doi 10.26089/NumMet.v22r314

Preprocessing of system monitoring data for workload analysis of HPC systems

M. I. Martyshov
Lomonosov Moscow State University, Research Computing Center, Moscow, Russia
ORCID: https://orcid.org/0000-0002-7335-1122, *e-mail:* maxitux@icloud.com
D. A. Nikitenko
Lomonosov Moscow State University, Research Computing Center, Moscow, Russia
ORCID: https://orcid.org/0000-0002-2864-7995, *e-mail:* dan@parallel.ru

Abstract: HPC systems are complex in architecture and contain millions of components. To ensure reliable operation and efficient output, functioning of most subsystems should be supervised. This is done on the basis of collected data from various logging and monitoring systems. This means that different data sources are used, and accordingly, data analysis can face multiple issues processing this data. Some of the data subsets can be incorrect due to the malfunctioning of used sensors, monitoring system data aggregation errors, etc. This is why it is crucial to preprocess such monitoring data before analyzing it, taking into the consideration the analysis goals. The aim of this paper is, being based on the MSU HPC Center monitoring data, to propose an approach to data preprocessing of HPC monitoring systems, giving some real life examples of issues that may be faced, and recommendations for further analysis of similar datasets.

Keywords: supercomputing, supercomputers, system monitoring data analysis, system monitoring data cleaning, system monitoring data reduction.

Acknowledgements: The research is carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University. The reported study was funded by Russian Foundation for Basic Research, project No. 20-07-00864, and project No. 21-57-12011 in part of noise characterization.

For citation: M. I. Martyshov and D. A. Nikitenko, "Preprocessing of system monitoring data for workload analysis of HPC systems," Numerical Methods and Programming. **22** (3), 230–238 (2021). doi 10.26089/NumMet.v22r314.

1. Introduction. Supercomputers have always been distinguished as the most complicated and largest computing systems. These machines contain more components, then any other computers. And what is more, the requirements for sustainability of operation and efficient output are set very high, because these systems are extremely expensive and are used for solving frontier applied problems with highest computational complexity.

That is why supercomputers have to be under total monitoring, all the subsystems should be supervised — compute nodes, storage, interconnect, whole software stack, cooling, power, etc.

The variety of subsystems and different levels of criticalness obviously lead to using diverse monitoring and logging systems, which fit specific monitored objects best. These data collectors can greatly vary in reliability, accuracy, data generation rate, and all other possible attributes.

At the same time most of the data analysis tasks focus on specific data and do not use the whole dataset. So, once the data analysis goal is fixed, it becomes needless to take care only of the corresponding data. For example, if the research is performed on the base of logs of resource manager only, there is no need to preprocess other data, corresponding to analysed jobs.

2. Proposed principles of monitoring data preprocessing. The questions and strategies of data preprocessing are widely discussed now with the explosive advance in data mining and big data analytics. Many works, which are still actual, date back to the late 1990x and early 2000x [1–9], when many new opportunities of big data appeared. Many of these works have become classics of data processing.

In this paper we focus on peculiarities of specific data, thus, limiting means of data preprocessing to reduce complexity of data preparation, proving necessity of data preprocessing for this type of data.

Based on performed previous research and some recent experiments, we propose the following general approach for data preprocessing for the HPC monitoring systems. This approach is based on data processing principles, described in [10], one of popular data mining handbooks.

1. The first step is defining the goals of analysis. This step significantly reduces preparatory actions with data.
2. The second step is to review the available data in terms of types of meaning, values, range, mean, median, etc. This is necessary for choosing appropriate methods of analysis to achieve goals set on the first step.
3. Next, being continuation of step 2, it is vital to analyze the suitability of available dataset for the study. At this stage, the questions of data availability, its cleanliness are addressed. For example, if we have two categories of data — resource manager logs and integral job characteristics, the second datatype is much less reliable because of many factors, for example, due to monitoring system malfunction or system noise and interapplication influence.
4. Based on previous steps data cleaning and data reduction are performed.
5. The analysis is performed based on the selected data preprocessed subset to obtain representative and reliable results.

3. Monitoring data preprocessing. In our analysis, the logs and system monitoring data from the Lomonosov-2 supercomputer were used. This is the largest academic installation in Russia and one of the largest in Eastern Europe [11, 12]. The facility provides access for scientific and educational workgroups on the free of charge basis, as well as commercial usage. There's about one thousand job runs per day for such installations with a total diversity of research areas, used software, implemented algorithms, scales of applications, etc [13–15].

Such large-scale systems unavoidably run in automated modes, with all the subsystems being monitored. This data in whole looks redundant, but as soon as we focus on a specific goal, the dataset is accordingly reduced, and sometimes can look even insufficient. For example, data for resilience systems like OctoTron [16], data for user applications efficiency analysis [17, 18], and data for general workflow analysis [19–22] may intersect weakly. Nevertheless, most of the available data is usually stored in an integrated storage, except raw data which is stored in a special write-oriented database.

The questions of data integration, preprocessing and visualization are quite widely discussed nowadays, so we will not stop on that, as the typical operations are usually performed with datasets, and the key question is to select the data, which is suitable for specific analysis.

