naturally supporting spatial databases. DBSCAN has been widely used in the field of spatial data mining. This paper studies the parallelization design and realization of the DBSCAN algorithm based on the Spark platform, and solves the following problems that arise when computing macro data: the requirement of a great deal of calculation using the single-node algorithm; the low level of resource-utilization with the multi-node algorithm; the large time consumption; and the lack of instantaneity. The experimental results indicate that the proposed parallel algorithm design is able to achieve more stable speedup at an increased involved spatial data scale[MDPI 2017].

S. N o.	Author/Title/Year/Publication	Method Used	Description

1.	Xiangliang Zhang et. Al./Data Stream Clustering with Affinity Propagation/ /HAL-Inria2014	Affinity Propagation (AP) algorithm	Clustering data streams with evolving data distributions. STRAP confronts the arriving items to the current AP model.
2.	Jieming Shi et.l./Density-based Place Clustering in Geo-Social Networks/2014/ SIGMOD	DBSCAN	Studied for the first time the problem of Density based Clustering Places in Geo-Social Networks (DCPGS). Clustering model extends the density-based clustering paradigm to consider both the spatial and social distances between places.
3.	Saif Ur Rehman et. Al./DBSCAN: Past, Present and Future/2016/IEEE	DBSCAN	Presented the summary information of the different enhancement of density-based clustering algorithm called the DBSCAN.
4.	SANJAY CHAKRABORTY et. Al./Analysis and Study of Incremental DBSCAN Clustering Algorithm/2014 /ARXIV	DBSCAN	Propose an improve DBSCAN clustering approach which provides better and fastest result compare to the existing DBSCAN clustering algorithm up to some certain point of change in the original database.
5.	Xiangliang Zhang et. Al./ Data Stream Clustering With Affinity Propagation/IEEE 2016	STRAP	Presented STRAP algorithm combines AP with a statistical change point detection test; the clustering model is rebuilt whenever the test detects a change in the underlying data distribution. Besides the validation on two benchmark data sets, the presented algorithm is validated on a real-world application.
6.	Ali Seyed Shirkhorshidi et. Al/Big Data Clustering: A Review/Springer 2014	MapReduce, Parallel Clustering	This study is aimed to review the trend and progress of clustering algorithms to cope with big data challenges from very first proposed algorithms until today's novel solutions.
7.	Zexuan Ji et. Al./Interval-valued possibilistic fuzzy C-means clustering algorithm/Elsevier 2013	Fuzzy C-means clustering	Author compared the proposed algorithm with five fuzzy clustering approaches, including the FCM, PCM, PFCM, IFCM and IPCM, on two-dimensional Gaussian data sets and four multi-dimensional benchmark data sets. Also applied these clustering techniques to segment the brain magnetic resonance images and natural images. Results of this paper show that the proposed IPFCM algorithm is more robust to outliers and initializations and can produce more accurate clustering results.

III. Findings

There are several implementations of DBSCAN algorithms.

El-Sonbaty et. al. has proposed a efficient density based algorithm which require preprocessing over input dataset to improve the performance of clustering, dataset partitioning used in pre-processing step. Next stage is merge stage for merging the nearest dense region which can be done by applying the DBSCAN separately on each partition in order to achieve final cluster.

Advantage:

1. Algorithm is scalable.
2. In future parallel version of algorithm can be implemented which will improve the performance of algorithm.

Disadvantage:

1. Before clustering preprocessing over input dataset is required.
2. I/O load may be augmented.

Adriano Moreira et. al. has proposed DBSCAN implementation which starts by identifying the k nearest neighbours of each point and identify the farthest k nearest neighbour (in terms of Euclidean distance) . The average of all this distance is then calculated. After that, for each point of the dataset the algorithm identifies the directly density-reachable points (using the Eps threshold provided by the user) and classifies the points into core or border points.

