Parallel structure of algorithms and training computational technology specialists

A S Antonov\textsuperscript{1}, Vl V Voevodin\textsuperscript{1,2} and N N Popova\textsuperscript{2}

\textsuperscript{1}Research Computing Center, Lomonosov Moscow State University, 119234 Russia, Moscow, Leninskie gory, 1, bld. 4
\textsuperscript{2}Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University, 119991 Russia, Moscow, Leninskie gory, 1, bld. 52
E-mail: asa@parallel.ru, voevodin@parallel.ru, popova@cs.msu.ru

Abstract. A universal method for describing an algorithm’s parallel structure was developed as a part of AlgoWiki Open Encyclopedia of Parallel Algorithmic Features. AlgoWiki is not just an important source of information on specific algorithms, but also a good training ground for students. To develop a high-quality algorithm description, one needs to study its structure, identify its key properties, obtain the data by practical experiments on a supercomputer and learn to visualize the results. This article describes the experience of conducting a computer lab course on parallel algorithm structure at the Faculty of Computational Mathematics and Cybernetics at Lomonosov Moscow State University.

1. Introduction
Parallel computations are becoming a part of our lives increasingly. Once considered an attribute of the most powerful and exotic computers, parallelism is now present in the mobile devices we use every day. Moreover, we can be confident that this trend will continue to develop, with the number of processing cores rapidly increasing in every computing device around us. That means an ever-growing need for specialists who have the knowledge and skills to use parallel devices efficiently. Recognition of this fact is a key to new specialist training programs being developed all over the world [1, 2, 3, 4, 5, 6].

The specialists of this profile have been trained at the Faculty of Computational Mathematics and Cybernetics at Lomonosov Moscow State University for some time now. However, the need to offer large-scale training for parallel computation specialists requires new forms of organization. A few years ago, only some groups of students were allowed to perform practical work on real supercomputers; now, absolutely every student at the Faculty attends supercomputer lab courses. This applies to both Bachelor’s and Master’s Degree students.

The present paper describes the experiences from the Supercomputer Simulation and Technologies computer lab course (2016) which was completed by all second-year Master’s degree students at the Faculty of Computational Mathematics and Cybernetics at Lomonosov Moscow State University.

2. Algorithms and parallelism
The history of high-performance computing can be presented roughly as a change in computational paradigms: from vector computers, computers with shared and distributed
memory, multi-core CPU-based computers, accelerator-based computers, etc. New parallel computing architectures undoubtedly will continue to appear in the future. Each paradigm shift results in the need to rewrite the existing parallel code — efficient usage of a new platform requires considering different properties of the same algorithm, using other parallel programming technologies, etc.

One may ask, whether the users are doomed to rewrite endlessly the entire store of parallel programs, or can something from the past experience be used for new generations of parallel computers? Fortunately, not all prior software must be abandoned.

Generally we assume that *changes in computer architecture do not change algorithms* [7]. If this assumption is true, a parallel algorithm structure only needs to be described only once, and this knowledge can be used for new computational paradigms.

In 2014, this idea resulted in the launch of the AlgoWiki, the Open Encyclopedia of Parallel Algorithmic Features, at Lomonosov Moscow State University.

3. AlgoWiki, the Open Encyclopedia of Parallel Algorithmic Features

| (i) Properties and structure of the algorithm |
|---------------------------------------------|
| 1.1 General description of the algorithm   |
| 1.2 Mathematical description of the algorithm |
| 1.3 Computational kernel of the algorithm |
| 1.4 Macro structure of the algorithm       |
| 1.5 Implementation scheme of the serial algorithm |
| 1.6 Serial complexity of the algorithm     |
| 1.7 Information graph                      |
| 1.8 Parallelization resource of the algorithm |
| 1.9 Input and output data of the algorithm |
| 1.10 Properties of the algorithm           |

| (ii) Software implementation of the algorithm |
|-----------------------------------------------|
| 2.1 Implementation peculiarities of the serial algorithm |
| 2.2 Locality of data and computations         |
| 2.3 Possible methods and considerations for parallel implementation of the algorithm |
| 2.4 Scalability of the algorithm and its implementations |
| 2.5 Dynamic characteristics and efficiency of the algorithm implementation |
| 2.6 Conclusions for different classes of computer architecture |
| 2.7 Existing implementations of the algorithm |

| (iii) References |

**Figure 1.** Description of algorithm properties and structure in the AlgoWiki project

The AlgoWiki Open Encyclopedia of Parallel Algorithmic Features (https://algowiki-project.org) [8] is a project based on wiki-technologies which allows joint work of the computing community to describe the algorithm properties and their implementations. The main goal of this project is to formalize the mapping of algorithms onto the architecture of parallel computing systems.