Another issue which is not addressed often, is whether system monitoring data represents real application or node behavior, or the behavior influenced by other running applications, system noise and other injected events.

There are some related works, for example, presented in [23, 24], but there's still a gap especially in internode issues. This was a motivation for a recently started joint Russian-German ExtraNoise project [25], devoted to deep study of revealing real values from the measured both from theoretical point of view, and from the technical system monitoring data analysis.

Moscow State University contributes in noise characterization (Fig. 1 illustrates noise characterization as one of preliminary project results) and methods of capturing noise and corrupted behavior by means of system monitoring, Juelich Supercomputer Center and Technical University of Darmstadt focus on noise modelling. All the project members contribute in real life experiments, using available high-grade supercomputer facilities.

What is important in respect to the topic of this paper, is that the nature of noise is very different, so many appropriate types of system monitoring data should be treated carefully.

4. Dataset analysis. The analyzed dataset contains various attributes for 403 095 jobs, with both resource manager data and integral job characteristics. Thus, data may be dirty. Fig. 2 is the visualisation of the data's availability. Each row corresponds to a unique job. Each column represents specific characteristics for the job, where white spaces are missing data, and the black spaces are the available data.

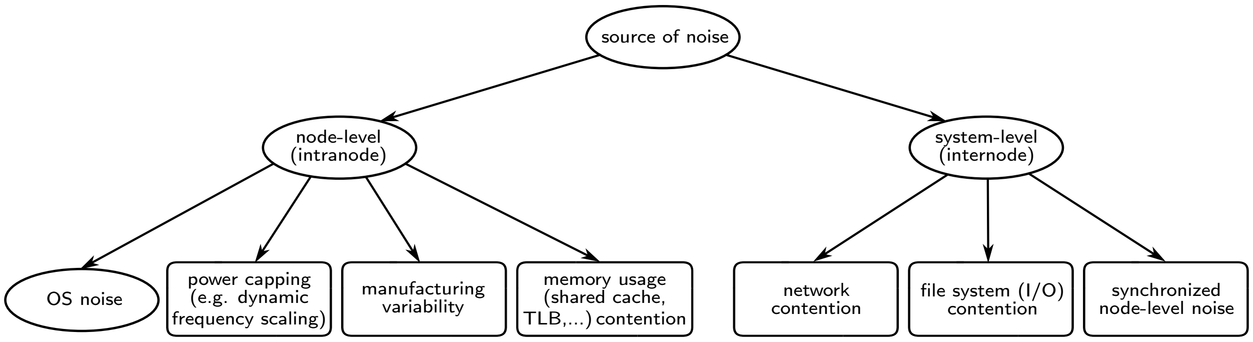
source of noise
node-level (intranode)
system-level (internode)
OS noise
power capping (e.g. dynamic frequency scaling)
manufacturing variability
memory usage (shared cache, TLB,...) contention
network contention
file system (I/O) contention
synchronized node-level noise

Figure. 1. Preliminary noise characterization by its source

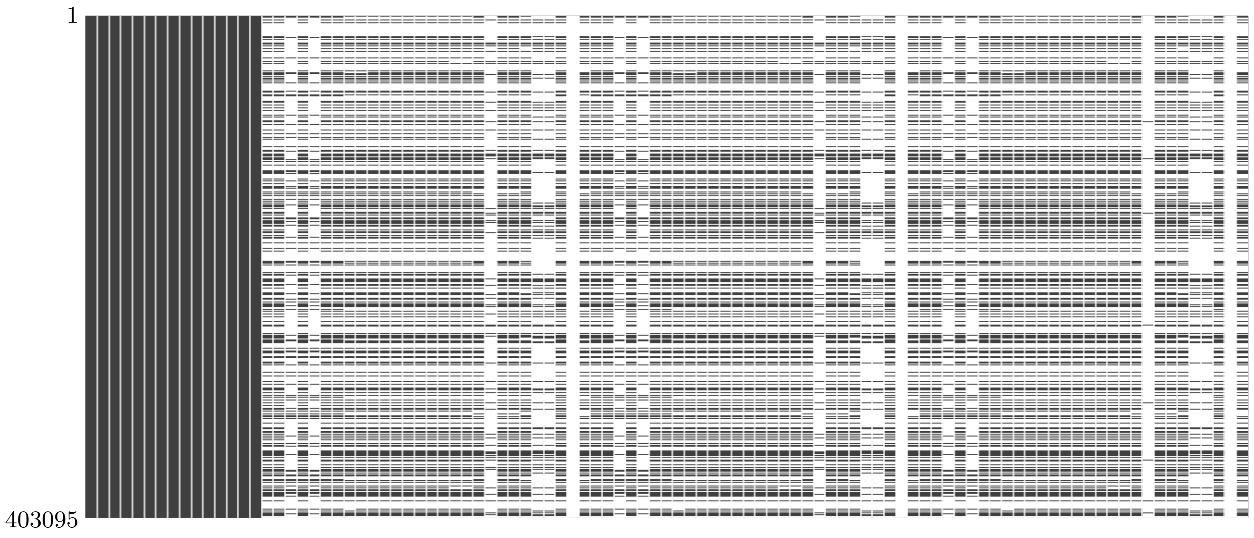

Figure 2. Data availability. White space represents unavailable values in the dataset.
Triple repeats correspond to min-avg-max values of same attributes

After that, the team decided to look at the overall statistics on the data. It helped to reduce artefacts — the records, which can significantly affect statistics. For example, any negative value of load average, or missing job finish timestamp.

