UDC 004.722, 004.725, 519.254 10.23947/2587-8999-2018-2-1-33-43

Analysis of delays structure of interconnections in supercomputer by means of DBScan and divisive clustering algorithms[*]

A.N. Salnikov[**], A.A. Begaev, A.I. Maysuradze

Lomonosov Moscow State University, Moscow, Russia

In this paper we propose method for estimating and analysis measurements of delays in the computational cluster interconnection subsystem. Delays are combined into the set of pairs (source, destination). We have measurements of delays extracted by network_test2 utility from interconnections of following supercomputers: BlueGene/P, Lomonosov-1, Lomonosov-2 (Lomonosov MSU) and Jurope (Julich). We have clustered pairs of delays by DBscan and Divisive algorithms. Results of clusterisation revealed that DBScan is more accurate algorithm then divisive and allows to extract clusters, which correspond to the actual features in the supercomputer interconnections. Clusters gather near the same components of supercomputer network infrastructure. Gained clusters were visualized in 2-D by special tool, developed by authors.

Keywords: computing cluster; clustering algorithm; interconnect; parallel computing

Introduction. Modern supercomputers are used to solve a wide range of problems, in particular: mathematical modeling and processing of large amounts of data. Supercomputers typically have an architecture of a computer cluster. Computations on supercomputers occur in parallel on many processing elements: cluster nodes equipped with multi-core processors, however, transfers between processors and cluster nodes can significantly slow down the parallel program. In order to minimize application performance losses, it is necessary to understand between which processors the delivery of messages occurs most quickly or vice versa slowly. It is necessary to optimally distribute the computations for the processors making up the supercomputer.

The cluster nodes in the supercomputer are combined into some topology, via network cards, wires and switches. Parallel program developers often assume that the delay in sending messages between these nodes is determined by the number of switches that will pass this message, and the delay between neighboring processors is assumed to be a constant value. Because of this assumption it is easier to develop parallel programs. But in practice, such a model of delays is often too naive, which is due to the actual physical state of the communication environment of the cluster. Thus, it is necessary to take into account the state of the environment and to have an idea of the supercomputer topology (hidden not documented) when planning calculations.

It was noticed that delays between some pairs of source-receiver nodes have similar pattern and the number of classes of similar pairs is not so large. To detect these patterns, clustering of

[*] The research is partially financially supported by Russian Foundation for Basic Research (Projects No. 17-07-01562, 16-01-00196, 17-01-00917) and Lomonosov Moscow State University (Project No. 5.1.16).
[**] E-mail: salnikov@cs.msu.ru.

processor pairs will be performed. Clustering will allow us to better understand the real topology of a supercomputer, to find out which processors are closer to each other. Also, this will decrease the volume of storing delay data. In our previous work [5] we clustered data on delays, using another method. This article highlights a continuation of these works.

The aim of the paper is to investigate the methods of clustering applicable to this problem, as well as methods for visual representation of the clusters obtained.

Description of the main tasks. It is required to choose the optimal clustering algorithm and the metric to find the required clusters. Develop a format for storing the results of clustering, visualize the results.

Choice of metric. Let's consider the plots «message length-delay value» of four source-receiver nodes pairs, constructed by data obtained from Lomonosov-1 supercomputer testing result.

