DOI: 10.14529/jsfi210305

Administration, Monitoring and Analysis of Supercomputers in Russia: a Survey of 10 HPC Centers

Vadim V. Voevodin[1], *Roman A. Chulkevich*[2], *Pavel S. Kostenetskiy*[2], *Vyacheslav I. Kozyrev*[2], *Anton K. Maliutin*[3], *Dmitry A. Nikitenko*[1], *Sergey G. Rykovanov*[3], *Artemiy B. Shamsutdinov*[2], *Yurii N. Shkandybin*[3], *Sergey A. Zhumatiy*[1]

Supercomputer technologies are in demand for solving many important and computationally-intensive tasks in various fields of science and technology. Therefore, it is not surprising that there are several dozen supercomputer centers only in Russia. However, the goals of creating such centers, as well as the range of tasks solved in them, can vary greatly, therefore the structure of supercomputers and the policies for their usage can significantly differ. This leads to the fact that many supercomputer centers live an isolated life – the administrators of such centers tend to solve administration-related tasks on their own, despite the fact that solutions for many similar tasks have already been developed and applied in other centers. This can happen due to different reasons, but in any case, this situation could and should be improved. To do this, it is worth establishing a closer connection between supercomputer centers, which will allow more actively exchanging experience or jointly developing desired system software. In order to understand the current situation in this area, a survey was conducted of representatives among 10 large supercomputer centers in Russia, and its results are presented in this paper. Two relevant topics about using monitoring data in practice and real-life examples of supercomputer functioning improvement are also discussed here in more detail. Their vision on these topics is provided by the system administrators of HSE University, Skoltech and Moscow State University.

Keywords: supercomputer, high-performance computing, administration, survey, monitoring, performance.

Introduction

The high-performance computing (HPC) area is in great demand in the modern world [19]. There are various important problems in all major subject areas that cannot be solved without supercomputer technologies, and the number of such problems is constantly growing. For this reason, the HPC area itself also grows [11]. Each year supercomputers are becoming more powerful and more complex, and there are more and more of them, which contributes to the faster development of science and technology.

At the same time, the architecture of many supercomputers is noticeably different – in some cases, it is more important to use accelerators, but for someone a fast memory or a powerful interconnect is of main interest; in some organizations, a universal system is needed that allows effectively solving a variety of problems from different scientific areas, in others – a specialized system designed for extremely efficient solving of one specific class of problems [9]. Therefore, approaches to dealing with the issues of maintaining, monitoring and ensuring the efficient functioning of supercomputers also often differ.

All this leads to the fact that many supercomputer centers (SC) live a rather isolated life – if it is necessary to solve a certain administration-related task (e.g., implement new user access policies or quotas, configure file system or resource manager, monitor different aspects of compute

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

[2] HSE University, Moscow, Russian Federation

[3] Skolkovo Institute of Science and Technology (Skoltech), Moscow, Russian Federation

nodes utilization, etc.), new solutions are often developed and implemented, despite the fact that similar solutions have already been proposed in another center. This often happens because either a previously developed solution was not designed to be portable, or that solution is not directly suitable and needs to be adapted. In such cases, it is usually easier for administrators to develop their own solution than to try to adapt an existing one. Also, very often administrators were simply not aware that such a solution had already been found and implemented by someone. Moreover, administrators often have to solely struggle with the issues of deciding which system software is most suitable or how to optimally configure it, although exchange of experience with other colleagues could significantly simplify this task.

In our opinion, this situation can be improved. It can be achieved by exchanging experience in organizing the efficient functioning of SCs, as well as using the best practices from various SCs when developing new software tools and methods. For these purposes, at the end of year 2020, a working group on the analysis and quality assurance of supercomputer center functioning was created [3]. This group brings together system administrators and analysts from different Russian supercomputing centers. The circle of major interests of this group can be outlined as follows:

- efficiency of using supercomputer resources in general and executing HPC applications in particular;
- technologies for holistic monitoring of a supercomputer functioning as well as its individual components (both performance and operability issues are of interest);
- methods and system software for a comprehensive performance analysis of supercomputer applications;
- effective organization of the supercomputer work (project management, access rights, quotas, policies, etc.).

In order to study the current situation with the administration of supercomputers in practice, this working group has conducted a survey of 10 different Russian supercomputer centers on the above issues. The results, as well as a more detailed discussion of several questions raised in the survey, are given in this paper.

At the moment, only one similar work has been discovered [14], which has focused on studying operational data measurement, collection and analysis in 9 different large centers in the USA, Germany, Italy and Japan.

The main contribution of this paper is the collection and detailed analysis of various information about the current situation with the administration of modern Russian HPC centers. This information can be primarily useful in practice for system administrators who want to learn from the experience of others. And it can as well be useful for HPC center management and common users in order to understand the general picture of what is happening in this area and the relevance of the issues solved there.

The rest of the paper is organized as follows. Section 1 briefly describes how a survey has been conducted. In section 2, the most interesting results from the conducted survey are presented and analyzed. Section 3 discusses two actual topics about using monitoring data in practice and real-life examples of supercomputer functioning improvement in more detail. In this section, the vision on these topics is provided by the system administrators of HSE University, Skoltech (Skolkovo Institute of Science and Technology) and Lomonosov Moscow State University. The conclusions are made in the last section.

1. Conducting a Survey

The survey was completed by system administrators of 10 different supercomputer centers. Brief description of these centers (showing affiliation and the most powerful system in each case) is presented in Tab. 1. "# in Top50" column refers to the position of the corresponding system in the Top50 supercomputing rating [6].

Table 1. List of supercomputing centers that participated in the survey and corresponding most powerful systems with their parameters

#	Top HPC sites at each center	Performance, TFlop/s	Nodes	# in Top50
1	Lomonosov-2, Supercomputing center of M.V. Lomonosov Moscow State University, Moscow	max: 6669 peak: 8789.76	1696	2
2	Polytechnic RSC Tornado, Peter the Great St. Petersburg Polytechnic University, St. Petersburg	max: 910.31 peak: 1309	784	4
3	cHARISMa, HSE University, Moscow	max: 653.7 peak: 1003.2	48	6
4	Zhores, Skoltech, Moscow	max: 495.9 peak: 1011.6	82	8
5	Govorun, SKYLAKE component Joint Institute for Nuclear Research, Dubna	max: 312.62 peak: 463.26	104	12
6	Lobachevsky, Lobachevsky Nizhni Novgorod State University, Nizhny Novgorod	max: 289.5 peak: 573	180	13
7	Tornado SUSU, South Ural State University, Chelyabinsk	max: 288.2 peak: 473.64	384	14
8	Uran, Krasovskii Institute of Mathematics and Mechanics, UB RAS, Ekaterinburg	max: 194.77 peak: 326.85	76	18
9	NKS-1P, SSCC, SB RAS, Novosibirsk	max: 85.45 peak: 136.94	48	37
10	Polus, M.V. Lomonosov Moscow State University, Faculty of Computational Mathematics and Cybernetics, Moscow	max: 40.39 peak: 55.84	5	N/A

The survey was conducted using Google Forms and included 25 questions concerning the following topics:

- monitoring data collection;
- usage of different system software;
- automation of administrative routines;
- understanding the general behavior of supercomputer systems;
- resource management;
- supercomputer support.

Most of the questions had multiple predefined options to choose from, but almost always there was an opportunity to give your own answer. There were cases when several administrators from one center completed the survey; their answers were combined and presented as one.

2. Analyzing Survey Results

This section discusses the most interesting topics covered in the survey. A presentation with all questions and answers (in Russian) can be found on the web-site of the working group [5].

The first two questions of the survey were related to the data collection, on the compute nodes (Fig. 1) and the engineering infrastructure (Fig. 2) correspondingly.

Figure 1 shows that most of the supercomputer centers collect information about CPU load and file systems usage – 7 out of 10 administrators have chosen these options. A little less (6 out of 10) centers collect more information about memory usage (DRAM, I/O), network load, power consumption and temperature. In our opinion, these characteristics are quite expectedly the most popular ones, since they reflect in general the usage of main supercomputer resources as well as basic operability characteristics. It is interesting to notice that 6 out of 10 centers can analyze the performance of user applications since they bind the collected monitoring data to the tasks launched on compute nodes.

It is also worth noting that not so many supercomputer centers are interested in getting more detailed information about node and/or task behavior using processor counters. Example of what data could be of interest and how such data can be used is provided in section 3.1.3.

