DOI: 10.14529/cmse240304

TASC SOFTWARE FOR HPC PERFORMANCE ANALYSIS: CURRENT STATE AND LATEST DEVELOPMENTS

© 2024 V.V. Voevodin, D.I. Shaikhislamov, V.A. Serov

Lomonosov Moscow State University

(Leninskie Gory 1, Moscow, 119991 Russia)

E-mail: vadim@parallel.ru, sdenis1995@gmail.com, sva@srcc.msu.ru

Received: 04.09.2024

To ensure high operating efficiency of modern supercomputers, it is necessary to constantly analyze and control various aspects of their behavior, paying special attention to the flow of supercomputer applications running on these machines. To solve this problem, the TASC (Tuning Applications for SuperComputers) software suite was previously developed. It automatically detects performance issues in HPC applications and evaluates the efficiency of using supercomputer resources, provides supercomputer administrators with a flexible report tool for analyzing different aspects of supercomputer functioning with the desired level of detail, and estimates the noise level on compute nodes. This paper provides full-scale description of current TASC structure and capabilities, including the stages of data processing and storing, as well as performing different types of analysis. It also describes new results obtained and methods developed within one of the main TASC components — assessment system for quick and accurate evaluation of HPC resources usage efficiency.

Keywords: high-performance computing, supercomputer, performance analysis, workload analysis, operational data analytics, monitoring.

FOR CITATION

Voevodin V.V., Shaikhislamov D.I., Serov V.A. TASC Software for HPC Performance Analysis: Current State and Latest Developments. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2024. Vol. 13, no. 3. P. 61–78. DOI: 10.14529/cmse240304.

Introduction

Supercomputers are used to substantially increase the speed of performing computational experiments, which are needed in a wide variety of different scientific and commercial researches in any major subject area. This means that the speed of supercomputer computations is one of the key characteristics, and the task of increasing it is among the most important ones. However, it turns out in practice that the performance of many supercomputer applications is quite low [1], since they efficiently utilize only a fraction of resources provided to them. This is due to both the extreme complexity of the structure of modern supercomputers (which leads to the fact that correctly taking into account all the hardware peculiarities to achieve maximum performance has become a very difficult task), and the complexity of writing high-performance parallel applications in general. And this becomes more and more topical, since the HPC market is growing [2, 3].

The problem is complicated by the fact that many users are unaware that their applications are not working efficiently. And performance issues can arise for various reasons: inefficient implementation of the application itself; poor matching of the selected software implementation and the target hardware; external factors caused by the surrounding supercomputer environment, and many others. One possible way to solve this problem of unawareness about inefficient use of supercomputer resources is to constantly analyze the flow of running applications, notifying

both the users about their inefficient applications and the supercomputer administrators about the situation in general.

With this in mind, the TASC (Tuning Applications for SuperComputers) software suite was previously developed at the Research Computing Center of Lomonosov Moscow State University. This suite includes several software components that use different approaches for solving the stated task. A tool for automatic detection of job-level performance issues [4] allows to promptly notify users about potential performance issues in their applications, which is done based on a set of predefined rules. An assessment system [1] is designed to automatically provide a quick and accurate evaluation of the efficiency of using supercomputer resources. An "endless" web report system [5] provides administrators with a very flexible set of performance-related dashboards that allows them to analyze only the required data subset with desired level of detail. And a noise level estimation system [6] helps to understand to what extent system processes (which are always running in the background on compute nodes) actively interfere with the execution of user applications. In this case, "noise" is defined as an external influence of the software and hardware environment leading to a change in the execution time or other properties of user applications running on the supercomputer, and one of the most common noise sources is operating system processes.

This paper provides for the first time a full-scale description of the current state of the TASC software suite, as well as presents the latest methods developed within it and results obtained using this solution.

The question of the efficiency analysis of applications running on supercomputers is touched upon in several existing studies. There are a number of papers like [7–9] presenting holistic reports on the operation of specific HPC systems. In these papers, many major aspects of supercomputer functioning are considered: job launch properties (duration, number of allocated processors, used software libraries, etc.); job efficiency in terms of CPU or GPU load, intensity of communication network usage; distribution of job, node-hours or efficiency characteristics between partitions or subject areas; and many others. However, these were mostly manually performed studies, therefore not portable and not directly applicable in other supercomputer centers. There are also works that study the efficiency of individual aspects of supercomputer behavior in more detail. For example, the authors of [10] in detail discuss how the job properties and trends in resource consumption of two large supercomputers, Intrepid and Mira, have changed over several years of operation. In paper [11], authors present a software solution for automatic detection of problems with job efficiency and behavior. Another paper [12] discusses the causes of low I/O throughput and provides a dedicated software solution to detect them. There are other similar works as well, but the difference in TASC is that it aims at holistic automated analysis of many different aspects, also focusing on thorough analysis of monitoring data.

The main paper is organized as follows. Section 1 shows the current TASC architecture and capabilities, listing the input data being used and describing approaches to storing this data (subsection 1.1), as well as showing main tools for performance analysis (subsection 1.2). Section 2 is devoted to the description of new methods, tools and results that have been obtained recently, which are all related to the assessment system. In subsection 2.1, we describe new visualization tools for the assessment system, new methods for informing users about the results obtained are described in 2.2, and in 2.3 we describe the data mining method for continuous prediction of assessments. We show interesting examples of using this system in section 3. Conclusions summarize the work done.

1. TASC Current State

TASC software suite is intended for carrying out ODA (Operational Data Analytics, e.g., see [13, 14]) on a supercomputer. It performs continuous performance analysis and visualization of data on behavior of supercomputer in general and applications running on in particular. In this section, we will describe TASC architecture and functionality, using its existing implementation running on the Lomonosov-2 supercomputer [15] as an example.

The overall working algorithm of TASC software suite is quite straightforward. Most input data needed for performance analysis carried out by TASC is collected using monitoring systems running on compute and service nodes of a supercomputer. The data is sent via network to the dedicated virtual machine running main TASC services, where it is processed and stored into databases. Analysis tools are launched on this virtual machine at the selected frequency, retrieving data from databases, studying it and storing results back to databases. The obtained insights are available to supercomputer users and administrators via websites and notifications using email or messengers. Thus, two main stages of TASC operation can be distinguished: 1) processing and storing the required input data, and 2) subsequent analysis of this data. We will describe these stages in more detail below.

1.1. Processing and Storing Data for Subsequent Analysis

TASC does not directly collect raw data (monitoring systems are mainly used for this purpose), so this process is not described here. However, the variety of collected input data is of interest. Figure 1 lists the types and sources of data that are being regularly collected on the Lomonosov-2 supercomputer. Each grey rectangle corresponds to a specific target (hardware or software) being monitored, a yellow rounded rectangle — to a specific tool and the data it collects. Within the rounded rectangle, it is shown what tool is used for data collection (top right), what types of data is collected (center) and what the granularity of data collection is (bottom right).

