Суперкомпьютерный консорциум университетов России
Российская академия наук

СУПЕРКОМПЬЮТЕРНЫЕ ДНИ В РОССИИ

Труды международной конференции

25–26 сентября 2023 г., Москва

МАКС ПРЕСС

МОСКВА – 2023

УДК 519.7
ББК 22.18
С89

https://elibrary.ru/vdhwjz

Под редакцией члена-корреспондента РАН *Вл. В. Воеводина*

Суперкомпьютерные дни в России : Труды международной конференции. 25–26 сентября 2023 г., Москва / Под. ред. Вл. В. Воеводина. – Москва : МАКС Пресс, 2023. – 224 с.

С89

ISBN: 978-5-317-07071-7 e- ISBN: 978-5-317-07070-0
https://10.29003/m3478.978-5-317-07070-0

Данный сборник содержит полные статьи на русском языке, короткие статьи и аннотации стендовых докладов, включенных в программу Международной конференции «Суперкомпьютерные дни в России».

УДК 519.7
ББК 22.18

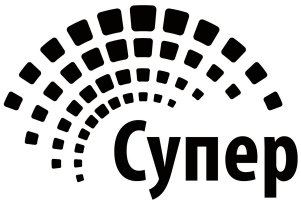
Russian Supercomputing Days : Proceedings of the International Conference. September 25–26, 2023, Moscow, Russia / Ed. by Vl. Voevodin. – Moscow : MAKS Press, 2023. – 224 p.

ISBN: 978-5-317-07071-7 e- ISBN: 978-5-317-07070-0
https://10.29003/m3478.978-5-317-07070-0

The book contains full papers in Russian, short papers and poster abstracts included in the agenda of the International Conference “Russian Supercomputing Days”.

Подробную информацию о конференции можно найти в сети Интернет по адресу http://RussianSCDays.org

ISBN: 978-5-317-07071-7
e- ISBN: 978-5-317-07070-0
https://10.29003/m3478.978-5-317-07070-0



Суперкомпьютерные дни в России

Полные и короткие статьи

Efficient CFD technologies for high performance combustion modeling in academic research

A.D. Kiverin, I.S. Yakovenko

Joint Institute for High Temperatures of the Russian Academy of Sciences (JIHT RAS)

Computational technologies implemented in Joint Institute for High Temperatures of RAS for high-performance computing of the complex transient combustion processes are briefly described. Computing performance of the developed CFD package is tested on different computing environments and the most vivid examples of the package successful application for solving various problems in a field of numerical combustion are presented. The problem of validation, verification and benchmarking of the CFD software aimed at combustion modeling is discussed.

Keywords: high-performance computing, combustion modeling, MPI, OpenMP, reproducible research

1. Introduction

In recent decades, computational fluid dynamics has become one of the main tools for solving a great variety of engineering problems in such areas as aerospace and automotive, architecture, the process industry, and many others. Such diversity in applications could not be possible without significant progress in the development of precise and efficient numerical approaches for solving systems of partial differential equations which constitute physical models of the considered phenomena. The no less important role also belongs to the continuous improvement in knowledge of the underlying physical principles and mechanisms. However, a more crucial factor that allowed us to utilize computational approaches for detailed analysis of industrial problems is a relatively recent breakthrough in the field of high-performance computing technologies.

Contemporary supercomputers can provide enough computational resources for comprehensive modeling of complex fluid- or gas dynamic flows in real-scale geometries and with an account of heat transfer, diffusion in multicomponent mixtures and multiphase transport, chemical transformations in reactive media, and many other processes. And while some problems of continuum motion are not very demanding, there are still areas in which it is necessary to utilize the full capacity of available computing power. One of the vivid examples of such problems is combustion modeling. To perform a numerical simulation of combustion processes in multicomponent gaseous mixtures, one needs to integrate a conventional Navier-Stokes equations system coupled with chemical source terms. The overall governing equations system can be considered as stiff, as different spatial and temporal scales have to be resolved. Most severe restrictions are imposed by chemical transformations, along with small scales of the characteristic spatial length of the reaction zone chemical kinetics requires a system of ordinary differential equations to be solved in each cell of the computational grid. For hydrocarbon-based mixtures, the amount of additional ODEs is up to several thousand, and the computational overhead associated with the solution of chemical kinetics equations becomes much greater than the solution of the governing equations system itself. For industrial applications, it is common to utilize simplified models aimed to alleviate restrictions imposed by chemical kinetics, e.g. to utilize reduced chemical kinetic schemes with fewer ODEs or tabulate solutions for combustion kinetics, to implement sub-grid combustion models to reduce the number of cells in the computational domain. While those assumptions can provide reasonable insight on the average quantitative behavior of the flow, sufficient for engineering purposes, in the field of combustion scientific research, oversimplified models often lead to the inability to reproduce the detailed structure of the flow and, as a result, restricts the correct interpretation of the physical phenomena. Thereby, academic

research requires even more efficient techniques to carry on numerical simulations involving all accessible computational resources of different architecture, such as conventional personal computers supplied with multi-core processors with shared memory or complex cluster systems consisting of a large number of nodes with distributed memory. In this paper, we introduce an in-house software package aimed at detailed high-performance numerical simulation of combustion processes, which was developed and used in the Joint Institute for High Temperatures of RAS over the past decades for solving a wide range of problems in combustion physics.

2. Software package

Although there is a wide range of proprietary CFD programming packages that are able to operate efficiently on multi-core and cluster computers source code of that software is usually closed. That greatly restricts the scientific utilization of those packages, as academic research always requires some freedom to implement new physical models or numerical algorithms. On the other hand, available open-source solutions tend to be too complicated and equipped with many unnecessary or poorly implemented features. That makes it challenging to understand code structure and steepens the learning curve for successful utilization of the package for a particular task. That is why it is so common in the scientific community to develop in-house software aimed at solving problems in a particular field of interest. The major drawback of such an approach is that numerical research becomes poorly reproducible. Developing well-documented open-source research code can be the way to increase the transparency of the research, enhance collaboration and help the community to better understand the peculiarities of the utilized computational approaches and considered problem setups [1]. A remarkable example of such codes is Pencil Code developed under the auspices of Nordic Institute for Theoretical Physics [2] (https://github.com/pencil-code/pencil-code).

In JIHT Reactive CFD software package we developed a simple yet robust software framework using features of 2003 standard of the Fortran programming language. The computational package is targeted to perform computations on both multi-core and cluster computer architectures run by Linux as well as Windows operating systems (see package schematic overview in Fig. 1). It consists of three main elements namely *package_library*, *computing_module* and *package_interface*. In *package_library* necessary mathematical abstractions such as scalar, vector and tensor fields are implemented in a form of Fortran derived types. Also *package_library* provides data input/output utilities, types of boundary conditions and other features. The core library is designed in accordance with the object-oriented concept with deviating from those principles only where encapsulation can lead to severe performance losses [14]. In *computing_module* several solvers for gas dynamic convection, multicomponent diffusion, heat transfer and chemical kinetics are implemented on the basis of data types defined in *package_library*. Different levels of parallelism are present in *package_library* and *computing_module*. Solvers residing in *computing_module* are equipped with OpenMP directives for computations on shared memory architectures. For cluster computations MPI wrappers on the basis of the derived field data types are specially implemented in *package_library* for interprocessor communications. Those wrappers allow high-level parallelization of the solvers in *computing_module* via domain decomposition paradigm. Despite the benefits, development of the fully capable graphic user interface is a challenging problem that can be avoided with the use of more simple interface solutions. In our case, *package_interface* represents itself merely a short Fortran executable for the problem setup. Format of the problem setup files produced via *package_interface* is general, so the solution procedure can be started on a personal computer as well as on a cluster machine, only *computing_module* has to be recompiled to operate on a specific system. That behavior is very useful as one can transfer computations started on one machine to another, maybe with different architecture or operating system.

One of the major topics in contemporary numerical combustion research concerns the choice

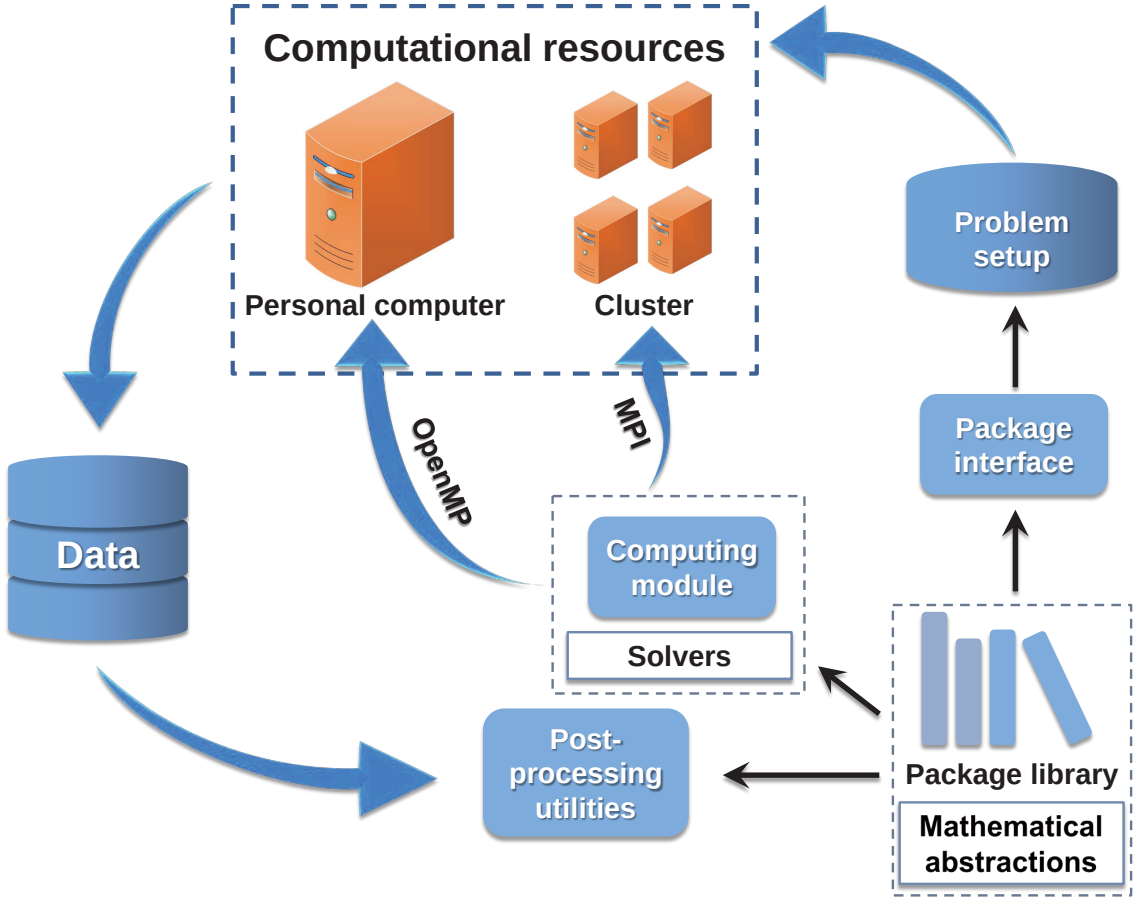


Figure 1. Schematic workflow of JIHT Reactive CFD package.

of the appropriate numerical approach suitable for detailed combustion modeling. Based on the developed computational package, several gas dynamic solvers were implemented. Among them are the conventional euler-lagrange technique (CPM) and the novel CABARET numerical technique that possess unique dissipation-free and low-dispersion features [7]. Both methods are second order in space and have the smallest computational stencil that makes them very efficient for cluster computing, as the amount of data involved in interprocessor communications is minimized. The performance of the package on multi-core and cluster computers is presented in Fig. 2.

Load imbalance is one of the crucial issues related to the detailed simulation of reactive flows via parallel computing techniques. Due to the intrinsic nature of the flame dynamics, chemical reactions are proceeding intensively only in the thin reaction zone, which is usually by several orders of magnitude smaller than the characteristic size of the computational domain. As a result, the most time-consuming stage of the simulation, namely the solution procedure for equations of chemical kinetic, is carried out only by some part of the processors or threads involved in the solution process. This leads to severe load imbalance when using the static domain decomposition approach. Speedup dependencies on the number of cores and threads given in Fig. 2 are obtained for the typical problem setup of a shock-tube experiment where chemical transformations proceed only in the compressed and heated gaseous mixture behind the shock wave front. In this case, parallel performance is decreased due to uneven load balance. To reduce the effect of load imbalance, the performance of the software package was assessed for the problems of the volumetric thermal explosion, where a gas mixture in the computational domain is initially heated to high temperature, so the reaction immediately begins in the entire

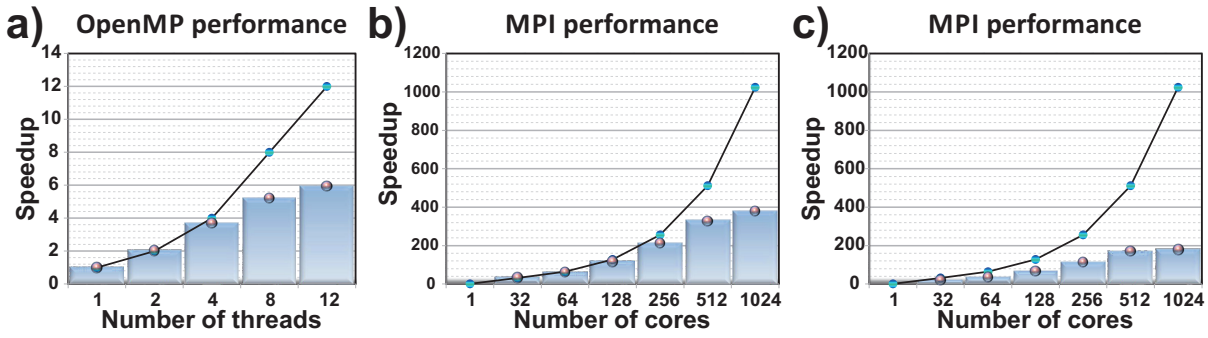


Figure 2. a) OpenMP performance of the package. Multi-core processor Intel® Core™ i7-3930K, 3.2 GHz. b) MPI performance of the package. JSCC RAS MVS-10P cluster, Intel® Xeon™ E5-2697Av4, 2.6 GHz. c) MPI performance of the package. MSU "Lomonosov" cluster, Intel® Xeon™ X5570, 2.93 GHz. Ideal speedup is represented by black line.

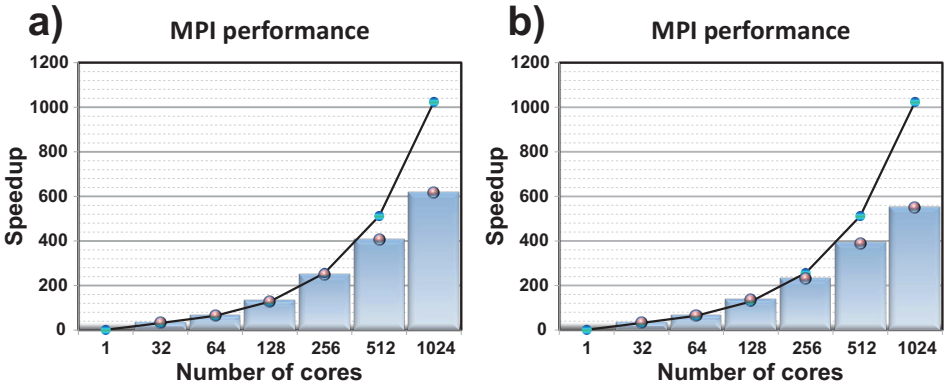


Figure 3. MPI performance of the package. JSCC RAS MVS-10P cluster, Intel® Xeon™ E5-2697Av4, 2.6 GHz. a) Thermal explosion test b) Gas dynamics of inert mixture test. Ideal speedup is represented by black line.

domain, and gas dynamics of inert gaseous mixtures. Obtained results are presented in Fig. 3. It can be seen from Fig. 3 that enhanced load balancing leads to higher parallel performance. To achieve optimal load balance for the transient combustion processes of combustion waves propagation, more sophisticated techniques of parallel computing should be implemented, for example, dynamic domain decomposition approaches supported with automatic load balancing algorithms [12].

3. Transient combustion modeling

The developed CFD software package allows detailed numerical research of the transient combustion processes, such as flame acceleration and deflagration to detonation transition, detonation propagation regimes, ultra-lean flame structure and stability, detonation initiation in gaseous mixtures with non-uniform dispersed particles distribution, direct detonation initiation and many others. The dynamics of the unsteady combustion processes are determined mainly by the chosen model of chemical kinetics. In practice [3, 4, 6, 11], it is necessary to use detailed chemical kinetic mechanisms to provide a satisfactory qualitative description of the peculiarities of unsteady combustion. One of the key features of detailed chemical mechanisms is the ability to reproduce the evolution of freely propagating flames with high precision in a wide range of conditions. On this basis, one can obtain laminar burning velocity, which is one of

the fundamental characteristics of the combustion process. Dependencies of the laminar burning velocity on the mixture composition obtained experimentally and by numerical analysis of freely propagating flames using discussed software package is presented in Fig. 4. One can see that numerical results are in good agreement with experimental ones. For detailed numerical analysis and interpretation of real physical processes, reasonable quantitative reproduction of the peculiarities of gas dynamic flows in the vicinity of the flame front is also of importance and has been demonstrated in our previous articles [5, 13].

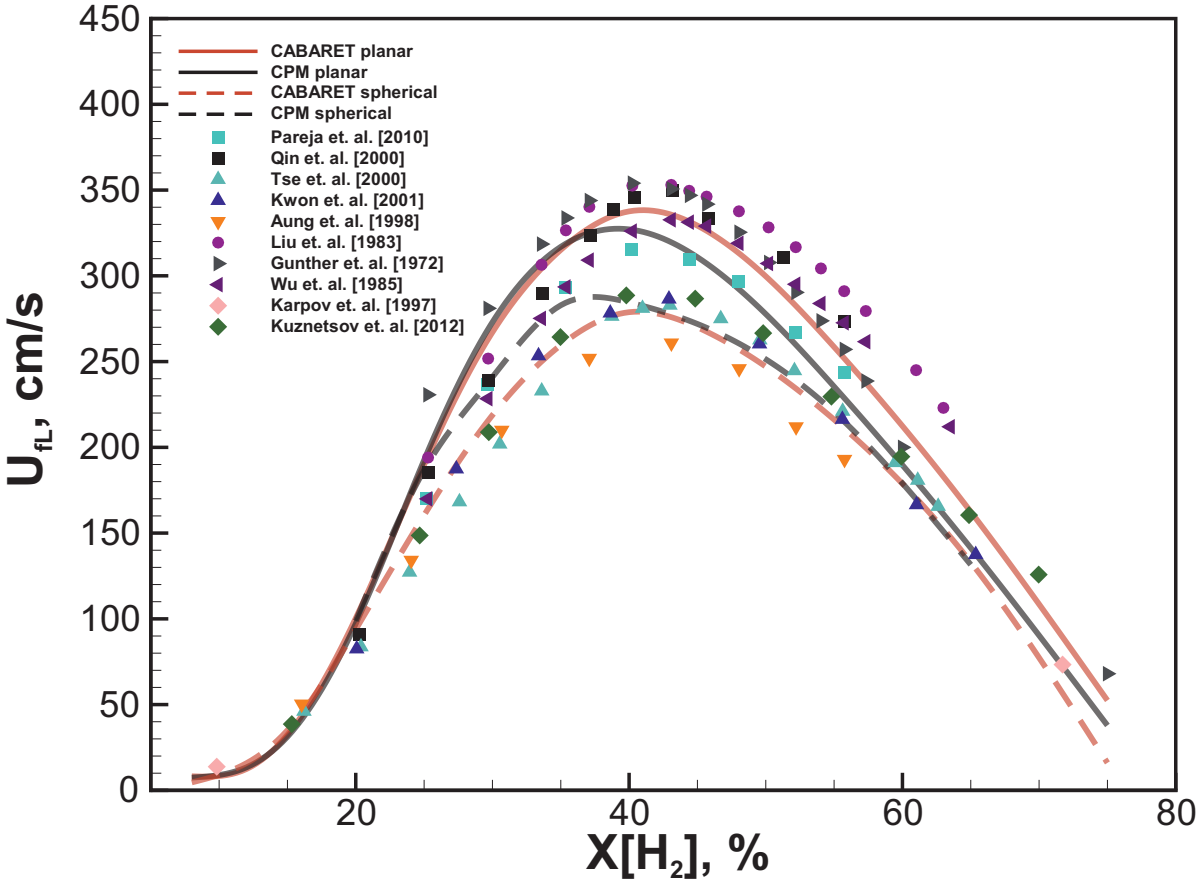


Figure 4. Computed laminar burning velocities along with experimental measurements adopted from [10] and [9].

Among the most representative results obtained via this software is three-dimensional modeling of the flame acceleration and deflagration to detonation transition processes in the rectangular channel with no-slip walls. Calculations were performed on the fine grids ($\Delta x = 0.025$ mm, 400 millions of cells), using a detailed chemical kinetic scheme (see. Fig. 5 taken from [5]). Another vivid example is the recently obtained result on the numerical simulation of the flow patterns behind a shock wave that allowed better understanding of the mechanisms of ignition kernels formation during the shock-tube experiment (see. Fig. 6 from [8]).

Conclusion

Computational fluid dynamics of the reactive media is still challenging and computationally demanding area in the field of numerical modeling. To perform detailed simulation of the combustion phenomena suitable for scientific research purposes, a lot of requirements related to reliability and accuracy have to be met. It is necessary to implement contemporary high-performance computing techniques for the effective use of available computational resources,

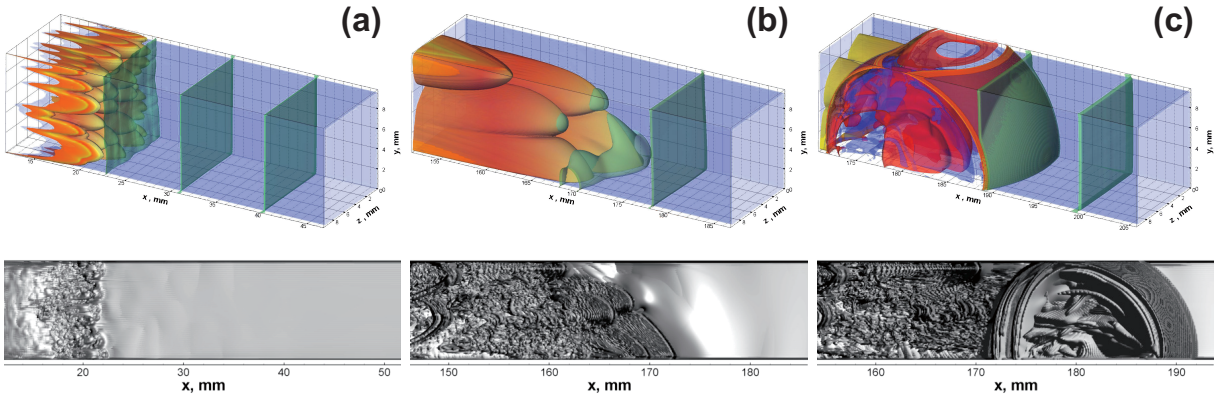


Figure 5. Numerical analysis of 3D flame patterns in the process of flame propagation and transition to detonation in channel filled with hydrogen-oxygen mixture [5].

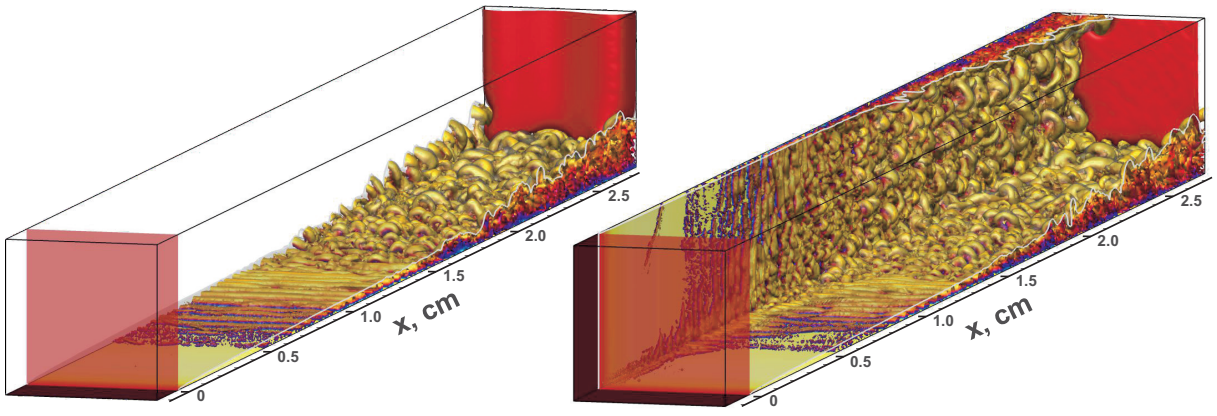


Figure 6. Mathematical modeling of the flow pattern and ignition behind shock wave in the shock-tube experiment [8].

including multi-core and cluster computing, to meet these requirements. Developed in JIHT RAS Reactive CFD software provides a simple and robust framework for high-performance simulations of the reactive flows via different computer architectures and allows the performing of complex simulations of the combustion processes aimed at the solution of acute problems of contemporary combustion theory and applications.

Acknowledgements

The research was carried out using the equipment of the shared research facilities of the HPC computing resources at Lomonosov Moscow State University and using supercomputers at the Joint Supercomputer Center of the Russian Academy of Sciences (JSCC RAS).

References

1. Barnes, N.: Publish your computer code: it is good enough. Nature **467**(7317), 753–753 (2010). DOI 10.1038/467753a. URL http://www.nature.com/doifinder/10.1038/467753a

2. Brandenburg, A., Dobler, W.: Hydromagnetic turbulence in computer simulations. Computer Physics Communications **147**(1-2), 471–475 (2002). DOI 10.1016/S0010-4655(02)00334-X

3. Dounia, O., Vermorel, O., Misdariis, A., Poinsot, T.: Influence of kinetics on DDT simulations. Combustion and Flame **200**, 1–14 (2019). DOI 10.1016/j.combustflame.2018.11.009. URL https://doi.org/10.1016/j.combustflame.2018.11.009

4. Ivanov, M., Kiverin, A., Liberman, M.: Flame acceleration and DDT of hydrogen–oxygen gaseous mixtures in channels with no-slip walls. International Journal of Hydrogen Energy **36**(13), 7714–7727 (2011). DOI 10.1016/j.ijhydene.2011.03.134. URL http://linkinghub.elsevier.com/retrieve/pii/S0360319911007245

5. Ivanov, M., Kiverin, A., Yakovenko, I., Liberman, M.: Hydrogen–oxygen flame acceleration and deflagration-to-detonation transition in three-dimensional rectangular channels with no-slip walls. International Journal of Hydrogen Energy **38**(36), 16,427–16,440 (2013). DOI 10.1016/j.ijhydene.2013.08.124. URL http://linkinghub.elsevier.com/retrieve/pii/S0360319913021344

6. Ivanov, M.F., Kiverin, A.D., Liberman, M.A., Fortov, V.E.: The flame-acceleration mechanism and transition to detonation of a hydrogen-oxygen mixture in a channel. Doklady Physics **55**(10), 480–484 (2010). DOI 10.1134/S1028335810100022. URL http://link.springer.com/10.1134/S1028335810100022

7. Karabasov, S.A., Goloviznin, V.M.: Compact Accurately Boundary-Adjusting high-REsolution Technique for fluid dynamics. Journal of Computational Physics **228**(19), 7426–7451 (2009). DOI 10.1016/j.jcp.2009.06.037. URL http://dx.doi.org/10.1016/j.jcp.2009.06.037

8. Kiverin, A.D., Yakovenko, I.S.: Evolution of wave patterns and temperature field in shock-tube flow. Physical Review Fluids **3**(5), 053,201 (2018). DOI 10.1103/PhysRevFluids.3.053201. URL https://link.aps.org/doi/10.1103/PhysRevFluids.3.053201

9. Pareja, J., Burbano, H.J., Ogami, Y.: Measurements of the laminar burning velocity of hydrogen–air premixed flames. International Journal of Hydrogen Energy **35**(4), 1812–1818 (2010). DOI 10.1016/j.ijhydene.2009.12.031. URL http://linkinghub.elsevier.com/retrieve/pii/S0360319909019387

10. Sánchez, A.L., Williams, F.A.: Recent advances in understanding of flammability characteristics of hydrogen. Progress in Energy and Combustion Science **41**(1), 1–55 (2014). DOI 10.1016/j.pecs.2013.10.002Review. URL http://dx.doi.org/10.1016/j.pecs.2013.10.002

11. Smirnov, N.N., Nikitin, V.F., Stamov, L.I., Nerchenko, V.A., Tyrenkova, V.V.: Numerical Simulations of Gaseous Detonation Propagation Using Different Supercomputing Architechtures. International Journal of Computational Methods **14**(04), 1750,038 (2017). DOI 10.1142/S0219876217500384. URL http://www.worldscientific.com/doi/abs/10.1142/S0219876217500384

12. Streng, M.: Load balancing for computational fluid dynamics calculations. In: P. Wesseling (ed.) High Performance Computing in Fluid Dynamics, pp. 145–172. Springer Netherlands, Dordrecht (1996)

13. Yakovenko, I., Ivanov, M., Kiverin, A., Melnikova, K.: Large-scale flame structures in ultra-lean hydrogen-air mixtures. International Journal of Hydrogen Energy (2017). DOI 10.1016/j.ijhydene.2017.11.138. URL https://www.sciencedirect.com/science/article/pii/S0360319917345524?via%3Dihub

14. Zaghi, S.: OFF, Open source Finite volume Fluid dynamics code: A free, high-order solver based on parallel, modular, object-oriented Fortran API. Computer Physics Communications **185**(7), 2151–2194 (2014). DOI 10.1016/j.cpc.2014.04.005. URL http://dx.doi.org/10.1016/j.cpc.2014.04.005

HPC, Big Data, ML – три базисных вектора магистратуры по программной инженерии

Н.С. Силкина, Л.Б. Соколинский, С.У. Турлакова

Южно-Уральский государственный университет

В статье описаны принципы разработки магистерской программы «Искусственный интеллект и инженерия данных» по направлению 09.03.04 Программная инженерия. Работа выполнялась в рамках гранта Минобрнауки России на разработку программ бакалавриата и программ магистратуры по профилю «искусственный интеллект», а также на повышение квалификации педагогических работников образовательных организаций высшего образования в сфере искусственного интеллекта, с учетом потребностей регионального рынка труда, традиций и достижений научно-педагогической школы университета и требований федерального законодательства. Уникальность этой программы заключается в объединении трех актуальных направлений в сфере информационных технологий: высокопроизводительных вычислений, анализа и обработки больших данных и машинного обучения. В работе описаны структурные взаимосвязи дисциплин внутри каждого блока, обеспечивающие формирование соответствующих компетенций. Представлено краткое описание содержания профильных курсов.

Ключевые слова: программа магистратуры, учебный план, искусственный интеллект, программная инженерия, высокопроизводительные вычисления, большие данные, машинное обучение.

1. Введение

В связи с увеличением вычислительных возможностей программно-аппаратных комплексов, в том числе в результате использования графических процессоров и распределенных архитектур вычислительных систем на сегодняшний день стало доступным широкое применение интеллектуального анализа больших данных, а также методов машинного обучения на базе множества вычислительных систем, организованных по принципу нейронных сетей (по аналогии с человеческим мозгом), что привело к значительному повышению качества разрабатываемых технологических решений. В октябре 2019 года вышел Указ Президента РФ «О развитии искусственного интеллекта в Российской Федерации» [1], в котором утверждается Национальная стратегия развития искусственного интеллекта на период до 2030 года. Согласно данным документам, искусственный интеллект представляет собой комплекс технологических решений, позволяющий имитировать когнитивные функции человека (включая самообучение и поиск решений без заранее заданного алгоритма) и получать при выполнении конкретных задач результаты, сопоставимые, как минимум, с результатами интеллектуальной деятельности человека. Комплекс технологических решений включает в себя информационно-коммуникационную инфраструктуру, программное обеспечение, процессы и сервисы по обработке данных и поиску решений.

На текущий момент указанным направлениям ИТ уделяется внимание во многих программах магистратуры. Например, в МГУ реализуется программа магистратуры «Открытые информационные системы» направления 02.04.02 «Фундаментальная информатика и информационные технологии» [2]. В университете ИТМО реализуются ряд программ магистратуры [3]: «Большие данные и машинное обучение», «Глубокое обучение и генеративный искусственный интеллект», «Программирование и искусственный интеллект» направления 01.04.02 «Прикладная математика и информатика», «Искусственный интеллект» направления 09.04.01 «Информатика и вычислительная техника», «Нейротехнологии и программная инженерия», «Проектирование и разработка систем больших данных» направления 09.04.04 «Программная инженерия». Данные программы посвящены модулю Интеллектуальный анализ данных и машинное обучение.

Таким образом, подготовка специалистов в области параллельных вычислений, анализа данных и машинного обучения в настоящее время стала очень актуальной задачей.

2. Структура образовательной программы «Искусственный интеллект и инженерия данных»

Образовательная магистерская программа «Искусственный интеллект и инженерия данных» [4] по направлению подготовки «09.04.04 Программная инженерия» была разработана на основе ФГОС ВО, профессионального стандарта «06.003 Архитектор программного обеспечения», в рамках Соглашения с Минобрнауки России от 28.09.2021 г. № 075-15-2021-1047 о предоставлении из федерального бюджета гранта в форме субсидии на разработку программ бакалавриата и программ магистратуры по профилю «искусственный интеллект», а также на повышение квалификации педагогических работников образовательных организаций высшего образования в сфере искусственного интеллекта с учетом потребностей регионального рынка труда, традиций и достижений научно-педагогической школы университета и требований федерального законодательства. Образовательная программа была разработана с участием регионального образовательного партнера ФГБОУ ВО «Челябинский государственный университет», а также совместно с ИТ-компанией ООО "ТРИДИВИ". Программа направлена непосредственно на подготовку профессиональных программистов, специалистов в области искусственного интеллекта, которые будут создавать сами программные механизмы. В 2022 году в ЮУрГУ и ЧелГУ состоялся первый набор студентов на данную программу.

Образовательная программа реализует модель компетенций в сфере искусственного интеллекта для уровня образования «Магистратура» с ориентацией на разработку систем искусственного интеллекта [5]. Модель компетенций была реализована в образовательной программе с учетом рекомендаций Министерства науки и высшего образования Российской Федерации. Основным блоком образовательной программы является блок «Дисциплины (модули)», который содержит 23 дисциплины (81 зачетная единица, зе), из них формируемая часть и профиль содержат 12 дисциплин. В таблице 1 представлена структура блока «Дисциплины (модули)» образовательной программы

Таблица 1. Структура блока «Дисциплины (модули)» образовательной программы «Искусственный интеллект и инженерия данных»

Разделы	I семестр	II семестр	III семестр
Искусственный интеллект	• Разработка систем ИИ на языке Python • Разработка интеллектуальных систем на языке R • Машинное обучение	• Этические аспекты применения ИИ	• Нейробайесовские методы в машинном обучении
Нейронные сети	• Глубокие нейронные сети	• Анализ естественного языка методами ИИ • Компьютерное зрение	• Нейросетевые технологии в задачах синтетических медиа
Обработка больших данных	-	• Интеллектуальный анализ данных • Методы и системы обработки больших данных	• Анализ и прогнозирование временных рядов • Технологии распределенной обработки данных
Высокопроизводительные вычисления	-	• Архитектура распределенных вычислительных систем	• Технологии параллельного программирования • Облачные технологии

Разделы	I семестр	II семестр	III семестр
Общие дисциплины	• Иностранный язык в профессиональной деятельности • Методология научного познания • Защита информации методами ИИ • Научно-практический семинар «ИИиИД»	• Иностранный язык в профессиональной деятельности • Объектно-ориентированные CASE-технологии • Научно-практический семинар «ИИиИД»	• Современные методы DevOps • Управление проектами в сфере ИИ • Научно-практический семинар «ИИиИД»

Вторым блоком образовательной программы является блок «Практика», который содержит учебную и производственную практики в объеме 30 зе. В течение IV семестра студенты проходят производственную практику в формате научно-исследовательской работы. В ходе практики студенты проводят исследования, связанные с будущей выпускной квалификационной работой. Третий блок «Государственная итоговая аттестация» включает подготовку и защиту выпускной квалификационной работы магистра (9 зе).

Таким образом, около 40% учебной нагрузки образовательной программы посвящено научно-исследовательской работе, практике и выполнению выпускной квалификационной работы, менее 30% учебной нагрузки посвящены изучению фундаментальных и прикладных дисциплин в области искусственного интеллекта и анализа данных, около 10% учебной нагрузки отводятся для изучения технологий высокопроизводительных вычислений, оставшаяся часть учебной нагрузки отводится для общих дисциплин.

2. Искусственный интеллект и машинное обучение

Обучение по данному разделу проходит в течение трех семестров. В первом семестре студенты изучают дисциплины:

- Глубокие нейронные сети, 3 зе;
- Защита информации методами ИИ, 2 зе;
- Машинное обучение, 4 зе;
- Разработка интеллектуальных систем на языке R, 3 зе;
- Разработка систем ИИ на языке Python, 4 зе.

Дисциплина «Глубокие нейронные сети» включает 32 часа лекций и 16 часов практических занятий. В дисциплине изложены наиболее важные понятия, определения и принципы построения нейронных сетей. В курс входят следующие разделы: введение в искусственные нейронные сети, градиентный спуск, метод обратного распространения ошибки, стоимостная функция на основе перекрестной энтропии, переобучение нейронной сети и регуляризация, техники, улучшающие обучение нейронных сетей, сверточные нейронные сети, рекуррентные нейронные сети, автокодировщики, история развития нейронных сетей.

Дисциплина «Защита информации методами ИИ» включает 16 часов лекций и 16 часов практических занятий. В дисциплине изложены основные направления применения методов искусственного интеллекта в задачах защиты информации: противодействие сетевым атакам путем интеллектуального анализа данных о сетевом трафике, обнаружение вредоносной активности на вычислительных узлах, особенности реализации антивирусных программ с использованием ИИ, противодействие мошенничеству в прикладных сервисах и фильтрация спам-рассылок в почтовых сервисах.

Дисциплина «Машинное обучение» включает 32 часа лекций и 32 часа практических занятий. В дисциплине изложены основные типы задач, решаемых с помощью методов машинного обучения, основы подготовки входных данных и оценки качества моделей, готовые реализации методов машинного обучения в современных библиотеках: метод kNN, деревья решений и ансамблевые методы. Рассматриваются методы понижения размерности и визуализации данных, а также рассматривается задача отбора признаков.

Дисциплина «Разработка интеллектуальных систем на языке R» включает 16 часов лекций и 32 часа практических занятий. В дисциплине изложены принципы программирования на языке R, рассматриваются имитационное моделирование и профилирование кода в R, основы

статистического анализа и регрессионный анализ данных в R, визуализация данных, а также введение в машинное обучение в R.

Дисциплина «Разработка систем ИИ на языке Python» включает 16 часов лекций и 48 часов практических занятий. В дисциплине изложены основные конструкции и базовые типы языка Python, объектно-ориентированное программирование в Python, применение языка для анализа и визуализации данных (библиотеки NumPy, Pandas, Matplotlib), веб-разработка на Python, в том числе с использованием обученной модели нейронной сети.

Во втором семестре студенты изучают дисциплины:

- Анализ естественного языка методами ИИ, 3 зе;
- Компьютерное зрение, 4 зе;
- Нейробайесовские методы в машинном обучении, 3 зе;
- Нейросетевые технологии в задачах синтетических медиа, 3 зе;
- Этические аспекты применения ИИ, 3 зе.

Дисциплина «Анализ естественного языка методами ИИ» включает 16 часов лекций и 32 часа практических занятий. В дисциплине изложены основы обработки естественного языка, в том числе предобработка текстовых данных, лемматизация и стемминг. Рассматриваются основные задачи обработки естественного языка: классификация текстов на основе частотных мер, на основе методов машинного обучения и глубоких нейронных сетей, а также решение задачи кластеризации текстов. В курсе также рассматриваются трансформеры, генеративные нейронные сети, а также принципы построения диалоговых систем.

Дисциплина «Компьютерное зрение» включает 16 часов лекций и 32 часа практических занятий. В дисциплине изложены вопросы применения сверточных нейронных сетей в задачах компьютерного зрения, применение библиотек OpenCV и Pillow для обработки изображений. Рассмотрена задача сегментации изображений и детекции объектов, в том числе с применением SSD, YOLO, FASTER RCNN, Mask R-CNN. В курсе рассматриваются вопросы разметки датасета изображений и аугментации данных. Рассматривается задача оценки схожести двух изображений и поиска изображений по содержанию. Рассматривается задача распознавания текста на изображении, обнаружения и трекинга объектов в видеопотоке. Рассматриваются вопросы компрессии моделей компьютерного зрения.

Дисциплина «Нейробайесовские методы в машинном обучении» включает 16 часов лекций и 16 часов практических занятий. В дисциплине изложены вопросы стохастического вариационного вывода, вариационного автокодировщика, байесовских нейронных сетей, а также генеративно-состязательных сетей.

Дисциплина «Нейросетевые технологии в задачах синтетических медиа» включает 16 часов лекций и 32 часа практических занятий. В дисциплине изложены виды синтетических медиа, рассматриваются вопросы анализа аудио, распознавания и синтеза речи, анализа и синтеза изображений и видео по содержанию, синтеза лица, генерации текста.

Дисциплина «Этические аспекты применения ИИ» включает 16 часов лекций и 16 часов практических занятий. В дисциплине рассматриваются основные нормативно-правовые документы в области этики искусственного интеллекта, этические аспекты безопасности, доверенности и корректности работы автономных интеллектуальных систем, а также вопросы этики самообучающихся автономных интеллектуальных систем. В курсе также рассматриваются этические принципы для разработки рекомендательных систем в различных областях (медицина, робототехника и др.), этические кодексы ведущих компаний по разработке систем ИИ.

В третьем семестре студенты изучают дисциплину «Управление проектами в сфере ИИ», 3 зе. Дисциплина включает 16 часов лекций и 16 часов практических занятий. В дисциплине изложены вопросы проектного и продуктового управления, рассматриваются особенности проектов в сфере ИИ, а также гибкие методы управления проектами. В курсе изучаются принципы управления RnD командами на примере передовых ИТ-компаний в сфере ИИ.