Table 1 shows the table with data numerical description (where `num_cores` is number of cores, `num_nodes` is number of nodes, `t_end` (all times are in UNIX time) is the time when the job ended, `t_start` is the time when the job was started, and `t_submit` is the time when the job was submitted, `avg_lustre_read_bytes` is average reading speed of network interface (bytes/s), `avg_lustre_write_bytes` is average writing speed of

Table 1. Data numerical description

	count	mean	std	min	median	max
`num_cores`	418726.0	104.1455	133.3526	1.0	70.0	15568.0
`num_nodes`	418726.0	1263254,00	9.5365	1.0	5.0	1112.0
`t_end`	418726.0	1573227359.2249	26325969.0876	1536324379.0	1569378627.5	1618922701.0
`t_start`	418726.0	1573210957.0998	26324670.1768	1536324359.0	1569356519.0	1618922320.0
`t_submit`	418726.0	1573191271.2930	26314736.7207	1536309795.0	1569317464.5	1618922309.0
∼`avg_lustre_read_bytes`	72998.0	10045847.8817	120220950.0377	0.0	2643.3034	17736330193.6312
∼`avg_lustre_write_bytes`	72997.0	2513664.1481	33326411.7866	0.0	9795.1668	3927772505.3968
∼`avg_loadavg`	145776.0	12.6878	0.1099	0.0	12.7414	1819.0197

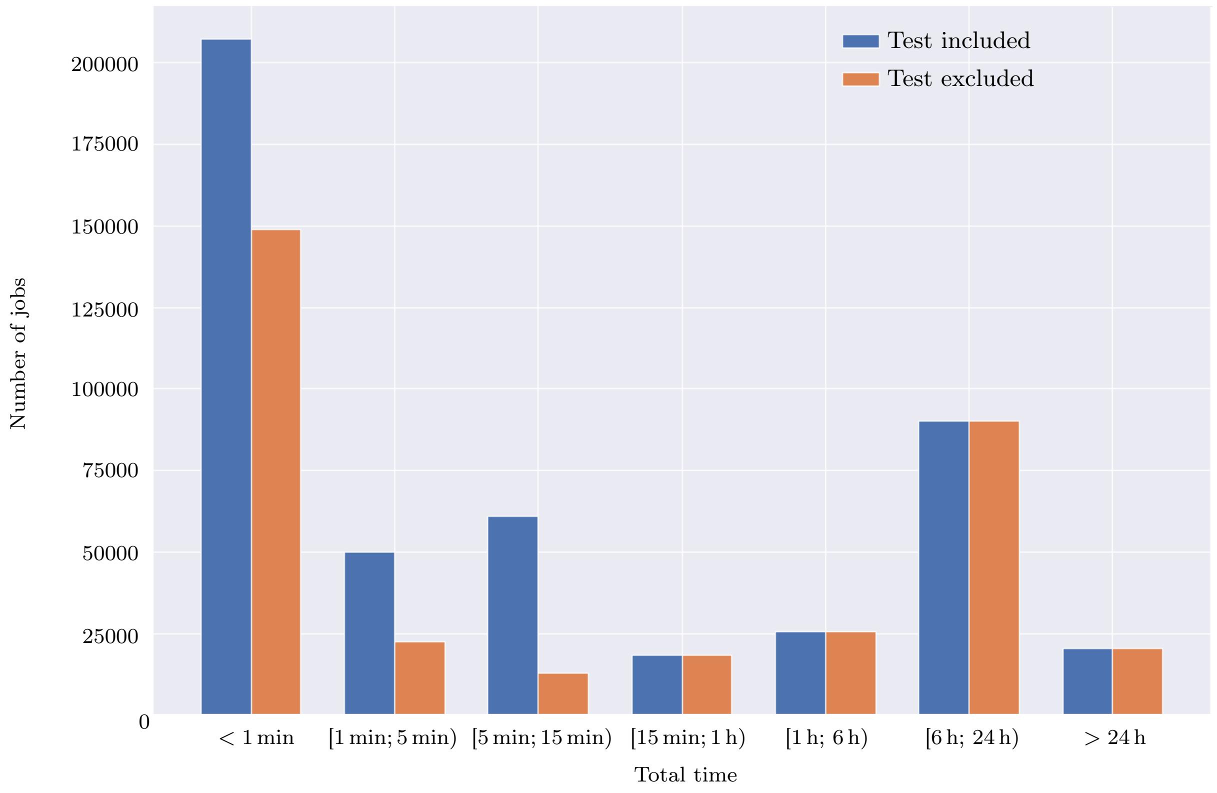

Figure 3. Number of jobs in each time cluster with and without *test* queue. The *test* queue time limit is set to 15 minutes

network interface (bytes/s), `avg_loadavg` is average number of parallel processes) which was gathered using Python *Pandas* library (method *describe*) [26]. All the applications are usually run in the queues, which usually correspond to specific compute node clusters, equipped with specific accelerators, RAM, HDDs and so on. At the same time there are usually available test and service queues with different limitations on job size and length. In this, in order to get authentic results on the data the team reached the conclusion that the test queue in the partition column should be excluded because this queue is usually used for short, small jobs or job debugging.

