Statistics of Software Package Usage in Supercomputer Complexes

Pavel Shvets, Vadim Voevodin, and Sergey Zhumatiy

Research Computing Center of Lomonosov Moscow State University, Moscow, Russia
vadim@parallel.ru

Abstract. Any modern high-performance computing system is a very sophisticated complex that contains a variety of hardware and software components. Performance of such systems can be high, but it reaches its maximum only if all components work consistently and parallel applications efficiently use provided computational resources. The steady trend of all recent years concerning large supercomputing centers – constant growth of the share of supercomputer applications that use off-the-shelf application packages. From this point of view, supercomputer software infrastructure starts to play a significant role in the efficient supercomputer management as a whole. In this paper we will show why the analysis of software package efficiency should be taken seriously, what is worth paying attention to, and how this type of analysis can be conducted within the supercomputer complex.

Keywords: parallel computing · high-performance computing · supercomputer · efficiency · efficiency analysis · software package · job flow · monitoring data

1 Introduction

Low efficiency of supercomputer system functioning has been an actual problem in high-performance computing field for many years. Over the years, many possible approaches and methods have been developed for analysis and optimization of both individual supercomputer applications and entire supercomputer functioning as a whole. Due to increasing large-scale use of this area, i.e. growing number of users of supercomputer systems, as well as the progress in the development of HPC packages, off-the-shelf software packages are more and more often used for solving different scientific problems. For example, around 500-1000 applications are run on Lomonosov and Lomonosov-2 supercomputers daily. Part of these applications is written by the users themselves with traditional parallel programming techniques: MPI, OpenMP, OpenCL, CUDA, TBB, etc. The other part is based on ready-to-use packages from different subject areas: Gromacs, Gaussian, FireFly, CP2K, Amber, VASP, OpenFOAM and many others, and the share of such applications is steadily growing. This being said, the vast majority of users also rely on the other types of program packages – system software, for example, various versions of compilers. And the peculiarities of their usage can also greatly affect the performance of user applications.

This leads to the fact that the task of studying the performance of applied application packages becomes more and more important, since both the speed of conducting particular scientific experiments and the overall efficiency of supercomputer system functioning depend on it. The experience of supporting and maintaining such systems accumulated in Research Computing Center of Lomonosov Moscow State University (RCC MSU) shows that such efficiency in many cases is unfortunately really low, which leads to computational resources being idle [1, 2]. It is necessary to develop and implement special software tools aimed at analyzing this particular aspect of supercomputer functioning. It is worth noting that similar researches were conducted earlier (for example, [3]), but they were originally intended to study only specific computing systems and were not portable.

Therefore, RCC MSU decided to study the statistics of software package usage in MSU Supercomputing Center. The main goal of this study is to understand how efficient and in-demand different software packages are on each machine. Using the examples of Lomonosov and Lomonosov-2 supercomputers, in this paper we show what information can be collected and how it can be obtained. First of all, this is of interest to management and administrators of supercomputer complexes, since it allows analyzing and improving the efficiency of the whole supercomputer. In particular, this helps to make a decision whether a particular package should be further used (e.g. if nobody uses it, there is no need to renew the license) or whether there is a need of more fine tuning or more detailed analysis of a particular package with very low usage efficiency. But such kind of information can be useful for common users as well, since it allows to make a choice in favor of more efficient package. For example, a user can compare the performance of two linear algebra packages installed on a particular cluster and choose a more suitable one, or check the performance of locally deployed proprietary solution.

The results shown in this paper were obtained in this Center, but the overall approach can be quite easily implemented in other supercomputing centers as well, which is a part of our plans for future.

2 Collecting Statistics on the Usage of Software Packages

To perform the analysis of package usage, it is necessary to collect information on all jobs running on the supercomputer; that is, we need to constantly monitor the overall supercomputer job flow. And it is necessary to collect, integrate and analyze different types of data:

1. general information on the job flow;
2. data on the efficiency of each particular job;
3. detailed information on the job itself that allows to classify used package and associate it with obtained efficiency data.

Information on the job flow describes when and on which nodes each job was launched. Such information can be easily collected from the resource manager;

this process is already well known and implemented. In MSU Supercomputing Center the Slurm manager [4] is being used.