Advantage:

1. Does not entail priori specification of number of clusters.
2. Capable to recognize noise data while clustering.

Disadvantage:

1. Algorithm miss the mark in case of varying density clusters.
2. Performance degrades in the case of high dimension input dataset.

Duan et.al. has proposed Local Density Based Spatial Clustering Algorithm With Noise having time complexity is O(n). In the proposed algorithm, LOF (local outlier factor) is used to detect the noise and has overawed the problem of global density parameter.

Advantage:

1. Easier for the user to prefer the suitable parameters.

Disadvantage:

1. Cluster analysis is difficult.Bing Liu has proposed Fast Density Based Clustering Algorithm for Large Databases in which objects are organized by certain dimensional coordinates. This algorithmuses global Eps parameter. If the value of Eps is less, then the few or single cluster containing all objects is formed and if the value of Eps is high, many trifling clusters are produced.

Advantage:

1. Time complexity condensed.

Disadvantage:

1. The problem of diverse density clusters is not analyzed.

Zhang et. al has proposed A Linear DBSCAN Algorithm Based on LSH (Locality-Sensitive Hashing). The improvement of using LSH is that it decreases the time complexity and the scale of data.The time complexity of algorithm is O(NlogN).

Advantage:

1. Comparative original DBSCAN time complexity reduces.
2. Outlierseradicated using LSH and nearest neighbor points are obtained.

Disadvantage:

1. Preprocessing over input dataset is required.

Pathway et. al. proposed A New Scalable Parallel DBSCAN Algorithm Using Disjoint-Set Data Structure To build clusters, a tree based bottom-up method is used. The disjoint-set data structure is cast-off to break the data access order and to achieve the merging efficiently.

Advantage:

1. The master-slave method speeds up the process.

Disadvantage:

1. Upsurges the I/O load and also effects the cost.

IV. Proposed Methodology

To overcome the different bottlenecks of sequential DBSCAN algorithm we have proposed multicore programming based DBSCAN algorithm which will use K-Dimension tree as data structure for parallel implementation.

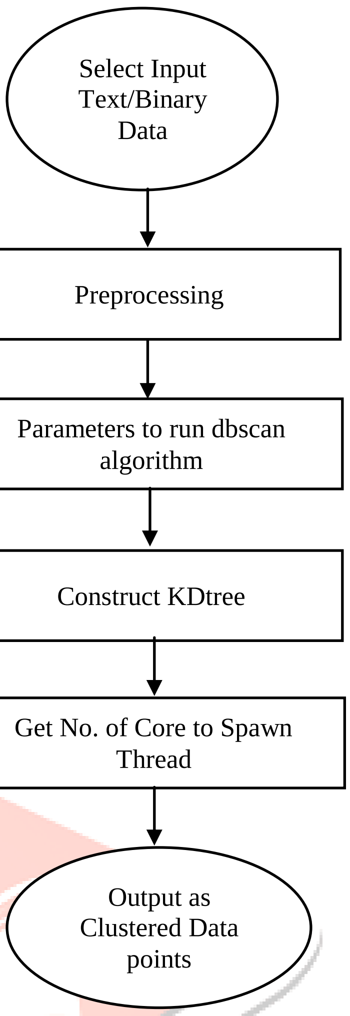

Select Input Text/Binary Data
Preprocessing
Parameters to run dbscan algorithm
Construct KDtree
Get No. of Core to Spawn Thread
Output as Clustered Data points

Fig.-3 Proposed System Architecture

Proposed Algorithm

```
Step-1 Select input text or binary dataset.
Step-2 Set DBSCAN parameter eps and minpoints.
Step-3 Remove noise.
Step-4 Construct KD Tree.
Step-5 Construct Minimum spanning tree using REM's algorithm.
Step-6 Set #pragma to each node.
Step-7 Union of cluster node.
Step-8 Merge all visited and non-visited node of KD tree.
```