The main source of performance data is compute nodes, and we use DiMMon monitoring system [16] developed in RCC MSU to collect it. This data is collected once per second, then it is aggregated and sent to TASC once per minute. For each job running on Lomonosov-2, TASC gets the following information:

- CPU load and load average;
- intensity of sent/received bytes and packets via MPI network;
- GPU load and memory usage;
- Lustre file system usage (amount of read/write bytes and opened/closed files);
- performance monitoring counters describing CPU usage activity and cache misses;
- amount of available RAM;
- existence of ECC errors.

On compute nodes, we additionally collect information about noise level, which helps us to detect cases when noise becomes notable. For this purpose, a special script was implemented, which regularly runs quick noise measurement in the epilogue of user jobs (see [6] for more details).

There is no need to study service nodes in such detail, however, it is also useful to collect information about them, especially regarding their reliability. That is why we obtain such basic data as server availability using ping command, load average and available disk space. On head

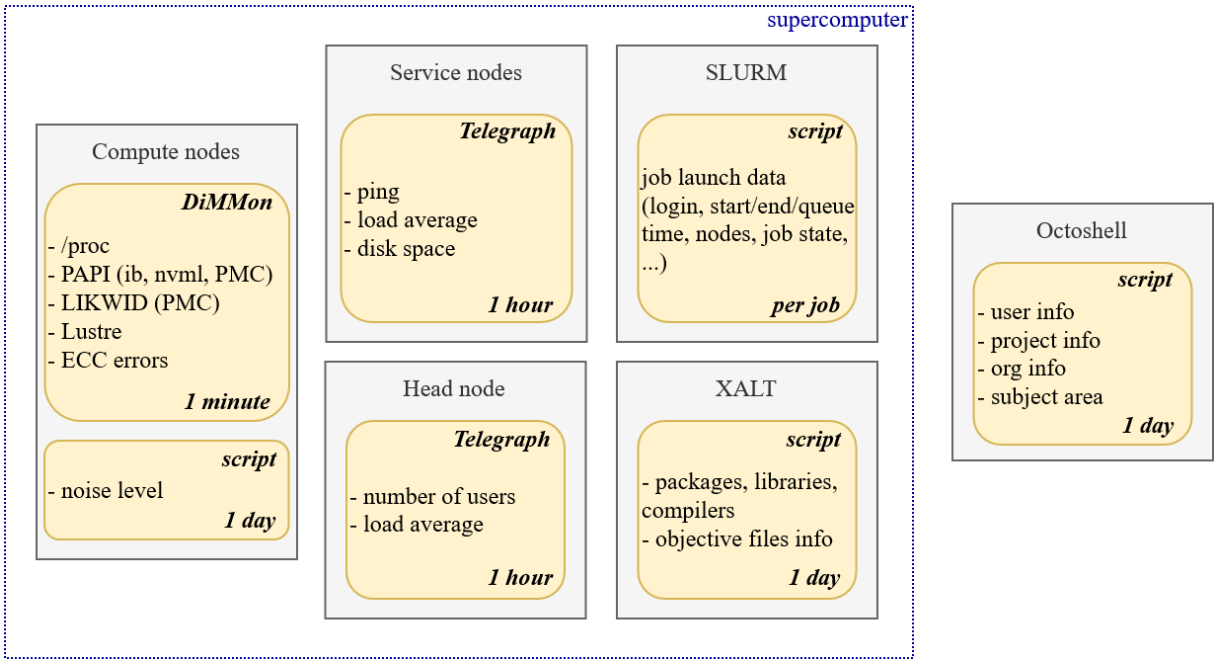


Figure 1. Monitoring data collected on the Lomonosov-2 supercomputer

node, we are interested in the number of active users currently working on Lomonosov-2, as well as in load average.

In order to get a detailed picture of supercomputer behavior, information from compute and service nodes alone is not enough. We additionally collect data from different system software. Of course, SLURM is used to get information about launches of user jobs — who, when and where launched each job, how many nodes were allocated, what the job exit state was, etc. We also use XALT [17] to get useful information about popular software packages (like NAMD, GROMACS, etc.) and compilers being used in jobs.

There is one source of data for TASC that does not run on the supercomputer itself. We regularly request Octoshell [18] (supercomputer center management system developed in RCC MSU) located on a separate server to obtain information about users as well as scientific projects and organizations they work in.

All this variety of data needs to be properly stored, so that TASC analysis modules can easily and quickly retrieve all the information they need. The overall data storage organization scheme is shown in Fig. 2. The main database used to store almost all types of data is a MongoDB database. This DB type was initially chosen because we work with very diverse data, and we also need to be able to easily add new data sources. For this purpose also, we have developed our own unified API, which significantly simplifies this process of importing different types of data in our MongoDB. All the data in this database is never deleted, enabling us to work with all the historical information we want.

The only exception is the main performance-related data from compute nodes. The reason why it is imported differently is the volume of this data. More than 40 characteristics are collected on each compute node using DiMMon, and for each of them minimum, average and maximum value is sent every minute. Taking into account that the number of nodes in the Lomonosov-2 supercomputer is more than 1500, and also considering our desire to store this information for a long time, it is obvious that the main database will very quickly become too large and slow. That

