doi 10.26089/NumMet.v22r102

DEVELOPING A MODEL FOR HOLISTIC WORKLOAD ANALYSIS OF LARGE SUPERCOMPUTER SYSTEMS

P. A. Shvets[1], V. V. Voevodin[2,3], and S. A. Zhumatiy[4]

Any modern supercomputer has an extremely complex architecture, and efficient usage of its resources is often a very difficult task, even for experienced users. At the same time, the field of high-performance computing is becoming more and more in demand, so the issue of efficient utilization of supercomputers is very urgent. Therefore, users should know everything important about performance of their jobs running on a supercomputer in order to be able to optimize them, and administrators should be able to monitor and analyze all the nuances of the efficient functioning of such systems. However, there is currently no complete understanding of what data are best to be studied (and how it should be analyzed) in order to have a whole picture of the state of the supercomputer and the processes taking place there. In this paper, we make our first attempt to answer this question. To do this, we are developing a model that describes all the potential factors that may be important when analyzing the performance of supercomputer applications and the HPC system as a whole. The paper provides both a detailed description of this model for users and administrators and some interesting real-life examples discovered on the Lomonosov-2 supercomputer using a software implementation based on the proposed model.

Keywords: high-performance computing, supercomputer, workload analysis, application performance, model development.

1. Introduction. In the field of modern high-performance computing, the paradox of supercomputer efficiency can be observed. The point of this paradox is that supercomputers seem to work efficiently, but in reality this is not entirely true. Let us consider this problem in more detail.

The area of supercomputing is becoming more and more in demand [1] . The reason is that solving of an increasing number of scientific tasks requires computationally expensive experiments. For these purposes, cloud computing or servers are often not suitable, and that is when supercomputers come to the fore. Therefore, an increasing number of specialists from various scientific fields (such as astrophysics, genomic research, nanotechnology, big data analysis and artificial intelligence, as well as many, many others) are beginning to use supercomputer resources. As a result, the demand for supercomputers is growing. This leads to the situation that all available HPC resources are constantly occupied. Moreover, users often have to queue long enough, waiting for their turn to start their jobs. For example, the average waiting time in the main queue of the Lomonosov-2 supercomputer [2] in 2020 (until mid-December) was 20 hours. Moreover, during the same period of time, the average utilization of the supercomputer — the average share of occupied compute nodes (on which jobs are running) among all available ones — is very high and equals to 97%. In this case, we can say that available supercomputer resources are used very efficiently.

However, things are not that simple. If we consider the situation in more detail and look in a different way at the concept of "efficiency", it turns out that many jobs show rather low performance, efficiently utilizing only a small part of the resources allocated to them [3]. And such a picture, in our experience, is inherent in many modern large HPC systems. There can be a variety of reasons for this — inefficient implementation of the program itself, incorrectly chosen launch parameters, failures of the supercomputer hardware, suboptimal

[1] Lomonosov Moscow State University, Research Computing Center; Leninskie Gory, Moscow, 119991, Russia; Programmer, e-mail: shvets.pavel.srcc@gmail.com

[2] Lomonosov Moscow State University, Research Computing Center; Leninskie Gory, Moscow, 119991, Russia; Ph.D., Senior Scientist, e-mail: vadim@parallel.ru

[3] Moscow Center of Fundamental and Applied Mathematics; Leninskie Gory, Moscow, 119991, Russia

[4] Lomonosov Moscow State University, Research Computing Center; Leninskie Gory, Moscow, 119991, Russia; Ph.D., Leading Scientist, e-mail: serg@parallel.ru

quotas and resource usage policies, inefficient or incorrect configuration of installed application packages, and so on. Thus, if you look not only at the node occupancy and the state of the job queue, it becomes clear that useful utilization of a supercomputer is in fact not so high. And we only mentioned the issues related to the efficiency of the applications being executed, but the overall performance of the supercomputer functioning may decrease for other reasons. For example, some of the jobs complete incorrectly (even if their performance is high), which means that the result is not reliable and the computing resources are wasted. Furthermore, sometimes not all compute nodes are available (for example, due to the need for repairs or software updates). Another aspect is also important — how often the queued jobs do not start at all due to some errors? All this, like many other things, can significantly reduce the quality of the supercomputer functioning.

Thus, it becomes clear that the question of efficient operation of supercomputer centers (SC) is very difficult. In order to be able to understand this multitude of causes and their effects, system administrators need to have complete information about the state and behavior of a supercomputer. However, the problem is that there is probably not a single administrator at any large modern supercomputer center who is able to collect all the needed data in practice. A similar problem is observed among users who need detailed (and conveniently presented) information about the properties and behavior of their applications in order to quickly conduct a comprehensive analysis of their application performance. And one of the difficulties in solving this problem is that there is still no clear understanding of what kind of data need to be studied in order to have such a complete picture.

The work presented in this paper is aimed at solving this problem. We develop a general model, the goal of which is to take into account all the potential needs of users and administrators when analyzing the efficiency of individual applications and the functioning of supercomputer centers in general. The software implementation based on such a model will make it possible to determine all the most important and useful scenarios when conducting such a study, which will help to conveniently carry out a holistic workload analysis of a supercomputer. It should be noted that currently there seems to be no reasonable way to prove the completeness of the proposed model, however, in our opinion, this model and its implementation take into account all the main points, and there is nothing else important to be included in it. In this paper we focus only on the issues concerning computing infrastructure of a supercomputer, but the proposed model can be extended to consider the engineering infrastructure as well.

There are many existing works devoted to study of certain aspects related to the quality of HPC systems functioning. Particular attention is paid to the detailed performance analysis of individual applications, for which various profilers (for example, Intel VTune [4] or Valgrind [5]), trace analysis tools for parallel programs (Scalasca [6], Vampir [7], etc) and debuggers (such as ARM Forge [8], TotalView [9]) have been developed. And rightly so, because it is very difficult to write a highly efficient parallel application, and poor application performance is probably the most important factor affecting the performance of a supercomputer. However, this does not give an idea of the general situation as a whole. Next, there is a number of works aimed particularly at supercomputer workload analysis. There are works devoted to a detailed and versatile analysis of not a single system [10], but also a number of systems at once [11, 12]. However, despite their high level of detail, they do not raise issues of completeness when conducting such an analysis. So, at the moment we do not know any related research that allows us to solve the previously posed problem.

The main contribution of this work is the development of a model aimed at describing all the potential factors that may be important when analyzing the performance of supercomputer applications in particular and workload of HPC systems as a whole. This model is divided into two parts, according to the different needs of two groups of people of interest — users and administrators of a supercomputer. The obtained solution, although it is mostly theoretical, can be useful in practice, for example, to ensure the completeness of various software solutions aimed at monitoring and analyzing the quality of SC functioning.

The rest of the paper is organized as follows. Section 2 describes theoretical aspects of developing the model for supercomputer users and administrators. Section 3 shows the details on using such a model in practice. Section 4 summarizes the results achieved in this work.

