Efficiency of Exascale Supercomputer Centers and Supercomputing Education

Conference Paper · April 2016

DOI: 10.1007/978-3-319-32243-8_2

2 authors:

Vladimir Voevodin
Lomonosov Moscow State University
65 PUBLICATIONS **591** CITATIONS
SEE PROFILE

Vadim Voevodin
Lomonosov Moscow State University
73 PUBLICATIONS **323** CITATIONS
SEE PROFILE

Efficiency of Exascale Supercomputer Centers and Supercomputing Education

Vladimir Voevodin(✉) and Vadim Voevodin

Research Computing Center, Moscow State University,
119234, Leninskie Gory, 1, bld. 4, Moscow, Russia
{voevodin,vadim}@parallel.ru

Abstract. The efficient usage of all opportunities offered by modern computing systems represents a global challenge. To solve it efficiently we need to move in two directions simultaneously. Firstly, the higher educational system must be changed with a wide adoption of parallel computing technologies as the main idea across all curricula and courses. Secondly, it is necessary to develop software tools and systems to be able to reveal root causes of poor performance for applications as well as to evaluate efficiency of supercomputer centers on a large task flow. We try to combine both these two directions within supercomputer center of Moscow State University. In this article we will focus on the main idea of wide dissemination of supercomputing education for efficient usage of supercomputer systems today and in the nearest future as well as describe the results we have reached so far in this area.

1 Introduction

Computing technologies quickly evolve penetrating constantly to new areas of our life. A very illustrative example is science. Until recently, science was founded on two components: theory and experiment, but now many scientists hardly will do their research without numerical experiments, which have become the third pillar of the science. Computing experiments are everywhere and more and more researchers start using benefits of computer sciences in their everyday scientific work. This is especially true in large supercomputer centers, which attract large groups of users from different areas of the science.

Performance of the best supercomputers in the world is measured in Petaflops providing unprecedentedly powerful scientific instruments for research. At the same time, the efficient usage of all opportunities offered by modern computing systems represents a global challenge. Using full potential of parallel computing systems and distributed computing resources requires new knowledge, skills and abilities. Most users did not get a proper education in computer architecture, numerical methods and parallel programming technologies therefore efficiency of their applications and entire supercomputer centers is critically low.

This is a common serious problem, which is typical for all large supercomputer centers. To solve it efficiently we need to move in two directions simultaneously. Firstly, the higher educational system must be changed with a wide adoption of parallel

I. Gitler and J. Klapp (Eds.): ISUM 2015, CCIS 595, pp. 14–23, 2016.
DOI: 10.1007/978-3-319-32243-8_2

computing technologies as the main idea across all curricula and courses. But this cannot be done immediately or quickly. Simultaneously with the transformation of education it is necessary to develop software tools and systems to be able to reveal root causes of poor performance for applications as well as to evaluate efficiency of supercomputer centers on a large task flow during any period of time.

We try to combine both these two directions within supercomputer center of Moscow State University. Currently the supercomputer center includes two flagship supercomputers with peak performance of 1.7 and 2.5 Petaflops, which are used by more than 700 scientific groups from MSU, institutes of the Russian academy of sciences and Russian universities. Fast development of software tools suite ensuring efficiency of the supercomputer center goes with active incorporation of supercomputer education ideas into the education process of the university.

The importance of education in the area of parallel, distributed and supercomputer technologies is understood by the international education community. A number of large-scale projects can be cited which try to elaborate on and structure this area of education, offering recommendations on preparing teaching materials and providing examples of training courses. These projects include:

– the Russian Supercomputing Education national project [1, 2], where the Moscow State University was the principal coordinator,
– activities related to developing a computer science curricula (Computer Science 2013) by the ACM and IEEE-CS international societies [3],
– The Curriculum Initiative on Parallel and Distributed Computing project, supported by NSF/IEEE-TCPP [4].

Important recommendations in this area are given as part of the SIAM-EESI project for developing education and research in the area of Computational Science and Engineering [5].

To ensure the efficiency of large supercomputing centers, we use an approach based on a suite of interrelated software systems, technologies and instruments. It is designed for a coordinated analysis of the entire hierarchy of a supercomputing center: from hardware and operating system to users and their applications. This analysis requires going through both system level (CPU-related data, network performance, I/O operations, memory subsystem, etc.) and the upper level of entire supercomputing center (users, applications, queues, task flows, supercomputer sections, etc.).