Научно-практический семинар «ИИиИД», 12 зе. Дисциплина проводится в течение трех семестров и включает 64 часа практических занятий в каждом семестре. В рамках дисциплины реализуется план научно-практических семинаров по ключевым направлениям искусственного интеллекта в рамках национальной стратегии развития ИИ: компьютерное зрение, обработка естественного языка, синтез и распознавание речи, предиктивная аналитика и интеллектуальная поддержка принятия решений, системы анализа и обработки больших данных, перспективные

методы искусственного интеллекта. От обзорных докладов по актуальным задачам и проблемам в рамках семинара в ходе освоения магистерской программы идет переход к выполнению НИОКР и представлению промежуточных результатов по выбранным магистрантам темам. Выполнение проектов завершается представлением результатов: разработанных интеллектуальных систем или новых технологий (или отдельных модулей, прикладных сервисов) по задачам индустриальных партнеров или научных лабораторий и институтов.

3. Интеллектуальный анализ данных

Обучение интеллектуальному анализу данных (Data Mining) осуществляется во втором и третьем семестрах. Во втором семестре студенты изучают дисциплины:

- Интеллектуальный анализ данных, 4 зе;
- Методы и системы обработки больших данных, 2 зе.

Дисциплина «Интеллектуальный анализ данных» включает 32 часа лекций и 32 часа практических занятий. В дисциплине изложены основные понятия, а также особенности решения задач интеллектуального анализа данных: поиск шаблонов, классификация данных, кластеризация данных, поиск аномалий.

Дисциплина «Методы и системы обработки больших данных» включает 16 часов лекций и 16 часов практических занятий. В дисциплине изложены основные понятия обработки запросов в системах баз данных, а также вопросы разбора, конверсии, логической оптимизации запроса. Рассматриваются вопросы организации системы баз данных, применения индексов, реализации исполнителя запросов, а также рассматриваются алгоритмы соединения.

В третьем семестре студенты изучают дисциплины:

- Анализ и прогнозирование временных рядов, 3 зе;
- Технологии распределенной обработки данных, 3 зе.

Дисциплина «Анализ и прогнозирование временных рядов» включает 16 часов лекций и 32 часа практических занятий. В дисциплине изложены вопросы поиска подпоследовательностей по образцу, поиска аномалий во временных рядах, понятие матричного профиля временного ряда и примитивов анализа данных на его основе, а также вопросы восстановления пропусков и прогноза значений временного ряда.

Дисциплина «Технологии распределенной обработки данных» включает 32 часа лекций и 16 часов практических занятий. В дисциплине изложены вопросы секционирования и репликации данных, дается введение в Hadoop и его экосистему (YARN, Apache Kafka и др.). Рассматриваются вопросы анализа данных в Hadoop на основе Apache Spark. В курсе также рассматриваются NoSQL-решения.

4. Высокопроизводительные вычисления

Обучение студентов высокопроизводительным вычислениям осуществляется кафедрой системного программирования ЮУрГУ уже с 2004 года. В 2010-2012 годы ЮУрГУ участвовал в реализации всероссийского проекта «Суперкомпьютерное образование» [6, 7]. В ходе проекта Суперкомпьютерным консорциумом университетов России были разработаны и внедрены в российских образовательных учреждениях курсы по высокопроизводительным вычислениям, в том числе «Многопоточные вычисления на основе технологий MPI и OpenMP», «Многопоточные вычисления на основе технологий CUDA и OpenCL», «Суперкомпьютерные технологии для гибридных кластерных систем» и др. В разработке учебно-методических материалов по высокопроизводительным вычислениям принимали участие и сотрудники кафедры системного программирования [8, 9].

Обучение магистров направления 09.04.04 высокопроизводительным вычислениям осуществляется во втором и третьем семестрах. Во втором семестре студенты изучают дисциплину «Архитектура распределенных вычислительных систем», 3 зе. Дисциплина включает 16 часов лекций и 16 часов практических занятий. В дисциплине изложены основы распределенных программных систем, рассматриваются протоколы распределенных программных систем, клиент-серверная концепция, удаленный вызов процедур и методов, а также очереди вычислений. В

курсе также рассматриваются сервис-ориентированная архитектура распределенных вычислительных систем, концепция REST сервисов и графовый API.

В третьем семестре студенты изучают дисциплины:

- Технологии параллельного программирования, 3зе;
- Облачные технологии, 3 зе.

Дисциплина «Технологии параллельного программирования» включает 16 часов и 32 часа лекций практических занятий. В дисциплине рассматриваются виды параллельной обработки данных, классификация параллельных вычислительных систем (классификация Флинна, классификация MIMD-систем), рассматриваются способы оценки производительности многопроцессорных систем, модели программирования для различных архитектур. Изучаются стандарты OpenMP, MPI и CUDA. Рассматриваются вопросы анализа эффективности параллельных алгоритмов. Студенты также знакомятся с программированием для гибридной архитектуры.

Дисциплина «Облачные технологии» включает 16 часов лекций и 16 часов практических занятий. В дисциплине изложены основные понятия в области облачных вычислений, рассматриваются технологии виртуализации и контейнеризации. Изучаются платформы Docker, Kubernetes. Изучаются основные понятия и паттерны микросервисной архитектуры.

В настоящий момент ЮУрГУ располагает мощными суперкомпьютерными ресурсами для обучения студентов высокопроизводительным вычислениям [10]: суперкомпьютер «Торнадо ЮУрГУ», вычислительный кластер «СКИФ Урал» и комплекс «Нейрокомпьютер». Комплекс «Нейрокомпьютер» включает в себя два сервера с графическими ускорителями Nvidia Ampere A100, три сервера с графическими ускорителями Nvidia Ampere A30 и один сервер с графическими ускорителями Nvidia Tesla V100. Таким образом, магистранты получают возможность планирования и проведения экспериментов в области искусственного интеллекта на графических ускорителях Nvidia.

5. Практика и ГИА

Практика магистерской программы «Искусственный интеллект и инженерия данных» по направлению подготовки «09.04.04 Программная инженерия» включает в себя учебную летнюю и производственную распределенную практики. Учебная практика проводится на первом курсе в летний период в формате технологической (проектно-технологической) практики, целью которой является закрепление на практике и углубление теоретических знаний о технологиях искусственного интеллекта и практических навыков по реализации проектов по созданию систем искусственного интеллекта, полученных студентами при изучении дисциплин по профилю магистратуры. Практика может проводиться как на предприятии по направлению ВУЗа, так и в самом ВУЗе.

Производственная практика проводится в формате научно-исследовательской работы на втором курсе в течение четвертого семестра. Целью практики является проведение магистрантом научных исследований под руководством научного руководителя. Задачами практики являются: обзор литературы по теме исследования, создание и обучение модели искусственного интеллекта, а также проектирование и реализация прототипа системы, демонстрирующего эффективность реализованной модели.

Государственная итоговая аттестация образовательной программы включает защиту выпускной квалификационной работы (ВКР), в том числе подготовку к процедуре защиты и процедуру защиты. ВКР магистра представляет собой законченную разработку, связанную с решением актуальной теоретической и (или) прикладной задачи, определяемой особенностями подготовки по данной образовательной программе. Рекомендуемый объем ВКР магистра: 40-50 страниц (без учета приложений), объем библиографии: не менее 20 источников. Тематика ВКР магистра может включать в себя разработку систем компьютерного зрения, анализа или синтеза естественного языка, генерации изображений и другие темы в сфере искусственного интеллекта. Таким образом, список тем ВКР может включать в себя следующие формулировки:

1. Разработка программы распознавания сканированных паспортных данных для идентификации личности;

2. Реализация системы мониторинга и прогнозирования загрязнения атмосферного воздуха на территории Российской Федерации на основе алгоритмов машинного обучения;

3. Распознавание узора меха сайменской нерпы с целью ее идентификации;

4. Обнаружение эпилептических припадков по углубленному изучению ЭЭГ-сигналов методами машинного обучения;

5. Разработка системы управления приложениями на основе распознавания жестов;

6. Разработка и реализация параллельного алгоритма для построения моделей нейронных сетей Ваттса-Строгаца;

7. Поиск похожих подпоследовательностей временного ряда на кластерных вычислительных системах с ускорителями архитектуры Intel MIC;

8. Параллельный алгоритм решения задачи анализа рыночной корзины для многоядерного ускорителя Intel Xeon Phi;

9. Разработка нейросетевой модели для системы машинного зрения фасеточного типа;

10. Разработка программной системы по обнаружению дефектов керамической плитки на производственной линии с использованием искусственных нейронных сетей;

11. Разработка нейронной сети для задач классификации изображений с малой обучающей выборкой;

12. Разработка системы распознавания дефектов сварных швов труб по снимкам, полученным с установки рентгенотелевизионного контроля;

13. Разработка системы автоматической генерации заголовков новостных статей на основе нейросетевых технологий;

14. Применение методов машинного обучения для анализа медицинских показателей у пациентов с псориазом;

15. Разработка системы ранжирования потенциальных клиентов телекоммуникационной компании на основе машинного обучения;

16. Разработка программного фильтра запрещенных слов в аудиофайлах на основе нейросетевых технологий;

17. Разработка программной системы для классификации дефектов металлического листа с покрытием с использованием нейросетевых технологий.

Данный список был составлен на основе выпускных квалификационных работ магистров кафедры системного программирования за 2022 и более ранние годы [11].

6. Заключение

Реализация магистерской программы «Искусственный интеллект и инженерия данных» по направлению подготовки «09.04.04 Программная инженерия» позволит повысить уровень подготовки студентов в области разработки систем искусственного интеллекта. Студенты приобретают знания как в сфере искусственного интеллекта, так и в сфере высокопроизводительных вычислений, что позволит повысить конкурентную способность выпускников данного направления.

Литература

1. Указ Президента РФ от 10 октября 2019 г. № 490 "О развитии искусственного интеллекта в Российской Федерации". URL: http://www.kremlin.ru/acts/bank/44731 (дата обращения 29.04.2023 г.)
2. Программы магистратуры факультета вычислительной математики и кибернетики МГУ. URL: http://edu.msu.ru/faculties/02.shtml (дата обращения 05.06.2023 г.).
3. Программы магистратуры ИТМО. URL: https://abit.itmo.ru/programs/master?directions=information_security%2Cprogramming%2Cartificial_intelligence (дата обращения 05.06.2023 г.)
4. Образовательная магистерская программа «Искусственный интеллект и инженерия данных» по направлению подготовки «09.04.04 Программная инженерия». URL: https://www.susu.ru/ru/grant-art-intelligence/090404 (дата обращения 29.04.2023 г.)

5. Письмо Министерства науки и высшего образования Российской Федерации от 21 декабря 2021 г. № МН-5/22720
6. Антонов А.С., Воеводин Вл.В., Гергель В.П., Соколинский Л.Б. Системный подход к суперкомпьютерному образованию // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2013. Т. 2. No 2. С. 5-17
7. Антонов А.С., Артемьева И.Л., Бухановский А.В., Воеводин В.В., Гергель В.П., Демкин В.П., Коньков К.А., Крукиер Л.А., Попова Н.Н., Соколинский Л.Б., Сухинов А.И. Проект «Суперкомпьютерное образование»: 2012 год // Вестник Нижегородского университета им. Н.И. Лобачевского. 2013. No 1-1. С. 12-16.
8. Соколинский Л.Б. Параллельные системы баз данных. М.: Издательство Московского университета, 2013. 184 с.
9. Долганина Н.Ю. Опыт преподавания суперкомпьютерных технологий на инженерных направлениях подготовки в ФГБОУ ВПО «ЮУрГУ» (НИУ) // Суперкомпьютерные дни в России: Труды международной конференции (28-29 сентября 2015 г., г. Москва). М.: Изд-во МГУ, 2015. С. 776-778.
10. Вычислительные ресурсы НОЦ «Искусственный интеллект и квантовые технологии» ЛСМ ЮУрГУ. URL: http://supercomputer.susu.ru/computers/ (дата обращения 29.04.2023 г.)
11. Выпускные квалификационные работы магистров кафедры системного программирования ЮУрГУ. URL: https://sp.susu.ru/student/masterthesises.html (дата обращения 29.04.2023 г.)

MSk - the package for a dense matrix approximation in the mosaic-skeleton format*

B. Valiakhmetov[1], E. Tyrtyshnikov[1, 2]

[1]Lomonosov Moscow State University,
[2]Marchuk Institute of Numerical Mathematics of the Russian Academy of Sciences

In this paper we present the package for approximation dense matrices in mosaic-skeleton format. It is flexible in terms of matrix appearance, ambient domain decomposition and approximation method. A user can choose default implemented methods or replace any of them with his own code without any compatibility issues. The package is extensible for any other block-independent approximation formats. MSk is written on C++ and supports parallel execution on shared and distributed memory with OpenMP, C++ threads and MPI standard.

Keywords: mosaic-skeleton method, hierarchical matrix, low-rank, matrix cross approximation.

1. Introduction

Mosaic-skeleton approximations arise naturally in many applications. It is the way of approximation dense functional matrices based on some knowledge about the function and its definition domain. Typically, smooth functions of two points that decay with arguments distancing can not be approximated with low-rank matrices but fit mosaic-skeleton format well. Detailed discussion about such functions can be found in the paper [1]. This matrix format is also referenced as hierarchical (H-matrices) and was studied from algebraic point of view [2].

In this paper we present our view on the implementation of this method. It is designed in spirit of object-oriented approach. It helps us to gather different matrix structures into one mosaic format. Another advantage is its flexibility in terms of user specifications (see Sec. 3.2 and Sec. 3.3). Generally, we provide the rules of interaction between data structures. A user can provide his own types that follow the rules, and they will integrate into the approximation scheme. Default data structures for operating with matrices in mosaic-skeleton format are provided.

The way to parallelize the approximation algorithm is presented too. The aim is to balance time spent on matrix elements evaluation and other computations (see Sec. 3.4). It works for user-defined structures too if they follow the parallelism model too.

2. Mosaic-skeleton format

2.1. Notation

Let us have a functional matrix A:

$$A = [a_{ij}] \in \mathbb{C}^{m \times n}, \quad a_{ij} = f(y_i, x_j),$$

on a couple of grids:

$$Y = \left\{ y_i \in \mathbb{R}^d : i \in \overline{1, m} \right\},$$
$$X = \left\{ x_j \in \mathbb{R}^d : j \in \overline{1, n} \right\},$$

*The work was supported by the Russian Science Foundation project № 19-11-00338, https://rscf.ru/en/project/19-11-00338/.

where $\overline{s,t}$ denotes $\{s, s+1, \ldots, t-1, t\}$. We also refer to the matrix in form $A = A(Y, X)$. Further we may recall points from X as *sources* and points from Y as *receivers*. The matrix can be real valued too so all techniques remain the same.

Generally, sets X and Y may have non-empty intersection or even be the same set of points. We define the general set:

$$Z = X \cup Y = \left\{ z_k : k \in \overline{1,N} \right\}.$$

So we can think of X and Y as taking a subset of Z at certain indices:

$$Y = Z(I), \quad X = Z(J),$$

where I and J are ordered sets of the indices such that $I_i = k : z_k = y_i$ and $J_j = k : z_k = x_j$. Following the notation we will refer the matrix in Matlab/Python programming language style:

$$A(I, J) \equiv A(Z(I), Z(J)) = A(Y, X).$$

A subblock of the matrix A can be denoted in form $A(I_k, J_l)$ where $I_k \subset I$ and $J_l \subset J$ are the indices of rows and columns taken, respectively.

2.2. Low-rank decomposition

A process of approximation is always about finding a representation of an object with smaller number of parameters. A matrix A of rank r can be decomposed in the following form:

$$A = UV^* \quad \text{or} \quad A = UBV^*.$$

Here matrix $U \in \mathbb{C}^{m \times r}$ can be treated as the basis of columns of A with some matrices $V \in \mathbb{C}^{n \times r}$ and $B \in \mathbb{C}^{r \times r}$. In some cases it is useful to write such decomposition in the following form:

$$A = UV^T \quad \text{or} \quad A = UBV^T.$$

Here both matrices U and V can be treated as the bases of columns and rows of the matrix, respectively. For example, such format arises in matrix cross-approximation method [3]. If A has low rank: $\operatorname{rank} A = r \ll \min(m, n)$, then we can store not more than $r(m + n + r)$ values. Number of operations required for matrix-vector product is also $O(r(m + n))$.

The same holds for matrices with numerical low rank or the ones with low-rank approximation:

$$\|A - A_r\| \leq \varepsilon \|A\|,$$

where $\operatorname{rank} A_r = r \ll \min(m, n)$. It is usually called ε-rank of a matrix: $\operatorname{rank}_\varepsilon A$. Then we are guaranteed to have

$$\|Ax - A_r x\| \leq \varepsilon \|A\| \|x\|.$$

How to determine if a matrix has low rank? For arbitrary given (stored in memory) matrix it can be extracted from some factorization. The exact way is to apply singular value decomposition (SVD) and reveal the rank from the singular values decay. It immediately provides the factorization itself and we are free to select the rank of approximation from required accuracy or some rank bounding reasons. The main disadvantages of SDV are $O(mn \min(m, n))$ computational cost and $O(mn)$ memory requirements.

In the case when the matrix is given as a black-box function of matrix-vector product there exist some randomized methods [5, 6]. You can evaluate the whole matrix from multiplication by the identity matrix and then take SVD, but it is usually not applicable because of time or memory limitations.

For functional matrices SVD sometimes may be time-consuming due to high element computation cost. Another choice is matrix cross-approximation [3, 4]. It computes low-rank factorization in form

$$A = UC^{-1}V^T.$$

Here matrix $U = A(I, J_l)$ contains columns of A indexed by J_l, matrix $V^T = A(I_k, J)$ contains rows of A indexed by I_k and matrix $C = A(I_k, J_l)$. This method evaluates only $O(r)$ rows and columns of the matrix so total number of evaluated elements is $O(r(m + n))$.

2.3. Low-rank functions

For mosaic-skeleton method to be applicable we assume that the function f of the matrix element evaluation has such property: if some subset of the receivers at indices I_α is *far* from some subset of the sources at indices J_β then the functional submatrix $A(I_\alpha, J_\beta)$ on this subgrid has low enough rank. Saying "low enough rank" we mean that rank $A(I_\alpha, J_\beta)$ is much less than $\min(|I_\alpha|, |J_\beta|)$. So it is reasonable to perform the approximation and retrieve the representation with lower number of parameters.

What sets can be called "far"? It is fully problem specific question. The answer stands on the function properties, mutual location of points and required approximation accuracy. For example, we may have some finite element grid that is split into a number of regions. Then we may define all unmatched regions as "far", or only those which are not neighbors.

3. Implementation details

The main concept of the MSk package is flexibility. It is designed as the main pipeline of mosaic matrix approximation. User can complete it with specifics: what and how actually to approximate. For this reason we follow the object-oriented approach.

Mainly, there are 3 steps of the algorithm:

1. Splitting the matrix into blocks, then defining block structures and corresponding approximation algorithms.
2. Approximation of each block.
3. Post-computations and balancing.

User must define at least the function of the matrix element evaluation. It is also useful to provide the information about geometry of the problem. Otherwise it is treated as linear: j-th row or column corresponds to j-th point of regular grid on a segment. This leads to exactly HSS matrix format [7].

Then at the first step tree-like structures of points and blocks are built, blocks are assigned algorithms. The second step is typically the longest one where the whole approximation is going. The third step is to balance blocks between computational processes for more uniform memory usage and future faster computations in the case of share memory usage.

After the approximation is done user receives the matrix in mosaic format. It provides matrix-vector product and some additional information such as norm, data distribution and others.

3.1. Oracle

First step of the algorithm consists in splitting matrix into blocks which then should be approximated. This work is delivered to the class `Oracle`. It holds the whole information about problem's geometry and properties of the matrix.

Basically, `Oracle` should be able to answer the following question: "What should the algorithm do with the block $B(I, J)$?" Here I and J hold indices of rows and columns of the block B. The `Block` $B(I, J)$ is guaranteed to be previously received from `Oracle`. So it sequentially processes blocks.

There are only two types of the answer: "to split" or "to approximate". With the first answer `Oracle` should return resulting list of blocks $\{B(I_k, J_k)\}$: $\cup_k I_k \times J_k = I \times J$. Generally, this

operation requires permutation of the indices in the ordered sets I and J. Second answer means that current block is to be approximated in some way. The `Oracle` should return an instance of the class `DataBlock` (see Sec. 3.2) and a corresponding instance of the class `Computer` (see Sec. 3.3). They denote the way the block's data will be stored and the method to compute this representation. So `Oracle` builds hierarchical tree-like structure with blocks as nodes and block-subblock inheritance for edges. Leaf nodes correspond to blocks that are "filled" with data.

User can design its own `Oracle` with any logic inside. All this building work can be done in advance. A user can precompute and save the tree, and then use it e.g. for different matrices on the same mesh.

Current implementation provides two heuristic oracles. Both of them operate with points $Z(I)$ and $Z(J)$ of a block $B(I, J)$ at a single step in the following way.

1. Compute centroids: $c_I = \frac{1}{|I|}\sum\limits_{i\in I} z_i$, $c_J = \frac{1}{|J|}\sum\limits_{j\in J} z_j$.
2. Compute radii of encircling spheres: $r_I = \max\limits_{i\in I} \|c_I - z_i\|_p$, $r_J = \max\limits_{j\in J} \|c_J - z_j\|_p$.
3. If $\rho(c_I, c_J) \geq \rho_0(r_I + r_j)$ then $B(I, J)$ is treated as low-rank.
4. Else if $|I|$ and $|J|$ are small enough then $B(I, J)$ is treated as dense.
5. Else *split* indices: $I = I_1 \cup I_2$ and $J = J_1 \bigcup J_2$ and return 4 blocks: $B(I_\alpha, J_\beta)$, $\alpha, \beta \in \{1, 2\}$.

Typically, we take the constant $\rho_0 = 1 + \varepsilon$, so only blocks with separated points are treated as low-rank. The smallness of I and J is controlled by the user defined constant too. If only one of the index sets I or J is small then the block is split into two subblocks only.

One of the oracles is named `SimpleOracle`. It assumes $I_0 = J_0$ and $z_j = j$ (regular mesh on a segment) where I_0 is the whole set of indices: $Z = Z(I_0)$. For every block $B(I, J)$ it splits J into $J_1 = \{z_k \in I : z_k \leq c_I\}$ and $J_2 = \{z_k \in I : z_k > c_I\}$, the same is done for I. This produces the structure that is often described in works about H^1 matrices [2]. If $\rho_0 = 1$ then it is exactly HSS structure [7].

More complicated one is `PointOracle` which requires some mesh $Z \subset \mathbb{R}^d$. For splitting points $Z(I)$ it uses heuristics whose complexity is linear on the number of points. For $p < \infty$ in $\|\cdot\|_p$ it computes some direction in which $Z(I)$ is elongated: $d_I = \sum\limits_{i\in I} z_i - c_I$. Then it splits point with the hyperplane that is orthogonal to d_I and passes through the centroid c_I. For $p = \infty$ it finds the longest projection of $Z(I)$ onto coordinate axis $l \in \overline{1, d}$. Then it splits points as in case $p < \infty$ with $d_I = e_l$ (unit vector along l-th axis). The same is done for $Z(J)$.

This part of the algorithm is currently being developed. Some other heuristics are to be tested in future.

3.2. Block types

There are several types of blocks. The main one is called `Block`. It holds the information about the indices of its rows and columns and references to data. The indexing is provided via the class `Index`, which stores ranges and permutation of the indices.

Block's data is represented by two interfaces. The first one is the class `DataBlock` that provides matrix-vector product and some additional information such as Frobenius norm, complexity of matrix-vector product. Current implementation supports 2 types of the data representation types: `DenseBlock` and `LowRankBlock` which is enough for mosaic-skeleton format. A user can provide any other block for data representation but it should be derived from the class `DataBlock`. For example, sparse, circulant, Toeplitz etc. Such block type should be provided with at least one method of obtaining its representation.

The second interface is described in the next section.

3.3. Computer types

The class `Block` also stores the information about the method, which is used for the computation of A given instance of the `DataBlock`. This logic is provided by the class `Computer`. Its instance should provide 2 ways of approximation: one-time, and step-by-step. One-time approximation is performed at 1 call. It has access to the matrix (as element computation function) and its rows and columns permutations (taken from the global `Index` class instance). Step-by-step approximation is designed to fit non-blocking parallel model (see Sec 3.4). At each call it does some computational work and asks for a list of matrix elements required for next portion of work.

Current implementation provides 3 methods of computation. `Dense` just evaluates all elements of a `DenseBlock` and stores them. Step-by-step variant does it column-by-column. `SVD` evaluates a whole block and then applies singular value decomposition. `Cross` implements matrix cross-approximation method [8, 9]. It perfectly fits step-by-step model because it requires only several rows or columns of the matrix at every iteration.

We note that user can provide any other method of evaluation both for available and new types of blocks.

3.4. Parallelism model

The main parallelism idea of the MSk package is to create tasks and execute them asynchronously.

On the first stage `Oracle` starts processing blocks from the main that correspond to the whole matrix. Then it goes down the tree of the blocks with more than one block at each level. This work can be done in parallel. We use OpenMP threading here. Processing of each block is assigned to an OpenMP task. The information about the block is received from the `Oracle`, and then new tasks are assigned if it is split into new blocks.

This procedure requires synchronization which is significant for designing any `Oracle`. When the block $B(I, J)$ is processed it may be split into several subblocks $B(I_k, J_l)$, where $\cup_k I_k = I$ and $\cup_l J_l = J$. For each k, indices in I_k must go in a row so it may require an update of their order inside global permutation of I. Then blocks $B(I_k, J_l)$ and $B(I_s, J_t)$ such that $I_k \cap I_s \neq \emptyset$ must have the same rows indices order. Usually blocks have $I_s = I_k$ (when they are on the same level of the tree) or $I_k \subset I_s$ (when the second block is on the lower level than the first). So it is easy to compute such split $I_k = \cup_q I_q$ one time under global lock, save it and then use for any block which requires it. Currently available variants of `Oracle` implement exactly this algorithm. This procedure is independently done at each computational process in case of distributed memory parallelism being enabled.

For the actual approximation stage we provide queue-like model of parallelism. After the list of the blocks is defined it is split and distributed by computational processes. Every block performs its approximation independently so it can be done in parallel. We assume that every block does its own approximation step-by-step (see Sec. 3.3). So there are two types of operations: evaluation of the matrix elements and some approximation work. We are trying to balance these steps because in common cases such as integral operators element evaluation takes considerable time [10].

Our solution is to use a pool of C++ threads. “Steps” of the blocks are just assigned to the class `ThreadPool`. At each process there is one queue of elements to be computed which is filled by the blocks. This logic is provided by the thread-safe class `ElementsQueue`. It is controlled by the master-thread which extracts batches of elements and assigns their evaluation to the same `ThreadPool`. This approach helps to keep the balance between the elements evaluation and intermediate computational work.

There is another option to balance elements evaluation in the case of multiprocessing execution. The instances of `ElementsQueue` on different processes can exchange elements if some

of them are overloaded. For this reason we provide class `RequestsQueue` which holds the logic of elements delivery from an evaluating thread to a block. It is based on asynchronous messages technique (MPI standard). Every `Computer`'s request there exists corresponding MPI-Request which can be accomplished by any `ElementsQueue` on some process.

After the approximation is finished blocks may be optionally rebalanced between computational processes. It is performed in a greedy way. We iteratively pick one block from the most loaded process and redirect it to the least loaded one. After fixed number of iterations or if processes are balanced enough blocks are send to assigned places.

3.5. Source code

The source code is written on C++ language. For linear algebra operations we use C++ template interfaces for BLAS and LAPACK libraries. It helps to easily switch between different types of matrix elements.

For distributed memory parallelism we rely on MPI 2.1 standard. For shared memory parallelism we use OpenMP library and C++ `std::thread`. This choice is made because of some performance issues with switching OpenMP tasks.

The package may seem complicated but its simplest usage is following:

1. Instantiate `Matrix` (user-defined) which provides only element evaluation.
2. Instantiate `Oracle` (predefined or user-defined).
3. Define additional parameters such as number of threads, logger, etc.
4. Instantiate `MSk<Matrix, Oracle>`, which starts the approximation procedure.

4. Numerical experiments

For numerical experiments we have chosen Helmholtz kernel function $H(y, x) = \dfrac{e^{ikr}}{4\pi r}$, $r = \|y - x\|$ on regular grids. Table 1 contains the results for the unit segment $[0, 1]$ and Table 2 is for the unit square $[0, 1]^2$. In each table N is the matrix size, *compression* is the percentage of stored values relative to the dense matrix (N^2) and *imbalance* is the maximal relative difference between the number of values stored on one process and the mean number. Minimal block size was chosen as 128, guaranteed relative approximation accuracy is 10^{-4} in Frobenius norm. These tests were taken on 4 nodes of the INM RAS cluster, with 1 process per node and 40 threads per process. For the detailed platform characteristics visit `cluster2.inm.ras.ru`.

Table 1. Compression on the segment

N	Compression	Imbalance
1000	35.88%	7.875%
2000	19.24%	2.111%
4000	10.1%	1.006%
6000	5.446%	0.985%

These tests show that the algorithm is applicable for dense matrix compression. The blocks data distribution imbalance between processes decays with the matrix size growth. We consider

Table 2. Compression on the square

N	Compression	Imbalance
1024	35.91%	3.175%
2025	17.8%	2.519%
4096	12.82%	0.1428%
6084	7.816%	0.1259%

preparation of additional tests for complicated problems as an important direction of future research.

Another batch of tests is related to the algorithm scalability. The matrix element function is still $H(y, x)$. The grid is on the same unit square with 300 points along each side so $N = 90000$. Table 3 contains the algorithm scalability results for the fixed number of threads $T = 8$ (for both OpenMP and C++ threads parallel regions) and number of MPI processes $P \in \{1, 2, 3, 4, 5\}$. All tests were taken on a single node. The last column just shows speedup factor with respect to $P = 1$ process.

Table 3. Speedup on processes, $T = 8$

P	Time (sec)	Speedup
1	40.35	1.0
2	15.18	2.7
3	12.42	3.2
4	11.02	3.7
5	9.09	4.4

Table 4 contains the algorithm scalability results for the fixed number of threads $T = 40$ and number of cluster nodes $D \in \{1, 2, 3, 4\}$ with $P = 1$ MPI process per node. The last column shows speedup factor with respect to $D = 1$ node.

Table 4. Speedup on nodes, $P = 1$, $T = 40$

D	Time (sec)	Speedup
1	41.06	1.0
2	21.18	1.93
3	14.68	2.80
4	12.01	3.42

Numerical experiments show that presented algorithm scales almost linearly for both shared and distributed memory usage cases. We are aiming to inspect the parallelism model for more complicated problems in future.

5. Conclusion

Presented algorithm of a dense matrix approximation is based on an object-oriented model. It provides user with the flexible interface to the mosaic matrix format. All its sections are designed as parts of the construction kit. A user can build his own approximation algorithm of a matrix with independently computed blocks. Current implementation fully supports mosaic-skeleton format and can be tuned with parameters. It can be executed on shared and distributed memory computers.

Our model of parallelism is chosen to balance the element evaluation function calls and intermediate computations. It stands on top of the algorithm as an interface, so user defined structures and methods are paralleled automatically too. Numerical experiments show the scalability potential for large scale problems.

The package is currently being developed so its heuristics and some synchronization techniques need to be improved for the better performance.

References

1. Tyrtyshnikov E. Mosaic-skeleton approximations // Calcolo. 1996. Vol. 33, P. 47–57.

2. Börm S., Grasedyck L., Hackbusch W. Hierarchical matrices // Lecture notes. 2003. Vol. 21.

3. Goreinov S.A., Zamarashkin N.L., Tyrtyshnikov E.E. Pseudo-skeleton approximations of matrices. In Doklady Akademii Nauk // Russian Academy of Sciences. 1995. Vol. 343, No. 2.

4. Bebendorf M. Adaptive cross approximation of multivariate functions // Constructive approximation. 2011. Vol. 34. P. 149–179.

5. Boullé N., Townsend A. A generalization of the randomized singular value decomposition // 2021. arXiv preprint arXiv:2105.13052.

6. Halko N., Martinsson P.G., Tropp J.A. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions // SIAM review. 2011. Vol. 53. No. 2. P. 217–288.

7. Xia J. et al. Fast algorithms for hierarchically semiseparable matrices // Numerical Linear Algebra with Applications. 2010. Vol. 17, No. 6, P. 953–976.

8. Tyrtyshnikov E. Incomplete cross approximation in the mosaic-skeleton method // Computing. 2000. Vol. 64, P. 367–380.

9. Zheltkov D.A., Tyrtyshnikov E.E. A parallel implementation of the matrix cross approximation method // Numerical Methods and Programming (Vychislitel'nye Metody i Programmirovanie). 2015. Vol. 16. P. 369–375.

10. Setukha A. V., Stavtsev S. L., Tret'yakova R. M. Application of mosaic-skeleton approximations of matrices in the physical optics method for electromagnetic scattering problems // Computational Mathematics and Mathematical Physics. 2022. Vol. 62, No. 9, P. 1424–1437.

Анализ масштабируемости параллельной реализации сеточно-характеристического метода для решения задач распространения упругих волн*

Д.Р. Саидбаталов[1,2], Р.К. Газизов[2,3]

[1]Институт математики им. С.Л. Соболева СО РАН
[2]ООО «РН-БашНИПИнефть»
[3]Уфимский университет науки и технологий

В работе рассматривается параллельная реализация сеточно-характеристического метода для численного моделирования сейсмических волн в трехмерных неоднородных изотропных упругих средах. Приемники сейсмических волн располагаются на верхней грани рассматриваемой области, которая свободна от внешних нагрузок, а на остальных гранях используются поглощающие граничные условия в формулировке идеально согласованных слоев (PML). Предложен параллельный алгоритм, реализованный с использованием технологии MPI для систем с распределенной памятью, технологии OpenMP для систем с общей памятью и с совместным использованием технологий MPI и OpenMP. Исследуется сильная и слабая масштабируемость реализованных алгоритмов.

Ключевые слова: сейсмические волны, многопроцессорные вычисления, декомпозиция области, MPI, OpenMP, сильная и слабая масштабируемость.

1. Введение

Численное моделирование распространения сейсмических волн представляет существенную часть работ при проведении геологоразведки в нефтяной отрасли. Математическое моделирование проводится в различных геологических средах, в том числе, в слоистых средах и в средах с наличием различных неоднородностей (например, трещин или каверн). Появление высокопроизводительных вычислительных систем с параллельной архитектурой открыло новые возможности в моделировании и изучении сейсмических волновых полей в геологических средах путем проведения численных экспериментов. Расчет волновых полей необходим при решении обратных задач сейсморазведки, в частности, в методе обращения полных волновых полей - метод, позволяющий восстановить скоростную модель сложноустроенной геологической среды. В рамках данного метода необходимо производить крупномасштабные вычисления с высокой точностью и скоростью расчетов, что является актуальной задачей в разработке численных методов и алгоритмов моделирования волновых процессов в сложноустроенных геологических средах.

В работе рассматривается сеточно-характеристический метод [11] (с.-х. метод) решения задачи распространения упругих волновых возмущений. Семейство с.-х. методов учитывает свойства гиперболических систем уравнений, которые переписываются в терминах инвариантов Римана в виде линейно независимых уравнений переноса. Для решения данных уравнений в работе используются нелинейные разностные схемы (к таким относятся гибридные схемы, то есть схемы с переменным порядком аппроксимации), описанные в [1]. Данные схемы разрабатывались для преодоления ограничений, следующих из теоремы Годунова (известная теорема о невозможности построения линейных монотонных схем с порядком аппроксимации выше первого). Одним из преимуществ данных схем является возможность расчета методом сквозного счета, т.е. без явного выделения разрывов. Хотя

* Д.Р. Саидбаталовым выполнено построение программного обеспечения, ориентированного на использование высокопроизводительных вычислительных систем с параллельной архитектурой, и проведена серия численных экспериментов. Его исследования проводились за счет гранта Российского научного фонда № 22-11-00104, https://rscf.ru/project/22-11-00104.

линейные уравнения теории упругости таких решений не допускают, при решении задачи распространения волновых возмущений от точечного источника могут возникнуть решения с большими градиентами и близкие к разрывным. В отличие от классических конечно-разностных схем, гибридные конечно-разностные схемы позволяют получить неосциллирующее решение. В работе [2] проведено сравнение сеточно-характеристического метода с разрывным методом Галеркина, которое продемонстрировало более высокую скорость расчета с использованием с.-х. метода, а также показало применимость метода к практическим задачам.

В качестве неотражающих граничных условий используется технология Perfectly Matched Layers (PML) – поглощающий слой, применяемый для усечения вычислительной области в задачах с открытыми границами, в котором попадающие в него волны из вычислительной области не отражаются на границе раздела. Данное свойство позволяет PML в значительной степени поглощать волны с малым коэффициентов отражения (порядка 1-0,5%).

За счет использования явных конечно-разностных схем и расчета задачи на регулярных сетках можно эффективно реализовать параллельный численный алгоритм решения задачи распространения упругих волновых возмущений в неоднородной изотропной среде с использованием технологий MPI и OpenMP для распараллеливания в системах с общей и распределенной памятью. В частности, в работе [3] показаны результаты распараллеливания сеточно-характеристического метода с расчетом по схеме Русанова [4] с использованием технологий MPI, OpenMP, POSIX Threads и CUDA. Авторами продемонстрирована эффективность распараллеливания в системах с общей памятью до 80% с использованием OpenMP и POSIX Threads. Для систем с распределенной памятью с использованием технологии MPI было достигнута эффективность распараллеливания до 70% при слабой масштабируемости до 16 тысяч вычислительных ядер.

В данной работе разработан параллельный алгоритм решения задачи распространения упругих волновых возмущений в неоднородной изотропной среде в трехмерной постановке с использованием PML-области. Использование PML-области позволяет сократить область вычисления, что наряду с параллельными алгоритмами приводит к высокоэффективной системе решения задачи моделирования сейсмических волн.

Приведены результаты численных расчетов для модели слоистой по вертикали среды. Выполнен анализ масштабируемости предоженных реализаций разработанного алгоритма: приведены графики слабой и сильной масштабируемости для MPI и OpenMP версий и показана эффективность распараллеливания гибридного варианта (MPI+OpenMP). Расчеты проведены для разной толщины PML-области. Показана слабая зависимость эффективности распараллеливания от толщины PML-слоя.

2. Математическая модель

В работе рассматривается модель сплошной идеально упругой среды, определяемая соотношениями линейной динамической теории упругости. Для трехмерной изотропной среды данные соотношения в терминах скоростей и напряжений задаются в виде гиперболической системы из 9-ти уравнений. В ортонормированной системе координат (x_1, x_2, x_3) система уравнений записывается как:

$$\rho \frac{\partial u_i}{\partial t} = \sum_{j=1}^{3} \frac{\partial \sigma_{ij}}{\partial x_j} + f_i, \ \mathrm{i} = 1,2,3,$$
$$\frac{\partial \sigma_{ij}}{\partial t} = \lambda \sum_{m=1}^{3} \frac{\partial u_m}{\partial x_m} \delta_{ij} + \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) + g_{ij}, \ \mathrm{i,j} = 1,2,3, \mathrm{j} \geq \mathrm{i}. \quad (1)$$

Здесь ρ – плотность, u_i – компоненты вектора скорости смещения, σ_{ij} – компоненты тензора напряжений Коши. λ и μ – упругие константы (постоянные) Ламе, характеризующие упругие свойства изотропной среды, f_i и g_{ij} – источники внешних сил.

Константу μ называют модулем сдвига, от которой зависит величина сдвига при заданном касательном напряжении. Другая константа Ламе, λ, простого физического истолкования не имеет. Данные параметры выражаются через скорость продольных P-волн и скорость поперечных S-волн, которые соответственно равны $c_p = \sqrt{\frac{\lambda+2\mu}{\rho}}$ и $c_s = \sqrt{\frac{\mu}{\rho}}$.

3. Сеточно-характеристический метод

Запишем систему уравнений (1) в матричной форме:

$$\frac{\partial W}{\partial t} - \sum_{i=1}^{3} A_i \frac{\partial W}{\partial x_i} = F, \tag{2}$$

где $W = \left(u_x, u_y, u_z, \sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \sigma_{xy}, \sigma_{yz}, \sigma_{xz}\right)^T$ – вектор искомых функций, $F = \left(\frac{f_x}{\rho}, \frac{f_y}{\rho}, \frac{f_z}{\rho}, g_{xx}, g_{yy}, g_{zz}, g_{xy}, g_{yz}, g_{xz}\right)^T$ – вектор функция, определяющая источник упругих колебаний, A_i – квадратные матрицы порядка $n = 9$.

Для построения численного решения применим метод расщепления [5] (или метод дробных шагов) по пространственным переменным, получим четыре системы уравнений:

$$\begin{gathered}\frac{\partial W}{\partial t} - A_1 \frac{\partial W}{\partial x} = 0,\\ \frac{\partial W}{\partial t} - A_2 \frac{\partial W}{\partial y} = 0,\\ \frac{\partial W}{\partial t} - A_3 \frac{\partial W}{\partial z} = 0,\\ \frac{\partial W}{\partial t} = F.\end{gathered} \tag{3}$$

В дальнейшем пересчет с одного временного слоя на следующий по формулам (3) будем называть соответственно шагом по *X*, по *Y* и по *Z*. Данное расщепление имеет первый порядок аппроксимации по времени. В работе [6] описано так называемое расщепление Странга, обладающее вторым порядком аппроксимации, определяющее последовательность вычисления шагов по направлениям *x*, *y* и *z*, например, циклически в следующем порядке: *xyz*, *yxz*, *zxy*, *xzy*, *yzx*, *zyx*. В отличие от классического метода расщепления, в расщеплении Странга второй порядок точности является суммарным.

Поскольку каждая из систем уравнений в (3) является гиперболической, то матрицы A_i имеют полный набор действительных собственных значений и каждому такому собственному значению соответствует собственный вектор.

Для изотропной среды матрицы A_i имеют одинаковый набор собственных значений. После замены переменных

$$S_i = \Omega_i^{-1} W, \tag{4}$$

где $\Omega_i^{-1} A_i \Omega_i = \Lambda = \mathrm{diag}(-c_p, -c_s, -c_s, 0, 0, 0, c_s, c_s, c_p)$, Ω_i – матрица, составленная из собственных векторов матрицы A_i, перейдем к системе

$$\begin{gathered}\frac{\partial S_1}{\partial t} - \Lambda \frac{\partial S_1}{\partial x} = 0,\\ \frac{\partial S_2}{\partial t} - \Lambda \frac{\partial S_2}{\partial y} = 0,\\ \frac{\partial S_3}{\partial t} - \Lambda \frac{\partial S_3}{\partial z} = 0,\end{gathered} \tag{5}$$

записанной в инвариантах Римана.