Fig. 3 is shown to demonstrate how test queue jobs distract from the actual result. It shows the number of jobs in each total time of the job cluster with the test queue included and excluded.

Next, another attempt to visualize the data was made. Fig. 4 demonstrates plotted data with $\log_2$ total time of the job in seconds on the horizontal axis and $\log_2$ of number of nodes requested for the job on the vertical axis. The different colours that stand for the state with which the job was finished. The distribution of colors and dots density depicts that data can be clustered, what is shown in the section 5.

Despite the traditional methods of data cleaning the team recommends checking the data on correct values. The integral characteristics could often be incorrect due to the sensors' failure. Another example that perfectly illustrates how incorrect data could alter the results is when the missing data may be automatically changed from `null` to `NaN [Not a Number]` or `0` (may happen when the data is loaded in, for example, *Pandas DataFrame*). In this case the alteration in the results is guaranteed. Fig. 5 has 8 missing bars in the middle.

These missing node clusters was an unexpected result for the team, so it was agreed to look further. As it turned out, on this particular amount of nodes sensors were not working, thus the data values were supposed to be filled with `null`. However, when these values were printed, they were `0` instead of `null`. Because of that the average read and write speeds were also 0, and that is why there are no bars in the middle of the plot.

5. Monitoring data analysis examples. After the data was preprocessed the team started to analyze it in order to draw conclusions. Figures below show several examples of data visualization and conclusions made from them.

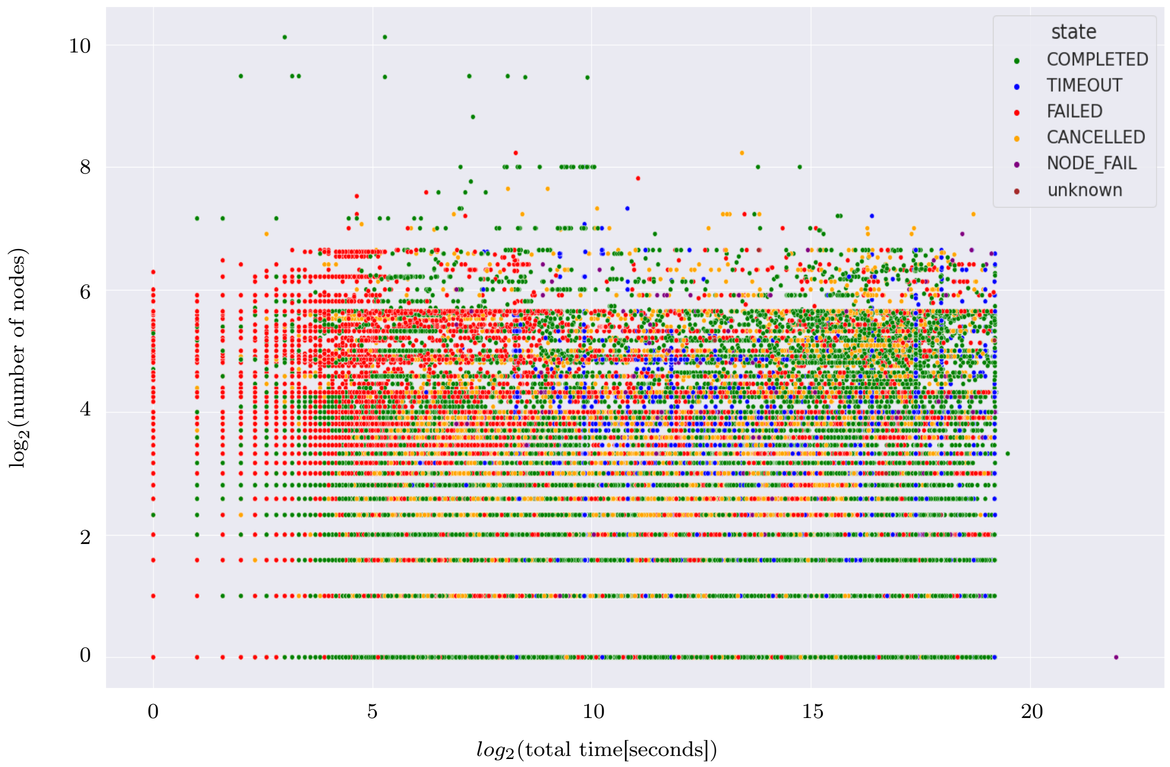

Figure 4. Total time of the job in seconds (horizontal axis) and the number of nodes requested for the job (vertical axis). Colours stand for the job states. For example, it demonstrates the majority of FAILED state on the early stages of application runs

