AN APPROACH TO SPEEDUP DENSITY BASED CLUSTERING FOR LARGE SCALE DATASETS BASED ON FUZZY TECHNIQUE AND PARALLEL-DISTRIBUTED PROCESSING

Axyonov Sergey Vladimirovich[1], Lycom Dmitri Nicolayevich[2]

[1] – PhD, Assistant Professor, Director of TPU's Center of Supercomputing, National Research Tomsk Polytechnic University, Tomsk, Russia

2 – BSc, Master Student of Institute of Cybernetics, National Research Tomsk Polytechnic University, Tomsk, Russia

Annotation. The article presents a new parallel-distributed approach to speed-up the processing of data mining methods implemented to analysis of star energy activities database from ESA. This technique uses Gustaffson-Kessel fuzzy clustering procedure to perform the effective data distribution and original method to get the most general clustering from some generated versions. It's shown how can increase the performance of clustering by presented approach.

Key words. Parallel-Distributed Clustering. Space Dataset Processing and Analysis. Fuzzy Clustering.

Introduction

The Space Exploration is one of direction of high intellectual complex software implementation. The software developing for the space projects needs specific system requirements, reliability and performance. The goal of our project is to create such software to analyze large-scale datasets that store information about stars features.

Next year the European Space Agency (ESA) will launch the spaceship to perform the project to create a large-scale map of the Universe [1]. It's called Gaia mission. The Gaia spacecraft is going to discover new celestial objects, such as failed stars and extra-solar planet and to get a lot of information from billiards stars. It's impossible to analyze this dataset receiving from Gaia systems by the standard data mining technique because of large scale of the set. The most effective way to explore the data and detect dependencies between objects in the set is the usage of parallel-distributed computing techniques to increase the performance of intellectual algorithms [2]. However advanced high-performance hardware to implement such methods needs a new way to computation and it needs the transformation of developed software. The more detail our objective is to create effective parallel-distributed technique to analyze some generated dataset connected with the Gaia project and form clusters of star energy activity features with DBSCAN clustering procedure. The table 1 presents an all attributes will be received from space sensors of Gaia.

Table. 1 Attribute description

Attribute Name	**Meaning**
log-f1	Log of the first frequency
log-f2	Log of the second frequency
log-af1h1-t	Log amplitude, first harmonic, first frequency
log-af1h2-t	Log amplitude, second harmonic, first frequency
log-af1h3-t	Log amplitude, third harmonic, first frequency
log-af1h4-t	Log amplitude, fourth harmonic, first frequency
log-af2h1-t	Log amplitude, first harmonic, second frequency
log-af2h2-t	Log amplitude, second harmonic, second frequency

log-crf10	Amplitude ratio between harmonics of the first frequency
pdf12	Phase difference between harmonics of first frequency
Varrat	Variance ratio before and after first frequency subtraction
B-V	Colour index
V-I	Colour index

DBSCAN Analysis

Next year we are going to analyze the performance of the software with real datasets. But now the real data are unavailable and ESA provides synthetic datasets based on the OGLE subset of 2193 objects that are generated by Uninova Institute, Lisbon, Portugal. To generate the synthetic datasets, each original row form OGLE was replicated k times and Gaussian random noise added; the original dataset characteristics are maintained, as well as the overall distribution of the various star types. In table 2 information about synthetic datasets is presented:

Table. 2 Synthetic Datasets Description

Dataset	# Instances	# Attributes	Usage
Synth43K	43860	13	Validation of clustering quality and implementation of algorithm
Synth-10^5	100878	13	Scalability performance testing
Synth-10^6	1008780	13	Scalability performance testing
Synth-10^7	10087800	13	Scalability performance testing
Synth-10^8	100878000	13	Scalability performance testing

Although there are many clustering procedures in data mining technology the DBSCAN clustering procedure is the most desirable to analyze stars features. The main advantage of DBSCAN clustering procedure is that it allows to get clusters with any shape [3]. The main problem of this procedure is a cost. By comparison with other techniques such as SOM [4] or K-means [5] it works by some digits slower. Hereby we describe the transformation of the standard algorithm to achieve higher performance.