Так как матрица Λ является диагональной, то системы уравнений (5) распадаются на независимые уравнения переноса.

Для численного решения скалярного линейного уравнения переноса

$$\frac{\partial u}{\partial t} + \lambda \frac{\partial u}{\partial x}, \lambda = \text{const} > 0, \tag{6}$$

авторами работ [1, 7] были предложены нелинейные (гибридные) конечно-разностные схемы 2-3-го порядка аппроксимации:

$$u_m^{n+1} = u_m + \frac{\sigma}{2}(\Delta_0 + \Delta_1) + \frac{\sigma^2}{2}(\Delta_0 - \Delta_1) + \alpha^0(\Delta_{-1} - 2\Delta_0 + \Delta_1), \tag{7}$$

где $\Delta_1 = u_m^n - u_{m+1}^n$, $\Delta_0 = u_{m-1}^n - u_m^n$, $\Delta_{-1} = u_{m-2}^n - u_{m-1}^n$.

Параметр α^0 выбирается из следующего условия монотонности:

$$0 \le w = \frac{\sigma(1+\delta_1)}{2} + \frac{\sigma^2(1-\delta_1)}{2} + \alpha^0(\delta_{-1} - 2 + \delta_1) \le 1, \tag{8}$$

где $\delta_{-1} = \Delta_{-1}/\Delta_0$, $\delta_1 = \Delta_1/\Delta_0$, $\Delta_0 \neq 0$,$\sigma = \frac{\lambda\tau}{h} < 1$ – число Куранта.

Значение параметра $\alpha^0 = \frac{\sigma(\sigma^2-1)}{6}$ соответствует схеме третьего порядка аппроксимации – схеме Русанова [4]. В случае невыполнения условия монотонности порядок схемы снижается до второго и выбирается одна из следующих схем:

$$\begin{aligned} \alpha^0 &= 0 \text{ – схема Лакса-Вендроффа,} \\ \alpha^0 &= \tfrac{\sigma(\sigma-1)}{2} \text{ – схема Бима-Уорминга.} \end{aligned} \tag{9}$$

В случае $\lambda < 0$ – $\sigma = \frac{|\lambda|\tau}{h} < 1$, $\Delta_1 = u_m^n - u_{m-1}^n$, $\Delta_0 = u_{m+1}^n - u_m^n$, $\Delta_{-1} = u_{m+2}^n - u_{m+1}^n$.

4. Граничные и контактные условия

Рассматриваемая система уравнений решается в области, представляющей собой прямоугольный параллелепипед.

На дневной поверхности заданы граничные условия свободной поверхности, т.е. верхняя грань z=0 предполагается свободной от внешних нагрузок:

$$\sigma_{xz}|_{z=0} = \sigma_{yz}\big|_{z=0} = \sigma_{zz}|_{z=0} = 0, \tag{10}$$

На остальных гранях рассматриваемой области заданы поглощающие граничные условия в формулировке идеально согласованных слоев (PML - Perfectly Matched Layers). Уравнения, моделирующие распространения волн внутри PML и обеспечивающие затухание волновых возмущений принимают следующий вид:

$$\begin{aligned} \frac{\partial S_1}{\partial t} + d(x) - \Lambda\frac{\partial S_1}{\partial x} = 0, \\ \frac{\partial S_2}{\partial t} + d(y) - \Lambda\frac{\partial S_2}{\partial y} = 0, \\ \frac{\partial S_3}{\partial t} + d(z) - \Lambda\frac{\partial S_3}{\partial z} = 0, \end{aligned} \tag{11}$$

где S_i - инварианты Римана, полученные по формуле (4), $d(\mathrm{s})$ - демпфирующая функция.

Согласно [8], демпфирующая функция $\mathrm{d(s)}$ выбирается следующим образом:

$$d(\mathrm{s}) = \frac{2\mathrm{c_p}}{\mathrm{L}}\log\left(\frac{1}{\mathrm{R}}\right)\left(\frac{\mathrm{s}}{\mathrm{L}}\right)^4, \tag{10}$$

где $\mathrm{s} \in [0, \mathrm{L}]$, L – ширина поглощающего слоя, R – константа, характеризующая коэффициент поглощения.

Для замыкания конечно-разностных схем на границе PML области взято граничное условие с использованием мнимых точек, рассмотренное в [9].

В трехмерном случае корректировка проводится по следующим формулам:
по направлению x:

$$\begin{gathered}\overrightarrow{v_0} = \overrightarrow{v_1},\\ \overrightarrow{v_{-1}} = \overrightarrow{v_2},\\ \sigma_0 = \begin{pmatrix} -\sigma_1^{11} & -\sigma_1^{12} & -\sigma_1^{13} \\ -\sigma_1^{12} & 0 & 0 \\ -\sigma_1^{13} & 0 & 0 \end{pmatrix},\\ \sigma_{-1} = \begin{pmatrix} -\sigma_2^{11} & -\sigma_2^{12} & -\sigma_2^{13} \\ -\sigma_2^{12} & 0 & 0 \\ -\sigma_2^{13} & 0 & 0 \end{pmatrix},\end{gathered} \tag{11}$$

по направлению y:

$$\begin{gathered}\overrightarrow{v_0} = \overrightarrow{v_1},\\ \overrightarrow{v_{-1}} = \overrightarrow{v_2},\\ \sigma_0 = \begin{pmatrix} 0 & -\sigma_1^{12} & 0 \\ -\sigma_1^{12} & -\sigma_1^{22} & -\sigma_1^{23} \\ 0 & -\sigma_1^{23} & 0 \end{pmatrix},\\ \sigma_{-1} = \begin{pmatrix} 0 & -\sigma_2^{12} & 0 \\ -\sigma_2^{12} & -\sigma_2^{22} & -\sigma_2^{23} \\ 0 & -\sigma_2^{23} & 0 \end{pmatrix},\end{gathered} \tag{12}$$

по направлению z:

$$\begin{gathered}\overrightarrow{v_0} = \overrightarrow{v_1},\\ \overrightarrow{v_{-1}} = \overrightarrow{v_2},\\ \sigma_0 = \begin{pmatrix} 0 & 0 & -\sigma_1^{13} \\ 0 & 0 & -\sigma_1^{23} \\ -\sigma_1^{13} & -\sigma_1^{23} & -\sigma_1^{33} \end{pmatrix},\\ \sigma_{-1} = \begin{pmatrix} 0 & 0 & -\sigma_2^{13} \\ 0 & 0 & -\sigma_2^{23} \\ -\sigma_2^{13} & -\sigma_2^{23} & -\sigma_2^{33} \end{pmatrix}.\end{gathered} \tag{13}$$

В формулах (11) - (13) нулевым нижним индексом обозначается первая мнимая точка, нижним индексом -1 обозначается вторая мнимая точка, а нижними индексами 1 и 2 обозначены в направлении от границы вычислительной области внутренние точки. Верхними индексами обозначены компоненты симметричного тензора напряжений.

Для корректировки значений численного решения на границе двух геологических сред используется контактный корректор полного слипания [9].

В дальнейшем рассматривается модель слоистой среды. Модель имеет 3 слоя, каждый из которых обозначен отдельным цветом на рисунке 1. Параметры среды приведены в таблице 1. В качестве точечного источника был выбран импульс Рикера, действующий на главные компоненты тензора напряжений в (1):

$$g_{xx} = g_{yy} = g_{zz} = F(t) = A2\pi v_0\sqrt{e}(t - t_0)e^{-2(\pi v_0(t-t_0))^2}, \tag{14}$$

где v_0 – частота импульса, равная 30 Гц.

Таблица 1. Параметры геологических слоев

Слой	v_p	v_s	ρ	Число узлов по оси Z
1	6,4	3,64	5,4	100
2	11,8	6,84	5,7	100
3	13,9	8,06	5,62	100

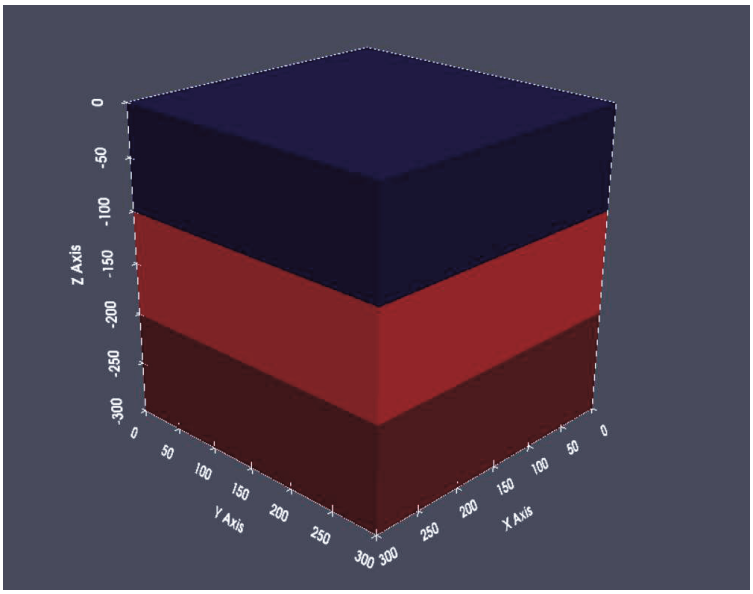


Рис. 1. Структура слоистой среды

5. Распараллеливание с использованием MPI

Поскольку сеточно-характеристический метод основан на явных конечно-разностных схемах и реализован на регулярной сетке, для реализации параллельного алгоритма выбрана геометрическая декомпозиция расчетной области. Такой подход позволяет добиться равномерного распределения узлов расчетной сетки по вычислительным узлам системы.

В работе реализован параллельный алгоритм с использованием технологии MPI (Message Passing Interface) [10]. Данная технология является наиболее подходящей для расчетов на системах с распределенной памятью и обеспечивает возможность запуска программы на большинстве доступных многопроцессорных систем.

С помощью вызова функции MPI_Cart_create создается двумерная декартова топология процессов, а с помощью функции MPI_Dims_create рассчитывается число процессов в каждом измерении и определяется разбиение расчетной области на равные по размеру блоки. Количество блоков соответствует количеству MPI-процессов. С помощью функции MPI_Cart_shift определяются соседние процессы, с которыми осуществляется обмен приграничных узлов. Для упрощения пересылок данных между процессами используется пользовательский тип данных MPI: MPI_Datatype subarray, создаваемый вызовом функции MPI_Type_create_subarray, определяющий некоторый n-мерный подмассив, где под n подразумевается размерность решаемой задачи. Использование такого типа эффективно для пересылок данных, которые располагаются не последовательно в памяти. При вызове функции MPI_Sendrecv передается массив, содержащий в себе всю расчетную подобласть процесса (а именно массив, элементами которого являются структуры, хранящие значения найденного численного решения в узлах расчетной сетки), а также конкретный тип MPI_Datatype subarray, описывающий область для отправки / чтения данных. Такой подход позволяет унифицировать обмен между процессами, вне зависимости от того, как располагаются данные в памяти.

Так же MPI предоставляет возможность распараллеливания ввода/вывода. В частности, в программе были реализованы параллельное чтение параметров среды из заданных файлов, и параллельная запись расчетов в файлы при помощи функций MPI_File_read_all и MPI_File_write_all соответственно, также при помощи пользовательского типа данных - подмассива. Для каждого процесса определяется трехмерный подмассив, описывающий область в файле, в которую нужно записать полученные расчеты / считать параметры среды.

Параметры среды, записанные в файл, описывают лишь исходную трехмерную расчетную область, т.е. параметры среды для PML-области отсутствуют. При чтении данных из файла,

процессы, чья расчетная область содержит PML, копируют значения из находящихся на границе исходной расчетной области точек в PML-область.

Для построения графика эффективности и слабой масштабируемости предложенной параллельной реализации были проведены тестовые расчеты на сетке размером 300^3 узлов. Толщина PML-области бралась равной 10-ти узлам, т.е. итоговый размер расчетной сетки составил 320^3 узлов. Под эффективностью в данной работе понимается отношение ускорения к количеству задействованных процессов.

Расчеты проводились на кластере ССКЦ НКС-1П на вычислительных узлах, оснащенные двумя 16-ти ядерными процессорами Intel Xeon E5-2697v4 2.6GHz с 32 потоками в режиме гиперпоточности. На кластере используется коммуникационная среда (интерконнект) Intel Omni-Path с пропускной способностью 100 Гбит/с. Пиковая производительность составляет 182 ТФлопс. При запуске параллельной версии алгоритма до 64 MPI-процессов использовался один вычислительный узел, а при расчетах на 128 MPI-процессах было задействовано 2 вычислительных узла, на которых было запущено по 64 процессов MPI. На рисунке 2 приведены результаты эффективности и сильной масштабируемости.

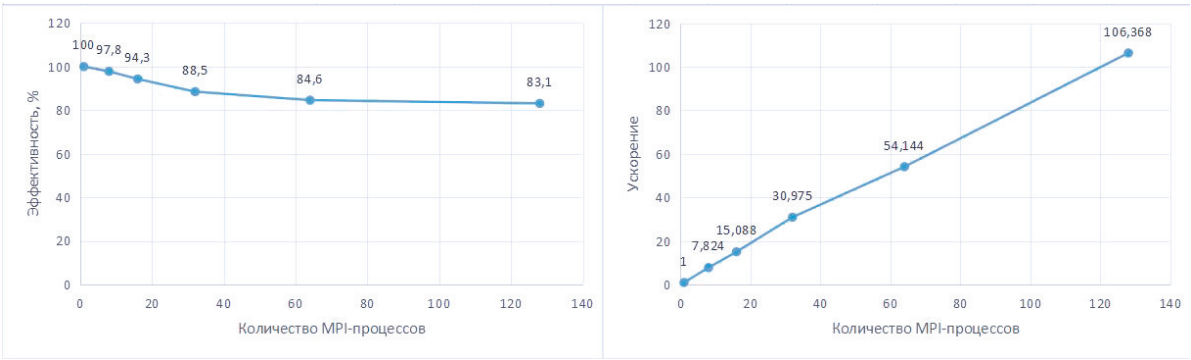


Рис. 2. Эффективность (слева) и сильная масштабируемость (справа) параллельной реализации в среде MPI

Можно заметить, что представленная реализация распараллеливается до 128 MPI-процессов с эффективностью 83% относительно времени работы последовательной версии алгоритма.

На рисунке 3 приведены результаты эффективности и слабой масштабируемости при расчетах на сетках разной размерности. Примерное количество расчетных узлов на один MPI-процесс было взято равным 216 тысяч. Расчеты проводились на сетках из 100^3, 170^3 и 280^3 узлов без учета PML области для 8, 32 и 128 MPI-процессов соответственно. В расчетах толщина PML области бралась равной 10-ти узлам, т.е. общее количество узлов для трех расчетных сеток было равным 120^3, 190^3 и 300^3 узлов соответственно.

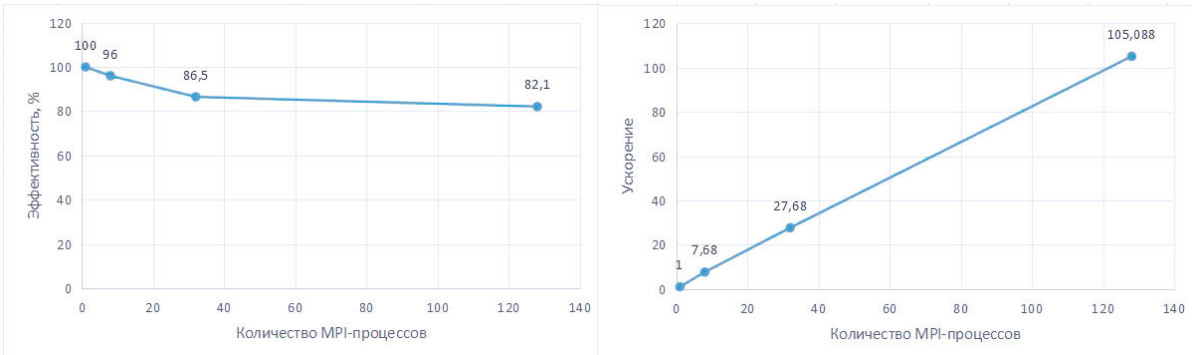


Рис. 3. Эффективность (слева) и слабая масштабируемость (справа) параллельной реализации в среде MPI

Как можно заметить, полученные графики сильной и слабой масштабируемости практически совпадают. Однако, ускорение слабой масштабируемости уступает сильной на небольшое значение в силу выбранного количества расчетных узлов сетки на один MPI-

процесс. При расчете слабой масштабируемости размеры расчетной области подбирались с учетом того, что количество узлов расчетной сетки на один MPI-процесс составляет около 216 тысяч, в то время как для расчета сильной масштабируемости количество узлов расчетной сетки на один MPI-процесс составило 256 тысяч при запуске параллельной реализации на 128 MPI-процессах. При запуске программы на 32 MPI-процессах в случае сильной масштабируемости количество узлов расчетной сетки на один MPI-процесс составило 1024 тысяч. Аналогично можно показать и для расчетов на 8 MPI-процессах. Таким образом, относительно небольшое увеличение ускорения параллельной реализации при проведении расчетов для построения графика сильной масштабируемости объясняется большим количеством узлов расчетной сетки на один MPI-процесс.

Рассмотрим результаты профилирования реализованного параллельного алгоритма с помощью двух инструментов - Intel Trace Analyzer and Collector (ITAC). Intel Trace Collector — это инструмент командной строки для отслеживания и анализа производительности приложений MPI. Он перехватывает все вызовы MPI и генерирует файлы трассировки с расширением «.stf». Intel Trace Analyzer — это графический инструмент, который отображает и анализирует данные трассировки событий, используя файлы трассировки с расширением «.stf», созданные инструментом Intel Trace Collector в качестве входных данных. Intel Trace Analyzer помогает понять поведение приложения, обнаружить проблемы с производительностью и программные ошибки. На рисунке 4 приведен пример работы программы Intel Trace Analyzer для расчетов на сетке, состоящей из 190^3 узлов.

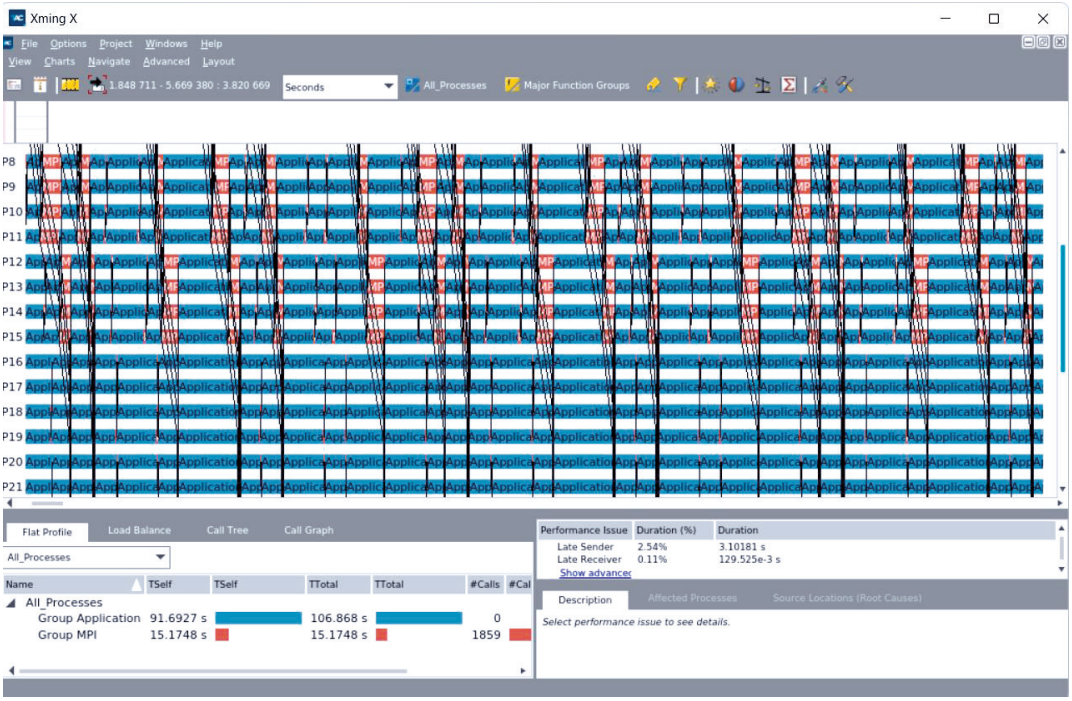


Рис. 4. Пример работы программы Intel Trace Analyzer для расчета на 32 MPI-процессах на сетке, состоящей из 190^3 узлов.

Поясним некоторую отображаемую информацию на рисунке 4. На рисунке изображена карта трассировки, показывающая активность функций MPI с течением времени работы параллельной программы. Активность функций MPI отображается красным цветом. В левом нижнем углу синим цветом отображено время работы параллельной программы, затрачиваемое на вычисление, а красным цветом обозначено время, затрачиваемое на работу функций MPI.

У части процессов наблюдается увеличенное время MPI-пересылок. Можно предположить, что снижение эффективности распараллеливания связано с тем, что при декомпозиции расчетной области количество узлов по каждому направлению для каждого MPI-процесса не одинаковое (т.е. в случае нецелочисленного деления при расчете количества узлов на каждый процесс по двум направлениям).

6. Распараллеливание с использованием OpenMP

Для систем с общей памятью в работе реализован параллельный алгоритм сеточно-характеристического метода с использованием технологии OpenMP. С помощью данной технологии было выполнено распараллеливание циклов при решении уравнений (3), (5), (11) при помощи директив компилятора «#pragma parallel for». Для более равномерного распределения нагрузки также использовалась директива collapse, появившаяся в стандарте OpenMP 3.0. Данная директива позволяет выполнять распараллеливание не только внешних циклов, но и вложенных, что позволяет повысить эффективность распараллеливания за счет «сворачивания» циклов.

Тестовые расчеты проводились на расчетных сетках, описанные в пункте 5, за исключением расчета, проводимого на 128 MPI-процессов, вместо которого проводится расчет на 64 OpenMP-потоках с расчетной сеткой размером 220 узлов без учета размера PML-области. Толщина PML-области бралась равной 10-ти узлам, т.е. итоговый размер расчетной сетки составил 240^3 узлов. Размер такой сетки объясняется тем, что количество узлов на один OpenMP-поток, как и в пункте 5, было взято примерно равным 216 тысячам. На рисунках 5 и 6 представлены эффективность, слабая и сильная масштабируемости реализованного параллельного алгоритма.

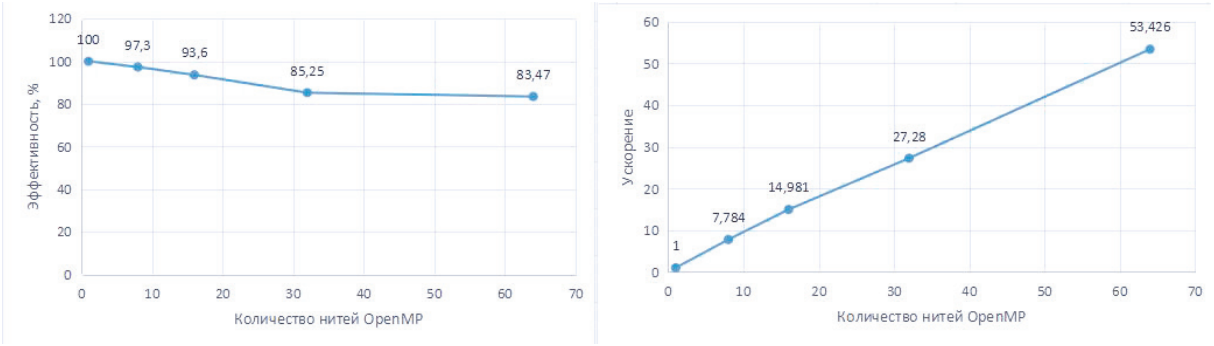


Рис. 5. Эффективность (слева) и сильная масштабируемость (справа) параллельной реализации для систем с общей памятью

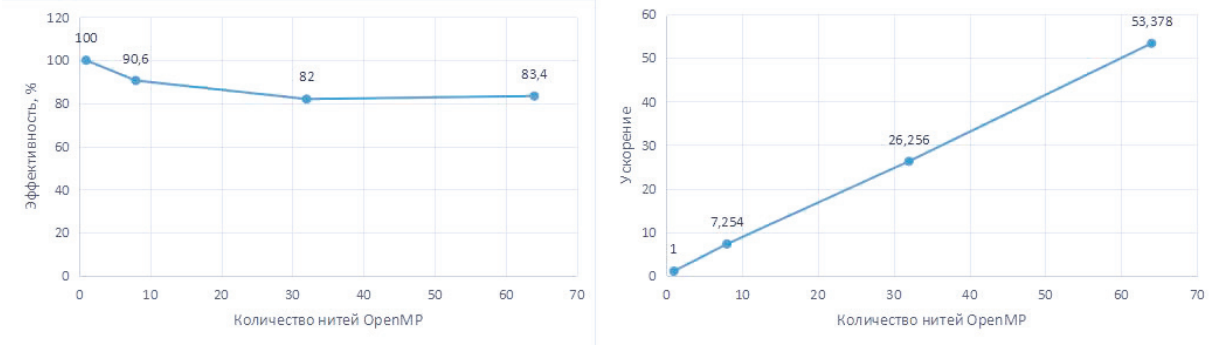


Рис. 6. Эффективность (слева) и слабая масштабируемость (справа) параллельной реализации для систем с общей памятью

Можно заметить, что полученные результаты практически совпадают с графиками на рисунках 2 и 3. Для реализованной OpenMP версии параллельного алгоритма получена эффективность 83% при расчетах на 64 нитях OpenMP.

7. Распараллеливание с применением MPI и OpenMP подходов

При совместном использовании технологий MPI и OpenMP реализован гибридный параллельный алгоритм сеточно-характеристического метода. Для распараллеливания внутри вычислительного узла, т.е. в системе с общей памятью, используется технология OpenMP, а для возможности расчетов на нескольких вычислительных узлах используется MPI. На каждом вычислительном узле запускается один MPI-процесс, порождающий OpenMP-потоки.

На рисунке 7 приведены эффективность и сильная масштабируемость реализованного параллельного алгоритма. При построении графиков тестовые расчеты проводились на сетке из 200^3 узлов без учета PML-области (220^3 узлов с учетом PML-области), и использовалась модель среды, описанная в пункте 5. На рисунке 7 приведены результаты относительно времени вычислений на одном вычислительном узле, при этом каждый MPI-процесс порождал 64 нити OpenMP. Расчеты проводились при распараллеливании до 6 вычислительных узлов.

На рисунке 8 приведены эффективность и слабая масштабируемость реализованного параллельного алгоритма. При построении графиков тестовые расчеты проводились на сетках такой размерности, при которой на один вычислительный приходится примерно 10 920 тыс. узлов (столько узлов содержит расчетная сетка, на которой производились расчеты для построения графика сильной масштабируемости). Расчеты проводились при распараллеливании до 6 вычислительных узлов. Таким образом, размеры расчетных сеток составляют 220^3, 280^3, 320^3, 350^3, 380^3 и 400^3 узлов соответственно (толщина PML области равна 10 узлам).

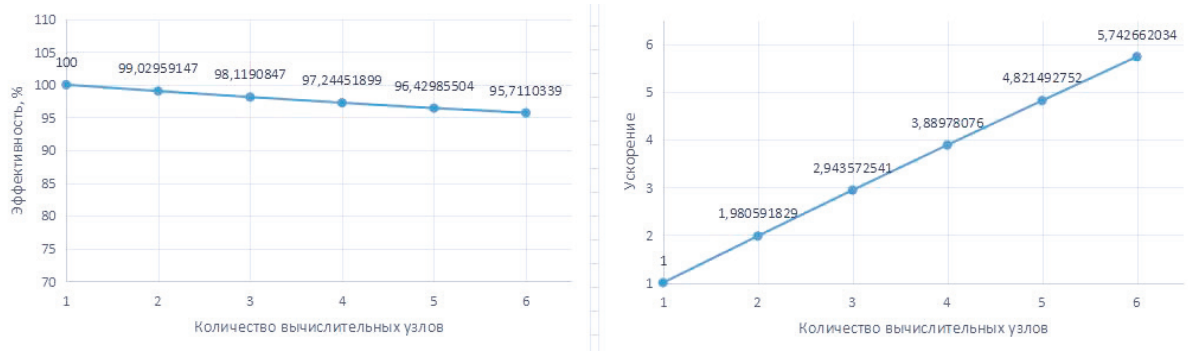


Рис. 7. Эффективность (слева) и сильная масштабируемость (справа) параллельной реализации для системы с распределенной памятью

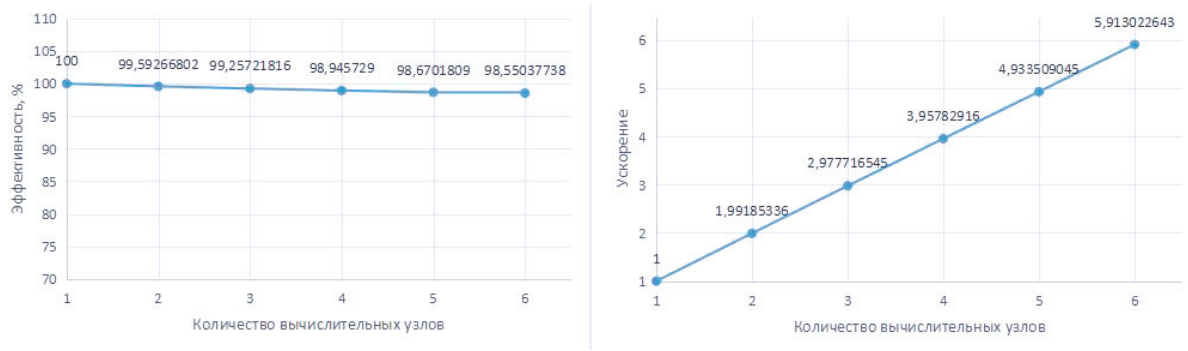


Рис. 8. Эффективность (слева) и слабая масштабируемость (справа) параллельной реализации для системы с распределенной памятью

В отличие от графиков на рисунках 3 и 6, на рисунке 8 можно заметить, что значения ускорения и эффективности превышают значения на рисунке 7. Это можно объяснить тем, что в предыдущих расчетах (раздел 5 и 6), количество узлов расчетной сетки на один MPI-процесс, при расчетах для построении графика слабой масштабируемости, было равным количеству узлов расчетной сетки на один MPI процесс при распараллеливании программы на 128 MPI-процессов. При этом размер задач получался меньше, чем при расчетах для построения графиков сильной масштабируемости (графики на рисунках 2 и 5).

Из приведенных графиков на рисунках 7 и 8 можно сделать вывод, что предложенный гибридный подход к распараллеливанию масштабируется практически без потери эффективности. Таким образом, получена эффективность 82%, относительно последовательной версии алгоритма (для расчетов графика слабой масштабируемости), при запуске программы на 6 вычислительных узлах, внутри которых было выполнено распараллеливание на 64 OpenMP-

потока. Получена эффективность распараллеливания, равная 98% при масштабировании до 6 вычислительных узлов.

8. Зависимость эффективности распараллеливания от толщины PML-области

В PML-области для расчета используются модифицированные схемы (7)-(9), требующие большей вычислительной сложности, что возможно может привести к неравномерной нагрузке MPI-процессов. В работе исследовано ускорение для двух реализаций параллельного алгоритма сеточно-характеристического метода в зависимости от количества узлов в PML-области. Тестовые расчеты проводились для параллельных реализаций с использованием технологии MPI и OpenMP.

Как правило, толщина PML-области варьируется от 5 до 20 узлов. На рисунках 9 и 10 представлены графики ускорения в зависимости от толщины PML-слоя. В расчетах использовалась сетка с исходным размером равным 200x200x200 узлов. Размеры исходной расчетной сетки взяты меньше, чтобы увеличить отношение количества узлов в PML к количеству узлов в исходной области. Используемая модель среды описана в пункте 5. Значение «size» на графиках означает количество узлов по координатным осям в PML-области.

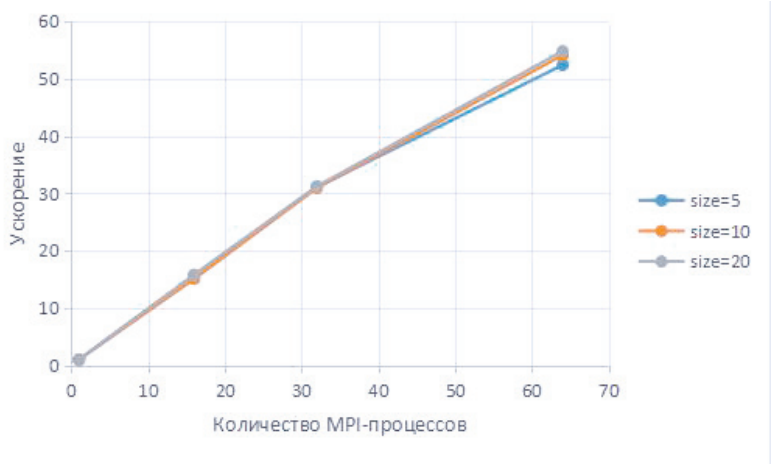


Рис. 9. Зависимость ускорения параллельной реализации для систем с распределенной памятью от толщины PML-области

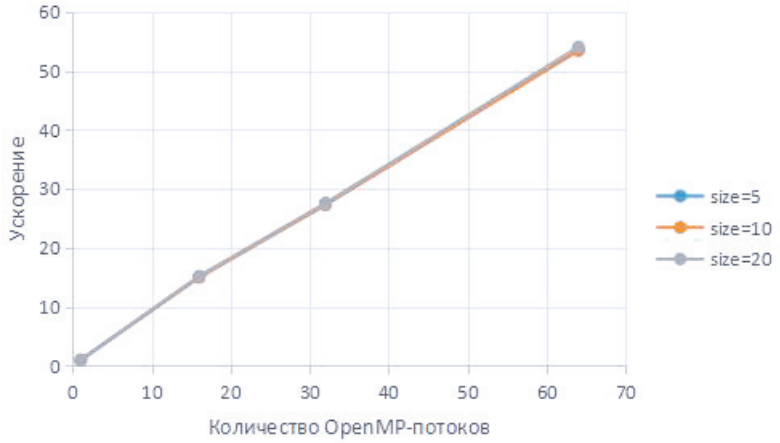


Рис. 10. Зависимость ускорения параллельной реализации для систем с общей памятью от толщины PML-области

Как можно видеть на представленных графиках, несмотря на то что в PML-области для расчета используются модифицированные схемы (7)-(9), вычисление которых требует большее число арифметических операций, эффективность реализованных параллельных алгоритмов не зависит от количества узлов в PML-области.

9. Примеры расчетов

Рассматривается модель слоистой среды, параметры которой приведены в таблице 1. Расчетная область представляет регулярную сетку в единичном объеме. В качестве источника внешних сил используется точечный импульс Рикера с частотой 30 Гц (14), действующий на главные компоненты тензора напряжений Коши в объеме верхнего слоя. Дневная поверхность предполагается свободной от внешних нагрузок. На остальных участках границы задаются поглощающие граничные условия в формулировке идеально согласованных слоев (PML). На рисунках 11-13 приведены волновые поля в последовательные моменты времени, в качестве параметра визуализации выбран модуль скорости перемещений. В правом нижнем углу изображены ортогональные срезы - плоскости, перпендикулярные осям координат, в которых визуализируется решение, а в остальных изображены ортогональные срезы с видом на плоскости, в которых они лежат.

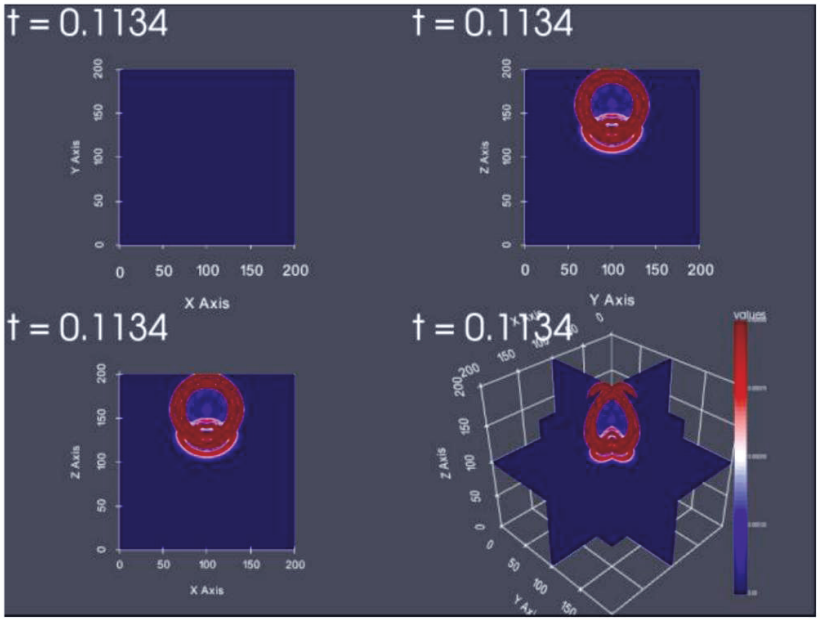


Рис. 11. Волновое поле в момент времени t=0.1134 - модуль скорости перемещений

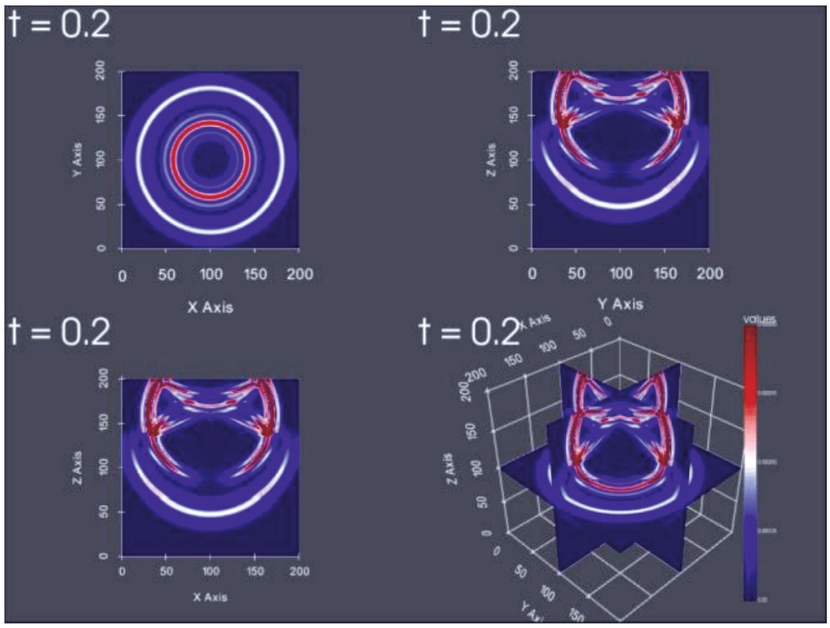


Рис. 12. Волновое поле в момент времени t=0.2 - модуль скорости перемещений

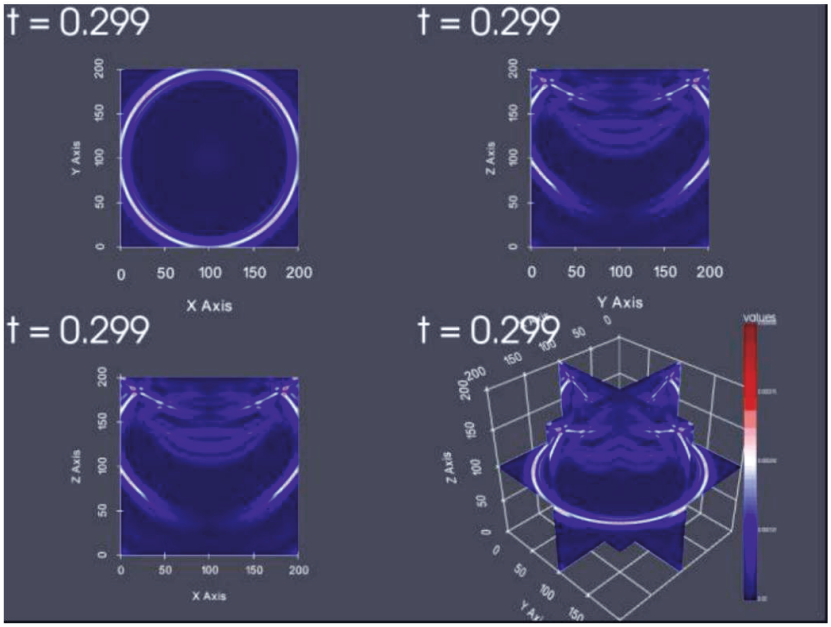


Рис. 13. Волновое поле в момент времени t=0.299 - модуль скорости перемещений

10. Заключение

В работе рассмотрена задача распространения упругих волновых возмущений в изотропной неоднородной среде. В качестве численного метода решения использовался сеточно-характеристический метод. Были предложены и реализованы три алгоритма распараллеливания с использованием технологии MPI для систем с распределенной памятью, технологии OpenMP для систем с общей памятью и алгоритм с совместным использованием технологий MPI и OpenMP.

Приведены графики эффективности распараллеливания, слабой и сильной масштабируемости. Для трех предложенных алгоритмов получена эффективность 84% на одном вычислительном узле относительно времени работы последовательной версии программы. Достигнута эффективность 98%, при масштабировании параллельного гибридного алгоритма до 6 двухпроцессорных вычислительных узлов. Для предложенных параллельных алгоритмов показана слабая зависимость ускорения от количества узлов в PML-области.

Литература

1. Холодов А.С. О критериях монотонности разностных схем для уравнений гиперболического типа / А.С. Холодов, Я.О. Холодов // Ж. вычисл. матем. Изм. Лист № докум. Подпись Дата Лист 41 3952.103209.000 ПЗ и матем. физ. / гл. ред. Е.Е. Тыртышников. – М.: Наука, 2006. – Т. 46, N 9. – С. 1638-1667
2. Бирюков В.А. Моделирование распространения упругих волн в геологической среде: сравнение результатов трех численных методов / В.А. Бирюков, В.А. Миряха, И.Б. Петров, Н.И. Холодов // Ж. вычисл. матем. и матем. физ. / гл. ред. Е.Е. Тыртышников. – М.: Наука, 2006. – Т. 56, N 6. – С. 1104-1114.
3. Хохлов Н.И., Петров. И.Б. Применение сеточно-характеристического метода для решения задач распространения динамических волновых возмущений на высокопроизводительных вычислительных системах. Труды ИСП РАН, том 31, вып. 6, 2019 г., стр. 237-252. DOI: 10.15514/ISPRAS–2019–31(6)–16
4. Разностные схемы третьего порядка точности для сквозного счета разрывных решений / В.В. Русанов // Доклады АН СССР. – М.: Наука, 1968. – Т. 180, N 6. – С. 1303-1305.

