Comprehensive Collection of Time-consuming Problems for Intensive Training on High Performance Computing

Iosif Meyerov[1], Sergei Bastrakov[2,1], Alexander Sysoyev[1], Victor Gergel[1]✉

[1] Lobachevsky State University of Nizhni Novgorod, Nizhni Novgorod, Russia
[2] Helmholtz-Zentrum Dresden-Rossendorf, Dresden, Germany
iosif.meerov@itmm.unn.ru
sergey.bastrakov@itmm.unn.ru
alexander.sysoyev@itmm.unn.ru
gergel@unn.ru

Abstract. Training specialists capable of applying models, methods, technologies and tools of parallel computing to solve problems is of great importance for further progress in many areas of modern science and technology. Qualitative training of such engineers requires the development of appropriate curriculum, largely focused on practice. In this paper, we present a new handbook of problems on parallel computing. The book contains methodological materials, problems and examples of their solution. The final section describes the automatic solution verification software. The handbook of problems will be employed to train students of the Lobachevsky University of Nizhni Novgorod.

Keywords: Parallel Computing · High Performance Computing · Education.

1 Introduction

Modern computational science is becoming increasingly interdisciplinary and concerns all spheres of human life. The development of computer architectures and related technologies has led to the emergence of new applied fields of science (bioinformatics, computational biomedicine, computer vision, machine learning, big data and so forth). Many existing areas of the natural sciences received further development under the influence of supercomputers. It is widely recognized that supercomputers are driving technological progress in the world.

Further development of the industry requires significant efforts from the community. Among the important areas is the problem of training highly qualified specialists in application of models, methods and technologies of parallel computing to solve scientific and engineering problems using supercomputers. This area is rapidly expanding in many directions, which, on the one hand, enhances its expressive power, but, on the other hand, it places ever higher demands on engineers, their knowledge and skills. Our community needs special training courses on parallel programming, mainly oriented to practice.

In this paper we briefly review the new handbook of problems on parallel computations developed at the Institute of Information Technology, Mathematics and Mechanics (ITMM) of the Lobachevsky University on the basis of many years of experience in training engineers in supercomputing technologies. The book aims to guide students through a wide range of problem domains, including low level parallel programming, basic parallel algorithms, and parallel numerical methods. We hope that the book can complement the modern lecture courses on parallel programming.

2 Related work

A significant amount of methodical literature on parallel computations has been prepared during the last decades. Theoretical information on parallel computations and parallel algorithms is presented in the books [1][2][3][4][5]. The development of parallel programs for distributed memory systems using the MPI (including not only a description of the contents and capabilities of the MPI, but also examples of programs) is discussed in detail in the books [6][7][8][9] and the in first volume of [10]. Parallel programming for shared memory system based on the OpenMP technology can be studied from the book [11] and the second volume of the book [10]. Development and optimization of parallel programs for Intel Xeon Phi processors are presented in detail in [12][13]. The books [14][15] contain various examples of optimizing programs for Xeon Phi from different problem domains. Programming for GPU accelerators with a large number of examples is discussed in detail in [16][17][18]. Paper [19] should also be noted, where the study of the MPI technology is conducted on the example of solving a number of typical problems of parallel programming – matrix calculations, sorting, processing graphs, etc. The questions of teaching parallel algorithms and technologies are also of considerable interest. In this regard, the activity of the NSF / IEEE-TCPP Curriculum Initiative on Parallel and Distributed Computing [20][21] should be mentioned. In Russia, significant progress has been made in this direction within the framework of the government program Supercomputing Education [22].

The presented handbook continues a series of publications on parallel programming education. Among these publications are textbooks [5][10], tutorials and practical classes on various technologies of parallel programming and their application to scientific research in different problem domains [23][24]. A considerable part of the prepared publications is presented on the web page of the Center for Supercomputer Technologies of Lobachevsky University in Russian (http://hpc-education.unn.ru/ru) and in English (http://hpc-education.unn.ru/en/trainings/collection-of-courses). The training course "Introduction to parallel computing" is available on the website of the e-learning system (http://mc.e-learning.unn.ru/; in Russian).

3 Handbook overview

Preparing a specialist capable of effectively using modern parallel computing systems is a challenging problem. It may seem that any person who has good theoretical back-

ground in mathematics and, for example, physics (chemistry, biology, medicine – depending on a specific problem domain), and also able to implement sequential programs, can easily study parallel programming. However, it is often not enough to read descriptions of OpenMP and MPI and learn how to develop and run software on a supercomputer. The efficiency of parallel software depends highly on specific details, and it turns out that there are quite a lot of these details. For example, it is necessary to have skills of system programming and debugging of parallel code, which is significantly more complicated than the sequential one. In addition, the architecture of modern computing systems has become so complicated that very little manipulation of code and run modes can lead to an order of magnitude difference in computational time, and sometimes even more. Therefore, training of specialists with skills to develop and run parallel programs is still of great importance.