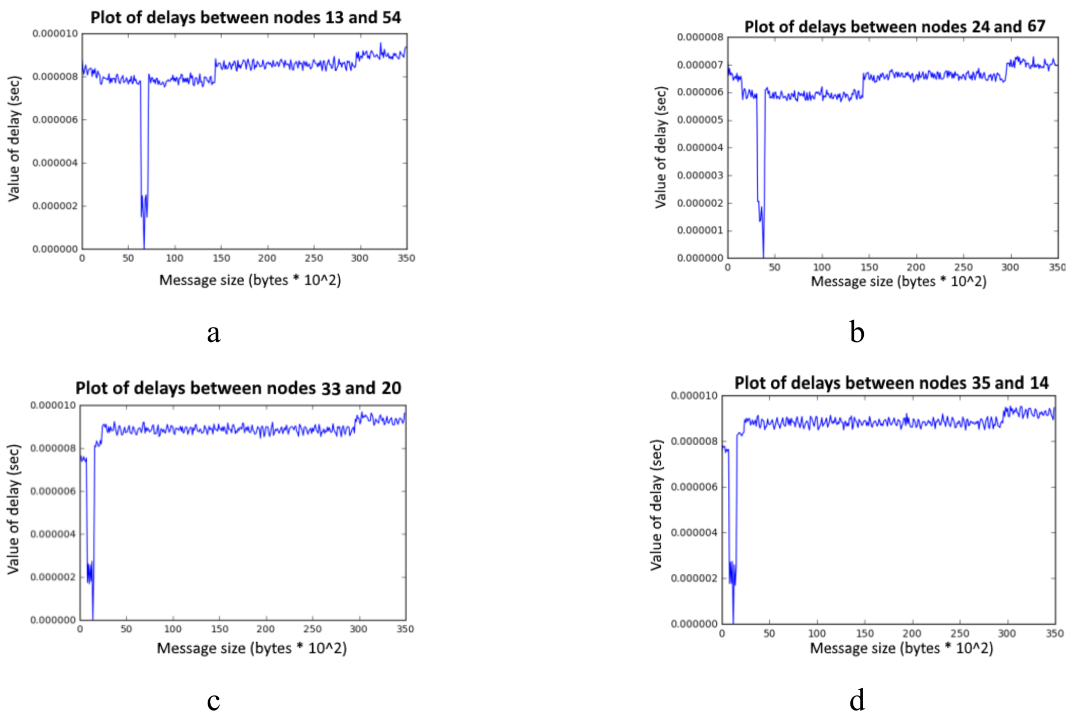

Fig. 1. Plots «message length-delay value» for node «source-sink» pairs of supercomputer Lomonosov-1 (13, 54), (24, 67), (33, 20), (35, 14)

As we can see on the plots *a, b, c, d* on the Fig. 1 the behaviors of delays between pairs (13, 54), (24, 67) and (33, 20), (35, 14) are similar. To find out which other pairs of nodes also have a same structure of delays, it is required to introduce a «distance» function between these plots. This function will represent the similarity of delays behavior between two selected source-receiver pairs: S_1 and S_2.

From the result of supercomputer testing we have extracted statistics of delays between the nodes: median – m, standard deviation – σ. The use of these statistics makes it possible to determine the «distance» function between pairs tolerant to variability in delays. This variability is accused by peaks in data of medians between nodes. Peak is sharp increase of delay value in some neighborhood of messages sizes. It is also necessary to normalize, for correct calculations of distances between the plots like on Fig. 1.

In this research Manhattan distance metric was suggested as basic. [3] In this metric, the influence of the peaks on the average distance between the objects is weak. Thus, the metric for the given problem takes the following form:

$$\rho(S_1, S_2) = \frac{\sum_{i=l_{min}}^{l_{max}} \frac{|m_i^{S_1} - m_i^{S_2}|}{(\sigma_i^{S_1})^2 + (\sigma_i^{S_2})^2}}{\sum_{j=l_{min}}^{l_{max}} \frac{1}{\left(\sigma_j^{S_1}\right)^2 + \left(\sigma_j^{S_2}\right)^2}} \qquad (1)$$

where l_{min} and l_{max} are the lengths of the smallest and largest messages, respectively. m_i corresponds to the median value of measured delay for fixed message size l_i, σ_i corresponds to the standard deviation of measured delay for for fixed message size l_i.

Choice of clustering algorithms. Because we do not know the finite number of clusters, but it is known that their number is small, it is necessary to choose algorithms that can work without a specific number of clusters and give a correct result. For these purposes, DBScan algorithms and the hierarchical clustering algorithm are suitable. They are widely used in practice.

Description of the DBScan algorithm. This algorithm was proposed by Martin Esther, Hans-Peter Kriegel in 1996 [6]. In this algorithm it is assumed that in space there are «condensations» of objects that form a cluster to each other. They are characterized by the greatest density within the cluster and the smallest outside. The main task of the DBScan clustering algorithm is to search for these condensations.

Our implementation of DBScan algorithm operates the following definitions:

- eps - the maximum distance between neighboring objects (cluster internal distance);
- minPts - the minimum number of neighboring objects around some object to consider it as core object;
- core object - an object in the eps-neighborhood of which located above minPts objects;
- boundary object - an object, which is reachable from one group of core objects (the distance between it and the nearest core object is less than eps).

The distance between objects is calculated by formula (1). A cluster is formed from set of core objects, located in eps-neighborhoods of each other, and set of boundary objects that can be reached from these core objects. All objects not included in any cluster consider as “noise”.