5. Ковеня В.М. Алгоритмы расщепления при решении многомерных задач аэрогидродинамики / В.М. Ковеня; отв. редактор Ю.И. Шокин; Рос. акад. наук, Сиб. отд-ние, Институт вычислительных технологий. – Новосибирск. Издво СО РАН, 2014. – 280 с.
6. Strang G. On the construction and comparison of difference schemes. SIAM Journal on Numerical Analysis, vol. 5, no. 3, 1968, pp. 506–517
7. Холодов Я.А. Монотонные разностные схемы высокого порядка аппроксимации для одномерных уравнений гиперболического типа : учеб. пособие / Я.А. Холодов, П.С. Уткин, А.С. Холодов. – М.: МФТИ, 2015. – 69 с.
8. E.H. Saenger, N. Gold, S.A. Shapiro Modeling the propagation of the elastic waves using a modified finite-difference grid // Wave Motion. 2000. Vol. 31. №. 1. P. 77–92.
9. Фаворская А.В. Метод исследования пространственных волновых явлений в средах со сложной структурой с помощью вычислительных экспериментов : автореферат дис. ... доктора физико-математических наук: 05.13.18 / Фаворская Алена Владимировна; [Место защиты: Моск. физ.-техн. ин-т (гос. унт)]. - Москва, 2018. - 47 с.
10. Абрамян М.Э. Параллельное программирование на основе технологии MPI 2.0 : учебник / М. Э. Абрамян. – Ростов н/Д; Таганрог: Изд-во ЮФУ, 2018. – 357 с.
11. Магомедов К.М., Холодов А.С. Сеточно-характеристические численные методы. – М.: Наука, 1988. – 287 с.

Исследование структуры серии измерений задержки при нагрузочном тестировании коммутационной среды вычислительного кластера

А.П. Волчанинов[1], А.Н. Сальников[1,2]

[1]Московский Государственный Университет им. М.В. Ломоносова,[2]Федеральный исследовательский центр «Информатика и управление» РАН.

Коммуникационная среда вычислительного кластера неоднородна и состоит из набора сложного в эксплуатации оборудования. В данной статье рассмотрен подход к измерению величин задержек между различными узлами кластера средствами библиотеки MPI в нескольких режимах. Особое внимание уделяется повторяющимся измерениям величины задержки с фиксированным размером сообщения и возникающим при этом цикличным конструкциям в данных. В ходе работы были собраны данные с нескольких экспериментальных стендов: вычислительные кластеры ФИЦ ИУ РАН и специальным образом организованный стенд на основе сети компьютерных классов факультета ВМК МГУ имени М.В. Ломоносова. Применены методы анализа периодичности данных, в частности, сравнение спектров при применении дискретного преобразования Фурье. Показаны некоторые характерные особенности для сетей на основе Infiniband и Ethernet.

Ключевые слова: MPI, сетевая инфраструктура, тестирование производительности, цикличность, Фурье анализ.

1. Введение

При проектировании суперкомпьютерного кластера и его эксплуатации одним из важнейших факторов, влияющих на производительность, является оптимальное состояние сетевой инфрастуктуры. Некоторые части инфраструктуры могут быть в пограничном состоянии, когда аппаратура ещё не фиксирует ошибку, но скорость передачи данных уже падает. Состояние сетевой инфраструктуры интересно как системным администраторам вычислительного кластера, так и пользователям/программистам, которым необходимо прогнозировать поведение программы, после того, как она будет помещена на узлы кластера для своего исполнения.

На текущий момент существует множество средств тестированния коммуникационной среды, которые составляют диагностические отчеты о состоянии сети. К таким средствам относятся: LinkTest by JSC [1], MVAPICH by NBCL of The Ohio State University [2], Intel MPI Benchmark [3] и другие. Использование подобных инструментов позволяет:

- Получить представление о задержках и пропускной способности при передаче данных между различными узлами, процессорными ядрами и даже отдельными видеокартами, размещёнными на узле (узлах).
- Обнаружить проблемные линки.
- Верифицировать предполагаемую модель коммуникаций в сети, найти несоответствия и на основе всего этого прогнозировать накладные расходы на передачу данных по сети.

Данная работа рассматривает метод на основе оценки времени пересылки сообщений между MPI-процессами. Как правило, проводится серия измерений некоторой величины и как ответ пользователю выдаются некоторые усреднённые значения. Существуют и другие работы, которые рассматривают структуру серии значений задержки, в том числе в

статьях [4] и [5]. Вв данной работе рассматривается структура полученных величин для серии измерений. В дальнейшем эти знания можно использовать для оценки качества набора измерений, для оценки достаточности измерений с точки зрения получаемых усреднённых значений, однако это остаётся за рамками данной работы. Здесь будут подмечены особенности внутри серии измерений. Данное исследование проводится на основе проекта clustbench [6–8]. Проект содержит множество программ для тестирования коммуникационной среды средствами MPI, для визуализации и анализа полученных результатов. В нем представлено большое количество тестов, различных по характеру нагрузки на сетевую инфраструктуру. В данной статье из всех возможных режимов будут приведены наблюдения только для измерений в режимах one_to_one и all_to_all. В ходе исследования были проанализированы данные, собранные с вычислительных кластеров ФИЦ ИУ РАН [9, 10], а также эксперементального стенда на базе компьютерных классов факультета ВМК МГУ, чтобы продемонстрировать работоспособность подхода не только для сетей суперкомпьютеров, но и «обычных» сетей построенных на «обычных» Ethernet карточках и не слишком мощных коммутаторах.

2. Подход к определению количества измерений

Данная работа является частью большого исследования коммутационной среды вычислительных кластеров. Результаты исследований, приведенные далее, будут использованы для оценки качества собранных данных и составления критерия достаточности проводимых измерений (остановки процесса тестирования). Для этого авторами было решено провести подробный анализ структуры данных, с которыми приходится иметь дело. Также рассмотреть подходы к оценке качества измерений задержки в аналогичных проектах. При изучении документации рассмотренных проектов, оказалось, что все, кроме intel mpi benchmark, оставляют вопрос определения количества итераций на откуп пользователю и предлагают ему самому определить статическое число итераций. В проекте от intel реализовано следующее решение: есть опция iterpolicy, которая может принимать значения: dynamic/multiple_np/off. dynamic: Уменьшает количество итераций, когда задержка достигает максимального допустимого значения. multiple_np: Уменьшает количество итераций по мере роста размера сообщения. Однако, как можно видеть, данные решения не основываются ни на каких проверках «качества» собираемых данных, что говорит о том, что для нашей задачи данные решения не подходят.

3. Способ измерения задержек

Под «задержкой» понимается время доставки MPI- сообщения. Более полное представление об этом будет сформировано далее. В режиме one_to_one единовременно пересылаются сообщения только между одной парой запущенных MPI-процессов. Схема вычисления задержки следующая: MPI_Barrier осуществляет синхронизацию между отправителем и получателем. После чего:

- На стороне отправителя: происходит посылка процессу-получателю сообщения необходимой длины при помощи MPI_Send.
- На стороне получателя: ставится временная метка ts_1, происходит получение посылки от процесса-отправителя с помощью MPI_Recv, ставится временная метка ts_2.
- Значение задержки вычисляется по формуле: $ts_2 - ts_1$.

В режиме all_to_all все MPI-процессы шлют сообщения одновременно всем другим процессам. Схема вычисления задержки следующая:

1. При помощи MPI_Barrier происходит синхронизация между всеми p процессами. Здесь p – число процессов в тесте.

2. В каждом процессе ставится временная метка ts_{i0}, где i – номер процесса.

3. Все процессы используют MPI_Isend для посылки сообщения нужной длины всем процессам.

4. Все процессы n раз используют MPI_Irecv.

5. Далее в цикле по выходу из MPI_Waitany фиксируется время окончания соответствующего обмена и изменяется временная метка окончания. Таким образом, будут получены сообщения от всех остальных процессов. После получения сообщения от процесса j в процессе i ставится временная метка ts_{ij}.

6. Значение задержки между процессом j и процессом i вычисляется по формуле: $ts_{ij} - ts_{i0}$.

Из-за возможного неодновременного выхода задействованных MPI-процессов из функции MPI_Barrier метод может быть не точен для сообщений малой длины. Начиная с определенной длины сообщения описанный эффект будет незначительным. Кроме того существуют различные решения, реализующие барьерную синхронизацию [13, 14]. Использование подобных методов синхронизации может позволить повысить точность представленного инструмента тестирования. Далее будут описаны стенды, на которых были запущены описанные режимы тестирования.

4. Описание стендов

В ходе работы тесты были запущены на суперкомпьютерных комплексах ФИЦ ИУ РАН и модельном стенде в машинном зале ВМК МГУ. Приведем их описание:

1. Модельный стенд в машинном зале.
   В состав стенда вошли 16 машин со следующими характеристиками: 1*CPU Intel Xeon E3-1245 v5 (3.50GHz, 4 Core), 8 Gb RAM, топология сети представлена на рис. 1. Пунктирными линиями обозначено наличие иных сетевых интерфейсов, подключенных к устройствам (эти линки при тестировании не задействованы).

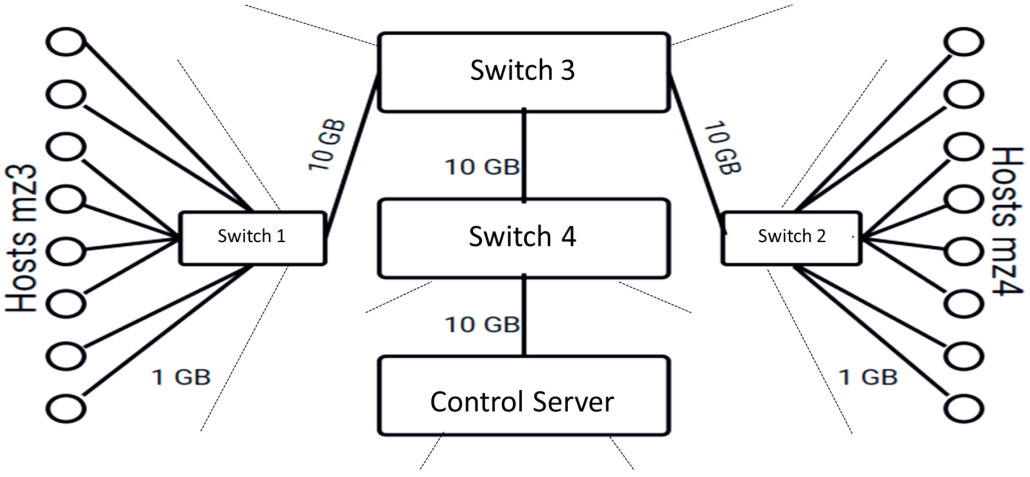


Рис. 1. Топология модельного стенда в машинном зале

2. Высокопроизводительный вычислительный комплекс архитектуры Intel с IB EDR 100Gb/s (шина PCIe 4.0)
   В состав комплекса входят 14 узлов на основе серверной платформы FusionServer

2288H со следующими характеристиками: 2*CPUIntel Xeon Gold 6338 (2.0 GHz, 32 Core), 512 Gb RAM, 2*100G InfiniBand, 2*100G Ethernet. Топология и модели сетевых устройств представлены на рис. 2.

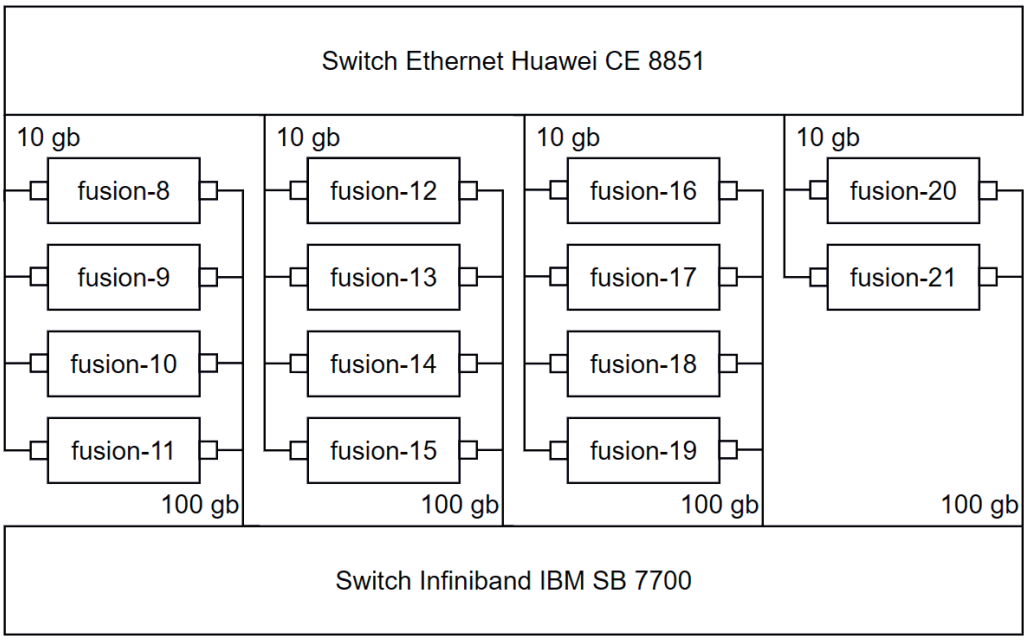


Рис. 2. Топология комплекса intel_ib ФИЦ ИУ РАН

3. Высокопроизводительный вычислительный комплекс архитектуры Intel.
   В состав комплекса входят 9 узлов на основе серверной платформы Huawei Server XH620 со следующими характеристиками: 2*CPU Intel Xeon CPU E5-2683 v4 (2.1 GHz, 16 Core), 512 Gb RAM, 2*10G Ethernet. Топология и модели сетевых устройств представлены на рис. 3.

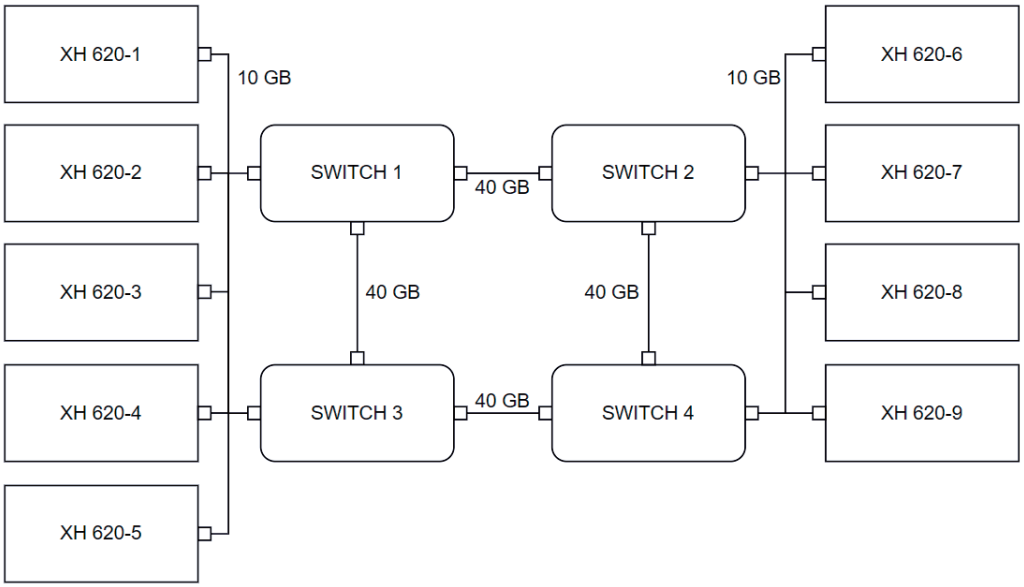


Рис. 3. Топология комплекса intel ФИЦ ИУ РАН. Модель коммутаторов: HUAWEI CE 8860

5. Формат собираемых данных

Для того, чтобы дальнейшие иллюстрации были понятны, приводится описание схемы сбора информации о задержках. При запуске теста пользователь указывает следующие параметры: beg, fin, step, n. Где beg – начальная длина сообщений, fin – конечная длина сообщений. Длина сообщения увеличивается линейно с шагом, равным step. n – количество итераций при пересылке сообщений одной длины между одной парой MPI-процессов. В результате выполнения теста создается трехмерная матрица со следующими координатами: $< src, dest, len >$, где src – номер процесса-отправителя, dest – номер процесса получателя, len – длина сообщений. Рассмотрим ячейку $< i, j, k >$ матрицы $A \in R^3$. Процесс с номером i шлет процессу с номером j ровно n сообщений длины $beg + k \cdot step$. Итого, получаем вектор из n значений задержки. После этого по вектору, как по набору значений дискретной случайной величины, вычисляем четыре статистических параметра: математическое ожидание (avg), дисперсию (dev), минимум (min) и медиану (med). Статистические параметы, как вектор из четырех элементов $< avg, dev, min, med >$ записываем в ячейку i, j, k матрицы A. В ходе исследования было решено добавить возможность собирать еще одну матрицу $B \in R^4$. Она образутеся также, как и матрица A, но в ячейку $< i, j, k >$ записывается весь вектор из n значений измеренной величины задержки. Таким образом, добавляется еще одна координанта r, соответствующая номеру итерации, а в ячейках матрицы теперь содержатся индивидуальные (не усреднённые) значения задержки.

6. Анализ собранных данных

В процессе исследования тесты были запущены на вычислительных кластерах ФИЦ ИУ РАН, сетевые инфраструктуры которых основаны на технологиях 10G Ethernet и 100G Infiniband. Также, тесты были запущены на «стенде», построенном на основе компьютеров в «машинном зале» ВМК МГУ, объединенных в локальную сеть. На рис. 4 приведен пример типичной диагностики, полученной по результатам тестирования: в данном случае это срезы трёхмерной матрицы A при фиксированной длине сообщения. Данные были собраны при запуске тестов на кластере ФИЦ ИУ РАН, сетевая инфраструктура которого основана на технологии Ethernet. На рисунке легенда с соответствием цвета конкретным значениям приведена для параметра avg. Данный пример подтверждает адекватность используемого метода измерения задержки: как можно видеть, задержка внутри узлов получается меньше, чем между узлами. Однако, данные получаются не совсем однородными, поэтому было решено более подробно изучить структуру самой серии измерений задержки.

На рис. 5 и рис. 6 приведены срезы 4-х мерных матриц B для тестов в режимах one_to_one и all_to_all. Зафиксировав две координаты: длину сообщения (msg_len) и номер процесса-получателя (dest). Здесь получается двумерный срез, где по вертикали будет отложен номер процесса отправителя, а по горизонтали – номер итерации. Тесты проводились с числом итераций, равным 1000, поэтому матрицы представлены частично.

Под рисунками указаны:

- Стенд, с которого были собраны данные (RAS – ФИЦ ИУ РАН, MZ – машинный зал ВМК МГУ).
- Технология, используемая в сетевой инфраструктуре.
- Конфигурация использовавшихся узлов и процессов: $M \times N$, где M – число задействованных узлов, N – число процессов, запущенных на каждом узле.
- Диапазон отображаемых итераций.

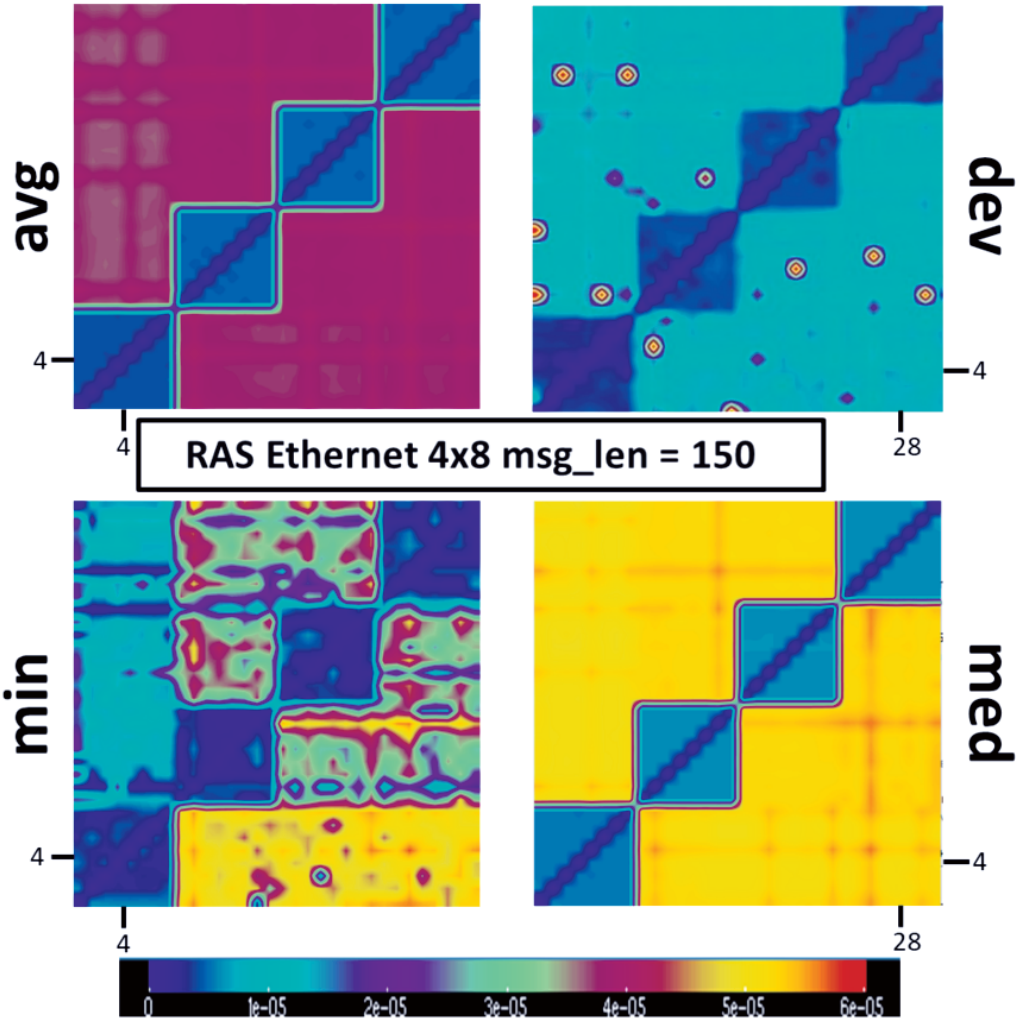


Рис. 4. Срез матрицы задержек вычислительного кластера ФИЦ ИУ РАН. Используемая технология: 10G Ethernet

- Длина сообщения (значение msg_len, по которому сделан срез).
- Номер процесса-получателя (значение dest, по которому сделан срез).

Как можно видеть, в режиме one_to_one периодически происходит увеличение задержки для всех отправителей, находящихся в узле, отличном от узла, зафиксированного получателя. Этим всплескам соответсвуют вертикальные линии: фиолетовые - на верхнем рисунке и желтые - на нижнем. В режиме all_to_all скачки происходят чаще, однако, можно сделать вывод об определенном чередовании наблюдаемых значений. Характер наблюдаемых измерений в обоих режимах явно цикличен.

7. Применение дискретного преобразоавния Фурье

Чтобы более формально и подробно изучить характер наблюдаемой цикличности, было решено прибегнуть к применению алгоритма быстрого преобразования Фурье для подсчета ДПФ [11] (см. формулу (1)).

$$X_k = \sum_{-N/2}^{N/2-1} x_n e^{-\frac{2\pi i}{N} kn} \qquad k = -N/2, .., N/2 - 1 \tag{1}$$

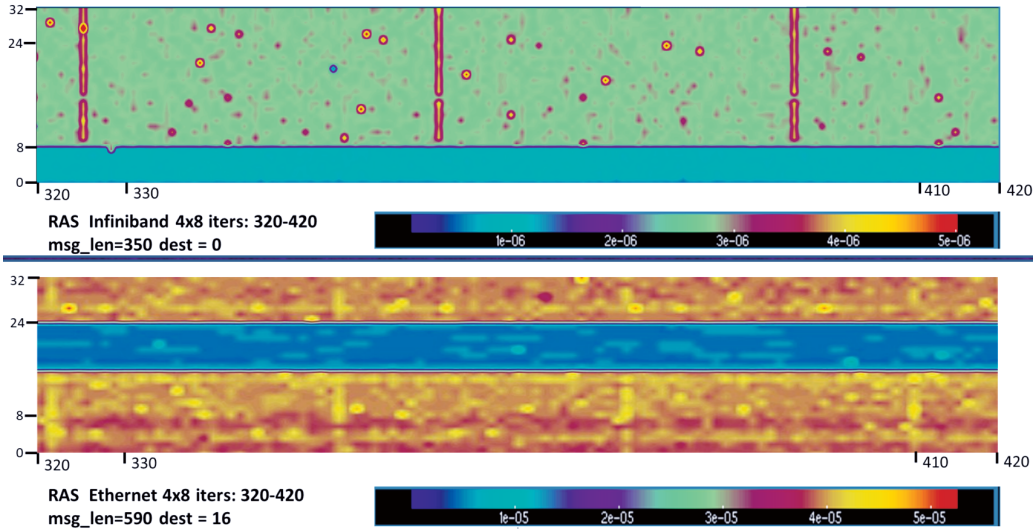


Рис. 5. значения задержек в режиме one_to_one

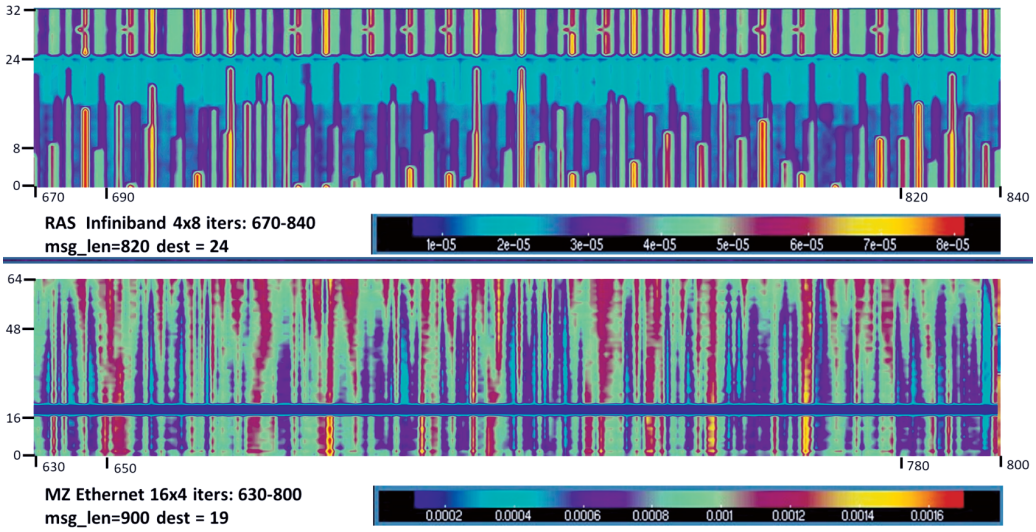


Рис. 6. Значения задержек в режиме all_to_all

Для исследования данных с помощью ДПФ был написан модуль на языке Python3. Приведем его описание:

- Данные на входе.
  1. Путь к файлу в формате netcdf, содержащему четырехмерную матрицу B.
  2. Длина сообщения msg_len, по которой будет сделан срез.
  3. Длина окна window_len для подсчета ДПФ.
- Процесс вычисления.
  1. В данной четырехмерной матрице B, делаем срез по длине сообщения msg_len.
  2. Идем в цикле по всем возможным парам отправитель-получатель. Для каждой такой пары нам известен вектор задержек длиной n_iters (кол-во итераций, использовавшееся в тесте).

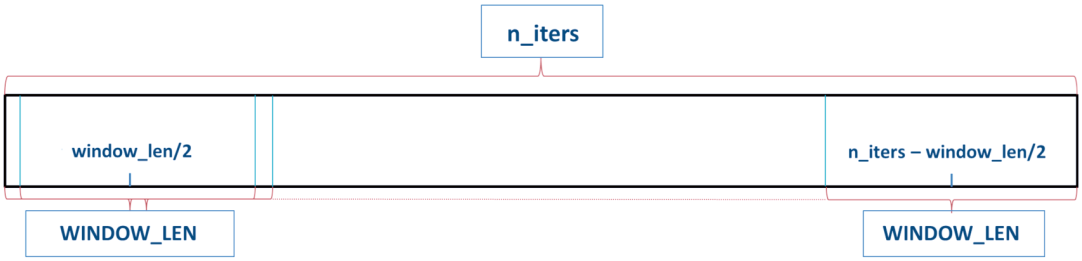


Рис. 7. Способ выбора окон для применения FFT.

3. Выбираем все возможные окна длины window_len (состоящие из подряд идущих элементов) из данного вектора. Всего их будет $n\_iters - window\_len$, ведь координата центра самого левого окна – $window\_len/2$, а самого правого – $n\_iters - window\_len/2$ (см. рис. 7).
4. В каждом окне считаем мощности гармоник с помощью функции rfft библиотеки scipy.fft. Гармоник получится ровно $N/2$. Мощности гармоник получаются комплексными, поэтому для дальнейшего анализа было решено записывать модули получающихся комплексных чисел.
5. Записываем значения $N/2$ гармоник для всех $n\_iters - window\_len$ окон для всех пар процессов-отправителей и процессов-получателей (src, dest).

- Данные на выходе.
  Файл в формате netcdf, содержащий четырехмерную матрицу D со следующими измерениями:

  1. Номер процесса-отправителя (src).
  2. Номер процесса-получателся (dest).
  3. Номер окна (win).
  4. Номер гармоники (sp).

  В ячейках матрицы содержатся вещественные числа, характеризующие мощность соответствующих гармоник.

8. Анализ результатов применения дискретного преобразования Фурье

С помощью описанного в предыдущем пункте модуля были произведены исследования рассмотренных ранее матриц B. Двумерные срезы полученных матриц D при фиксированных номерах процесса-получателя и отправителя приведены на рис. 8 и рис. 9.
Под каждым из рисунков указаны:

- Стенд, с которого были собраны данные (RAS – ФИЦ ИУ РАН, MZ – машинный зал ВМК МГУ).
- Технология, используемая в сетевой инфраструктуре.
- Конфигурация использовавшихся узлов и процессов: $M \times N$, где M – число задействованных узлов, N – число процессов, запущенных на каждом узле.
- Диапазон отображаемых окон (нумерация от левого к правому).
- Длина сообщения (значение msg_len, по которому пользователь указал сделать срез).

- Номер процесса-отправителя (значение src, по которому сделан срез).
- Номер процесса-получателя (значение dest, по которому сделан срез).
- Длина окна (значение window_len, указанное пользователем при запуске программы).

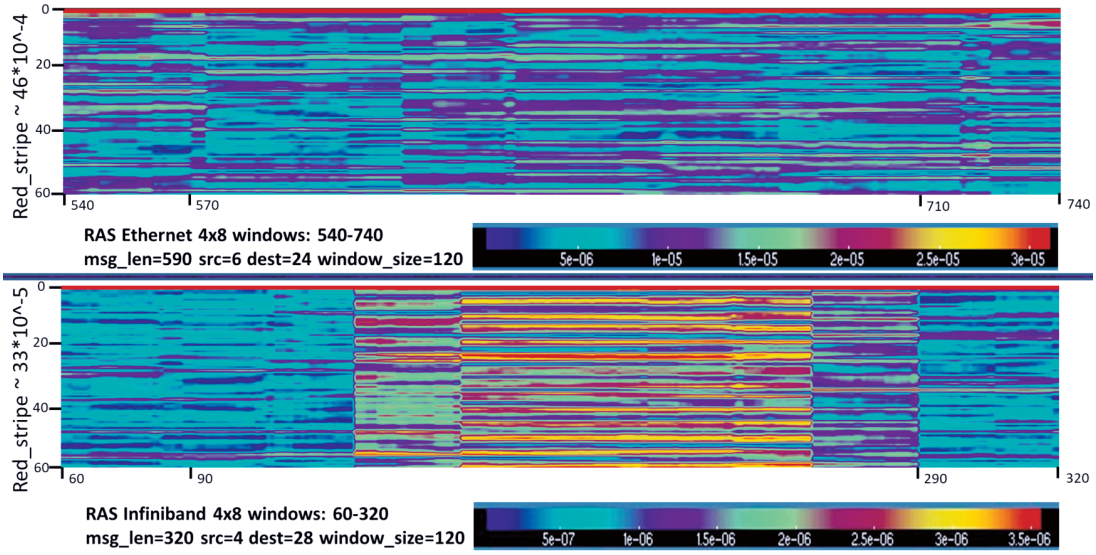


Рис. 8. Мощности гармоник в режиме one_to_one.

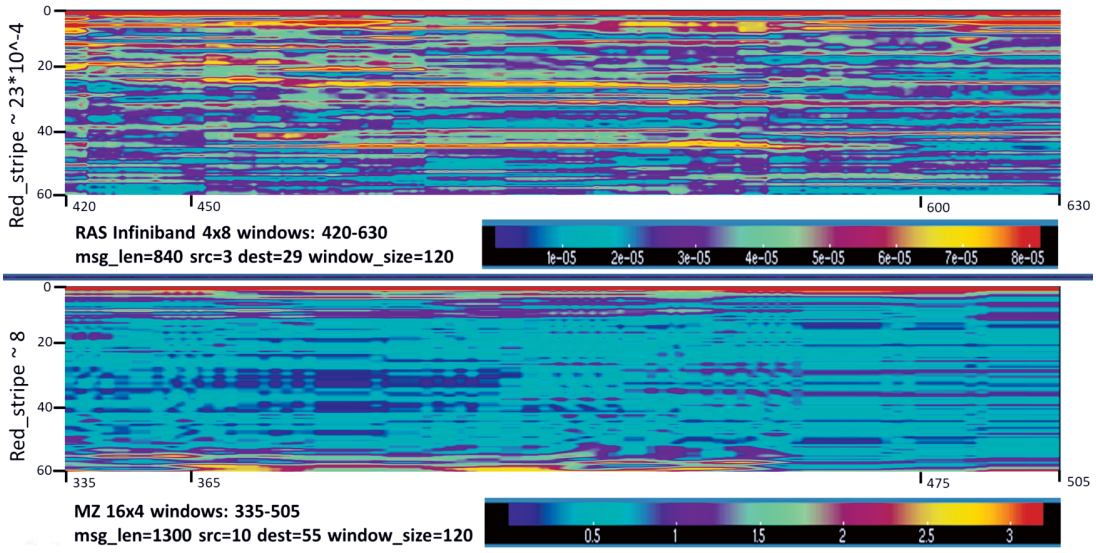


Рис. 9. Мощности гармоник в режиме all_to_all.

На обоих рисунках в отображаемой двумерной матрице по горизонтали – номер окна, а по вертикали – номер гармоники. Диапазон отображаемых значений был выбран таким образом, чтобы продемонстрировать, что характер данных не совсем однородный. Так как на всех четырех рисунках явно выделяется строка с наибольшими значениями, которая соответствует значениям самой «медленной» гармоники в каждом из окон (первые строки матриц выделяются красным цветом, который соответсвует наибольшим значениям). Из-за этого появляется иллюзия однородности. Впрямую, если не отсекать по порогу значения медленной гармоники, быстрые гармоники будут выглядеть как гармоники с нулевым вкладом.

Фактические значения мощностей медленных гармоник кратно превышают значения в легенде: средние значения в них приведены слева от рисунков в формате «red_Stripe {val}». Видно, что в случае проведения теста на оборудовании, предназначенном для суперкомпьютеров, мощность самой медленной гармоники превышает мощности других гармоник почти на два десятичных порядка, тогда как на экспериментальном стенде в машинном зале ВМК, мощности отличаются всего в 3-4 раза. Визуализация результатов тестирования позволяет свидетельствовать о периодичности наблюдаемых данных.

9. Выводы

Подводя итоги, можно сказать следующее. В результате проведенных исследований во всех собранных данных наблюдались повторяющиеся паттерны. Более дорогие и совершенные сетевые инфраструктуры, ориентированные на работу в суперкомпьютерных кластерах, показали более однородные и цикличные результаты. При тестировании в режиме all_to_all наблюдаются частые скачки задержки. Предложенная модификация с возможностью сбора четырехмерной матрицы позволила отслеживать моменты переполнения буферов в сети на пути следования посылок — при достижении критического размера сообщения значения задержки равномерно увеличивались для всех пар задействованных процессов. Анализ мощностей гармоник позволил утвердить факт периодичности наблюдаемых данных. В дальнейшем, планируется применить полученные результаты вместе с результатами, полученными в статье [12] для того, чтобы автоматизировать определение количества итераций при тестировании.

Литература

1. Jülich Supercomputing Centre (JSC). LinkTest. URL: https://www.fz-juelich.de/en/ias/jsc/services/user-support/jsc-software-tools/linktest (дата обращения: 24.04.2023).

2. MVAPICH. MPI over InfiniBand, Omni-Path, Ethernet/iWARP, RoCE, and Slingshot. URL: https://mvapich.cse.ohio-state.edu/ (дата обращения: 24.04.2023).

3. Gergana S. Slavova. Introducing Intel® MPI Benchmarks. URL: https://www.intel.cn/content/www/cn/zh/developer/articles/technical/intel-mpi-benchmarks.html (дата обращения: 24.04.2023).

4. Сальников Алексей Николаевич, Майсурадзе Арчил Иериевич, Андреев Дмитрий Юрьевич, Костин Григорий Александрович Кластеризация результатов тестирования коммуникационной среды многопроцессорных систем: единицы анализа, исследование методов, визуализация результатов // Вестник УГАТУ. 2012. №6 (51). URL: https://cyberleninka.ru/article/n/klasterizatsiya-rezultatov-testirovaniya-kommunikatsionnoy-sredy-mnogoprotsessornyh-sistem-edinitsy-analiza-issledovanie-metodov (дата обращения: 25.04.2023).

5. Kishwar Ahmed, Mohammad Obaida, Jason Liu, Stephan Eidenbenz, Nandakishore Santhi, and Guillaume Chapuis. 2016. An Integrated Interconnection Network Model for Large-Scale Performance Prediction // Proceedings of the 2016 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (SIGSIM-PADS '16). Association for Computing Machinery, New York, NY, USA, 177–187. URL: https://doi.org/10.1145/2901378.2901396.

6. Salnikov A.N., Andreev D.Yu, Lebedev R.D. Toolkit for analyzing the communication environment characteristics of a computational cluster based on MPI standard functions //

Moscow University Computational Mathematics and Cybernetics 2012. Том 36. Стр 41-49. DOI: 10.3103/S0278641912010074.

7. Сальников А. Н, Бегаев А. А. Инструментальная система покомпонентного тестирования сети вычислительного кластера // Труды Международной конференции «Параллельные вычислительные технологии (ПаВТ) 2021». DOI: 10.14529/pct2021.

8. Clustbench. Benchmarks and analysis of interconnection in HPC cluster. URL: https://github.com/clustbench/network-tests2/tree/new-tests_as_plugins (дата обращения: 24.04.2023).

9. Центр коллективного пользования «Высокопроизводительные вычисления и большие данные» ФИЦ ИУ РАН. Инструкция по доступу к высокопроизводительному вычислительному кластеру. URL: http://docs.ckp.frccsc.ru/manual-hpc-access.docx (дата обращения: 24.04.2023).

10. ЦКП «Информатика». Перечень оборудования. URL: https://www.frccsc.ru/ckp/hard (дата обращения: 24.04.2023).

11. Кандидов В. В., Чесноков С. С., Шленов С. А. Дискретное преобразование Фурье// Отдел оперативной печати физического факультета МГУ. Москва 2019. ISBN: 978-5-8279-0179-2.

12. Хабирова Э. Р, Сальников А. Н. Метод описания топологической структуры вычислительных кластеров, основанный на операциях произведений подграфов // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика 2022. DOI: 10.14529/cmse220302.

13. Волобуев, С. В. Анализ средств барьерной синхронизации / С. В. Волобуев, В. Н. Николаев // Молодой ученый. — 2010. — № 11 (22). — Т. 1. — С. 50-52. — URL: https://moluch.ru/archive/22/2293/ (дата обращения: 07.06.2023).

14. Макагон Д.В. Варианты реализации барьерной синхронизации для высокоскоростных коммуникационных сетей с топологией «многомерный тор». / Макагон Д.В., Сыромятников Е.Л. // Успехи современной радиоэлектроники, № 7, 2012, с. 21-28.

Метод прореживания нейронный сетей в процессе обучения

Д.А. Новиков, Д.Ю. Буряк

Московский государственный университет имени М.В.Ломоносова, Факультет вычислительной математики и кибернетики

В рамках данной работы был разработан метод прореживания глубокой нейронной сети в процессе обучения. Идея метода состоит в исключении параметров из модели и последующей оценке их влияния на точность работы нейронной сети на каждой эпохе обучения. После сбора статистики за несколько итераций обучения происходит окончательное исключение тех параметров модели, которые не удовлетворяют критерию. Проведено сравнительное исследование разработанного метода с другими методами прореживания.

Ключевые слова: сверточные нейронные сети, прореживание, обучение

1. Введение

Современные успехи и развитие в сфере глубокого обучения обусловлены не только разработкой значительного количества новых архитектур нейронных сетей и вовлечением в данную область большого числа людей, но и постоянно растущими вычислительными мощностями.

Глубокие нейронные сети имеют десятки, а иногда сотни скрытых слоев и сотни нейронов в каждом из этих слоев. Такие сети имеют огромное количество параметров [1]. Их обучение стало возможно благодаря массовому распространению высокопараллельных вычислительных устройств – GPU, TPU и других. Но большинство людей не имеют возможности запускать приложения на мощных вычислительных устройствах с тысячами CUDA ядер.

Однако известно, что многие параметры нейронной сети излишни и оказывают незначительное влияние на результат ее работы. Таким образом, обнаружение и удаление таких параметров позволит существенно сократить вычислительную сложность сети при минимальном ухудшении ее точности. Для этого применяются алгоритмы прореживания нейронной сети.

Целью данной работы является анализ существующих алгоритмов прореживания глубоких сверточных нейронных сетей и исследование подходов к повышению их эффективности.

2. Обзор существующих подходов

Прореживание нейронных сетей – метод упрощения структуры модели нейронной сети, заключающийся в поиске и удалении параметров, которые оказывают малое влияние на точность работы сети. Обнаружение таких параметров выполняется путем вычисления различных критериев, характеризующих степень их значимости для сети.

Рассмотрим основные стратегии, применяемые в алгоритмах прореживания, а также критерии, используемые для выделения параметров сети, удаление которых минимально повлияет на качество ее работы.

