Comprehensive efficiency analysis of supercomputer resource usage based on system monitoring data

A A Mamaeva¹, D I Shaykhislamov¹,², Vad V Voevodin² and S A Zhumatiy²

¹ Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University, 119991 Russia, Moscow, Leninskie gory, 1, bld. 52
² Research computing center, Lomonosov Moscow State University, 119234 Russia, Moscow, Leninskie gory, 1, bld. 4

E-mail: vadam@parallel.ru

Abstract. One of the main problems of modern supercomputers is the low efficiency of their usage, which leads to the significant idle time of computational resources, and, in turn, to the decrease in speed of scientific research. This paper presents three approaches to study the efficiency of supercomputer resource usage based on monitoring data analysis. The first approach performs an analysis of computing resource utilization statistics, which allows to identify different typical classes of programs, to explore the structure of the supercomputer job flow and to track overall trends in the supercomputer behavior. The second approach is aimed specifically at analyzing off-the-shelf software packages and libraries installed on the supercomputer, since efficiency of their usage is becoming an increasingly important factor for the efficient functioning of the entire supercomputer. Within the third approach, abnormal jobs – jobs with abnormally inefficient behavior that differs significantly from the standard behavior of the overall supercomputer job flow – are being detected. For each approach, the results obtained in practice in the Supercomputer Center of Moscow State University are demonstrated.

1. Introduction
Supercomputer technologies are used very widely in the modern world. In different fields of science, such as medicine, astronomy, quantum physics, hydrodynamics, molecular biology and many others supercomputers are used on an ongoing basis to conduct simulations and perform numerical calculations. Thus, the speed of scientific research in these areas depends to a large extent on the efficiency of supercomputer usage. However, it often turns out that this efficiency is very low. There are several reasons for this. The first reason is an incredible complexity of the supercomputer architecture, which makes it almost impossible to take into account all the features and capabilities of the hardware platform when writing a parallel program. Another reason is an increase of large-scale use of supercomputing technologies – more and more people tend to use supercomputers, and in most cases new users are excellent experts in their subject areas, but they are not so experienced in the field of parallel computing.

This leads to the fact that the efficiency of supercomputer resource usage can be very low [1]. To solve this problem, it is necessary to study the behavioral features of both the supercomputer as a whole and the individual applications being executed on it. There are a lot of ready-to-use tools for analyzing the performance of individual parallel applications (trace tools, profilers, debuggers, etc),
but, in our opinion, the problem of studying the performance of a supercomputer as a whole has not been studied in many respects. One of the possible research directions in this area is to study the entire flow of supercomputer applications based on system monitoring data, which is the purpose of this research. Solving this issue will allow the management and system administrators of supercomputer centers to increase the performance of their computing systems, which, in turn, will allow users to solve scientific problems faster and more efficiently.

In this article, we will describe three different approaches for conducting a comprehensive study of the resource utilization statistics, the performance of application packages, and anomalies in the application flow, accordingly. All these approaches are being actively developed in Research computing center of Lomonosov Moscow state university (RCC MSU).

The results in this paper are obtained in the MSU Supercomputer Center, i.e. allow to analyze its efficiency. But these results can be applied outside this center as well, since they show how similar analysis can be carried out on other systems and what useful conclusions it allows to make. Moreover, all these approaches are being developed with their portability in mind. Also it should be noted that these studies are a part of a general work carried out in RCC MSU aimed at analyzing the most diverse efficiency aspects of both individual parallel applications and supercomputer center as a whole [1].

Next, the description of these approaches will be given, as well as several interesting results obtained using these approaches in practice.