KD Tree Construction

```
Step-1. if P contains only one point
Step-2. then return a leaf storing this point
Step-3. else if depth is even
Step-4. then Split P with a vertical line ` through the median x-coordinate into P1 (left of or on `) and P2 (right of `)
Step-5. else Split P with a horizontal line ` through the median y-coordinate into P1 (below or on `) and P2 (above `)
Step-6. νleft ← BuildKdTree(P1,depth+1)
Step-7. νright ← BuildKdTree(P2,depth+1)
Step-8. Create a node ν storing `, make νleft the left child of ν, and make νright the right child of ν.
Step-9. return ν
```

V. Result and Discussion

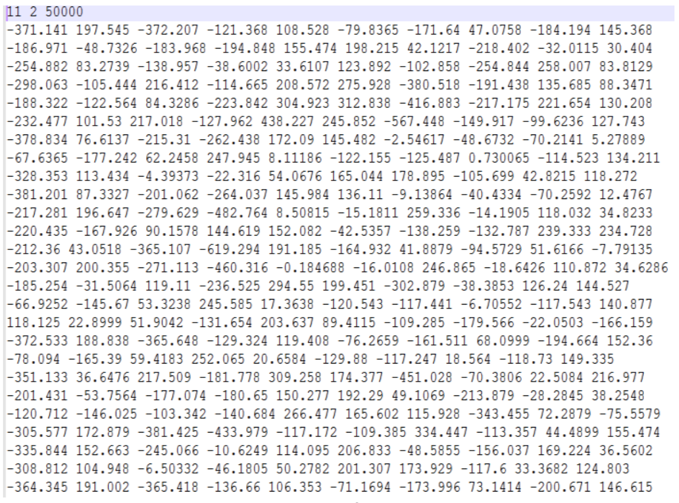
For implementation of proposed algorithm we have used OpenMP, gcc compiler in Ubuntu 16.0 operating system. OpenMp is an open source library for multicore programming. We have implemented sequential DBSCAN and proposed DBSCAN and tested on different dataset. Proposed implementation takes following input:

1. Input file name
2. Input parameter of DBSCAN, min points to form a cluster.
3. Input parameter of DBSCAN, radius or threshold on neighborhoods retrieved.
4. Clustering results file name.
5. Number of threads

Example

```
./multicore_dbscan -i filename -b -m minpts -e epsilon -o output -t threads
./multicore_dbscan -i example.dat –b -m 5 -e 30 -t 8 -o output
./multicore_dbscan -i example1.dat–b -m 5 -e 30 -t 8 -o output
./multicore_dbscan -i clus5k.txt -m 5 -e 30 -t 8 -o output
./multicore_dbscan -i clus7.5k.txt -m 5 -e 30 -t 8 -o output
./multicore_dbscan -i clus10k.txt -m 5 -e 30 -t 8 -o output
./multicore_dbscan -i clus12.5k.txt -m 5 -e 30 -t 8 -o output
./multicore_dbscan -i clus15k.txt -m 5 -e 30 -t 8 -o output
./multicore_dbscan -i clus20k.txt -m 5 -e 30 -t 8 -o output1
./multicore_dbscan -i clus25k.txt -m 5 -e 30 -t 8 -o output
./multicore_dbscan -i clus30k.txt -m 5 -e 30 -t 8 -o output
./multicore_dbscan -i clus35k.txt -m 5 -e 30 -t 8 -o output
./multicore_dbscan -i clus40k.txt -m 5 -e 30 -t 8 -o output
./multicore_dbscan -i clus45k.txt -m 5 -e 30 -t 8 -o output
./multicore_dbscan -i clus50k.txt -m 5 -e 30 -t 8 -o output
./multicore_dbscan -i clus60k.txt -m 5 -e 30 -t 8 -o output
./multicore_dbscan -i clus70k.txt -m 5 -e 30 -t 8 -o output
./multicore_dbscan -i clus75k.txt -m 5 -e 30 -t 8 -o output
./multicore_dbscan -i clus80k.txt -m 5 -e 30 -t 8 -o output
./multicore_dbscan -i clus90k.txt -m 5 -e 30 -t 8 -o output
./multicore_dbscan -i clus100k.txt -m 5 -e 30 -t 8 -o output
```

