DOI: 10.14529/jsfi230304

Recurrent Monitoring of Supercomputer Noise

Vadim V. Voevodin[1], *Dmitry A. Nikitenko*[1]

The presence of noise in supercomputers has long been known, but its scale and impact on the behavior of user applications are nevertheless considerably unclear. Therefore, we decided to develop an approach to determining the noise level on an ongoing basis, which makes it possible to assess the global impact of noise over a long time period. This paper describes a method for recurrent monitoring the noise level and analyzing collected statistics on a real modern supercomputer, and also presents the implementation and evaluation of this method on the Lomonosov-2 supercomputer. The usage of the proposed approach in practice made it possible to identify previously unknown issues and peculiarities, like detection of a faulty compute node, presence of nodes that tend to be more susceptible to noise or the global nature of the noise, which leads to the appearance of noise at multiple nodes simultaneously. This method can as well be ported to other similar computing systems without significant changes.

Keywords: supercomputing, high-performance computing, monitoring, noise, noise measurement, noise level.

Introduction

Modern supercomputers are very complex objects that consist of tens, hundreds of thousands, or even millions of components. They also include a large number of compute nodes; for example, the most powerful supercomputer in the world called Frontier [4] (#1 from the Top500 [6] list for November 2023) has more than 9000 nodes, and the Lomonosov-2 supercomputer [25] (#6 in the Top50 [3] list for March 2023) has about 1700 nodes [2]. To ensure their correct, consistent and efficient operation within a supercomputer, various system software is required (resource manager, monitoring system, distributed file system, operating system (OS), etc.). In addition, many different user applications are run simultaneously on a supercomputer, and in some systems several applications can be launched in parallel on a single node. All this leads to the fact that almost all modern supercomputers suffer from the so-called "noise". Hereinafter, we will define "noise" as the influence of the software and hardware environment, which leads to a change (most often a slowdown) in the execution time or other properties of applications running on a supercomputer.

The causes of noise can vary: changes in hardware (for example, the occurrence of ECC errors), contention for shared resources with other user applications (for example, for a shared communication network or file system), changes in operating conditions (for example, underclocking the processor due to overheating). One of the most common causes of noise is the operating system.

Although the presence of noise in supercomputers has long been known, the extent of its impact on the behavior of user applications is often unknown, especially given that this impact can differ dramatically on different computing systems. Therefore, we decided to develop an approach to determining noise level on a supercomputer, not as a one-time study, but on an ongoing basis.

The main contribution of this work is the development of a method for recurrent monitoring the noise level and analyzing the collected statistics on a real modern supercomputer, which

[1]Lomonosov Moscow State University, Moscow, Russian Federation

allows evaluating the long-term impact of noise in practice. We have implemented and evaluated this method on the Lomonosov-2 supercomputer, but it can be ported to other similar computing systems without significant changes.

The rest of the paper is organized as follows. Section 1 shows existing work aimed at studying noise level on supercomputers. Section 2 is devoted to the description of the proposed approach for continuous monitoring of noise level and its implementation on the Lomonosov-2 supercomputer. Section 3 shows the results of evaluating the approach in practice. In conclusion, the main results of this work are briefly described.

1. Related Work

There is quite a lot of research devoted to the study of noise that occurs in high-performance systems. For example, in the papers [7, 8, 13, 14, 16, 18] the influence of noise on the behavior of supercomputer applications is studied. Other articles (e.g. [9, 10, 12, 17, 19, 24]) explore what causes noise and how one can tackle it. Most of these works are devoted to the study of one specific type of noise: OS noise, which is the most common topic of study in existing research. We can also highlight works [11, 16], which propose models and simulators to predict the scalability of applications, taking noise into consideration. However, these works are mainly aimed at a one-time study of noise and not on studying the noise level on a supercomputer on an ongoing basis.