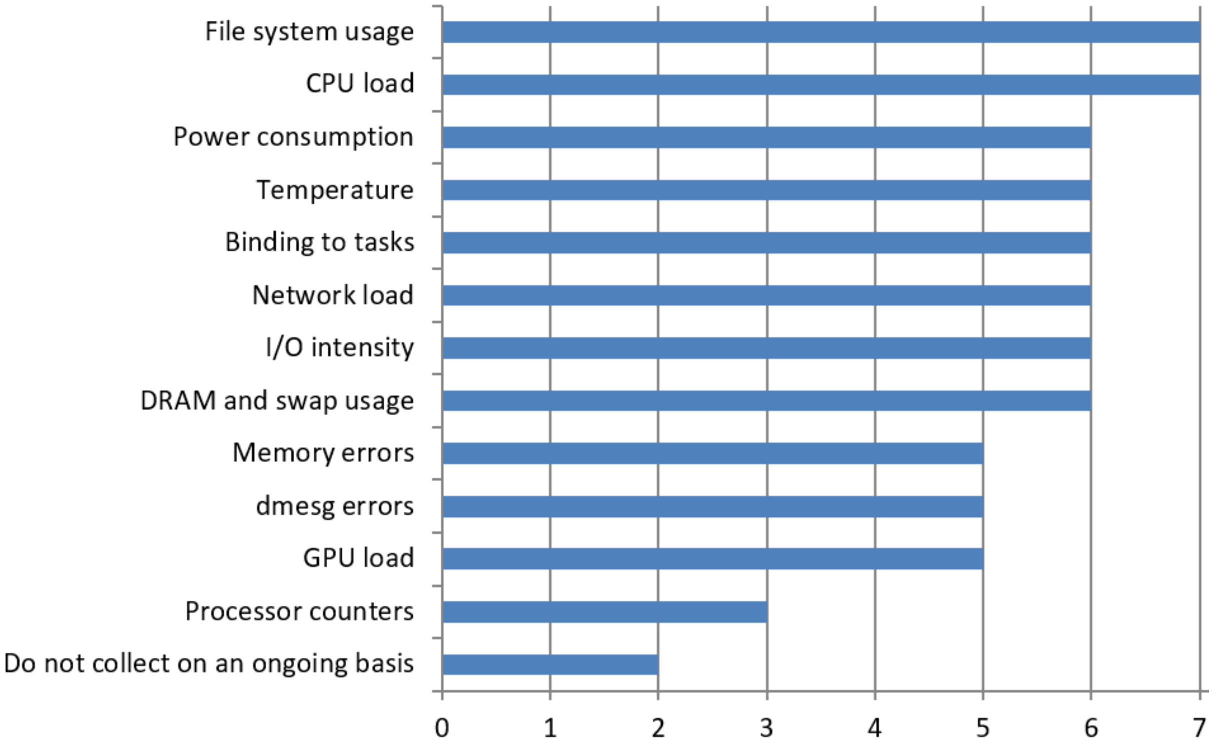


Figure 1. Answers to the question "What data do you collect on the compute nodes?"

The statistics shown in Fig. 2 are quite expected as well. The most crucial question for administrators is the condition and operability of the supercomputer, so almost every center collects data on the temperature (8 out of 10), estimated battery life (7 out of 10) and humidity (7 out of 10). The security question is less crucial but still important, that is why different centers collect indoor video (4 out of 10) as well as information about who entered the room (3 out of 10) and from motion sensors (1 out of 10).

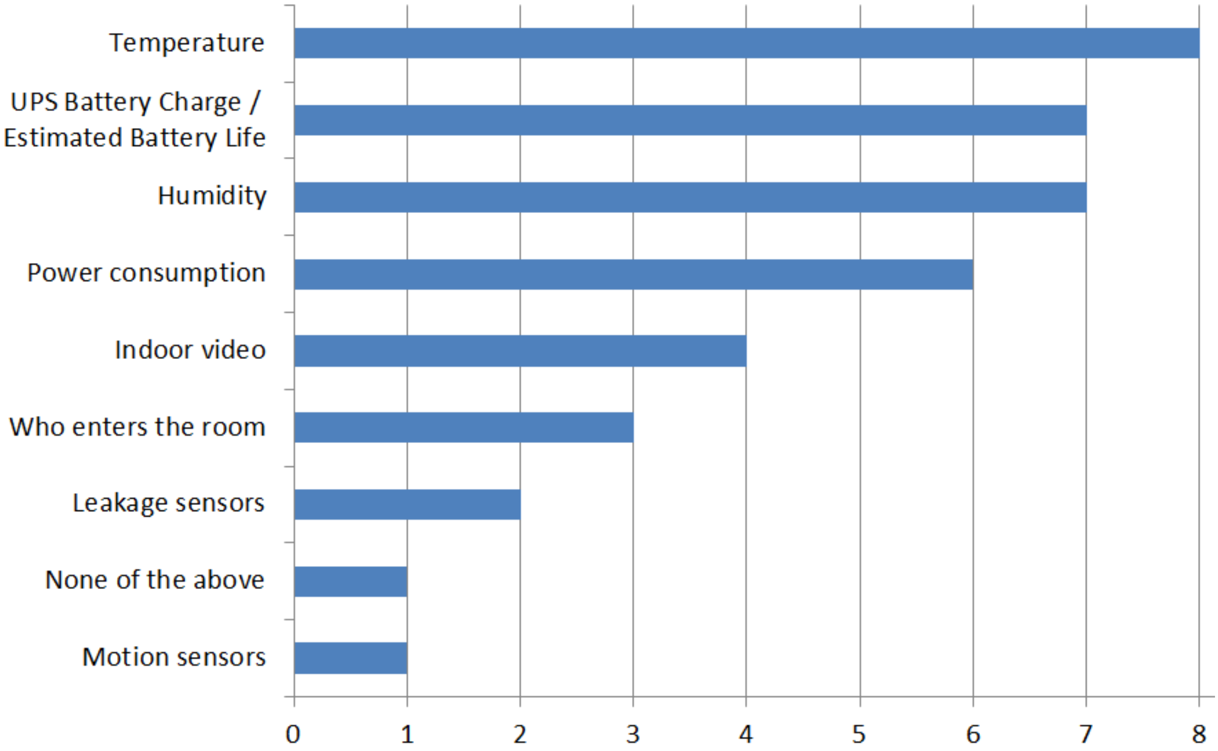


Figure 2. Answers to the question "What data do you collect about the engineering infrastructure?"

One of the most interesting topics to investigate was to find out about the usage of different system software that can help to obtain more insights about the state, behavior and usage of a supercomputer. For example, it was interesting to study whether there are generally accepted ready-to-use solutions that are widely used, and what proprietary solutions are used. The survey included questions about the use of the following software:

- monitoring systems;
- database management systems (DBMS);
- data visualization;
- data stream processing;
- data analysis.

Every question had a predefined list of the most accepted software, with the ability to specify other solutions as well. For each option, it was necessary to indicate how it is used: 1) as a main solution; 2) as a supplementary tool; 3) planned to be used in future.

First, let us take a closer look at monitoring systems. Figure 3 provides the distribution of answers to the following question: "What monitoring systems do you use to collect data on the work of your supercomputer center?".

These results were quite surprising for us. As seen, the most popular option is the "proprietary solutions", which means that administrators use their own developed software more often than any of existing ready-to-use monitoring systems. In our opinion, this is a rather alarming situation, since the development of your own monitoring system is most often cumbersome. The question arises why administrators are not satisfied with ready-made solutions and whether

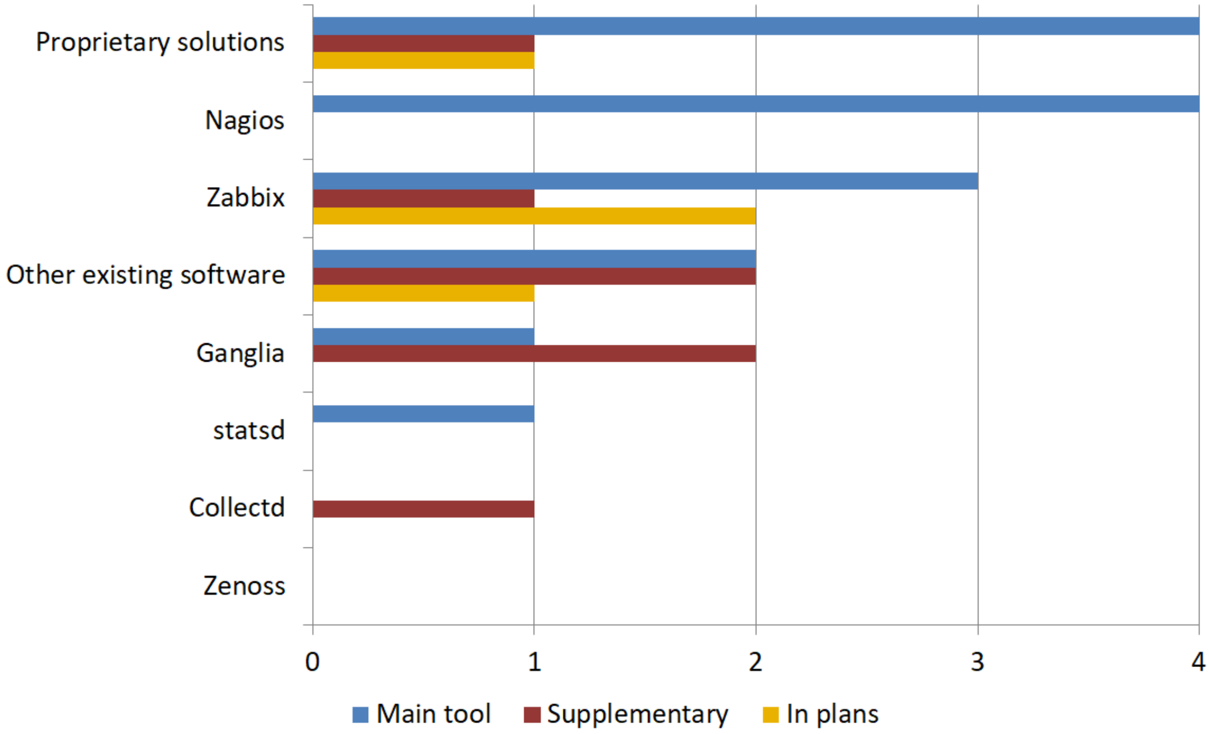


Figure 3. Information about monitoring systems used in different supercomputer centers

it is possible to simplify the solution of this problem by exchanging experience or sharing the developed ideas. This issue is further discussed in the section 3.1.

Other results in Fig. 3 show that Nagios and Zabbix are by far the most popular existing solutions, both mentioned 4 times as being used (2 centers also plan to use Zabbix in future). It should be mentioned that Telegraf and Icinga were mentioned twice each in the "other existing software" category, what makes them more popular than statsd and Collectd.