2. Development of a model for workload analysis of supercomputer systems. In order to start developing a general model, it is necessary to first determine what is the central subject of consideration (what is the main goal when analyzing the quality of the supercomputer). This varies for different groups of people. Three main groups can be distinguished: users, administrators, and SC management. Since the goals and objectives for these groups are different, it is worth considering them separately. In this case the overall model splits into three components, one for each group. At the same time, different models may well overlap: for example, it is important for an administrator to take into account not only the state of the supercomputer as

a whole, but also the interests of users, while the management needs to take into account issues of interest to both other groups. In this paper, we will look at models for a user and an administrator. It should be also noted that in this work we do not touch upon issues related to the operation of the engineering infrastructure of a supercomputer, focusing only at computing infrastructure.

The general process of developing the model for these groups is the same and consists of the following main steps:

- First, global goals for the selected group should be defined: what target characteristics can be identified? What is ultimately the most important to people in this group?
- Next, it is necessary to determine what factors describing the behavior of applications as well as software and hardware environment of the supercomputer can affect the values of the selected target characteristics. The most important factors should be then selected (which should be considered in the first place).
- It is then necessary to determine how it is possible to analyze the influence of the selected factors on the target characteristics in practice.

Therefore, this model consists of the following elements: 1) target characteristics — global goals, the achievement of which is of most importance for each group; 2) factors related to properties of the supercomputer job flow and the operation of the supercomputer software and hardware that can affect the achievement of these goals; 3) specific ways (in terms of particular data presentation) to analyze the impact of these factors in practice.

The first two elements are largely universal and apply to most modern supercomputers. However, some of the factors as well as the choice of the most important factors may vary, depending, for example, on the general purpose of the supercomputer and the objectives of the analysis. The last element depends to the greatest extent on the properties of the target supercomputer, since it relies on a specific list and properties of the collected data on its functioning, and also because it is based on the selected most important factors.

It is also worth noting the following. The first two elements of the model are mostly machine-independent and describe theoretically possible cases that are interesting in principle when performing workload analysis. But during the implementation of step 3 of model development for a specific supercomputer (i.e. determining ways to analyze the impact of factors in practice), some of the data necessary for such an analysis may be not be available. This simply means that it will be impossible to analyze certain aspects of this system operation. And study of the proposed model will help understand what other data is necessary to perform more complete supercomputer workload analysis.

At present there is no quantitative estimates of the target characteristics, but it will be hopefully realized in the future.

In this paper, we focus more on the first two steps of the general model development process. However, some details concerning step 3 will be given in Section 3.1 in connection with the Lomonosov-2 supercomputer.

2.1. Developing a model for the user. From the user's point of view, the most important goal for him (when working on a supercomputer) is his scientific tasks to be completed as quickly as possible, and nothing should prevent them from performing successfully. Therefore, when performing the SC workload analysis, the most important thing for the user, in our opinion, is the **quality and successfulness of his tasks solving**. This is the target characteristics in that case.

Every user solves their tasks by running needed amount of supercomputer jobs. Let us start by considering one job, and then move on to studying the whole set of user jobs.

In this case, by the quality of task solving we mean how quickly and with what amount of resources a supercomputer job (which solves specific task) is executed. The successfulness of task solving will be defined as follows: if the corresponding job completes incorrectly (with an incorrect job exit state and / or received an incorrect result), then the task solving is considered as unsuccessful; it is successful otherwise. For each target characteristic, it is necessary to assess factors influencing these characteristics. That is, what prevents from or helps to achieve successfulness more often and / or greater quality of task solving? This is of the greatest interest for the analysis.

The process of job consideration begins from the moment it is placed in the queue. First, let us determine what we know about the job at the very first moment, i.e. define what is the fixed input that the supercomputing environment cannot influence in any way:

1) time when job is placed in a queue;
2) user account, as well as their affiliation to the project, organization and subject area;

3) used application packages (by an application package we mean a software package or library for solving tasks in a certain subject area, for example GROMACS [13], NAMD [14] or VASP [15]);
4) the used compiler and its options;
5) the job launch command (including executable name, launch directory etc);
6) the number (or range) of requested nodes and/or cores;
7) the specified time limit;
8) the selected supercomputer partition;
9) the list of selected nodes (if was specified by user).

It should be noted that we also consider the supercomputer hardware (number of nodes installed, how they are connected to each other, etc.) as fixed and therefore unchangeable. As we have said, the performance and availability of this hardware can vary.

The process of running a job can be divided into two stages: 1) before the actual start of the execution (i.e., job waiting in the queue); 2) job execution. Next, we will act in accordance with the general process of developing a model as shown at the beginning of Section 2 — we will determine the target characteristics at each of the two stages and define the factors influencing them, from which we will select the most important ones.

The stage before job execution includes placing the job in the queue and waiting for its launch. In this case the target characteristics are the following:

1. **Quality of task solving** is determined by the job start time — i.e. how long the job has to wait in a queue.
2. **Successfulness of task solving** is determined by whether all the prerequisites for correct job launch have been met.

Below is a complete (in our opinion) list of factors affecting the specified target characteristics. All the factors are highlighted in italics, and the most important ones are highlighted in bold. Note that both quality and successfulness can depend on the properties of the job itself and on the properties of the queue in which the job is placed; therefore, the factors influencing them are divided into two corresponding groups.

We have identified the following factors at the first stage of the job life cycle:

1. Quality of task solving (at this stage it is determined by how long the job waited in the queue):
   1.1. Properties of the job itself:
      1.1.1. *Time when job was placed in the queue,* ***number (or range) of requested nodes and/or cores, specified time limit, selected supercomputer partition****, list of selected nodes (if specified).*
      1.1.2. *Job priority in the queue.*
   1.2. Properties of the queue:
      1.2.1. ***Number of jobs in the queue****, queue limits and quotas.*
2. Successfulness of task solving (at this stage is determined by the successfulness of job launch):
   2.1. Properties of the job itself:
      2.1.1. *Exceeding the allowed limits: number of requested nodes / cores, specified time limit, list of requested nodes (if specified).*
   2.2. Properties of the queue:
      2.2.1. *Exceeded quotas per user (for example, the maximum number of all/running jobs or the number of nodes per job).*

Considering the target characteristic of the quality of task solving (that is, the waiting time in the queue), the most important factors are: the number or range of requested cores and nodes per job, the specified time limit for the job execution, the selected supercomputer partition, and, of course, the number of jobs in the queue. It is not so interesting to analyze successfulness at this stage (since when a job is queued, it is usually immediately clear whether it meets all requirements for successful launch or not), therefore, for the second target characteristic, at this stage, not a single important characteristic has been highlighted.

At the second stage, the stage of job execution, the target characteristics are the following:

1. **Quality of task solving** is determined by the job execution time (depending on the requested resources).