The input parameters of the algorithm are "eps" and "minPts". The algorithm works by the method of searching in width and stops with the exhaustion of all clustering objects.

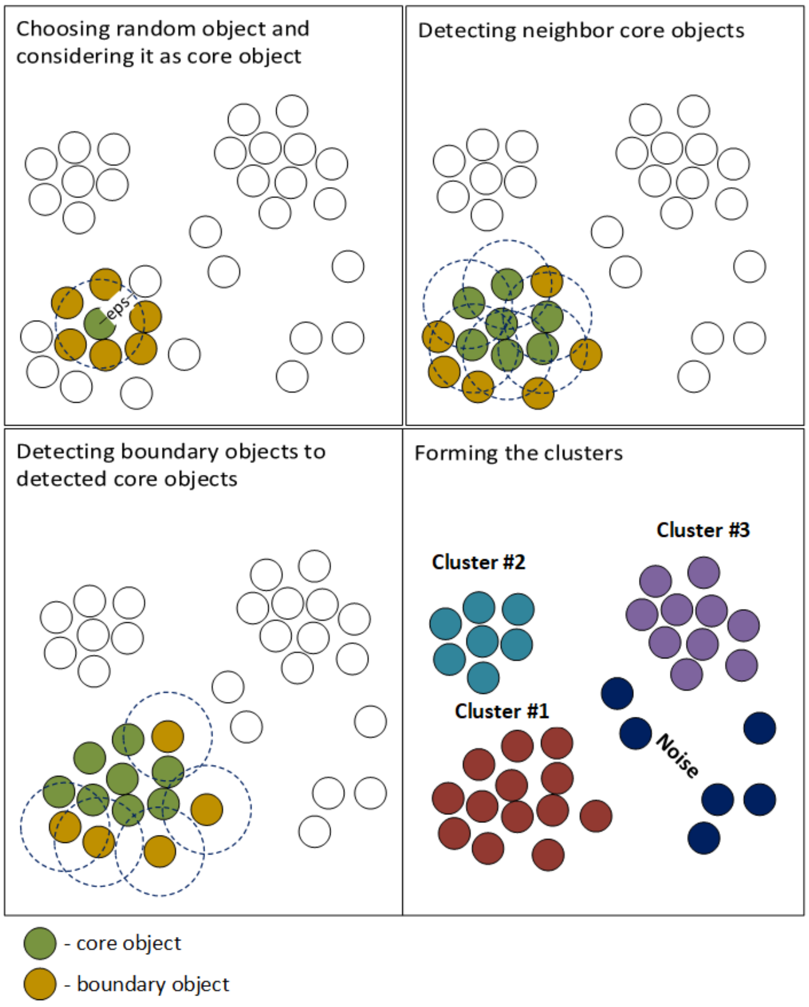
Choosing random object and considering it as core object
eps
Detecting neighbor core objects
Detecting boundary objects to detected core objects
Forming the clusters
Cluster #3
Cluster #2
Cluster #1
Noise
- core object
- boundary object

Fig. 2. Illustration of the DBScan algorithm operations. In this case minPts = 4.

Description of the algorithm of divisive clustering. The main idea of the divisional hierarchical clustering [1] is that all objects form one cluster. In this cluster a random object (S_O) is selected and the farthest object (S_A) to it is searched. Then, for the S_A object the farthest object (S_B) is searched. The distance between the elements S_A and S_B is called the cluster diameter. The distance between objects is calculated by formula (1). We will split the initial cluster by this diameter: the objects closer to the element S_A will belong to one subcluster and closer to S_B to another. This process is iterative: after this split each cluster is considered separately, and this algorithm applies to each cluster. As a result of this operation, a binary tree of clusters is obtained. The number of objects is large and to determine the end of clusters calculation the input parameter lvlNum is introduced. So, after in this binary tree the number of levels reaches lvlNum the algorithm stops.