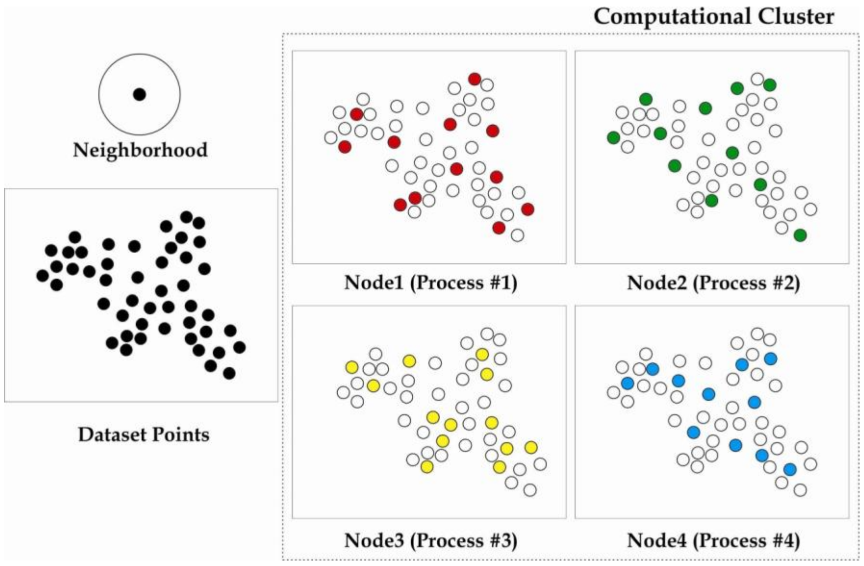

Fig. 1. Random data distribution of points in the dataset over the communicator doesn’t allow to form clusters within process (the number of points to form a cluster equals to 5 and the neighborhood is shown)

When we use distributed computation it is necessity to split analyzing information over nodes in the communicator. The random data scattering is ineffective. Figure 1 illustrates the random points distribution. To form a cluster the algorithm needs at least 5 points in the point neighborhood. After random vector distribution there are no such point that has more even four points in its neighborhood and each process can’t detect part of cluster [6].

To avoid the problem we have to perform the preclustering to get effective scattering the points of dataset or rows [7]. The effective scattering means that all rows to be sent to the first process have smaller distances between each other then distances between them and rows to sent to the another process. The figure 2 shows the difference between random and suggested distributions.

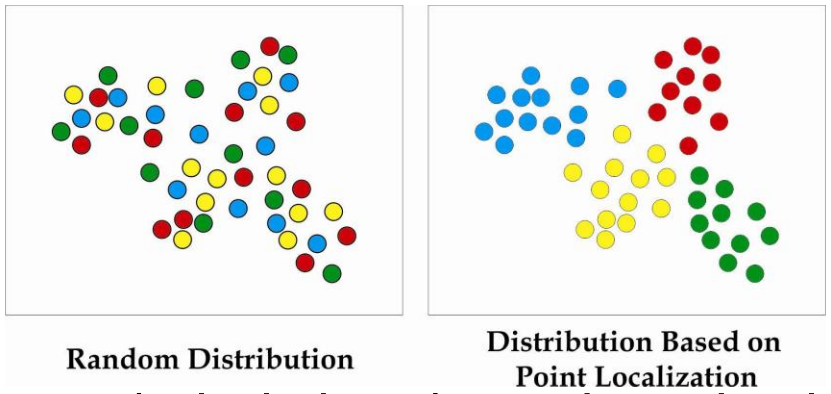

Fig.2 Comparison of random distribution of vectors and suggested point lacalization distribution. Right distribution lets to create connected clusters in the corresponding processes.