AlgoWiki provides a comprehensive description of algorithms. In addition to classical algorithm properties such as serial complexity, AlgoWiki presents new features, which together provide a complete description of the algorithm: its parallel complexity, parallel structure, determinacy, data locality, performance, efficiency and scalability estimates, communication profiles for specific implementations, dynamic characteristics and performance efficiency timelines, and many others. All of these concepts, and many others, are used to describe
an algorithm’s features from different perspectives, and they all are quite necessary in practice under various situations.

An algorithm description in AlgoWiki consists of two key parts. The first part describes the actual algorithms and their properties. These descriptions are machine-independent and do not change in transitioning to a new computing platform. The second part describes features of the algorithms’ software implementation on the specific software and hardware platforms [9]. By studying fundamental implementation properties such as execution time, performance, data locality, efficiency and scalability, one is able to give some estimate of the potential implementation quality for a given algorithm on a specific computation platform and lay the foundation for comparative analysis of various computers with regards to the algorithms presented in AlgoWiki Encyclopedia.

Both parts are equally important since together they provide an exhaustive description of an algorithm, which helps to evaluate its implementation potential on a given parallel computing platform.

The description of the algorithms in AlgoWiki has been conceived as universal, independent of the specific computational algorithm being described. The description structure is the same for all algorithms and includes the following sections (see figure 1).

The central part of the description of an algorithm’s machine-independent properties is dedicated to building and analyzing the information graph [10, 11, 12]. This is a directed acyclic graph, where the vertices correspond to operations of the algorithm being analyzed, and the edges correspond to the transmission of information between the vertices (vertices A and B are connected with an edge if the data computed in vertex A are used in vertex B).

An information graph is vital for studying algorithm properties, as it contains all of the necessary information about its parallel structure. The skills for working with an information graph are very important in practice, as they help to evaluate an algorithm’s parallel complexity and the application’s parallelism resource, understand the algorithm’s bottlenecks and find different options for a parallel implementation.

The information graphs can be built with varying levels of detail, from elementary program operations to large macro-operations. This results in either an extremely detailed graph that is hard to visualize and analyze, or in an extremely trivial graph that does not provide any useful information on the algorithms properties. A more detailed representation requires either gradually increasing the macro operations, trying not to miss the relevant properties, or complementing the macro graph with detail on the information structure of the macro vertices. Figure 2 shows a detailed information graph for the Lanczos Method for finding eigenvalues in a symmetric matrix [13], as presented in a student’s work.

Figure 2 a) shows an information graph at the macro operation level which corresponds to a sequence of iterations. Figure 2 b) provides detail on the vertex corresponding to the algorithm’s iterations.

Many useful properties of an information graph can be obscured by visualizing a multidimensional object on a plane. In many cases it is easier to understand an algorithm’s structure and define its specific properties by looking at an interactive 3D visualization of the information graph, also developed as part of the AlgoWiki project. The interactive representations obtained using an information graph 3D visualization tool called AlgoView [14], are available on several pages of the AlgoWiki Open Encyclopedia of Parallel Algorithmic Features. They allow the user to look at the information graph from various perspectives, get projections to coordinate planes, build multilevel structures, etc.

Figure 3 provides an overview of certain visual capabilities of the AlgoView system, using an information graph for the Givens rotation method for the QR decomposition of a square matrix.
Figure 2. The Lanczos Method for finding eigenvalues in a symmetric matrix: a) macro graph (Op represents an iteration of the algorithm, Eigen is the procedure for computing eigenvalues); b) algorithm iteration detail.

4. Computer lab assignment for the Supercomputer Simulation and Technologies course
The Supercomputer Simulation and Technologies course is delivered at the Faculty of Computational Mathematics and Cybernetics at Lomonosov Moscow State University for the 2nd-year Master’s degree students. About 250 students study this discipline every year. The teachers from various chairs at the CMC faculty are invited to take part in the development and delivery of lectures, along with employees from certain IT companies involved in the design and use of supercomputer systems.
Figure 3. 3D visualization of an information graph for the Givens rotation method of QR decomposition of a square matrix.