The situation with DBMS used to store data on SC work is shown in Fig. 4. SQL-related databases are traditionally very popular, with MySQL/MariaDB on the first place (used by 4 out of 10 centers) and PostgreSQL on the second place (3 out of 10). Time-series databases are used quite rarely – only 3 centers in total claim to use them, with InfluxDB being the most popular. This was quite surprising: they were expected to be used more often since most of the collected data is presented in the form of time series. Also, MongoDB is used by 2 centers, but only as the supplementary one. The ElasticSearch solution was also expected to be used more often, since the ELK stack (ElasticSearch, Logstash, Kibana) seemed to be quite famous for solving similar tasks, but it turned out not to be the case.

The next question was about visualization systems that help to present the information about different aspects of supercomputer functioning. The situation here is quite expected: Graphana is by far the most commonly used solution (5 out of 10 centers), followed by Kibana (2 centers). All other mentioned software – Redash, Jupyter Notebook, and two commercial solutions (IBM Blue Gene Navigator and HPE CMU) – is used only in one center each. Only one center uses its own developed solution in this case.

The situation with data stream processing tools is even more expected – most of the centers do not use such tools. This can be explained by the fact that the collected data streams are currently not so large and complicated that they can be processed without such tools. Only two ready-to-use solutions were mentioned – kapacitor and RabbitMQ, used by one center each. Also,

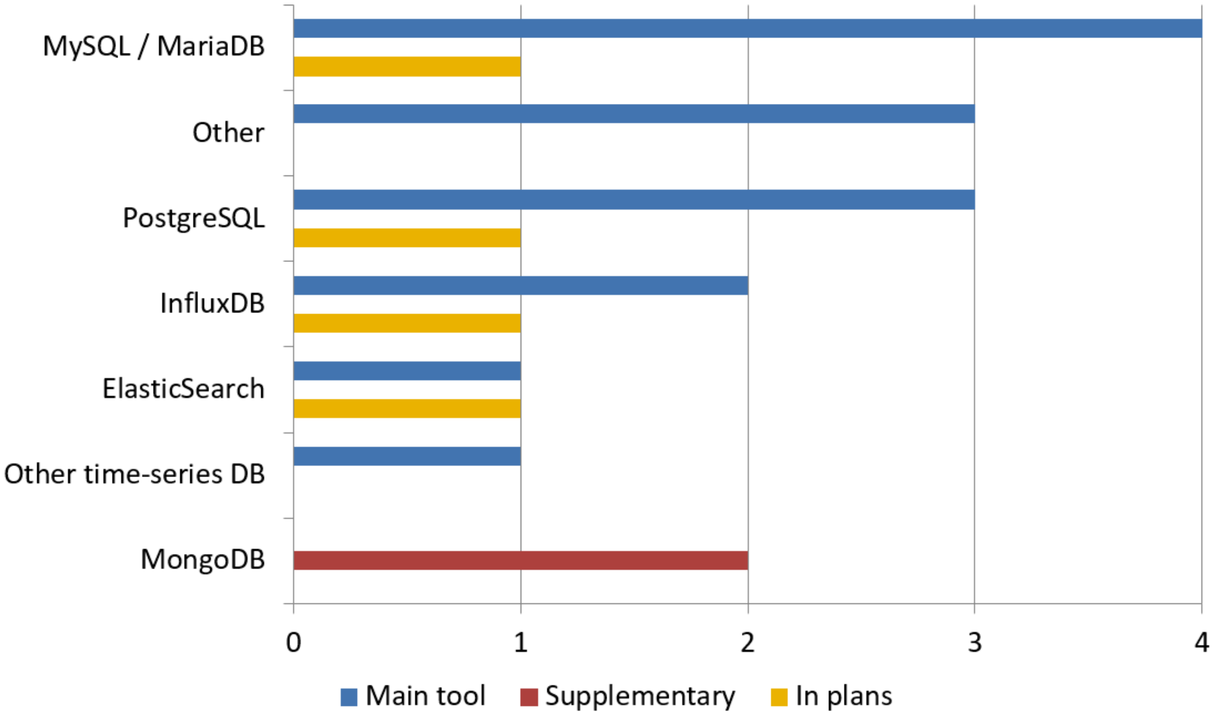


Figure 4. Information about DBMS systems used in different supercomputer centers

two centers claim to apply their own developed solutions, and in this case it would be interesting to drill into further details to find out why existing software does not fit their needs.

Interesting answers were received to the question "What software do you use to analyze data on supercomputer work?", shown in Fig. 5. By far the most common answer in this case is "proprietary solutions", with 4 centers using them as main or supplementary solutions. This is caused by different reasons, and, in our opinion, one of the main reasons is that there is no existing solution that can fully address this issue. All existing software mentioned in Fig. 5 help to process and analyze data, but they require additional work (sometimes a lot) to integrate them into the whole flow of working with supercomputer data. This shows that this area is currently the least developed one among those considered, and it is worth paying special attention to the joint solution of this issue.

The last question we want to discuss in this section is a general one – "What do you lack for a more complete understanding of the state of your supercomputer center?". The distribution of answers is shown in Fig. 6. The most important conclusion from the data presented is that not a single administrator responded that they had everything they needed to fully understand the behavior of their supercomputers. This means that the potential for development in this area is great, and it is worthwhile to join efforts to improve this situation. Otherwise, the distribution of answers in general is quite natural: many centers are already collecting data on the correctness of the supercomputer's operation, so this option received the least number of votes. At the same time, the main problem that prevents administrators from understanding in more detail the state of supercomputers is the lack of time and people to help conduct such an analysis. We would like to point out that many centers also want more "intellectual" data analysis software, which once again emphasizes the lack of such ready-to-use solutions and their relevance in practice.

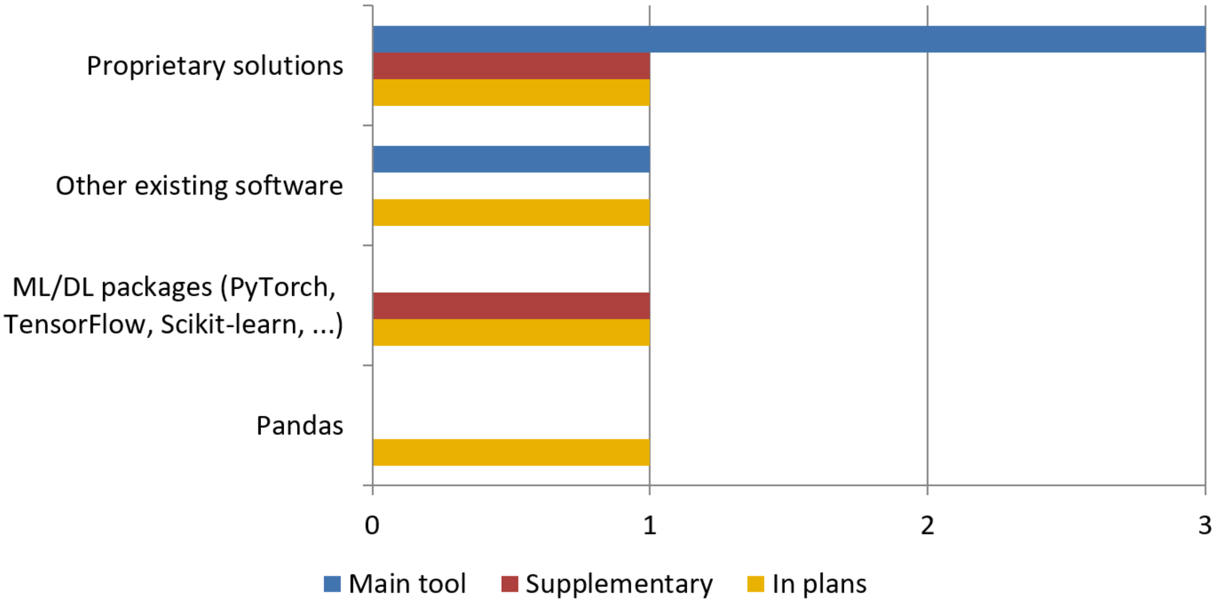


Figure 5. Information about data analysis systems used in different supercomputer centers

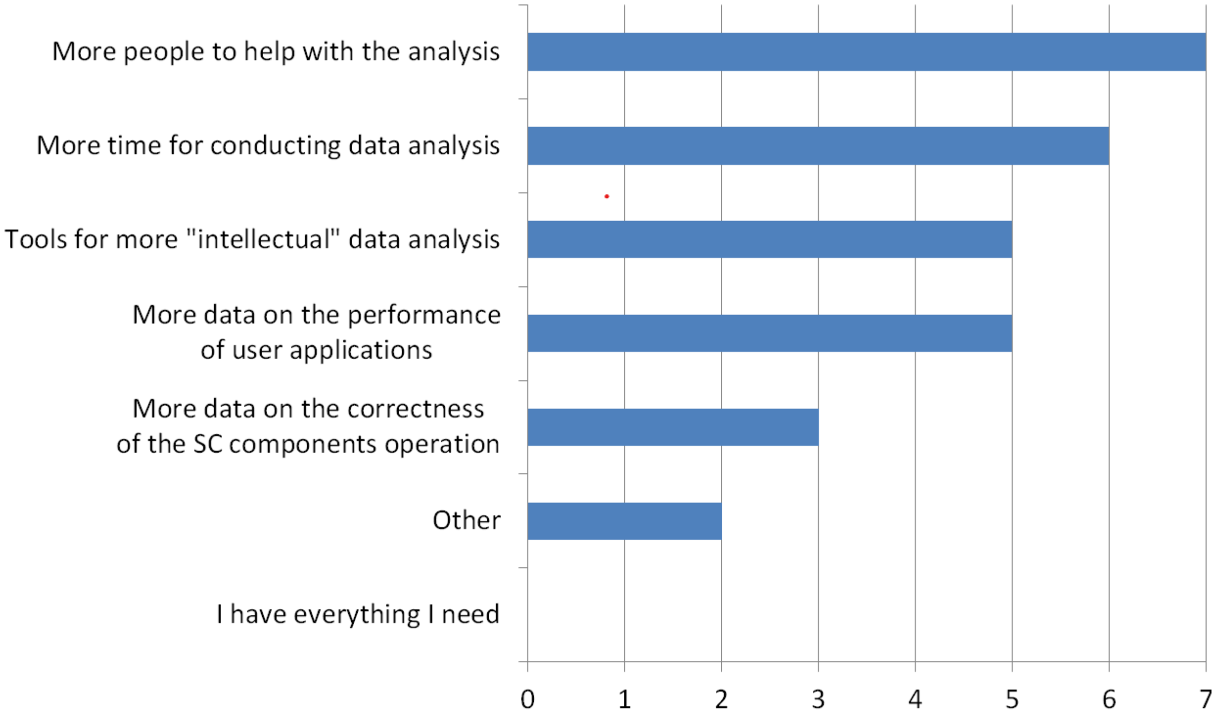


Figure 6. Answers to the question “What do you lack for a more complete understanding of the state of your supercomputer center?”