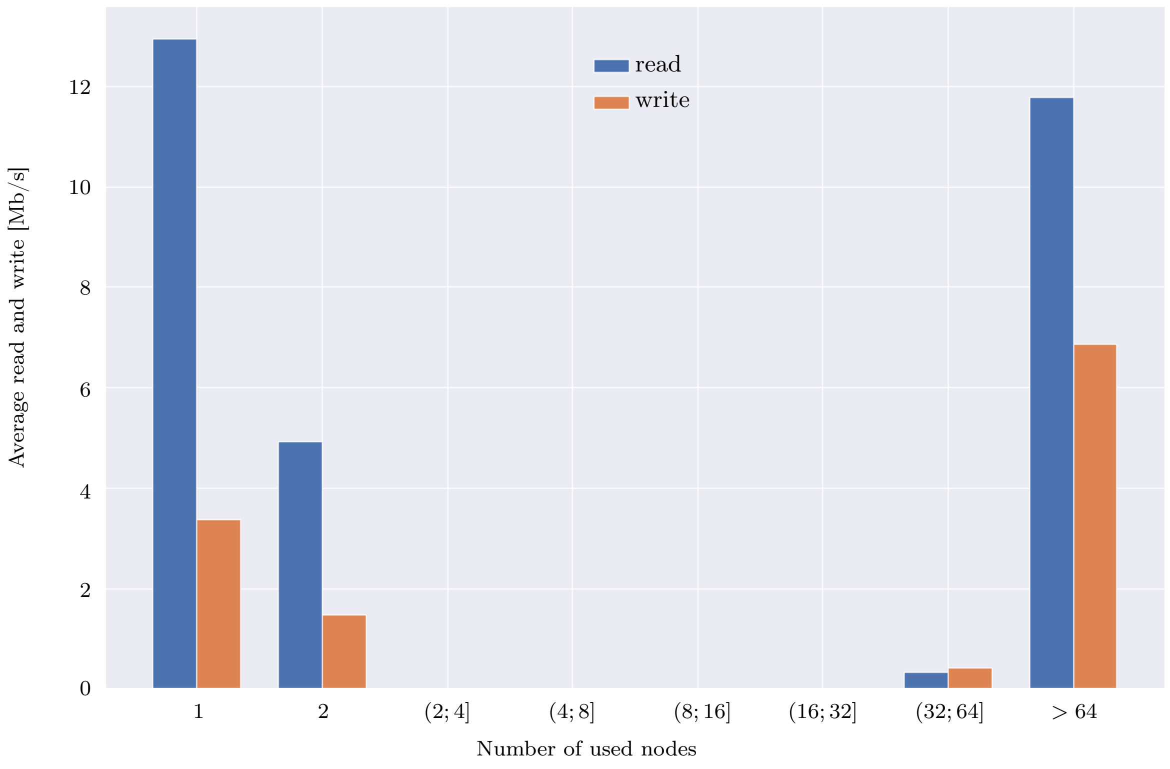

Figure 5. Missing data example. Null values have been stored for 3-32 node runs instead of actual monitoring data