2. **Successfulness of task solving** is determined by whether the job execution is able to start and complete correctly.

In our opinion, at this stage the quality of task solving, i.e. the execution time, is completely determined by two aspects: 1) the parameters of the job launch (the number of allocated cores / nodes, the specified time limit, the selected partition of the supercomputer); 2) dynamic characteristics (such as the CPU and GPU load, the intensity of memory usage and MPI data transmission over the network, etc.). These aspects need to be analyzed to study the job performance. However, the values of the dynamic characteristics can be influenced by many other factors associated with the properties of the job itself or with the external influence of the software and hardware supercomputer environment, which must also be considered. In particular, the dynamic characteristics (and, therefore, the job execution time) are affected by the features of the selected application package or the used library, the program compilation options, and so on. When considering external influence, we identified 3 groups associated with the features of the operation of the communication network, file system and compute nodes. For example, the last group includes the following factors — operating system noises on the node or non-critical failures in the operation of the node hardware, which slow down the job execution.

Similar groups are selected when analyzing the successfulness of the task solving, with the only difference that it does not depend on dynamic characteristics, and therefore there is no such item in the list.

The following is a list of factors for this stage of the job lifecycle:

1. Quality of task solving (at this stage it is determined by how long the job executed):
   - 1.1. ***Number of allocated nodes and cores, specified time limit, selected supercomputer partition.***
   - 1.2. ***Dynamic characteristics of job execution:***
     - 1.2.1. Properties of the job itself:
       - a) *Features of the selected algorithm or its software implementation (unknown without information from the user).*
       - b) ***The used compiler and its options, the application package, the libraries, the environment variables.***
       - c) *Location of selected nodes.*
     - 1.2.2. External influence:
       - a) Communication network:
         - *Insufficient bandwidth of links or switches.*
         - *Non-critical network failures (for example, a failure of one of the links or ports, leading to longer route used for data transfer).*
       - b) File system:
         - ***The overload of file system servers.***
       - c) Compute nodes:
         - *OS noise.*
         - ***Non-critical hardware errors on the node (ECC errors, network errors, etc.).***
2. Successfulness of task solving (at this stage it is determined by the correctness of job completion):
   - 2.1. ***Job exit state.***
   - 2.2. Properties of the job itself:
     - 2.2.1. *Inconsistency of the selected compilers, libraries, environment variables.*
     - 2.2.2. *Incorrect software implementation (unknown without information from the user).*
   - 2.3. External influence:
     - 2.3.1. ***Critical errors that could lead to unsuccessful job execution:***
       - a) *Communication network (for example, critical failures in operating links or switches).*
       - b) *File system (critical failures in the file system).*
       - c) *Compute nodes (critical network or OS errors, etc).*

With this description, we made the first attempt to describe all possible factors, the influence of which on the job may be important for understanding its behavior and performance. Therefore, it is likely to be adjusted in the future, since the range of issues considered here is very large.

Now we describe what may be important or interesting to the user when considering a set of jobs.

At this stage, statistics on target characteristics and factors influencing them are important. In our opinion, it is necessary to get answers to the following questions for a certain set of jobs:

1. Concerning target quality characteristics:
   1.1. What is the distribution of target quality characteristics (queue and execution time) among the user's jobs?
   1.2. What is the distribution of the values of the factors that can influence the target quality characteristics, among the user's jobs?
   1.3. How do factors influence the performance of jobs? I.e. dependence of the performance dynamic characteristics of the job execution on the values of the factors influencing the execution time.
   1.4. How do important factors explicitly affect the target quality characteristics? I.e. dependence of the queue / execution time on the values of the factors.
2. Concerning successfulness target characteristics:
   2.1. What is the frequency of bad values of target successfulness characteristics (i.e. the job did not start or completed incorrectly)?
   2.2. What is the distribution of the values of the factors that can influence the target successfulness characteristics (i.e., the successfulness of job completion), among the user's jobs?
   2.3. How do important factors generally affect target successfulness characteristics? I.e. dependence of the job completion successfulness on the factors' values.

Note that the set of user jobs under consideration can be different:

- All user jobs within the chosen time period.
- Launches of the chosen application package or library.
- Job launches in the selected partition.
- Launches of a certain class of jobs (using a large number of cores, with a particular job exit state, etc.).
- Job that solve a specific scientific problem (unknown without information from the user).

The above description corresponds to the first two steps in the process described at the beginning of Section 2 — the selection of target characteristics and the identification of the factors affecting them (as well as the selection of the most important of these factors). After that, it is possible to proceed to step 3 — defining the formats and the structure of data presentation that can help analyze in practice the influence of the selected factors on the target characteristics. This description is provided below in Section 3.1.

2.2. Developing a model for the administrator. By analogy with the previous model, when creating a model for the administrator, it is also necessary to highlight the main target characteristics of the most importance when analyzing the supercomputer center functioning. And for these characteristics it is necessary to determine those factors that can influence these characteristics, and also to assess the degree of their influence.

Before moving on to defining the target characteristics, let us determine the fixed input data (as we did when considering the model for the user). These are the parameters of the supercomputer operation that the administrator cannot influence. We believe that in this case the following input data are fixed:

- The content and structure of supercomputer equipment (we assume that the administrator cannot influence the number of physically available equipment, its structure and connections);
- the job flow from users (however, the administrator can certainly influence the execution of these jobs);
- the policies and quotas set by the management for executing the flow of jobs.

In this formulation of the problem, the following main target characteristics (determining what is important for the administrator) can be distinguished:

1. **Availability and correct setting of supercomputer resources.** It takes into account how optimally the supercomputer is prepared for executing user jobs. This is primarily influenced by factors related to the operability of the supercomputer hardware, but this also includes issues related to the optimal setting of the system software.

2. **Efficiency of supercomputer resources usage.** When considering this characteristic, we assume that the supercomputer is tuned as "ideal" as possible (for the estimation of which the previous characteristic is responsible), and, based on this, we try to estimate how well the flow of user jobs utilizes the computing system. In other words, we study the utilization of all types of available computing resources. Here we can distinguish two groups of factors influencing this characteristic: 1) utilization of supercomputer resources, which allows us to understand what part of the computing resources is idle; 2) the adequacy of the utilization of resources, which assesses how correctly the resources are utilized, regardless utilization level. At the same time, we divided the first group into 2 parts. The first one focuses on the utilization without considering jobs, in order to evaluate the utilization of various types of resources (i.e., a view focused on the equipment utilization); the second one focuses on the utilization of resources based on job behavior (i.e., a view from the user's perspective).

As above, the following is a list of factors that influence the respective target characteristics. The factors themselves are in italics. In this case, in our opinion, all the selected factors are important for the administrator and should be presented in the model. Note that the factors in different groups are sometimes duplicated, however, the proposed structure, although it is redundant in places, results in a more complete tracking of the factors that need to be analyzed to assess the quality of the SC.