2.1. Итеративное прореживание

Методы итеративного прореживания применяются к предварительно обученным нейронным сетям. Используя выбранный критерий выполняется удаление части параметров

сети. После этого обновленная архитектура дообучается в течение нескольких эпох. Описанные действия повторяются, пока качество результата работы прореженной нейронной сети не уменьшиться ниже заданного порога относительно качества первоначальной модели, либо пока не будет удалено необходимое количество параметров.

Рассмотрим некоторые критерии исключения параметров нейронной сети.

2.1.1. Прореживание на основе величины (Magnitude-based pruning)

Одним из наиболее эффективных и распространенных подходов является magnitude–based pruning [2]. Основная идея критерия – исключение из модели нейронной сети весов, абсолютная величина которых меньше заданного положительного порогового значения. Например, операции свертки имеют вид:

$$y_i = \sum_{j=0}^{n} w_{ij} x_j, i = \overline{1, m},$$

из которой следует, что веса с наименьшей абсолютной величиной вносят наименьший вклад в результирующую сумму, и, следовательно, их удаление может незначительно повлиять результаты работы сети.

Из-за простоты реализации magnitude-based pruning часто используется на практике, однако его эффективность падает, когда нейронная сеть имеет сильную прореженность.

2.1.2. Прореживание на основе разложения функции потерь в ряд Тейлора (Optimal Brain Damage)

Обучение модели нейронной сети с параметрами w на обучающей выборке D заключается в нахождении таких параметров w_t, для которых значение функции потерь L_D будет наименьшим. Критерий Optimal Brain Damage основан на разложении оптимизируемой функции потерь в ряд Тейлора [3]

Найдем локальную аппроксимацию функции L_D в окрестности точки w_t с помощью разложения в ряд Тейлора:

$$L_D(w + \Delta w) = L_D(w) + g^T(w)\Delta w + \tfrac{1}{2}\Delta w^T H \Delta w + o(|w^3|),$$

где Δw – возмущение вектора параметров w, $g = \frac{\partial L_D}{\partial w}$ – градиент, $H = \frac{\partial^2 L_D}{\partial w^2}$ – матрица Гессе. Предполагается, что функция $L_D(w)$ достигает своего минимума при значении параметров $w = w_t$ и ее поверхность квадратична. Таким образом, предыдущее выражение можно упростить и представить в виде:

$$\Delta L_D = L_D(w + \Delta w) - L_D(w) = \tfrac{1}{2}\Delta w^T H \Delta w.$$

Для нахождения исключаемого параметра требуется минимизировать квадратичную форму $\Delta w^T H \Delta w$ относительно Δw при ограничениях $e_i^T \Delta w + w_i = 0$, для всех значений i. Где e_i вектор, i–й элемент которого равен единице, все остальные элементы равны нулю. Индекс i элемента, для которого достигается минимум квадратичной формы:

$$i = \arg\min_{j=1}(\min_{\Delta w}(\Delta w^T H \Delta w | e_i^T \Delta w + w_i = 0)).$$

Задача условной минимизации решается с помощью введения Лагранжиана

$$S = \Delta w^T H \Delta w - \lambda(e_i^T \Delta w + w_i),$$

где λ – множитель Лагранжа. Дифференцируя Лагранжиан по приращению параметров и приравнивая его к нулю получаем (для каждого индекса i параметра w_i):

$$\Delta w = \frac{w_i}{[H^{-1}]_{ii}} H^{-1} e_i.$$

Этому значению вектора приращений параметров соответствует минимальное значение Лагранжиана:

$$E_i = \frac{w_i^2}{2[H^{-1}]_{ii}}.$$

Полученное выражение называется мерой выпуклости функции потерь L_D при изменении параметра w_i.

В соответствие с алгоритмом Optimal Brain Damage из нейронной сети удаляется заданная доля параметров с минимальными значениями E_i.

2.1.3. Прореживание на основе L_1 нормы

Данный метод прореживания нейронной сети заключается в удалении целых фильтров в сверточных слоях нейронной сети, L_1 норма которых меньше определенного порогового значения [5].

L_1 норма тензора $F \in \mathbb{R}^n$ вычисляется на каждой итерации прореживания нейронной сети и имеет вид:

$$\|F\| = \sum_{k=1}^{n} |w_k|.$$

2.1.4. Прореживание на основе расстояния Евклида

Данный метод прореживания нейронной сети заключается в удалении параметров или тензоров фильтров сверточных слоев, расстояние Евклида между которыми меньше определенного порогового значения [6].

Евклидово расстояние между каждой парой тензоров параметров $P, Q \in \mathbb{R}^n$ вычисляется на каждой итерации прореживания модели и имеет вид:

$$d(P, Q) = \sqrt{\sum_{k=1}^{n} (w_k^{(p)} - w_k^{(q)})^2}.$$

Таким образом, вычисляется симметричная матрица расстояний $D \in \mathbb{R}^{I \times I}$:

$$D = \begin{cases} \mathrm{d}(\mathrm{F}_i, F_j), i \neq j \\ 0, \mathrm{i} = \mathrm{j} \end{cases},$$

здесь I – количество тензоров параметров модели, F_i – тензор с номером i.

Далее, согласно значениям расстояний в вычисленной матрице, из модели удаляется по одному из каждой пары тензоров, расстояние между которыми меньше некоторого порогового значения.

2.2. Прореживание модели в процессе обучения

Существуют алгоритмы, позволяющие начать прореживание нейронной сети сразу после инициализации, то есть уже в процессе обучения исключать параметры модели на основе некоторого критерия. Достоинством данного подхода является сокращение времени на построение прореженной нейронной сети, т.к. обучение и удаление избыточных параметров выполняется одновременно.

Выделяют две основные стратегии:

- **Фиксированная прореженность**

В данном подходе маски нулевых и ненулевых весов фиксируются на первом шаге обучения нейронной сети,согласно некоторому критерию, и не будут меняться в процессе обучения.

В качестве примеров данного подхода будут рассмотрены два метода: **SNIP** и **GraSP**.

- **Динамическая прореженность**

  В данном подходе маски нулевых и ненулевых весов обновляются согласно некоторому критерию. Часть весов, что были отличны от нуля исключаются, но, следуя определенному методу, добавляются новые, которые ранее были нулевыми.

  В качестве примеров данного подхода будут рассмотрены два метода: **SET** и **RigL**.

2.2.1. Прореживание на основе влияния соединений (SNIP)

Метод заключается в исключении весов и связей нейронной сети, несущих избыточную информацию, еще до обучения модели. Для этого вводится критерий для измерения чувствительности в отношении того, какое влияние оказывают параметры на функцию потерь независимо от того, положительное оно или отрицательное. [7]. Таким образом, критерий фактически обнаруживает параметры нейронной сети, которые важны для данной задачи.

2.2.2. Прореживание на основе сохранения градиента (GraSP)

После исключения некоторых параметров (например весов или связей) нейронной сети из обучения, модель будет иметь редкую связность, что может привести к проблеме затухающего градиента, то есть обучение проходит неэффективно или останавливается. Поэтому основной целью подхода является сохранение, а возможно и увеличение градиента после прореживания [8].

Операция прореживания заключается в добавлении возмущения δ к исходным весам. Затем используется приближение Тейлора, чтобы охарактеризовать, как удаление одного веса повлияет на градиент:

$$S(\delta) = \Delta L(\theta_0 - \delta) - \Delta L(\theta_0) = 2\delta^T H g + O(\| x \|_2^2),$$

θ_0 – начальные веса модели, L – функция потерь, $S(\delta)$ приблизительно измеряет изменение функции потерь. H – матрица Гессе фиксирует зависимости между разными весами и, таким образом, помогает предсказывать эффект удаления нескольких весов.

GraSP использует для вычисления значимости каждого веса, критерий, позволяющий отражать изменение градиента после исключения веса. В частности, если $S(\delta)$ отрицательный, то удаление соответствующих весов уменьшает градиент, а если он положительный, то увеличит. Целью является исключить те веса, удаление которых не уменьшает градиент. По результатам вычисления значимости весов для расчета градиента получается маска исключенных весов.

2.2.3. Эволюционное обучение (SET)

В основе метода SET лежат эволюционные алгоритмы [9]. Начиная прореживание нейронной сети с этапа проектирования (до обучения), приводит к существенному сокращению количества параметров, а также к сокращению памяти и повышению эффективности вычислений. При исследовании подхода обнаружено, что нейронные сети отлично работают с редко связанными слоями. Редко связанные слои, обученные с помощью SET, могут заменять любые полностью связанные слои в нейронной сети без снижения точности, имея при этом квадратично меньшее количество параметров даже на этапе проектирования (до обучения). Это приводит к уменьшению требований к памяти и может привести к сокращению времени вычислений.

Для каждого разреженного связного слоя SC_k нейронной сети в конце периода обучения удаляется часть весов, наиболее близких к нулю. Затем новые веса добавляются случайным образом в том же количестве, что и ранее удаленные. Далее выполняется новая тренировочная эпоха, и процедура удаления и добавления весов повторяется. Процесс продолжается в течение конечного числа эпох обучения, как обычно при обучении модели.

В процессе обучения после каждой эпохи обучения удаляется часть наименьших положительных весов и наибольших отрицательных весов слоя SC_k. Удаленные веса – самые близкие к нулю, поэтому их удаление не окажет заметного влияния на производительность модели. Далее к слою SC_k добавляется столько новых случайных связей, сколько ранее было удаленно. Таким образом, количество соединений в слое SC_k остается постоянным в процессе обучения. По окончании обучения сохраняется слой SC_k, как полученный на последнем шаге удаления весов, без добавления новых случайных связей.

2.2.4. Эволюционное обучение на основе градиента (RigL)

Прореживание нейронной сети методом RigL начинается с этапа проектирования модели (до обучения) [10]. Через определенные промежутки времени удаляется часть соединений в зависимости от их величины и активируются новые, в соответствии с критерием добавления. После добавления новых соединений обучение продолжается до следующего этапа обновления.

Основная идея критерия добавления новых связей заключается в учете влияния добавленных значений на градиент. То есть добавляются те связи – для которых градиент наибольший, которые могут потенциально оказать значительное влияние на функцию потерь.

3. Алгоритм прореживания нейронной сети в процессе обучения

В данной работе предложен алгоритм прореживания нейронных сетей в процессе обучения. Он основан на идее метода «Случайный выбор функции подмножества» [4], где решение задачи классификации строится в виде набора классификаторов, которые используют случайные подмножества признаков, при этом каждому признаку приписан показатель его качества (релевантности), определяемый точностью классификаторов, в которых он участвует.

Прореживание в рамках предложенного алгоритма выполняется в процессе обучения нейронной сети.

Пусть $r_j \in (0, +\infty)$ - значение релевантности параметра нейронной сети f_j из множества всех параметров сети F. Введем набор фиктивных параметров $z_j \in Z$ и соответствующие им значения релевантности $q_j \in (0, +\infty)$.

Во время каждой i–ой итерации алгоритма обучения нейронной сети выполняются следующие действия:

1) Осуществляется случайный выбор с равномерным распределением n параметров из множества F, которые формируют множество S_i.

2) Вычисляется значение функции потерь c_i для нейронной сети, у которой удалены все параметры кроме тех, которые входят в множество S_i.

3) Обновляются значения релевантности r_j для выбранных параметров $f_j \in$F:

$$\texttt{new\_r}_j = r_j + c_i - E(c), \tag{1}$$

здесь c_i – значение функции потерь на итерации i, $E(c)$ – математическое ожидание функции потерь (оценивается как среднее значение c_i на всех предыдущих итерациях.)

4) Обновляются значения релевантности q_j для фиктивных параметров z_j:

$$\texttt{new\_q}_j = q_j + c_i - E(c). \tag{2}$$

Обновление весов нейронной сети происходит в соответствие со стандартными алгоритмами обучения, при этом участвуют все обучаемые параметры.

Фиктивные параметры не используются в процессе обучения и вычислении функции потерь, но их релевантность накапливается, как и у истинных параметров.

Таким образом, релевантность q_j фиктивного параметра z_j становится случайным процессом, не соответствующим реальной точности нейронной сети. Релевантность фиктивных параметров обеспечивает базовый уровень r_{rand}, который должен быть превышен истинным параметром, чтобы он был признан значимым при минимизации функции потерь нейронной сети.

Наконец, чтобы найти набор параметров $S \subset F$ которые действительно превышают базовый уровень релевантности фиктивных параметров, проводится статистическая проверка. В частности, требуется, чтобы релевантность r_j параметра f_j удовлетворяла:

$$p(r_j > r_{rand}) \geq \delta, \forall f_j \in S_i, F, \tag{3}$$

здесь δ – фиксированный порог вероятности. Случайный базовый уровень r_{rand} моделируется как нормальное распределение фиктивных релевантностей q_j, таким образом, вероятность того, что функция более актуальна, чем фиктивная функция, получается из нормального распределения:

$$p(r_j > r_{rand}) = \frac{1}{\sigma\sqrt{2\pi}} \int_0^{r_j} e^{(\frac{-(x-\mu)^2}{2\sigma^2})} dx, \tag{4}$$

здесь μ и σ – среднее значение и стандартное отклонение релеватности q_j по всем фиктивным параметрам множества Z.

Псевдокод предложенного алгоритма прореживания приведен в Алгоритме 1.

Алгоритм 1 Алгоритм прореживания нейронной сети в процессе обучения

for *iter* **от** 1 **до** количествоИтерайцийОбучения **do**
  for k **от** 1 **до** количествоСлоевМодели **do**
    Исключаем часть множество фильтров слоя[k]
  end for
  Проводим итерацию обучения модели
  for k **от** 1 **до** количествоСлоевМодели **do**
    for j **от** 1 **до** количествоФиктивныхПараметровСлоя[k] **do**
      Обновляем значение релевантности $q[j]$ фиктивного параметра[j]
    end for
    for j **от** 1 **до** количествоФильтровСлоя[k] **do**
      Обновляем значение релевантности $r[j]$ фильтра[j]
      if ($iter$ *mod* количествоИтерайцийСбораСтатистики $=$ 0) **и** ($p(r[j] > r\_base) \leq \delta$) **then**
        Удаляем фильтр[j]
      end if
      Восстанавливаем исключенные фильтры, если они не удалены
    end for
  end for
end for

4. Обучение моделей нейронных сетей

В качестве моделей глубоких сверточных нейронных сетей для решения задачи классификации выбраны ResNet–18 и VGG–16. В качестве выборки объектов, для которой будет решаться задача классификации изображений используется датасет CIFAR–10 [13]. Обе нейронные сети обучены на тренировочной выборке, содержащей 45000 изображений, валидационная и тестовая выборка содержат соответственно 5000 и 10000 изображений.

Обучение производилось с помощью метода стохастического градиентного спуска на протяжении 70 эпох. В качестве функции потерь была использована функция кросс–энтропии:

$$Crossentropy = -\frac{1}{q}\sum_{i=1}^{q}\sum_{j=1}^{l} y_{ij}\log a_{ij},$$

здесь q – число параметров в выборке, l – число классов, a_{ij} – результат работы алгоритма, а именно вероятность принадлежности i–ого объекта к j–ому классу, $y_{ij} = 1$ если i–й объект принадлежит к j–ому классу, иначе $y_{ij} = 0$.

Соотношения объектов, принадлежащих различным классам, в обучающей, валидационной и тестовой одинаковое, поэтому в качестве оценки классификации будет использоваться:

$$accuracy = -\frac{1}{q}\sum_{i=1}^{q}(y_i = \overline{y_i}),$$

здесь q – число параметров в выборке, y_i – истинное значение класса i–ого объекта, $\overline{y_i}$ – предсказанное значение класса i–ого объекта.

На тестовой выборке точность модели нейронной сети VGG–16 составляет 91%, а модели нейронной сети ResNet–18 – 90.3%.

5. Результаты тестирования и сравнения

Для проведения экспериментального исследования были выбраны методы удаления фильтров из сверточных слоев нейронной сети на основе L_1 нормы и расстояния Евклида.

5.1. Удаление фильтров на основе L_1 нормы

Нормы фильтров в одном слое нейронной сети ResNet–18 имеют сильное отличие друг от друга (особенно для большого числа фильтров), и удаление фильтров с низкой нормой не должно существенно снизить точность работы модели. В нейронной сети VGG–16 нормы фильтров одного слоя имеют менее значительное отличие друг от друга, а значит потери в точности могут оказаться больше, чем при прореживании модели ResNet–18.

Будет проведено три независимых процесса прореживания. То есть для каждой из моделей нейронных сетей будет удален определенный процент фильтров: 20%/40%/60%. Затем будет проводиться дообучение прореженной модели на протяжении 2–ух эпох.

После удаления 20% фильтров из нейронной сети VGG–16 точность прореженной модели на тестовой выборке составила 90.3%. Таким образом, снижение точности составило всего 0.7%. При прореживании на 40% и 60% потери в точности составили соответственно 2.51% и 6.91%

Точность модели ResNet–18 после удаления 20% фильтров составила 89.7%. Таким образом, снижение точности составило всего 0.6% При прореживании на 40% и 60% потери в точности составили соответственно 1.07% и 1.93%

Таблица 1. Результаты прореживания на основе L_1 нормы.

Модель	Прореженность, %	Точность, %	Потери в точности, %
VGG–16	20	90.3	0.7
	40	88.49	2.51
	60	84.09	6.91
ResNet–18	20	89.7	0.6
	40	89.23	1.07
	60	88.37	1.93

Результаты работы метода прореживания сверточных слоев на основе L_1 нормы подтвердили предположение, что потери в точности при прореживании нейронной сети VGG–16 окажутся больше, чем при аналогичном прореживании модели ResNet–18.

5.2. Удаление фильтров на основе расстояния Евклида

Для нейронной сети ResNet–18 определить номера фильтров похожих в смысле расстояния Евклида легче, чем для модели VGG–16, в которой значения расстояния для разных фильтров более схожи между собой. Следовательно потери в точности при прореживании нейронной сети VGG–16 могут оказаться существеннее, чем для модели ResNet–18.

На каждом сверточном слое будем просматривать элементы матрицы расстояний, расположенные над главной диагональю, расстояние Евклида которых меньше некоторого порогового значения. Для каждого такого элемента в строке с номером i и столбце с номером j проверим, был ли удалён хотя бы один из фильтров с номером i или j, если нет, то из нейронной сети удаляется любой из фильтров с номером i или j, иначе осуществляется переход к следующему элементу

Процесс будет повторяться до тех пор, пока не будет удалено необходимое количество фильтров.

Как и в случае эксперимента по удалению фильтров на основе L_1 нормы, из каждой модели будет удален определенный процент фильтров: 20%/40%/60%. Затем будет проводиться дообучения полученной модели длиной в 2 эпохи.

После удаления 20% фильтров из нейронной сети VGG–16 точность прореженной модели на тестовой выборке составила 90.3%. Таким образом, снижение точности составило всего 0.7%. При прореживании на 40% и 60% потери в точности составили соответственно 3.47% и 8.6%

Точность модели ResNet–18 после удаления 20% фильтров составила 89.57%. Таким образом, снижение точности составило всего 0.63% При прореживании на 40% и 60% потери в точности составили соответственно 1.34% и 2.36%

Таблица 2. Результаты прореживания на основе расстояния Евклида.

Модель	Прореженность, %	Точность, %	Потери в точности, %
VGG–16	20	90.3	0.7
	40	87.53	3.47
	60	82.4	8.6
ResNet–18	20	89.67	0.63
	40	88.96	1.34
	60	87.94	2.36

Потери в точности при прореживании нейронной сети ResNet–18 оказались значительно ниже, чем при аналогичном прореживании модели VGG–16.

5.3. Удаление фильтров в процессе обучения

Для достижения необходимого процента прореженности сверточных нейронных сетей несколько раз проводилось исключение части фильтров из сверточных слоев в процессе обучения моделей.

Как и в предыдущих экспериментах по удалению фильтров на основе L_1 нормы и расстояния Евклида, процент исключенных из модели фильтров составит: 20%/40%/60%.

После удаления 20% фильтров из нейронной сети VGG–16 точность прореженной модели на тестовой выборке составила 90.4%. Таким образом, снижение точности составило всего 0.6%. При прореживании на 40% и 60% потери в точности составили соответственно 1% и 2.79%

Точность модели ResNet–18 после удаления 20% фильтров составила 89.72%. Таким образом, снижение точности составило всего 0.58% При прореживании на 40% и 60% потери в точности составили соответственно 0.95% и 1.67%

6. Заключение

В работе предложен новый алгоритм прореживания нейронный сетей в процессе их обучения. Он основан на оценке значимости обучаемых параметров сети по значению ее функции потерь. Определение параметров, которые необходимо удалить, выполняется по статистическому критерию.

Предложенный алгоритм был реализован и протестирован применительно к архитек-

Таблица 3. Результаты прореживания в процессе обучения.

Модель	Прореженность, %	Точность, %	Потери в точности, %
VGG–16	20	90.4	0.6
	40	90	1
	60	88.21	2.79
ResNet–18	20	89.72	0.58
	40	89.35	0.95
	60	88.63	1.67

турам нейронных сетей VGG–16 и ResNet–18 на задаче классификации изображений для базы CIFAR-10. Для проведения сравнения аналогичное тестирование было выполнено для алгоритма итеративного прореживания с применением критериев на основе нормы L_1 и Евклидова расстояния.

При прореженности нейронной сети VGG–16 в 20% метод прореживания в процессе обучения по точности классификации превосходит на 0.1% методы прореживания на основе L_1 нормы и расстояния Евклида, которые показывают одинаковые результаты. При прореженности в 40% метод прореживания в процессе обучения превосходит метод прореживания на основе L_1 нормы на 1.51%, а метод прореживания на основе расстояния Евклида на 2.47%. А при прореженности в 60% на 4.21% и 5.81% соответственно.

При прореженности нейронной сети ResNet–18 в 20% метод прореживания в процессе обучения по точности классификации превосходит на 0.02% метод прореживания на основе L_1 нормы и на 0.05% метод прореживания на основе расстояния Евклида на 2.47%. При прореженности в 40% точность метода прореживания в процессе обучения больше точности метода прореживания на основе L_1 нормы на 0.12%, а метод прореживания на основе расстояния Евклида на 0.39%. А при прореженности в 60% на 0.26% и 0.69% соответственно.

Предложенный метод прореживания сверточных нейронных сетей в процессе обучения продемонстрировал превосходство в итоговой точности классификации над остальными рассматриваемыми методами прореживания.

Программная реализация была выполнена на языке программирования Python3 с использованием фреймворка PyTorch. Обучение и тестирование нейронных сетей проводилось с применением устройств GPU. Исходные коды программных модулей необходимых для проведения экспериментов, в том числе реализация предложенного алгоритма прореживания, методов для сравнения, а также моделей нейронных сетей доступны на сервисе github [15].

Литература

1. Romero A. DALL·E 2, Explained: The Promise and Limitations of a Revolutionary AI

2. Han S., Pool J., Tran J., Dally W. Learning both Weights and Connections for Efficient Neural Network.

3. Yann Le Cun, John S. Denker and Sara A. Solla. Optimal Brain Damage

4. Jouni P., Okko R., November 2013. Feature Selection Methods and Their Combinations in High-Dimensional Classification of Speaker Likability, Intelligibility and Personality Traits

5. Wang H., Zhang Q., Wang Y., Yu L., Hu H. Structured Pruning for Efficient ConvNets via Incremental Regularization.

6. Guerra L., Zhuang B., Ian R.,Tom D. Automatic Pruning for Quantized Neural Networks.

7. Lee N., Ajanthan T., Philip H. SNIP: Single-shot Network Pruning based on Connection Sensitivity.

8. Chaoqi Wang, Guodong Zhang, Roger Grosse. Picking Winning Tickets Before Training by Preserving Gradient Flow.

9. Constantin M., Elena M., Peter S., Phuong H. Nguyen, Madeleine G., Antonio L. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science.

10. Utku E., Trevor G., Jacob M., Pablo Samuel C., Erich E. Rigging the Lottery: Making All Tickets Winners.

11. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition.

12. Karen S., Andrew Z. Very deep convolutional networks for large-scale image recognition.

13. Alex K. Learning Multiple Layers of Features from Tiny Images.

14. Torsten H., Dan A., Tal B–N, Nikoli D., Alexandra P. Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks.

15. https://github.com/dimchanov/Research-into-effectiveness-of-neural-networks-pruning-method-in-training-process

Методика анализа производительности вывода глубоких нейронных сетей на примере задачи классификации изображений

М.Р. Алибеков[1], Н.Е. Березина[2], И.Б. Вихрев[2], Е.П. Васильев[1], Ю.Д. Камелина[2], В.Д. Кустикова[1], З.А. Маслова[2], И.С. Мухин[1], А.К. Сидорова[1], В.Н. Сучков[1]

[1]Нижегородский государственный университет им. Н.И. Лобачевского,
[2] ООО «Ядро Центр Исследований и Разработки»

Внедрение глубоких нейронных сетей требует анализа производительности этапа вывода на целевом аппаратном обеспечении. Результаты производительности позволяют принимать решение о возможности внедрения построенных моделей и/или необходимости их предварительной оптимизации. В работе описана методика анализа и сравнения производительности вывода на примере решения задачи классификации изображений: конвертация обученной модели под разные фреймворки, анализ качества, определение оптимальных параметров запуска вывода, оптимизация модели и повторный анализ качества, анализ и сравнение производительности. Разработана система Deep Learning Inference Benchmark для поддержки цикла анализа производительности. Методика продемонстрирована на примере открытой модели MobileNetV2.

Ключевые слова: глубокое обучение, нейронные сети, вывод, производительность, MobileNetV2, Deep Learning Inference Benchmark.

1. Введение

В настоящее время применение методов ***глубокого обучения*** (deep learning) для решения прикладных задач является актуальным направлением исследований во многих областях. Разработано значительное количество архитектур глубоких нейронных сетей, которые позволяют с высокой точностью решать классические задачи компьютерного зрения и обработки естественного языка. Обученные модели находятся в открытом доступе в сети Интернет, они обучены или сконвертированы под разные фреймворки глубокого обучения [1, 2].

Типовой подход к применению накопленных знаний – ***перенос обучения*** (transfer learning). Перенос обучения предполагает использование части архитектуры открытой нейронной сети, относящейся к извлечению признаков (как правило, часть сети без последних полносвязных слоев), дополнение отобранной последовательности слоев необходимыми полносвязными слоями и последующее обучение модифицированной модели на данных, специфичных для рассматриваемой прикладной задачи. Цикл «модификация сети → обучение» повторяется до момента достижения удовлетворительного качества решения задачи. Далее разработанная модель внедряется в реальное приложение. ***Внедрение*** (deployment) предполагает многократный прямой проход по обученной модели – ***вывод*** (inference) – на вновь поступающих данных. На практике требуется, чтобы вывод выполнялся в режиме реального времени на целевом аппаратном обеспечении. Поэтому на этапе внедрения анализ и сравнение производительности вывода глубоких моделей является одной из наиболее важных задач.

Существует множество факторов, влияющих на производительность вывода. Сложность глубокой модели определяет количество вычислительных операций, выполняемых в процессе прямого прохода. Поэтому, когда получены низкие показатели производительности на целевом оборудовании, реализуется цикл «оптимизация/модификация модели → проверка качества → анализ производительности». Проверка качества модели – неотъемлемый этап, поскольку в процессе оптимизации качество может сильно изменяться. Выбор фреймворка для реализации вывода также оказывает значительное влияние на производительность. Наличие программных оптимизаций для некоторых вычислительных операций под конкретную аппаратную архитектуру

позволяет ускорить вывод и достичь необходимых для внедрения модели показателей производительности.

В настоящей работе предлагается методика анализа и сравнения производительности вывода глубоких нейросетевых моделей, применяемая в ходе их внедрения. Для поддержки процесса разрабатывается программный инструмент Deep Learning Inference Benchmark (DLI) с целью автоматической верификации корректности решения задачи и сбора показателей производительности вывода [3-6]. Отличительная особенность системы – расширяемость поддерживаемых фреймворков для вывода глубоких моделей, аппаратных архитектур и множества тестируемых моделей. Результаты производительности вывода для большого числа моделей на имеющемся аппаратном обеспечении публикуются в открытом доступе [7]. Инструмент DLI может использоваться сторонними разработчиками на этапе внедрения собственных моделей или в процессе их оптимизации. Также опубликованные результаты производительности позволяют оценить скорость вывода моделей на целевом оборудовании, если нейронная сеть спроектирована с использованием переноса обучения на базе какой-либо открытой модели.

Работа построена следующим образом. Вначале дается обзор подходов и инструментов к анализу производительности вывода глубоких нейронных сетей. Ставится задача классификации изображений. Приводится общая схема анализа и сравнения производительности. Далее дается информация о тестовых моделях и фреймворках, через которые запускается вывод. Описываются тестовые данные и используемая для экспериментов инфраструктура. Вводятся показатели качества классификации изображений и показатели производительности вывода. Методика анализа и сравнения производительности демонстрируется на примере решения задачи классификации изображений с использованием широко известной открытой модели MobileNetV2 [23].

2. Обзор литературы

Проблема анализа и повышения производительности глубоких нейронных сетей является актуальной на этапе их внедрения. Процедура оптимизации производительности вывода, как правило, сложная и трудоемкая, поскольку включает не только программную оптимизацию операций, но и сжатие сети, повторное обучение, анализ качества решения задачи и последующий анализ производительности вывода. Такой цикл может повторяться многократно до тех пор, пока на практике не будет найден компромисс между качеством и производительностью вывода модели. Существуют различные подходы к анализу качества и оценке производительности, разработаны программные инструменты, которые обеспечивают автоматизацию процесса на разных этапах внедрения.

Разработчики компании Intel в [8] предлагают инструмент Deep Learning Workbench [9], входящий в состав пакета Intel Distribution of OpenVINO Toolkit [16]. Инструмент обеспечивает поддержку всего цикла внедрения глубоких моделей. В состав входит множество открытых моделей, обученных для решения различных задач компьютерного зрения и обработки естественного языка; инструменты для конвертации обученных моделей во внутреннее представление OpenVINO, которое эффективно исполняется на аппаратных платформах Intel, инструменты для оптимизации моделей посредством понижения точности весов с FP32 и FP16 до INT8 – ***квантизации***; инструменты для определения качества и производительности вывода моделей.

В [10] приводится анализ качества для широко известных моделей глубокого обучения, развернутых на аналоговых устройствах. Аналоговое аппаратное обеспечение является перспективным для систем с ограниченным энергопотреблением. Аналоговый характер устройства и множество источников шума приводят к изменению весов в обученных моделях, которые развернуты в таких системах. Поэтому в данной работе исследуется устойчивость широко известных моделей при добавлении к весам сети аддитивного белого гауссовского шума.

Авторы [11] разрабатывают бенчмарк MLMark [12] для сравнения качества и производительности вывода на граничных устройствах (edge devices), анализируют результаты на нескольких ускорителях. Бенчмарк поставляется в открытых исходных кодах. Обеспечивается вывод для трех нейросетевых моделей средствами библиотек TensorFlow, TensorRT, ArmNN/TFL, Google TPU и OpenVINO. Сторонние исследователи могут самостоятельно запустить MLMark на собственных ускорителях для трех поддерживаемых моделей и опубликовать результаты на официальной странице проекта.

MLPerf Inference Benchmark [13, 14] – один из наиболее известных бенчмарков. Авторы пытаются эмулировать сценарии вывода, возникающие в реальных приложениях [13]. При этом в зависимости от сценария вводятся различные показатели производительности. Определен набор эталонных моделей, для которых публикуются результаты производительности в сети Интернет [14]. По аналогии с MLMark разработчики могут опубликовать полученные результаты на собственной аппаратуре на странице бенчмарка.

В [15] приводится анализ и сравнение производительности вывода, который реализован с использованием различных фреймворков, на Intel Cascade Lake CPUs. Поэтапно демонстрируется процедура для Intel Optimization of Caffe, TensorFlow, PyTorch, MXNet, Intel Distribution of OpenVINO Toolkit и OpenCV на примере двух глубоких моделей: подбор оптимальных параметров запуска вывода, анализ и сравнение масштабируемости, сравнение качества работы моделей, сравнение лучших показателей производительности вывода.

Настоящая работа – попытка обобщить и автоматизировать схему анализа и сравнения производительности, описанную в [15]. В отличие от [9] разрабатываемая система не ориентирована аппаратное обеспечение Intel. Имеющиеся реализации вывода можно запускать на различных ускорителях, в частности, NVIDIA GPU. В отличие от [11] и [13, 14] поддерживается вывод для значительного количества широко известных нейросетевых моделей, более того, в рамках системы можно запускать и анализировать производительность вывода собственных моделей. Также в рамках DLI обеспечивается регулярная публикация результатов производительности вывода на доступных аппаратных конфигурациях.

3. Постановка задачи классификации изображений

Задача классификации изображений с большим числом категорий состоит в том, чтобы поставить в соответствие изображению класс объектов, содержащихся на этом изображении. Как правило, на входе имеется трехканальное изображение в формате RGB, которое представляется трехмерным тензором I с пространственными размерами w и h, отвечающими ширине и высоте изображения, и глубины 3. Каждый элемент тензора содержит значение интенсивности соответствующего пикселя в диапазоне от 0 до 255 (или от 0 до 1, если выполнена нормировка). На выходе классификационной нейронной сети формируется вектор вещественных значений. Длина вектора соответствует количеству категорий изображений с учетом или без учета фона, а каждый элемент вектора отвечает достоверности принадлежности изображения определенному классу.

4. Общая схема анализа и сравнения производительности вывода

При выполнении анализа производительности вывода предполагается, что имеется глубокая нейросетевая модель, которая обучена средствами какого-либо фреймворка. Для данной модели достигнуто качество решения прикладной задачи, приемлемое для ее дальнейшего внедрения.

Общая схема анализа и сравнения производительности вывода в процессе внедрения глубоких нейронных сетей состоит из нескольких этапов.

1. Обучение разработанной архитектуры с использованием других фреймворков или конвертация обученной модели в форматы хранения сторонних фреймворков. Отметим, что при обучении необходимо воспроизвести условия и параметры тренировки исходной модели, чтобы получить близкие значения обученных параметров сети.

2. Анализ и сравнение качества полученного набора моделей. Необходимо гарантировать, что вновь обученная и/или сконвертированная модели обеспечивают качество решения прикладной задачи, сравнимое с исходной моделью.

3. Определение оптимальных параметров запуска вывода для каждого фреймворка. Анализ масштабируемости программных реализаций вывода в различных фреймворках, поиск оптимальной реализации вывода.

4. Сжатие и оптимизация разработанных моделей. В простейшем случае можно выполнить квантизацию весов модели (переход от весов в формате FP32 или FP16 к INT8) с использованием инструментов глубокого обучения. Более сложная оптимизация предполагает модификацию архитектуры сети и повторное обучение.

5. Анализ качества решения задачи с использованием оптимизированных моделей, сравнение показателей качества. Если достигнутые показатели качества не являются достаточными для внедрения, то переход к шагу 4.

6. Сравнение производительности вывода для набора обученных/сконвертированных и оптимизированных моделей. Если полученная производительность не является достаточной для внедрения, то возврат на шаг 4.

Далее демонстрируется реализация представленной схемы на примере задачи классификации изображений с большим числом категорий.

5. Вычислительные эксперименты

5.1 Программная система Deep Learning Inference Benchmark

DLI – программная система, разрабатываемая в ННГУ [3, 5]. Инструмент позволяет оценивать производительность вывода глубоких нейронных сетей, который реализован с использованием разных фреймворков, на доступном аппаратном обеспечении. В настоящее время имеются реализации вывода на Python для Intel Distribution of OpenVINO Toolkit [16], Intel Optimization for Caffe [17], Intel Optimizations for TensorFlow [18], TensorFlow Lite [19], MXNet [20], OpenCV [21] и реализации вывода на C++ для ONNX Runtime [22] и OpenCV. Архитектура системы предусматривает возможность одновременного проведения экспериментов на нескольких узлах с разным составом, находящихся в одной сети. При этом запуск возможен непосредственно на узле или внутри docker-контейнера, что упрощает подготовку окружения. Результаты качества и производительности вывода для большого числа глубоких моделей публикуются в открытом доступе [7].

5.2 Фреймворки для вывода глубоких моделей

В процессе анализа и сравнения производительности вывода используются следующие программные инструменты и библиотеки: Intel Distribution of OpenVINO Toolkit [16], TensorFlow [18], TensorFlow Lite [19], MXNet [20] и OpenCV [21]. Приведенный набор фреймворков выбран по следующим причинам.

1. OpenVINO – широко известный открытый инструмент для вывода глубоких сетей, оптимизированный под аппаратные архитектуры Intel (CPU, GPU, VPU или GNA). При этом вывод может выполняться как на отдельных устройствах, так и одновременно на нескольких устройствах, установленных на узле. Более того, OpenVINO содержит два режима вывода: режим максимизации времени выполнения одного запроса (latency mode) и режим максимизации пропускной способности (throughput mode), которые могут быть реализованы средствами синхронного (Sync API) и асинхронного (Async API) программных интерфейсов. Далее в работе используется latency-режим, реализованный с использованием синхронного интерфейса, что соответствует реализациям вывода во всех остальных фреймворках. Подробнее режимы вывода рассматриваются и анализируется в [5, 15].

2. TensorFlow – одна из популярных библиотек глубокого обучения, которая широко используется сообществом. Она обеспечивает обучение и тестирование глубоких моделей. Реализована поддержка символьных вычислений и автоматического дифференцирования.

3. TensorFlow Lite представляет собой реализацию TensorFlow, в которой вывод оптимизирован под мобильные архитектуры, что имеет особое значение на этапе внедрения глубоких нейросетевых моделей.

4. MXNet – менее известный фреймворк глубокого обучения. Разработчики обеспечивают 8 программных интерфейсов для обучения и тестирования нейронных сетей на разном аппаратном обеспечении. Эффективность вычислений в процессе вывода обеспечивается за счет применения символьных вычислений.

5. OpenCV – открытая библиотека алгоритмов компьютерного зрения, которая, в частности, содержит модуль для вывода нейронных сетей, обученных с помощью известных фреймворков.

Значительное количество систем видеонаблюдения разрабатывается на базе указанной библиотеки, а доступный функционал позволяет быстро создавать прототипы программных решений с использованием классических алгоритмов, а также глубоких нейросетевых моделей.

5.3 Тестовые модели

Из набора моделей, вывод которых поддерживается во всех перечисленных фреймворках, выбрана известная сверточная сеть MobileNetV2 [23]. На базе указанной классификационной модели строится большое количество глубоких сетей для детектирования объектов на изображениях, а также семантической сегментации изображений.

5.4 Тестовые данные

Для определения качества работы моделей используется валидационная выборка из набора данных ImageNET (50 000 изображений) [27], а для анализа производительности моделей – подмножество этой выборки, состоящее из 320 изображений. При многократном запуске вывода используется кольцевой алгоритм чтения изображений.

5.5 Тестовая инфраструктура

Эксперименты выполняются на тестовой инфраструктуре, параметры которой приведены в таблице 1. В общем случае, состав инфраструктуры обусловливается тем, на каком оборудовании предполагается запускать вывод глубоких моделей на этапе внедрения.

Таблица 1. Тестовая инфраструктура

CPU	Intel® Core™ i7-8700 3.20GHz (6 ядер и 12 потоков)
GPU	Intel® Gen9 HD Graphics (iGPU)
Оперативная память	64 ГБ
Операционная система	Ubuntu 20.04.4 LTS
Библиотеки	Intel Distribution of OpenVINO Toolkit 2022.3 TensorFlow 2.9.3 TensorFlow Lite 2.9.3 (part of TensorFlow) MXNet 1.9.1 OpenCV 4.7

5.6 Показатели качества классификации изображений

При оценивании качества классификации изображений рассматриваются метрики top-1 и top-5. Введем эти показатели. Допустим, имеется N категорий изображений. Для каждого изображения $I_j, j = \overline{1,S}$ в исходном наборе на выходе сети формируется вектор достоверностей $p^j = \left(p_1^j,\ p_2^j, \dots, p_N^j\right)$, где p_i^j – достоверность того, что изображение I_j принадлежит классу i.

Точность top-k (top-k accuracy) – отношение числа правильно проклассифицированных изображений к общему их количеству [5]. В случае top-1 изображение считается проклассифицированным правильно, если искомому классу соответствует максимальное значение достоверности, а в случае top-5 – если искомому классу соответствует одно из пяти наибольших значений достоверностей. В общем случае формула для вычисления top-k имеет вид:

$$topK = \frac{\sum_{j=1}^{S} 1_{\left\{i_1^j, i_2^j, \dots, i_K^j\right\}}\left(l_j\right)}{S},$$

где $\left\{i_1^j, i_2^j, \dots, i_K^j\right\} \subseteq \{1,2,\ \dots,\ N\}$ – подмножество допустимых классов, а $p_{i_1^j}^j, p_{i_2^j}^j, \dots, p_{i_K^j}^j$ – K наибольших достоверностей, l_j – класс, которому реально принадлежит изображение I_j (разметка), $1_{\left\{i_1^j, i_2^j, \dots, i_K^j\right\}}\left(l_j\right)$ – индикаторная функция, функция принимает значение 1, если искомый класс (разметка) содержится в подмножестве $\left\{i_1^j, i_2^j, \dots, i_K^j\right\}$.

5.7 Показатели производительности вывода

В процессе измерения показателей производительности вывода запросы на прямой проход по сети выполняются последовательно и многократно. Число повторений – ***итераций*** – параметр теста производительности. Далее указанный параметр во всех экспериментах устанавливается равным 1 000. Прямой проход предполагает решение задачи для подмножества входных данных – ***пачки***. Следующий запрос выполняется после завершения предыдущего. Для каждого запроса измеряется продолжительность его выполнения. Стандартное среднеквадратическое отклонение рассчитывается на основании набора полученных длительностей. Длительности, которые выходят за пределы трех стандартных отклонений относительно среднего времени вывода, отбрасываются. Результирующий набор времен используется для вычисления двух метрик производительности [6].

1. ***Латентность*** (Latency) – медиана времен выполнения запросов.
2. ***Среднее количество кадров, обрабатываемых за секунду*** (Frames per Second, FPS) – отношение размера пачки изображений к латентности.

Далее рассматриваются только показатели FPS, поскольку для каждого фиксированного размера пачки латентность можно вычислить на основании данной метрики. Отметим, что приведенный перечень показателей справедлив для всех фреймворков, включая latency-режим инструментария OpenVINO.