2. A description of the approaches to analyze supercomputer resource utilization efficiency

All the approaches described in this article are based on using system monitoring data. This allows to obtain all of the needed information on the execution of various jobs, at the right level of detail. In this case, studying supercomputer utilization efficiency requires examining the flow of parallel applications executed on the supercomputer. For that purpose, various dynamic characteristics are measured for each job executed on the supercomputer, allowing to describe its behavior. Since we are primarily interested in the efficiency of job execution, we will analyze the following characteristics: CPU utilization (CPU user load on the processing cores, system load (loadavg)), interaction with the memory subsystem (cache misses per second, memory reads/writes per second, etc.), InfiniBand or Ethernet network usage intensity (bytes and packets sent and received per second), file system usage intensity, etc.

Various existing monitoring tools can be used to obtain this data, such as Nagios [2], Zabbix, Zenoss, etc. MSU supercomputers currently use a set of custom-built solutions, but these will be replaced in the near future with DiMMon [3], a system being developed at the RCC MSU, designed specifically to run on extremely large-scale supercomputers and therefore offering good scalability, flexibility, and high performance. Since in this study we are interested in the overall supercomputer efficiency, not the efficiency of each particular application, in most cases only integral values need to be considered for each measurement of a specific job – for example, average user load or average number of L1 cache misses per second during the entire job runtime. This is not enough to conduct a detailed analysis of individual applications, but quite sufficient for studying the behavior of the supercomputer in general. In this case, a reasonable amount of data that does not require specific methods for big data processing is produced, which is also important for this kind of research.

General information on each job should also be collected, such as the job ID and user name; start and finish times; the name of used partition of the supercomputer; the number of cores and nodes used, etc. This data can easily be obtained from the resource manager, which is responsible for distributing and queuing supercomputer applications. The Slurm resource manager is used in MSU supercomputers.

Next, we will discuss three individual approaches in more detail.

2.1. Analysis of computing resource utilization statistics

The first approach is aimed at collecting and analyzing usage statistics for the supercomputer’s computing resources. It is based on the analysis of the job flow structure – information is collected on
how each job utilizes the available resources [4]. This study helps answer the following questions, among others:

- Are supercomputer resources – compute cores, nodes, sections, etc. – efficiently loaded?
- Are policies and quotas configured appropriately, is there any imbalance between users’ activity or partitions’ usage?
- Are queue planning strategies working properly?

There aren’t many studies aimed at analyzing computing resource utilization statistics based both on resource manager data and information from system monitoring tools. The works that do exist are mainly intended for use on a specific supercomputer only (e.g., [5, 6]), and cannot be applied in our case. However, studying these works helped us to determine needed research directions for this study.

As previously mentioned, this analysis was performed based on the integral values for all available dynamic characteristics that describe the supercomputer resource utilization efficiency. This information can be taken directly from the database of the monitoring system, so obtaining input data is not an issue in this case.

It is much more of a challenge to figure out what useful conclusions can be derived from this information. Described approach is intended to answer this question. A number of methods were suggested that help obtain the most relevant statistics and evaluate resource utilization from different perspectives. A description of the three methods and their results is presented below. All of the methods are described in the article [4].

2.1.1. Identifying different classes of jobs. We have identified several classes of jobs with specific peculiarities that help us to evaluate the structure of the supercomputer job flow. For example, one class is “ideal-for-supercomputer” programs that actively utilize both compute nodes and communication network. These programs are very well suited to run on supercomputers, so one should try to increase the share of such jobs running on the machine. The statistics on the number of jobs in this class on the Lomonosov-2 supercomputer, along with some other classes, are shown in Table 1.

We can see that “ideal” programs are unfortunately not very common – only 5.5% of the total number of jobs launched since the beginning of the year. However, one must keep in mind that by far not all algorithms, let alone their specific implementations, can generally exhibit the efficient behavior required to be included in this class. Other classes, such as “suspicious” (very low CPU load and loadavg) or “non-communicative” (a job that occupies many compute nodes but almost does not use the communication network) are, on the contrary, less efficient in occupying supercomputer resources, so the share of these jobs should be decreased if possible. As can be seen from Table 1, these jobs are not numerous – less than 2% in sum on the Lomonosov-2 supercomputer since the beginning of the year.