The work [15] should be mentioned separately, which presents the open-source software tool called Netgauge. This tool allows conveniently measuring OS noise on a machine, mainly on a single server or compute node. Let us briefly describe the operation of Netgauge as it is used in this work. Netgauge runs a larger number of simple identical iterations and calculates what percentage of iterations take significantly longer to execute than the reference iteration. The execution time of a reference iteration is calculated at startup, for which it runs a small number of iterations and determines the minimum time execution value (which must remain minimal for at least 100 iterations in a row). By default, a “noisy” iteration is considered to be an iteration which execution time is at least 9 times higher than the reference one, but this parameter can be changed if desired.

2. Monitoring Noise Level

2.1. Proposed Approach

As the result of our studies under the ExtraNoise project [20], it is proposed to recurrently measure the noise level on a supercomputer in a following way.

On a supercomputer, after each user job completes, a script is run in Slurm epilogue that updates noise level information on the nodes used to run the job itself. Independently for each of these nodes, the script performs the following:

- Checks when the noise level on this node was last measured. To do this, each node has its own empty file, the modification time of which is changed (by the `touch` command) with each new measurement of the noise level.
- If less than a day has passed since the last measurement of the noise level at this node (the threshold can be easily changed), then nothing else is done.

- If more than a day has passed, the script launches the Netgauge software to determine the noise level on this node. The result of Netgauge is saved in a separate file for each node.

Netgauge runs using MPI on all available logical cores (if HyperThreading is enabled on the node, then the number of logical cores is 2 times the number of physical ones). The binding of MPI processes to cores is also used. This allows obtaining accurate and stable results when assessing noise level.

Also, the running time of Netgauge is selected in such a way that the epilogue script runs for no more than 1 minute on average. To do this, a heuristically selected value of a parameter that indicates the number of "noisy" iterations that Netgauge needs to determine before completing its work was specified. Note that the running time of Netgauge itself is not explicitly limited, since it works until it collects the required number of "noisy" iterations, which is non-deterministic and therefore can take varying amounts of time. However, taking into account the experiments carried out to measure the running time, it was decided that there is no need to further reduce the selected value for the number of "noisy" iterations to detect, especially since in this case the stability of the collected noise level decreases.

2.2. Implementation of Proposed Approach

The described approach was implemented on a petaflop-scale supercomputer Lomonosov-2.

The first step was to determine that the proposed solution would not notably slow down the execution of user job flow. Figure 1 shows the running time of 29 thousand script launches (data was collected from November 15 to December 5 2023 and sorted by increasing operating time). Let us remind that each launch is performed on one node and no more than once a day. It can be seen that in the absolute majority of cases (99.88%) the operating time did not exceed 60 seconds, and the maximum was 131 seconds. In 2/3 of the cases, the script worked instantly, since a new recalculation of the noise level was not required, because less than a day has passed since the last measurement.

Taking into account the information above, we can say that the impact of this solution on the execution of user applications is insignificant. We also note that the script for measuring the noise level is never launched while user applications are running, but only after they are completed. Therefore, the operation of the script only affects by the fact that it slightly delays the launch time of new jobs.