S. No.	Dataset size	Execution time in sec	
		Sequential	Proposed
1.	54	0.0011	0.000990349
2.	106	0.052	0.000991960
3.	5000	0.8837	0.000993491
4.	7500	2.2899	0.00117969
5.	10000	5.5712	0.001884051
6.	12500	5.5556	0.00372282
7.	15000	7.9576	0.003791021
8.	20000	26.6375	0.0038201
9.	25000	32.8844	0.00388885
10.	30000	32.1504	0.00554149
11.	35000	45.7200	0.0056283
12.	40000	114.8900	0.005866381
13.	45000	140.6387	0.00627656
14.	50000	143.8052	0.006368
15.	60000	154.0679	0.0064895
16.	70000	209.5745	0.006605
17.	75000	289.8763	0.007475
18.	80000	527.0518	0.007840121
19.	90000	630.1514	0.00858002
20.	100000	702.6091	0.124919



```
11 2 50000
-371.141 197.545 -372.207 -121.368 108.528 -79.8365 -171.64 47.0758 -184.194 145.368
-186.971 -48.7326 -183.968 -194.848 155.474 198.215 42.1217 -218.402 -32.0115 30.404
-254.882 83.2739 -138.957 -38.6002 33.6107 123.892 -102.858 -254.844 258.007 83.8129
-298.063 -105.444 216.412 -114.665 208.572 275.928 -380.518 -191.438 135.685 88.3471
-188.322 -122.564 84.3286 -223.842 304.923 312.838 -416.883 -217.175 221.654 130.208
-232.477 101.53 217.018 -127.962 438.227 245.852 -567.448 -149.917 -99.6236 127.743
-378.834 76.6137 -215.31 -262.438 172.09 145.482 -2.54617 -48.6732 -70.2141 5.27889
-67.6365 -177.242 62.2458 247.945 8.11186 -122.155 -125.487 0.730065 -114.523 134.211
-328.353 113.434 -4.39373 -22.316 54.0676 165.044 178.895 -105.699 42.8215 118.272
-381.201 87.3327 -201.062 -264.037 145.984 136.11 -9.13864 -40.4334 -70.2592 12.4767
-217.281 196.647 -279.629 -482.764 8.50815 -15.1811 259.336 -14.1905 118.032 34.8233
-220.435 -167.926 90.1578 144.619 152.082 -42.5357 -138.259 -132.787 239.333 234.728
-212.36 43.0518 -365.107 -619.294 191.185 -164.932 41.8879 -94.5729 51.6166 -7.79135
-203.307 200.355 -271.113 -460.316 -0.184688 -16.0108 246.865 -18.6426 110.872 34.6286
-185.254 -31.5064 119.11 -236.525 294.55 199.451 -302.879 -38.3853 126.24 144.527
-66.9252 -145.67 53.3238 245.585 17.3638 -120.543 -117.441 -6.70552 -117.543 140.877
118.125 22.8999 51.9042 -131.654 203.637 89.4115 -109.285 -179.566 -22.0503 -166.159
-372.533 188.838 -365.648 -129.324 119.408 -76.2659 -161.511 68.0999 -194.664 152.36
-78.094 -165.39 59.4183 252.065 20.6584 -129.88 -117.247 18.564 -118.73 149.335
-351.133 36.6476 217.509 -181.778 309.258 174.377 -451.028 -70.3806 22.5084 216.977
-201.431 -53.7564 -177.074 -180.65 150.277 192.29 49.1069 -213.879 -28.2845 38.2548
-120.712 -146.025 -103.342 -140.684 266.477 165.602 115.928 -343.455 72.2879 -75.5579
-305.577 172.879 -381.425 -433.979 -117.172 -109.385 334.447 -113.357 44.4899 155.474
-335.844 152.663 -245.066 -10.6249 114.095 206.833 -48.5855 -156.037 169.224 36.5602
-308.812 104.948 -6.50332 -46.1805 50.2782 201.307 173.929 -117.6 33.3682 124.803
-364.345 191.002 -365.418 -136.66 106.353 -71.1694 -173.996 73.1414 -200.671 146.615
```