### Table 1. Statistics on the number of jobs in different classes for Lomonosov-2 supercomputer (on October 14, 2017).

| Class               | Last day | Last week | Last month | Since the beginning of the year |
|---------------------|----------|-----------|------------|---------------------------------|
| Overall number of jobs | 388      | 5 317     | 16 676     | 298 676                         |
| «Ideal»              | 17       | 187       | 725        | 16 254                          |
| «Suspicious»         | 9        | 64        | 171        | 450                            |
| «Non-communicative»  | 10       | 130       | 735        | 5 173                          |
| «Serial»             | 3        | 10        | 39         | 1 304                          |
| «Single-node»        | 83       | 1143      | 3 612      | 28 348                         |

The last two classes in Table 1 refer to jobs that may use the resources quite efficiently but require very few of them – one node (“single-node” jobs) or even one compute core (“serial”). The supercomputer is probably not needed for these jobs; they can be executed just fine on a standard
server or a workstation. So, ideally the number of these tasks should be kept to a minimum as well. It can be seen that very few of the tasks running on the Lomonosov-2 supercomputer belong to the “serial” class – less than 0.5%. However, single-node tasks are fairly common; unfortunately, this situation is similar at many supercomputer centers.

2.1.2. Studying the dynamics of supercomputer behavior. Another useful way of analyzing resource utilization statistics is to study how different aspects of supercomputer behavior change over time. How did the overall resource utilization change over the last months? Did some policy/quota adjustments help to increase the overall efficiency? Is there any periodicity in the usage of the supercomputer during a week/month/year? Answering these questions helps administrators of a supercomputer center evaluate the overall trends.

One of such interesting trends that are useful to analyze is the behavior of “big” jobs – jobs that require a lot of compute cores. We have analyzed and compared the data from Lomonosov-2 supercomputer within this year for 2 classes: “big” jobs with more than 256 cores, and all others (“small” jobs). The results have shown that the average values of both CPU load and loadavg for these classes are very similar. This means that average CPU utilization does not differ much for computationally-intensive and more light-weight jobs in our case. This is also true in terms of memory usage intensity – the number of load/store operations per second tends to be quite the same.

But the situation is different for communication network intensity. Figure 1 shows the distribution by month of an average number of bytes sent per second for two classes (the results for January are not available due to the lack of system monitoring data). Here we can see that in general “big” jobs show more intensive network usage. But it should be noted that “small” class includes jobs running on one node that does not use network at all, so this kind of behavior seems like an expected one. Taking this into account, the low values for “big” class in February as well as in October could mean that “big” jobs may not have used communication network efficiently. Of course, this could happen for many different reasons and does not necessarily indicate a performance issue, but in any case system administrator should conduct a more detailed analysis to find root causes of such behavior.

![Figure 1. A number of bytes sent per second using Infiniband network, distribution by month for “small” and “big” classes.](image-url)
2.1.3. **Studying the distribution of values by characteristics.** In this method, an independent detailed study of a certain characteristic is performed across the entire task flow. This helps identify specific patterns and assess certain performance features for a given supercomputer. Let’s consider the following example we came across during our practical study. We made a list of users on the Lomonosov supercomputer with the lowest loadavg values (the average value was calculated among all the jobs launched by each user). Only those users whose jobs occupied more than 100,000 core-hours in total were included in the list. So, the list included users that: 1) consumed a substantial amount of computing resources; 2) used them inefficiently (using loadavg as a metric). One user stood out from the list, as his tasks had a very low average loadavg value of 0.07, which in general indicates totally abnormal behavior. Further study showed that most of his tasks were executed normally (with loadavg > 1), but more than 25 launches had a loadavg less than 0.1, which probably meant that the jobs were stalled. The appearance of such failed launches stopped after a certain point in time; apparently, the user was able to find the problem and fixed the situation. However, a lot of computing resources had been wasted up to that point. This is why this kind of information is very important for system administrators – situations like this need to be detected promptly and their root causes identified and excluded.