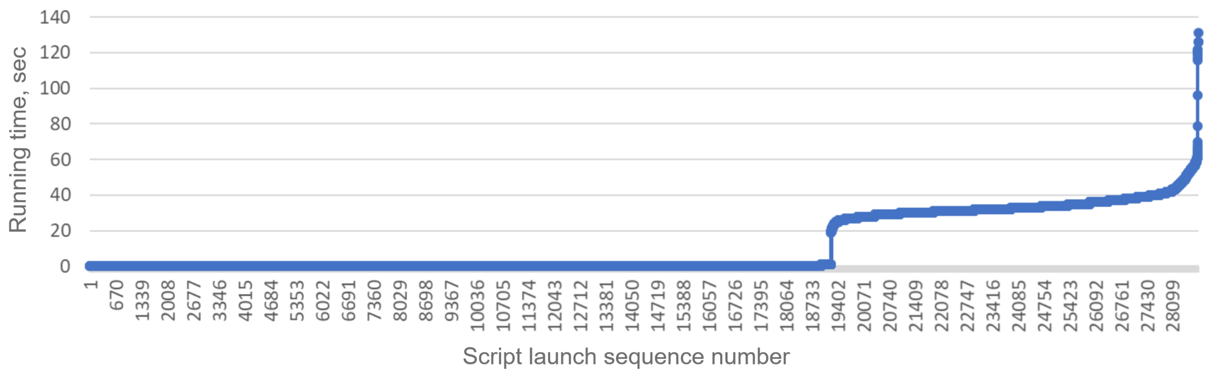


Figure 1. Distribution of running time of the script for noise level measurement

After the collection of data on the noise level was implemented, it was necessary to implement aggregation, storage and analysis of the obtained data. The Lomonosov-2 supercomputer runs the TASC [22] system, which is aimed at a comprehensive analysis of the quality of the supercomputer functioning. Among other things, this system implements a convenient mechanism for saving and visualizing data on the supercomputer operation, which we decided to use.

For these purposes, a script was created that once a day copies the noise level measurements to a separate server, parses data, collects the necessary information (noise level, time it was obtained, node ID) and saves it in MongoDB used in TASC. To visualize the results obtained, a web service based on Redash [5] is used, which allows connecting to different types of databases in a simple and convenient way, request data from there and visualize them using various graphs and tables. In addition, the display of noise level results was also implemented as part of the JobDigest [21] reporting system, which allows not only administrators, but also users of the supercomputer to obtain information about the noise level on the nodes which were used in their applications.

3. Evaluation of Proposed Methods

This section analyzes the results of evaluating the proposed approach on the Lomonosov-2 supercomputer. The data was collected December 1 through December 7 from all supercomputer nodes on which user jobs were launched. Figure 2 shows the average noise level at different nodes over this period, sorted in descending order of noise. The X axis shows the nodes, the left Y axis (blue area on the graph) shows the noise level, and the right Y axis (red dots) shows the number of noise level measurements on the corresponding node. The noise level can range from 0% (no noise) to 100% (all measurements using Netgauge recorded the presence of noise). Hereinafter, all graphs are constructed using Redash and are accessible using designated web-site.