In this article all software systems mentioned above will be shortly described. And at the same time, we will explain why supercomputing education is tightly connected with all these systems. It should be specially mentioned that the paper is written in a quite untraditional manner. There is no “related works” section, there is no description of future plans. It was made intentionally. In the present short communications we tried to focus on the main idea of wide dissemination of supercomputing education for efficient usage of supercomputer systems today and computing systems in the nearest future.

2 Software Tools and Efficiency of Supercomputers

Most of the systems use system-level data which is collected by *the total monitoring system*. We call it “total monitoring” since we need to know almost everything about dynamic characteristics of supercomputers to be sure in their efficiency at each time point. Ensuring the efficient functioning of a supercomputing center requires monitoring absolutely everything that happens inside the supercomputer, and that task alone requires sifting through tons of data. Even for the relatively small “Lomonosov” supercomputer at the Moscow State University (1.7 Pflops peak, 100 racks, 12 K nodes, 50 K cores) [6], the necessary data arrive at the rate of 120 Mbytes/s (about 30 different metrics analyzed for each node, measured at the frequencies of 0.01–1.00 Hz). It means 3+ Pbytes for 365 days of a year. This is an incredibly large amount of data which must be collected with a minimum impact on application performance. Nevertheless, a total monitoring system can be built by reaching a reasonable compromise on two key issues: what data must be stored in the database, and when should it be analyzed.

The monitoring system gives us very useful and detailed information about features of applications and about the supercomputers as well. For example, Fig. 1 presents an average CPU_load parameter across entire 6000-core supercomputer “Chebyshev” for a particular period of time. At the same time, Fig. 2 shows the same CPU_load parameter, but for a particular application. Information from the Fig. 1 is very important for system administrators. To reveal reasons of low efficiency sysadmins should fulfill a root cause analysis what is a serious problem since so many hardware and software components can directly or indirectly be involved and influent system efficiency.

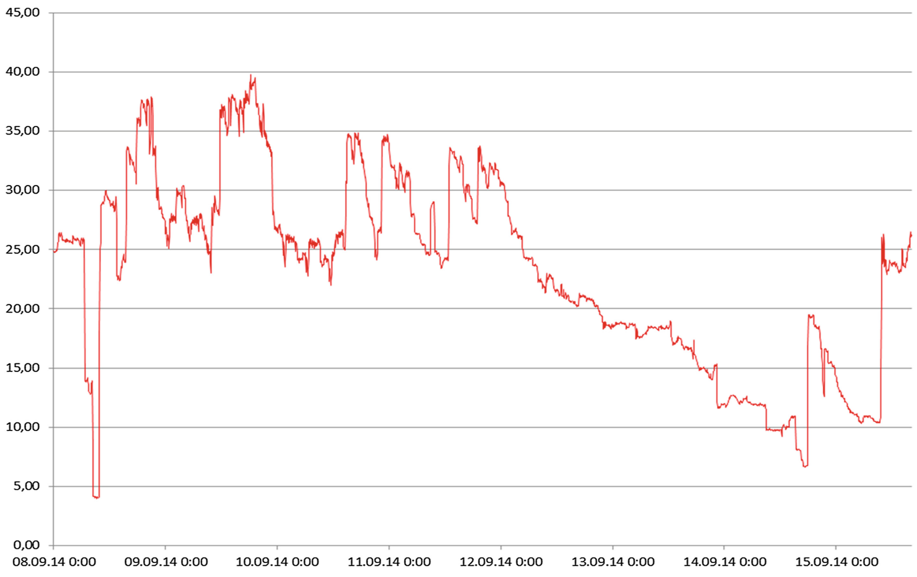

Fig. 1. Overall efficiency of “Chebyshev” supercomputer.

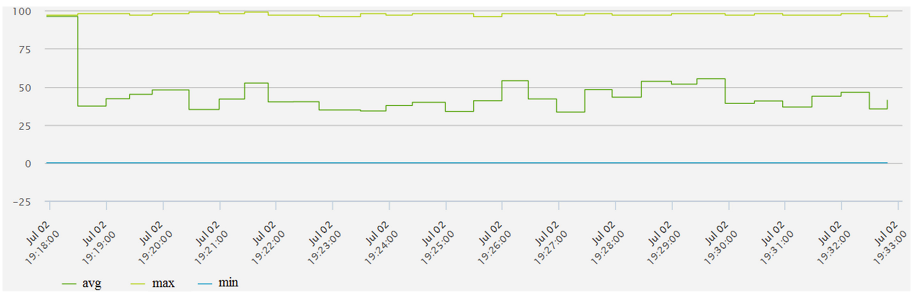

Fig. 2. CPU load timeline graph for an application.