3. Diving into Details of Supercomputer Administration in Practice

In this section, the following questions of interest raised in the survey are considered in more detail:

1. Using monitoring data in practice.
2. Real-life examples of supercomputer functioning improvement.

In order to get a more complete picture of each of them, their vision on these issues is provided by the system administrators of HSE Univesity, Skoltech and Moscow State University.

3.1. Using Monitoring Data in Practice

One of the main questions that arises when administering supercomputers is what data to collect about its health and performance, how to collect this data and what insights can be obtained from it. We will talk about this in this section.

3.1.1. HSE University

HSE University has developed its own HPC cluster monitoring system. The key feature of the system is the interactive dashboard, which displays the state of the entire supercomputer on one screen (see Fig. 7). The system collects data from various sources, processes and logically combines it. This allows simultaneously visualizing the real load, the distribution of resources between users, the status of the task queue, and the health of computing nodes. The system's dashboard has functionality for managing the task queue. For example, the head of an HPC department can directly raise an urgent user task in the queue through the web interface, and the system will automatically recalculate the priorities of tasks and pass them to the task scheduler. Also, the system allows centrally changing the resource limits for different types of users (students, employees), which is convenient during holidays, when the cluster load decreases, and the restrictions on the number of computing resources used by users can be loosened.

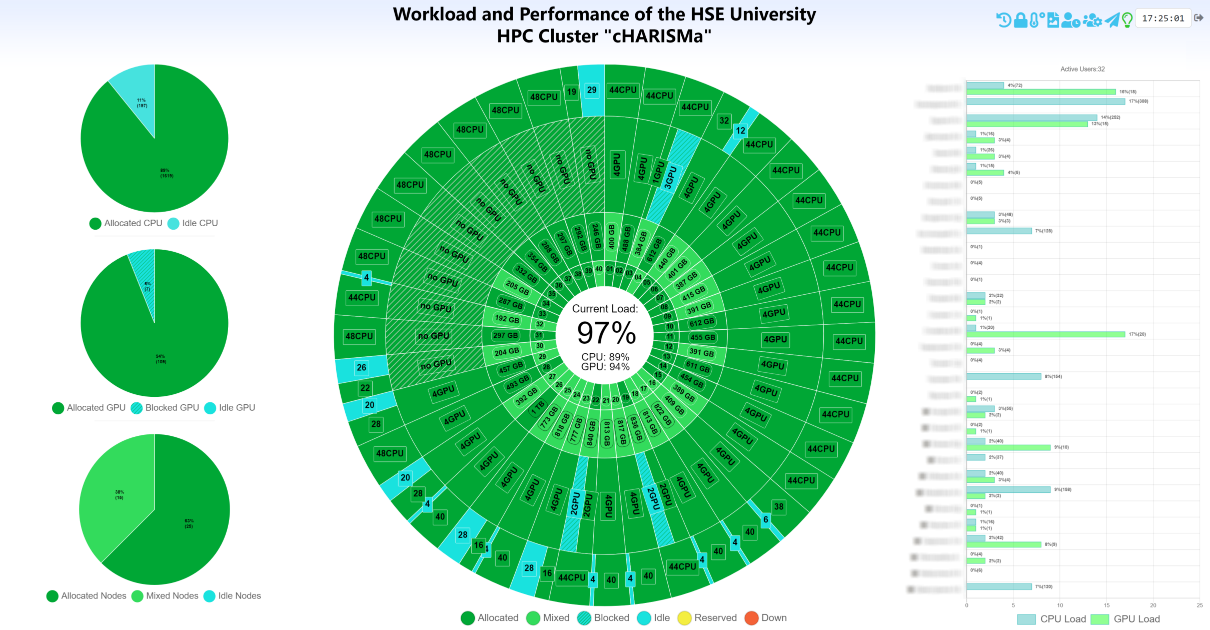


Figure 7. Monitoring system of the HSE university supercomputer facilities

Access to the monitoring system of the HSE University supercomputer center is available not only to the system administrators of the HPC center but also to the top management of the university. Different users of the monitoring system receive various functionality. The system allows managers to create summary graphs and reports on the use of the supercomputer.

With the help of the developed monitoring system, it was possible to detect and eliminate many non-optimal settings in the configuration of the supercomputer. As a result, many cluster optimizations were performed, some of which are listed below.

Running multiple tasks on the same node. The compute nodes in the HSE HPC cluster have a configuration that differs from many other high-performance installations. Instead of a large number of nodes with average performance and small memory, *cHARISMa* has powerful nodes with a large amount of RAM (up to 1.5 TB). Therefore, the minimum entity available to the user is not the entire computing node, but 1 processor core, or 1 GPU. As a result, up to 48 tasks of different users can run simultaneously on one node. Thus, it became possible to significantly increase the number of simultaneously solved tasks, which greatly increased the efficiency of the whole computing cluster.

Optimization of the queue system. The fact that the task manager in the HSE HPC cluster does not allocate the entire node, but specific cores or GPUs, can lead to the high fragmentation of the computational field. By default, SLURM tries to place new tasks on the least loaded nodes. As a result, the cluster can be loaded by 30–40 %, but a large number of tasks will wait in the queue, which requires many processor cores on one node. To reduce resource downtime, an improved task grouping (sched/backfill + pack-serial-at-end) was configured. Tasks are now placed in such a way as to maximize a load of already partially occupied dedicated compute nodes.

Protection of resources from accidental use. Another feature of the system is the protection from the possible accidental usage of other user's resources. By default, the control over which resources on the node are available to the user is carried out using the operating system environment variables. At the same time, the user can accidentally or intentionally change these variables and get access to resources that are not allocated to him. To solve this problem, isolation of tasks from each other on a specific compute node (group, etc.) was implemented. As a result, the executed task process sees only his computing resources and does not imply the existence of any others.

Fair distribution of computing resources. Monitoring the statistics in the monitoring system allowed us to discover that some users sometimes abuse the available computing resources, for example, when one user launches several hundred tasks of the same type, which occupy the cluster for a long time. As a solution, a limit on the number of resources used simultaneously by one user was implemented. For the convenience of administrators and managers, the ability to manage this restriction in the web interface of the monitoring system was added.

Protection against GPU blocking. Another point that the system monitors is the blocking of graphics accelerators. To perform a task on a graphics accelerator, the user needs at least one processor core. However, if the tasks of another user have occupied all the available processor cores on this node, then the graphics accelerators will not be available for allocation. To solve this problem, an additional task queue plugin was implemented. When a task appears that can block the graphics accelerator on the node, the plugin issues a warning and slightly changes the number of resources to avoid blocking.

Script automation. The administration of a cluster complex is usually a time-consuming process. The complexity is a consequence of the fact that a cluster consists of many compute nodes and network equipment. Automation of routine processes using scripts allows simplifying the administration of cluster. An example of script automation is the simple Supercomputer

Emergency Shutdown System (ESS), developed at HSE University for the *cHARISMa* HPC cluster. The system is essential because there are events that require immediate response:

- failure of the cooling system of the compute node;
- failure of the indoor air conditioning system;
- switching the UPS to battery mode;
- low level of battery power of the UPS.

The emergency shutdown system makes asynchronous requests and checks:

- The temperature at the input of the compute node (RedFish REST API). A high temperature indicates a general malfunction of the indoor air conditioning system.
- The temperature at the output of the compute node (RedFish REST API). A high temperature indicates a malfunction of the cooling system of a particular node.
- UPS input status and remaining battery life (SNMP). The lack of input voltage and low battery charge indicates a possible accident in the power supply system.

The developed ESS has successfully supplemented the local overheating prevention system built into the firmware of compute nodes. It automated and streamlined the process of shutting down the supercomputer in the event of an accident, preventing equipment breakdowns. The system has already been triggered during a power outage and has shown its effectiveness and usefulness.