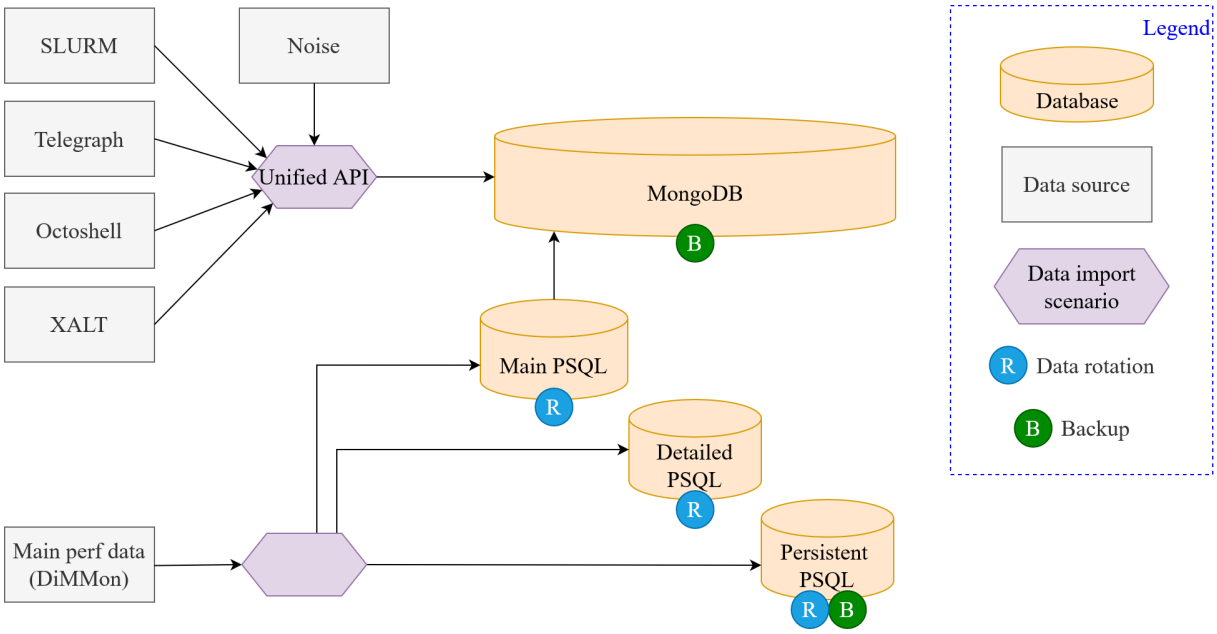


Figure 2. Data storage organization in TASC

is why we decided to use the following approach. All DiMMon data with 1-minute granularity is stored to PostgreSQL database called Main PSQL. This data is regularly aggregated per job and sent to MongoDB. Since this database is used frequently and queries to it must be executed quickly, it can not be very large, so it rotates the data and stores data for the last ∼3 weeks. But we want the data not to disappear so quickly, so it was decided to add two more databases — Detailed and Persistent PSQL. The first one stores the same data, but for a much longer time period (∼5 months). It is rarely used, usually to perform heavy queries involving large amounts of data. The Persistent PSQL stores data almost endlessly (for 3 years now, planned to make it even longer), but since the database would become unacceptably large over such a period of time, the data granularity was reduced from 1 minute to 10 minutes.

The described scheme has proven itself well in practice, allowing to easily deal with large amounts of real-life data.

1.2. Performing Data Analysis

The part of TASC responsible for performance analysis consists of several generally independent solutions, each of which is aimed at carrying out ODA for studying different aspects of supercomputer behavior. The main analysis software tools implemented and used in practice are:

- tool for automatic detection of job-level performance issues;
- web report system of “endless” workload analysis;
- assessment system for evaluating HPC resource efficiency;
- noise measurement tool.

The first solution automatically detects performance issues in all user jobs running on the supercomputer. This is performed using a set of predefined rules; each rule describes one particular issue with job performance and includes issue description, criteria for its automatic detection using monitoring data, supposition of its root cause as well as recommendation on possible further steps for its elimination. On the Lomonosov-2 supercomputer, there are currently ∼30 rules

implemented that are aimed at capturing MPI usage inefficiency, imbalance, usage of wrong partitions, abnormally inefficient jobs, etc. The examples and more detailed description of these rules can be found in [4].

"Endless" workload analysis [5] is a web-based system for HPC system administrators that can provide performance-related reports with a desired level of detail. This flexible tool can show only the needed subset of data for a specified level of consideration, starting from particular job launches and to the HPC system as a whole. The ability to fine-tune both the subset of data being considered and the object of consideration (job launch, user, project, organization, software package, scientific area) generates a tremendous number of possible report variants, which gives this system its "endless" name.

One way to assess the performance of an HPC application is to determine how efficiently it uses resources allocated to it. However, commonly used metrics, which are collected on-the-fly for all applications running on the supercomputer (i.e., without the need to restart the application in profiling, tracing, or debugging mode), usually do not allow estimating this with sufficient accuracy. For example, CPU user load shows processor utilization, but it does not indicate if CPU is busy with useful calculations. Further, common memory-related characteristics like the frequency of cache misses or read/write operations also do not allow one to accurately assess whether memory is being used efficiently. In order to solve this problem, a set of assessments was proposed [1], which automatically calculate accurate measurements of resource usage efficiency (CPU, memory, MPI network, I/O, GPU) for all jobs being executed on the supercomputer. The main idea is to estimate to what extent dealing with a specific resource interferes with useful computations. This allows both identifying applications that show low efficiency and determining directions for further performance analysis using existing profilers, debuggers, etc.

Most modern supercomputers contain so-called noise on compute nodes, which was previously defined as an external influence of the software and hardware environment leading to a change in the behavior of user applications. It is usually quite insignificant, but in some cases it can notably affect the behavior of user applications. In order to assess the noise level, a tool within TASC was proposed and developed [6] that allows regular estimation of the noise level on compute nodes. This tool is based on Netgauge software [19, 20] aimed at measuring OS noise. It starts in the SLURM epilogue after each job completion (but no more than once a day, so as not to impact user applications itself) and provides a noise level metric ranging from 0 (no noise at all) to 100 (maximum possible noise level). This helps, for example, to identify dysfunctional nodes and send them to system administrators for further detailed study and repair.

2. TASC Recent Developments

The main recent developments within TASC are aimed at enhancing one analysis component — the assessment system. In this section, we will describe them as well as provide real-life examples of useful insights obtained using this system.

As mentioned earlier, the assessment system allows for automatic estimation of the efficiency of using various resource types for all jobs running on the supercomputer. Previously, an approach for creating such assessment system was proposed [1], as well as a software implementation of the assessments and its evaluation in practice were performed. This system provides assessment scores for 4 main types of resources (GPU scores were proposed but not implemented due to limitations of currently available hardware): CPU, memory subsystem, MPI network, I/O. Each score estimates resource usage efficiency ranging from 0 (working with this resource did not

hinder useful computations in any way during job execution) to 100 (it constantly hindered useful computations during job execution).

This section describes further work in this direction. An approach to visualizing the collected information on assessments was developed, a method for informing supercomputer users about detected efficiency problems was proposed, and an analysis of the collected statistics on the assessments obtained on the Lomonosov-2 supercomputer was conducted. Also, we implemented a solution for continuous prediction of assessment values in cases when the main method of their calculation is not applicable.

2.1. Dashboards for Assessment Results Visualization

The main goal of this service is to provide a convenient web tool for studying all the main results obtained by the assessment system. It should be noted that it is only available to supercomputer administrators; users receive results from the assessment system in a much smaller volume and in a simpler and more convenient way (see subsection 2.2). This is done, in particular, because excessively detailed information is not required by users and will only confuse them; moreover, users should not have access to information about the performance of other users and projects.

To provide detailed information about the quality of supercomputer resources usage, this web service uses so-called dashboards — web reports dedicated to a certain part of results available in the assessment system. In total, three main dashboards are implemented: 1) general information about resource usage efficiency; 2) detailed information about particular user; 3) detailed information about particular scientific project. Each dashboard can be flexibly configured using different filters, such as specific time period, user, project, organization, used software package, supercomputer partition, number of nodes or node-hours, presence of particular performance issues automatically detected by rules in TASC (briefly described in subsection 1.2). It should be noted that all figures shown below in the paper are screenshots obtained from these dashboards.