The course studies must be backed by a supercomputer lab course. Another challenge of this method is that the lectures for the course are presented by several specialists, each covering his/her own expertise within the subject. Therefore, the practical assignment must be comprehensive enough to form practical skills in several areas at once. Ideally, the assignment should also be useful in practice, rather than something disconnected from reality.

One of the tasks during the lab course required describing the structure and properties of a selected algorithm from the AlgoWiki Open Encyclopedia of Parallel Algorithmic Features.

The students could select and describe any one of 30 algorithms proposed, particularly:
- The “Divide and conquer” method for calculating eigenvalues and vectors in a symmetric three-dimensional matrix.
- The Jacobi method for finding eigenvalues in a symmetric matrix.
- The Lanczos algorithm (iterative method for computing eigenvalues in a symmetric matrix) for precise arithmetic (without orthogonalization).
- Gram-Schmidt orthogonalization.
- Computing definite integrals using an adaptive condensing grid.
- Addressing the initial Cauchy Problem value for a system of ordinary differential equations using the Runge-Kutta fourth-order method.
- The CURE (Clustering Using REpresentatives) algorithm.
- and others.

The algorithms were distributed among the students at their discretion; any algorithms that were too popular were excluded from the sampling. This ensured a more or less even distribution of students between the different assignment options. In addition to the algorithms proposed, the students could try to describe any algorithm at their choice, subject to the tutor’s approval. Several students took this opportunity to describe, for example, a face recognition algorithm and several cryptoanalysis algorithms.

The students were encouraged to produce algorithm descriptions following the structure presented in figure 1, excluding sections 2.1–2.3, 2.5 and 2.6. The main focus of the task was not on describing the algorithm (this part could simply be copied from the textbook or an Internet source, including the respective links), but on studying its properties — primarily the algorithm’s information structure and parallelism resource. A description of these properties is rarely found in the literature, so this part of the assignment required conducting independent research. The selected algorithm was provided together with a link to one of the known sources of information for a given algorithm. This guaranteed unequivocal agreement between the tutor and the student on the specific algorithm to be described.

The requirement to present an information graph of the algorithm was the biggest challenge, as expected. The students were not accustomed to this notion, so the most common error was presenting a variety of control flow graphs instead of an information graph. Many students also presented information graphs at the macro operational level, which clearly wasn’t enough to evaluate the parallel structure of the algorithm. For those students who tried to produce a full information graph, the greatest challenge was to place the vertices properly and draw the edges so as to visualize the information structure with the highest possible detail. As a result, high-quality information graph images similar to the one presented in figure 2, were fortunately a rare exception. Figure 4 shows three variants of the information graph representation for one algorithm submitted by three different groups of students. However, the task itself was quite complex, so the tutors weren’t very critical of the information graphs that were less than perfect.

To describe parallelization resource of the algorithm students should estimate the algorithm’s parallel complexity: the number of steps it takes to execute the algorithm assuming an infinite number of processors (functional units, computing nodes, cores, etc.). The parallel complexity of an algorithm is understood as the height of its canonical parallel form. It is necessary to indicate in terms of which operations the estimate is provided. It is also necessary to describe the balance of parallel steps by the number and type of operations, which determines the layer width in the canonical parallel form and the composition of operations within the layers. For example, the Lanczos Method for finding eigenvalues in a symmetric matrix which information graph is shown in figure 2 is qualified as a linear complexity algorithm in terms of the parallel form height. In terms of the parallel form width, its complexity is also linear.

To describe the scalability [15] of algorithm implementations in Section 2.4, students were allowed to select any implementation of the algorithm or write their own, using any available parallel programming technologies (MPI, OpenMP, Cuda, etc.), parallel libraries (Intel MKL, PETSc, FFTW, ScaLAPACK, etc.). The goal of this part of the assignment was not to learn the respective parallel programming technologies or software development environments, but to develop the skills needed to perform computational experiments on supercomputers and properly interpret the results for describing algorithm properties. It was important to choose a proper correlation between the number of processors and the scope of the problem, so as to reflect
all features in behavior of the parallel program, particularly achieving maximum performance, and more subtle effects arising, for example, from the algorithm’s block structure or memory hierarchy.