Figure 2 represents one of the most important parameters of an application and is intended for a user. An average CPU load is very unbalanced varying from 0 to 100 % across all the cores during runtime. To improve efficiency of the application the user need to fulfill a supercomputing co-design analysis starting from optimization of the code up to possible redesign of an algorithmic approach. Practice shows that this is a tough problem for users and the reason of this situation is clear. How many of students around you have seen dynamic parameters of their parallel codes? How many of them are familiar with the notion of supercomputing co-design? This is really a question of supercomputing education, since if we examine existing curricula in computer science of the most universities we won't find much about parallel computing, efficiency, scalability, etc.

A modern supercomputing center is not only about managing supercomputers per se – this function has been studied well enough already – but about efficiently organizing a multitude of related issues: managing software licenses, user quotas, project registration and maintenance, technical support, tracking warranty and post-warranty service and repairs, monitoring of software components usage and many other tasks. All these issues are closely linked to one another and, given the huge number of components, it is clear how hard it is to efficiently organize work flows and maintain the entire supercomputing center in an up-to-date condition.

The OctoShell system combines data on all critical components for efficiently operating a supercomputing center, describing their current state and connections between components: the hardware and software components being utilized, users, projects, quotas, etc. Can this information be used to improve efficiency of large supercomputer centers? Yes, in can be used in many ways. Figure 3 presents current statistics on different software components usage for "Lomonosov" supercomputer. It should immediately draw attention of sysadmins and management of the supercomputer center due to at least two reasons. Firstly, the most popular packages and libraries must be carefully installed and highly optimized (FFTW, Intel MKL, BLAS, etc.…). Secondly, tutorials and trainings on highly demanded packages in specific areas should be provided (Gromacs, VASP, etc.…) otherwise efficiency of the supercomputer will certainly be low. Again, practice shows lack of proper education, even for experienced users, so continuous education is a must for advanced users as well as diverse educational activities is a must for large supercomputer centers.

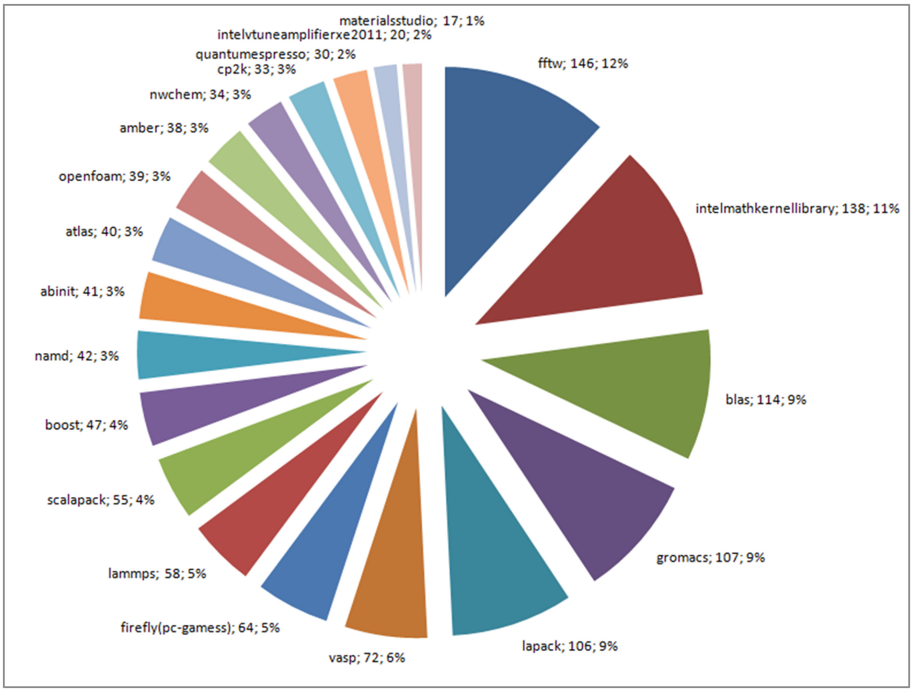

Fig. 3. Top used software components for “Lomonosov” supercomputer.

The practice of supercomputing center maintenance defines a set of strict technology and tool requirements for supporting the functioning of supercomputer systems. This includes maintaining high performance for the supercomputing infrastructure, constant monitoring of potential emergencies, continual performance monitoring for all critical software infrastructure modules, automatic decision-making on eliminating emergency situations and increasing the performance of supercomputer operations, guaranteed operator notification about the current status of the supercomputer and actions taken by the automatic maintenance system, and others. Until all these requirements are met, neither the efficient operation of a supercomputing center, nor the safety of its hardware can be guaranteed.