2.2. **Studying the efficiency of application package usage**

While the first approach is aimed at studying all jobs across the supercomputer job flow, the second approach is focused particularly on using off-the-shelf software packages. More and more users from all subject areas are using supercomputers these days; in this respect, the number of off-the-shelf application packages installed and used in practice is also increasing. This means the efficiency of these application packages will have an increasingly powerful effect on the overall supercomputer utilization efficiency. We are primarily interested in applied packages that are designed for solving particular tasks in different application areas – such as the Gromacs package [7] for modeling the molecular dynamics for physical and chemical processes, or the OpenFOAM package for the numeric modeling in continuous mechanics. However, it is also important to study the usage of system software packages – compilers, MPI libraries, etc. This study is aimed at addressing all these issues.

First of all, the study could be of interest to administrators at other supercomputer centers, as the software package study method described here can easily be implemented on most modern supercomputers. However, ordinary supercomputer users can also benefit from this information, as it helps evaluate and compare the efficiency of various packages on a given supercomputer.

Two tasks should be addressed before we can obtain the data needed to assess the efficiency of off-the-shelf software packages. The first task is to obtain execution efficiency data for all jobs; this can easily be done using monitoring systems, so we are not going to discuss this in detail.

The second task is to determine whether a certain job is using one software package or another. That would require obtaining additional information for each job, such as determining the list of linked static and dynamic libraries, the modules used, and also collecting data on the used compilers and MPI libraries as well as their launch options. In this study, we used XALT [8], a tool developed by the Texas Advanced Computing Center (TACC), to collect this kind of information. This tool was chosen because it is an open-source software that can be easily installed on most modern supercomputers and provides all the functionality needed for our study. This tool works as follows:

1. XALT replaces the standard link and launch scripts (particularly, ld and mpirun) used on supercomputers with its own wrappers.
2. Each time any user links or launches his program, XALT obtains all the information needed (libraries used, full file names, environment variables, input parameters used, etc.) and stores it in the user’s home directory.
3. Obtained data is periodically stored in a single MySQL database. This database aggregates all of the “raw” data from the supercomputer and is in fact the entry point for analysis.
4. Using SQL queries, the administrator can obtain all of the relevant statistics on the utilization of software packages or libraries.
Using XALT on MSU supercomputers helped us analyze the application package behavior from various perspectives. In this article we will consider the results obtained on the Lomonosov-2 supercomputer for one specific application package – NAMD [9], which is frequently used in molecular dynamics.

Studying just the integral values for the package already provides interesting information. For example, it turned out that NAMD was the leading package in terms of job execution duration – the average execution time of a single job was more than 1 day. An average NAMD job requires about 80 compute cores, which is quite a lot for application packages. This means that the efficiency of each launch is particularly important, as it has a substantial impact on the efficiency of the entire supercomputer functioning. It is also interesting to note that NAMD uses the communication network very actively, with sustained intensity of more than 150 MB sent per second; the only off-the-shelf package showing higher network load is CP2K, which is also used in molecular dynamics.

Naturally, average values alone are not enough. For example, the average CPU load and loadavg values are 38.74% and 12.36, respectively. These may look like good values, but what is hiding behind these averages? Let’s look at the distribution of these values by individual users (Figure 2).

![Figure 2. Average CPU load and loadavg values for NAMD package launches, distribution by individual users (sorted by the CPU load value).](image)

You can see that most users use NAMD in a very similar manner, except for just 4 users. The most notable is user 11, as his loadavg value is clearly an outlier. A more detailed study of individual launches showed that the first three launches of this package by that user had a loadavg value of more than 100, which is definitely an abnormal behavior (normal loadavg value for Lomonosov-2 supercomputer is less than 30 active processes per node). But after that it returned to normal. Apparently the user was initially learning how to use the package (or maybe tuning package options for this particular supercomputer), after which he started utilizing the software in a normal fashion.