Availability and correct setting of supercomputer resources:

1. Availability of supercomputer hardware:
   - 1.1. *Availability of compute nodes (CPU, GPU, memory, network cards):*
     - 1.1.1. *Node reachability.*
     - 1.1.2. *Presence of hardware failures (for example, ECC or network errors, incorrect processor work modes, hardware overheating).*
   - 1.2. *The same for the communication network (links, switches).*
   - 1.3. *The same for the file system.*
   - 1.4. *The same for the control node.*
   - 1.5. *The same for the service nodes (DHCP, TFTP, NFS, DNS, NTP, LDAP, license, monitoring, statistics).*
2. Optimality of system software configuration:
   - 2.1. Operation optimality of the system software on the hardware described in the previous paragraph. For each type of equipment, it is necessary to evaluate:
     - 2.1.1. *How much hardware is loaded.*
     - 2.1.2. *How well the software performs its task.*
   - 2.2. *Configuration optimality of the resource manager.*
   - 2.3. *Configuration optimality of the monitoring system.*

Efficiency of supercomputer resources usage:

1. Utilization of all types of supercomputer resources (without considering jobs, a view focused on the equipment utilization):
   - 1.1. Utilization of compute nodes:
     - 1.1.1. *Dynamic characteristics of job performance.*
   - 1.2. Utilization of shared computing infrastructure components:
     - 1.2.1. *Overall load by the executed jobs (different characteristics are analyzed for different components). It is calculated for hardware and system software from the list above.*
2. Utilization of all types of supercomputer resources (considering jobs, a view from the user's perspective). Here the activity and efficiency of the most important subjects responsible for the job execution (user, project, organization) are of interest:
   - 2.1. Activity of subjects:
     - 2.1.1. *the number of jobs that users launch or currently execute.*

2.2. The quality of work of individual subjects. In other words, here we are looking to see if there are problems:

2.2.1. with the execution efficiency of subject's jobs

a) *Same factors as in the user model.*

2.2.2. concerning negative impact from other users. This negative impact occurs due to the fact that too high activity of simultaneously running jobs leads to excessive loading of shared resources (file system, communication network, etc.). This is influenced by:

a) *Resource usage intensity of each of the jobs launched on shared resources.*

b) *Location of jobs on resources.*

2.2.3. with efficient usage of different software: application packages, libraries, compiler, etc. It depends entirely on the efficiency of the job launches

a) *Same factors as in the user model (but looking from the side of the software of interest).*

3. Adequacy of resource utilization:

3.1. Optimality of quotas and policies specified by administrators. There are important factors that should be considered:

3.1.1. *Uniform distribution of resources between subjects (users, projects, organizations).*

3.1.2. *Achievement of job limits by users (by the number of running or the total number of jobs, utilization of the control node, etc.).*

3.2. Adequacy of resource usage by supercomputer jobs, not directly related to the intensity of resource utilization:

3.2.1. *Undesirable / incorrect behavior of jobs (in particular, the usage of incorrect job launch modes).*

3.2.2. *Insufficient parallelism of jobs (using an unreasonably small number of nodes and / or processes per node).*

3.2.3. *Using an incorrect partition (for example, launching non-GPU jobs in the partition for GPU-based jobs).*

The above description, in our opinion, concerns the most important issues that may interest the user and administrator regarding the analysis of the performance of supercomputer applications in particular and workload of HPC systems as a whole. The described models will certainly be supplemented and tuned over time, but the proposed variant can already be used in practice to ensure the completeness of the analysis of the supercomputer performance.

3. Software implementation based on the developed model. On the basis of the developed model for the user and the administrator, software solutions were implemented to analyze the quality of the work of supercomputer applications and the computing system as a whole. These solutions provide web reports designed to easily analyze the influence of the factors included in the model. Note that when creating these reports, it was necessary to complete step 3 from the list of actions for developing a model given at the beginning of Section 2. That is, here it was necessary to determine how the data on application and supercomputer behavior should be represented (what input data to use and how to present it, in terms of graphs, tables, etc.) in order to evaluate the influence of the factors identified in the model as conveniently and fully as possible.

The first versions of summary reports have been implemented for the Lomonosov-2 supercomputer, which allows us to conduct an overall analysis and get a general information of all the main aspects without going deep into details. In the future, these summary reports can be supplemented and detailed in accordance with the model, for example, using an "endless" report system [16] also developed by the authors of this paper. Therefore, at this step, it was necessary to select only the most important, useful and convenient forms of presentation. Note that the implementation of some of the presentation options in the administrator report turns out to be currently impossible, since there is no necessary input data available about the work of the supercomputer.

These reports are implemented using the Redash data visualization and analysis system [17]; data is stored in the MongoDB. The volume of stored data is not so large (several GB), however, the heterogeneity and strong interconnection of this data significantly complicates data extraction, which required significant optimization of both the structure of the data and the settings of the database.

3.1. Implementing a summary report based on the model for the user. Let us give a description of the chosen presentation options for the user on the Lomonosov-2 supercomputer (all data is considered for a certain period of time):

1. Overall user activity and performance:
   1.1. A table showing the activity of the user (by the number of launches and occupied node-hours), as well as the activity rating among all Lomonosov-2 users. Likewise for the selected application package. This allows us to evaluate the activity of a given user and compare it to others.
   1.2. A table showing the average user performance according to performance characteristics (CPU and GPU load, memory or communication network usage, etc.), as well as user rating based on each characteristic among all users. Likewise for the selected application package. There is also a table comparing the average performance of the selected user and all other users when launching a specific application package. This allows us to understand how the user is capable of efficiently using the provided resources.
2. Efficiency of using common application packages:
   2.1. 2D graph with a detailed performance comparison of a specific application package launches by the selected user and all other users. An example of such a graph is shown in Fig. 1. The performance characteristics are plotted along the axes (in this case, the CPU load in percent along the X-axis and the GPU load in percent along the Y-axis, but this can be changed if needed), each point on the graph corresponds to one job launch. Blue dots are the launches of the package (in this case, the GROMACS package) by the selected user, red dots — launches by all other users of this package for the time period under consideration. These graphs allow us to understand in detail if a certain user has problems with the optimal usage of a particular package, and how more efficient these launches can be in practice. The above graph allows us to see that the selected user often utilizes both GPU and CPU very effectively when working with the package, i.e. no problems are observed.
   2.2. A similar graph, but showing only the launches of a certain package by the selected user, the color indicates the job size (the number of used nodes). This allows us to see how the job performance depends on their size.
   2.3. Another similar graph, but for all used packages and their versions. Only the jobs of the selected user are shown here. Such a graph helps to compare the efficiency of different types of launches and, for example, determine which version of the application package can give higher performance.