Education in a rapidly changing field cannot be reduced to studying theoretical material and a number of standard use cases. In our universities we need training courses with many practical problems exemplifying typical challenges from different areas of science and engineering. Thus, the development of parallel programs for solving sparse algebra problems requires the study of special data structures and methods for their effective use in parallel computations. Parallelization of Monte Carlo methods requires the understanding of how to use pseudo-random number generators in parallel programs. The experience of running standard performance tests forms extremely useful skills of customization the computational environment and analysis of the experimental data. Each section of the presented handbook illustrates typical data structures, algorithms and approaches in some specific problem domain. We stress that we did not try to write a book on system programming, parallel algorithms or numerical methods. On the contrary, we recommend readers to study the literature that has become classical in this field, giving corresponding links in each section. Theoretical material, which is given in the handbook, is the minimum necessary to understand the problems, and where it can be read about it. Further, we consider data structures and the approaches to parallelization, give problem statements, discuss possible solutions, and formulate the set of problems which should be solved by students. These set of problems can be easily extended to reach the goals of education. Thus, algorithms can be implemented for different types of computing systems with shared and distributed memory using appropriate technologies. By adjusting the input data and, correspondingly, computational load, we can assess the quality of implementation checking correctness, performance, and scaling efficiency. The book also describes a system for automated testing of parallel programs, developed by students. The use of such a system makes it possible to significantly reduce the efforts of teachers in verifying the homework.

4 Handbook Structure and Contents

The book outline includes the following sections:

1. Parallel programming for beginners.
2. Parallel execution of threads and processes.

3. Communication methods.
4. Performance tests.
5. Calculations with matrices.
6. Interpolation and approximation of functions.
7. Numerical integration.
8. Sorting algorithms.
9. Graph algorithms.
10. Numerical solution of systems of ordinary differential equations.
11. Numerical solution of systems of partial differential equations.
12. Methods of global optimization.
13. Monte Carlo methods.
14. Computational geometry algorithms.
15. Computer graphics and image processing algorithms.
16. Automatic solution verification system SoftGrader.

Most of the sections are organized as follows. A section begins with a brief introduction to the problem area and provides a rationale for the importance of discussed tasks and methods in HPC applications. The following material describes ways of representing and storing data, standard for typical problems in the considered area. Then we give a brief overview of parallelization schemes. The section continues with detailed statements of one or two typical problems. Parallel methods for their solution are outlined and discussed. Then, we formulate tasks for students. An example of a problem statement accompanied by a demonstration of program code finalizes the section.

Let us overview the book contents. In the *Preface* we briefly outline our motivation to write the book. The first section presents the computational problems for beginners and helps to form an introductory laboratory practical work on various academic disciplines in the field of parallel computing (for example, to study the OpenMP and MPI technologies).

The next three sections relate to the area of system programming. In *Section 2* we consider the principles of threads execution planning, synchronization mechanisms and standard problems, offer tasks for the use of typical mechanisms of threads interaction. *Section 3* proposes tasks for the development of data transfer programs between processors on distributed memory. When carrying out homework, it is necessary to evaluate the expected time for executing communication operations, implement sample data transfer programs, and compare the results obtained with theoretical estimates. To evaluate the effectiveness of the implemented software, it is necessary to run computational experiments comparing the developed programs with standard tools (for example, with functions of the similar purpose in an MPI library). *Section 4* deals with standard tests that identify the performance parameters of computational clusters. Relevant work with such tests allows an engineer to make decisions about the configuration of the future computing system, calculate the standard performance parameters of ready to use systems, and identify their bottlenecks.

The following three sections concern numerical methods. In *section 5* we consider the following basic linear algebra algorithms: matrix-vector (dense and sparse) and

matrix-matrix multiplication. In this regard, we discuss data structures, especially for sparse matrices, and different schemes of parallelization. In *section 6* we discuss interpolation and approximation methods, give examples and propose several tasks for students. In *section 7* we consider one of the simplest formulas of numerical integration (the rectangle rule) and its application to the problem of calculating an integral of a function of two variables. When performing each of the tasks, it is necessary to check correctness, construct theoretical estimates of the run time of an algorithm, perform computational experiments and compare the results with theoretical estimates.