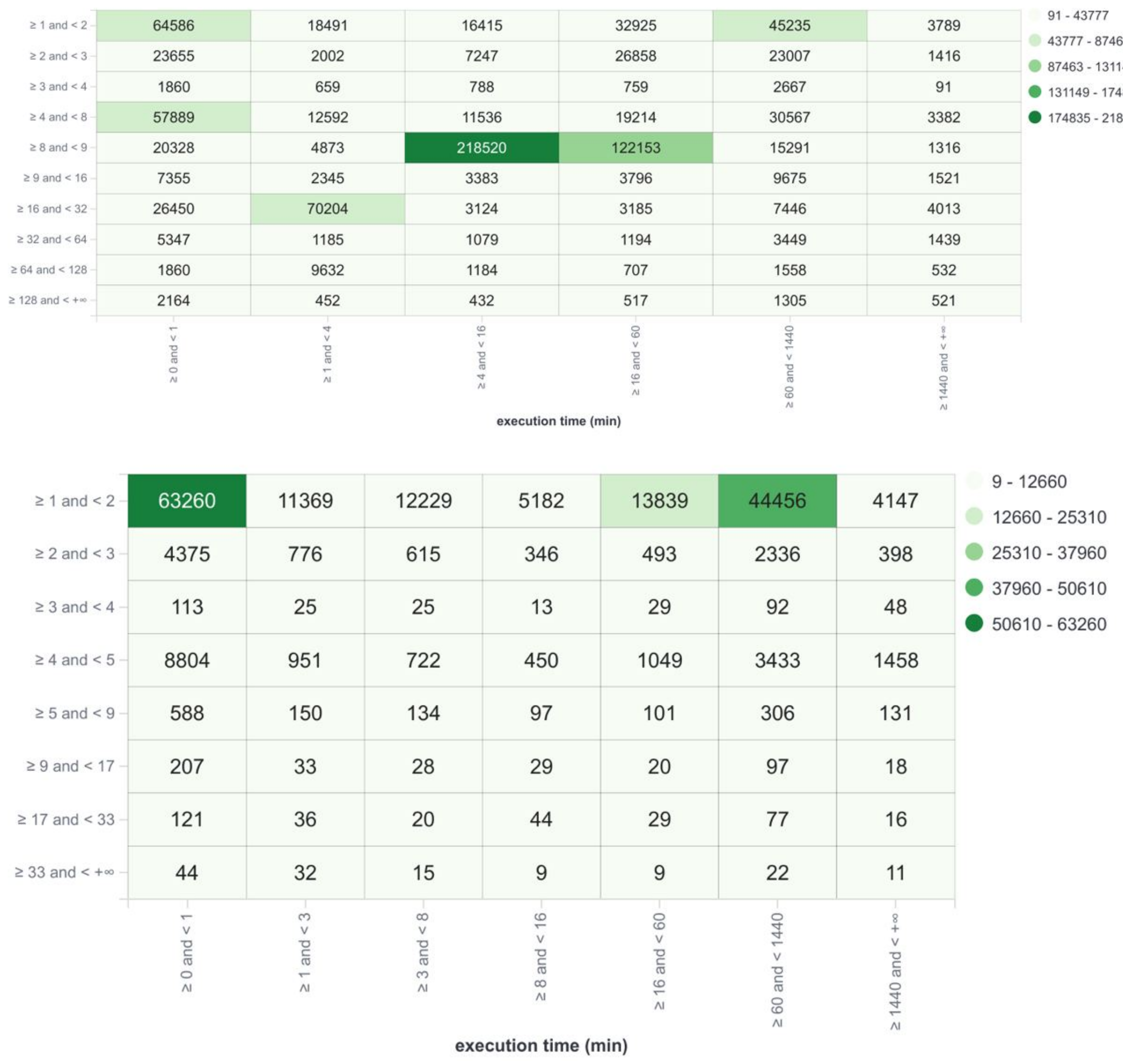
3.1.2. Skoltech

The range of tasks that a modern supercomputer solves goes beyond the scope of classical HPC tasks. In Skoltech researchers and engineers working on fundamental and applied science are faced with different types of problems. These might be high throughput computing (HTC) tasks that require a large number of single CPU cores or GPUs running independently, HPC tasks that require a large fraction of all resources to work on a single problem, or data-intensive tasks with ML/DL approach applied to solving them. The difficulty is that all types of tasks have to be solved effectively within a single supercomputer both from the user and from the economical/environmental perspectives (the latter can be described as the amount of time the system is running idle and amount of time the system is performing productive calculations). In Skoltech the task of increasing the “Zhores” supercomputer load efficiency has been gradually performed since the start of the supercomputing facility late in 2018 [23].

First, the inventory of tools that can obtain data on computing efficiency was performed giving the profile of typical tasks performed on a supercomputer and an estimate of their average efficiency in terms of resource utilization. The tools that were used for gathering analyzing data are the following:

- Slurm workload manager [22];
- Elasticsearch [10] for job properties aggregation;
- Kibana for data visualization;
- Zabbix for infrastructure monitoring;
- a home brewed tool for a more detailed “Zhores” cluster; monitoring [24];
- users surveys and individual interaction.

An example of the heatmap of number of launched jobs as a function of running time and requested resources for the “Zhores” cluster is presented in Fig. 8. This allowed us to create a “profile” of an average user, find out the needs of the users and tune the Slurm queues correspondingly (see sec. 3.2.2 for further details). After an initial assessment of the collected data, it



	≥ 0 and < 1	≥ 1 and < 4	≥ 4 and < 16	≥ 16 and < 60	≥ 60 and < 1440	≥ 1440 and < +∞
≥ 1 and < 2	64586	18491	16415	32925	45235	3789
≥ 2 and < 3	23655	2002	7247	26858	23007	1416
≥ 3 and < 4	1860	659	788	759	2667	91
≥ 4 and < 8	57889	12592	11536	19214	30567	3382
≥ 8 and < 9	20328	4873	218520	122153	15291	1316
≥ 9 and < 16	7355	2345	3383	3796	9675	1521
≥ 16 and < 32	26450	70204	3124	3185	7446	4013
≥ 32 and < 64	5347	1185	1079	1194	3449	1439
≥ 64 and < 128	1860	9632	1184	707	1558	532
≥ 128 and < +∞	2164	452	432	517	1305	521

	≥ 0 and < 1	≥ 1 and < 3	≥ 3 and < 8	≥ 8 and < 16	≥ 16 and < 60	≥ 60 and < 1440	≥ 1440 and < +∞
≥ 1 and < 2	63260	11369	12229	5182	13839	44456	4147
≥ 2 and < 3	4375	776	615	346	493	2336	398
≥ 3 and < 4	113	25	25	13	29	92	48
≥ 4 and < 5	8804	951	722	450	1049	3433	1458
≥ 5 and < 9	588	150	134	97	101	306	131
≥ 9 and < 17	207	33	28	29	20	97	18
≥ 17 and < 33	121	36	20	44	29	77	16
≥ 33 and < +∞	44	32	15	9	9	22	11

Figure 8. Heatmaps of the number of jobs launched on the "Zhores" cluster as function of time spent on the running time (longitudinal axis) and requested resources (number of CPU cores for upper plot and number of GPUs for the lower plot, vertical axis). Each cell shows the number of launched jobs

was found that the performance of tasks using GPU accelerators is significantly lower compared to tasks using exclusively CPU computing resources. For CPU tasks it was 55–60 %, for GPU tasks – less than 25 %. Three main problems that get in the way of efficient cluster usage were identified:

1. Tasks interfere with each other, large tasks block the launch of small ones, and vice versa. The queuing time increases, and large windows of equipment downtime appear.
2. Resources are allocated by the user, but tasks are launched with a long delay or poorly load the computational resources, which is especially true for GPU tasks for deep learning often run using Jupyter notebooks [8].
3. Under certain conditions some tasks work inefficiently. For example, they might lack a needed amount of RAM or are poorly optimized for parallel performance. As a result, the resources utilization is low while the allocation is high.

Possible solutions to these three identified problems are given below in sec. 3.2.2.

3.1.3. Moscow State University

Collecting data. Data collection in the Lomonosov-2 supercomputer [21] installed in MSU has been organized since the beginning and is constantly being improved. At the current moment, the performance monitoring of compute nodes is conducted using our own DiMMon monitoring system [18]. We were not satisfied with existing common solutions like Zabbix, Nagios, etc, because we wanted to have a more flexible and lightweight solution with richer functionality, such as:

- ability to change data collection interval on-the-fly for certain nodes, e.g., nodes running a specific job;
- ability to turn off data collection for nodes not running any jobs;
- automatic aggregation of data from nodes running one job;
- ability to turn off data collecting for nodes running a specific job (e.g., to collect data via trace collector or similar tools).

On each node, we collect the following types of data:

- different types of CPU load (user, iowait, idle, system) and load average;
- GPU usage (GPU load and memory utilization);
- Infiniband usage intensity (amount of bytes/packets sent/received per second);
- Lustre file system usage (number of opened and closed files, amount of read/written bytes plus service data like page faults);
- performance hardware counters (number of retired instructions and unhalted cycles per second, number of cache misses to L1 and LLC per second);
- other memory characteristics (ECC errors and free memory).

This data is collected once every second, being aggregated once every minute and stored in PostgreSQL database. It should be noted that this performance data from compute nodes could be efficiently stored in time-series DB as well (since it is a set of time series), but PostgreSQL, initially selected for different reasons, copes without problems with this amount of data (several GBs per day).

This node performance data is also then binded to the user jobs running on these nodes, and integral information about each job is stored separately in the MongoDB database, together with more data describing different features of these jobs: Slurm data with launch parameters, information about the project and organization for which the user works, data on software packages and libraries being used on jobs.

Besides monitoring of compute nodes, service servers are also monitored, in order to control their load and correctness of their work. We check ping as well as load average and disk space on each server, using Telegraph monitoring, with data being stored in VictoriaMetrics [4] database. On Lustre servers disks activity, Infiniband stats, CPU system time, Lustre read/writes, Lustre cache access rate and memory statistics also are collected via Telegraph service and stored in the VictoriaMetrics database.