The data on job execution efficiency is collected via different monitoring systems. These systems provide information on the dynamics of job execution as a set of characteristics which describe CPU load, intensity of communication network usage, number of active processes on the node, presence of GPU activity, etc. This set of required data can differ and depends on the desired criteria of package usage efficiency. Depending on this, different monitoring systems (Collectd [5], Zabbix, Nagios, Zenoss, Cacti or DiMMon system [6] being developed in RCC MSU) may be more appropriate. A set of proprietary tools is currently used in MSU Supercomputing Center for collecting needed monitoring data. Since these are the first steps in this study, for now we are interested only in the most general indicators of computational resource usage – CPU and GPU user load as well as average system load (loadavg), but in the future we also plan to analyze such characteristics as the intensity of memory subsystem and network usage.

Note that in order to collect statistics on the package usage, it is sufficient to collect and store only integral characteristics for each launch, and this task can be successfully solved by existing monitoring systems. Moreover, this information was being stored for a while in MSU Supercomputing Center for other purposes [2], so this data was already available for our research.

Thus, in this study the main difficulty in collecting the required data was to solve the last issue – to collect detailed information on each running job about the packages used. We were interested in information about used compilers, application packages, linked libraries, etc. It was decided to use XALT tool [7, 8] developed by Texas Advanced Computing Center (TACC) for solving this issue. This tool was chosen due to several reasons:

1. it is an off-the-shelf system software product that is ready for use;
2. it has all the functionality needed for our purposes;
3. it is open source.

This tool collects the data by intercepting calls to the linker and job launcher. For this purpose, wrappers are created that override standard front-end scripts. Each time a user links or launches his program, XALT stores all necessary information on performed actions in user home directory in JSON format. In particular, used compiler and its options, list of linked static libraries, full path to the compiled file and compilation time are stored during linking operation. When the job launcher is used, XALT stores the name on executed file, the way it was launched, used MPI library version, linked dynamic libraries, values of environmental variables and many other useful information. Other ways of storing the data (besides writing JSON files in home user directories) are possible – using syslog or direct connection to the database.

The collected data periodically (or by administrator request) is stored in a single MySQL database which accumulates all statistics on the executed jobs. After that, all that needed is to compose correct SQL query in order to obtain corresponding data slice required for analysis. Though it is necessary to learn

the (quite simple) database format in order to manually create needed SQL queries, many ready-to-use examples of such queries enough to perform first quick analysis are provided in the XALT manual. This MySQL database is built to aggregate data from different clusters, so it is quite easy to analyze the data about the whole supercomputer complex as well as about one particular machine.

3 Analysis of Results Obtained in MSU Supercomputing Center

The specified XALT tool was installed in MSU Center in June 2017, so some information is available only from that moment. However, during XALT installation and testing it was discovered that XALT scripts for defining software packages can be applied to all previously collected data for executed jobs, so the results presented in Chapters 3.1 and 3.2 are based on data since the beginning of year 2017. Below are the main interesting results that were obtained in this study.

3.1 Statistics on the Popularity of Application Packages

To ensure the efficiency of supercomputing centers we need to move from "our feelings what is needed" to real facts. From this point of view, even a simple general statistics on the frequency of using application packages is already very useful. Figure 1a and 1b show the distribution of the frequency of running certain packages on Lomonosov and Lomonosov-2 supercomputers. The frequency of package usage is shown as a share (in percents) of the total number of runs on a supercomputer. It can be seen that in both cases top 3 places in total provide more than 50% of all package launches, but there is only one package that is in both top 3 lists – Firefly package for quantum chemical calculations. In the whole list some overlaps can be seen: for example, packages like VASP, Gromacs, NAMD and DL_POLY are frequently used on both systems.

In general statistics on both supercomputers is quite different, and this illustrates the fact that there is no big reason to compare such results on different machines. This is because the architecture of supercomputers, as well as the overall research interest of users (and therefore the package usage) can vary greatly in each case. The main goal of analyzing such kind of information is to get the basic understanding of package usage on each particular cluster independently.

Now let's look at the use of packages from the other point of view and evaluate the CPU hours consumed by various packages (Fig. 2a and 2b for Lomonosov and Lomonosov-2 supercomputers correspondingly). The picture changes significantly; for example, the leading position has changed – in both cases the first place is now occupied by the Gromacs package, which is used to model physicochemical processes in molecular dynamics. It can also be noted that the DL_POLY package is not represented on these charts, although it was in top lists in Fig. 1. This indicates that this package is launched very often, but on average it requires not so many computational resources.

Taking into account that dozens of packages are installed and used on supercomputers, such analysis helps to prioritize their installation, updating and fine-tuning. Together with similar data, but specified for certain users, this information shows the actual demand for particular packages.