Users 14-16 are also outliers, with much lower average values than the other users. An analysis of individual launches for user 16 (Figure 3) shows that inefficient use of the NAMD package is quite common and continues to date – CPU load is always less than 10%, while a common value is above 40%. In this case we have a very clear suspicion that application package is used inefficiently on a regular basis. A direct contact with this user is needed to figure out if it is possible for him to use more efficient implementations within this package.
One of the possible reasons why package usage efficiency can be quite low is that the particular version or the configuration of the package installed on the supercomputer does not suit the needs of the user. In this case – as well as in other cases when a user is unsatisfied with generally available package – the user may build by himself a local, more optimal for him version of this package. Administrators of supercomputer centers should also track this kind of activity: if a lot of people use other package versions installed locally, maybe it means that the generally available package installed by administrators should be updated or changed somehow. From the other point of view, maybe the user could not figure how to properly use the generally available package, in this case an administrator should help him to solve the issue.

That is why it is also important to track what common users install locally. We have collected the data about the usage of local libraries on Lomonosov-2 supercomputer since June this year. Top 15 most launched local libraries are shown in Table 2. Since we are interested in locally installed packages, and each package usually contains a lot of libraries that should be included in the list which makes it hard to analyze, only one mostly launched library is shown for each user.

Table 2. Top most used locally installed libraries on Lomonosov-2 supercomputer.

| Library path                                                                 | Number of launches |
|-----------------------------------------------------------------------------|--------------------|
| <USER1>/PROJECT/ergo_project/repos/chunks_and_tasks/source/libcht.a          | 731                |
| <USER2>/soft/netcdf4-HDF5-serial/add/lib/libz.so.1.2.8                       | 576                |
| <USER3>/OpenFOAM/OpenFOAM-5.0/platforms/linux64GccDPInt32Opt/lib/libOpenFOAM.so | 495                |
| <USER4>/ww3/external/lib/libz.a                                             | 422                |
| <USER5>/gromacs/src/gmxlib/libs/libgmx_d.so.6.0.0                            | 398                |
| <USER6>/gcc-5.4/lib/gcc/x86_64-unknown-linux-gnu/5.4.0/crtbegin.o          | 291                |
| <USER7>/progs/babel/build/lib/libopenbabel.so.5.0.0                         | 284                |
| <USER8>/opt_debug/lib/libnetcdf.so.7.2.0                                    | 225                |
| <USER9>/gromacs-5.1.4/build/lib/libgromacs_mpi.so.1.4.0                    | 209                |
| <USER10>/Soft/third_party_libs/clapack/LIB/libblas.a                        | 173                |
| <USER11>/software/plumed-2.3.2/conf/test-main.o                             | 150                |
| <USER12>/FATHOM_DIR_OMPI/zlib/zlib-1.2.8/build/lib/libbz.so.1.2.8            | 131                |
| <USER13>/boost_1_65_1/../../lib/libboost_mpi.so.1.65.1                      | 125                |
| <USER14>/NAMD_2.10_Source/charm-6.6.1/mpi-linux-x86_64/tmp/test.o           | 111                |
| <USER15>/gromacs-2016.4/build-double/lib/libgmock.a                         | 84                 |

**Figure 3.** Average CPU load for each NAMD run by user 16, sorted by program start time.
It can be seen from this table that 3 different versions of GROMACS package (which is one of the most used packages on Lomonosov-2 supercomputer) are installed locally. Generally available GROMACS version is older – it seems like we should update it or install several versions for users to choose from. Also, there are 2 users that use NetCDF library which is currently not generally available on our supercomputer. Similar reasoning is also suitable for other packages and libraries on the list like NAMD, OpenFOAM, Boost, etc. All this information can be very helpful for administrators since it helps to understand what supercomputer users need in practice in terms of software packages.