The general information dashboard is designed to provide primary insights about what is happening on the supercomputer in general. The dashboard provides the following information:

- integral values of different assessment scores;
- dynamics of change in score values over time;
- distribution of score values for different jobs;
- top worst values for particular jobs, users and projects.

For example, Fig. 3 shows the distribution of $\text{score}_{\text{cpu}}$ (axis X) and $\text{score}_{\text{mem}}$ (axis Y) for individual jobs, where each point corresponds to one job and its color — to different users. Consider, for example, the point in the upper left corner, highlighted by the green circle in the figure. This job has a $\text{score}_{\text{cpu}}$ of 45, which is generally within the normal range, and a $\text{score}_{\text{mem}}$ of 80, which is much higher (i.e., worse) than normal. In this case, we can conclude that the processor often waits for data from memory during the execution of this application, though this is not a significant bottleneck, since the share of useful calculations is still high. The situation is quite different for the application highlighted by the red circle: here, the $\text{score}_{\text{cpu}}$ is above 90, and the $\text{score}_{\text{mem}}$ is slightly above 40. Such a high $\text{score}_{\text{cpu}}$ value indicates serious performance problems, and memory usage is likely one of the reasons for this (although clearly not the only one, since the $\text{score}_{\text{mem}}$ value is not so high). This job definitely requires a more detailed study, as can be concluded based on this graph.

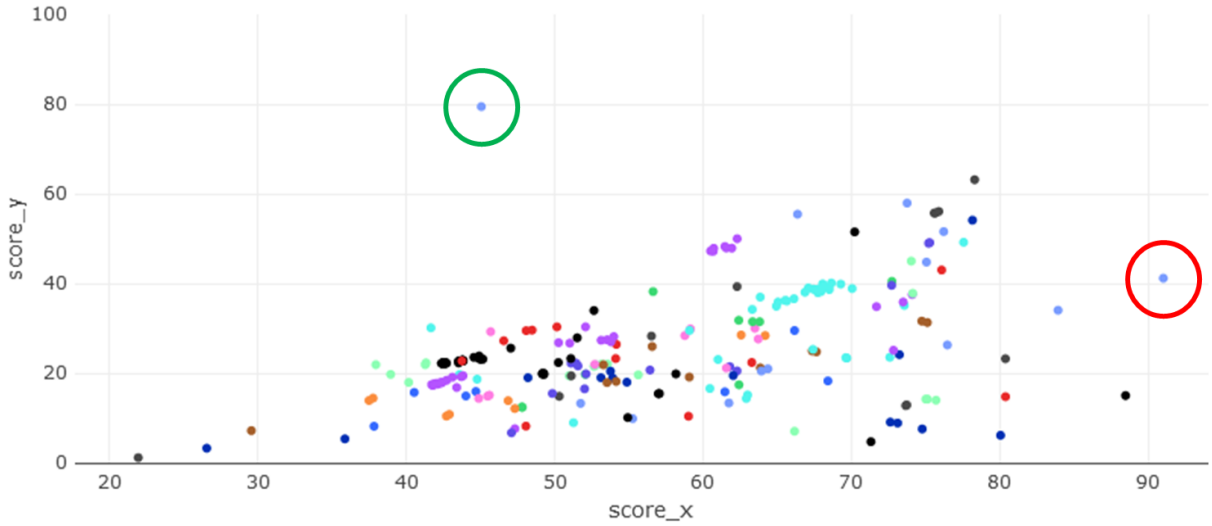


Figure 3. 2D graph showing distribution of values for $\text{score}_{\text{cpu}}$ and $\text{score}_{\text{mem}}$ for individual jobs (screenshot from web dashboard)

The dashboard with detailed information about particular user shows the following information:

- overall user activity: number of job launches and consumed node-hours for the selected time period;
- aggregated score values for the selected period, as well as user rank according to these scores (in general and for the selected software package);
- distribution of score values, separately for each resource type;
- detailed information on each job;
- the same information as for the first dashboard (but filtered for this user only): dynamics of change in score values over time and distribution of score values for different jobs.

As an example, Fig. 4 shows aggregated score values with user ranks (the lower the ranking, the worse). This figure shows that the user has no problems with I/O (the average score is 0, so no I/O-related issues at all, which obviously corresponds to the 1st place in the rating), but the share of useful calculations is not so great — $\text{score}_{\text{cpu}}$ is 67, and this is the 91st place out of 115 (this is the total number of users who launched their tasks during the time period under review). At the same time, note that the average $\text{score}_{\text{mem}}$ value is less than 40, which is generally considered a normal value for supercomputer applications. But the low rank, according to this assessment, suggests that memory usage can be probably optimized, since most of other users show better memory usage efficiency.

These dashboards can be useful for quick but multi-sided primary analysis of resource usage efficiency, starting from the most general level and gradually getting a more detailed picture. Of course, to get conclusions about specific performance issues and ways to deal with them, one need to use specialized software for more detailed performance analysis, but this solution helps to take the first step and understand whether there are issues in general and in what direction to move further during performance analysis.

2.2. Informing Users about Performance Issues

As said earlier, users do not need to be provided with such detailed information. In the vast majority of cases, users are not interested in values of some scores they are not aware of. They

name	value	ranking
score_cpu	66.89	№ 91 / 115
score_mem	35.71	№ 102 / 115
score_mpi	0.62	№ 67 / 115
score_io	0.00	№ 1 / 115

Figure 4. Score values and user rank according to these assessments (screenshot from web dashboard)

want to know whether there are performance issues in their applications and how to fix them. For this purpose, automatic notification of users about such cases based on the results of the assessment system was developed.

This is implemented as follows. Within the framework of TASC, rules were previously proposed that automatically detect certain performance issues. In order to notify users of the Lomonosov-2 supercomputer about these issues, a reporting system was implemented within the Octoshell web-site (used as a "single window" for organizing the work with Lomonosov-2 for all users). Among other, this reporting system shows a list of detected issues for each launched job, along with an assumption about their occurrence and recommendations for their elimination. It was decided to add additional rules to this solution that notify users about issues detected by the assessment system. To do this, it was necessary to draw up criteria for rule triggering, including defining threshold values for each score, which was done both on the basis of our own experience and existing materials [21].

As a result, each user always gets notifications about whether his applications have significant issues with the efficiency of using various resources, and what software can be further used to study these issues in more detail. Figure 5 shows an example of such output for a job. The top record is associated with $\text{score}_{\text{cpu}}$, middle one — with score_{io}, lower one — with $\text{score}_{\text{mem}}$. In all cases, the corresponding rule triggered due to the score being so high that it indicates significant impact on overall job performance, leading to this notification to appear. It should be noted that such notifications appear only if resource usage efficiency was really low, which means that false positives are rare in this case.