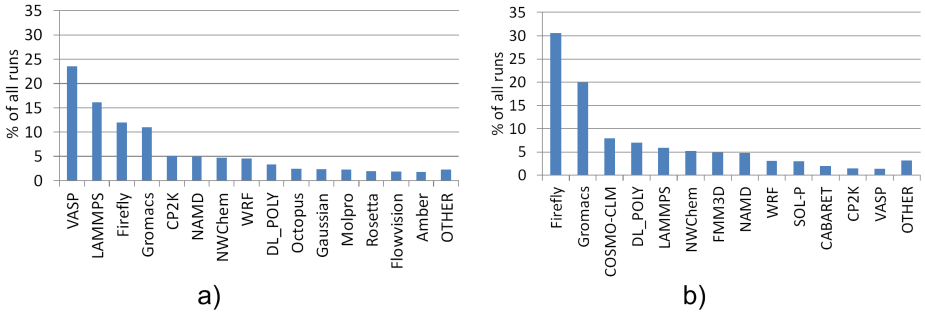

Fig. 1. The distribution of the frequency of running certain packages on Lomonosov (a) and Lomonosov-2 (b) supercomputers (a share of total number of runs is shown)

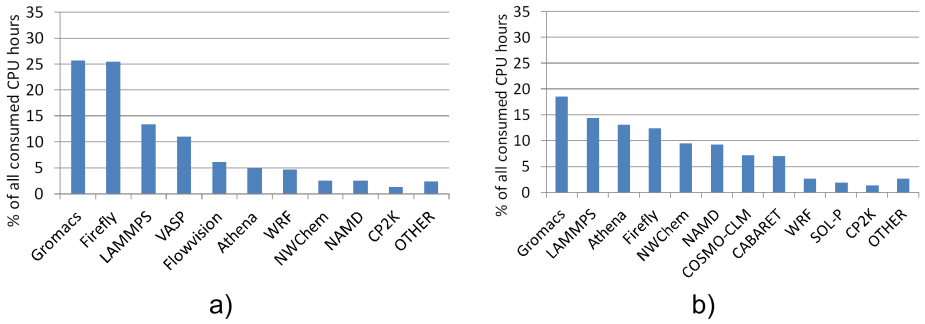

Fig. 2. The distribution of the CPU hours consumed by various packages on Lomonosov (a) and Lomonosov-2 (b) supercomputers (a share of total amount of consumed CPU hours is shown)

3.2 Studying the Efficiency of Application Package Usage

Another huge part of information is provided by the binding of dynamic characteristics describing the efficiency of parallel job execution to specific software packages. Figures 3a and 3b describe average value of loadavg for jobs that use different packages on Lomonosov and Lomonosov-2 supercomputers. Loadavg

parameter estimates the number of processes on a node that are ready for execution, which describes degree of parallelism in an application from a system point of view. If an application makes the best use of hardware platform capabilities, this value is usually equal to the number of cores on a node. Nodes of Lomonosov supercomputer have two 4-core or 6-core processors, nodes of Lomonosov-2 – one 14-core processor. The corresponding optimal loadavg values based on the number of physical cores are shown with horizontal lines in Fig. 3. Note that processors in both systems have Hyper-Threading technology enabled, which doubles the number of logical cores, so potentially optimal loadavg value can be twice as high.

It can be seen in Fig. 3 that many packages show high loadavg values meaning that they use available resources quite actively, although there are notable exceptions like Molpro package used on Lomonosov or VASP used on Lomonosov-2. It is also interesting to note that Gromacs package on Lomonosov seems to use Hyper-Threading quite often (its loadavg value is higher than the number of physical cores on a node), since apparently this allows further performance increase.

Such data is very important: it reflects the accumulated experience and helps to understand the way the packets are used by the computing community. This is necessary as well for evaluating the performance of the package on a particular supercomputer, and this also helps to make adjustments to the use of the particular package by individual users.

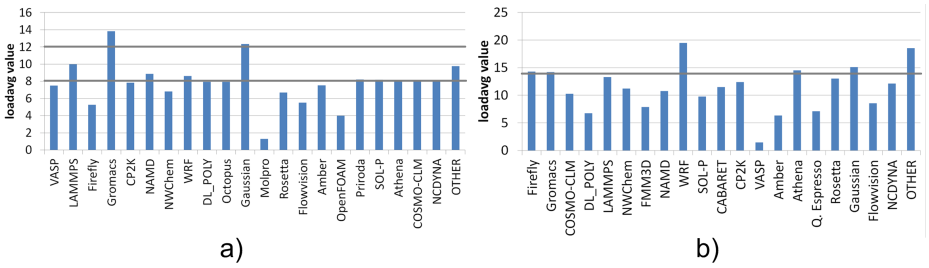

Fig. 3. Average loadavg value for different packages on Lomonosov (a) and Lomonosov-2 (b) supercomputers