How we use the data we collect. There are several different ways all this collected data is used in practice in order to help us to control and improve the performance of the Lomonosov-2 supercomputer.

One of the obvious and quite efficient ways to analyze job performance based on the collected monitoring data is to investigate various graphs showing timelines with job behavior according to a specific characteristic. This idea was implemented in the JobDigest [13] – a software for generating web reports presenting different information on the performance of the chosen job. JobDigest automatically creates such reports for all jobs running on the Lomonosov-2, which are available both for users (job owners only) and system administrators.

One example of how this data can be used is shown in Fig. 9. It is a heatmap showing the change of a number of L1 cache misses per second, with distribution among nodes used in the job. It should be noted that this data is collected using processor counters via PAPI [20]. As seen, most nodes show very low number of cache misses, while one node constantly shows over 280 million of misses per second, which is a very high value. In this case, we can assume that a program uses memory both very intensively and inefficiently on this node. Judging from this graph alone we cannot be sure that this leads to a decrease in the performance of the program, but this is definitely a potential bottleneck which should be primarily investigated during performance analysis.

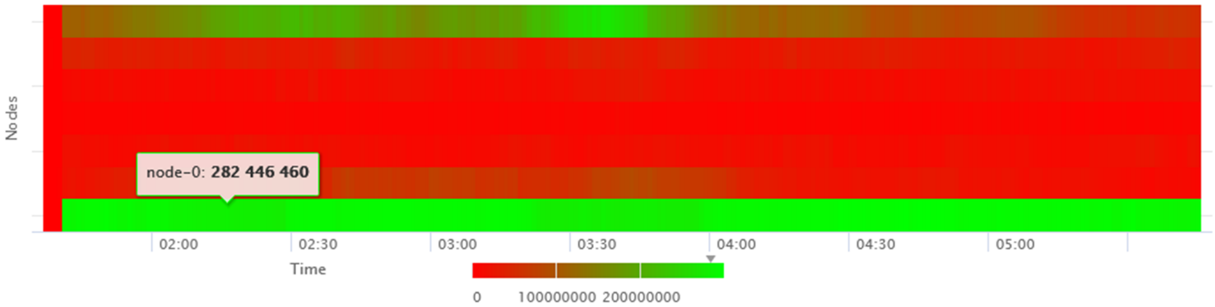
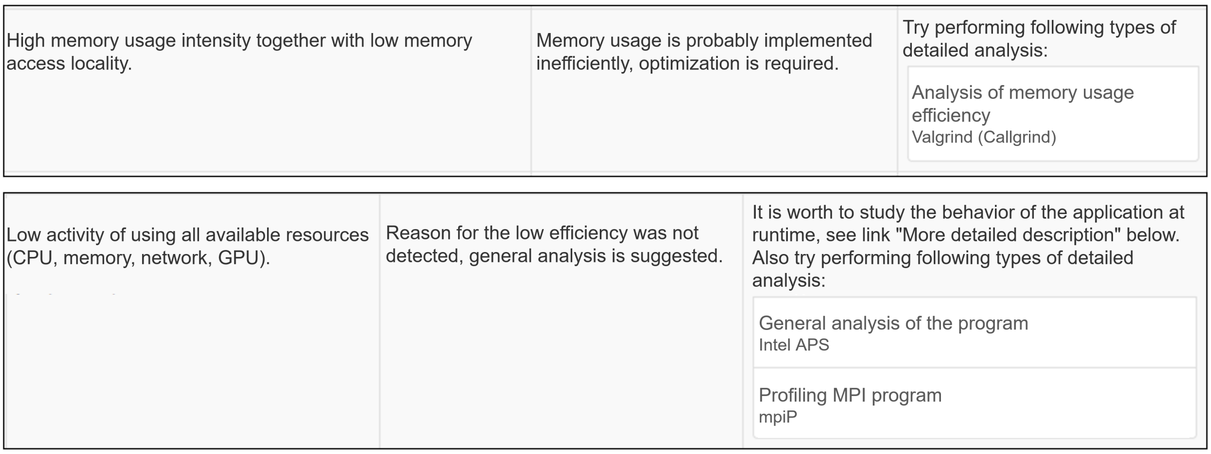


Figure 9. Distribution of the maximum number of L1 cache misses per second between nodes during job execution

JobDigest reports can be useful, but it is usually hard for users to properly analyze such low-level information and get insights from it. To help with this, a software package called TASC (Tuning Applications for SuperComputers) [17] for more “intellectual” data analysis was developed at the Moscow State University. Its main purpose is to help administrators and users of a supercomputer in detecting and eliminating a variety of issues with the performance of a supercomputer in general and individual applications in particular. For this purpose, it provides users with a detailed information about different performance characteristics of their applications as well as automatically detects potential performance issues (concerning CPU or memory utilization, efficiency of network or I/O usage, optimality and correctness of job launches, etc.) and notifies users about them.

An example of performance issues automatically detected in a job is shown in Fig. 10. Such information is available (in Russian) within the Octoshell system for all users of Lomonosov-2 supercomputer about their jobs. For each detected issue, there is a description of what seems to be the problem, what potentially could cause it and recommendations on its further analysis. Note that when clicking on the desired type of further analysis, a detailed manual on how to conduct it and what to look at is provided to a user.

The aforementioned examples show how the collected data can be useful for the supercomputer users. But also TASC, by integrating and analyzing together a huge amount of data on each main aspect of supercomputer behavior, allows system administrators to quickly study any aspect of interest with the desired level of details, as well as automatically notifies them about different critical issues automatically detected on the level of the whole supercomputer. Among such issues that were detected in practice are the inefficient usage of software packages by sev-



High memory usage intensity together with low memory access locality.	Memory usage is probably implemented inefficiently, optimization is required.	Try performing following types of detailed analysis: Analysis of memory usage efficiency Valgrind (Callgrind)

Low activity of using all available resources (CPU, memory, network, GPU).	Reason for the low efficiency was not detected, general analysis is suggested.	It is worth to study the behavior of the application at runtime, see link "More detailed description" below. Also try performing following types of detailed analysis: General analysis of the program Intel APS Profiling MPI program mpiP

Figure 10. Description of performance issues detected in a user job with recommendations on its further study (originally available in Russian)

eral users, failures in file system functioning, as well as inefficient job launches with excessively high load of compute nodes. Another direction of analyzing the collected monitoring data being of interest for the administrators, which is also studied in MSU, is using data mining methods for detecting similar jobs in order to get more insights on the structure and peculiarities of supercomputer job flow [16].

Data on service servers is visualized using Grafana package [2]. Most critical monitored metrics are controlled using Grafana alerts – in any suspicious or dangerous situation an alert message is being sent via email and Telegram to the administrators and engineers. We are planning to change alerting procedure to another service, because of low flexibility of Grafana alerting abilities – now we are testing the open-source software package "balerter" [1].

Grafana is also used to visualize current and historical supercomputer usage – number of used, free and disabled nodes, number of running and waiting jobs. Special dashboard shows current statuses of all nodes and list of reasons for node disabling. Another important dashboard shows incoming and outgoing water temperatures in the air conditioners and current energy consumption. Data for these dashboards are collected via SNMP protocol from air conditioners and power distribution units and stored into VictoriaMetrics database. All these dashboards are used on a daily basis by our system administrators to monitor and analyze the behavior of the Lomonosov-2 supercomputer.

3.2. Real-life Examples of Supercomputer Functioning Improvement

This topic is based on the survey question stated as "Please describe examples of real-life situations when the analysis of data on the supercomputer functioning helped you to distinctly improve the quality of its work.". Such examples appeared in three stated supercomputer centers are provided below.

3.2.1. HSE University

HSE University has a complex task efficiency monitoring system of its own development on the *cHARISMa* HPC cluster. It is called *HPC TaskMaster*. The system allows monitoring tasks of *cHARISMa* users, creating reports with task information and dynamic graphs, and automatically detects inefficient tasks by finding problems related to resource utilization and creating conclusions about the work of the task. The system is designed to help cluster users run their tasks more efficiently, therefore saving expensive supercomputer machine time.

Finding inefficient tasks is an important problem that all large clusters face. To classify a task as inefficient, it is necessary to rely on its various indicators. Such indicators can be low or extremely high rates of components utilization and too short or too long task duration. For many tasks, it is essential to determine their type before analyzing its effectiveness: for example, Jupyter Notebook and Gromacs will have completely different behavior and their indicators must be analyzed in different ways.

The principle of operation of *HPC TaskMaster* is briefly described in Fig. 11. From each compute node of the *cHARISMa* cluster, the *HPC TaskMaster* system collects time series of such characteristics as usage of CPU, GPU, file system, RAM, InfiniBand network. After that, the system aggregates the time series, obtaining the average, minimum and maximum values for the characteristics. The resulting aggregated metrics are stored as a tuple of values, which is further processed by the indicator definition functions. The resulting tuple contains indicator levels for each type of processed value from 0 to 1 according to the level of manifestation.