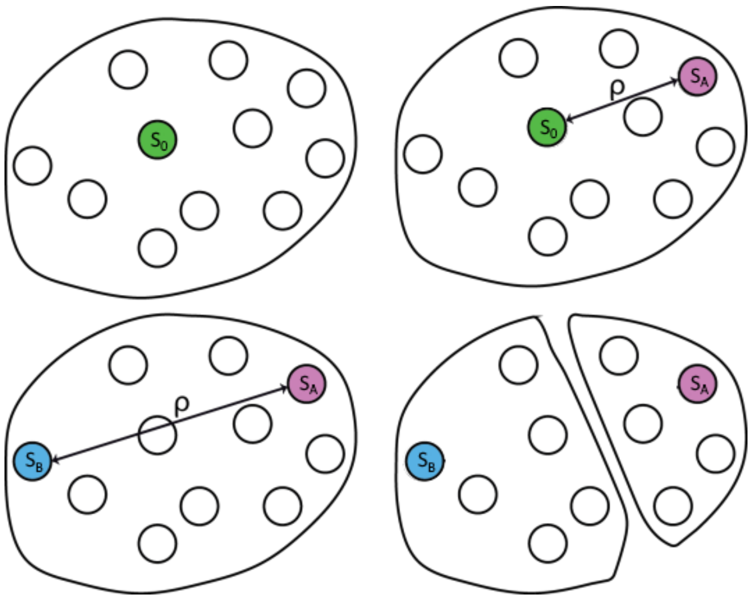

Fig. 3. Illustration of the divisive clustering algorithm step.

Description of the file format for saving the clustering result. Obtained clusters are saved in text format, described by following grammar:

$< digit > ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9".$

$< float > ::= \{ digit \}$ [" e "] ["+" | "-"] $\{ digit \}$.

< proc _ num > :: = " PROC _ NUM =" { digit }. // the number of processors.

< begmeslen > :: = " BEG _ MES _ LEN =" { digit }. // the length of the shortest message.

< endmeslen > :: = " END _ MES _ LEN =" { digit }. // the length of the longest message.

< steplen > :: = " STEP _ LEN =" { digit }. // the difference between the neighboring messages.

< clust > :: = " Cluster #" { digit }. // the number of cluster.

< pair > :: = "(" { digit } "," { digit } ")".

< cldata > :: = { pair }. // pairs of processors «receiver-source», included in the cluster.

< med > :: = { float } // mean delay values of pairs in the cluster at all message sizes.

< dev > :: = { float } // mean deviation values of pairs in the cluster at all message sizes.

Examples of the results of the program on real data:

```
PROC_NUM = 128, BEG_MES_LEN = 25, END_MES_LEN = 4000, STEP_LEN = 50

Cluster# 0

(0, 0) (1, 1) (2, 2) (3, 3) (4, 4) …

0 0 0 0 0 0 … 0 0 0

0 0 0 0 0 0 … 0 0 0

…

Cluster # 51

(1, 127) (2, 127) (3, 127) (13, 113) (13, 114)… (127, 1) (127, 2) (127, 3)

4.43708e-06 4.7181e-06 4.96488e-06 5.166e-06 5.57538e-06 … 5.87367e-06 5.97445e-06
5.96344e-06

1.99302e-22 3.98604e-22 9.96509e-23 -3.48778e-22 -2.49127e-22 … 1.49476e-22 9.96509e-23
3.98604e-22
```

The data obtained during testing of the Bluegene / P supercomputer was processed. The first 128 processors were selected, clustering was performed by the DBScan algorithm with eps = 0.01 and minPts = 10.

```
PROC_NUM = 128, BEG_MES_LEN = 0, END_MES_LEN = 10000, STEP_LEN = 100
Cluster# 0
(0, 0) (0, 1) (0, 2) (0, 3) (0, 4) … (127, 125) (127, 126) (127, 127)

1.22306e-06 1.30781e-06 1.3106e-06 1.31154e-06 1.312e-06

0 0 0 0 0 … 0 0 0

Cluster# 1
(0, 8) (0, 9) (0, 10) (0, 11) (0, 12) … (127, 117) (127, 118) (127, 119)

3.2883e-06 3.55088e-06 4.91068e-06 5.02736e-06 5.10795e-06 … 1.26559e-05 1.2784e-05 0

1.92714e-21 -3.07216e-20 1.36512e-21 1.89976e-22 -9.36534e-21 … 4.94235e-21 -2.48441e-21 0
```

The data obtained during the testing of the supercomputer Lomonosov was processed. The first 128 processors were selected, clustering was performed by the algorithm of divisional hierarchical clustering with lvlNum = 6.