The degree of parallelism is defined not only by the number of used cores on a node, but also by an overall number of used nodes. Our practice shows that, in an attempt to reduce the running time of applications, users increase the number of nodes, not always paying attention to the limited scalability of applications. As a result – part of supercomputer resources is wasted. A good method for evaluating efficiency is to analyze how CPU cores are loaded by each package. An example showing average core load for Lomonosov and Lomonosov-2 supercomputers is shown in Fig. 4.

Comparison of Fig. 3 and 4 shows that loadavg and core load characteristics correlate well enough – relative change of loadavg for different packages is similar to corresponding change in the values of the core load. This indicates that on average each packet equally loads each active process. However, there are exceptions – the loadavg value for WRF package on Lomonosov-2 supercomputer is much higher than the values for other packages, but its core load is at average level. Perhaps this is due to very active memory usage or data transfer using communication network, but it is also likely that this package uses resources rather inefficiently. We should notice that this is true only for this particular system; the behavior of WRF package on other systems can differ.

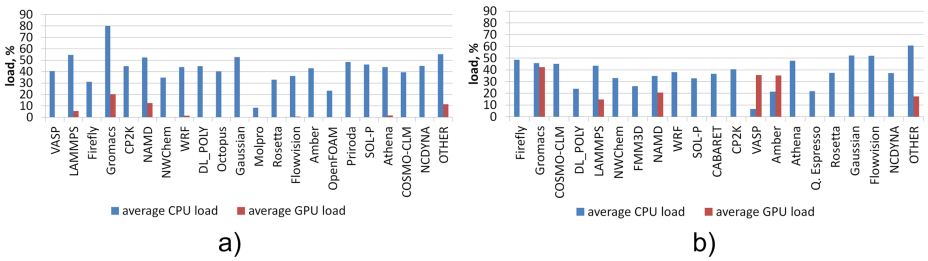

Fig. 4. Average core load for different packages on Lomonosov (a) and Lomonosov-2 (b) supercomputers

In general, low core load can be explained by different reasons: bad package scalability, errors made by user or his desire to allocate more memory per process on a node. There are many possible reasons, and each case must be considered separately. In any case, such information should be available to system administrators to control the situation.

Similarly, package usage efficiency can be analyzed using other monitoring data, for example, GPU usage load (also shown in Fig. 4). The package can be optimized for GPU usage, but only real-life statistics can show, how effectively someone actually uses it and whether he forgot to specify the "use the GPU" option. In particular, Fig. 4b explains low loadavg value for VASP package: the graph shows that this package uses GPU very actively, and apparently the computations are almost not performed on the CPU side, so it does not need to start a lot of CPU of processes.

Many packages are sensitive to hardware parameters, and the analysis of dynamic characteristic values helps to optimize both overall functioning of supercomputer complex and behavior of particular user applications. In this regard, it is very important to detect anomalous values of dynamic characteristics in time, which again may happen due to different reasons like incorrect package installation by the system administrator or incorrect package usage by the common user. There are many examples of anomalies: excessively high or low loadavg value, high core idle value, large number of cache misses, low GPU load,

high intensity of I/O operations, high fluctuation of dynamic characteristics, etc, but all of them indicate potential problems with the efficiency of supercomputer complex. In this research it is planned to develop a methodology for detecting such anomalies in future.

3.3 Distribution of Used Compilers

Results shown earlier relate directly to the job launch process. But linking process is also of interest. In particular, one of the main questions – which compilers are most in demand? Figure 5 shows the distribution of compilers based on the number of uses on Lomonosov-2 supercomputer. It can be seen that GNU compilers are used much more often than Intel compilers. For instance, GNU C compiler is used in almost half the cases. Interestingly, Fortran language, according to this statistics, is used almost as often as C language (44% vs 50%), and C++ language is far less common (only 6%).