To reach this distribution it is necessary to use a sophisticated clustering algorithm that allows to generate clusters based on point density and to give additional information about membership probability to all clusters.

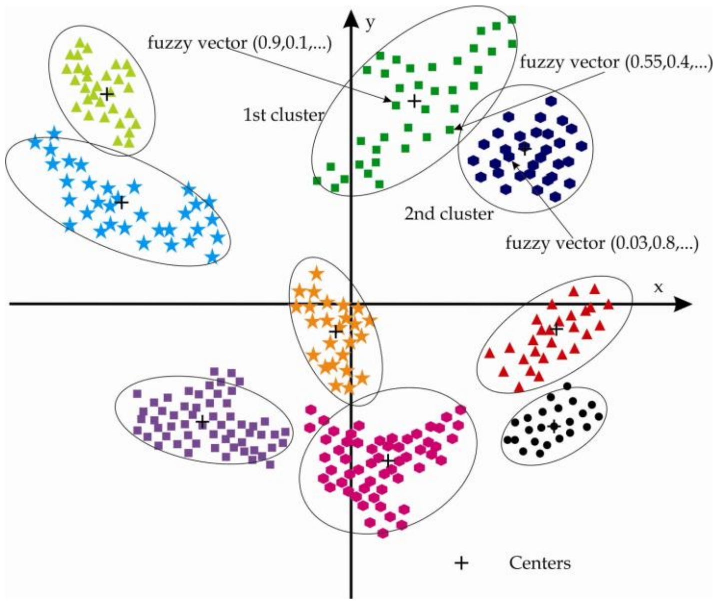
y
x
fuzzy vector (0.9,0.1,...)
fuzzy vector (0.55,0.4,...)
1st cluster
2nd cluster
fuzzy vector (0.03,0.8,...)
Centers

Fig.3 An example of the Gustaffson-Kessel clustering for the dataset. Borders are shown for membership functions 0.5 levels.

Because of the following reasons such as fuzzy ellipse-shapes cluster generation, correlation analysis for each cluster and probability to use membership function value as a measure that give useful information about location of each point over detected clusters we would like to suggest use Gustaffson-Kessel fuzzy clustering technique to perform this scattering. In figure 3 an example of Gustaffson-Kessel clustering implementation is shown. We note that the final result depends on algorithm precision and the number of cluster that we want to detect. So it's very important to know the number of clusters. We suggest to use the following technique. If there are N multi-core nodes in the computational cluster the goal of the preclustering is to detect N interconnected groups of points. It's no matter the number of real clusters is more or less then N. If it's less then N after the DBSCAN procedure we carry the join of clusters out.

Otherwise some clusters detected by Gustaffson-Kessel technique have several real groups of points. Anyway the processing speedup is made.

Algorithm description

The following algorithm describes some steps to get effective data distribution.

Step 1. Select N groups of points (each set has 1000 points) to clusterize with Gustaffson-Kessel clustering procedure and scatter them over the communicator.

Step 2. Each process executes Gustaffson-Kessel algorithm and saves centers of clusters and covariance matrices. In figure 4 some different clusterings are presented. But what clustering from them is the best to split the dataset? Hereby my suggest original method to detect the equivalent clusters from different nodes.