2.3. Development of Method for Continuous Assessment Prediction

All assessments within the proposed system are calculated based on monitoring data. Two of them — $\text{score}_{\text{cpu}}$ and $\text{score}_{\text{mem}}$ — require collecting additional data from hardware performance monitoring counters, which leads to overheads due to the usage of multiplexing mode (see [22] for details on estimation of such overheads). Therefore, on the Lomonosov-2 supercomputer this data collection is currently organized only for a part of the jobs (for every third job at this moment, in the future this will likely be increased). In addition, technical problems with the monitoring itself may arise. This leads to the fact that assessments are not available for all jobs. Therefore, the task of predicting scores for the remaining jobs arise.

Description	Supposition	Recommendation
The share of useful CPU computations is small.	Not enough information to determine the reasons for this issue.	Try performing the following type of more detailed analysis: General program analysis VTune Amplifier
The usage of I/O network presumably interferes with useful computations.	It is likely that the file system usage is not organized efficiently.	Try performing the following type of more detailed analysis: I/O analysis VTune Amplifier
Processor are idle for a notable time while waiting for memory operations to be performed.	Memory usage is presumably organized inefficiently and requires optimization.	Try performing the following type of more detailed analysis: Analysis of memory usage efficiency Valgrind (Callgrind)

Figure 5. Examples of user notification about resource usage efficiency being too low, notably interfering with useful computations (screenshot from web dashboard for users)

A general method has been earlier proposed that allows for highly accurate score predictions based on data mining methods [1]. However, this method cannot be used on a regular basis because it is too computationally expensive, in particular because it analyzes data on all previously completed jobs. We propose an adapted, faster version of this method that works as follows:

- The prediction process is started every 30 minutes. At each iteration, a list of jobs that were completed after the previous iteration and the duration of which was more than 30 minutes is extracted (if duration is less, the volume of collected monitoring data is insufficient to accurately assess job behavior).
- The knowledge base is loaded. The knowledge base includes the data on N previous jobs with known score values that were calculated using main method without prediction (we will further talk about selecting the optimal value of N).
- The jobs are traversed from older to newer ones, for each of them the following is performed:
  - Prediction is performed based on static information about program source code and system monitoring data collected during program execution. Prediction is based on searching for similar supercomputer jobs in the knowledge base and subsequent aggregation of assessments for the list of detected similar jobs (this has not changed from the original method, see [1]).
  - If it was possible to predict scores, these assessment values are saved.
  - Also, if the true values of the predicted assessments (obtained from the main method) are known for the current job, the knowledge base is updated in FIFO (first in first out) mode.

One of the main questions that arises here is the selection of the N value, which sets the number of jobs under consideration, on the basis of which the prediction will be made. To solve the problem of selecting N, an experiment was conducted in which the method accuracy was evaluated depending on the value of N. In this experiment, the prediction of scores was carried out for the jobs, for which the true score values (obtained using the main method) were known.

Accuracy was evaluated based on 2 metrics: Mean Absolute Error (MAE) и Mean Squared Error (MSE). Both MSE and MAE are positive numbers, and the lesser the value — the better. The task of predicting assessment values is the regression task, and MAE shows average error of the regression model. MSE squares the error, and only then an average value is computed. This way it ensures that predictions with high error have greater effect on the score, which, in combination with MAE, allows for a more precise accuracy evaluation.

Within the experiment, jobs for several months of the Lomonosov-2 supercomputer operation were studied. We only considered jobs with known assessments that lasted more than 30 minutes. The total number of such jobs was ∼4600. Accuracy comparison based on MAE and MSE metrics showed that the best results are achieved with $N = 1250$. Table 1 shows the corresponding metric values. Note that for assessment scores an error of ±10 is considered acceptable, so the proportion of predictions for which the error was more than 10 is evaluated separately.

Table 1. Accuracy results for the continuous assessment prediction method, $N = 1250$

	% of jobs with predicted values	MAE	MSE	% of jobs with AE>10
$\text{score}_{\text{cpu}}$	81.39	2.18	24.83	3.68
$\text{score}_{\text{mem}}$	81.41	2.53	32.59	4.51

It is worth noting that the proposed method for continuous prediction can predict 7% fewer jobs than the original method. However, the proportion of jobs for which scores were predicted, as well as the accuracy of these predictions, are quite high. For example, the average error is less than 3, and the proportion of notable errors (greater than 10) does not exceed 5%.

The proposed method was implemented and is used on a regular basis in everyday practice on the Lomonosov-2 supercomputer. It should be also noted that if both the true and predicted scores (calculated using main and proposed methods, respectively) are available for a certain job, then the true value is used for further analysis as the more accurate one. However, the predicted value is also saved; this will be useful, in particular, for subsequent accuracy evaluations of the prediction method in the future.

We also evaluated the performance of the proposed method on the same experiment data as described earlier. This evaluation was performed on a single Intel Xeon Silver 4214 (1 core was used) with 8 GB RAM (usually less than 2 GB was used). It showed that on average it takes only 1.6 seconds to predict estimates for one job, which is orders of magnitude faster than current real-life needs of the Lomonosov-2 supercomputer.

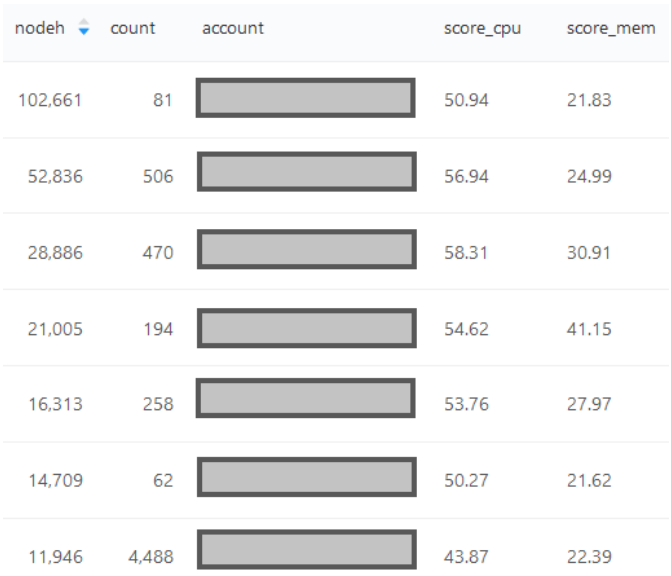
3. Real-life Application of the Assessment System

In this section, we will describe several examples obtained on real-life data from the Lomonosov-2 supercomputer to demonstrate practical application of the proposed assessment system. All these examples were found out using dashboards described earlier.