The goal of *the Octotron project* is to design an approach that guarantees the reliable autonomous operation of large supercomputing centers. The approach is based on a formal model of a supercomputing center, describing the proper functioning of all its components and their interconnections. The supercomputer continually compares its current state with the information from the model. If practice (monitoring data of the current supercomputer state) deviates from theory (the supercomputer model), Octotron can perform one of the predefined or dynamically selected actions, such as notifying the operator via email and/or SMS, disabling the malfunctioning device, restarting a software component, displaying an alert on the systems administrator screen, etc. No human is capable of monitoring millions of components and processes inside a supercomputer, but the supercomputer itself can do this. Importantly, this approach guarantees not only reliable operation of the existing fleet of systems at a supercomputing center, but also ensures maintenance continuity when moving to a new generation of machines. Indeed, once an emergency situation arises, it is reflected in the model, along with the root causes and symptoms of its existence, and an adequate reaction is programmed into the model. It may never happen again in the future, but if it does, it will immediately be intercepted and isolated before it has any effect on the supercomputer operation.

The thesis similar to "no human is capable to control computing systems" certainly should be reflected in modern curricula. Courses on computer architectures, operating systems, system programming, and parallel programming technologies should clearly explain high complexity of computing systems. This is important today but it will be much more important in the future [7]. Like supercomputers, mobile devices are already slightly parallel today but they will be highly parallel in several years becoming more and more complex. What we see in supercomputing area today forces us to change computer education in general to make our students well prepared for living in the hyper-parallel computer world in the future.

The main purpose of *the Situational screen* is to give system administrators full and prompt control over the state of the supercomputing center. The screen provides detailed information on what is happening inside the supercomputer, and provides updates on the status of hardware and system software, task flow, individual user activity and/or the performance of individual supercomputing applications. The situational screen and its underlying instruments are designed to meet a number of strict requirements: the need to reflect all key performance parameters of the supercomputing systems, grant of complete control over their status, scalability, expandability, configuration and minimization of the impact of the situational screen on the supercomputer performance.

Based on many years of experience in administrating supercomputer systems, we can point out the most important components which the situational screen must reflect:

- supercomputer hardware: computing modules, hosts, secondary servers, storage, networks, engineering infrastructure elements. Here it is important to know both the overall characteristics and the status of individual components;
- system software: the status for the entire supercomputer and individual nodes, conducting audits with specified criteria for any number of nodes;
- user task flows: tasks queued, tasks being executed, current and historical data, statistics on various partitions of the supercomputer, statistics on application package usage;
- activity of a particular user: what is being executed now, what is queued, overall performance over a recent time period, etc.;
- status of any working application: statistics on performance, locality, scalability and system-level monitoring data.

The OctoStat statistics collection system provides highly valuable information on the performance of supercomputing systems with regards to the flow of tasks queued for execution. Task queue length, waiting time before execution, the distribution of processors needed for applications of different classes, statistics by partitions of the supercomputer, by users or application packages, intensity of application flow for execution at different times… All of this – and many other metrics – need to be analyzed and used to optimize quotas, priorities and strategies for distributing valuable supercomputing resources. Moscow State University supercomputing center uses OctoStat to generate daily statistics on the supercomputer usage, enabling prompt decision-making.

One key issue when analyzing the efficiency of supercomputing centers is the user application runtime performance analysis. Today this is a crucial question (and it always was). According to our estimates only a few percent of users are able to analyze properly efficiency of their applications, and most often they do not think about efficiency at all. Performance of a supercomputer on real-life applications today is quite low, amounting to just a small percentage of the peak performance characteristics. As parallelism increases, this figure is bound to decline. There are a multitude of reasons for this decline in efficiency, and we use supercomputing system hardware and software monitoring data to identify them. A lot of data is required for the analysis: CPU load, cache misses, flops, number of memory references, Loadavg, IB usage, I/O usage, etc. This data is used to build a runtime profile for each application, which is presented as a set of timeline graphs demonstrating the changes in monitoring data during the application execution. This profile, along with a number of aggregate system characteristics (we call it *JobDigest of an application*) gives a good first estimate of the application performance and its features. If any issues are observed with the application overall performance, additional analysis of the monitoring data is performed to identify the causes. Particular attention is paid to analyzing application properties such as efficiency, data locality, performance, and scalability, which are extremely important for the supercomputing systems of the future.