Fig.-5 Snippet of Input dataset

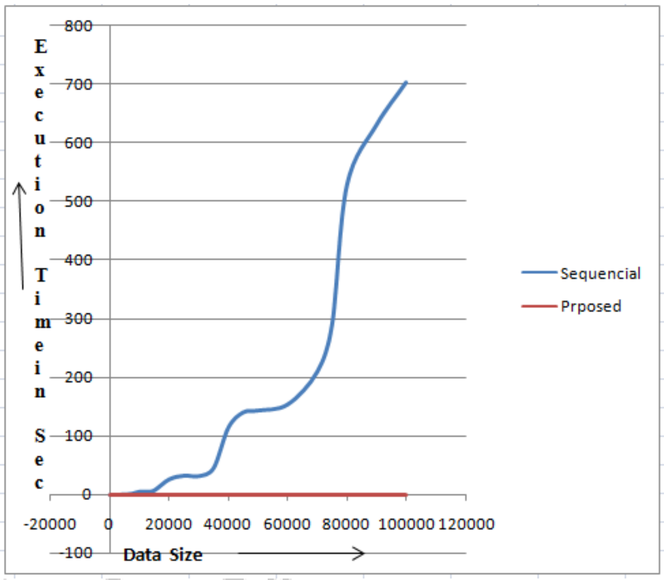
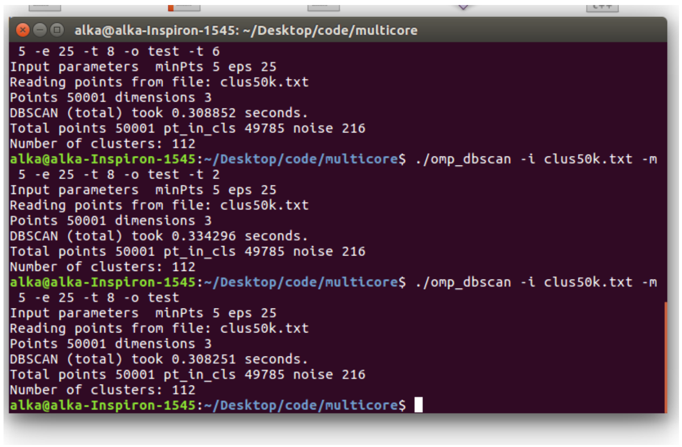