5.8 Результаты экспериментов

Обучение и/или конвертирование обученной модели. Согласно описанной схеме анализа производительности первый шаг – обучение и/или конвертация обученной модели в форматы, доступные для чтения в различных фреймворках. В качестве исходной модели используется MobileNetV2 в формате библиотеки TensorFlow, загруженная из Open Model Zoo [24]. Обученная модель конвертируется в промежуточное представление (intermediate representation, IR) с весами FP32 и FP16 для вывода средствами OpenVINO с помощью инструмента omz_converter, входящего в его состав. Модель с весами в формате FP16 получается посредством понижения точности обученных весов без дополнительной тонкой настройки модели на каких-либо сторонних данных. В процессе вывода MobileNetV2 через TensorFlow и OpenCV модель подается в исходном виде. Для преобразования модели в формат библиотеки TensorFlow Lite используется конвертер, доступный в разрабатываемой системе DLI. Для вывода средствами MXNet используется соответствующая модель, входящая в состав GluonCV Model Zoo [25]. Модель MobileNetV2 в форматах библиотек TensorFlow и MXNet обучена и провалидирована на наборе данных ImageNET [27] сторонними исследователями и выложена в открытый доступ. При этом условия и параметры обучения в обоих фреймворках соответствуют описанным в работе авторов модели [23].

В рамках имеющейся тестовой инфраструктуры вывод модели MobileNetV2 с весами FP32 для всех фреймворков доступен для запуска на CPU, для OpenVINO также имеется возможность запуска с весами FP32 и FP16 на интегрированной графической карте. Отметим, что для некоторых фреймворков имеется возможность конвертации модели в формат весов FP16, но в процессе вывода моделей с весами FP16 на CPU все операции выполняются с числами в формате FP32, поскольку отсутствует аппаратная поддержка операций в FP16, т.е. такой переход позволит получить модель меньшего размера с точки зрения объема памяти, необходимой для ее хранения, но не обеспечит выигрыша в производительности вывода.

Анализ и сравнение качества. После того, как сформирован набор обученных и/или сконвертированных моделей для запуска под разные фреймворки, необходимо провалидировать качество решения задачи классификации с помощью этих моделей. Ниже (рис. 1) приведены полученные значения показателей точности top-1 и top-5. Из построенной гистограммы можно видеть, что результаты качества работы моделей в разных фреймворках практически совпадают: наибольшее отличие в значении top-1 – 0.11%, top-5 – 0.33%. Незначительное отличие для моделей с весами в форматах FP32 и FP16 (третий результат по вертикали, рис. 1) объясняется более низкой точностью хранения весов модели и исполнения. Разница в качестве для модели в формате MXNet и других фреймворков является следствием того, что исследуемая модель в составе

GluonCV Model Zoo обучена с использованием MXNet, а не сконвертирована напрямую из формата TensorFlow, как для всех остальных запускаемых фреймворков. Вероятнее всего отличие обусловлено выбором начального приближения весов в процессе обучения модели в разных фреймворках с использованием метода обратного распространения ошибки.

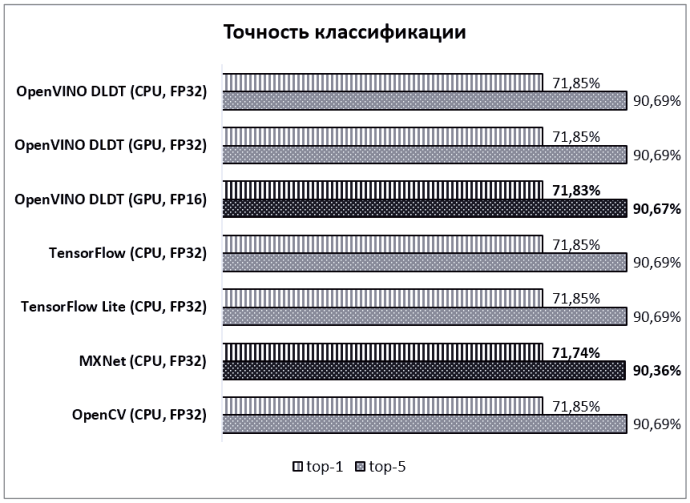


Рис. 1. Точность классификации изображений с использованием модели MobileNetV2 с весами в форматах FP32 и FP16

Определение оптимальных параметров запуска вывода, анализ масштабируемости, поиск оптимальной реализации вывода. Следующий шаг – подбор оптимальных параметров запуска вывода. В [15] описывается процедура подбора параметров запуска вывода (количество потоков и другие) для инструмента OpenVINO. Показано, что лучшие результаты производительности достигаются при установленных значениях по умолчанию. В [5] описывается подбор параметров запуска для Intel Optimizations for TensorFlow. Общая схема подбора параметров для всех фреймворков идентична, отличие состоит только в назначении и количестве допустимых параметров. Эксперименты, проводимые в настоящей работе, предполагают запуск вывода с параметрами по умолчанию, чтобы выровнять сложность этого этапа для всех фреймворков. Отметим, что вывод средствами MXNet запускается без подключения библиотеки oneDNN (ранее MKL-DNN) без (no hybrid) и с (hybrid) использованием символических вычислений. Запуск с oneDNN также требует подбора параметров, обеспечивающих эффективное исполнение операций, распараллеленных с помощью OpenMP. Оптимальные параметры для запуска вывода в значительной степени зависят от архитектуры нейронной сети.

К настоящему моменту можно гарантировать, что обученные и сконвертированные модели одинаково работают в разных фреймворках, а также определены параметры окружения для запуска вывода и дальнейшего анализа масштабируемости. Далее необходимо собрать результаты производительности при изменении допустимого размера пачки данных для каждого фреймворка. Ниже (рис. 2) показаны графики изменения FPS при переборе размеров пачки, составляющими степени двойки {1, 2, 4, 8, 16, 32, 64}. Для всех фреймворков наблюдается рост производительности до некоторого размера пачки (2 – для OpenVINO (CPU, FP32) и MXNet (CPU, FP32), 4 – для TensorFlow и OpenCV, 32 – для OpenVINO (GPU, FP32 и FP16)), далее происходит постепенное снижение показателя. Размер пачки, на котором достигается максимальная производительность вывода, во многом определяется размером доступного кэша. Использование символьных вычислений в реализации MXNet (MXNet (CPU, FP32, hybrid)) дает небольшой прирост производительности, но в целом показатели ниже, чем для других фреймворков. Следует отметить, что в MXNet все операции выполняются асинхронно, поэтому после каждого прямого прохода выполняется ожидание завершения вывода. Вероятнее всего, асинхронность позволит получить лучшие результаты в других сценариях исполнения, например, когда необходимо минимизировать общее время выполнения набора запросов. Тем не менее, все рассматриваемые фреймворки работают в режиме реального времени независимо от размера пачки входных данных. Отдельно следует выделить результаты, полученные при запуске FP16-модели с использованием OpenVINO на Intel GPU (OpenVINO DLDT (GPU, FP16)). Вывод на GPU при размерах пачки,

превышающих 8 изображений, работает быстрее реализации на TensorFlow, TensorFlow Lite, MXNet, OpenCV и других вариантов запуска OpenVINO при очень небольших потерях качества классификации. Приведенный факт говорит о перспективности использования интегрированных карт, поскольку это позволяет освободить CPU для решения более трудоемких задач.

В процессе внедрения выбор фреймворка во многом зависит от скорости вывода для фиксированного размера пачки данных на целевой аппаратуре (срез построенного графика по одному значению оси абсцисс). Размер пачки, как правило, определяется тем, с какой скоростью поступают данные с устройства, в частности, изображения с камеры для задач компьютерного зрения.

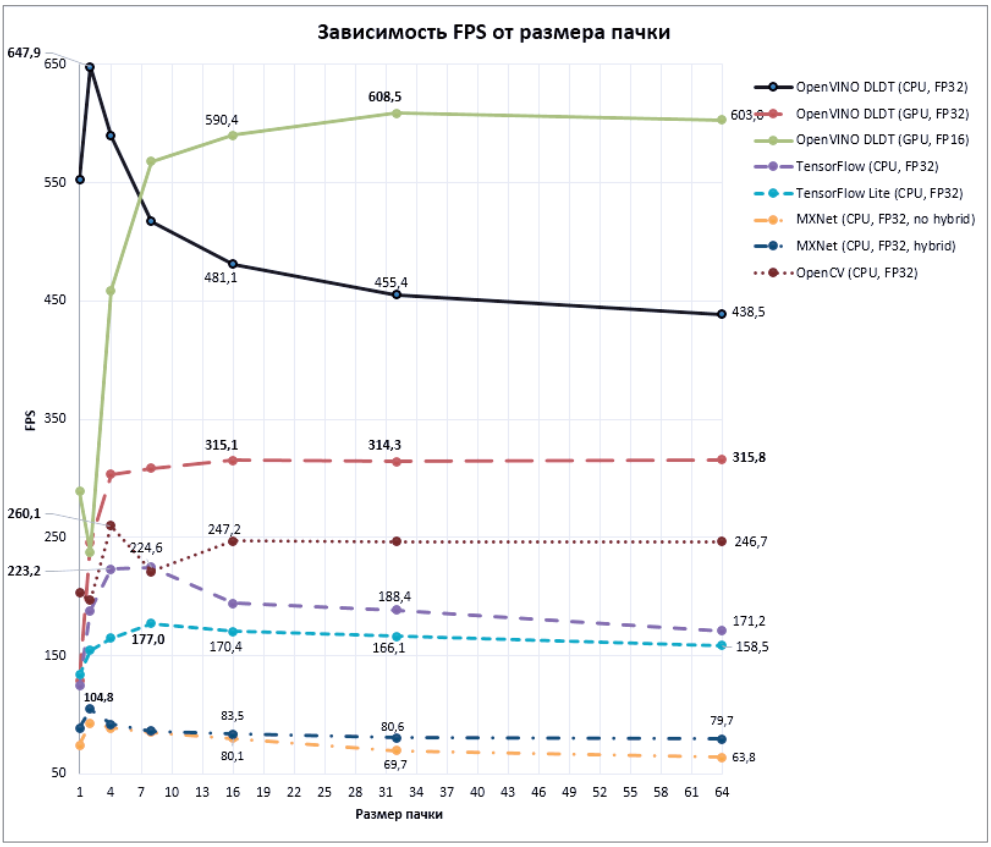


Рис. 2. Изменение показателя FPS при разных размерах входной пачки изображений для модели MobileNetV2 с весами в форматах FP32 и FP16

Сжатие и оптимизация моделей. На следующем этапе внедрения выполняется оптимизация построенных моделей. В простейшем случае – квантизация весов, т.е. переход от формата весов FP32 или FP16 к INT8 или UINT8. Квантизованные модели с весами INT8 для OpenVINO получены средствами Post-Training Optimization Tool, входящего в состав самого инструментария. Для фреймворка TensorFlow Lite соответствующая квантизованная модель с весами UINT8 находится в открытом доступе [26]. Отметим, что некоторые рассматриваемые фреймворки (TensorFlow и MXNet), также обеспечивают квантизацию моделей. На практике имеет смысл выполнять квантизацию и запуск экспериментов для всех доступных фреймворков, но цель настоящей работы – продемонстрировать методику анализа производительности, поэтому перебор всех возможных вариантов выходит за рамки исследования.

Анализ качества оптимизированных моделей. Для полученного набора оптимизированных моделей осуществляется анализ качества классификации. Ниже (рис. 3) приведены значения точности top-1 и top-5 для всего набора моделей. Из полученных результатов можно видеть, что квантизация приводит к снижению качества классификации менее, чем на 1% (метрика top-1 не более, чем на 0.91%, и top-5 не более, чем на 0.71%, при переходе от OpenVINO DLDT (CPU, FP32) к OpenVINO DLDT (CPU, INT8)). В целом полученные показатели сопоставимы.

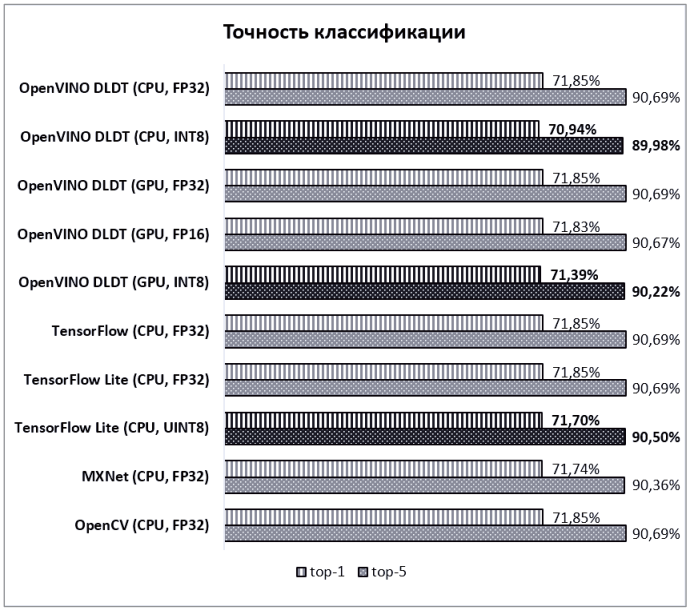


Рис. 3. Изменение точности классификации изображений с использованием модели MobileNetV2 при переходе от весов в форматах FP32 и FP16 к INT8 и UINT8

Сравнение производительности вывода для набора обученных/сконвертированных и оптимизированных моделей. Сравним производительность вывода исходных и квантизованных моделей для OpenVINO и TensorFlow Lite. Заметим, что общий тренд в изменении показателей FPS с ростом размера пачки данных сохраняется и для квантизованных моделей (рис. 4).

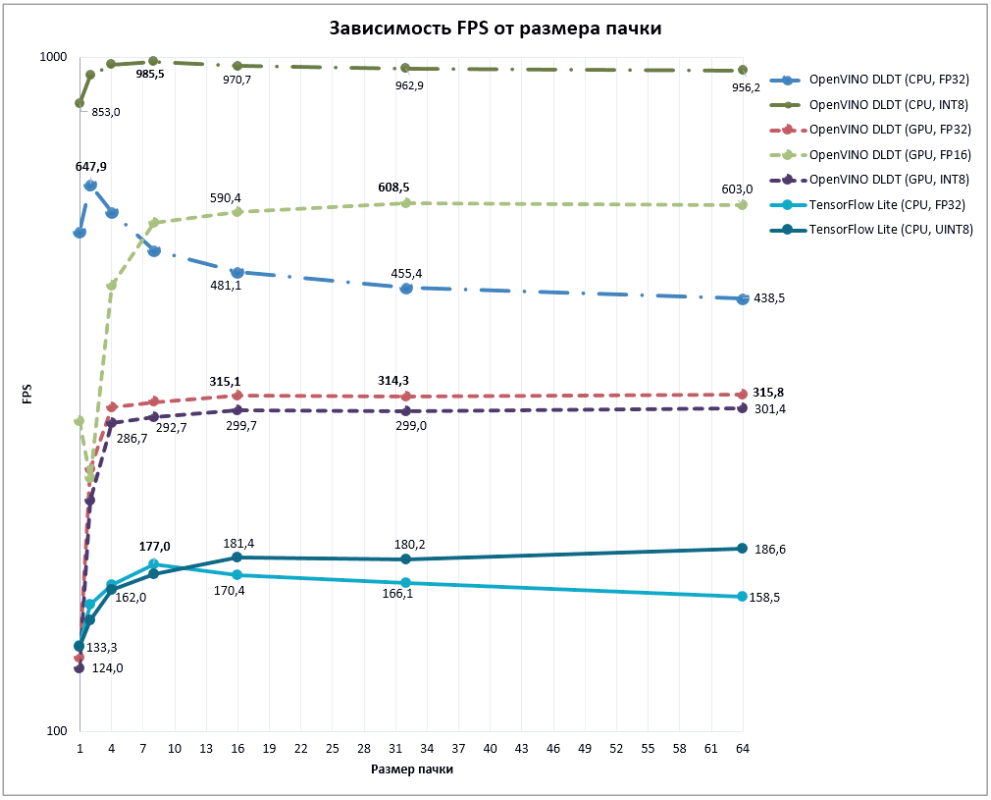


Рис. 4. Изменение показателя FPS при разных размерах входной пачки изображений для модели MobileNetV2 при переходе от весов в форматах FP32 и FP16 к INT8 и UINT8

При запуске вывода средствами OpenVINO на CPU переход к весам в формате INT8 дает выигрыш от 1.45 на пачке в 2 до 2.18 раза на пачке в 64 изображения, на GPU такой переход приводит к замедлению. Разработчики фреймворка отмечают, что исполнение квантизованных моделей на

GPU реализовано через DP4a, а поддержка DP4a появилась с 11-ого поколения Intel Core GPUs [28]. При запуске UINT8-модели через TensorFlow Lite для размера пачки, меньшей 16, наблюдается уменьшение показателя FPS, далее увеличение на 6-18% относительно модели с весами в формате FP32 в зависимости от размера пачки, выигрыш производительности менее заметный, чем для OpenVINO.

Обсудим лучшие показатели производительности, полученные в ходе экспериментов. На рисунке 5 приведена гистограмма с максимальными значениями FPS, достигнутыми для каждого фреймворка при разных вариантах запуска. Лучшие результаты наблюдаются при выводе MobileNetV2 с весами в формате INT8 с использованием библиотеки OpenVINO на размере тестовой пачки в 8 изображений. При незначительных потерях в качестве классификации вывод работает в ~1.52 раза быстрее по сравнению с аналогичным запуском модели с весами FP32 (бин OpenVINO DLDT (CPU, FP32, bs=2)). Второй лучший результат получен при запуске FP32-модели средствами OpenVINO на CPU (рис. 5, бин OpenVINO DLDT (CPU, FP32, bs=2)). Третий результат достигается при выводе FP16-модели на GPU (рис. 5, бин OpenVINO DLDT (GPU, FP16, bs=32)). Остальные реализации вывода работают значительно медленнее, тем не менее, все они решают задачу классификации в режиме реального времени. На основании приведенной гистограммы можно сделать выбор конфигурации запуска вывода в процессе внедрения. Если требуется достичь максимальной производительности, то следует запускать квантизованную модель средствами OpenVINO DLDT на CPU. Если необходимо освободить CPU для решения других задач, то вполне возможен запуск FP16-модели на Intel GPU.

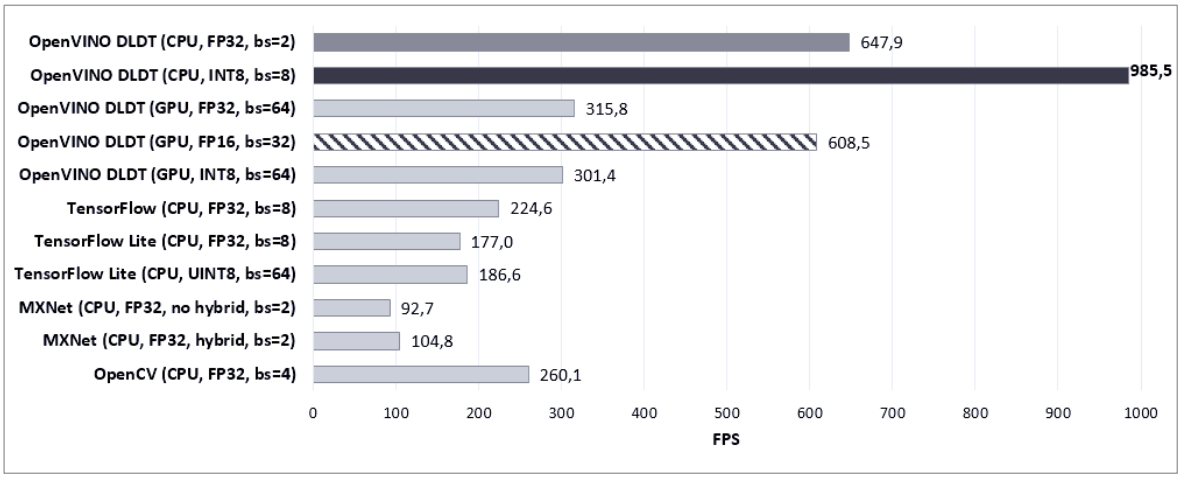


Рис. 5. Наиболее высокие показатели производительности вывода модели MobileNetV2, полученные при проведении экспериментов (в скобках указано устройство, формат весов модели и размер пачки входных данных bs, на которой получена наибольшая производительность)

В заключение следует отметить несколько важных моментов.

1. Тонкая настройка параметров запуска вывода возможно позволит улучшить полученные показатели производительности. Тонкая настройка параметров – тема отдельного исследования для каждого фреймворка и вида сети.

2. Производительность вывода средствами разных фреймворков может значительно варьироваться в зависимости от архитектуры (топологии) нейронной сети и целевой аппаратной конфигурации.

3. Конкретные реализации вывода в фреймворках могут быть оптимизированы под конкретную аппаратуру, в связи с чем на более слабом аппаратном обеспечении способны демонстрировать лучшие показатели производительности.

4. Использование асинхронного выполнения операций или запросов на вывод в некоторых фреймворках (например, OpenVINO) может приводить к более эффективному решению прикладной задачи с точки зрения общего времени работы приложения.

В связи с этим для разных глубоких моделей необходимо отдельно выполнять анализ и сравнение производительности в процессе их внедрения. Указанная процедура, как правило, требует не меньше времени, чем разработка новых архитектур нейросетей для решения прикладной задачи.

6. Заключение

Анализ производительности вывода глубоких нейронных сетей – важная проблема, возникающая на этапе их внедрения. В исследовании проработана методика анализа и сравнения производительности вывода. Разработана программная система Deep Learning Inference Benchmark для поддержки сбора результатов и автоматизации процесса. Система является расширяемой в плане набора поддерживаемых фреймворков для вывода, аппаратных архитектур и множества тестируемых моделей, что существенно отличает ее от аналогов. Практическое применение методики продемонстрировано на примере решения задачи классификации изображений с использованием модели MobileNetV2, вывод которой запускается через широко известные фреймворки и библиотеки (Intel Distribution of OpenVINO Toolkit, TensorFlow, TensorFlow Lite, MXNet, OpenCV). Проведенные эксперименты на доступной тестовой инфраструктуре (Intel CPU, Intel GPU) показывают, что вывод MobileNetV2 средствами всех сравниваемых фреймворков работает в режиме реального времени. Лучшие показатели производительности получены при выводе модели с весами INT8 средствами OpenVINO. Следует отметить эффективность квантизации модели при исполнении на Intel CPU через OpenVINO (дает выигрыш почти в 1.5 раза по сравнению с соответствующим запуском FP32-модели), а также перспективность использования Intel GPU для MobileNetV2, если необходимо освободить центральный процессор для других вычислений (модель с весами FP16 на GPU отстает от FP32 на CPU всего лишь на 6% на лучшем размере пачки данных).

Далее планируется решить ряд технических задач в рамках разрабатываемой системы DLI:

1. Обеспечение автоматического сбора результатов производительности и качества для доступных аппаратных конфигураций (использование CI-инструментов, в частности, Jenkins).
2. Расширение набора поддерживаемых фреймворков и конфигураций запуска вывода (интеграция PyTorch находится на стадии разработки).
3. Расширение множества тестируемых моделей.
4. Расширение множества аппаратных конфигураций, на которых выполняется сбор результатов производительности вывода.

Программная система выложена в открытый доступ, поэтому может быть использована разработчиками на этапе внедрения собственных глубоких нейросетевых моделей. Система открыта для модификации и расширения.

Дополнительно необходимо исследовать оптимальные параметры запуска для фреймворков TensorFlow, TensorFlow Lite, MXNet, OpenCV по аналогии, с тем, как это сделано в [15] для Intel Distribution of OpenVINO Toolkit, в [5] для Intel Optimization for Caffe и Intel Optimizations for TensorFlow. Наряду с этим, представляют интерес возможности асинхронного выполнения операций в MXNet и запуск вывода с использованием oneDNN.

Литература

1. Open Model Zoo for OpenVINO toolkit. URL: https://docs.openvino.ai/latest/model_zoo.html (дата обращения: 24.04.2023).
2. GluonCV Model Zoo. URL: https://cv.gluon.ai/model_zoo/index.html (дата обращения: 24.04.2023).
3. Deep Learning Inference Benchmark. URL: https://github.com/itlab-vision/dl-benchmark (дата обращения: 24.04.2023).
4. Kustikova V., Vasilyev E., Khvatov A., Kumbrasiev P., Rybkin R., Kogteva N. DLI: Deep Learning Inference Benchmark // Communications in Computer and Information Science. V.1129.2019. P. 542-553.
5. Сидорова А.К., Алибеков М.Р., Макаров А.А., Васильев Е.П., Кустикова В.Д. Автоматизация сбора показателей производительности вывода глубоких нейронных сетей в системе Deep Learning Inference Benchmark // Математическое моделирование и суперкомпьютерные технологии. Труды XXI Международной конференции (Н. Новгород, 22–26 ноября 2021 г.). 2021. 423 с.

6. DLI: Deep Learning Inference Benchmark. URL: https://hpc-education.unn.ru/dli-ru (In Russian), http://hpc-education.unn.ru/dli (In English) (дата обращения: 24.04.2023).
7. Deep Learning Inference Benchmark Wiki. URL: https://github.com/itlab-vision/dl-benchmark/wiki (дата обращения: 24.04.2023).
8. Demidovskij A., etc. OpenVINO Deep Learning Workbench: Comprehensive Analysis and Tuning of Neural Networks Inference // ICCV Workshop, 2019. URL: https://openaccess.thecvf.com/content_ICCVW_2019/papers/SDL-CV/Gorbachev_OpenVINO_Deep_Learning_Workbench_Comprehensive_Analysis_and_Tuning_of_Neural_ICCVW_2019_paper.pdf (дата обращения: 24.04.2023).
9. OpenVINO Deep Learning Workbench Overview. URL: https://docs.openvino.ai/latest/workbench_docs_Workbench_DG_Introduction.html (дата обращения: 24.04.2023).
10. Fagbohungbe O., Qian L. Benchmarking Inference Performance of Deep Learning Models on Analog Devices // 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China. 2021. P. 1-9.
11. Torelli P., Bangale M. Measuring Inference Performance of Machine-Learning Frameworks on Edgeclass Devices with the MLMark Benchmark, 2019. URL: https://www.eembc.org/techlit/articles/MLMARK-WHITEPAPER-FINAL-1.pdf, https://www.eembc.org/mlmark/scores.php (дата обращения: 24.04.2023).
12. EEMBC's Machine-Learning Inference Benchmark targeted at edge devices. URL: https://github.com/eembc/mlmark (дата обращения: 24.04.2023).
13. Reddi V.J., etc. MLPerf Inference Benchmark, 2019. URL: https://arxiv.org/abs/1911.02549 (дата обращения: 24.04.2023).
14. MLPerf Inference Benchmarks for Image Classification and Object Detection Tasks. URL: https://github.com/mlcommons/inference (дата обращения: 24.04.2023).
15. Vasiliev E.P., Kustikova V.D., Volokitin V.D., Kozinov E.A., Meyerov I.B. Performance Analysis of Deep Learning Inference in Convolutional Neural Networks on Intel Cascade Lake CPUs // Communications in computer and information science. 2021. V. 1413. P. 346-360.
16. Intel Distribution of OpenVINO Toolkit. URL: https://docs.openvino.ai/latest/home.html (дата обращения: 24.04.2023).
17. Intel Optimization for Caffe. URL: https://github.com/intel/caffe (дата обращения: 24.04.2023).
18. Intel Optimizations of TensorFlow. URL: https://pypi.org/project/intel-tensorflow (дата обращения: 24.04.2023).
19. TensorFlow Lite. URL: https://www.tensorflow.org/lite (дата обращения: 24.04.2023).
20. MXNet. URL: https://mxnet.apache.org (дата обращения: 24.04.2023).
21. OpenCV. URL: https://opencv.org (дата обращения: 24.04.2023).
22. ONNX Runtime. URL: https://onnxruntime.ai (дата обращения: 24.04.2023).
23. Sandler M., Howard A., Zhu M., Zhmoginov A., Chen L.-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks, 2018. URL: https://arxiv.org/abs/1801.04381 (дата обращения: 24.04.2023).
24. OpenVINO Toolkit – Open Model Zoo repository. URL: https://github.com/openvinotoolkit/open_model_zoo (дата обращения: 24.04.2023).
25. GluonCV Model Zoo. URL: https://cv.gluon.ai/model_zoo/index.html (дата обращения: 24.04.2023).
26. TensorFlow Hub. Mobilenet V2 trained on Imagenet. URL: https://tfhub.dev/iree/lite-model/mobilenet_v2_100_224/uint8/1 (дата обращения: 24.04.2023).

27. ImageNet. URL: https://www.image-net.org (дата обращения: 24.04.2023).
28. Intel Launches World’s Best Processor for Thin-and-Light Laptops: 11th Gen Intel Core. URL: https://www.intc.com/news-events/press-releases/detail/1411/intel-launches-worlds-best-processor-for-thin-and-light (дата обращения: 24.04.2023).

О новом подходе к решению задач линейного программирования на кластерных вычислительных системах*

Л.Б. Соколинский, И.М. Соколинская

Южно-Уральский государственный университет
(национальный исследовательский университет)

В статье представлен новый масштабируемый метод линейного программирования, получивший название «апекс-метод». Ключевой особенностью этого метода является построение пути, близкого к оптимальному, на поверхности допустимой области от определенной начальной точки до точного решения задачи линейного программирования. Под оптимальным путем понимается путь в направлении максимального увеличения значения целевой функции. Апекс-метод основан на схеме предиктор/корректор и состоит из двух стадий: Quest (предиктор) и Target (корректор). Получена аналитическая оценка ресурса параллелизма для этого алгоритма. Также приведен алгоритм, реализующий стадию Target, и доказана его сходимость. Описаны вычислительные эксперименты на кластерной вычислительной системе по применению апекс-метода для решения различных задач линейного программирования.

Ключевые слова: линейное программирование, апекс-метод, итерационный метод, метод проекционного типа, фейеровское отображение, параллельный алгоритм, оценка масштабируемости.

1. Введение

Важным классом приложений линейного программирования (ЛП) являются нестационарные задачи, связанные с оптимизацией нестационарных процессов [1]. В нестационарных задачах ЛП целевая функция и/или ограничения изменяются в течение вычислительного процесса. В качестве примеров можно привести следующие нестационарные задачи: поддержка принятия решений в высокочастотной торговле [2, 3], задачи гидрогазодинамики [4], оптимальное управление технологическими процессами [5–7], транспортные задачи [8–10], оперативное планирование [11, 12].

Один из стандартных подходов к решению нестационарных задач оптимизации состоит в том, чтобы рассматривать каждое изменение как появление новой задачи оптимизации, которую необходимо решать с нуля [1]. Однако такой подход часто непрактичен, поскольку решение проблемы с нуля без повторного использования информации из прошлого может занять слишком много времени. Таким образом, желательно иметь алгоритм оптимизации, способный непрерывно адаптировать решение к изменяющейся среде, повторно используя информацию, полученную в прошлом. Этот подход применим для процессов реального времени, если алгоритм достаточно быстро отслеживает траекторию движения оптимальной точки. В случае больших задач ЛП последнее требует разработки масштабируемых методов и параллельных алгоритмов ЛП.

Одним из наиболее перспективных подходов к решению сложных задач в режиме реального времени является использование нейросетевых моделей [13]. Самой популярной моделью нейронной сети является нейронная сеть прямого распространения. Обучение и использование таких сетей могут быть очень эффективно реализованы на графических процессорах [14]. Важным свойством нейронной сети прямого распространения является то, что время решения задачи не зависит от ее параметров. Это свойство необходимо для

*Исследование выполнено при финансовой поддержке РНФ (проект № 23-21-00356).

работы в режиме реального времени.

В недавней статье [15] была предложена n-мерная математическая модель визуализации задач ЛП. Эта модель позволяет использовать нейронные сети прямого распространения, включая сверточные сети [16], для решения многомерных задач ЛП, допустимой областью которых является замкнутое ограниченное непустое множество. Однако в научной литературе практически отсутствуют работы, посвященные использованию сверточных нейронных сетей для решения задач ЛП [17]. Причина в том, что сверточные нейронные сети ориентированы на обработку изображений, но до настоящего времени отсутствовали методы построения обучающих наборов данных, основанные на визуальном представлении многомерных задач ЛП.

В данной работе описывается новый масштабируемый итерационный метод для решения многомерных задач ЛП, получивший название «апекс-метод». Апекс-метод позволяет генерировать обучающие наборы данных для разработки нейронных сетей прямого распространения, способных находить решение нестационарной многомерной задачи ЛП на основе ее визуального представления в режиме реального времени. Статья организована следующим образом. Раздел 2 содержит теоретический базис, используемый в описании апекс-метода. В разделе 3 представлено формализованное описание апекс-метода. Раздел 3.1 посвящен разработке алгоритма построения псевдопроекции и аналитическому исследованию масштабируемости его параллельной версии. В разделе 3.2 описывается стадия Quest. Раздел 3.3 содержит описание стадии Target. В разделе 4 представлены информация о программной реализации апекс-метода и результаты вычислительных экспериментов. В заключении суммируются представленные в статье результаты и намечаются направления дальнейших исследований.

2. Теоретический базис

Данный раздел содержит необходимый теоретических базис, используемый для описания апекс-метода. Рассмотрим задачу ЛП в следующем виде:

$$\bar{x} = \arg\max_{x \in \mathbb{R}^n} \{\langle c, x\rangle \,|Ax \leqslant b\}\,, \tag{1}$$

где $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m\times n}$, $m > 1$, $c \neq \mathbf{0}$. Здесь $\langle\cdot,\cdot\rangle$ обозначает скалярное произведение двух векторов. Мы предполагаем, что ограничение

$$-x \leqslant \mathbf{0} \tag{2}$$

также включено в матричное неравенство $Ax \leqslant b$. Обозначим через $\mathcal{P}$ множество индексов, нумерующих строки матрицы A:

$$\mathcal{P} = \{1, \cdots, m\}\,. \tag{3}$$

Пусть $a_i \in \mathbb{R}^n$ обозначает вектор, представляющий i-тую строку матрицы A. Мы предполагаем, что $a_i \neq \mathbf{0}$ для всех $i \in \mathcal{P}$. Обозначим через $\hat{H}_i$ замкнутое полупространство, определяемое неравенством $\langle a_i, x\rangle \leqslant b_i$, а через H_i — его ограничивающую гиперплоскость:

$$\hat{H}_i = \{x \in \mathbb{R}^n | \langle a_i, x\rangle \leqslant b_i\}\,; \tag{4}$$

$$H_i = \{x \in \mathbb{R}^n | \langle a_i, x\rangle = b_i\}\,. \tag{5}$$

Определим допустимый многогранник

$$M = \bigcap_{i\in\mathcal{P}} \hat{H}_i, \tag{6}$$

представляющий множество допустимых точек задачи ЛП (1). Заметим, что M в этом случае будет замкнутым выпуклым множеством. Мы будем предполагать, что $M \neq \emptyset$, то есть задача ЛП (1) имеет решение.

Определение 1 *Полупространство $\hat{H}_i$ называется рецессивным, если*

$$\forall x \in \hat{H}_i, \exists \lambda \in \mathbb{R}_{>0} : x + \lambda c \notin \hat{H}_i. \tag{7}$$

Геометрический смысл этого определения состоит в том, что луч, исходящий из любой точки рецессивного полупространства в направлении вектора c, выходит за пределы этого полупространства. Легко показать, что следующее условие является необходимым и достаточным для того, чтобы полупространство $\hat{H}_i$ являлось рецессивным:

$$\langle a_i, c \rangle > 0. \tag{8}$$

Обозначим через $\pi_i(x)$ ортогональную проекцию точки $x \in \mathbb{R}^n$ на гиперплоскость H_i. Известно, что такая ортогональная проекция может быть вычислена по формуле

$$\pi_i(x) = x - \frac{\langle a_i, x \rangle - b_i}{\|a_i\|^2} a_i. \tag{9}$$

Здесь и далее $\|\cdot\|$ обозначает евклидову норму.

Определение 2 *Пусть $M \neq \emptyset$ — выпуклое замкнутое множество. Однозначное отображение $\varphi : \mathbb{R}^n \to \mathbb{R}^n$ называется M-фейеровским отображением [18], если*

$$\forall x \in M : \varphi(x) = x, \tag{10}$$

и

$$\forall x \notin M, \forall y \in \mathbb{R}^n : \|\varphi(x) - y\| < \|x - y\|. \tag{11}$$

Теорема 1 *Пусть $M \neq \emptyset$ — выпуклое замкнутое множество, $x^{(0)}$ — произвольная точка в $\mathbb{R}^n$. Если $\varphi(\cdot)$ является непрерывным M-фейеровским отображением, то последовательность*

$$\left\{ x^{(k)} = \varphi^k \left(x^{(0)} \right) \right\}_{k=1}^{\infty},$$

порождаемая этим отображением, сходится к точке, принадлежащей M:

$$x^{(k)} \to \tilde{x} \in M. \tag{12}$$

Доказательство. Сходимость непосредственно следует из теоремы 6.1 и следствия 6.1 в [18].

Теорема доказана.

Следующая теорема дает нам непрерывное M-фейеровское отображение, которое будет использоваться в апекс-методе.

Теорема 2 *Пусть $M \neq \emptyset$ — допустимый многогранник задачи ЛП (1):*

$$M = \bigcap_{i=1}^{m} \hat{H}_i. \tag{13}$$

Известно, что в этом случае M является выпуклым замкнутым множеством. Для произвольной точки $x \in \mathbb{R}^n$ определим множество индексов

$$\mathcal{J}_x = \{ i \,|\langle a_i, x \rangle > b_i; i \in \mathcal{P} \}. \tag{14}$$

Другими словами, $\mathcal{J}_x$ — множество индексов полупространств $\hat{H}_i$, которые не содержат точку x. Однозначное отображение $\psi : \mathbb{R}^n \to \mathbb{R}^n$, задаваемое формулой

$$\psi(x) = \begin{cases} x, & \text{если } x \in M; \\ \frac{1}{|\mathcal{J}_x|} \sum\limits_{i \in \mathcal{J}_x} \pi_i(x), & \text{если } x \notin M, \end{cases} \tag{15}$$

является непрерывным M-фейеровским отображением.

Доказательство. Очевидно, что отображение $\psi(\cdot)$ является непрерывным. Покажем, что выполняется условие (11). Доказательство проведем по общей схеме, представленной в [18]. Пусть $y \in M$ и $x \notin M$. Это означает, что

$$\mathcal{J}_x \neq \emptyset. \tag{16}$$

В силу (14) для всех $i \in \mathcal{J}_x$ справедливо неравенство

$$\|\pi_i(x) - x\| > 0. \tag{17}$$

Согласно лемме 3.2 в [18] для всех $i \in \mathcal{J}_x$ также выполняется следующее неравенство:

$$\|\pi_i(x) - y\|^2 \leqslant \|x - y\|^2 - \|\pi_i(x) - x\|^2. \tag{18}$$

Отсюда следует

$$\begin{aligned}
&\|y - \psi(x)\|^2 = \left\| y - \tfrac{1}{|\mathcal{J}_x|} \sum_{i \in \mathcal{J}_x} \pi_i(x) \right\|^2 = \left\| \tfrac{1}{|\mathcal{J}_x|} \sum_{i \in \mathcal{J}_x} (y - \pi_i(x)) \right\|^2 \leqslant \tfrac{1}{|\mathcal{J}_x|^2} \sum_{i \in \mathcal{J}_x} \|y - \pi_i(x)\|^2 \leqslant \\
&\leqslant \tfrac{1}{|\mathcal{J}_x|} \sum_{i \in \mathcal{J}_x} \|y - \pi_i(x)\|^2 \leqslant \tfrac{1}{|\mathcal{J}_x|} \sum_{i \in \mathcal{J}_x} \left(\|x - y\|^2 - \|\pi_i(x) - x\|^2 \right) \leqslant \\
&\leqslant \|x - y\|^2 - \tfrac{1}{|\mathcal{J}_x|} \sum_{i \in \mathcal{J}_x} \|\pi_i(x) - x\|^2.
\end{aligned}$$

В соответствии с (16) и (17) следующее неравенство имеет место:

$$\tfrac{1}{|\mathcal{J}_x|} \sum_{i \in \mathcal{J}_x} \|\pi_i(x) - x\|^2 > 0. \tag{19}$$

Отсюда

$$\forall x \notin M, \forall y \in \mathbb{R}^n : \|\psi(x) - y\| < \|x - y\|.$$

Теорема доказана.

Определение 3 *Пусть $M \neq \emptyset$ — допустимый многогранник задачи ЛП* (1), *$\psi(\cdot)$ — отображение, определяемое формулой* (15). Псевдопроекцией $\rho_M(x)$ *точки x на допустимый многогранник M называется предельная точка последовательности* $[x, \psi(x), \psi^2(x), \ldots, \psi^k(x), \ldots]$:

$$\lim_{k \to \infty} \left\| \rho_M(x) - \psi^k(x) \right\| = 0. \tag{20}$$

Корректность этого определения вытекает из теорем 1 и 2.

3. Описание апекс-метода

В этом разделе мы опишем новый масштабируемый итерационный метод решения задачи ЛП (1), получивший название «апекс-метод». Апекс-метод построен по схеме предиктор/корректор и включает в себя две последовательные стадии: Quest (предиктор) и Target (корректор). Стадия Quest находит грубое начальное приближение для задачи ЛП (1). Стадия Target уточняет это начальное приближение с определенной точностью. Основной операцией, используемой как на стадии Quest, так и на стадии Target, является операция вычисления псевдопроекции (см. определение 3). Следующий раздел посвящен описанию и исследованию алгоритма вычисления псевдопроекции.

Алгоритм 1 Последовательное вычисление псевдопроекции $\rho_M(x)$

Require: $\hat{H}_i = \{x \in \mathbb{R}^n | \langle a_i, x\rangle \leqslant b_i\}$, $M = \bigcap_{i=1}^m \hat{H}_i$, $M \neq \emptyset$

1: **function** $\rho_M(x)$
2: $k := 0$
3: $x^{(0)} := x$
4: **repeat**
5: $\mathcal{J} := \emptyset$
6: **for** $i = 1 \ldots m$ **do**
7: **if** $\left\langle a_i, x^{(k)} \right\rangle > b_i$ **then**
8: $\mathcal{J} := \mathcal{J} \cup \{i\}$
9: **end if**
10: **end for**
11: **if** $\mathcal{J} = \emptyset$ **then**
12: **return** $x^{(k)}$
13: **end if**
14: $S := 0$
15: **for all** $i \in \mathcal{J}$ **do**
16: $S := S + \left(\left\langle a_i, x^{(k)} \right\rangle - b_i\right) a_i / \left\|a_i\right\|^2$
17: **end for**
18: $x^{(k+1)} := x^{(k)} - S / \left|\mathcal{J}\right|$
19: $k := k + 1$
20: **until** $\left\|x^{(k)} - x^{(k-1)}\right\| < \epsilon$
21: **return** $x^{(k)}$
22: **end function**

3.1. Алгоритм вычисления псевдопроекции

Базовой операцией, используемой в апекс-методе, является операция псевдопроектирования, заключающаяся в последовательном применении отображения $\psi(\cdot)$, задаваемого формулой (15), к исходной точке. В данном разделе мы рассмотрим реализацию операции псевдопроектирования в виде последовательного и параллельного алгоритмов. Согласно определению 3 операция псевдопроектирования $\rho_M(\cdot)$ отображает произвольную точку $x \in \mathbb{R}^n$ в точку $\rho_M(x)$, принадлежащую допустимому многограннику M, представляющему допустимую область задачи ЛП (1). Вычисление $\rho_M(x)$ организуется в виде итерационного процесса с использованием формулы (15). Последовательная реализация этого процесса представлена в виде алгоритма 1. Кратко прокомментируем шаги этого алгоритма. Основной итерационный процесс, вычисляющий последовательность фейеровских приближений, представлен в виде цикла **repeat–until** (шаги 4–20). На шагах 5–10 строится множество $\mathcal{J}$, содержащее индексы полупространств $\hat{H}_i$, которым не принадлежит текущее приближение $x^{(k)}$. На шагах 14–18 вычисляется следующее приближение $x^{(k+1)}$ по формуле (15). Алгоритм завершает свою работу, когда расстояние между соседними приближения станет меньше малой положительной константы ϵ.