But no matter how rich the set of tools we use, high performance is not achievable in principle with some applications. The reason is that the structure of these applications do not correspond to the specific architecture of the computational platform. From the standpoint of supercomputing resource efficiency, a lot depends on the properties of algorithms used by these applications. What are these properties? What should be discovered and expressed explicitly in existing algorithms when a new parallel architecture appears? How to ensure efficient implementation of an algorithm on a particular parallel computing platform? The idea of deep a priori analysis of algorithm properties and their implementations forms the grounds for a web-oriented system: *AlgoWiki, an open encyclopedia of algorithm features* [8], and this encyclopedia is intended to answer this sort of questions. The main purpose of AlgoWiki is to present a description of fundamental properties of various algorithms, giving a complete understanding of both their theoretical potential and the particular features of their implementations for various classes of parallel computing systems.

A description of algorithm properties and features in AlgoWiki consists of two parts. The first part describes the actual algorithms and their properties, while the second one is dedicated to a description of their implementation features on various software and hardware platforms. This distinction is intentional, to single out machine-independent properties of various algorithms, which determine the quality of their implementation on parallel computing systems, and describe them separately from a number of issues related to the subsequent stages of algorithm programming and program execution.

At first sight, the idea of such an encyclopedia seems simple, but it is a daunting and non-trivial task in reality. Three stages of the implementation process need to be analyzed for each item: (a) studying the properties of algorithms, (b) studying the properties of the resulting programs, and (c) comparing the identified properties with the hardware features. Throughout the process – all three stages – two principal properties which determine implementation quality need to be considered: resource of parallelism and data usage. Additional complexity is introduced by the fact that these two properties need to be studied in continuity between the stages and coordinated during each stage.

One issue of utmost importance is to ensure a complete description of the implementation process, determining control over the efficiency of both new parallel programs and those ported from one computing platform to another. In other words, all principally important issues affecting the efficiency of the resulting programs must be reflected in the description of an algorithm properties and structure.

AlgoWiki provides an exhaustive description of an algorithm. In addition to classical algorithm properties such as serial complexity, AlgoWiki also presents complementary important information, which together provides a complete description of the algorithm: its parallel complexity, parallel structure, information graph (data dependency graph), determinacy, data locality, performance and scalability estimates, communication profiles for specific implementations (for distributed memory machines), and many others. As an example, Fig. 4 presents the performance depending on the number of processors and the problem size and Fig. 5 shows a memory access profile for a specific sparse matrix algorithm.

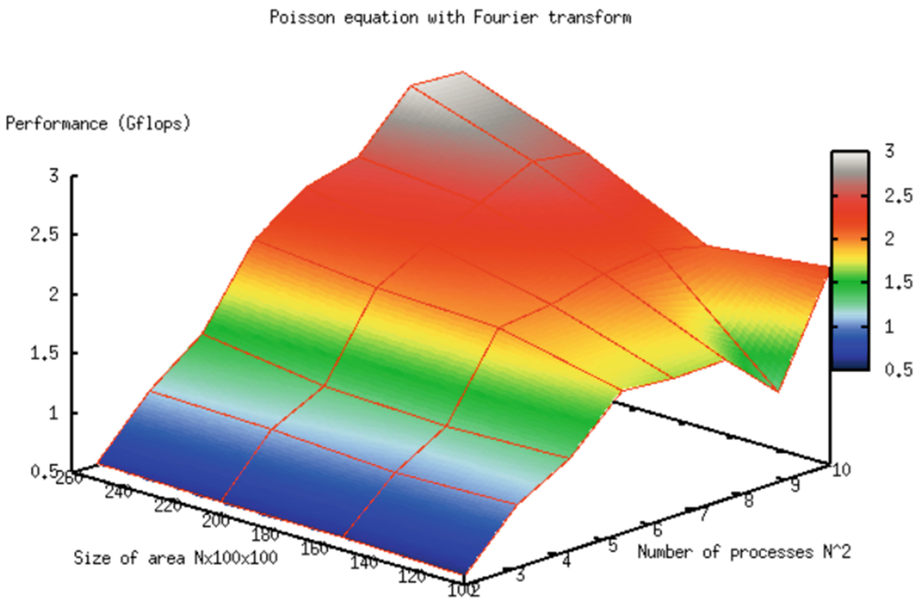

Fig. 4. Performance distribution over number of processors and problem size.