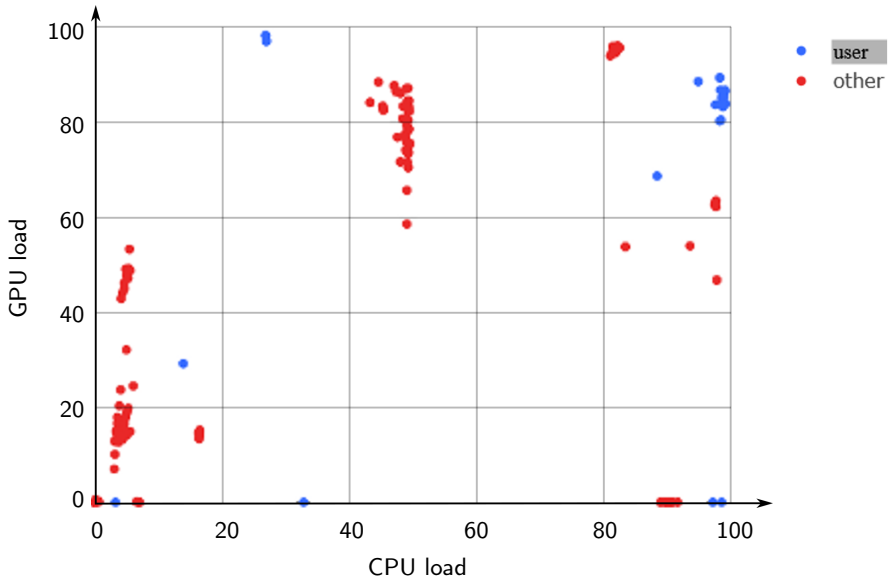

Figure 1. Comparison of average CPU and GPU load (in percent) for the jobs of the selected user and all other users of GROMACS package. Each dot represents one job

3. Presence of performance issues:
   3.1. A table with a list of the most critical performance issues found in the jobs of this user. Critical issues are those that signal about a significant decrease in the job efficiency; the presence of such issues is detected by the TASC software complex [18] developed at the Research Computing Center of Moscow State University. These issues can detect, for example, a hung job or an abnormally inefficient use of all available resources. This table helps identify the most common problems that lead to reduced resource utilization as well as determine the jobs where these problems occur.
   3.2. A table listing any performance issues in the launches of a particular application package. This helps to identify difficulties with optimal use of these packages.
4. Detailed information on job launches:
   4.1. 2D graph showing the distribution of job queue and execution time depending on the job size. An example of such a graph is shown in Fig. 2. Each point on the graph corresponds to one job, the X-axis is the execution time, the Y-axis is the queue time (both in hours). The colors represent the job size (number of nodes). Such a graph helps to find out, for example, what job size allows conducting experiments faster, taking into account the waiting time in the queue (and if such a dependence exists at all). The graph below shows that there is a general dependence on the waiting time — large jobs (16–63 nodes, blue dots) wait on average longer (sometimes – much longer) in the queue, while average jobs (4–15 nodes, red dots) wait almost the same time as small jobs (1–3 nodes, green dots). This information can help a user optimize the process of running experiments.
   4.2. A pie chart showing the distribution of the number (and node-hours) of job launches depending on the job exit state. This graph is needed for immediate assessment of how often jobs are completed incorrectly. First of all, this information is of interest to administrators, but in some cases it is also useful for users.
   4.3. Timeline indicating the number of jobs completed with a specific state during every day or week. This allows us to determine if at some point there were some general job failures, as well as to study the activity of launches by time.
5. A table with detailed information about individual job launches. This table contains a description of the parameters for each launch (the launch time, the job size, the exit state, the used application packages, the launch command), as well as the average values of the main job performance characteristics (CPU and GPU load, the amount of bytes transmitted / received by MPI and FS networks per second, the frequency of

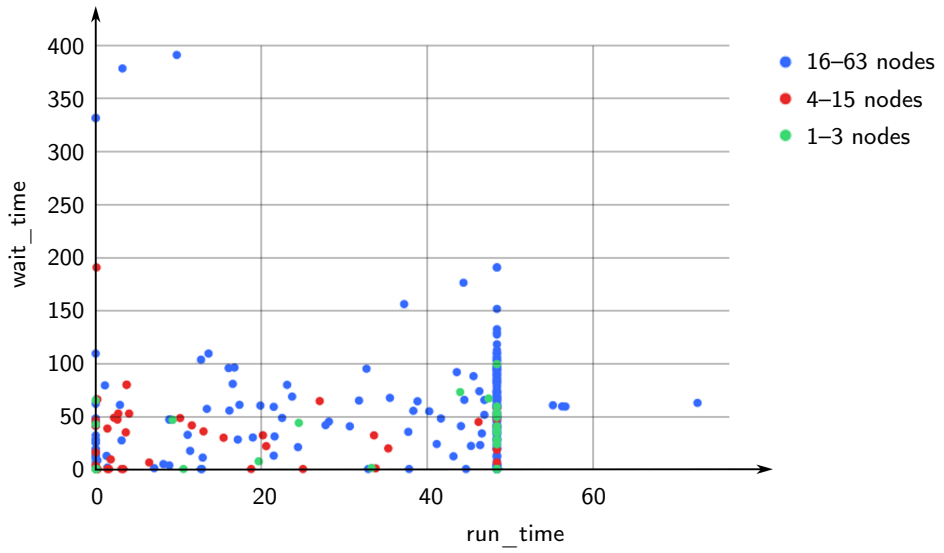

Figure 2. Distribution of queue and execution time (in hours) for different jobs of the selected user, depending on the amount of used nodes

read / write memory operations per second, the amount of memory used). This most detailed information is needed for a thorough study of the questions that arise when analyzing the above described parts of the web report.

In our opinion, such a web report allows the user to quite easily and quickly obtain general information on all the most important aspects concerning the performance of his jobs and the use of standard application packages (which in practice are used very often, and therefore particular attention should be paid to the analysis of their launches). For a further detailed study of arising issues, other analysis tools (e.g. profilers or debuggers) should be used. This web report has already been implemented on the Lomonosov-2 supercomputer, but is working in test mode. After completing its approbation, it is planned to provide access for it to all supercomputer users.

Note that in a similar way, taking into account the developed model, a web report was implemented for the administrator. This report helps evaluate not only standard aspects of the supercomputer operation (for example, the number of running and waiting jobs, the availability of compute nodes or a list of the most active users and projects), for which the information can be obtained from existing solutions, but also some non-trivial aspects that allow, in our view, looking at the supercomputer workload analysis in a different way. This includes, for example, the data on nodes with the lowest values for each performance characteristic, which helps to determine the nodes that have problems in their operation that prevent the achievement of optimal utilization rates, or information on the frequency of users reaching the set limits on the number of simultaneously running or waiting jobs, which allows assessing the adequacy of the quotas. Specific examples demonstrating the capabilities of the obtained solutions are given in the next section.

3.2. Real-life examples of the performed analysis. Below are some examples of how the developed software helps in practice obtain useful information about various aspects of the SC functioning. Let us start by looking at the examples obtained by implementation of the model for the user.