Известно, что в случае больших задач ЛП проекционный метод может потребовать значительных временных затрат [19]. Потому мы разработали параллельную версию алгоритма 1, представленную в виде алгоритма 2. Параллельный алгоритм построен на основе модели параллельных вычислений BSF [20], ориентированной на кластерные вычислительные системы. Модель BSF использует схему распараллеливания «мастер–рабочие» и требует представление алгоритма в виде операций над списками с использованием функций высшего порядка *Map* и *Reduce*. В качестве второго параметра функции высшего порядка

Алгоритм 2 Параллельное вычисление псевдопроекции $\rho_M(x)$

мастер	l**-тый рабочий** $(l = 0, \ldots, L-1)$
input $n, x^{(0)}$	**input** n, m, A, b, c
	$L :=$ NumberOfWorkers
$k := 0$	$\mathcal{L}_{map(l)} := [lm/L, \ldots, ((l+1)m/L) - 1]$
repeat	**repeat**
Bcast $x^{(k)}$	**RecvFromMaster** $x^{(k)}$
	$\mathcal{L}_{reduce(l)} := Map(\mathrm{F}_{x^{(k)}}, \mathcal{L}_{map(l)})$
	$(u_l, \sigma_l) := Reduce(\oplus, \mathcal{L}_{reduce(l)})$
Gather $\mathcal{L}_{reduce}$	**SendToMaster** (u_l, σ_l)
$(u, \sigma) := Reduce(\oplus, \mathcal{L}_{reduce})$	
$x^{(k+1)} := u/\sigma$	
$k := k + 1$	
$exit := \left\|x^{(k)} - x^{(k-1)}\right\| < \epsilon$	
Bcast *exit*	**RecvFromMaster** *exit*
until *exit*	**until** *exit*
output $x^{(k)}$	
stop	**stop**

Map в алгоритме 2 используется список $\mathcal{L}_{map} = [1, \ldots, m]$, содержащий порядковые номера ограничений задачи ЛП (1), а в качестве первого параметра фигурирует параметризованная функция

$$\mathrm{F}_x : \mathcal{P} \to \mathbb{R}^n \times \mathbb{Z}_{\geqslant 0},$$

определенная следующим образом:

$$\begin{gathered} \mathrm{F}_x(i) = (u_i, \sigma_i)\,; \\ u_i = \begin{cases} \pi_i(x), & \text{если } \langle a_i, x\rangle > b_i; \\ \mathbf{0}, & \text{если } \langle a_i, x\rangle \leqslant b_i; \end{cases} \\ \sigma_i = \begin{cases} 1, & \text{если } \langle a_i, x\rangle > b_i; \\ 0, & \text{если } \langle a_i, x\rangle \leqslant b_i. \end{cases} \end{gathered} \tag{21}$$

Таким образом, функция высшего порядка $Map\,(\mathrm{F}_x, \mathcal{L}_{map})$ преобразует список номеров ограничений $\mathcal{L}_{map}$ в список пар (u_i, σ_i):

$$Map\,(\mathrm{F}_x, \mathcal{L}_{map}) = [\mathrm{F}_x(1), \ldots, \mathrm{F}_x(m)] = [(u_1, \sigma_1), \ldots, (u_m, \sigma_m)]\,. \tag{22}$$

Здесь u_i является ортогональной проекцией точки x на гиперплоскость H_i в том случае, когда $x \notin \hat{H}_i$, и нулевым вектором в противном; σ_i соответственно принимает значение 1 или 0. Обозначим $\mathcal{L}_{reduce} = [(u_1, \sigma_1), \ldots, (u_m, \sigma_m)]$. Определим бинарную ассоциативную операцию

$$\oplus : \mathbb{R}^n \times \mathbb{Z}_{\geqslant 0} \to \mathbb{R}^n \times \mathbb{Z}_{\geqslant 0},$$

являющуюся первым параметром функции высшего порядка *Reduce*:

$$\left(u', \sigma'\right) \oplus \left(u'', \sigma''\right) = \left(u' + u'', \sigma' + \sigma''\right). \tag{23}$$

Функция высшего порядка $Reduce\,(\oplus, \mathcal{L}_{reduce})$ редуцирует список $\mathcal{L}_{reduce}$ к одной паре путем последовательного применения операции $\oplus$ ко всем элементам списка:

$$Reduce\,(\oplus, \mathcal{L}_{reduce}) = (u_1, \sigma_1) \oplus \ldots \oplus (u_m, \sigma_m) = (u, \sigma), \tag{24}$$

где

$$u = \sum_{i=1}^{m} u_i; \tag{25}$$

$$\sigma = \sum_{i=1}^{m} \sigma_i. \tag{26}$$

Параллельная работа алгоритма 2 организована по схеме «мастер–рабочие» и включает в себя $L+1$ процесс: один процесс–мастер и L процессов–рабочих. Процесс–мастер осуществляет общее управление вычислениями, распределяет работу между процессами–рабочими, получает от них результаты и формирует итоговый результат. Для простоты будем предполагать, что количество ограничений m в задаче ЛП (1) кратно количеству рабочих L. На шаге 1 мастер вводит исходные данные: размерность пространства n и начальную точку $x^{(0)}$. На шаге 3 мастер присваивает счетчику итераций k значение 0. Шаги 4–14 реализуют основной цикл **repeat–until**, вычисляющий псевдопроекцию. На шаге 5 мастер рассылает текущее приближение $x^{(k)}$ всем рабочим. На шаге 8 он получает от рабочих частичные результаты, которые на шаге 9 редуцируются в пару (u, σ). Последняя используется на шаге 10 для вычисления следующего приближения $x^{(k+1)}$. На шаге 11 мастер увеличивает на единицу счетчик итераций k. На шаге 12 мастер проверяет условие завершения и присваивает результат проверки логической переменной $exit$. На шаге 13 мастер рассылает всем рабочим значение логической переменной $exit$. Если логическая переменная $exit$ принимает значение «истина», цикл **repeat–until** завершается на шаге 14. На шаге 15 мастер выводит последнее приближение $x^{(k)}$ в качестве результата псевдопроекции. Шаг 16 завершает работу процесса–мастера.

Все рабочие выполняют один и тот же код, но над различными данными. На шаге 1 l-тый рабочий вводит исходные данные задачи ЛП. Затем он формирует подсписок своих номеров ограничений для обработки (шаги 2–3). Для удобства программирования нумерация ограничений начинается с нуля. Подсписки различных рабочих не пересекаются, и их объединение дает полный список номеров ограничений:

$$\mathcal{L}_{map} = \mathcal{L}_{map(0)} +\!\!+ \ldots +\!\!+ \mathcal{L}_{map(L-1)}. \tag{27}$$

Символ $+\!\!+$ здесь обозначает операцию конкатенации списков. Цикл **repeat–until** рабочего соответствует циклу **repeat–until** мастера (шаги 4–14). На шаге 5 рабочий получает от мастера текущее приближение $x^{(k)}$. На шаге 6 рабочий вызывает функцию высшего порядка Map, которая, в свою очередь, применяет параметризованную функцию $\mathrm{F}_{x^{(k)}}$, определенную по формуле (21), ко всем элементам подсписка $\mathcal{L}_{map(l)}$, формируя на выходе подсписок пар $\mathcal{L}_{reduce(l)}$. Этот подсписок на шаге 7 редуцируется рабочим в единственную пару (u_l, σ_l) с помощью функции высшего порядка $Reduce$, которая последовательно применяет бинарную операцию $\oplus$, определенную по формуле (23), ко всем элементам подсписка $\mathcal{L}_{reduce(l)}$. На шаге 13 рабочий получает от мастера значение логической переменной $exit$. Если эта переменная принимает значение «истина», то рабочий процесс завершается. В противном случае продолжает выполняться цикл **repeat–until**. Операторы обмена **Bcast**, **Gather**, **RecvFromMaster** и **SendToMaster** обеспечивают неявную синхронизацию работы процесса–мастера и процессов–рабочих.

Выполним оценку границы масштабируемости описанного параллельного алгоритма, используя стоимостную метрику модели BSF [20]. Под границей масштабируемости парал-

лельного алгоритма понимается максимальное число процессорных узлов, до которого наблюдается рост ускорения. Стоимостная метрика модели BSF включает в себя следующие параметры.

m : длина списка $\mathcal{L}_{map}$;
D : латентность (время, необходимое мастеру, чтобы послать одному рабочему сообщение длиной в один байт);
t_c : время, необходимое мастеру, чтобы переслать одному рабочему текущее приближение $x^{(k)}$ и получить от него пару (u_l, σ_l) с учетом латентности;
t_{Map} : время, требуемое одному рабочему, чтобы выполнить функцию высшего порядка Map для всех элементов списка $\mathcal{L}_{map}$;
t_a : время, необходимое для выполнения бинарной операции $\oplus$, определяемой по формуле (23).

Согласно формуле (14) из [20], граница масштабируемости L_{max} параллельного алгоритма 2 может быть оценена следующим образом:

$$L_{max} = \frac{1}{2}\sqrt{\left(\frac{t_c}{t_a \ln 2}\right)^2 + \frac{t_{Map}}{t_a} + 4m} - \frac{t_c}{t_a \ln 2}. \tag{28}$$

Вычислим временные параметры в формуле (28). Введем следующие обозначения для одной итерации цикла **repeat–until** (шаги 4–14 алгоритма 2):

c_c : количество чисел, пересылаемых от мастера рабочему и обратно в ходе одной итерации;
c_F : количество арифметических операций и операций сравнения, необходимых для вычисления функции F_x, определяемой по формуле (21);
$c_\oplus$: количество арифметических операций и операций сравнения, необходимых для выполнения бинарной операции $\oplus$, определяемой по формуле (23).

На шаге 5 мастер посылает l-тому рабочему вектор размерности n. Затем на шаге 8 мастер получает от l-того рабочего пару, состоящую из вектора размерности n и одного вещественного числа. Кроме этого, на шаге 13 мастер посылает l-тому рабочему одно логическое значение. Последняя пересылка состоит в пересылке одного бита и равносильна одному добавлению латентности D, что будет сделано позже. Следовательно,

$$c_c = 2n + 1. \tag{29}$$

Принимая во внимание формулы (9), (21) и предполагая, что значения $\|a_i\|^2$ для всех $i = 1, \ldots, m$ вычислены заранее, получаем

$$c_F = 3n + 2. \tag{30}$$

Исходя из (23), для $c_\oplus$ справедлива следующая формула:

$$c_\oplus = 2n + 1. \tag{31}$$

Обозначим через τ_{op} время выполнения одной арифметической операции или операции сравнения. Обозначим через τ_{tr} время пересылки одного вещественного числа без учета латентности. Тогда на основе (29), (30) и (31) имеем

$$t_c = c_c\tau_{tr} + 3D = (2n + 1)\tau_{tr} + 3D; \tag{32}$$
$$t_{Map} = c_F m\tau_{op} = (3n + 2)m\tau_{op}; \tag{33}$$
$$t_a = c_\oplus\tau_{op} = (2n + 1)\tau_{op}. \tag{34}$$

Подставляя правые части этих формул в (28) и добавляя латентность D, получаем

$$L_{max} = \frac{1}{2}\sqrt{\left(\frac{(2n+1)\tau_{tr}+3D}{(2n+1)\tau_{op}\ln 2}\right)^2 + \left(\frac{n+1}{2n+1}+5\right)m} - \frac{(2n+1)\tau_{tr}+3D}{(2n+1)\tau_{op}\ln 2},$$

где n — размерность пространства, m — количество ограничений. Для больших значений n и m отсюда вытекает следующая приближенная оценка:

$$L_{max} \approx O(\sqrt{m}). \tag{35}$$

Полученная оценка свидетельствует о том, что параллельный алгоритм 2 обладает слабой масштабируемостью[1].

3.2. Стадия Quest

Стадия Quest играет роль предиктора и состоит из следующих шагов.

1. Найти допустимую точку $\tilde{x} \in M$.
2. Вычислить точку апекса z.
3. Построить начальное приближение $u^{(0)}$, являющееся псевдопроекцией точки апекса z на допустимый многогранник M.

Допустимая точка $\tilde{x}$ на шаге 1 может быть вычислена с помощью формулы

$$\tilde{x} = \begin{cases} \mathbf{0}, & \text{если } \mathbf{0} \in M; \\ \rho_M(\mathbf{0}), & \text{если } \mathbf{0} \notin M, \end{cases} \tag{36}$$

где $\rho_M(\cdot)$ — операция псевдопроектирования на допустимый многогранник M (см. определение 3).

Точка апекса z на шаге 2 может быть вычислена следующим образом:

$$z = \tilde{x} + \left(\eta + \max\left\{\frac{b_i - \langle a_i, x'\rangle}{\langle a_i, e_c\rangle}\,\middle|\, i \in \mathcal{I}\right\}\right) e_c, \tag{37}$$

где $\mathcal{I}$ — множество индексов, для которых полупространство $\hat{H}_i$ является c-рецессивным; $\eta \in \mathbb{R}_{>0}$ — положительный параметр, определяющий удаление точки z от точки $\tilde{x}$. Подобный выбор точки апекса z основывается на эвристике, согласно которой псевдопроекция такой точки будет находиться «не очень далеко» от точного решения задачи ЛП. При этом значение параметра η может существенно влиять на близость точки $\rho_M(z)$ к точному решению. Оптимальное значение η может быть получено путем нахождения максимума целевой функции с использованием метода последовательной дихотомии.

На шаге 3 вычисляется точка $u^{(0)}$ по формуле

$$u^{(0)} = \rho_M(z). \tag{38}$$

Эта точка служит начальным приближением на стадии Target. Многочисленные вычислительные эксперименты, выполненные нами на искусственных и реальных невырожденных задачах ЛП, показывают, что итерационный процесс вычисления псевдопроекции, стартуя с произвольной внешней точки, всегда сходится к точке на границе допустимого многогранника M. Однако, в настоящий момент у нас отсутствует строгое доказательство этого факта.

[1] Если граница масштабируемости определяется формулой $L_{max} = O(m^{\alpha})$, то мы полагаем, что параллельный алгоритм обладает сильной масштабируемостью при $\alpha \geqslant 1$, слабой масштабируемостью при $0 < \alpha < 1$, и масштабируемость отсутствует при $\alpha \leqslant 0$.

Алгоритм 3 Стадия Target

Require: $\hat{H}_i = \{x \in \mathbb{R}^n | \langle a_i, x\rangle \leqslant b_i\}$, $M = \bigcap_{i=1}^m \hat{H}_i$, $M \neq \emptyset$

1: **input** $u^{(0)}$
2: $k := 0$
3: $v := u^{(k)} + \delta e_c$
4: $w := \rho_M(v)$
5: **while** $\left\langle c, w - u^{(k)}\right\rangle > \epsilon_f$ **do**
6: $\quad u := u^{(k)}$
7: $\quad d := w - u^{(k)}$
8: $\quad$ **while** $\|d\| > \epsilon_d$ **do**
9: $\quad\quad$ **if** $(u + d) \in M$ **then**
10: $\quad\quad\quad u := u + d$
11: $\quad\quad$ **else**
12: $\quad\quad\quad d := d/2$
13: $\quad\quad$ **end if**
14: $\quad$ **end while**
15: $\quad u^{(k+1)} := u$
16: $\quad k := k + 1$
17: $\quad v := u^{(k)} + \delta e_c$
18: $\quad w := \rho_M(v)$
19: **end while**
20: **output** $u^{(k)}$
21: **stop**

3.3. Стадия Target

Стадия Target играет в апекс-методе роль корректора и вычисляет последовательность точек

$$\left\{u^{(0)}, u^{(1)}, \ldots, u^{(k)}, \ldots\right\}, \tag{39}$$

обладающую следующими свойствами:

$$u^{(k)} \in \Gamma_M; \tag{40}$$

$$\left\langle c, u^{(k)}\right\rangle < \left\langle c, u^{(k+1)}\right\rangle; \tag{41}$$

$$\lim_{k\to\infty} \left\|u^{(k)} - \bar{x}\right\| = 0 \tag{42}$$

для всех $k \in \{0, 1, 2, \ldots\}$. Здесь Γ_M обозначает множество граничных точек допустимого многогранника M. Условие (40) означает, что все точки последовательности (39) лежат на границе допустимого многогранника M. Условие (41) говорит о том, что значение целевой функции в каждой точке последовательности (39) больше, чем в предыдущей. Согласно условию (42) последовательность (39) сходится к точному решению задачи ЛП (1).

Реализация стадии Target приведена в виде алгоритма 3. Дадим краткие комментарии по шагам алгоритма 3. На шаге 1 осуществляется ввод начального приближения $u^{(0)}$, полученного на стадии Quest. На шаге 2 счетчику итераций k присваивается значение 0. На шаге 3 вычисляется внешняя точка v как сумма векторов δe_c и $u^{(k)}$. Здесь e_c обозначает единичный вектор, сонаправленный с вектором c. Положительный параметр δ должен быть достаточно малым, чтобы отрезок $\left\{x \in \mathbb{R}^n \,\middle|\, x = (1-\lambda)w - \lambda u^{(k)}, 0 \leqslant \lambda \leqslant 1\right\}$ полностью лежал на границе допустимого многогранника M. На шаге 4 вычисляется точка w, являющаяся псевдопроекцией точки v на допустимый многогранник M. Шаги 5–19 реализуют

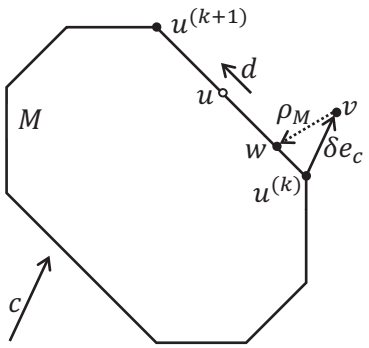


Рис. 1. Итерация основного цикла стадии Target.

основной цикл стадии Target, проиллюстрированный на рис. 1. Этот цикл выполняется, пока справедливо условие

$$\left\langle c, w - u^{(k)} \right\rangle > \epsilon_f. \tag{43}$$

Здесь ϵ_f — малый положительный параметр. На шаге 6 инициируется точка u, которая будет двигаться по поверхности допустимого многогранника M от точки $u^{(k)}$ до следующего приближения $u^{(k+1)}$. На шаге 7 вычисляется вектор d, задающий направление движения точки u. Цикл на шагах 8–14 перемещает точку u вдоль этого направления максимально далеко, но так, чтобы точка не покинула поверхность многогранника M. Это достигается делением вектора d пополам каждый раз, когда точка $(u + d)$ оказывается за пределами многогранника M. Перемещение завершается, когда длина вектора d становится меньше малого положительного параметра ϵ_d. На шаге 15 с помощью u определяется следующее приближение $u^{(k+1)}$. На шаге 16 счетчик итераций k увеличивается на 1. На шагах 17 и 18 вычисляются новая внешняя точка v и ее псевдопроекция w, используемые на следующей итерации основного цикла. После выхода из основного цикла на шаге 20 точка $u^{(k)}$ выводится в качестве приближенного решения задачи ЛП (1).

Следующая теорема гарантирует сходимость алгоритма 3.

Теорема 3 *Пусть допустимый многогранник M задачи ЛП* (1) *является замкнутым, выпуклым, непустым множеством. Тогда последовательность $\left\{u^{(k)}\right\}$, генерируемая алгоритмом 3, завершается через конечное число итераций $K \geqslant 0$ в некоторой допустимой точке, причем*

$$\left\langle c, u^{(0)} \right\rangle < \left\langle c, u^{(1)} \right\rangle < \left\langle c, u^{(2)} \right\rangle < \ldots < \left\langle c, u^{(K)} \right\rangle. \tag{44}$$

Доказательство. Случай $K = 0$ является тривиальным. Пусть $K > 0$, либо $K = \infty$. Сначала покажем, что для любого $k < K$ выполняется следующее неравенство:

$$\left\langle c, u^{(k)} \right\rangle < \left\langle c, u^{(k+1)} \right\rangle. \tag{45}$$

Действительно, из (43) следует, что

$$\left\langle c, u^{(k)} \right\rangle < \langle c, w \rangle. \tag{46}$$

Принимая во внимание шаг 7 алгоритма 3, это означает, что

$$d \neq \mathbf{0}. \tag{47}$$

Без потери общности мы можем считать, что $\left\| w - u^{(k)} \right\| > \epsilon_d$ (мы всегда можем добиться этого, уменьшая ϵ_d и/или увеличивая δ). Тогда в соответствии с шагами 8–15 мы получаем

$$u^{(k+1)} = u^{(k)} + \mu d, \tag{48}$$

где $\mu > 0$. Принимая во внимание неравенство (43) и шаг 7 алгоритма 3, отсюда следует

$$\left\langle c, u^{(k+1)} \right\rangle = \left\langle c, u^{(k)} + \mu d \right\rangle = \left\langle c, u^{(k)} + \mu \left(w - u^{(k)} \right) \right\rangle =$$
$$= \left\langle c, u^{(k)} \right\rangle + \mu \left\langle c, w - u^{(k)} \right\rangle > \left\langle c, u^{(k)} \right\rangle.$$

Теперь покажем, что $K < \infty$. Предположим противное, то есть алгоритм 3 генерирует бесконечную последовательность точек. В таком случае мы получаем бесконечную монотонно возрастающую числовую последовательность

$$\left\langle c, u^{(0)} \right\rangle < \left\langle c, u^{(1)} \right\rangle < \left\langle c, u^{(2)} \right\rangle < \dots \tag{49}$$

Поскольку допустимый многогранник M является ограниченным множеством, последовательность (49) ограничена сверху. Согласно теореме Вейерштрасса монотонно возрастающая ограниченная числовая последовательность имеет конечный предел, равный ее супремуму. Это означает, что существует $K' \in \mathbb{N}$ такой, что

$$\forall k > K' : \left\langle c, u^{(k+1)} \right\rangle - \left\langle c, u^{(k)} \right\rangle < \epsilon_d. \tag{50}$$

Отсюда следует

$$\forall k > K' : \langle c, w \rangle - \left\langle c, u^{(k)} \right\rangle < \epsilon_d, \tag{51}$$

что равносильно

$$\forall k > K' : \left\langle c, w - u^{(k)} \right\rangle < \epsilon_d. \tag{52}$$

Получили противоречие с критерием остановки (43), используемом на шаге 5 алгоритма 3.

Т е о р е м а д о к а з а н а.

Покажем, что последовательность $\left\{u^{(k)}\right\}$, генерируемая алгоритмом 3, сходится к точному решению задачи ЛП (1) при $\epsilon_f \to 0$. Для этого заметим, что при $\delta \to 0$ псевдопроекция сводится к метрической проекции, которая определяется следующим образом [18].

Определение 4 *Пусть Q является замкнутым выпуклым множеством в $\mathbb{R}^n$, и $Q \neq \emptyset$. Метрическая проекция $P_Q(x)$ точки $x \in \mathbb{R}^n$ на множество Q определяется формулой*

$$P_Q(x) = \arg\min \{ \|x - q\| \, | q \in Q \}. \tag{53}$$

Следующая теорема имеет место.

Теорема 4 *Последовательность $\left\{u^{(k)}\right\}$, генерируемая алгоритмом 3 с метрической проекцией $P_M(\cdot)$ вместо псевдопроекции $\rho_M(\cdot)$, завершается через конечное число итераций $K \geqslant 0$ в некоторой допустимой точке, причем*

$$\left\langle c, u^{(0)} \right\rangle < \left\langle c, u^{(1)} \right\rangle < \left\langle c, u^{(2)} \right\rangle < \dots < \left\langle c, u^{(K)} \right\rangle. \tag{54}$$

Д о к а з а т е л ь с т в о. Данная теорема доказывается по той же схеме, что и теорема 3.

Т е о р е м а д о к а з а н а.

Следующая теорема доказывает сходимость алгоритма 3 к точному решению задачи ЛП (1) для случая метрической проекции.

Теорема 5 *При замене псевдопроекции $\rho_M(\cdot)$ метрической проекцией $P_M(\cdot)$ алгоритм 3 завершается через конечное число итераций в точке $\bar{x}$, являющейся точным решением задачи ЛП* (1).

Доказательство. Обозначим через $\bar{u}$ конечную точку последовательности $\left\{u^{(k)}\right\}$, генерируемой алгоритмом 3 с использованием метрической проекции $P_M(\cdot)$. Такая точка существует в силу теоремы 4. Предположим противное, то есть $\bar{u} \neq \bar{x}$. Это равносильно

$$\langle c, \bar{u} \rangle < \langle c, \bar{x} \rangle . \tag{55}$$

Обозначим с помощью $S_\delta(v)$ открытый n-мерный шар радиуса δ с центром в точке v, где

$$v = \bar{u} + \delta e_c. \tag{56}$$

В силу (55) имеем

$$S_\delta(v) \cap M \neq \emptyset. \tag{57}$$

Положим

$$w = \arg\min \{ \|x - v\| \, | x \in S_\delta(v) \cap M \} . \tag{58}$$

Последнее эквивалентно

$$w = P_M(v). \tag{59}$$

Легко видеть, что справедливо неравенство

$$\langle c, w \rangle > \langle c, \bar{u} \rangle . \tag{60}$$

Тогда условия (56), (59) и (60) говорят, что $\bar{u}$ не является конечной точкой последовательности $\left\{u^{(k)}\right\}$, генерируемой алгоритмом 3. Получили противоречие.

Теорема доказана.

На практике заменить псевдопроекцию $\rho_M(v)$ в алгоритме 3 на метрическую проекцию $P_M(v)$ не представляется возможным, так как неизвестен алгоритм вычисления метрической проекции на выпуклый замкнутый многогранник в общем случае. Таким образом, теорема 5 в строгом смысле не доказывает сходимость алгоритма 3 к точному решению задачи ЛП (1), хотя на практике такая сходимость наблюдалась нами во всех случаях.

4. Программная реализация и вычислительные эксперименты

Мы реализовали параллельную версию апекс-метода на языке C++ с использованием программного BSF-каркаса [21], базирующегося на модели параллельных вычислений BSF [20]. BSF-каркас инкапсулирует все аспекты, связанные с распараллеливанием программы на основе библиотеки MPI. Исходные коды апекс-метода свободно доступны в репозитории GitHub по адресу `https://github.com/leonid-sokolinsky/Apex-method`. С помощью этой программы мы исследовали масштабируемость апекс-метода. Масштабные вычислительные эксперименты проводились на вычислительном кластере «Торнадо ЮУрГУ» [22], характеристики которого представлены в табл. 1. В качестве тестов мы использовали искус-

Таблица 1. Характеристики кластера «Торнадо ЮУрГУ»

Параметр	Значение
Количество процессорных узлов	480
Процессоры	Intel Xeon X5680 (6 cores, 3.33 GHz)
Число процессоров на узел	2
Память на узел	24 GB DDR3
Соединительная сеть	InfiniBand QDR (40 Gbit/s)
Операционная система	Linux CentOS

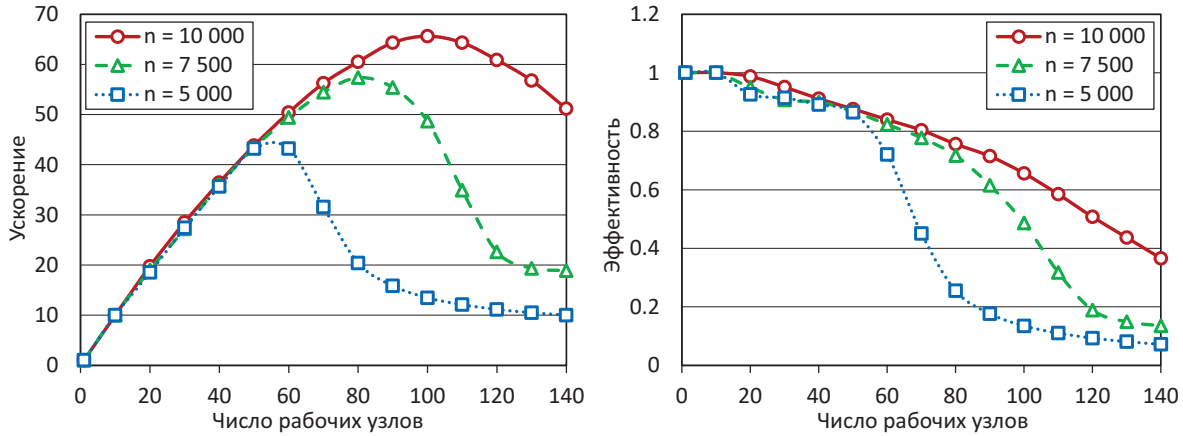


Рис. 2. Ускорение и параллельная эффективность апекс-метода.

ственные задачи, полученные с помощью генератора случайных задач линейного программирования FRaGenLP [23]. Верификация решений, выдаваемых апекс-методом, осуществлялась программой VaLiPro [24]. Была выполнена серия вычислительных экспериментов, в которой для задач ЛП различной размерности исследовались ускорение и параллельная эффективность в зависимости от количества используемых рабочих узлов. Результаты этих экспериментов представлены на рис. 2. В данном контексте ускорение $\alpha(L)$ определялось как отношение времени $T(1)$ решения задачи на конфигурации с узлом-мастером и единственным узлом-рабочим ко времени $T(L)$ решения той же задачи на конфигурации с узлом-мастером и L узлами-рабочими:

$$\alpha(L) = \frac{T(1)}{T(L)}. \tag{61}$$

Параллельная эффективность $\epsilon(L)$ вычислялась как отношение ускорения $\alpha(L)$ к числу L используемых узлов-рабочих:

$$\epsilon(L) = \frac{\alpha(L)}{L}. \tag{62}$$

Вычисления проводились для следующих размерностей: 5 000, 7 500 и 10 000. Число ограничений соответственно составило 10 002, 15 002 и 20 002.

Эксперименты показали, что граница масштабируемости параллельной реализации апекс-метода существенно зависит от размера задачи. Для $n = 5\,000$ граница масштабируемости составила приблизительно 55 рабочих узлов. Для задачи размерности $n = 7\,500$ эта граница увеличилась до 80 узлов, а для задачи размерности $n = 10\,000$ она оказалась близкой к 100 узлам. Дальнейшее увеличение размерности задачи приводило к ошибке компилятора «недостаточно памяти». Необходимо отметить, что вычисления проводились с двойной точностью, при которой число с плавающей точкой занимает в оперативной памяти 64 бита. Попытка использовать одинарную точность, требующую 32 бита для хранения числа с плавающей точкой, оказалась неудачной, так как при этом апекс-метод переставал сходиться к точному решению задачи ЛП.

Параллельная эффективность также продемонстрировала существенную зависимость от размера задачи ЛП. При $n = 5\,000$ эффективность показала падение ниже 50% уже на 70 рабочих узлах. Для $n = 7\,500$ и $n = 10\,000$ падение на 50% наблюдалось на 110 и 130 рабочих узлах соответственно.

Кроме этого, эксперименты показали, что параметр η в формуле (37), используемой на стадии Quest для вычисления точки апекса z, оказывает незначительное влияние на общее время решения задачи ЛП в случае, когда этот параметр принимает большие значения (более 100 000). Если точка апекса располагается недостаточно далеко от допустимого

Таблица 2. Применение апекс-метода для решения задач из Netlib-LP.

№	Задача из Netlib-LP		Стадия Quest		Стадия Target	
	Наименование	Точное решение	Грубое приближение	Ошибка	Уточненное приближение	Ошибка
1	adlittle	2.25494963E5	3.67140280E5	6.28E-1	2.2571324E5	9.68E-4
2	afiro	-4.64753142E2	-4.55961488E2	1.89E-2	-4.6475310E2	8.61E-9
3	blend	-3.08121498E1	-3.60232513E0	8.83E-1	-3.0811018E1	3.19E-5
4	fit1d	-9.14637809E3	-3.49931014E3	6.17E-1	-9.1463386E3	8.77E-7
5	kb2	-1.74990012E3	-1.39603193E3	2.02E-1	-1.6879152E3	3.54E-2
6	recipe	-2.66616000E2	-2.66107349E2	1.91E-3	-2.6660404E2	2.23E-5
7	sc50a	-6.45750770E1	-5.58016335E1	1.36E-1	-6.4568167E1	1.06E-4
8	sc50b	-7.00000000E1	-6.92167246E1	1.12E-2	-6.9990792E1	1.32E-4
9	sc105	-5.22020612E1	-4.28785710E1	1.79E-1	-5.1837995E1	6.97E-3
10	share2b	-4.15732240E2	-4.28792528E2	3.14E-2	-4.1572001E2	2.40E-5

многогранника, то ее псевдопроекция может оказаться на одной из его граней. Если же точка апекса располагается далеко от допустимого многогранника (в экспериментах использовалось значение $\eta = 20000n$), то ее псевдопроекция всегда оказывается в одной из его вершин. Также стоит отметить, что для искусственных задач, сгенерированных программой FRaGenLP, все точки последовательности $\{u^{(k)}\}$ оказывались на пути, близком к оптимальному[2].

Проведенные вычислительные эксперименты на искусственных задачах показали, что более 99% времени апекс-метод тратил на вычисление псевдопроекций (шаг 18 алгоритма 3). При этом вычисление одного приближения $u^{(k)}$ для задачи размерности $n = 10\,000$ на 100 рабочих узлах занимало 44 минуты.

Мы также протестировали апекс-метод на задачах из репозитория Netlib-LP [25], доступного по адресу `https://netlib.org/lp/data`. Набор задач линейной оптимизации Netlib-LP включает в себя множество реальных приложений, таких как оценка лесных ресурсов, задачи нефтепереработки, проектирование закрылков самолетов, модели пилотирования, планирование работы аудиторского персонала, расчет мостовых ферм, планирование расписаний авиакомпаний, расчет моделей промышленного производства и распределения ресурсов, восстановление изображений и задачи многосекторального экономического планирования. Netlib-LP содержит задачи ЛП размером от 32 переменных и 27 ограничений до 15 695 переменных и 16 675 ограничений [26]. Точные решения (оптимальные значения целевых функций) для всех задач были заимствованы из работы [27]. Результаты представлены в табл. 2. Эксперименты показали, что относительная ошибка грубого приближения, вычисляемого на стадии Quest, не превосходила 0.2 для всех задач, кроме *adlittle*, *blend*, и *fit1d*. Относительная ошибка уточненного приближения, получаемого на стадии Target, оказалась менее 10^{-3}, за исключением задач *kb2* и *sc105*, для которых ошибка составила 0.035

[2]Под оптимальным путем понимается путь движения по поверхности допустимого многогранника в направлении максимального увеличения значения целевой функции.

и 0.007 соответственно. Время решения указанных задач варьировалось от нескольких секунд для *afiro* до десятков часов для *blend*. Одним из главных параметров, влияющих на скорость сходимости апекс-метода, был параметр ϵ, используемый на шаге 12 параллельного алгоритма 2, вычисляющего псевдопроекцию. Прогоны всех задач доступны на GitHub по адресу `https://github.com/leonid-sokolinsky/Apex-method/tree/master/Runs`.

Заключение

В статье предложен новый масштабируемый итерационный метод линейного программирования, получивший название «апекс-метод». Ключевой особенностью этого метода является построение оптимального пути на поверхности допустимого многогранника от начальной точки к решению задачи линейного программирования. Под оптимальным путем понимается кратчайший путь в смысле евклидовой метрики. Практическая значимость предложенного метода состоит в том, что он открывает возможность применения искусственных нейронных сетей прямого распространения для решения нестационарных многомерных задач линейного программирования в режиме реального времени.

Для параллельного алгоритма построения псевдопроекции получена аналитическая оценка границы его масштабируемости на кластерной вычислительной системе. Эта граница не превышает $O\left(\sqrt{m}\right)$ процессорных узлов, где m — количество ограничений задачи линейного программирования. Описан алгоритм, строящий на границе допустимого многогранника оптимальный путь от начального приближения до точки решения задачи линейного программирования. Доказана сходимость этого алгоритма.

Параллельная версия апекс-метода реализована на языке C++ с использованием программного BSF-каркаса, основанного на модели параллельных вычислений BSF. Проведены эксперименты по исследованию масштабируемости апекс-метода на кластерной вычислительной системе. Вычислительные эксперименты показали, что для задачи линейного программирования с 10 000 переменными и 20 002 ограничениями граница масштабируемости не превышает 100 процессорных узлов. В то же время эксперименты показали, что более 99% времени, затрачиваемого на решение задачи линейного программирования апекс-методом, приходилось на вычисление псевдопроекций.

В дополнение, апекс-метод был протестирован на 10 задачах из репозитория Netlib-LP. Относительная ошибка на этих задачах составила от $3.5{\cdot}10^{-3}$ до $8.6{\cdot}10^{-9}$. Время вычислений варьировалось от нескольких секунд до нескольких десятков часов. Точность вычисления псевдопроекции оказалась основным параметром, влияющим на скорость сходимости апекс-метода.

В качестве направлений дальнейших исследований выделим следующие. Мы планируем разработать новый, более эффективный метод вычисления псевдопроекций на допустимый многогранник. Основная идея состоит в сокращении количества полупространств, используемых в рамках одной итерации. В то же время количество оставшихся в рассмотрении полупространств должно быть достаточным для эффективного распараллеливания. Мы рассчитываем, что новый метод превзойдет алгоритм 2 по скорости сходимости. Также мы собираемся доказать, что новый метод сходится к точке, лежащей на границе допустимого многогранника. Кроме этого мы планируем исследовать полезность использования в апекс-методе техники линейной супериоризации, предложенной в работе [28].

Литература

1. Branke J. Optimization in Dynamic Environments // Evolutionary Optimization in Dynamic Environments. Genetic Algorithms and Evolutionary Computation, vol. 3. Boston, MA: Springer, 2002. P. 13–29. DOI: 10.1007/978-1-4615-0911-0_2.

2. Brogaard J., Hendershott T., Riordan R. High-Frequency Trading and Price Discovery // Review of Financial Studies. 2014. Vol. 27, no. 8. P. 2267–2306. DOI: 10.1093/rfs/hhu032.

3. Deng S., Huang X., Wang J., *et al.* A Decision Support System for Trading in Apple Futures Market Using Predictions Fusion // IEEE Access. 2021. Vol. 9. P. 1271–1285. DOI: 10.1109/ACCESS.2020.3047138.

4. Seregin G. Lecture notes on regularity theory for the Navier-Stokes equations. Singapore: World Scientific Publishing Company, 2014. 268 p. DOI: 10.1142/9314.

5. Demin D.A. Synthesis of optimal control of technological processes based on a multialternative parametric description of the final state // Eastern-European Journal of Enterprise Technologies. 2017. Vol. 3, 4(87). P. 51–63. DOI: 10.15587/1729-4061.2017.105294.

6. Kazarinov L.S., Shnayder D.A., Kolesnikova O.V. Heat load control in steam boilers // 2017 International Conference on Industrial Engineering, Applications and Manufacturing, ICIEAM 2017 - Proceedings. IEEE, 2017. DOI: 10.1109/ICIEAM.2017.8076177.

7. Zagoskina E.V., Barbasova T.A., Shnaider D.A. Intelligent Control System of Blast-furnace Melting Efficiency // SIBIRCON 2019 - International Multi-Conference on Engineering, Computer and Information Sciences, Proceedings. IEEE, 2019. P. 710–713. DOI: 10.1109/SIBIRCON48586.2019.8958221.

8. Fleming J., Yan X., Allison C., *et al.* Real-time predictive eco-driving assistance considering road geometry and long-range radar measurements // IET Intelligent Transport Systems. 2021. Vol. 15, no. 4. P. 573–583. DOI: 10.1049/ITR2.12047.

9. Scholl M., Minnerup K., Reiter C., *et al.* Optimization of a thermal management system for battery electric vehicles // 14th International Conference on Ecological Vehicles and Renewable Energies, EVER 2019. IEEE, 2019. DOI: 10.1109/EVER.2019.8813657.

10. Meisel S. Dynamic Vehicle Routing // Anticipatory Optimization for Dynamic Decision Making. Operations Research/Computer Science Interfaces Series, vol. 51. New York, NY: Springer, 2011. P. 77–96. DOI: 10.1007/978-1-4614-0505-4_6.

11. Cheng A.M.K. Real-Time Scheduling and Schedulability Analysis // Real-Time Systems: Scheduling, Analysis, and Verification. John Wiley, Sons, 2002. P. 41–85. DOI: 10.1002/0471224626.CH3.

12. Kopetz H. Real-Time Scheduling // Real-Time Systems. Real-Time Systems Series. Boston, MA: Springer, 2011. P. 239–258. DOI: 10.1007/978-1-4419-8237-7_10.

13. Prieto A., Prieto B., Ortigosa E.M., *et al.* Neural networks: An overview of early research, current frameworks and new challenges // Neurocomputing. 2016. Vol. 214. P. 242–268. DOI: 10.1016/j.neucom.2016.06.014.

14. Raina R., Madhavan A., Ng A.Y. Large-scale deep unsupervised learning using graphics processors // Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09). New York, NY, USA: ACM Press, 2009. P. 873–880. DOI: 10.1145/1553374.1553486.

15. Ольховский Н.А., Соколинский Л.Б. Визуальное представление многомерных задач линейного программирования // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2022. Т. 11, № 1. С. 31–56. DOI: 10.14529/cmse220103.

16. LeCun Y., Bengio Y., Hinton G. Deep learning // Nature. 2015. Vol. 521, no. 7553. P. 436–444. DOI: 10.1038/nature14539.

17. Lachhwani K. Application of Neural Network Models for Mathematical Programming Problems: A State of Art Review // Archives of Computational Methods in Engineering. 2020. Vol. 27. P. 171–182. DOI: 10.1007/s11831-018-09309-5.