Sections 8 and 9 consider two classical areas of Algorithms and data structures: sorting algorithms and graph algorithms. In *section 8* we discuss different parallel merging schemes which can be used as a part of many sorting algorithms. In *section 9* we consider methods for storing graphs, the properties of graphs algorithms, general approaches to their parallelization, and consider some classical graph processing methods: breadth-first search, minimal spanning tree search, and single-source shortest path paths search.

In *sections 10 and 11* we consider parallel algorithms of solving systems of ordinary differential equations and partial differential equations, correspondingly. In *section 10* we discuss the Euler and Runge-Cutta methods for an ODE system. In *section 11* we consider the numerical solution of the Dirichlet problem for the Poisson equation.

In *section 12* we briefly overview global optimization methods, discuss the applicability of general parallelization approaches to global optimization and present a parallel version of the global search algorithm.

Section 13 concerns Monte Carlo methods. We outline methods of pseudorandom and quasirandom number generation, consider approaches for simulation of distributions with given properties, and discuss how to use random number generators in parallel computations correctly. Next, we introduce the problem of developing a generator of uniformly distributed numbers and demonstrate one of the methods for solving it. Then, we consider an example from financial mathematics: finding the fair price of an option of the European type using the parallel Monte Carlo method.

Section 14 is devoted to approaches to the parallel solution of computational geometry problems. The ideas are demonstrated by the example of constructing a convex hull of points on a plane. In *section 15*, computer graphics and image processing problems are presented. The methods of parallelization of image processing operations are considered.

In *section 16*, the SoftGrader automated testing system is described. We demonstrate main features of the software and emphasize the scheme for preparing the training course and its practical problems for their automated testing in SoftGrader.

5 Section Example

This section of the paper presents a more detailed description of the one of the book sections – Monte Carlo method. We start the section with a brief overview of the

problem area and give references to classical books and tutorials concerning the Monte Carlo method and its applications to different problems of modern computational science.

Then, we discuss the general scheme of the Monte Carlo method. First, we outline a problem of generating random numbers in a computer. Then, we give a description of the linear congruential generator and depict its advantages and disadvantages. After that, we briefly describe the Mersenne Twister algorithm for generating random numbers and give relevant references. The description of the Sobol sequence continues the section. As before, we present the advantages and disadvantages of the approach, and compare it with the algorithms mentioned above. At the end of the subsection we put references to the description of the pseudorandom number sampling methods mapping uniformly distributed numbers to other distributions.

In the next subsection we discuss parallelization of the Monte Carlo method. Although this method is known to be embarrassingly parallel, there are important details concerning usage of random number generators in parallel computations. In this regard, we consider the master-worker, leapfrog, and skipahead methods. We also describe some typical errors which are often made during the implementation and their consequences. The discussion regarding the performance of the Monte Carlo method implementation finalizes the subsection. Thus, we pay attention to performance of the random number generation algorithm, including cache efficiency and utilization of the SIMD instructions. We highly recommend using carefully tested high performance implementations given by mathematical software libraries in real-world applications (i.e. the Intel Math Kernel library).

To better understand the nature of random number generation algorithms, we recommend students to implement and test correctness and performance of the MCG59 algorithm. The relevant problem statement and discussion are given in the following subsection of the book. We give the algorithm and implementation overview, formulate the main requirements and recommend using the Pearson's chi-squared test and the Kolmogorov–Smirnov test to check correctness. Then, we ask students implementing a parallel algorithm to generate a chunk of uniformly distributed numbers and analyze its performance and scalability depending of the chunk size. Different parallel programming technologies, computer architectures and random number generation algorithms can be involved.

In the next subsection, we show an example of a financial mathematics problem solved by the Monte Carlo method. In this regard, we introduce the Black-Scholes financial market model, give the main definitions including the European option and fair price concepts. We consider the case where the option price can be found analytically using the Black-Scholes formula to simplify testing correctness of the Monte Carlo method implementation. Then, the general scheme of the Monte-Carlo method for this problem including the parallel algorithm is also discussed. As in the previous subsection, we formulate general requirements and tasks for students.

According to our approach, every problem statement for students mentioned in the previous two subsections of the book, could be widen significantly to make it possible to check correctness and performance automatically. Thus, we need giving a mathematical problem statement and formulate strict requirements for the further imple-

mentation including parameters, data structures, algorithm, programming language, technologies, input/output formats, correctness and performance testing criterions. The appropriate example of one task has been done in the end of the Monte Carlo method section.