2.3. Finding anomalies in the supercomputer job flow

The last approach stands somewhat aside from the first two: in this case we also study the entire range of jobs executed on the supercomputer, but the main goal is not to analyze the overall job flow statistics, or statistics for individual software packages, but to find individual applications with certain properties. Specifically, the purpose of this approach is to find individual launches with abnormally low efficiency in the overall job flow. In this case, an “anomaly” is understood not in absolute terms, but relative to the overall structure of the job flow. This way we can detect “complex” cases of inefficient behavior – low-efficiency launches that are difficult or impossible to trace using absolute threshold values. This approach uses the same system monitoring data for input as the other approaches previously described.

Since the criteria for abnormal behavior cannot be defined in advance (specifically because they will be different for job flows on other supercomputers), it was decided to use machine learning methods for this approach, namely classification methods. Each job from the overall job flow is analyzed and classified as “normal” (no efficiency issues found), “abnormal” (clearly exhibiting efficiency issues), or “suspicious” (no clear understanding of whether the job is behaving abnormally, a more detailed study is needed). It should be noted that supervised learning methods are used – the original training set is prepared and classified by hand, which is then used to train the classification method to identify the properties of a given job flow.

We considered several options for addressing this task. Initially only the integral values were considered for each property and each job [10] – just as in the two previous approaches described above. This method demonstrated a relatively high accuracy of identifying abnormal and suspicious launches, but it did have one serious drawback: it was impossible to obtain any additional information on abnormal behavior detected – when it started during the execution, its potential root causes, etc.

That is why we decided to try a different option. Execution time of each job is divided into smaller intervals. To identify intervals, we use an intellectual method that tries to detect substantial changes in the dynamic behavior of the program. Isolating such changes into separate intervals, we can accurately describe each of them using integral characteristics. Every interval is then classified separately, based on the monitoring data falling within the given interval. After each interval in the job is classified (as normal, abnormal or suspicious), the entire job is classified based on the interval information. The classification process is described in more detail in [10].

This classifier is currently working on the Lomonosov-2 supercomputer on a permanent basis, collecting statistics and notifying system administrators daily of any abnormal tasks. A brief description of the statistics collected by the developed classifier since it was launched into fully operational mode on the supercomputer (May 2017) is shown below.

A total of 251 abnormal and 5,882 suspicious jobs were detected during this period, which consumed 0.5 and 9.0 million core-hours respectively. For comparison – one day on the Lomonosov-2 supercomputer is 0.5 million core-hours; that is, all abnormal jobs have taken up about one day of supercomputer operation during the entire observation period. It should be noted that suspicious jobs have consumed a lot of computing time – about 18 days of the entire supercomputer operation; however, this suspicious behavior can be caused by a wide range of reasons. In particular, it is quite likely that the used algorithm simply cannot be implemented efficiently on the given supercomputer, so suspicious behavior may actually be normal, even though our detection of inefficient execution was correct.
Abnormal jobs clearly represent the most relevant area for analyzing supercomputer utilization efficiency—they waste computing resources, so their number should ideally be kept to a minimum. Figure 4 shows the total core-hours spent on abnormal jobs, with a breakdown by users. A total of 31 users executed abnormal jobs (and 72 had suspicious jobs). It can be seen that the first 4 users stand out from others. Even though their absolute values of consumed core-hours are not that high, this analysis warrants a discussion of the results with the users in question, to avoid wasting computing resources in the future.

![Figure 4](image)

Figure 4. List of top users by the amount of core-hours spent on abnormal jobs.

3. Conclusion
In this article, we described three different approaches to analyzing supercomputer resource utilization efficiency based on system monitoring data, which are being developed at the MSU Research Computing Center.