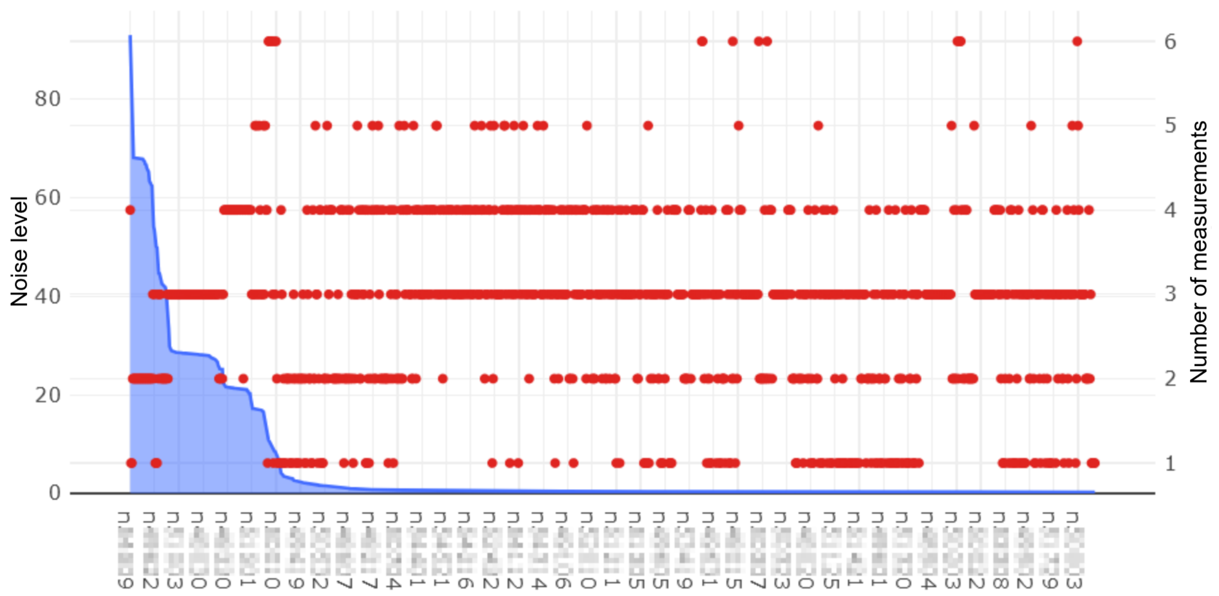


Figure 2. Average noise level at different nodes of a supercomputer. Blue area shows the average noise value for nodes, red dots show the number of noise level measurements on the basis of which the average was calculated

The following conclusions can be drawn from this graph:

- There is one node, the leftmost one on the graph, where the average noise level is significantly higher than the others and overall is very high – 93% (while the next one is 44%). The noise level at this node was collected 4 times during selected time period, i.e. this behavior is not a random anomaly, but is observed regularly. This node will further be considered in more detail.
- The number of noise level measurements at different nodes can vary significantly – from 1 to 6 measurements per node. In general, this is expected, since the frequency of measurements is influenced by many factors:
  - Execution time for user jobs: the longer each job takes to complete (assuming it runs for more than a day), the less often the measurement is performed.
  - Sometimes nodes require repair or reconfiguration, and in these cases they are made unavailable for running user jobs. During this period, noise level assessment is not performed on them.
  - The frequency is also affected by how often the node is idle waiting for jobs. Typically, a node is idle because more nodes are needed to run the next job in the queue than are currently available, so it has to wait for other nodes to become available. However, note that on Lomonosov-2 there are usually very few nodes being idle, due to the constant high load of the supercomputer and the usage of scheduling algorithms in Slurm such as Backfill [1].
  - There are 7 different partitions on the supercomputer, and some of them are small and specialized, where access is given to a limited number of users to carry out special calculations. The load on these partitions is therefore unstable, and during the time period examined, several nodes from these partitions were rarely occupied by user jobs.
- Most often, 2–4 noise level measurements were performed at the nodes during the period under consideration, which corresponds to the expected values.
- 77% of nodes have an average noise level of less than 1%, and such a low noise level can generally be neglected.
- 15% of nodes have an average noise level of more than 10%, and this is quite noticeable. This will also be further discussed.

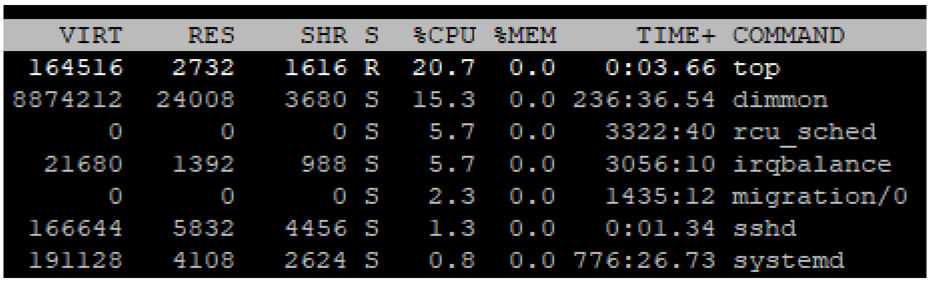
Let us now consider in more detail the most “noisy” node, which was discussed in the first item. First, let us study the chronology of noise level measurements on it during the considered time period December 1 through December 7 (Tab. 1).

Table 1. The noise level on the node under consideration

Date	Dec 01	Dec 02	Dec 04	Dec 05
Noise, %	92.72	92.76	92.70	93.86

During the considered 7 days, noise was measured 4 times. It can be seen that the noise level at this node is always significant: at least 92% of Netgauge iterations signaled the presence of noise, which is a very high value. In order to further understand the causes of such noise, the running processes on the node were studied. In particular, a typical example of the `top` command output is shown in Fig. 3. Here you can see that the behavior of the node indicates some abnormal situation, since processes that normally almost do not load the processors (top, rcu_sched, irqbalance, as well as the DiMMon monitoring system [23] used on Lomonosov-2) in this case occupy a noticeable part of the processor time. And this behavior generally persists over time; in particular, restarting the monitoring system and rebooting the node did not change the overall situation.