Visual analysis of clusters. The obtained results of the clustering are visualized in matrix form. Horizontal axis corresponds to the number of "source" processor, vertical axis corresponds to the number of "receiver" processor. Each cluster obtains unique color in gradation of green, the blue color corresponds to the "noise" (special cluster with elements, not included in any cluster).

This visualization is performed in special tool, written in C++, using Qt 4.8.6, QWT 5.2.2 libraries. This tool as input receives the file with clustering results in format mentioned above.

Benchmarking of supercomputers. All input data for clustering algorithms described in this article was obtained in the result of running benchmarking application network_test2 [4, 7].

Bluegene / P. The number of processors is 128, begin message size 25 bytes, end message size 4000 bytes, step size 50 bytes, testing mode one_to_one, number of repeats 15.

Lomonosov-1. The number of processors is 128, begin message size 0 bytes, end message size 10000 bytes, step size 100 bytes, testing mode one_to_one, number of repeats 500.

Lomonosov-2. The number of processors is 118, begin message size 0 bytes, end message size 10000 bytes, step size 100 bytes, testing mode one_to_one, number of repeats 100.

Juropa. The number of processors is 128, begin message size 0 bytes, end message size 1100 bytes, step size 100 bytes, testing mode all_to_all, number of repeats 100.

Bluegene / P (128 processors)

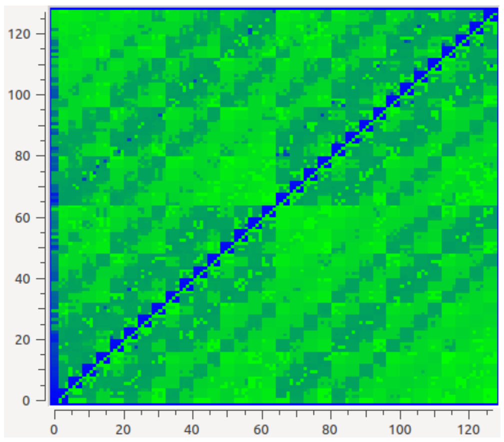

Fig. 4. DBScan. Eps = 100, minPts = 10

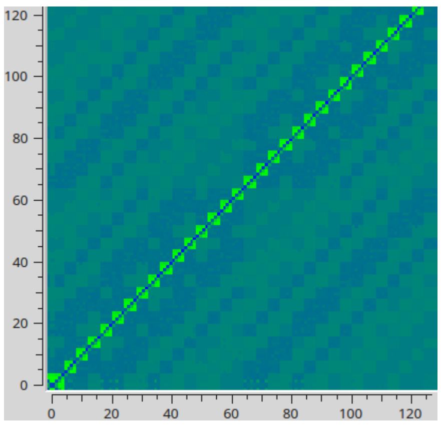

Fig. 5. Divisive, lvlNum = 8.

From the DBScan clustering result, the pattern is repeated every 64 processors.

Lomonosov-1 (128 processors)

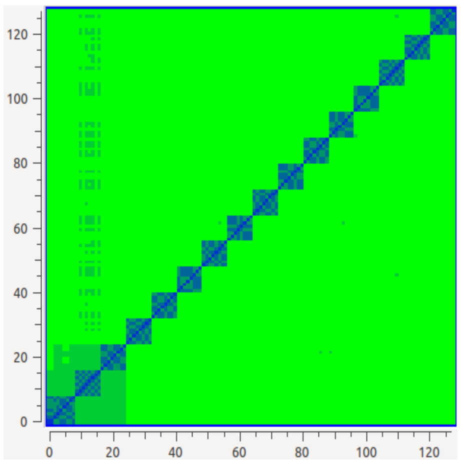

Fig. 6. DBScan. Eps = 1000, minPts = 10.

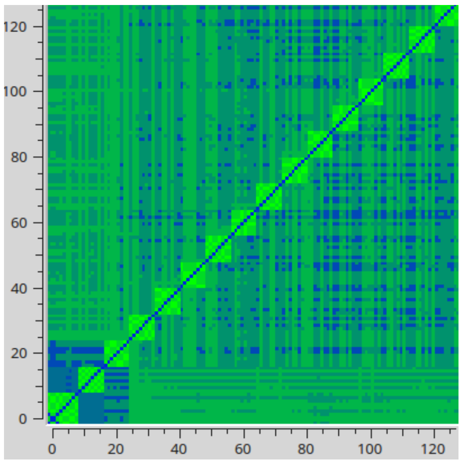

Fig. 7. Divisive, lvlNum = 6.