One of the questions important for the user when analyzing the performance of his applications is how efficiently he uses standard application packages. To study this important aspect, we provide, among other things, graphs that allow comparing the performance characteristics for launches of a certain package by a specific user and other users of this package.

An interesting example of this graph is shown in Fig. 3. It compares job performance for launches of the CP2K package [19] by a selected user (blue dots) and to all the others (red dots). The left graph shows the processor load (X-axis, in percent) and the amount of received MPI data per second (Y-axis, in millions of bytes). The right graph shows the processor load (X-axis) and the number of read / write operations to memory per second (Y-axis, in billions of operations). It can be seen that the processor load is almost always in the interval of 45–50%, which corresponds to the optimal load of all available physical cores (100% is achieved when all logical cores are fully loaded, the number is 2 times greater due to the use of HyperThreading technology on the

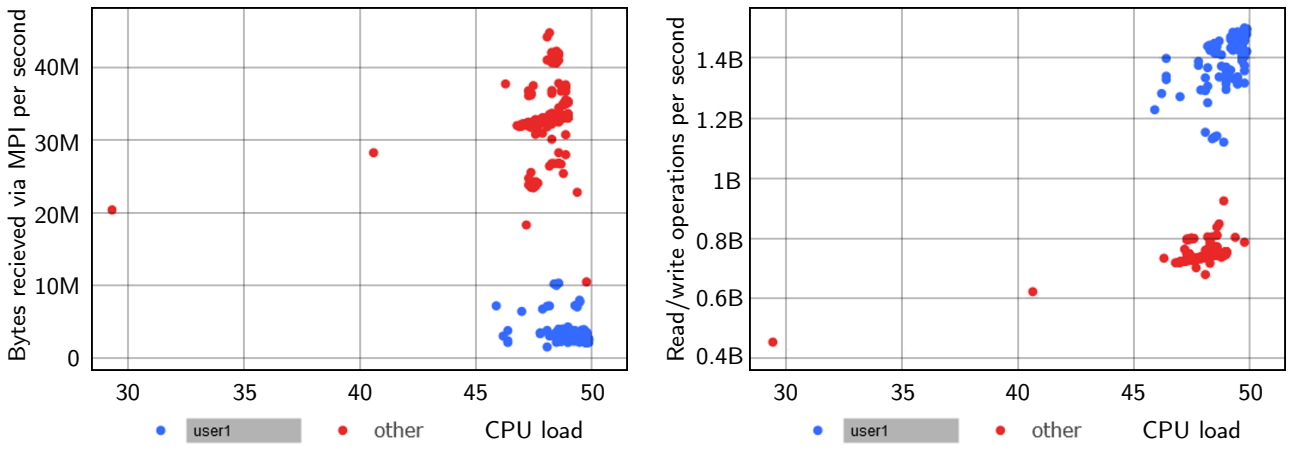

Figure 3. Performance comparison for launches of CP2K package by one selected user (blue dots) and all other users of this package (red dots). Axis X shows average CPU load in percent on both graphs, axis Y shows billions of memory operations per second (left graph) and millions of bytes received via MPI per second (right graph)

nodes). As for the other characteristics, the situation is different. All jobs of the selected user show a noticeably lower intensity of MPI usage, but a noticeably higher frequency of memory operations. This information can be useful for the user: having seen such a picture, he can realize that this package allows more intensive use of the communication network, which he might not have known before. Of course, it is possible that in the task he is solving, such an opportunity cannot be used at all due to algorithm or implementation restrictions, but in other cases the user can try to optimize his work with the package. On the right graph, the situation is reversed — the user utilizes memory much more intensively (and probably more efficiently), and this knowledge can be useful to other users of the CP2K package.

As mentioned above, the previously developed TASC software package automatically detects performance issues in all jobs being executed. Among these issues the most critical ones, leading to a significant part of all allocated resources being idle, are separately distinguished. Information about detected issues is also provided to the user. Fig. 4 shows an example of a real-life output to the user about critical issues found in his jobs during year 2020. Each row corresponds to a single detected issue, showing its description as well as the number of launches and node-hours spent on the jobs with this issue detected. One can see that during one year approximately 40,000 node-hours were spent on jobs in which supercomputer resources were almost not used in practice. This is not so much during the year, but on the other hand, these jobs accounted for 93% of all node-hours occupied by the user, i.e. the vast majority of this user's jobs were ineffective. Moreover, he may not even know about this situation, and in such cases, these graphs are vital for notification. By clicking the link in the left column, the user can see the details on individual launches, which helps better understand the reasons for such behavior. Note that in some cases this behavior may be normal and expected — for example, some algorithms simply cannot be effectively implemented on modern general-purpose supercomputers, but in any case, it is useful for the user to know about these situations.

url	nodeh	count	description
link	39,591	22	Low intensity of using all available resources (CPU, memory, network, GPU).
link	96	1	This job shows suspiciously low execution efficiency, comparing to other jobs on this supercomputer. It is likely to have significant performance issues.
link	1	1	Abnormally low intensity of using all available resources.

Figure 4. Information about most popular critical issues (that indicate significant performance loss) detected in user's jobs; "nodeh" refers to the amount of node-hours occupied by these jobs

Another important question that a user may be concerned about is how efficiently his program works with MPI and how well it scales (changes with an increase of the number of nodes used). An interesting example on this topic is shown in Fig. 5. In this graph, the points correspond to individual job launches, the X-axis shows the average processor load in each job, the Y-axis — the amount of bytes transferred per second over the MPI network. The color indicates the size of the job.

It can be seen that there is a clear reverse — the larger the job is, the less intensively the MPI network is used on average (points of any color at the very bottom probably correspond to test or incorrect runs, so we do not take them into account here). So, on average, for jobs running on 16–63 nodes (green dots), each node receives about 150–250 MB of data per second; for jobs on 4–15 nodes (blue dots) — about 300 MB per second, and for two jobs on 1–3 nodes (red dots) — 350 MB per second. This quite likely indicates suboptimal scalability of user jobs: with an increase in the number of nodes, the efficiency of MPI data transmission decreases. This situation requires a more detailed study (using, for example, the "endless" reports mentioned earlier [16]), but anyway such a graph could be helpful for the user, since it can facilitate understanding the situation.