Thanks to this information received, the supercomputer administrators paid attention to this node and began to study its behavior. The root causes are not yet completely clear, but the node turned out to be faulty and requires repair.



```
   VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 164516   2732   1616 R  20.7  0.0   0:03.66 top
8874212  24008   3680 S  15.3  0.0 236:36.54 dimmon
      0      0      0 S   5.7  0.0   3322:40 rcu_sched
  21680   1392    988 S   5.7  0.0   3056:10 irqbalance
      0      0      0 S   2.3  0.0   1435:12 migration/0
 166644   5832   4456 S   1.3  0.0   0:01.34 sshd
 191128   4108   2624 S   0.8  0.0 776:26.73 systemd
```

Figure 3. Output of top command on a “noisy” node

Finally, we will study in more detail the nature of the resulting noise level. Let us consider the same time period, but we will analyze not the average values for all nodes (as in Fig. 2), but the last received values for each node, sorted by the time the result was received (Fig. 4). Each value on the X axis corresponds to the last value of the noise level at a certain node. It can be seen that most often high noise levels are grouped by time, i.e. most likely some global processes, which lead to an increase of noise on several nodes at once, are taking place on the supercomputer. This may be due to the peculiarities of the system software used across the entire supercomputer (such as a resource manager or monitoring system) or the shared resources (such as a distributed file system). We are currently studying the reasons for the occurrence of such grouped noise.

You can also see that in this case there are only 4% of nodes with a noise level of more than 10%, which is noticeably lower than 15%, as was the case in Fig. 2. Further, the noise level is most often very low (less than a couple of percent), in other cases it is almost always quite high (40% and higher). This is also confirmed when considering the chronology of noise level measurements for individual nodes (similar to table 1).

Thus, the following conclusions can be drawn:

1. the noise level can be unstable and vary greatly over time;
2. on a compute node, there is normally almost no noise, but it occasionally becomes significant, and there are practically no intermediate cases (i.e. no average noise level values);
3. the high noise level is clearly grouped by time, i.e. most likely the reason lies in some global processes occurring on the supercomputer as a whole, which simultaneously affects many compute nodes;
4. most nodes are not affected (less than 1/4 of nodes were notably influenced during the considered time period).