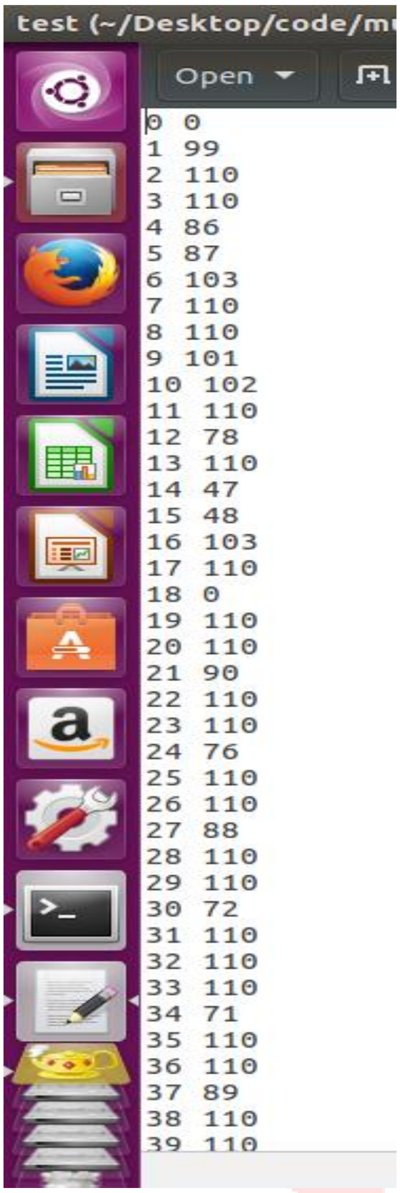
Fig.-6 Comparison based on Execution Time



```
 5 -e 25 -t 8 -o test -t 6
Input parameters  minPts 5 eps 25
Reading points from file: clus50k.txt
Points 50001 dimensions 3
DBSCAN (total) took 0.308852 seconds.
Total points 50001 pt_in_cls 49785 noise 216
Number of clusters: 112
alka@alka-Inspiron-1545:~/Desktop/code/multicore$ ./omp_dbscan -i clus50k.txt -m
 5 -e 25 -t 8 -o test -t 2
Input parameters  minPts 5 eps 25
Reading points from file: clus50k.txt
Points 50001 dimensions 3
DBSCAN (total) took 0.334296 seconds.
Total points 50001 pt_in_cls 49785 noise 216
Number of clusters: 112
alka@alka-Inspiron-1545:~/Desktop/code/multicore$ ./omp_dbscan -i clus50k.txt -m
 5 -e 25 -t 8 -o test
Input parameters  minPts 5 eps 25
Reading points from file: clus50k.txt
Points 50001 dimensions 3
DBSCAN (total) took 0.308251 seconds.
Total points 50001 pt_in_cls 49785 noise 216
Number of clusters: 112
alka@alka-Inspiron-1545:~/Desktop/code/multicore$
```

Fig.-7 Proposed System Output

Fig.-7 Shows the snippet of our proposed algorithm implementation, which shows number of point noise and execution time I/O time.



```
0 0
1 99
2 110
3 110
4 86
5 87
6 103
7 110
8 110
9 101
10 102
11 110
12 78
13 110
14 47
15 48
16 103
17 110
18 0
19 110
20 110
21 90
22 110
23 110
24 76
25 110
26 110
27 88
28 110
29 110
30 72
31 110
32 110
33 110
34 71
35 110
36 110
37 89
38 110
39 110
```

Fig.-8 Proposed System Output File

VI. Conclusion

Data clustering is a renowned technique in numerous areas of computer science and associated domains.DBSCAN is the illustrious method density centered data clustering method. Several implementation of DBSCAN has been done which made improvement in different aspects of DBSCAN and improved the performance. Existing parallel implementations of DBSCAN clustering algorithm espouse a master-slave tactic which can simply grounds an unbalanced workload and therefore result in low parallel competence. Earlier techniques are not time efficient. Multicore programming facilitated us to make time efficient DBSCAN algorithm.

References

[1] Xiangliang Zhang et. Al. Data Stream Clustering with Affinity Propagation HAL-Inria 2014
[2] Jieming Shi et.l. Density-based Place Clustering in Geo-Social Networks 2014 SIGMOD
[3] Saif Ur Rehman et. Al. DBSCAN: Past, Present and Future 2016 IEEE
[4] SANJAY CHAKRABORTY et. Al. Analysis and Study of Incremental DBSCAN Clustering Algorithm 2014 ARXIV
[5] Xiangliang Zhang et. Al. Data Stream Clustering With Affinity Propagation IEEE 2016
[6] Ali Seyed Shirkhorshidi et. Al Big Data Clustering: A Review Springer 2014
[7] Zexuan Ji et. Al. Interval-valued possibilistic fuzzy C-means clustering algorithm Elsevier 2013
[8] K.Kameshwaran, K.Malarvizhi Survey on Clustering Techniques in Data Mining IJCSIT 2014
[9] Ilias K. Savvas, and Dimitrios Tselios Parallelizing DBSCAN Algorithm Using MPI IEEE 2016