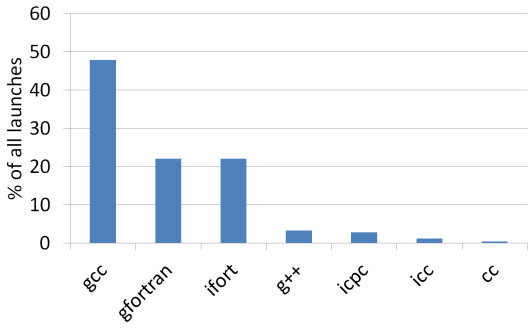

Fig. 5. The frequency of using different compilers on Lomonosov-2 supercomputer (a share of total number of launches is shown)

4 Conclusion and Future Work

This paper describes the research being conducted in RCC MSU aimed at studying the efficiency of software package usage in supercomputer systems. This research on the given examples shows how such analysis can be conducted within the majority of modern supercomputers, and what information can be obtained on its basis. We engage the management and system administrators of different supercomputer centers to perform such study, since this helps to improve the efficiency of the supercomputer in general, and also help users to select more suitable software packages for their tasks.

This research is based on the analysis of different types of information: data on the overall job flow that helps to determine when and on which nodes each

job was launched; data on the efficiency of each particular job; detailed data on the package usage. The task of collecting first two types of data is easy enough and was already implemented in MSU Supercomputing Center, so only the last question was to be solved. XALT package developed in TACC was used for this purpose. This package intercepts calls to the linker and job launcher, which allows to save required information each time they are used.

As a result, various statistics on package usage on Lomonosov and Lomonosov-2 supercomputers were collected. In particular, the distribution of packages by the number of launches, consumed CPU hours, average CPU load and loadavg, as well as the frequency of using different compilers was analyzed. Such analysis allows to perform holistic estimation of the popularity and usage efficiency of different software packages.

The analysis of the results described in this paper is just the first step in this direction. A variety of services intended to help common users and system administrators to optimize their work can be developed based on this data, which will be the focus of further work in this study.

The data on average dynamic characteristics for a package will help the user to correctly interpret the values obtained by his applications. One possible way of implementation – automatically generated notification based on the results of application execution, which contains a comparison of the current launch characteristics with the average characteristics for this package. This can be a mass service for all users with built-in analytics for each application package. This service can also be configured to compare only with the previous launches of the same user, so a deviation from "standard" values specific for this particular user can be clearly seen.

System administrators also require similar services that constantly analyze the integral characteristics of different packages. Such services can help to find out when a package reinstallation, software update or issue of a "How to efficiently use a package" guide is needed. Slightly more complex options are searching for more efficient alternative packages (based on price and efficiency criteria) or planning a software update budget for the next year.

Acknowledgments. The results described in all sections except Section 2 were obtained in the Lomonosov Moscow State University with the financial support of the Russian Science Foundation (agreement № 17-71-20114). The research presented in Section 2 was supported by the Russian Foundation for Basic Research (№ 17-07-00664).

References

1. Vladimir Voevodin and Vadim Voevodin. Efficiency of Exascale Supercomputer Centers and Supercomputing Education. In *High Performance Computer Applications: Proceedings of the 6th International Supercomputing Conference in Mexico (ISUM 2015)*, pages 14–23. Springer, Cham, 2016.

2. Dmitry Nikitenko, Vladimir Voevodin, Alexey Teplov, Sergey Zhumatiy, Vadim Voevodin, Konstantin Stefanov, and Pavel Shvets. Supercomputer Application Integral Characteristics Analysis for the Whole Queued Job Collection of Large-scale HPC Systems. In *Parallel Computational Technologies (PCT'2016)*, pages 20–30, 2016.
3. Matthew D. Jones, Joseph P. White, Martins Innus, Robert L. DeLeon, Nikolay Simakov, Jeffrey T. Palmer, Steven M. Gallo, Thomas R. Furlani, Michael Showerman, Robert Brunner, Andry Kot, Gregory Bauer, Brett Bode, Jeremy Enos, and William Kramer. Workload Analysis of Blue Waters. mar 2017.
4. Slurm Workload Manager. URL: http://slurm.schedmd.com. Cited 08 Aug 2017
5. Collectd – The system statistics collection daemon. URL: https://collectd.org. Cited 08 Aug 2017
6. Konstantin Stefanov, Vladimir Voevodin, Sergey Zhumatiy, and Vadim Voevodin. Dynamically Reconfigurable Distributed Modular Monitoring System for Supercomputers (DiMMon). *Procedia Computer Science*, 66:625–634, 2015.
7. Kapil Agrawal, Mark R. Fahey, Robert McLay, and Doug James. User Environment Tracking and Problem Detection with XALT. In *2014 First International Workshop on HPC User Support Tools*, pages 32–40. IEEE, nov 2014.
8. XALT tool homepage. URL: https://www.tacc.utexas.edu/research-development/tacc-projects/xalt. Cited 08 Aug 2017