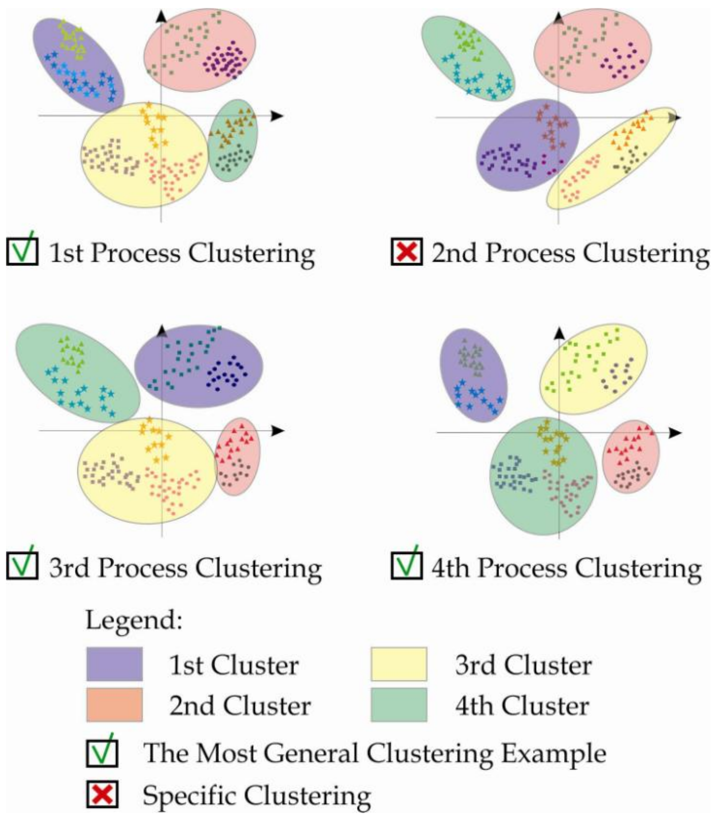

Fig.4 Four different generated clusterings in the communicator. Hereby some examples of the most general clustering are shown.

The most distributed clustering in the communicator is called the most general clustering. The next steps allow to get it.

Step 3. Select 1000 random points from the dataset to clusterize them with detected clusters and broadcast the set over the communicator. Each process gets this set and finds the membership function value for each point in the received message.

Step 4. The most general clustering search. Analysis the fuzzy matrices that are formed for the test set to get equivalence of detected clusters. Select an alternative from the list of equivalent clusterings and broadcast it.

There are three clusterings in figure 5 and point distributions in the A-C clusterings say that only tha A and C clusterings are similar. After clustering procedures nodes detected 4 clusters and gave them IDs. The color in figure means ID. The ID equals to 1, 2, 3, 4 for the blue, red, yellow and green clusters respectively. Here the red cluster in the A clustering looks like red clustering in the B clustering and yellow clustering in the C. Circle diagrams show the point distribution in two comparable clusters in different processes. When we compare the A and B clusterings all points in the yellow cluster from A is distributed within blue and yellow clusters from B.

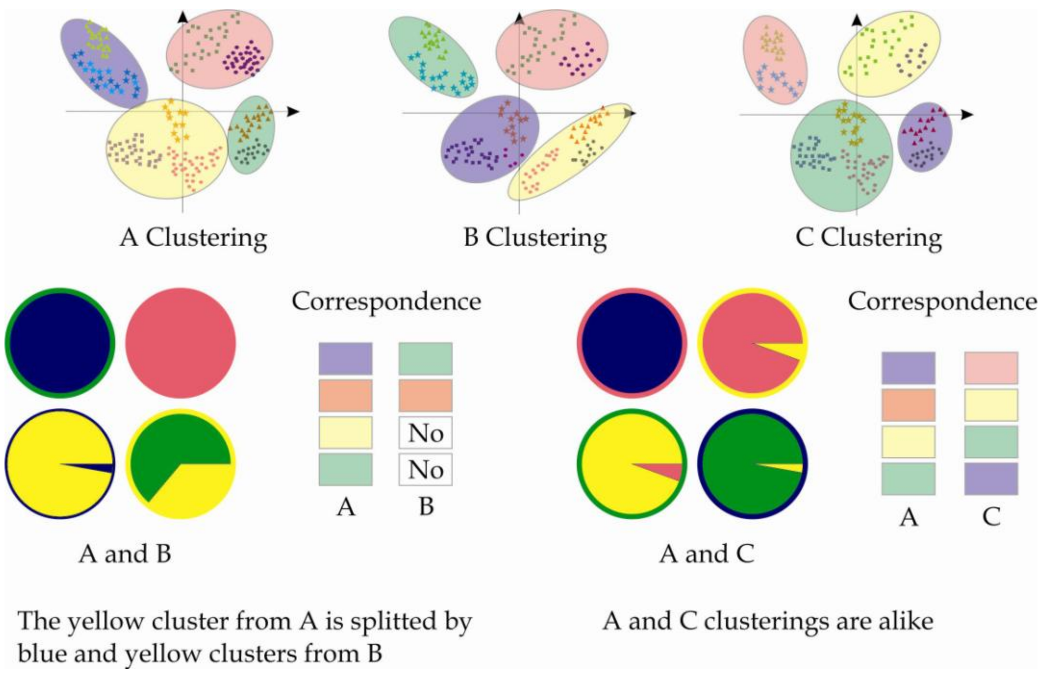
A Clustering
B Clustering
C Clustering
Correspondence
No
No
A
B
A and B
Correspondence
A
C
A and C
The yellow cluster from A is splitted by blue and yellow clusters from B
A and C clusterings are alike