Any computer could be used as the target platform for obtaining the properties of specific algorithm implementations, including MSU’s Lomonosov [16, 17] and IBM Blue Gene/P supercomputers which are available to all students.

An example of a scalability of the “Divide and conquer” method implementation for calculating eigenvalues and vectors in a symmetric three-dimensional matrix diagram [13] from one student’s work is shown in figure 5.

After the scalability diagrams were drawn, the students were asked to explain the data obtained. This is often hard to do, as it requires matching the various algorithm properties studied with features of the target software and hardware platform. Not every student completed this part of the assignment; some only made superficial conclusions and explanations.

For example, when describing the scalability diagram presented in figure 4, one student limited himself to the following simple conclusions:

- “As the number of processes increases, the overall program efficiency increases, as the time required to read and prepare the input matrix remains the same and the algorithm execution time is decreased; however, this growth is insignificant.
- As the matrix dimension increases, overall program efficiency decreases gradually, as the time required to read and prepare the input matrix grows faster than the algorithm execution time.
- If the number of processes increases simultaneously with the matrix dimensions, the overall program efficiency is decreased.”
Figure 5. The scalability of the “Divide and conquer” method implementation for calculating eigenvalues and vectors in a symmetric three-dimensional matrix.

Besides these conclusions being highly superfluous, the usage of the term “efficiency” is not quite justified; in addition, the term “matrix dimensions” is used incorrectly instead of “matrix size.”

For many algorithm-computer pairs, good implementations have already been developed which can and should be used in practice. Students provided references to existing serial and parallel implementations of an algorithm that were available for use. For any implementation should be specified: is an implementation open-source or proprietary, what type of license it is distributed under, the distributive location and any other available descriptions. If there is any information on the particular features, strengths and/or weaknesses of various implementations, these could be pointed out.

Together with the results of the scalability study, the students should provide the testing technology for the program implementing the algorithm in question. This is particularly important for the programs written by the students, otherwise there will be no confidence in the results presented.

The tutors checking the assignment were asked to achieve adequate quality for the algorithm properties description and to teach the students to make these descriptions rather than just to grade the assignment. This invariably meant that students had to interact with the tutors. Instead of immediately grading the work, the tutor would formulate his/her comments; the students would react to the feedback, then send the updated algorithm descriptions for further checking. This interaction was repeated as necessary, effectively only limited by the deadline for grading course results at the end of the semester.

This is very different from the traditional assignment grading process. Its goal is not only to make sure the course materials are perceived correctly, but also to teach the students to make
proper and high-quality descriptions of an algorithm’s properties. This requires much higher qualifications for the tutors and more time spent on the assessment, but the end results are of a much better quality.

Student descriptions of the algorithms were prepared within their personal spaces within the AlgoWiki Open Encyclopedia of Parallel Algorithmic Features, and interaction with the tutors was conducted by the discussions mechanism provided by the MediaWiki [18] engine used in the AlgoWiki project. This allowed every stage of the assignment to be tracked, including all changes made in the articles, tutor comments, response to tutor feedback, etc.

A total of 246 second-year Master’s degree students completed the assignment during the fall semester of 2016. Due to the high labor intensity, students were allowed to work in pairs while describing a single algorithm. As a result, each of the proposed algorithms was described by 4–5 groups.

The grades for the 146 student groups were as follows:

- 5 (excellent) — 59 works;
- 4 (good) — 36 works;
- 3 (satisfactory) — 48 works;
- 2 (unsatisfactory) — 3 works.

Thus, the average grade for the assignment was 4.03, which indicates the generally high level of descriptions produced. The quality of individual work produced from the assignment was in line with the high standard of works accepted at the AlgoWiki Open Encyclopedia of Parallel Algorithmic Features.

As a result, 1 to 6 description versions were written for each of the 30 proposed algorithms. All of the descriptions varied slightly from each other, as they described the same algorithm properties in somewhat different ways. So substantial skilled work is required before these descriptions can be added to the main branch of the AlgoWiki encyclopedia; the AlgoWiki project team is currently working on this. If necessary, the project participants will solicit help from volunteers among the students who performed the assignment. Everyone taking part in the resulting AlgoWiki article, including the students and tutors are named on the list of authors.