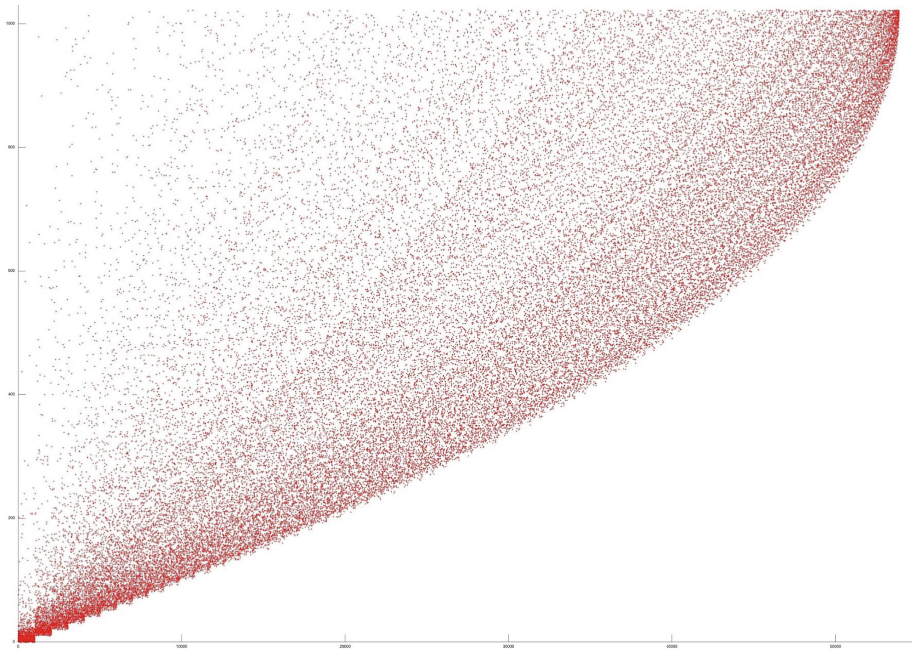

Fig. 5. Example of memory access profile.

The encyclopedia is built using Wiki technologies and is open to everyone, which enables the entire computing community to collaborate on algorithm descriptions. It is freely available at: AlgoWiki-Project.org/en.

3 Conclusions

All the systems described above are closely linked to one another, ensuring high efficiency for large supercomputing centers. Of course, they were designed while taking all key aspects of maintenance and usage into account for existing supercomputers. At the same time, the architecture of these systems was designed to be able to adopt large scale, complexity, high degree of parallelism and performance of the forthcoming exascale supercomputers.

Acknowledgments. The results were obtained with the financial support of the Ministry of Education and Science of the Russian Federation, Agreement N 14.607.21.0006 (unique identifier RFMEFI60714X0006).

References

1. Supercomputing education in Russia. Final report on the national project “Supercomputing Education”, supercomputing consortium of the Russian Universities (2012). http://hpc.msu.ru/files/HPC-Education-in-Russia.pdf
2. Voevodin, Vl.V., Gergel, V.P.: Supercomputing education: the third pillar of HPC. In: Computational Methods and Software Development: New Computational Technologies. vol. 11, no. 2, pp. 117–122. MSU Press, Moscow (2010)
3. Computing curricula computer science (2013). http://ai.stanford.edu/users/sahami/CS2013
4. NSF/IEEE-TCPP curriculum initiative on parallel and distributed computing. http://www.cs.gsu.edu/~tcpp/curriculum/
5. Future directions in CSE education and research. Report from a Workshop Sponsored by the Society for Industrial and Applied Mathematics (SIAM) and the European Exascale Software Initiative (EESI-2). http://wiki.siam.org/siag-cse/images/siag-cse/f/ff/CSE-report-draft-Mar2015.pdf
6. Sadovnichy, V., Tikhonravov, A., Voevodin, Vl., Opanasenko V.: “Lomonosov”: supercomputing at Moscow State University. In: Contemporary High Performance Computing: From Petascale toward Exascale (Chapman and Hall/CRC Computational Science), pp. 283–307. CRC Press, Boca Raton (2013)
7. Dongarra, J., et al.: The international exascale software roadmap. Int. J. High Perform. Comput. **25**(1), 3–60 (2011). ISSN 1094-3420
8. Antonov, A., Voevodin, V., Dongarra, J.: AlgoWiki: an open encyclopedia of parallel algorithmic features. J. Supercomput. Front. Innovations **2**(1), 4–18 (2015)