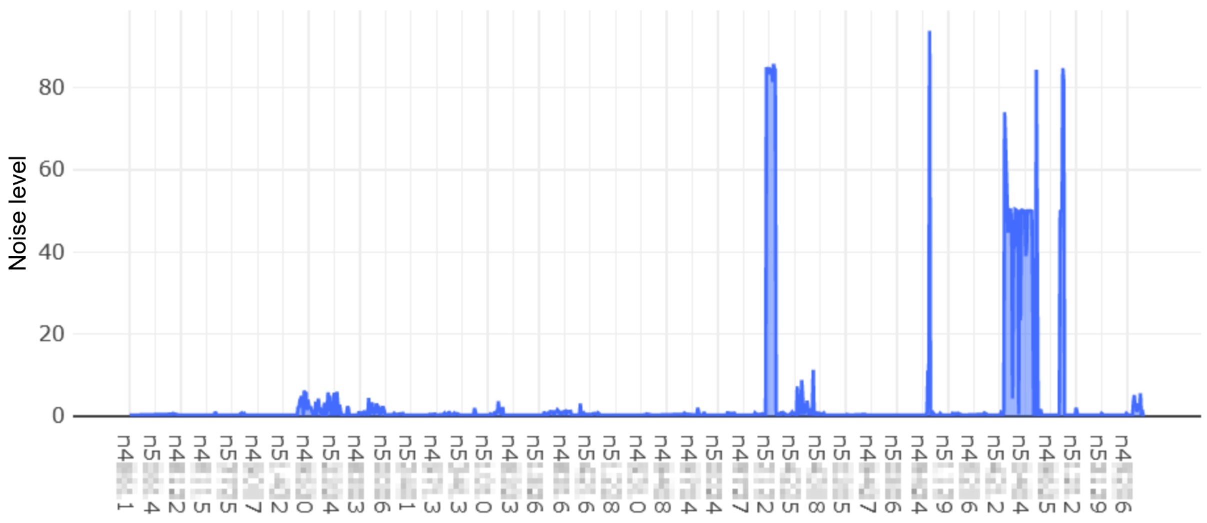


Figure 4. Last value of noise level at different nodes, sorted by the time it was received

Conclusions

This paper describes an approach to recurrent monitoring of OS noise on a supercomputer, which allows assessing the noise level on compute nodes as well as analyzing the dynamics of its change over time and the scale of its influence as a whole.

This approach was applied on the Lomonosov-2 supercomputer. Based on the evaluation results, it was found that the noise level is most often negligible, but on some nodes it can become significant. Further, it was found that noise most often occurs simultaneously on several nodes, which suggests that the cause of the noise is not localized within a node, but is global at the level of the supercomputer as a whole. Also, using the proposed approach, a node was discovered where the noise level is constantly very high, due to its malfunction.

Acknowledgements

The reported study was funded by RFBR and DFG, project number 21-57-12011. The research is carried out using the equipment of shared research facilities of HPC computing resources at Lomonosov Moscow State University.

References

1. Backfill scheduling, `https://slurm.schedmd.com/sched_config.html#backfill`
2. Characteristics of Lomonosov-2 supercomputer, `https://parallel.ru/cluster/lomonosov2.html`
3. Current rating of the 50 most powerful supercomputers in CIS, `http://top50.supercomputers.ru/?page=rating`

4. Frontier supercomputer debuts as world's fastest, breaking exascale barrier, https://www.ornl.gov/news/frontier-supercomputer-debuts-worlds-fastest-breaking-exascale-barrier

5. Redash homepage, https://redash.io/

6. TOP500 list, https://top500.org/lists/top500/

7. Afzal, A., Hager, G., Wellein, G.: Propagation and decay of injected one-off delays on clusters: a case study. In: 2019 IEEE International Conference on Cluster Computing (CLUSTER). pp. 1–10. IEEE (2019). https://doi.org/10.1109/CLUSTER.2019.8890995

8. Agarwal, S., Garg, R., Vishnoi, N.K.: The impact of noise on the scaling of collectives: A theoretical approach. In: High Performance Computing – HiPC 2005. HiPC 2005. Lecture Notes in Computer Science, vol. 3769, pp. 280–289. Springer (2005). https://doi.org/10.1007/11602569_31

9. Akkan, H., Lang, M., Liebrock, L.: Understanding and isolating the noise in the Linux kernel. The International Journal of High Performance Computing Applications 27(2), 136–146 (2013). https://doi.org/10.1177/1094342013477892

10. De, P., Kothari, R., Mann, V.: Identifying sources of operating system jitter through fine-grained kernel instrumentation. In: Proceedings of the 2007 IEEE International Conference on Cluster Computing, ICCC. pp. 331–340. IEEE (2007). https://doi.org/10.1109/CLUSTR.2007.4629247

11. De, P., Mann, V.: jitSim: A simulator for predicting scalability of parallel applications in presence of OS jitter. In: Euro-Par 2010 - Parallel Processing, 16th International Euro-Par Conference, Proceedings, Part I. Lecture Notes in Computer Science, vol. 6271, pp. 117–130. Springer (2010). https://doi.org/10.1007/978-3-642-15277-1_12

12. De, P., Mann, V., Mittal, U.: Handling OS jitter on multicore multithreaded systems. In: Proceedings of the 2009 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2009. pp. 1–12. IEEE (2009). https://doi.org/10.1109/IPDPS.2009.5161046