6 Conclusion

In this paper we presented a new handbook on parallel computing. The main idea of the book is to form the general principles of organizing a practical work on parallel programming with automated testing of developed implementations for correctness and performance. To this end, we have formed a pool of tasks in various areas of computer science and computational sciences, and also presented a way of formalizing the requirements for solving these problems, illustrating it with examples. The authors hope that the book could be useful in teaching students to solve problems using parallel algorithms.

References

1. Andrews, G. R. (1999). Foundations of parallel and distributed programming. Addison-Wesley Longman Publishing Co., Inc..
2. Kumar V., Grama A., Gupta A., Karypis G. Introduction to Parallel Computing. – Pearson Education, 2003. – 612 p.
3. Wilkinson B., Allen M. Parallel programming: techniques and applications using networked workstations and parallel computers. – Prentice Hall, 1999.
4. Voevodin V.V., Voevodin Vl.V. Parallel Computations. Saint-Petersburg, BHV-Petersburg, 2002 (In Russian).
5. Gergel V.P. Theory and Practice of Parallel Computations. Moscow, Binom, 2007 (In Russian).
6. Pacheco, P. (1996). Parallel Programming with MPI. – Morgan Kaufmann.
7. Gropp, W., Lusk, E., Skjellum, A. (1999a). Using MPI – 2nd Edition: Portable Parallel Programming with the Message Passing Interface (Scientific and Engineering Computation). – MIT Press.
8. Gropp, W., Lusk, E., Thakur, R. (1999b). Using MPI-3: Advanced Features of the Message Passing Interface (Scientific and Engineering Computation). – MIT Press.
9. Nemnyugin S., Stecik O. Parallel programming for multiprocessor computing systems. Saint-Petersburg, BHV-Petersburg, 2002.
10. Gergel V., Barkalov K., Meyerov I., Sysoyev A. et el. Parallel Numerical Methods and Technologies. UNN Press, 2013 (In Russian).
11. Chandra R., Dagum L., Kohr D. et al. Parallel Programming in OpenMP. Morgan Kaufmann Publishers, 2000.
12. Jeffers, J., Reinders, J. (2013). Intel Xeon Phi coprocessor high performance programming. Newnes.
13. Jeffers, J., Reinders, J., & Sodani, A. (2016). Intel Xeon Phi Processor High Performance Programming: Knights Landing Edition. Morgan Kaufmann.
14. Jeffers, J., Reinders, J. (2014). High Performance Parallelism Pearls Volume One: Multicore and Many-core Programming Approaches. Morgan Kaufmann.

15. Jeffers, J., Reinders, J. (2015). High Performance Parallelism Pearls Volume Two: Multi-core and Many-core Programming Approaches. Morgan Kaufmann.
16. Sanders, J., Kandrot, E. (2010). CUDA by example: an introduction to general-purpose GPU programming. Addison-Wesley Professional.
17. Pharr, M., & Fernando, R. (2005). Gpu gems 2: programming techniques for high-performance graphics and general-purpose computation. Addison-Wesley Professional.
18. Nguyen, H. (2007). Gpu gems 3. Addison-Wesley Professional.
19. Quinn M. J. (2004). Parallel Programming in C with MPI and OpenMP. – New York, NY: McGraw-Hill.
20. Prasad, S. K. et al. 2012. NSF/IEEE-TCPP Curriculum Initiative on Parallel and Distributed Computing – Core Topics for Undergraduates, Version I, Online: http://www.cs.gsu.edu/~tcpp/curriculum
21. Prasad, S. K., Gupta, A., Rosenberg, A. L., Sussman, A., Weems, C. C. (Eds.). (2015). Topics in parallel and distributed computing: introducing concurrency in undergraduate courses. Morgan Kaufmann.
22. Voevodin V., Gergel V., Popova N. (2015) Challenges of a Systematic Approach to Parallel Computing and Supercomputing Education. In: Hunold S. et al. (eds) Euro-Par 2015: Parallel Processing Workshops. Euro-Par 2015. Lecture Notes in Computer Science, 9523, 90–101. Springer, Cham.
23. Gergel V., Liniov A., Meyerov I., Sysoyev A. NSF/IEEE-TCPP Curriculum Implementation at University of Nizhni Novgorod / Proceedings of Fourth NSF/TCPP Workshop on Parallel and Distributed Computing Education, 2014. P. 1079-1084.
24. Gergel V., Kozinov E., Linev A., Shtanyk A. (2016) Educational and Research Systems for Evaluating the Efficiency of Parallel Computations. In: Carretero J. et al. (eds) Algorithms and Architectures for Parallel Processing. ICA3PP 2016. Lecture Notes in Computer Science, 10049, 278-290. Springer, Cham.