Fig.5. An example of A, B and C clusterings correspondence analysis. There are some clusters in the A and B clusterings without equivalents. The A and C clusters have very similar results.

We can’t see the A and B the same because the big chunk of yellow cluster in A is located the yellow cluster in B. The idea of such comparison is to detect clusters that approximately cover each other.

Step 5. Now all nodes have the best detected clustering and we distribute initial dataset equal parts over the processes.

Step 6. Each process computes the membership function values for the received chunk and it sends all points located in the *j* cluster to the *j* process.

Finally, each node has only interconnected points with small distances between each other.

The next problem is the cluster join. Fuzzy matrices allow to detect points that are located at the border of clusters. The membership function values of such points are near 0.5 for the dominating cluster and a bit less 0.5 for other cluster(s). Hereby to estimate an atomic cluster presence between two nodes we should analyze only the distance between such points in these processes. It is real good performance achievement because it doesn't have to execute additional computation (everything has computed before yet) and the number of boundary points that are satisfied defined condition are much less the number of all boundary points and all the more the number of all points in the cluster [8].

Parallel-distributed processing and experiments

Usually each node has some cores and the combined usage of them is other important way to reach good performance. We recommend to use the multiple thread processing for good parallizable matrix operations in the Gustaffson-Kessel procedure, such as matrix determinant computation, matrix multiplication, subtraction, etc. During the DBSCAN procedure the multiple thread processing is desirable too. It's possible to distribute data over the team of threads like initial data were distributed over the communicator.

We created a C# program to implement the suggested technique and estimate its performance. In the program the combination of Message Passing Interface libarary for .NET [9] to control distributed computation and parallel libraries in the Microsoft .NET 4 [10] for internal node computation is used.