5. Conclusion

This article demonstrates the use of technology to describe algorithm structures for purposes of the AlgoWiki Open Encyclopedia of Parallel Algorithmic Features project to complete a computer lab course as part of the Supercomputer Simulation and Technologies course studies at the Faculty of Computational Mathematics and Cybernetics at Lomonosov Moscow State University. In this form of lab course, the students get an idea of how parallel applications behave, which is a key to quick learning of parallel computing platforms in real practical work in the future. The experience can easily be adapted to other training courses, which enables large-scale education on various aspects of parallel computation usage.

6. Acknowledgements

The results displayed in Sections 2 and 3 were obtained at Lomonosov Moscow State University with the financial support of the Russian Science Foundation (agreement N 14–11–00190). The research is carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University supported by the project RFMEFI62117X0011.
References

[1] Computer Science Curricula 2013 URL: http://ai.stanford.edu/users/sahami/CS2013 (accessed on 22.11.2017)

[2] NSF/IEEE-TCPP Curriculum Initiative on Parallel and Distributed Computing URL: http://www.cs.gsu.edu/tdcpp/curriculum (accessed on 22.11.2017)

[3] SIAM: Graduate Education for Computational Science and Engineering / SIAM Working Group on CSE Education (2014) URL: http://www.siam.org/students/resources/report.php (accessed on 22.11.2017)

[4] 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) URL: http://wiki.siam.org/siag-cse/images/siag-cse/f/ff/CSE-report-draft-Mar2015.pdf (accessed on 22.11.2017)

[5] Voevodin VI V, Gergel V P and Popova N N 2015 Challenges of a Systematic Approach to Parallel Computing and Supercomputing Education Euro-Par 2015: Parallel Processing Workshops Springer Vienna pp 90–101

[6] Voevodin VI V and Gergel V P 2010 Supercomputing Education: the Third Pillar of HPC Computational Methods and Software Development: New Computational Technologies 11 2 Moscow State University Press pp 117–22

[7] Voevodin VI, Antonov A and Dongarra J 2016 Why is it hard to describe properties of algorithms? Procedia Comput. Sci. 101 pp 4–7

[8] Voevodin VI, Antonov A and Dongarra J 2015 AlgoWiki: an Open Encyclopedia of Parallel Algorithmic Features Supercomputing Frontiers and Innovations 2 1 pp 4–18

[9] Antonov A, Voevodin Vad, Voevodin VI and Teplov A 2016 A Study of the Dynamic Characteristics of Software Implementation as an Essential Part for a Universal Description of Algorithm Properties it 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing Proceedings, 17th-19th February 2016 pp 359–63

[10] Voevodin V 1992 Mathematical Foundations of Parallel Computing (World Scientific Publishing Co., Series in Computer Science) 33

[11] (in Russian) Voevodin V and Voevodin VI 2002 Parallel Computing (BHV-Petersburg, St. Petersburg)

[12] Sharp J A, Ed. 1992 Data Flow Computing: Theory and Practice (Ablex Publishing Corp.)

[13] Demmel J W 1997 Applied Numerical Linear Algebra (SIAM, Philadelphia, PA.)

[14] Antonov A S and Volkov N I 2017 An AlgoView Web-visualization System for the AlgoWiki Project Communications in Computer and Information Sci. 753 Springer International Publishing AG pp 3–13

[15] Antonov A and Teplov A 2016 Generalized Approach to Scalability Analysis of Parallel Applications Lecture Notes in Computer Science 10049 pp 291–304

[16] (in Russian) Voevodin VI V, Zhumaitii S A, Sobolev S I, Antonov A S, Bryzgalov P A, Nikitenko D A, Stefanov K S, Voevodin Vad V 2012 The Lomonosov supercomputer in practice Otkrytye Sistemy 7 pp 36–9

[17] Sadovnichy V, Tikhonravov A, Voevodin VI and Opanasenko V 2013 “Lomonosov”: Supercomputing at Moscow State University Contemporary High Performance Computing: From Petascale toward Exascale (Chapman & Hall/CRC Computational Science Boca Raton, USA, CRC Press) pp 283–307

[18] MediaWiki URL: http://www.mediawiki.org (accessed on 22.11.2017)