13. Ferreira, K.B., Bridges, P., Brightwell, R.: Characterizing application sensitivity to OS interference using kernel-level noise injection. In: Proceedings of the ACM/IEEE Conference on High Performance Computing, SC 2008. pp. 1–12. IEEE/ACM (2008). https://doi.org/10.1109/SC.2008.5219920

14. Garg, R., De, P.: Impact of noise on scaling of collectives: An empirical evaluation. In: High Performance Computing - HiPC 2006, 13th International Conference, Proceedings. Lecture Notes in Computer Science, vol. 4297, pp. 460–471. Springer (2006). https://doi.org/10.1007/11945918_45

15. Hoefler, T., Mehlan, T., Lumsdaine, A., Rehm, W.: Netgauge: A network performance measurement framework. In: High Performance Computing and Communications, Third International Conference, HPCC 2007, Proceedings. Lecture Notes in Computer Science, vol. 4782, pp. 659–671. Springer (2007). https://doi.org/10.1007/978-3-540-75444-2_62

16. Hoefler, T., Schneider, T., Lumsdaine, A.: Characterizing the influence of system noise on large-scale applications by simulation. In: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, SC'10. pp. 1–11. IEEE (2010). https://doi.org/10.1109/SC.2010.12

17. Jones, T.: Linux kernel co-scheduling for bulk synchronous parallel applications. In: Proceedings of the 1st International Workshop on Runtime and Operating Systems for Supercomputers. pp. 57–64. ACM (2011). https://doi.org/10.1145/1988796.1988805

18. Khudoleeva, A., Stefanov, K., Voevodin, V.: Evaluating the Impact of MPI Network Sharing on HPC Applications. In: Parallel Computational Technologies. PCT 2023. Communications in Computer and Information Science, vol. 1868, pp. 3–18. Springer (2023). https://doi.org/10.1007/978-3-031-38864-4_1

19. Mondragon, O.H., Bridges, P.G., Levy, S., Ferreira, K.B., Widener, P.: Understanding performance interference in next-generation HPC systems. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2016. pp. 384–395. IEEE (2016). https://doi.org/10.1109/SC.2016.32

20. Nikitenko, D., Mohr, B., Wolf, F., *et al.*: Influence of Noisy Environments on Behavior of HPC Applications. Lobachevskii Journal of Mathematics 42(7), 1560–1570 (2021). https://doi.org/10.1134/S1995080221070192

21. Nikitenko, D., Antonov, A., Shvets, P., *et al.*: JobDigest – Detailed System Monitoring-Based Supercomputer Application Behavior Analysis. In: Supercomputing. Third Russian Supercomputing Days, RuSCDays 2017, Moscow, Russia, September 25-26, 2017, Revised Selected Papers. Communications in Computer and Information Science, vol. 793, pp. 516–529. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-71255-0_42

22. Shvets, P., Voevodin, V., Nikitenko, D.: Approach to Workload Analysis of Large HPC Centers. In: Parallel Computational Technologies. PCT 2020. Communications in Computer and Information Science, vol. 1263, pp. 16–30. Springer (2020). https://doi.org/10.1007/978-3-030-55326-5_2

23. Stefanov, K., Voevodin, V., Zhumatiy, S., Voevodin, V.: Dynamically Reconfigurable Distributed Modular Monitoring System for Supercomputers (DiMMon). Procedia Computer Science 66, 625–634 (2015). https://doi.org/10.1016/j.procs.2015.11.071

24. Tsafrir, D., Etsion, Y., Feitelson, D.G., Kirkpatrick, S.: System noise, OS clock ticks, and fine-grained parallel applications. In: Proceedings of the 19th Annual International Conference on Supercomputing. pp. 303–312. ACM (2005). https://doi.org/10.1145/1088149.1088190

25. Voevodin, V., Antonov, A., Nikitenko, D., *et al.*: Supercomputer Lomonosov-2: Large scale, deep monitoring and fine analytics for the user community. Supercomputing Frontiers and Innovations 6(2), 4–11 (2019). https://doi.org/10.14529/jsfi190201