Next, let us take a look at some interesting real-world examples from the administrator model implementation. The use of the model allows us to look from a different angle, sometimes in a non-trivial way, at various aspects of the SC functioning, including the correctness of the hardware operation. One of the examples is the analysis of the rate of application failures on individual compute nodes. Supercomputing jobs can be completed with various exit states, some of which signal an incorrect completion. One of the most important is the NODE_FAIL state, indicating that the job completed incorrectly due to a failure in the node itself. Some

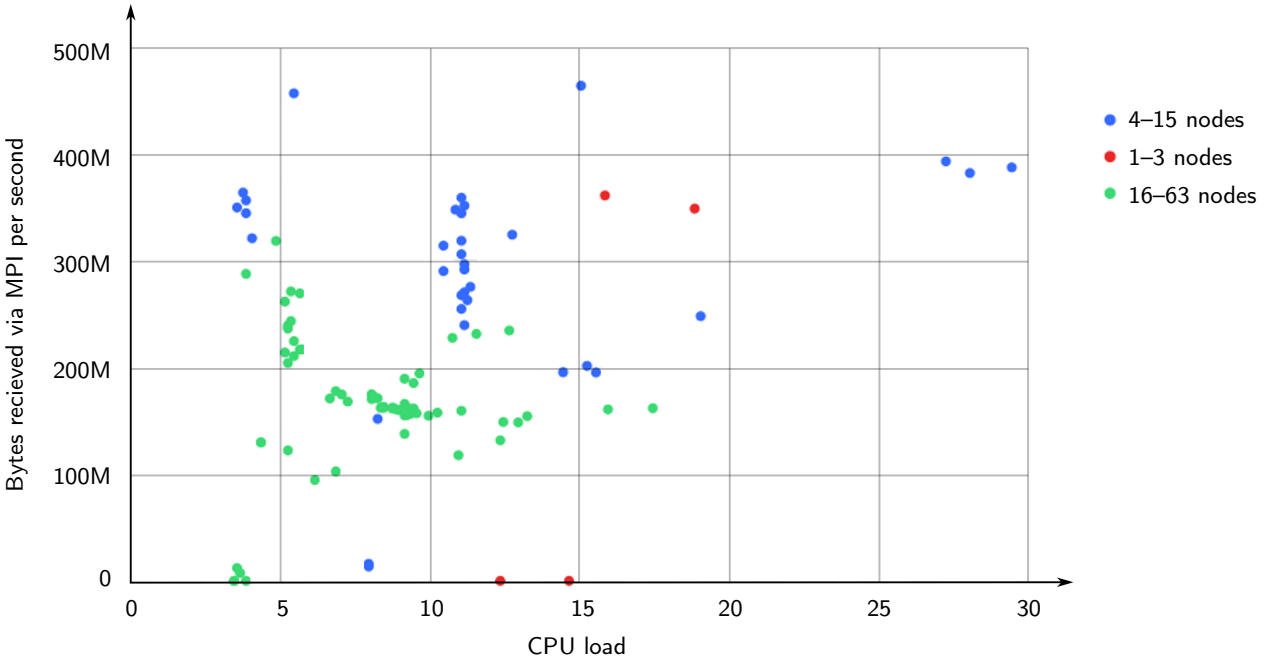

Figure 5. Correlation of job performance characteristics and memory job size. Axis X shows CPU load in percent for each job, axis Y — MPI usage intensity, in millions of bytes received via MPI per second

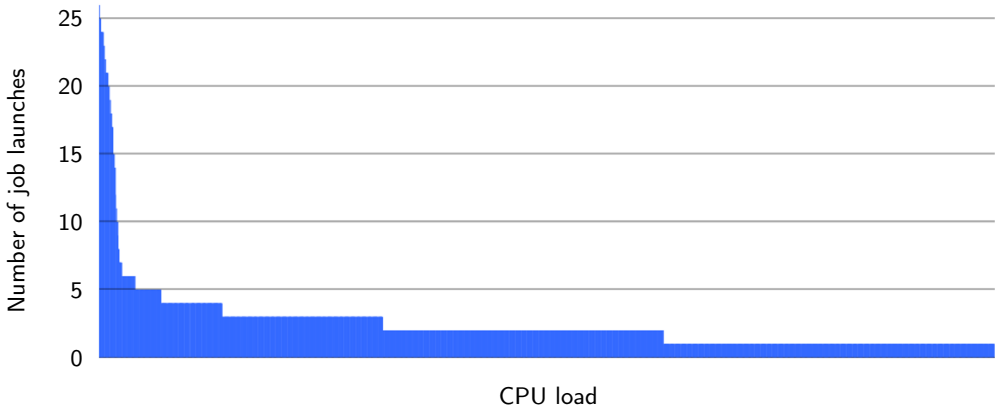

Figure 6. List of compute nodes on which jobs were completed incorrectly (with NODE_FAIL job exit state), mostly due to hardware failures

nodes can fail more often than others (for example, due to misconfiguration or defects in hardware), and these cases should be investigated and identified separately. An example of a graph helping to study this case, which is implemented in the administrator's web report, is shown in Fig. 6. This graph shows the number of jobs that completed with the NODE_FAIL state during 2020 for each compute node of the Lomonosov-2 (the graph is sorted in descending order of this value). One can see that for the most of the nodes the value is very small (nodes with a zero value are not shown here), but for a couple of dozens of nodes the values are considerably higher. Looking only at this graph, we cannot be sure that these nodes are working incorrectly (and some of them could have already been fixed within a year), but this is definitely a signal for administrators that these nodes should be given special attention — most likely, there are certain problems in the work of some of them.

Another very important topic that requires constant attention of administrators is the utilization of shared resources, for example, the file system. It is possible (and necessary) to check the operability of the file system in different ways, and one of the methods is to analyze the load of the file system servers. In the developed web report, we track the load average (the average number of active processes per node) for all servers; an example of such a graph is shown in Fig. 7. This shows the change in the value of the load average for all servers of the

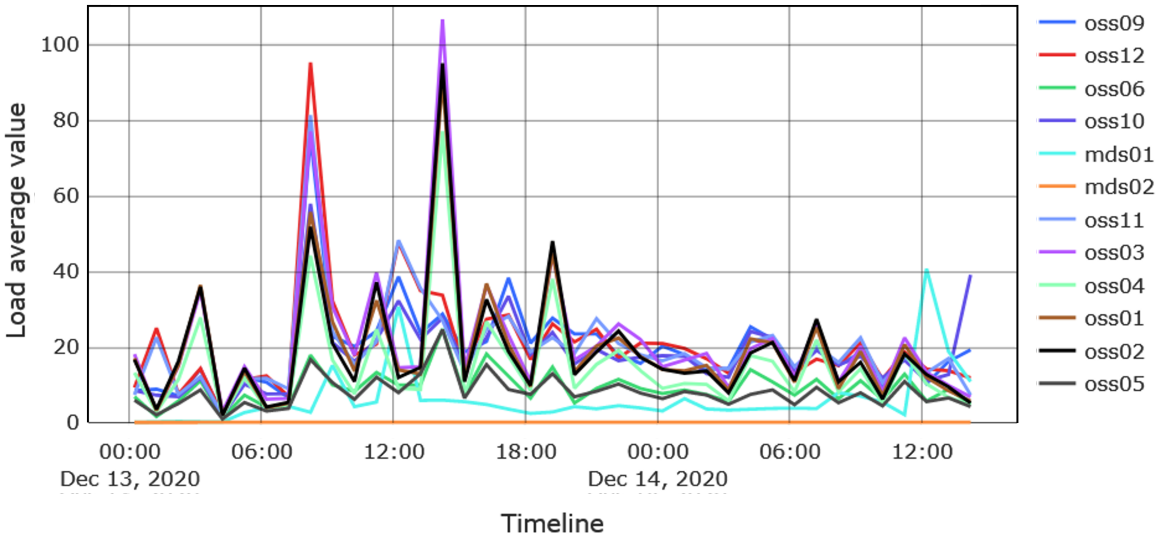

Figure 7. Timeline of load average values for Lustre server

Lustre file system used at the Lomosonov-2 during two days in mid-December. It can be seen that twice during this time (around 8 and 14 o'clock on December 13), the load on many servers jumped sharply and reached too high values. This suggests that at that moment Lustre could not cope with the incoming load from user jobs. In the shown examples, the load then immediately fell down, i.e. there was no critical overload of the file system, however, as in the previous case, the administrator must be able to track the occurrence and, if necessary, promptly respond to such situations.