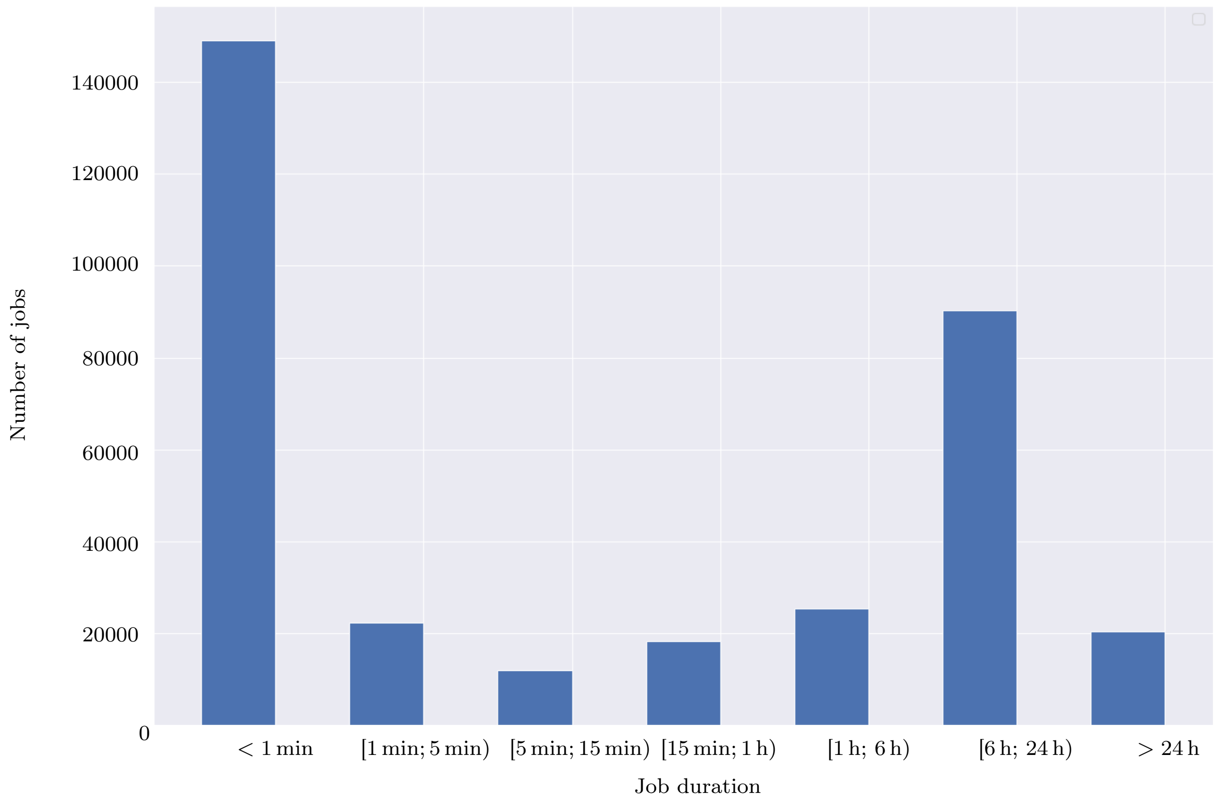

Figure 6. The proposed time clusters and the number of jobs in each time cluster

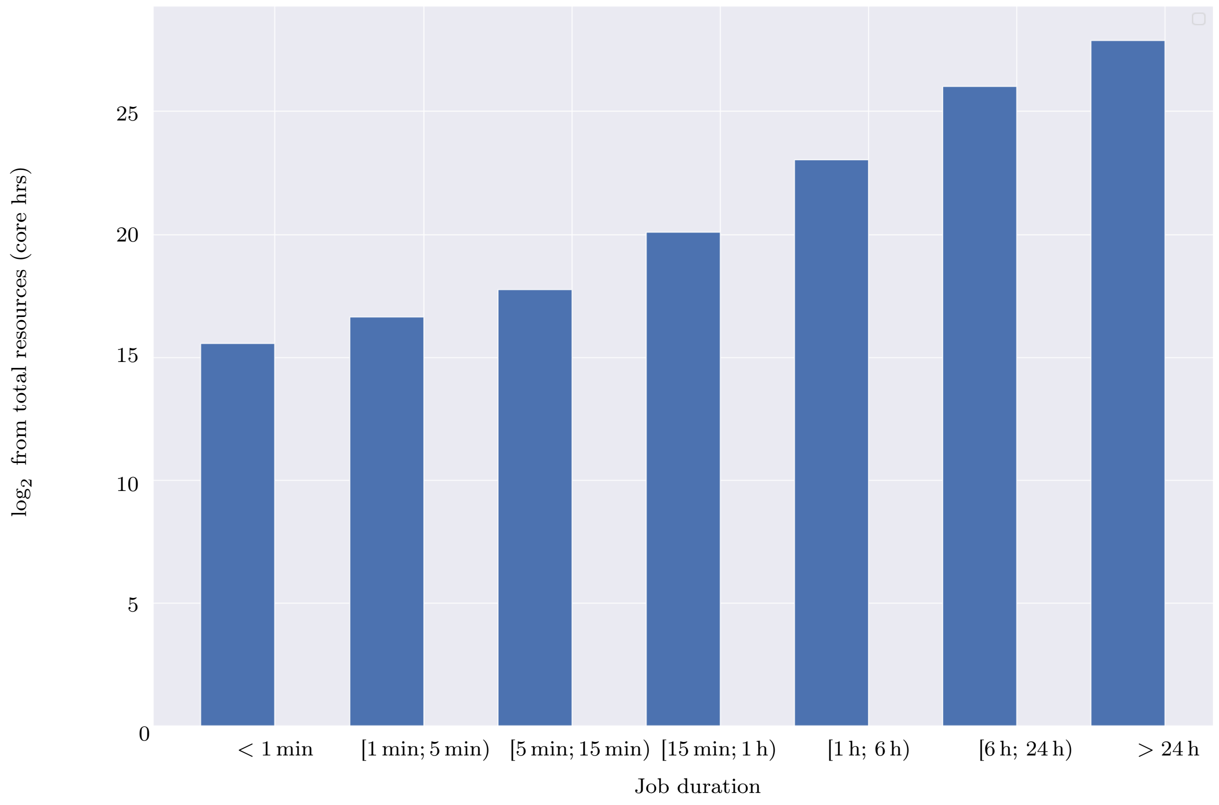

Figure 7. The amount of total utilized resources (corehours) in each proposed time cluster.