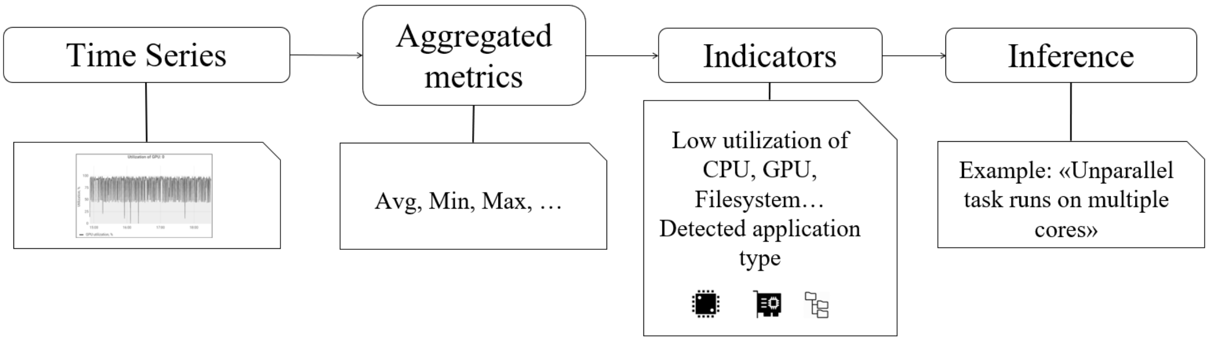

Time Series
Aggregated metrics
Indicators
Inference
Avg, Min, Max, …
Low utilization of CPU, GPU, Filesystem… Detected application type
Example: «Unparallel task runs on multiple cores»

Figure 11. Stages of data processing

After the tuple of indicators is created, it is compared with a list of all possible inferences specified in the system. When all the indicator values fall within the specified limits, the corresponding inference is assigned to the task.

There are some examples of inferences that can be generated by the system:

- *Incorrect launch of jupyter-notebook application.* The running task type is detected as a jupyter-notebook application, and there are indicators of low utilization of CPU or GPU.
- *Allocating resources without running computations.* The user has allocated resources for the task, but there are indicators of low resource utilization.
- *Running unparallel tasks on multiple nodes.* Several nodes are allocated for the task, but there are indicators of low InfiniBand network usage and low resource utilization.

If the indicator values fit several inferences, the one with the highest priority is selected. This system allows the administrator to conduct a flexible configuration of inferences for various types of tasks, without changing the source code of the system and quickly adding new inferences when new types of tasks appear on the cluster.

The start-up of this system allowed increasing the effective load of the HSE supercomputer by 16 %. This was made possible by detecting computing tasks with incorrect startup parameters and notifying users about them. In the near future, when the system is trained to find problems that are more complicated and issue expanded inferences, it will be possible to achieve even greater savings in computing resources.

3.2.2. Skoltech

The collected data and arising problems for the Skoltech supercomputer “Zhores” were described above in sec. 3.1.2. First, optimization of resource usage was performed by allowing out-of-order job launching in Slurm using the backfill method (see Fig. 12) as well as changing the properties of the queues. Backfill method drastically reduced queuing times and increased computation density.

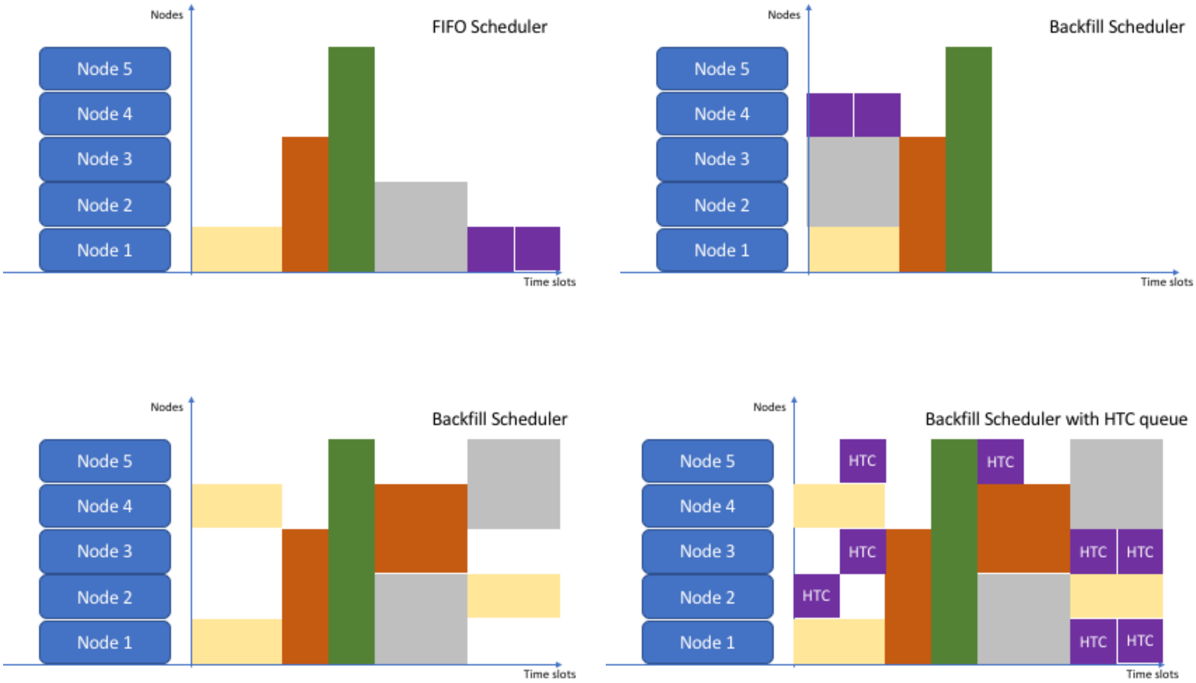


Figure 12. Backfill vs FIFO scheduler schematics. Backfill allows us to significantly densify the task launching (top left row for FIFO scheduler vs top right row for backfill scheduler). Allowing HTC tasks to be launched in free spots further leads to a more effective supercomputer usage (bottom left row for backfill scheduler without HTC vs bottom right row for backfill scheduler with HTC)

“Zhores” cluster has a hybrid architecture and consists of CPU-only nodes as well as nodes with modern GPUs. That is why the entire flow of computational tasks was divided into classes according to the required architecture (CPU, GPU and later big memory nodes) as well as required runtime. Before the majority of users started to use “Zhores” supercomputer the queues presented in Tab. 2 (only columns with the asterisk) were envisioned.

Later on the queuing scheme evolved into the one presented in Tab. 2 (only columns without the asterisk) and the following restrictions:

- one can not occupy more than half of all the resources of the queue;
- queues with lower time limit have higher priority;
- there is a minimum amount of RAM per allocated core.

Additionally, HTC tasks can be launched in all queues. HTC tasks are typically short and not resource-intensive but there is a huge amount of them. By filling all the free slots with HTC tasks, the utilization of CPU resources was increased by 15 %, and the time spent in the queue for HTC tasks was reduced by 6 times. Altogether, backfilling and restructuring the queues

Table 2. Queues distribution in "Zhores" cluster

*Queue name	*Time limit	Name	Queues description	DefaultTime	MaxTime	MaxNodesPerJob
cpu_debug cpu_small cpu_big	30 min. 24 hours 6 days	cpu	CPU nodes only	24h	6d	22
gpu_debug gpu_small gpu_big	30 min 24 hours 6 days	gpu	GPU nodes only	24h	6d	6
mem_debug mem_small mem_big	30 min 24 hours 6 days	mem	Big mem nodes	24h	6d	4
		htc	HTC jobs, 1 node per task all nodes above, less priority		24h	1

helped to reduce queuing times and increase CPU utilization by approximately 20–25 % and GPU utilization by approximately 5 %.

The next step was to increase the utilization of the most expensive resource – the GPUs. In the process of supercomputer performance monitoring it was found that users (mostly those working on ML/DL tasks and using Jupyter notebook environment) use nodes with powerful NVidia Tesla V100 GPUs for code prototyping and development. This leads to GPU resource allocation with almost zero utilization during the code development stage that can sometimes last longer than the actual running time. This is the usual difference between the traditional HPC workflow and "AI" workflow. To solve this problem, several dedicated servers each containing from 8 to 10 less powerful and lower cost NVidia GPUs (typically NVidia GTX 1080 or 2080 Ti) with support of the same software libraries that are used for the V100 (CUDA, Torch, Tensorflow, etc.) were purchased. Thus code prototyping and development zone was established on those servers with dynamic deployment of the Jupyter Hub environment and a set of all the necessary tools for using the GPU. By doing this it was achieved that in the vast majority only well debugged production codes were launched on the Tesla V100 GPUs and their utilization rate raised by approximately 30 %.

Third of the problems identified in sec. 3.1.2 was tackled by purchasing high-density servers with four processors and up to 3 terabytes RAM per each node. Such nodes could effectively solve problems where it was needed to keep a large amount of data in RAM, as well as problems that were poorly optimized for parallelization to several nodes. This differentiation of tools helped to solve user problems faster. With an identical load on computing resources, the average CPU task execution speed increased by 15 %.

Even if there is limited budget for hardware purchase, one can improve the efficiency of the supercomputer by increasing user awareness. They should not treat the cluster as a black box. Knowledge of its architecture, strengths and weaknesses enables the developer to create more efficient code. To increase HPC users knowledge and awareness, regular HPC seminars are held in Skoltech, an online system of interaction with users has been established, an HPC Wiki has been developed with examples of best practice. A Telegram channel has been created with news about the cluster and an overview of new approaches to solving computational problems. To better prepare students, lectures and laboratory works on basic architecture of a supercomputer were introduced into the HPC track of the Advanced Computational Science MSc educational program, within which students receive not only theoretical knowledge, but also build and learn to administer their own (small scale) supercomputer.

All the aforementioned steps have lead to the reduction of the difference between resources allocation percentage and utilization percentage. Average CPU utilization is approximately 90 % and average GPU utilization is more than 65 %.

3.2.3. Moscow State University

An aforementioned TASC software has been used on the Lomonosov-2 supercomputer for a couple years, and during this time it has helped to find and eliminate different performance issues. Let us discuss three of them.