We have studied a list of individual job launches with the highest $\text{score}_{\text{cpu}}$ scores for May-June of this year. Of interest were the jobs with high $\text{score}_{\text{cpu}}$ combined with high CPU user load — these are the cases when the processor is actively utilized, but at the same time the share of useful calculations is small. As a result, we found launches by two users from the same project with an average CPU load of about 95% (HyperThreading is enabled on most nodes of the Lomonosov-2 supercomputer, so such a load means that not only all physical, but also all logical cores were almost completely utilized in these programs), while the $\text{score}_{\text{cpu}}$ was ∼85

(which means that the processor was fully occupied by useful calculations only 15% of the time during job execution).

Further analysis of these jobs showed that the GROMACS package [23] was used in all cases. So we analyzed the table showing the most active GROMACS users within the year of 2024 (Fig. 6). For each user, this table shows total amount of consumed node-hours with GROMACS launches, number of job launches, as well as average values of $\text{score}_{\text{cpu}}$ and $\text{score}_{\text{mem}}$. It is clear from this table that the $\text{score}_{\text{cpu}}$ assessments are usually much better (less than 60, which is typically OK for HPC programs), meaning that $\text{score}_{\text{cpu}}$ equal to 85 is not typical for this package. Moreover, jobs showing high CPU load together with high $\text{score}_{\text{cpu}}$ appeared for these two GROMACS users only.



nodeh	count	account	score_cpu	score_mem
102,661	81		50.94	21.83
52,836	506		56.94	24.99
28,886	470		58.31	30.91
21,005	194		54.62	41.15
16,313	258		53.76	27.97
14,709	62		50.27	21.62
11,946	4,488		43.87	22.39

Figure 6. Top users of GROMACS software packages sorted by consumed node-hours (screenshot from web dashboard described in 2.1)

It was decided to take a more detailed look at the list of all last GROMACS package launches by these two users. It turned out that they each have several such launches, and these launches are notably different from the rest, since the activity of using other resources (except for the CPU) in them is noticeably lower. In particular, they show very low GPU load at less than 2%, while these users (and other users as well) usually launch GROMACS jobs that show much higher GPU load — up to 50% and higher. Other performance indicators like frequency of cache misses or amount of MPI data sent were also significantly lower. Based on this information, we can conclude that most likely these job launches for some reason did not execute properly. The exact reason can not be determined using the assessment system only, since its purpose is to provide a quick view of whether resources are being utilized efficiently or not. But further analysis, performed by other parts of TASC ("endless" report system and noise measurement tool), showed that these jobs were executed on two noisy compute nodes that were misbehaving at that moment of time. These nodes were sent to repair, and the problem was solved.

This information is surely of interest for both users and supercomputer administrators. It should be noted that these jobs in general showed "normal" performance indicators, did not fail and even completed with a successful finish state, meaning that it would be difficult to determine the root causes of these performance issues without the assessment system.

Let us switch to another example. We considered launches of the popular LAMMPS software package [24] for a couple of months on the Lomonosov-2 supercomputer. Sorting the jobs by the maximum $\text{score}_{\text{mpi}}$ value showed that the top 35 job launches were all performed by 3 users of the same project. This being said, the MPI network usage activity in these jobs was far from the highest — in many cases less than 200 MB/sec on average, while for other users of this package it is more than 1.5 GB/sec. Further study of these jobs with unusually high $\text{score}_{\text{mpi}}$ showed that almost all of them had two performance issues — too small average size of MPI packages, as well as too low network locality (compute nodes allocated for the job were located far from each other). The first issue often leads to additional overheads due to the need to transfer system data and due to higher impact of latency, the second one leads to increased latency. The presence of these issues can be noticeable, which seems to be the case here due to the high $\text{score}_{\text{mpi}}$. These cases should be reported to users so that they can be aware of such situations and try, if necessary, to optimize data transfer in these applications.

Here is another useful, though somewhat unexpected result obtained using the assessment system. Studying the list of individual job launches with the highest $\text{score}_{\text{mem}}$ values, it was found that almost all jobs with a score above 80 (which corresponds to CPU almost constantly stalled waiting data from memory) have a distinctive feature — only one of the nodes allocated to such job is actively used during execution. For example, a job with 45 allocated nodes was found, and the average CPU user load on 44 of them was less than 1% (the 45th node shows fairly high activity). All other such jobs behave similarly. In this case, it can be assumed that either users incorrectly submitted these jobs (for example, several nodes were requested while starting a single-node program), or some technical issues occurred during job launch process. It is not yet entirely clear why this behavior results in extremely high $\text{scores}_{\text{mem}}$, but detecting such incorrect launches is certainly useful, and measures will be taken to promptly identify and fix such situations.

Conclusions

This paper describes the current state and functionality of the TASC software suite. It mentions the data types being collected and analyzed, as well as how this variety of information is stored. The main analytical TASC capabilities are also briefly presented: 1) tool for automatic detection of job-level performance issues for all jobs running on a supercomputer; 2) "endless" workload analysis solution for detailed study of supercomputer performance with a needed level of detail; 3) assessment system for initial, but quick and accurate evaluation of HPC resource efficiency; 4) system for regular measurement of noise level on compute nodes.

The second part of the paper is devoted to the description of recent developments and updates within TASC, which are all related to the assessment system. An approach to visualizing the collected information on assessments was developed, a method for automatic informing supercomputer users about detected performance issues was proposed, and an analysis of the collected statistics on the assessments obtained on the Lomonosov-2 supercomputer was conducted. Also, we proposed a solution for continuous prediction of assessment values, which demonstrates high accuracy and performance.

The results shown in Section 2 were achieved at Lomonosov Moscow State University with the financial support of the Russian Science Foundation, agreement No. 21-71-30003. The research is carried out using the equipment of shared research facilities of HPC computing resources at Lomonosov Moscow State University.