When visualizing the DBScan clustering result, it is seen that the pairs of processors along diagonal form clusters.

Lomonosov-2 (118 processors)

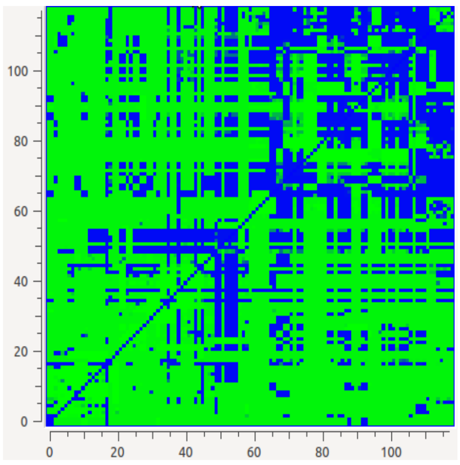

Fig. 8. DBScan. Eps = 1000, minPts = 10.

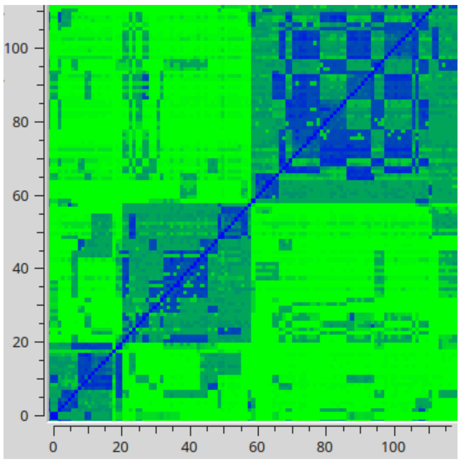

Fig. 9. Divisive, lvlNum = 6.

As can be seen from the image that visualizes the DBScan clustering result, the Lomonosov-2 test results contain a lot of noise.

Juropa (128 processors)

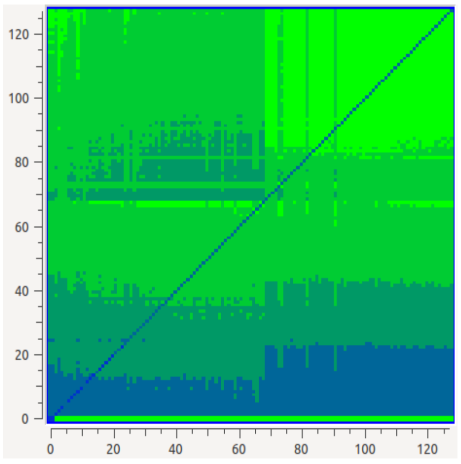

Fig. 10 . DBScan. Eps = 500, minPts = 10.

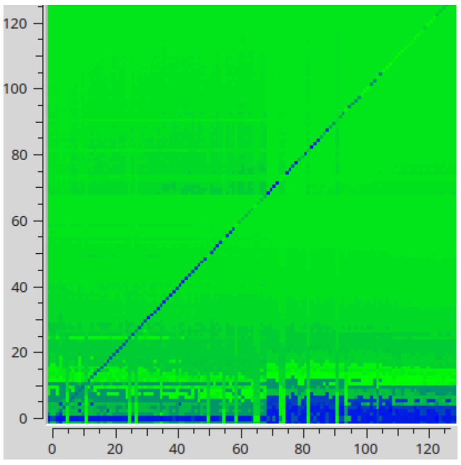

Fig. 11. Divisive, lvlNum = 6.

It is difficult to come up with any conclusion about the topology of the Juropa supercomputer.

Conclusion. Designed applications and clustering algorithms allows to split the set of delays into subsets corresponding to hardware features of supercomputer interconnections. This split illustrates the type of routes in communications. Algorithms of clustering detect processors, which are “close” to each other (the route of message sent between these processors passes the minimal number of network infrastructure components). All “source-receiver” pairs of neighbor processors will be located in one cluster. The delay value of message transmission between processors in one cluster will be minimal.

Completeness of extracted information about clusters depends on data structure and input parameters of algorithms. For benchmark data from Bluegene / P the DBScan algorithm found more clusters, than divisive clustering algorithm, but for Lomonosov-1, Lomonosov-2 and Juropa divisive clustering is more suitable. In such way we see trade-off between these algorithms. To find the most accurate set of clusters using DBScan, parameters eps and minPts should be selected correctly. Optimal number of levels in hierarchical structure of divisive clustering result should be selected carefully, because high or very low number of levels potentially will produce non-informative clusters.