TASC automatically detects different performance issues in all user applications running on the supercomputer. One of such issues, which is considered as critical, detects suspiciously low utilization of both CPU and GPU. TASC also provides web reports that allow analyzing the appearance of such issues among users, and such statistics for the first half of May 2020 has shown that 95 % of all "suspicious" node-hours belonged to one user. Further analysis of this user has revealed that he has made only 5 job launches during this time period, but they occupied 12000 node-hours (which is quite a lot) and all of them were "suspicious". This means that too many resources are idle, so this user was contacted. It turned out that these jobs were launched on 50 nodes each, but due to an error in a program only one node was actually involved. The user was unaware of it; after we contacted him this issue was eliminated.

Another example happened in July 2020. Among other things, TASC-based reports provide general statistics on overall user activity on Lomonosov-2. Figure 13 shows top 10 users based on the number of job launches during selected time period. It can be seen that the most active user has made over 1500 launches (leftmost column), which is almost 6 times more than the second most active user. At the same time, these jobs occupied only a small amount of node-hours (black line). Such situation is quite unusual, so it was decided to investigate it further. It appeared that almost all of the jobs were using Gromacs package [7] and lasted less than 5 minutes, which is even more unusual, especially for Gromacs users. We contacted this user, and it turned out that he was using a script that was automatically launching jobs which almost immediately fell with an error, forcing the script to start new jobs. After our request, the user fixed his script.

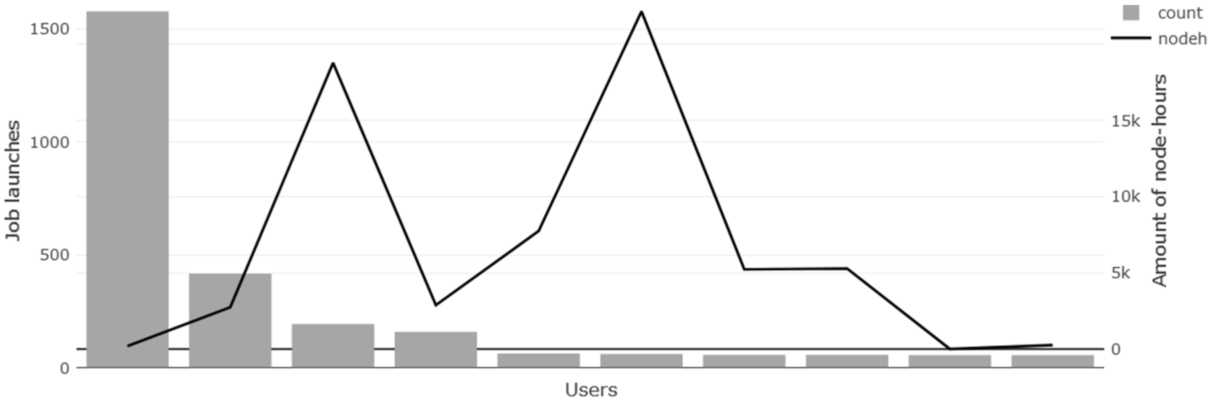


Figure 13. Top ten users of Lomonosov-2 supercomputer based on the number of job launches during beginning of July 2020

Another interesting example concerns the NAMD package [15] for solving molecular dynamics tasks. The Lomonosov-2 supercomputer has a special partition for GPU-intensive jobs. While studying the GPU utilization of this partition within year 2020, it was discovered that jobs of one active user was showing only 7 % of GPU load. Further analysis using TASC-based reports allowed us to see that almost all jobs were using NAMD package, while showing less than 10 % of GPU load each. It was the only user of NAMD package in this partition, but many other packages launched in this partition were showing much higher GPU load (Gromacs – 57 %, LAMMPS – 34 %). Next, we discovered that the GPU load in this user jobs when launching NAMD in another (main) partition was significantly higher – 57 %. This suggested that the problem was with NAMD build for the GPU-based partition, and it turned out to be so. After tuning this package and therefore fixing this issue, the GPU load of this package returned to normal.

Conclusions

This paper shows the results of a survey of system administrators from 10 large supercomputer centers in Russia regarding the issues of maintenance, monitoring and analysis of supercomputer behavior. This review allows, as a first approximation, to capture the general picture of the current state in this area. The information collected made it possible to find out what monitoring data is most interesting in practice, and what information is practically not collected; what existing systems for monitoring, storing, visualizing and analyzing data are most often used; etc. Of particular interest are the areas in which you most often have to develop your own solutions (i.e., data monitoring and analysis) – these are the areas in which there are no ready-made suitable solutions, and further development of these areas is worthwhile to be carried out collectively. At the same time, the results of the survey show that in all centers without exception there is a need for a more complete understanding of the state of supercomputers, which suggests that this area is important and needs to be further developed.

Two important topics are considered separately – using monitoring data in practice and real-life examples of supercomputer functioning improvement. For each topic, the administrators of three large centers (HSE University, Skoltech and Moscow State University) describe how they approach these issues in practice. Such a description helps to better understand the complexities and challenges in this area, as well as possible approaches to their solution.

Acknowledgements

This research was supported in part through computational resources of HPC facilities at HSE University [12]. The research is partially carried out using the equipment of shared research facilities of HPC computing resources at Lomonosov Moscow State University. The MSU part of the survey and the reported study was funded by the Russian Foundation for Basic Research, project no. 20-07-00864.

References

1. Balerter homepage. `https://balerter.com/0.8.1/getting_started/about.html`, accessed: 2021-08-26

2. Grafana: The open observability platform. `https://grafana.com/`, accessed: 2021-08-26

3. The working group on the analysis and quality assurance of supercomputer center functioning. https://scc-efficiency.parallel.ru/, accessed: 2021-08-26

4. VictoriaMetrics documentation. https://docs.victoriametrics.com/, accessed: 2021-08-26

5. Presentation with final survey results (in Russian). Tech. rep. (2021), https://scc-efficiency.parallel.ru/assets/final_scc_survey.pdf

6. Top 50 supercomputers list. http://top50.supercomputers.ru/list (2021), accessed: 2021-08-26

7. Abraham, M.J., Murtola, T., Schulz, R., et al.: GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. SoftwareX 1-2, 19–25 (2015). https://doi.org/10.1016/j.softx.2015.06.001

8. Community, E.B.: Jupyter Book (2020). https://doi.org/10.5281/zenodo.4539666

9. Deneroff, M.M., Shaw, D.E., Dror, R.O., et al.: Anton: A specialized ASIC for molecular dynamics. In: 2008 IEEE Hot Chips 20 Symposium (HCS). pp. 1–34 (2008). https://doi.org/10.1109/HOTCHIPS.2008.7476542

10. Gormley, C., Tong, Z.: Elasticsearch: The Definitive Guide. O'Reilly Media, Inc., 1st edn. (2015)

11. Joseph, E., Conway, S.: Major Trends in the Worldwide HPC Market. Tech. rep. (2017), https://hpcuserforum.com/presentations/stuttgart2017/IDC-update-HLRS.pdf

12. Kostenetskiy, P.S., Chulkevich, R.A., Kozyrev, V.I.: HPC Resources of the Higher School of Economics. Journal of Physics: Conference Series 1740, 012050 (2021). https://doi.org/10.1088/1742-6596/1740/1/012050

13. 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. pp. 516–529. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-71255-0_42

14. Ott, M., Shin, W., Bourassa, N., et al.: Global Experiences with HPC Operational Data Measurement, Collection and Analysis. In: IEEE International Conference on Cluster Computing, CLUSTER 2020. pp. 499–508. IEEE (2020). https://doi.org/10.1109/CLUSTER49012.2020.00071

15. Phillips, J.C., Braun, R., Wang, W., et al.: Scalable molecular dynamics with NAMD. Journal of Computational Chemistry 26(16), 1781–1802 (2005). https://doi.org/10.1002/jcc.20289

16. Shaikhislamov, D., Voevodin, V.: Solving the problem of detecting similar supercomputer applications using machine learning methods. In: Parallel Computational Technologies, PCT 2020. Communications in Computer and Information Science, vol. 1263, pp. 46–57. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-55326-5_4

17. 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, Cham (2020). https://doi.org/10.1007/978-3-030-55326-5_2

18. Stefanov, K., Voevodin, V., Zhumatiy, S., et al.: 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

19. Sterling, T., Anderson, M., Brodowicz, M.: High Performance Computing: Modern Systems and Practices. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edn. (2017). https://doi.org/10.1016/C2013-0-09704-6

20. Terpstra, D., Jagode, H., You, H., et al.: Collecting performance data with PAPI-C. In: Müller, M.S., Resch, M.M., Schulz, A., Nagel, W.E. (eds.) Tools for High Performance Computing 2009. pp. 157–173. Springer, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11261-4_11

21. Voevodin, V.V., Antonov, A.S., Nikitenko, D.A., 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

22. Yoo, A.B., Jette, M.A., Grondona, M.: Slurm: Simple linux utility for resource management. In: Feitelson, D., Rudolph, L., Schwiegelshohn, U. (eds.) Job Scheduling Strategies for Parallel Processing. pp. 44–60. Springer, Berlin, Heidelberg (2003). https://doi.org/10.1007/10968987_3

23. Zacharov, I., Arslanov, R., Gunin, M., et al.: "Zhores" – Petaflops supercomputer for data-driven modeling, machine learning and artificial intelligence installed in Skolkovo Institute of Science and Technology. Open Engineering 9(1), 512–520 (2019). https://doi.org/10.1515/eng-2019-0059

24. Zacharov, I., Panarin, O., Rykovanov, S., et al.: Monitoring applications on the ZHORES cluster at Skoltech. Program Systems: Theory and Applications 12(2), 73–103 (2021). https://doi.org/10.25209/2079-3316-2021-12-2-73-103