References

1. Voevodin V.V., Shaikhislamov D.I., Nikitenko D.A. How to assess the quality of supercomputer resource usage. Supercomputing Frontiers and Innovations. 2022. Vol. 9, no. 3. P. 4–18. DOI: 10.14529/jsfi220301.
2. High Performance Computing Market Size to Surpass USD 64.65. URL: https://www.globenewswire.com/news-release/2022/04/04/2415844/0/en/High-Performance-Computing-Market-Size-to-Surpass-USD-64-65-Bn-by-2030.html (accessed: 14.08.2024).
3. High Performance Computing Market Size, Growth Report. URL: https://www.fortunebusinessinsights.com/industry-reports/high-performance-computing-hpc-and-high-performance-data-analytics-hpda-market-100636 (accessed: 14.08.2024).
4. Shvets P., Voevodin V., Zhumatiy S. Primary automatic analysis of the entire flow of supercomputer applications. CEUR Workshop Proceedings. 2018. P. 20–32.
5. Shvets P., Voevodin V. "Endless" Workload Analysis of Large-Scale Supercomputers. Lobachevskii Journal of Mathematics. 2021. Vol. 42. P. 184–194. DOI: 10.1134/s1995080221010236.
6. Voevodin V.V., Nikitenko D.A. Recurrent Monitoring of Supercomputer Noise. Supercomputing Frontiers and Innovations. 2023. Vol. 10, no. 3. P. 27–35. DOI: 10.14529/jsfi230304.
7. Jones M.D., White J.P., Innus M., *et al.* Workload Analysis of Blue Waters. 2017. DOI: 10.48550/arXiv.1703.00924. arXiv: 1703.00924.
8. Simakov N.A., White J.P., DeLeon R.L., *et al.* A Workload Analysis of NSF's Innovative HPC Resources Using XDMoD. 2018. DOI: 10.48550/arXiv.1801.04306. arXiv: 1801.04306.
9. Hart D.L. Measuring TeraGrid: workload characterization for a high-performance computing federation. The International Journal of High Performance Computing Applications. 2011. Nov. Vol. 25, no. 4. P. 451–465. DOI: 10.1177/1094342010394382.
10. Patel T., Liu Z., Kettimuthu R., *et al.* Job characteristics on large-scale systems: long-term analysis, quantification, and implications. SC20: International conference for high performance computing, networking, storage and analysis. IEEE, 2020. P. 1–17. DOI: 10.1109/SC41405.2020.00088.
11. Kostenetskiy P., Shamsutdinov A., Chulkevich R., *et al.* HPC TaskMaster- Task Efficiency Monitoring System for the Supercomputer Center. International Conference on Parallel Computational Technologies. Springer, 2022. P. 17–29. DOI: 10.1007/978-3-031-11623-0_2.
12. Isakov M., Del Rosario E., Madireddy S., *et al.* HPC I/O throughput bottleneck analysis with explainable local models. SC20: International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, 2020. P. 1–13. DOI: 10.1109/SC41405.2020.00037.

13. Netti A., Shin W., Ott M., *et al.* A conceptual framework for HPC operational data analytics. 2021 IEEE International Conference on Cluster Computing (CLUSTER). IEEE, 2021. P. 596–603. DOI: 10.1109/Cluster48925.2021.00086.

14. Ott M., Shin W., Bourassa N., *et al.* Global experiences with HPC operational data measurement, collection and analysis. 2020 IEEE International Conference on Cluster Computing (CLUSTER). IEEE, 2020. P. 499–508. DOI: 10.1109/CLUSTER49012.2020.00071.

15. 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. 2019. Vol. 6, no. 2. P. 4–11. DOI: 10.14529/jsfi190201.

16. Stefanov K., Voevodin V., Zhumatiy S., Voevodin V. Dynamically reconfigurable distributed modular monitoring system for supercomputers (DiMMon). Procedia Computer Science. 2015. Vol. 66. P. 625–634. DOI: 10.1016/j.procs.2015.11.071.

17. Agrawal K., Fahey M.R., McLay R., James D. User Environment Tracking and Problem Detection with XALT. 2014 First International Workshop on HPC User Support Tools. IEEE, Nov. 2014. P. 32–40. DOI: 10.1109/HUST.2014.6.

18. Nikitenko D., Zhumatiy S., Paokin A., *et al.* Evolution of the Octoshell HPC center management system. International Conference on Parallel Computational Technologies. Springer, 2019. P. 19–33. DOI: 10.1007/978-3-030-28163-2_2.

19. Hoefler T., Mehlan T., Lumsdaine A., Rehm W. Netgauge: A network performance measurement framework. International Conference on High Performance Computing and Communications. Springer, 2007. P. 659–671.

20. Netgauge - Operating System Noise Measurement. URL: https://htor.inf.ethz.ch/research/netgauge/osnoise/ (accessed: 25.09.2024).

21. Top-down Microarchitecture Analysis Method. URL: https://www.intel.com/content/www/us/en/docs/vtune-profiler/cookbook/2023-0/top-down-microarchitecture-analysis-method.html#GUID-FEA77CD8-F9F1-446A-8102-07D3234CDB68 (accessed: 14.08.2024).

22. Voevodin V., Stefanov K., Zhumatiy S. Overhead analysis for performance monitoring counters multiplexing. Russian Supercomputing Days. Springer, 2022. P. 461–474. DOI: 10.1007/978-3-031-22941-1_34.

23. Abraham M.J., Murtola T., Schulz R., *et al.* GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. SoftwareX. 2015. Sept. Vol. 1–2. P. 19–25. DOI: 10.1016/j.softx.2015.06.001.

24. Thompson A.P., Aktulga H.M., Berger R., *et al.* LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. Comp. Phys. Comm. 2022. Vol. 271. P. 108171. DOI: 10.1016/j.cpc.2021.108171.

УДК 519.68 DOI: 10.14529/cmse240304

ПРОГРАММНЫЙ КОМПЛЕКС TASC ДЛЯ АНАЛИЗА ПРОИЗВОДИТЕЛЬНОСТИ СУПЕРКОМПЬЮТЕРОВ: ТЕКУЩЕЕ СОСТОЯНИЕ И ПОСЛЕДНИЕ РАЗРАБОТКИ

© 2024 В.В. Воеводин, Д.И. Шайхисламов, В.А. Серов

Московский государственный университет имени М.В. Ломоносова

(119991 Москва, Ленинские горы, д. 1)

E-mail: vadim@parallel.ru, sdenis1995@gmail.com, sva@srcc.msu.ru

Поступила в редакцию: 04.09.2024

Для обеспечения высокой эффективности работы современных суперкомпьютеров необходимо постоянно анализировать и контролировать различные аспекты их функционирования, уделяя особое внимание изучению потока суперкомпьютерных приложений, выполняющихся на таких системах. Для решения этой задачи ранее был разработан программный комплекс TASC (Tuning Applications for SuperComputers). Он автоматически обнаруживает проблемы с производительностью в суперкомпьютерных приложениях и оценивает эффективность использования ресурсов суперкомпьютера, предоставляет администраторам гибкий инструмент создания отчетов для анализа различных аспектов работы суперкомпьютера с требуемым уровнем детализации, а также оценивает уровень шума на вычислительных узлах. В данной работе приведено детальное описание текущей структуры и возможностей TASC, включая этапы обработки и хранения данных, а также выполнения различных видов анализа. Также описаны новые полученные результаты и разработанные методы в рамках одного из основных компонентов TASC — системы оценок для быстрого и точного определения эффективности использования суперкомпьютерных ресурсов.