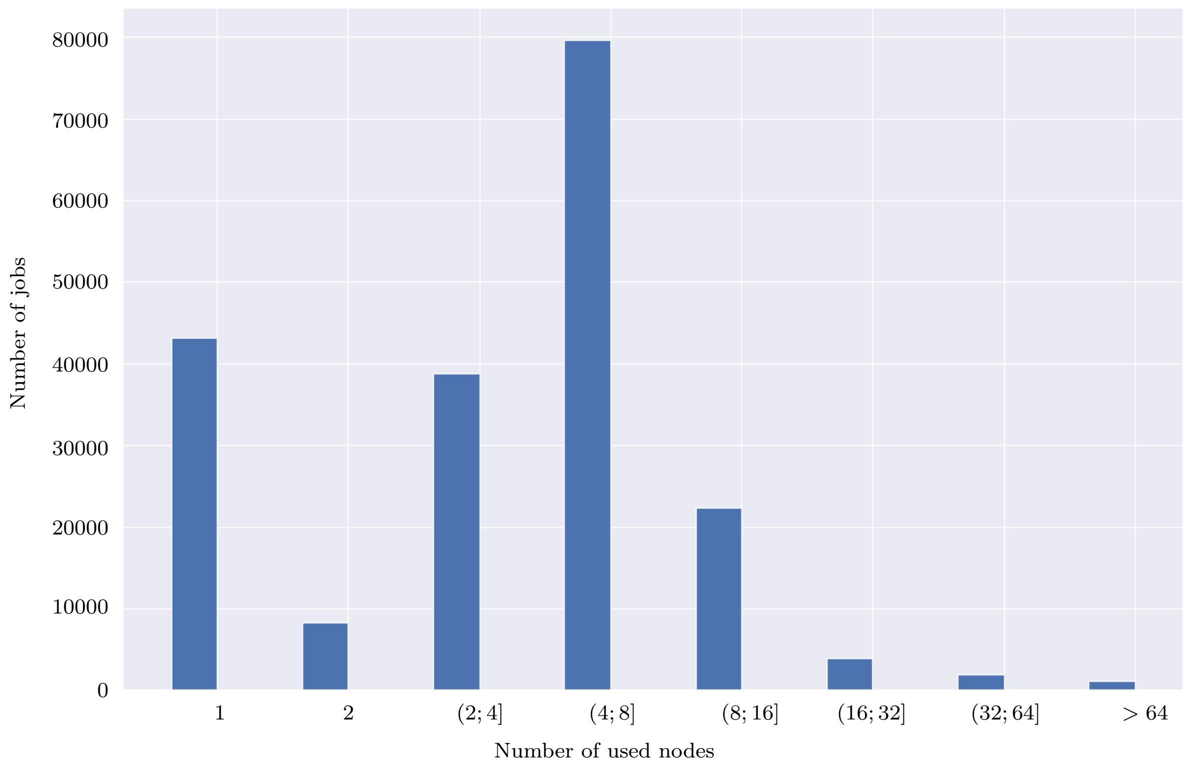

Fig. 8. Number of jobs in each node cluster. It demonstrates the users request about 8 nodes for their jobs the most

The first example is the clustering of total time of the jobs. The team decided to pick 7 groups of time that would allow seeing how many "big" and "small" jobs there are. The groups are jobs that are less than a minute, from 1 minute to 5 minutes, form 5 minutes to 15 minutes, from 15 minutes to 1 hour, from 1 hour to 6 hours, from 6 hours to 24 hours, and more than 24 hours long.

Fig. 6 shows the number of jobs in each category(time clusters are on the horizontal axis, number of jobs is on the vertical axis). The plot demonstrates that the biggest group is the first one, which is small jobs. However, these jobs do not require much resources, thus, the proposed grouping may be correct. This clustering together with finding missing data for specific job sizes, can be named the main contribution of this work. It is often proposed to use other clusters based on job duration, for example — 15min and less, 15min to 1h, 1h to 24h, 1day and above. The idea of authors was to propose clusters with comparable number of elements, but having dominant clusters with minimal and maximal resource utilization. In this, clusters for less than 1 minute and heavy 6h–24h were introduced.

In order to find out whether the proposed clusters are correct the team decided to visualize the amount of total resources in each time cluster (Fig. 7). The amount of resources is gradually increasing with the increase in time. This proves that the clustering is good.

The second example is the clustering by the number of requested nodes for the job. The team proposed 8 clusters: 1 node, 2 nodes, from 3 to 4 nodes, from 5 to 8 nodes, from 9 to 16 nodes, from 17 to 32 nodes, from 33 to 64 nodes, and greater than 64 nodes. The plot shown in Fig. 8 shows the number of jobs (vert. axis) in each node cluster (horiz. axis). It was inferred from the plot that users request 8 nodes for their jobs the most.

6. Conclusions and future work. In this paper we discussed the problem of monitoring data preprocessing for large HPC systems. We proposed to pay additional attention to the quality and peculiarities of analyzed data on the stage of analysis goals definition, because the whole dataset is built of multiple subsets with diverse availability, accuracy and reliability.

Based on the dataset of over 400 000 records corresponding to the MSU HPC center, data analysis has been performed. It was shown that the data read rate is higher, then the write rate, that proves that HPC applications in whole are more data mining, then data generating. Data availability was shown, clusters for job duration and job size have been proposed.

Although data preprocessing on the initial stage is done by hand, automatic data cleaners continue to evolve and grow, fueled by the desire for time conservation and operational efficiency [27].