The first approach is based on collecting and studying statistics across the entire job flow, which helps analyze the overall supercomputer resource utilization efficiency. In particular, this helps identify and analyze specific classes of jobs and the distribution of values for various characteristics to study the job flow structure.

The second approach focuses on studying more detailed statistics on the usage of application packages. Since these packages are becoming increasingly common in practice, we need to analyze and control their utilization efficiency. We addressed this task using XALT, a tool that helps track what application and system software packages were used in each job launch.

The third approach is based on detecting individual abnormal launches in the overall job flow. Its goal is to find jobs that substantially deviate from the standard behavior on a given supercomputer. Namely, this approach helps detect jobs with abnormally low execution efficiency. This task is accomplished by analyzing the dynamic characteristics for every job using machine learning techniques. Each job is divided into a set of time intervals; each interval is classified as normal, suspicious or abnormal independently. Then a classification of the whole job is performed based on interval classification results.

Each of these approaches enables us to comprehensively analyze supercomputer resource utilization efficiency from different perspectives. In the future, we plan to further explore each of these approaches, eventually merging the developed methods into a single tool aimed at supercomputer utilization efficiency analysis. One of the other main goals is to provide supercomputer users with the results of conducted efficiency analysis for their jobs, so they can also carry out the study by themselves if needed and provide us with valuable feedback on the correctness of the obtained results.
Acknowledgments
The results described in all sections except Subsection 2.3 were obtained in the Lomonosov Moscow State University with the financial support of the Russian Science Foundation (agreement N 17-71-20114). The research presented in Subsection 2.3 was supported by the Russian Foundation for Basic Research (No. 16-07-00972).

References
[1] Voevodin V and Voevodin V 2016 Efficiency of Exascale Supercomputer Centers and Supercomputing Education High Performance Computer Applications: Proceedings of the 6th International Supercomputing Conference in Mexico (ISUM 2015) (Springer, Cham) pp 14–23
[2] Infrastructure monitoring system Nagios URL: https://www.nagios.org/ (accessed on 22.11.2017)
[3] Stefanov K, Voevodin V, Zhumatiy S and Voevodin V 2015 Dynamically Reconfigurable Distributed Modular Monitoring System for Supercomputers (DiMMon) Procedia Comput. Sci. 66 pp 625–34
[4] (in Russian) Mamaeva A and Voevodin V 2017 Methods for statistical analysis of large supercomputer job flow Supercomput. Third Russ. Supercomput. Days, RuSCDays 2017, Moscow, Russ. Sept. 25–26, 2017, Revis. Sel. Pap. (MSU Publisher) pp 788–99
[5] Jones M D et al. 2017 Workload Analysis of Blue Waters Preprint arXiv:1703.00924
[6] Evans T, Barth W L, Browne J C, DeLeon R L, Furlani T R, Gallo S M, Jones M D and Patra A K 2014 Comprehensive Resource Use Monitoring for HPC Systems with TACC Stats 2014 First International Workshop on HPC User Support Tools (IEEE) pp 13–21
[7] Abraham M J, Murtola T, Schulz R, Páll S, Smith J C, Hess B and Lindahl E 2015 GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers SoftwareX 1–2 pp 19–25
[8] Agrawal K, Fahey M R, McLay R and James D 2014 User Environment Tracking and Problem Detection with XALT 2014 First International Workshop on HPC User Support Tools (IEEE) pp 32–40
[9] Phillips J C, Braun R, Wang W, Gumbart J, Tajkhorshid E, Villa E, Chipot C, Skeel R D, Kalé L and Schulten K 2005 Scalable molecular dynamics with NAMD J. Comput. Chem. 26 pp 1781–802
[10] Shaykhislamov D 2016 Using Machine Learning Methods to Detect Applications with Abnormal Efficiency Supercomputing. Second Russian Supercomputing Days, RuSCDays 2016, Moscow, Russia, September 26–27, 2016, Revised Selected Papers (Springer, Cham) pp 345–55