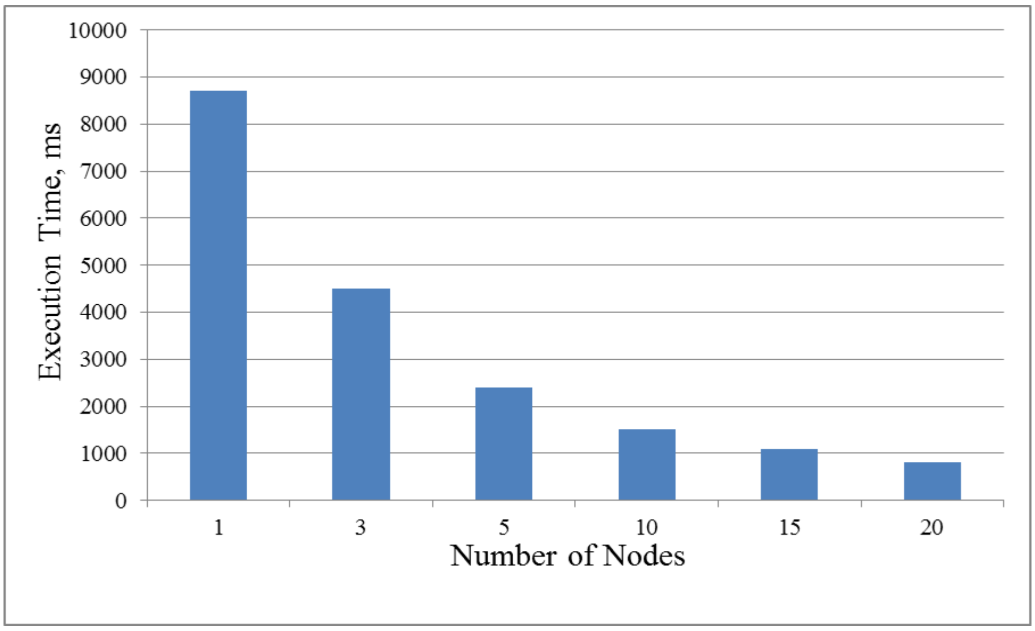

Fig.6 Performance of the suggested algorithm with different computational cluster capacity

The test sets are shown that the performance of suggested technique depends on the number of computational nodes and cores. Figure 6 illustrates dependence between the number of computational nodes (or cluster capacity) and the execution time with the Synth-10^5 dataset. The test with only one node is performed with single-core 3.4 GHz processor and other tests were with TPU computational cluster that has processors with similar frequency. It's clear that suggested technique speed-ups the performance. Standard

DBSCAN algorithm is needs approx. 18 min to process 10000 points dataset with then 10 nodes computational cluster needs just approx. 3 minutes to process such set.

Future development

The suggested technique has also ways to speed-up the processing. Unless the chunks of initial dataset to be sent to nodes have the same sizes. It leads that some nodes were able to process all points and others are still in process. In this case the nonblocking transmission of part of local dataset between overload nodes to free nodes is desirable. It balances the load of all parts in communicator and sequentially increases the performance.

Conclusion

In our work we were going to create a new parallel-distributed approach to speed-up performance of ESA dataset processing. We designed the new algorithm that allows effective vector distribution into computational cluster nodes based on Gustaffson-Kessel fuzzy clustering technique. The algorithm that detects the correspondence between different clustering is created. Next the C# program that performs all suggested computations is developed and tested. The set of test showed that original algorithm has a good prospective to be used with real-world tasks.

References

1. ESA Official Web-site on the Gaia project. ESA – Space Science - Gaia overview. URL: http://www.esa.int/esaSC/120377_index_0_m.html
2. Gropp W., Lusk E., Skjellum A. Using MPI, 2nd Edition: Portable Parallel Programming with the Message Passing Interface. Cambridge, MA, USA: MIT Press, 1999 – 310 p.
3. Han J., Kamber M., Pei J. Data Mining: Concepts and Techniques. Waltham, MA: Morgan Kauffman, 2012. - p.703.
4. Bigus J.P. Data Mining with Neural Networks. Cambridge: Cambridge University Press, 2009. – p.364.
5. Everitt B.S., Landau S., Leese M. Stahl D. Cluster Analysis. New York: John Wiley & Sons, 2011. – p.346.
6. Hoeppner F., Klawonn F., Kruse R., Runkler T. Fuzzy Cluster Analysis: Methods for Classification, Data Analysis and Image Recognition. New York: John Wiley & Sons, 1999. – p.300.
7. Abonyi J., Feil B. Cluster Analysis for Data Mining and System Identification. – Basel: Birkhaeuser Basel. – 2007. – p.203.
8. Indiana University Web-page on MPI.NET Standard. MPI.NET: High-Performance C# Library for Message Passing. URL: http:// http://osl.iu.edu/research/mpi.net/
9. Microsoft Corporation Official Web-site on parallel processing based on .NET. Design Patterns for Decomposition and Coordination on Multicore Architectures. URL: http://msdn.microsoft.com/en-us/library/ff963553.aspx
10. MPI Standard Official Site. Message Passing Interface Forum. URL: http://mpi-forum.org