The above examples are intended to show (at least to some extent) the variety of useful information available in the developed software solution. We separately note that the implementation of the showed model that not all the data needed for a complete analysis of the quality of the SC functioning is currently available, which once again emphasizes the importance of development and consideration of such a model. For example, it is worth collecting more complete information about the health of service servers (as an ideal, to implement functional tests for each of them checking not only its load, but also the quality of its work), and also to learn how to track the relationship between the excessive load of shared resources (the file system, the communication network, the control node) and performance characteristics for currently running jobs, helping determine the root causes of equipment overload more precisely. Therefore, in the future, we plan to implement the collection of missing data for putting into practice an analysis tool fully corresponding to the developed model.

4. Conclusions. In this paper, we present the model designed to help take into account all the potential needs of users and administrators when performing holistic analysis of the efficiency of supercomputer jobs and the quality of supercomputer center functioning in general. This model defines global goals, the achievement of which is the most important for each of two groups of people, describes all kinds of factors related to the properties of the supercomputer job flow and the operation of the supercomputer software and hardware that can affect the achievement of these goals, and also helps determine specific ways to analyze the impact of these factors in practice. The developed model is only the first version that will be presumably supplemented in the future, however, the results obtained in practice already show that this model can be useful, since it helps to ensure the completeness of the analysis and explore some aspects of the supercomputer's operation from new, sometimes non-obvious points of view.

In the future, we plan to provide users of the Lomonosov-2 supercomputer with access to web reports describing the behavior and performance of their jobs, developed in accordance with the model, and further refine the configuration of web reports (and possibly the model itself) based on the experience obtained by the developed solution.

4.1. Acknowledgments. The results described in this paper were obtained with the financial support of the grant from the Russian Federation President's Fund (MK–2330.2019.9).The research is carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University.

References

1. E. Joseph and S. Conway, "Major Trends in the Worldwide HPC Market," https://hpcuserforum.com/presentations/stuttgart2017/IDC-update-HLRS.pdf. Cited January 29, 2021.

2. V. V. Voevodin, A. S. Antonov, D. A. Nikitenko, et al., "Supercomputer Lomonosov-2: Large Scale, Deep Monitoring and Fine Analytics for the User Community," Supercomput. Front. Innov. **6** (2), 4–11 (2019). doi 10.14529/jsfi190201.

3. D. A. Nikitenko, P. A. Shvets, and V. V. Voevodin, "Why do Users Need to Take Care of Their HPC Applications Efficiency?," Lobachevskii J. Math. **41** (8), 1521–1532 (2020). doi 10.1134/S1995080220080132.

4. Intel VTune Amplifier Documentation. https://software.intel.com/en-us/vtune. Cited January 29, 2021.

5. N. Nethercote and J. Seward, "Valgrind: A Framework for Heavyweight Dynamic Binary Instrumentation," SIGPLAN Not. **42** (6), 89–100 (2007). doi 10.1145/1273442.1250746.

6. M. Geimer, F. Wolf, B. J. N. Wylie, et al., "The Scalasca Performance Toolset Architecture," Concurr. Comput. Pract. Exp. **22** (6), 702–719 (2010). doi 10.1002/cpe.1556.

7. Vampir Framework Home Page. https://vampir.eu. Cited January 29, 2021.

8. Arm Forge | Cross Platform Parallel Debugger for C++ and Cuda. https://www.arm.com/products/development-tools/server-and-hpc/forge. Cited January 29, 2021.

9. TotalView Debugger for HPC Computing. https://totalview.io. Cited January 29, 2021.

10. M. D. Jones, J. P. White, M. Innus, et al., *Workload Analysis of Blue Waters*, arXiv preprint: 1703.00924v1 [cs.DC] (Cornell Univ. Library, Ithaca, 2017). https://arxiv.org/abs/1703.00924. Cited January 29, 2021.

11. A. Brian et al., *2014 NERSC Workload Analysis.*
https://www.yumpu.com/en/document/read/55341970/2014-nersc-workload-analysis. Cited January 29, 2021.

12. N. A. Simakov, J. P. White, R. L. DeLeon, et al., *A Workload Analysis of NSF's Innovative HPC Resources Using XDMoD*, arXiv preprint: 1801.04306v1 [cs.DC] (Cornell Univ. Library, Ithaca, 2018).
https://arxiv.org/abs/1801.04306. Cited January 29, 2021.

13. M. J. Abraham, T. Murtola, R. Schulz, et al., "GROMACS: High Performance Molecular Simulations through Multi-Level Parallelism from Laptops to Supercomputers," SoftwareX **1–2**, 19–25 (2015). doi 10.1016/j.softx.2015.06.001.

14. J. C. Phillips, R. Braun, W. Wang, et al., "Scalable Molecular Dynamics with NAMD," J. Comput. Chem. **26** (16), 1781–1802 (2005). doi 10.1002/jcc.20289.

15. J. Hafner, "*Ab-initio* Simulations of Materials Using VASP: Density-Functional Theory and Beyond," J. Comput. Chem. **29** (13), 2044–2078 (2008). doi 10.1002/jcc.21057.

16. P. A. Shvets and V. V. Voevodin, "'Endless' Workload Analysis of Large-Scale Supercomputers," Lobachevskii J. Math. **42** (1) [in press].

17. Redash Homepage. https://redash.io. Cited January 29, 2021.

18. P. Shvets, V. Voevodin, and D. Nikitenko, "Approach to Workload Analysis of Large HPC Centers," in *Communications in Computer and Information Science* (Springer, Cham, 2020), Vol. 1263, pp. 16–30.

19. J. Hutter, M. Iannuzzi, F. Schiffmann, and J. VandeVondele, "cp2k: Atomistic Simulations of Condensed Matter Systems," Wiley Interdiscip. Rev. Comput. Mol. Sci. **4** (1), 15–25 (2014). doi 10.1002/wcms.1159.

Accepted
15 January 2021