References

1. Brian S. Everitt, Landau S., Leese M., Stahl D., Cluster Analysis, 5th edition // Wiley, 2011, pp. 49-53, 84-88, 220 – 222.

2. Charu C. Aggarwal, Chandan K. Reddy, Data Clustering. Algorithms and Applications // CRC Press, 2014, pp. 5 – 8.

3. A. Peiravi, A fast algorithm for connectivity graph approximation using modified Manhattan distance in dynamic networks // 2008 – Applied Mathematics and Computation, pp. 319-332

4. Salnikov A.N., Andreev D.Yu., Lebedev D.Yu., Toolkit for analyzing the communication environment characteristics of a computational cluster based on MPI standard functions // 2012. – Moscow University Computational Mathematics and Cybernetics, Vol. 36, no. 1, Moscow, publishing house Moscow State University, pp. 41-49, DOI: 10.3103/S0278641912010074.

5. Salnikov A.N., Maysuradze A.I., Andreev D.Yu., Kostin G.A., Klasterizatsiya rezul'tatov testirovaniya kommunikatsionnoy sredy mnogoprotsessornykh sistem: edenicy analiza, issledovaniye metodov, vizualizatsiya rezul'tatov // 2012. – Vestnik of UGATU, Vol. 16, no. 6, pp. 149-157.

6. Ester, Martin; Kriegel, Hans-Peter; Sander, Jörg; Xu, Xiaowei and other. A density-based algorithm for discovering clusters in large spatial databases with noise. // 1996. – Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), AAAI Press. pp. 226–231

7. Clustbench benchmarking suite: https://github.com/clustbench.

Authors:

Maysuradze Archil Iverievich, Lomonosov Moscow State University (Leninskie Gory 1, Moscow, Russian Federation), Associate Professor, Candidate of Science in Physics and Maths

Salnikov Alexey Nikolaevich, Lomonosov Moscow State University (Leninskie Gory 1, Moscow, Russian Federation), Senior Researcher, Candidate of Science in Physics and Maths

Begaev Artur Andreevich, Lomonosov Moscow State University (Leninskie Gory 1, Moscow, Russian Federation)

УДК 004.722, 004.725, 519.254

Анализ структуры задержек в коммутационной среде суперкомпьютера, посредством алгоритмов DBScan и дивизивной кластеризации [*]

А.Н. Сальников[**], А.А. Бегаев, А.И. Майсурадзе

Московский государственный университет имени М. В. Ломоносова, Москва, Российская Федерация

В данной работе предлагается метод оценки и анализа величин задержек в коммуникационной среде вычислительного кластера возникающих при передаче сообщений между парами процессоров «источник-приемник». К данным о задержках, полученным на суперкомпьютерах Bluegene/P, Ломоносов, Ломоносов-2, Juropa применяются 2 метода кластеризации: DBScan и дивизивная (разделяющая) кластеризация. Показано более точное совпадение найденных кластеров методом DBScan к реально существующим особенностям в коммуникационной среде по отношению к методу дивизивной кластеризации. Кластеры объединяются около одинаковых компонентов сетевой инфраструктуры суперкомпьютера. Полученные кластеры были визуализированы в 2-х мерном пространстве специальной утилитой, разработанной авторами.

Ключевые слова: вычислительный кластер, алгоритмы кластеризации, коммуникационная среда, параллельные вычисления

Авторы:

Майсурадзе Арчил Ивериевич, Московский государственный университет имени М. В. Ломоносова (119991, г. Москва, Ленинские горы, д. 1), доцент, кандидат физико-математических наук

Сальников Алексей Николаевич, Московский государственный университет имени М. В. Ломоносова (119991, г. Москва, Ленинские горы, д. 1), ведущий научный сотрудник, кандидат физико-математических наук

Бегаев Артур Андреевич, Московский государственный университет имени М. В. Ломоносова (119991, г. Москва, Ленинские горы, д. 1)

[*] Работа выполнена при частичной поддержке РФФИ (гранты 17-07-01562, 16-01-00196, 17-01-00917) и МГУ им. М.В. Ломоносова (НИР 5.1.16).

[**] E-mail: salnikov@cs.msu.ru.