References

1. L. P. English, *Improving Data Warehouse and Business Information Quality: Methods for Reducing Costs and Increasing Profits* (Wiley, New York, 1999).
2. D. Pyle, *Data Preparation for Data Mining* (Morgan Kaufmann, San Francisco, 1999).
3. D. Loshin, *Enterprise Knowledge Management: The Data Quality Approach* (Morgan Kaufmann, San Francisco, 2001).
4. T. C. Redman, *Data Quality: The Field Guide* (Digital Press, Boston, 2001).
5. T. Dasu and T. Johnson, *Exploratory Data Mining and Data Cleaning* (Wiley, Hoboken, 2003).
6. R. Y. Wang, V. C. Storey, and C. P. Firth, "A Framework for Analysis of Data Quality Research," IEEE Trans. Knowl. Data Eng. **7** (4), 623–640 (1995).
7. Y. Wand and R. Y Wang, "Anchoring Data Quality Dimensions in Ontological Foundations," Commun. ACM. **39** (11), 86–95, 1996.
8. D. P. Ballou and G. K. Tayi, "Enhancing Data Quality in Data Warehouse Environments," Commun. ACM. **42** (1), 73–78 (1999).
9. J. E. Olson, *Data Quality: The Accuracy Dimension* (Morgan Kaufmann, San Francisco, 2003).
10. J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques* (Morgan Kaufmann, San Francisco, 2012).
11. The Top500 list of the world's most powerful supercomputers. https://www.top500.org/. Cited September 26, 2021.
12. The Top50 list of supercomputers in the Russian Federation. http://top50.supercomputers.ru/newsfeed. Cited September 26, 2021.
13. MSU HPC Center. https://parallel.ru/cluster. Cited September 26, 2021.
14. V. V. Voevodin, A. S. Antonov, D. A. Nikitenko, et al., "Supercomputer Lomonosov-2: Large Scale, Deep Monitoring and Fine Analytics for the User Community," Supercomput. Front. Innov. **6** (2), 4–11 (2019).
15. V. Voevodin, A. Antonov, D. Nikitenko, et al., "Lomonosov-2: Petascale Supercomputing at Lomonosov Moscow State University," in *Contemporary High Performance Computing: from Petascale toward Exascale* (CRC Press, Boca Raton, 2019), Vol. 3, pp. 305–330.
16. S. I. Sobolev, A. S. Antonov, P. A. Shvets, et al., "Evaluation of the Octotron System on the Lomonosov-2 Supercomputer," in *Proc. Int. Conf. on Parallel Computational Technologies, Rostov-on-Don, Russia, April 2–6, 2018* (South Ural State Univ., Chelyabinsk, 2018), pp. 176–184.
17. A. V. Adinets, P. A. Bryzgalov, Vad. V. Voevodin, et al., "Job Digest: An Approach to Dynamic Analysis of Job Characteristics on Supercomputers," Numerical Methods and Programming **13**, 160–166 (2012).
18. D. A. Nikitenko, K. S. Stefanov, S. A. Zhumatiy, et al., "System Monitoring-Based Holistic Resource Utilization Analysis for Every User of a Large HPC Center," in *Lecture Notes in Computer Science* (Springer, Cham, 2016), Vol. 10049, pp. 305–318.
19. D. Nikitenko, S. Zhumatiy, and P, Shvets, "Making Large-Scale Systems Observable — Another Inescapable Step towards Exascale," Supercomput. Front. Innov. **3** (2), 72–79 (2016).
20. P. Shvets, Vad. Voevodin, and D. Nikitenko, "Approach to Workload Analysis of Large HPC Centers," in *Parallel Computational Technologies* (Springer, Cham, 2020), Vol. 1263, pp. 16–30.
21. D. A. Nikitenko, Vad. V. Voevodin, and S. A. Zhumatiy, "Deep Analysis of Job State Statistics on Lomonosov-2 Supercomputer," Supercomput. Front. Innov. **5** (2), 4–10 (2018).
22. S. I. Sobolev, Vl. V. Voevodin, A. S. Antonov, et al., "Making Supercomputers Smart: the Moscow State University Experience," in *Proc. 27th Int. Symp. on Nuclear Electronics and Computing (NEC 2019), Budva, Becici, Montenegro, September 30–October 4, 2019* CEUR Workshop Proc. **2507**, 1–6 (2019).
23. A. Shah, M. Müller, and F. Wolf, "Estimating the Impact of External Interference on Application Performance," in *Lecture Notes in Computer Science* (Springer, Cham, 2018), Vol. 11014, pp. 46–58.
24. T. Hoefler, T. Schneider, and A. Lumsdaine, "Characterizing the Influence of System Noise on Large-Scale Applications by Simulation," in *Proc. ACM/IEEE Conf. on Supercomputing (SC 2010), New Orlean, USA, November 13–19, 2010* (IEEE Press, Washington, DC, 2010), doi 10.1109/SC.2010.12
25. D. A. Nikitenko, F. Wolf, B. Mohr, et al., "Influence of Noisy Environments on Behavior of HPC Applications," Lobachevskii J. Math. **42** (8), 1560–1570 (2021).

26. pandas: Python Data Analysis Library. *https://pandas.pydata.org/*. Cited September 26, 2021.
27. M. Chien and A. Jain, "Gartner Magic Quadrant for Data Quality Solutions," *https://www.gartner.com/en/documents/3988016/magic-quadrant-for-data-quality-solutions*. Cited September 26, 2021.

Received
August 22, 2021

Accepted for publication
September 11, 2021

Information about the authors

Maxim I. Martyshov — Intern, Lomonosov Moscow State University, Research Computing Center, Leninskie Gory, 1, Building 4, 119991, Moscow, Russia.

Dmitry A. Nikitenko — PhD., Senior Scientist, Lomonosov Moscow State University, Research Computing Center, Leninskie Gory, 1, Building 4, 119991, Moscow, Russia.