Ключевые слова: высокопроизводительные вычисления, суперкомпьютер, анализ производительности, анализ качества работы, анализ операционных данных, мониторинг.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Voevodin V.V., Shaikhislamov D.I., Serov V.A. TASC Software for HPC Performance Analysis: Current State and Latest Developments // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2024. Т. 13, № 3. С. 61–78. DOI: 10.14529/cmse240304.

Литература

1. Voevodin V.V., Shaikhislamov D.I., Nikitenko D.A. How to assess the quality of supercomputer resource usage // Supercomputing Frontiers and Innovations. 2022. Vol. 9, no. 3. P. 4–18. DOI: 10.14529/jsfi220301.
2. High Performance Computing Market Size to Surpass USD 64.65. URL: https://www.globenewswire.com/news-release/2022/04/04/2415844/0/en/High-Performance-Computing-Market-Size-to-Surpass-USD-64-65-Bn-by-2030.html (дата обращения: 14.08.2024).
3. High Performance Computing Market Size, Growth Report. URL: https://www.fortunebusinessinsights.com/industry-reports/high-performance-computing-hpc-and-high-performance-data-analytics-hpda-market-100636 (дата обращения: 14.08.2024).

4. Shvets P., Voevodin V., Zhumatiy S. Primary automatic analysis of the entire flow of supercomputer applications // CEUR Workshop Proceedings. 2018. P. 20–32.

5. Shvets P., Voevodin V. "Endless" Workload Analysis of Large-Scale Supercomputers // Lobachevskii Journal of Mathematics. 2021. Vol. 42. P. 184–194. DOI: 10.1134/s1995080221010236.

6. Voevodin V.V., Nikitenko D.A. Recurrent Monitoring of Supercomputer Noise // Supercomputing Frontiers and Innovations. 2023. Vol. 10, no. 3. P. 27–35. DOI: 10.14529/jsfi230304.

7. Jones M.D., White J.P., Innus M., *et al.* Workload Analysis of Blue Waters. 2017. DOI: 10.48550/arXiv.1703.00924. arXiv: 1703.00924.

8. Simakov N.A., White J.P., DeLeon R.L., *et al.* A Workload Analysis of NSF's Innovative HPC Resources Using XDMoD. 2018. DOI: 10.48550/arXiv.1801.04306. arXiv: 1801.04306.

9. Hart D.L. Measuring TeraGrid: workload characterization for a high-performance computing federation // The International Journal of High Performance Computing Applications. 2011. Nov. Vol. 25, no. 4. P. 451–465. DOI: 10.1177/1094342010394382.

10. Patel T., Liu Z., Kettimuthu R., *et al.* Job characteristics on large-scale systems: long-term analysis, quantification, and implications // SC20: International conference for high performance computing, networking, storage and analysis. IEEE, 2020. P. 1–17. DOI: 10.1109/SC41405.2020.00088.

11. Kostenetskiy P., Shamsutdinov A., Chulkevich R., *et al.* HPC TaskMaster- Task Efficiency Monitoring System for the Supercomputer Center // International Conference on Parallel Computational Technologies. Springer, 2022. P. 17–29. DOI: 10.1007/978-3-031-11623-0_2.

12. Isakov M., Del Rosario E., Madireddy S., *et al.* HPC I/O throughput bottleneck analysis with explainable local models // SC20: International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, 2020. P. 1–13. DOI: 10.1109/SC41405.2020.00037.

13. Netti A., Shin W., Ott M., *et al.* A conceptual framework for HPC operational data analytics // 2021 IEEE International Conference on Cluster Computing (CLUSTER). IEEE, 2021. P. 596–603. DOI: 10.1109/Cluster48925.2021.00086.

14. Ott M., Shin W., Bourassa N., *et al.* Global experiences with HPC operational data measurement, collection and analysis // 2020 IEEE International Conference on Cluster Computing (CLUSTER). 2020. P. 499–508. DOI: 10.1109/CLUSTER49012.2020.00071.

15. 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. 2019. Vol. 6, no. 2. P. 4–11. DOI: 10.14529/jsfi190201.

16. Stefanov K., Voevodin V., Zhumatiy S., Voevodin V. Dynamically reconfigurable distributed modular monitoring system for supercomputers (DiMMon) // Procedia Computer Science. 2015. Vol. 66. P. 625–634. DOI: 10.1016/j.procs.2015.11.071.

17. Agrawal K., Fahey M.R., McLay R., James D. User Environment Tracking and Problem Detection with XALT // 2014 First International Workshop on HPC User Support Tools. IEEE, Nov. 2014. P. 32–40. DOI: 10.1109/HUST.2014.6.

18. Nikitenko D., Zhumatiy S., Paokin A., *et al.* Evolution of the Octoshell HPC center management system // International Conference on Parallel Computational Technologies. Springer, 2019. P. 19–33. DOI: 10.1007/978-3-030-28163-2_2.

19. Hoefler T., Mehlan T., Lumsdaine A., Rehm W. Netgauge: A network performance measurement framework // International Conference on High Performance Computing and Communications. Springer, 2007. P. 659–671.

20. Netgauge - Operating System Noise Measurement. URL: https://htor.inf.ethz.ch/research/netgauge/osnoise/ (дата обращения: 25.09.2024).

21. Top-down Microarchitecture Analysis Method. URL: https://www.intel.com/content/www/us/en/docs/vtune-profiler/cookbook/2023-0/top-down-microarchitecture-analysis-method.html#GUID-FEA77CD8-F9F1-446A-8102-07D3234CDB68 (дата обращения: 14.08.2024).

22. Voevodin V., Stefanov K., Zhumatiy S. Overhead analysis for performance monitoring counters multiplexing // Russian Supercomputing Days. Springer, 2022. P. 461–474. DOI: 10.1007/978-3-031-22941-1_34.

23. Abraham M.J., Murtola T., Schulz R., *et al.* GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers // SoftwareX. 2015. Sept. Vol. 1–2. P. 19–25. DOI: 10.1016/j.softx.2015.06.001.

24. Thompson A.P., Aktulga H.M., Berger R., *et al.* LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales // Comp. Phys. Comm. 2022. Vol. 271. P. 108171. DOI: 10.1016/j.cpc.2021.108171.

Воеводин Вадим Владимирович, к.ф.-м.н., заведующий лабораторией, Научно-исследовательский вычислительный центр, Московский государственный университет имени М.В. Ломоносова (Москва, Российская Федерация)

Шайхисламов Денис Ильгизович, техник, Научно-исследовательский вычислительный центр, Московский государственный университет имени М.В. Ломоносова (Москва, Российская Федерация)

Серов Владимир Александрович, научный сотрудник, Научно-исследовательский вычислительный центр, Московский государственный университет имени М.В. Ломоносова (Москва, Российская Федерация)