18. Васин В.В., Ерёмин И.И. Операторы и итерационные процессы фейеровского типа. Теория и приложения. Екатеринбург: УрО РАН, 2005. 211 с.

19. Gould N.I. How good are projection methods for convex feasibility problems? // Computational Optimization and Applications. 2008. Vol. 40, no. 1. P. 1–12. DOI: 10.1007/S10589-007-9073-5.

20. Sokolinsky L.B. BSF: A parallel computation model for scalability estimation of iterative numerical algorithms on cluster computing systems // Journal of Parallel and Distributed Computing. 2021. Vol. 149. P. 193–206. DOI: 10.1016/j.jpdc.2020.12.009.

21. Sokolinsky L.B. BSF-skeleton: A Template for Parallelization of Iterative Numerical Algorithms on Cluster Computing Systems // MethodsX. 2021. Vol. 8. Article number 101437. DOI: 10.1016/j.mex.2021.101437.

22. Dolganina N., Ivanova E., Bilenko R., Rekachinsky A. HPC Resources of South Ural State University // Parallel Computational Technologies. PCT 2022. Communications in Computer and Information Science, vol. 1618 / ed. by L. Sokolinsky, M. Zymbler. Cham: Springer, 2022. P. 43–55. DOI: 10.1007/978-3-031-11623-0_4.

23. Соколинский Л.Б., Соколинская И.М. О генерации случайных задач линейного программирования на кластерных вычислительных системах // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 2. С. 38–52. DOI: 10.14529/cmse210103.

24. Соколинский Л.Б., Соколинская И.М. О валидации решений задач линейного программирования на кластерных вычислительных системах // Вычислительные методы и программирование. 2021. Т. 22, № 4. С. 252–261. DOI: 10.26089/NUMMET.V22R416.

25. Gay D.M. Electronic mail distribution of linear programming test problems // Mathematical Programming Society COAL Bulletin. 1985. Vol. 13. P. 10–12.

26. Keil C., Jansson C. Computational experience with rigorous error bounds for the netlib linear programming library // Reliable Computing. 2006. Vol. 12, no. 4. P. 303–321. DOI: 10.1007/S11155-006-9004-7/METRICS.

27. Koch T. The final NETLIB-LP results // Operations Research Letters. 2004. Vol. 32, no. 2. P. 138–142. DOI: 10.1016/S0167-6377(03)00094-4.

28. Censor Y. Can linear superiorization be useful for linear optimization problems? // Inverse Problems. 2017. Vol. 33, no. 4. P. 044006. DOI: 10.1088/1361-6420/33/4/044006.

Об истоках создания первого института прикладной математики и основ "цифровой цивилизации". Посвящается памяти первого директора М.В.Келдыша и его заместителя А.Н.Тихонова в год 70-летия "Института Келдыша" АН СССР

Т.А. Сушкевич

Институт прикладной математики им. М.В. Келдыша РАН

Настоящая публикация приурочена к 70-летнему юбилею двух исторических событий – основание первого в мире академического Института прикладной математики (ИПМ) и испытание первой в мире «водородной бомбы» 12.08.1953. Оба события связаны с именами двух великих математиков – выпускников математической школы МГУ имени М.В.Ломоносова, руководителей «Трех стратегических проектов» – «Атомный», «Космический» и «Ракетно-ядерный щит», которые были реализованы с помощью математики на первых ЭВМ. Мстислав Всеволодович Келдыш, Трижды Герой Социалистического Труда (1956, 1961, 1971), первый директор ИПМ (1953-1978), лидер по «прикладной математике» и, как Леонард Эйлер, в 35 лет избран в академики (1946), Главный математик и Главный Теоретик космонавтики – входит в «Три К» вместе с И.В.Курчатовым и С.П.Королевым, лучший Президент (1961-1975) за всю почти 300-летнюю историю Академии наук. Академик (1966) Андрей Николаевич Тихонов, Дважды Герой Социалистического Труда (1954, 1986), заместитель (1953-1978) и директор (1978-1989) ИПМ, в 1939 г. избран член-корреспондентом по специальности «геофизика, математическая физика», в соавторстве с А.А.Самарским в 1951 г. издал первый учебник для вузов «Математическая физика» и заложил основы «вычислительной математики», в МГУ в 1955 г. основал первый вузовский Вычислительный центр, создал факультет вычислительной математики и кибернетики и был его первым деканом (1970-1990), идеолог и разработчик специальности «прикладная математика».

Ключевые слова: М.В.Келдыш, А.Н.Тихонов, МГУ имени М.В.Ломоносова, Академия наук, Институт прикладной математики, Атомный проект, Космический проект, Ракетно-ядерный щит, прикладная математика, компьютер, цифровая цивилизация

1. Введение

В 2023 г. в условиях тектонических геополитических сдвигов глобального порядка в мире и в России и новых вызовов вплоть до угроз третьей мировой «ядерной» войны вышел Указ Президента от 31.03.2023 № 229 «Об утверждении Концепции внешней политики РФ» [1]. Этот Указ фактически является актуализацией приоритетных направлений, целей и задач внешнеполитической деятельности в новых условиях и вызовов для отечественной науки и развития постиндустриального технологического уклада – «цифровой экономики», «цифровой цивилизации» и т.п.

XXI-й век – это век супервычислений, суперкомпьютеров и big date (ИИ – искусственный интеллект) или, как уже признано не только философами, век «цифровой цивилизации». «Человек цифровой цивилизации» – это формирование нового типа смысловой ментальности человека современной информационной цивилизации в условиях достаточно резкого изменения многих привычных стереотипов, которые до этого времени казались незыблемыми; это формирование нового типа коммуникативного пространства; это формирование нового типа человеческого сознания и новых когнитивных способностей. «Цифровая цивилизация» как современный этап развития информационного общества предполагает новые уровни развития науки, техники, технологий, общества, культуры, экономики, права, образования... А фундамент был заложен в Союзе Советских Социалистических Республик (СССР) и «Институте Келдыша» при реализации под руководством ученых Академии наук СССР (АН СССР) с помощью математики, расчетов и ЭВМ «Трех проектов» – «Атомный», «Космический», «Ракетно-ядерный щит», когда вычислительная

математика и «прикладная математика» стали производительной силой! Нас интересует роль отечественной Академии наук (АН) и та история математики в XVIII - XX веках, откуда возникли истоки создания в СССР первого в мире Института прикладной математики (ИПМ) и фундаментальные основы развития «прикладной математики» и становления «цифровой цивилизации».

«Цифровизация» – это инструмент! Фундаментом «цифровой цивилизации» являются две главные области знаний – физика и математика – «прикладная математика», на которых держатся высокие технологии. И тому подтверждение совпадение в 2023 г. юбилейных дат: 90 лет физическому и механико-математическому факультетам МГУ имени М.В.Ломоносова (МГУ) [2], созданных в 1933 г. после разделения физико-математического факультета, и двух исторических событий: 70-летие создания первого в мире ИПМ АН СССР («Институт Келдыша») [3] и успешного испытания в СССР первой в мире «водородной бомбы» [4] в 1953 г. Эти события связаны с именами двух выпускников МГУ – Мстислава Всеволодовича Келдыша (10.02.1911-24.06.1978), единственного математика Трижды Героя Социалистического Труда (1956, 1961, 1971), и Андрея Николаевича Тихонова (30.10.1906-07.10.1993), Дважды Героя Социалистического Труда (1954, 1986), которые получили базовое образование по «чистой математике», но Герои – за достижения по «прикладной математике», при этом М.В.Келдыш признан «властелином цифры».

«Цифровизация» библиотек и документов, в том числе рассекреченных, позволяет в настоящее время не только получить доступ к отечественному наследию в разных сферах, но и проводить исследования истории научно-технических достижений, отдавая дань памяти и благодарность тем выдающимся ученым, кто обеспечил высокий уровень науки и образования и безопасность страны. В связи с 70-летним юбилеем «Атомного проекта» в 2019 г. создана Электронная библиотека «История Росатома. Атомный проект СССР» [4]. По случаю 75-летия Ракетно-космической отрасли в 2021 г. появилась библиотека с историческими документами [5]. Развивается Информационная система «Архивы Российской академии наук» [6]. Перевели в электронную версию [7] уникальное издание «Персональный состав Академии наук (1724-2009)» в 4-х книгах. Пополняются Летопись МГУ [2] и «Страницы памяти» «Института Келдыша» [8].

До начала XX века научно-исследовательские институты в составе АН и в России отсутствовали. Не было и крупных научных коллективов, объединенных общим руководством и единой тематикой. Исследования и разработки выполнялись преимущественно на предметных кафедрах высших учебных заведений и в кабинетах АН. Создание первых научных учреждений стало возможным сначала благодаря Обществу содействия успехам опытных наук и их практических применений имени Х.С.Леденцова (24.02.1909), Обществу Московского научного института (16.04.1912) [9] (второе название «Общество в память 19 февраля 1861 г.» об отмене крепостного права) и других негосударственных фондов. Но после революций 1917 г. при личной поддержке А.В.Луначарского было организовано несколько научных институтов в АН [6]. А после создания в июле 1925 г. АН СССР как высшего ученого учреждения «пошел процесс» формирования научных институтов в её составе. По Уставу 1927 г. «Академия разделяется на два отделения: Отделение физико-математических наук (ОФМ) и Отделение гуманитарных наук (история, филология, экономика, социология и т.п.). Каждое из Отделений руководит ученой работой входящих в его состав исследовательских институтов, музеев, лабораторий и других ученых учреждений.»

Университетский комплекс на Ленинских горах строили всей страной после самой жестокой и разрушительной войны 1941-1945 гг. и называли дворцом для «самой передовой в мире социалистической науки». А в 2023 г., как и ИПМ, он отмечает 70-летие. Механико-математический факультет разместили в Главном здании, тем самым подчеркивали значимость математики для всех областей науки и всех факультетов МГУ. Более 40 лет назад в МГУ впервые создали автоматизированную (компьютерную) информационную систему (АИС) для приема абитуриентов («Абитуриент»), а позже – АИС для поддержки учебного процесса и педагогической нагрузки.

50 лет назад скончался Иван Георгиевич Петровский (18.01.1901-15.01.1973), Герой Социалистического Труда (1969), первый математик – ректор МГУ (1951-1973) [2], который признан лучшим ректором за всю историю Московского университета: он завершал строительство и первым осваивал новый комплекс – лучший в мире «храм науки и образования»! И.Г.Петровский в один год (1927) с А.Н.Тихоновым окончил физико-математический факультет, а в один день (30.11.1946) с М.В.Келдышем избран в академики. Три математика – М.В.Келдыш и беспартийные И.Г.Петровский и А.Н.Тихонов – решали многие проблемы организации и развития науки и образования в стране и в МГУ, а также приложений «прикладной математики». При ректоре

И.Г.Петровском в МГУ был создан факультет вычислительной математики и кибернетики [2], организатором и первым деканом которого был академик (01.07.1966) А.Н.Тихонов (1970-1990). А.Н.Тихонов в 1927-1930 гг. был аспирантом НИИ Математики и Механики при МГУ, в 1929 г. начал преподавание в МГУ на физико-математическом факультете, почти 35 лет в 1936-1970 гг. заведовал кафедрой высшей математики на физическом факультете и фактически определял содержание и методику подготовки физиков по математике в университетах и технических вузах, в том числе на «специальном отделении» физического факультета, которое было открыто в 1946 г. с целью ускоренной подготовки научных кадров для «Атомного проекта» [4].

МФТИ создали в 1951 г. на основе физико-технического факультета МГУ (1946-1951) и в большей мере МФТИ был ориентирован на подготовку кадров для ракетно-космической отрасли. Факультет управления и прикладной математики МФТИ (ФУПМ) организован в 1969 г. по инициативе членов АН СССР: Олег Михайлович Белоцерковский (29.08.1925-14.07.2015) ректор МФТИ в течение 25 лет (1962-1987), член-корреспондент с 28.11.1972, академик с 15.03.1979; Анатолий Алексеевич Дородницын (02.12.1910-07.06.1994) академик с 23.10.1953; Виктор Михайлович Глушков (24.08.1923-30.01.1982) академик с 26.06.1964; Александр Андреевич Самарский (19.02.1919-11.02.2008) член-корреспондент с 01.07.1966, академик с 23.12.1976; Дмитрий Евгеньевич Охоцимский (26.02.1921-18.12.2005) член-корреспондент с 10.06.1960, академик с 07.12.1991; Никита Николаевич Моисеев (23.08.1917-29.02.2000) член-корреспондент с 01.07.1966, академик с 26.12.1984. Первый декан ФУПМ Н.Н.Моисеев – академик по специальности «информатика» Отделения информатики, вычислительной техники и автоматизации (ОИВТА). ОИВТА основано 03.03.1983 – это было стратегическое решение и в 2023 г. ОИВТА с новым названием «Отделение нанотехнологий и информационных технологий» (ОНИТ с декабря 2007 г.) отмечает 40-летие, а его роль в «цифровой цивилизации» с каждым годом только возрастает – один из важнейших приоритетов современного научно-технологического развития.

Промышленная революция и индустриальное общество с капиталистической экономикой возникли в Англии в 1740-1780 гг. (первая паровая машина, металлургия, железные дороги, транспорт и т.д.), а затем распространились на другие страны Европы и США. Развитие промышленности стимулировало развитие физики и механики. С основания АН в 1724 г. физика получила полноправный статус самостоятельной науки. Физический институт им. П.Н.Лебедева АН СССР (ФИАН) «вырос» из Физического кабинета АН. Историю института в 1725-1945 гг. Сергей Иванович Вавилов (24.03.1891-25.01.1951), Президент АН СССР (17.07.1945-25.01.1951), как великолепный автор научно-популярной литературы увлекательно описал в книге [10]. В 1951 г. ФИАН переехал на Ленинский проспект, а в зданиях на Миуссах разместился «Институт Келдыша» [3], где всё напоминает об истории двух ключевых участников «Атомного проекта» – математиков и физиков [9-11]. О достижениях точных наук академик (04.03.1917) физик и первый биофизик Петр Петрович Лазарев (13.04.1878-23.04.1942) говорил в Докладе на Торжественном заседании, посвященном 200-летнему юбилею Академии Наук, Москва, 13.09.1925 [12].

Там же Анатолий Васильевич Луначарский (23.11.1875-26.12.1933) – писатель, литературовед, драматург, поэт, искусствовед, переводчик (знал семь европейских языков), публицист; первый народный комиссар просвещения (1917-1929); академик (01.02.1930) по истории литературы Отделения гуманитарных наук АН СССР – сделал блестящий доклад «К 200-летию Всесоюзной Академии наук» [13]. Молодому советскому государству повезло с министром [14]: успешно боролся с неграмотностью, науку и образование поддерживал и вопросы решал быстро, общаясь напрямую с учеными, поскольку обладал острым умом, не терпел бюрократию и о государстве заботился. В 1919 г. академик (01.07.1912) Владимир Андреевич Стеклов (09.01.1864-30.05.1926) договорился с А.В.Луначарским о создании в Петербурге «Математического кабинета» в составе АН, а в 1921 г. был создан первый академический «Физико-математический институт» (ФМИ). В 1923 г. В.А.Стеклов написал уникальную книгу «Математика и её значение для человечества» [15]. Директора ФМИ – академики: В.А.Стеклов (1921-1926), А.Ф.Иоффе (1926-1928), А.Н.Крылов (1928-1932), И.М.Виноградов (1932-1934). На Общем собрании АН СССР 28.04.1934 принято Постановление о разделении ФМИ – заложили фундамент двух отечественных мировых центров: Физический институт им. П.Н.Лебедева АН СССР (ФИАН), первый директор С.И.Вавилов (1934-1951), и Математический институт им. В.А.Стеклова АН СССР (МИАН) [16], первый директор И.М.Виноградов (1934-1941, 1944-1983). В МИАН с 1934 г. учился в аспирантуре

и докторантуре и был зам. директора М.В.Келдыш, основная работа которого была в ЦАГИ (1931-1946) и в РНИИ. В МИАН в 1953 г. был рожден «Институт Келдыша».

В критические моменты истории и научно-технологического прорыва необходимы лидеры, которые возьмут на себя ответственность и возглавят выход из кризиса или проект эпохального или цивилизационного значения во имя мира на земле или победы над конкурентами и развития.

Царь Петр Первый по личной инициативе стал лидером и возглавил цивилизационный проект в России. Вершиной преобразовательной деятельности Петра стал его главный замысел – сделать Россию новым научно-промышленным европейским центром, воспитать своих ученых, а для этого создать Академию наук и привлечь в неё для начала самых знаменитых ученых мужей Европы, вменив им в обязанность и обучение отечественных отроков. Одной из первых задач Петра было обеспечить рост наук, необходимых для создания флота и изучения огромных пространств империи, то есть астрономии, геодезии, картографии. По академическому Уставу звание профессора астрономии было отнесено к наивысшему, первому классу. Опорой же для этих наук были математика и механика (иначе физика), поэтому из 17 первых профессоров (так тогда называли членов Академии) в первом её составе было семь математиков и физиков.

После Февральской и Октябрьской революций 1917 г. в период радикальных перемен, гражданской войны и разрухи спасением Академии наук, науки, ученых занялись академики А.П.Карпинский, геолог, и В.А.Стеклов, математик, по своим убеждениям, как говорил М.В.Келдыш: «Родина у нас одна! Запасной нет…» – необходимо профессионально сотрудничать с государством. Результатами своей созидательной деятельности, любовью к науке и целеустремленностью они привлекли на свою сторону большинство научной элиты и стали признанными лидерами. Александр Петрович Карпинский (07.01.1847-15.07.1936) – первый избранный Президент РАН (15.05.1917-27.07.1925) и Первый Президент АН СССР (27.07.1925-15.07.1936). За высочайшие заслуги перед государством и народом только два Президента АН СССР – А.П.Карпинский и М.В.Келдыш удостоены захоронению на Красной площади в Кремлевском некрополе.

В середине XX-го века в разгар «холодной войны» и реальной угрозы «атомного удара» по СССР со стороны США появились советские лидеры – три русских ГЕНИЯ – «Три К» – Мстислав Всеволодович Келдыш – Главный Теоретик космонавтики и Главный математик; Сергей Павлович Королев (12.01.1907-14.01.1966) – Главный Конструктор космонавтики, Дважды Герой Социалистического Труда (1956, 1961); Игорь Васильевич Курчатов (12.01.1903-07.02.1960) – «Отец русской атомной бомбы», Трижды Герой Социалистического Труда (1949, 1951, 1954), которые возглавили и совершили научно-техническую революцию в СССР, и их имена навечно вошли в историю мировой цивилизации человечества, СССР и России. Эти академики – гордость ВЕЛИКОЙ Академии наук СССР! Эти Герои и прежде всего М.В.Келдыш [17-23] – лучший Президент Академии наук (19.05.1961-19.05.1975) за всю её почти 300-летнюю историю [22] – подняли престиж НАУКИ и УЧЕНЫХ на такую недосягаемую высоту, что АН СССР стала форпостом СССР в мире и СССР уважали во всем мире, а не только боялись! Это благодаря их научным подвигам и заслугам был обеспечен стратегический паритет двух политических систем и военных блоков во главе с СССР и США и более 75 лет нет глобальных мировых войн.

Выдающийся ученый и блестящий популяризатор науки академик (03.03.1912) Владимир Иванович Вернадский (12.03.1863-06.01.1945) в докладе на тему «Мысли о современном значении истории знаний» [24], прочитанном на первом заседании Комиссии по истории знаний АН 14.11.26, высказал много умных и полезных мыслей, актуальных и ныне. Важно помнить о преемственности в науке: «История науки является в такие моменты орудием достижения нового», говоря о переломных моментах или острых проблемах в истории государств. Первая мировая война выявила проблемы в изучении минерально-сырьевой базы России. В 1915 г. В.И.Вернадский выступил в качестве учредителя и председателя Комиссии по изучению естественных производительных сил (КЕПС), которую поддержал А.П.Карпинский, что помогло с топливом во время войны, а также позволило спасти Академию наук, науку, ученых после революций 1917 г. В.И.Вернадский организовал Радиевый институт и с 1922 по 1939 гг. был его директором. После войны 1941-1945 гг. без этого института СССР не справился бы оперативно с «Атомным проектом». Под руководством М.В.Келдыша и А.Н.Тихонова проведены расчеты для первых успешных испытаний «атомной» (29.08.1949) и «водородной» (12.08.1953) бомб [4]. А.Н.Тихонов – творец современной «прикладной математики» [25]. М.В.Келдыш и А.Н.Тихонов через «Три проекта» создавали фундаментальные основы для зарождения «цифровой цивилизации».

2. Из истории науки и математики в России XVIII века

С основания АН физика была самостоятельной областью науки, а математика ещё более 100 лет была приложением к другим наукам – физика, механика, астрономия, география, навигация, физиология, философия и т.д. И тому подтверждение разнообразные научные интересы Леонарда Эйлера и других первых членов АН. Как самостоятельная область науки математика началась с М.В.Остроградского в начале XIX века, в середине XIX века П.Л.Чебышев [26] создал Петербургскую математическую школу, а в конце XIX века началось формирование Московской математической школы. Расцвет математики свершился после создания АН СССР в 1925 г. и об этом доклады М.А.Лаврентьева [27] в 1947 г. и М.В.Келдыша [28] в 1966 г. Наиболее значимые факты отражены в изданиях, приуроченных к юбилеям 250 и 275 лет Академии Наук [29-32].

А всё начиналось с «Арифметики Магницкого» [33] – 662 страницы, посвященные арифметике, алгебре, геометрии, тригонометрии, астрономии, геодезии и навигации, изданной 320 лет назад по личной инициативе Царя Всея Руси Петра в 1703 г. Л.Ф.Магницкий: «Арифметика, сиречь наука числительная. С разных диалектов на славянский язык переведённая, и воедино собрана, и на две части разделённая». Начался многовековой процесс развития операций с числами и «цифровизации» и становления математики как самостоятельного направления в науке. Удивительный факт: «Институт Келдыша» основан в год 250-летия выхода первого учебника по математике и её приложениям, написанном сыном крестьянина Леонтием Теляшиным, увлеченным математикой выпускником Славяно-греко-римской академии. Петр даст первому математику фамилию Магницкий, который «как магнитом будет тянуть знания к людям».

22 января 1724 г. Л.Л.Блюментрост по поручению Петра представил «Проект положения об учреждении Академии наук и художеств» («генеральный регламент»), а 28 января (8 февраля) 1724 г. вышел именной указ Сената об учреждении Императорской Академия наук в Петербурге, извещавший, что Петр I «указал учинить Академию, в которой бы учились языкам, также протчим наукам и знатным художествам и переводили б книги». В мае 1917 г. она была переименована в Российскую академию наук. Основание АН пришлось на время бурного расцвета математики и механики в мире. На рубеже XVII-XVIII веков, как писал Энгельс, «первое место заняло элементарнейшее естествознание – механика земных и небесных тел, а наряду с ней, на службе у нее, открытие и усовершенствование математических методов. Здесь были совершены великие дела». АН состояла из трех классов: первый объединял математику, астрономию, механику с географией, второй – физику, химию и естественные науки, третий – гуманитарные дисциплины. АН предназначалась роль не только высшего научного учреждения, но и основного центра подготовки ученых и вспомогательного персонала для ее многочисленных учреждений – обсерваторий, лабораторий, инструментальных мастерских, библиотеки, музея, ботанического сада, издательства и типографии. С этой целью при АН учреждались университет и гимназия, а на академиков возлагались преподавание в них и индивидуальные занятия с наиболее способными студентами. АН стала основателем университетского образования и подготовки кадров в России.

С открытием АН кафедра физики и Физический кабинет становятся первым в России центром теоретических и экспериментальных исследований в этой области знаний. Первые ростки фундамента науки и «цифровой цивилизации» в России появились в Кунсткамере, где лично Петром I был создан первый Физический кабинет, и при подготовке издания «Арифметики Магницкого». До создания АН в России не было ученых и профессоров, университетов и научных институтов, лабораторий, потому в 1725 г. в АН были приняты только иностранные ученые, профессора и адъюнкты. От Физического кабинета до первого Физического института АН (1934) путь в два столетия. Математический институт (1934) явился первым математическим институтом АН, который начинал свой путь со скромного Кабинета математики Л.Эйлера. К началу XVIII века стало очевидным значение математического аппарата для разработки многих теоретических и практических проблем науки и техники, имеющих государственное и общественное значение. При организации АН это обстоятельство учитывалось в полной мере. Среди 23 академиков, приглашенных на работу в течение первых лет, семь являлись математиками. Это были Я.Герман, Х.Гольдбах, Ф.Х.Майер, Г.В.Крафт, молодые братья Николай и Даниил Бернулли и их совсем еще юный друг Л.Эйлер. Членами АН были пять братьев Бернулли!

Вначале академиков пришлось выписывать (приглашать персонально) из-за рубежа, преимущественно из Швейцарии и Германии. Первые в АН профессора по высшей математике – известные иностранные ученые, которые начали работать в 1725 г.: Герман Якоб (16.07.1678-11.07.1733), математик, профессор высшей математики с 08.01.1725, иностранный почетный член с 01.01.1731; Майер Фридрих Христофор (09.10.1697-24.11.1729), математик, философ, адъюнкт по математике с 01.07.1725, экстраординарный профессор с 29.01.1726; Гольдбах Христиан (18.03.1690-20.11.1764), математик, профессор математики с 01.09.1725, почетный член с 18.03.1742; Николай Бернулли (06.02.1695-29.07.1726), математик, философ, юрист, профессор математики с 27.10.1725, блестящий молодой ученый умер и похоронен в Санкт-Петербурге.

В 1725-1727 гг. первые физики в АН были междисциплинарные и с математическим уклоном: Крафт Георг Вольфганг (15.07.1701-18.07.1754), математик, физик, астроном, адъюнкт по астрономии с 01.07.1727, профессор математики с 01.01.1731, профессор физики с 31.01.1731, иностранный почетный член с 01.01.1745; Бюльфингер Георг (23.01.1693-18.02.1750), математик, физик, философ, профессор логики и метафизики с 01.03.1725, профессор экспериментальной и теоретической физики с 14.01.1726, иностранный почетный член с 01.01.1731; Мартини Христиан (01.07.1699-01.08.1739), физик, математик, философ, профессор физики с 13.01.1725, профессор логики и метафизики с 14.01.1726.

Математика развивались в АН с 1725 г. Академии повезло: огромную роль сыграли Д.Бернулли и Л.Эйлер – выходцы из Базельского университета – всемирно известные математики (хотя оба начинали со специальности «физиология»), участвовавшие в создании АН. Получив приглашения, Даниил и Николай Бернулли отправились в Петербург и обещали Эйлеру подыскать и для него место в АН. Узнав, что вакансия может открыться на отделении медицины, Л.Эйлер начинает в Базеле изучать физиологию и медицину. В 1726 г. Л.Эйлер получил официальное приглашение в Петербург для работы в АН адъюнктом по физиологии. Ему было 19 лет.

Даниил Бернулли (09.02.1700-17.03.1782), математик, профессор физиологии с 05.07.1725, профессор математики с 30.06.1727, иностранный почетный член с 23.03.1733. Обострились отношения с Иоганн-Даниилом Шумахером (1690-1761), начальником академической канцелярии и первым заведующим библиотекой АН, и 24.06.1733 Д.Бернулли уехал из Петербурга – заехал в Париж, в конце 1733 г. вернулся в Швейцарию, но контакты не прерывал – с Эйлером находился в постоянной переписке до 1750 г. Д.Бернулли получил в Базельском университете кафедру анатомии и ботаники, но больше занимался экспериментальной физикой и в 1750 г. он возглавил кафедру физики. Более всего Д.Бернулли прославился трудами в области математической физики и теории дифференциальных уравнений – его считают, наряду с Д'Аламбером и Л.Эйлером, основателем математической физики. Д.Бернулли основательно обогатил теоретическую механику, кинетическую теорию газов, гидродинамику и аэродинамику, теорию упругости и т.д. В математике Д.Бернулли опубликовал ряд исследований по теории вероятностей, теории рядов, численным методам и дифференциальным уравнениям. Он первый применил математический анализ к задачам теории вероятностей (1768), до этого использовали только комбинаторный подход. Д.Бернулли продвинул математическую статистику, рассмотрев с применением вероятностных методов ряд практически важных задач. Наука была единственной страстью Даниила Бернулли. Он не был женат и скончался тихо – заснул в рабочем кресле навсегда.

«Самодержавною рукой» царь «смело сеял просвещенье». Среди тех, кто первым приехал поднимать науку в Петербурге, был юный Л.Эйлер. В России произошло формирование Л.Эйлера как первого и крупнейшего наследника и продолжателя дела Ньютона по созданию нового естествознания – новой математики, механики и теоретической астрономии. В 1988 г. в Ленинграде по инициативе академика (23.12.1976) Людвига Дмитриевича Фаддеева (23.03.1934-26.02.2017) под эгидой Ленинградского отделения МИАН СССР был основан первый Международный математический институт имени Эйлера с целью организации международного научного сотрудничества и создания условий для совместных исследований советских, затем российских и зарубежных учёных. Ныне это Санкт-Петербургский международный математический институт имени Леонарда Эйлера, работа которого осуществляется на базе двух организаций – Санкт-Петербургского отделения Математического института имени В.А.Стеклова РАН и Санкт-Петербургского государственного университета при координации СПбГУ.

Л.Эйлеру (04.04.1707-07.09.1783) принадлежит исключительная роль в истории АН, мировой и отечественной науки и математики. Швейцарец с «русской душой» – один из величайших

математиков всех времен – отличался неудержимой тягой к знаниям и неуемной энергией. Его именем названы многие классические теоремы во всех областях математики. Л.Эйлер – математик, механик, физик – заложил основы первой в России математической школы. Однако его отношения с АН складывались сложно: адъюнкт по физиологии с 17.12.1726; профессор физики с 01.01.1731; профессор по высшей математике с 15.06.1733 по 05.06.1741; иностранный почетный член с 04.05.1742; вторично профессор с 26.04.1766. В 23 года он возглавляет кафедру экспериментальной и прикладной физики, уже через год официально становится профессором по математике, а еще через два года в 35 лет – академиком. А.Н.Колмогоров (25.04.1903-20.10.1987), как и М.В.Келдыш, тоже в возрасте 35 лет стал академиком (20.01.1939).

Только в 1733 г. в АН был избран первый русский ученый – математик Ададуров (Адодуров) Василий Евдокимович (15.03.1709-05.11.1780) из Нижнего Новгорода, адъюнкт по высшей математике с 26.10.1733 по апрель 1741 г., почетный член с 28.09.1778. Позднее в XVIII веке в АН работали Котельников Семен Кириллович – математик, Эйлер Иоганн Альбрехт – математик, физик, астроном, Фусс Николай Иванович – математик, Головин Михаил Евсеевич – математик, Шуберт Федор Иванович – математик, астроном, геодезист, Гурьев Семен Емельянович – математик, механик. Все они внесли заметный вклад в развитие математических наук и образования, но наиболее яркими оказались С.К.Котельников и М.Е.Головин.

Второй русский ученый в АН – Михаил Васильевич Ломоносов (08.11.1711-04.04.1765), физик, химик, астроном, специалист в области горного дела, географ, историк, филолог, поэт. В АН отношения не сложились и занимал скромные должности: с 08.01.1742 – адъюнкт физического класса, с 25.07.1745 – профессор химии; начальник научного и учебного департаментов канцелярии АН, член-корреспондент иностранных академий наук, хорошо известен в европейских научных кругах и при дворе государыни Елизаветы Петровны. Но «Российская грамматика» [34] принесла М.В.Ломоносову поистине всероссийскую известность, потому что не корпускулярно-кинетическая теория, не закон сохранения массы, не другие фундаментальные открытия ученого-энциклопедиста, а именно этот его научный труд напрямую касался всего грамотного населения империи. И главное достижение М.В.Ломоносова – в АН русский язык стал официальным! М.В.Ломоносов навсегда останется в истории российского государства как основатель в 1755 г. Московского университета – лучший университет страны носит его имя. М.В.Келдыш и А.Н.Тихонов, а также многие сотрудники «Института Келдыша» – достойные выпускники МГУ: они создали первый в мире легендарный академический ИПМ – центр мировой науки! Ни в одной стране мира не было ничего подобного и не случайно его называли «Институт Келдыша»!

На вопрос «Кто величайший ученый России XVIII века?» большинство скажет: «Конечно же, Михаил Ломоносов!». И только некоторые, может, вспомнят Л.Эйлера, который глубоко изучал медицину, химию, ботанику, воздухоплавание, теорию музыки, множество европейских и древних языков. И в этом очень похож на гениального Михаила Ломоносова. Л.Эйлер оставил свыше 850 фундаментальных работ. И в последние годы жизни был слепым. А ведь они жили в одно время, в одном городе и оба трудились на научном поприще в одной АН по близким темам и областям науки. И никогда не встречались! Загадка решается очень просто. М.Ломоносов приехал в Петербург после пятилетнего обучения в Германии 08.06.1741 – день отъезда Л.Эйлера. Прожив затем здесь 24 года, умер 04.04.1765, то есть за год с лишним до возвращения Л.Эйлера. В 1741-1766 гг. покидал Россию и работал в Берлинской АН. Но скончался и похоронен Л.Эйлер в Санкт-Петербурге. В ознаменование 250-летия со дня рождения Л.Эйлера в 1957 г. его могила была перенесена со Смоленского лютеранского кладбище на Лазаревское кладбище Александро-Невской лавры. Там же похоронен Михаил Ломоносов. Вот и встретились два гения. Две первые великие вершины в истории деятельности нашей АН – Л.Эйлер и М.В.Ломоносов – стали ярчайшим выражением исполнения замыслов и завещания преобразователя России – Петра Первого.

Особо следует отметить Барсова Антона Алексеевича (01.03.1730-21.12.1791) – ученика М.В.Ломоносова, который после окончания Санкт-Петербургского академического университета (1748-1753) был удостоен ученой степени магистра философии и свободных наук (1753), оставлен при АН, читал лекции по математике и занимался переводами. 12(25) января 1755 г. императрица Елизавета Петровна подписала в день памяти св. Татианы «Проект об учреждении Московского университета», а в феврале 1755 г. А.А.Барсов был зачислен И.И.Шуваловым в штат Московского университета преподавателем математики и проработал в нем 35 лет (1755-

1791). На Торжественной церемонии открытия университета 26.04.1755 г. выступили Н.Н.Поповский и А.А.Барсов – ученики Ломоносова. А.А.Барсов произнёс «Речь о пользе учреждения Московского Университета». А.А.Барсов преподавал математику в университетской гимназии (1755-1760), с 1757 г. читал лекции по математике, латинскую и российскую риторику, с января 1761 г. ординарный профессор кафедры красноречия философского факультета (1761-1791) – фактически первый профессор математики в МГУ. Летом 1785 г. А.А.Барсов был представлен университетской Конференцией к должности экстраординарного профессора математики, но не получил его из-за отсутствия такой должности в «Проекте об учреждении Московского университета». Он был одним из первых русских профессоров, стоявших у колыбели МГУ, декан философского факультета, заведующий кафедрой словесности. Поражает многосторонность интересов А.А.Барсова со студенческих лет: литература и философия, физика и астрономия, математика и история, музыка и иностранные языки (латинский, французский, немецкий). Но главное – это ломоносовская страсть к занятиям наукой во славу Отечества, страсть, которой он был одержим до конца своих дней. 21.10.1783 г. А.А.Барсов был приглашён на первое заседание Российской Академии наук, возглавляемой княгиней Е.Р.Дашковой, но членом АН и РАН не стал.

Второй русский математик в АН – С.К.Котельников (1723-30.03.1806), первый русский ученик Эйлера, – адъюнкт по математике с 01.03.1751, экстраординарный профессор высшей математики с 14.12.1756, ординарный профессор с 01.01.1760 по 16.02.1797, почетный член с 23.02.1797; в 1785-1796 гг. читал публичные лекции по высшей математике; ему принадлежат первый русский учебник по механике (1774), одно из первых русских руководств по геодезии (1766) и др. Как и Л.Эйлер, он занимался и «прикладной математикой.

Третий русский математик в АН – М.Е.Головин (1756-08.06.1790), механик, физик, адъюнкт по математике с 15.01.1776 по 19.02.1786, почетный член с 13.02.1786; племянник М.В.Ломоносова из Архангельской губернии. Его учителем в области математики был Л.Эйлер, а в области физики – Л.Ю.Крафт (профессор экспериментальной физики с 08.04.1771). Основные труды посвятил вопросам преподавания. Был первым отечественным физиком-методистом и организатором преподавания физики в народных школах. Впервые составил «Краткое руководство к арифметике. В 2-х ч.» (1783—1784), «Краткое руководство к физике» (1785), «Краткое руководство к геометрии» (1786). Написал «Краткое руководство к механике» (1785), в котором использовал понятие, близкое к сформировавшемуся в дальнейшем понятию вектора. Этот труд просмотрел и одобрил Л.Эйлер. М.Е.Головин много занимался переводами и изданием словарей. Перевел на русский язык несколько трудов Л.Эйлера. В качестве корректора М.Е.Головин участвовал в издании собрания сочинений М.В.Ломоносова.

3. Из истории науки и математики в России XIX века

Математика как самостоятельное направление в науке сформировалась только в XIX веке. Большинство профессоров и членов АН имели несколько специальностей и математика была одной из них как «прикладная». В XIX веке были созданы первые русские Петербургская и Казанская математические школы, а «чистая» и «прикладная» математика вышла на мировой уровень. В АН состояли выдающиеся математики XIX века: М.В.Остроградский, П.Л.Чебышёв, А.М.Ляпунов, А.А.Марков (старший), В.А.Стеклов, А.Н.Крылов. В Академию впервые была избрана женщина – С.В.Ковалевская (15.01.1850-10.02.1891), с 02.12.1889 иностранный член-корреспондент по разряду математических наук (математика) Физико-математического отделения. Но не все крупные российские математики оказались избранными в АН и ярчайшим примером является Н.И.Лобачевский – знаменитый русский и мировой геометр.

Георг Фридрих Паррот (Егор Иванович, 05.07.1767-20.07.1852) – физик, член-корреспондент с 04.12.1811, первый ординарный академик по «прикладной математике» с 26.04.1826, по кафедре физики с 24.03.1830, почетный член с 18.12.1840; немецкий ученый, заслуженный профессор физики и первый ректор (1802-1813 с перерывами) Императорского Дерптского (Юрьевского, Тартусского) университета в Дерпте (Тарту, Эстония) на территории Лифляндской губернии Российской империи; «личный друг» Александра I. В лице Г.Паррота АН получила крупного организатора науки и преподавателя, ведь именно он создал в Дерптском университете образцовые Физический кабинет и Астрономическую обсерваторию (Тыравере). Г.Паррот оставил боль-

шой след в истории АН и российской науки – создал Физический кабинет и Пулковскую Астрономическую обсерваторию, руководил деятельностью Лаборатории физики при АН в общей сложности 16 лет (1824-1840). Физический кабинет был перенесен из не вполне подходящего здания Кунсткамеры в саму Академию в 1828 г. Хотя его предшественники долго пытались добиться этого разрешения, но лавры «переносчика» все же достались Г.Парроту. Приняв от В.В.Петрова Физический кабинет, располагавшийся в Кунсткамере, Г.Паррот реорганизовал его и перевел в главное здание АН. Впоследствии из него выросла Физическая лаборатория, затем ФМИ (находились здесь до перевода АН в Москву в 1934 г.). С этого Кабинета начался ФИАН!

Михаил Васильевич Остроградский (12.09.1801-20.12.1861) – первый отечественный математик и механик (украинского происхождения, родился в Полтаве и до конца жизни говорил с акцентом), избранный в АН по специальностям «прикладная математика» и «чистая математика»: адъюнкт по «прикладной математике» с 17.12.1828, экстраординарный академик с 11.08.1830, ординарный академик по «прикладной математике» с 21.12.1831, ординарный академик по «чистой математике» с 15.06.1855. В 1826 г. первые научные успехи: М.В.Остроградский представил Парижской Академии наук работу «О распространении волн в цилиндрическом бассейне». Знаменитый французский математик Коши писал об М.В.Остроградском: «Этот русский молодой человек одарён большой проницательностью и весьма сведущий». В 1828 г. возвратился на родину с французским дипломом и с заслуженной репутацией талантливого учёного – первый признанный лидер математиков Российской империи в 1830-1860 гг. и стал знаменитостью мирового уровня. Основные работы относятся к прикладным аспектам математического анализа, математической физики, теории вероятностей, аналитической геометрии, механики, теории теплоты, теории упругости, гидродинамики, небесной механики, теории магнетизма и др.; многие труды посвятил математической физике и аналитической механике, став одним из тех, кто заложил фундамент этих наук; внёс также вклад в алгебру и теорию чисел.

К московской ветви школы М.В.Остроградского относятся выдающиеся ученые – Н.Д.Брашман, Н.Е.Жуковский, С.А.Чаплыгин, к петербургской – П.Л.Чебышев, А.М.Ляпунов, В.А.Стеклов, А.Н.Крылов. Многие другие математики и механики России также испытали на себе его влияние. В 1830-е годы все математические предметы в офицерских корпусах преподавали М.В.Остроградский либо его ученик и коллега В.Я.Буняковский. М.В.Остроградский учил математике Александра II. Благодаря выдающимся научным заслугам, М.В.Остроградский был избран членом-корреспондентом Парижской Академии наук, членом Американской, Римской и других академий и научных обществ. Почётный член Московского университета. М.В.Остроградский оказал огромное влияние на формирование отечественной математики: его научные труды и универсальные результаты вошли в фундамент современной «прикладной математики», которая имела важную роль для научно-технического прогресса в XX веке и востребована в XXI веке, а также для основ «цифровой цивилизации».

Виктор Яковлевич Буняковский (03.12-1804-30.11.1889), ученик М.В.Остроградского, – математик, адъюнкт по «чистой математике» с 07.05.1828, ординарный академик по Физико-математическому отделению с 08.01.1841, первый математик – вице-президент АН с 10.08.1863 по 26.09.1889! В 1846 г. появилась научная монография В.Я.Буняковского «Основания математической теории вероятностей», послужившая началом его всемирной известности. Этот обширный трактат, кроме теории, заключал в себя и историю возникновения и развития теории вероятностей; в нём впервые сведено вместе всё то, что было выработано по этой теории трудами известных математиков, начиная с Паскаля и Ферма. Форма трактата отличалась такой удобопонятностью и изяществом, что немец Карл Фридрих Гаусс, один из величайших математиков всех времён, «король математиков», и француз Ирене-Жюль Бьенеме, специалист по теории вероятностей и статистике, выучились русскому языку по этому сочинению. Оба были членами АН: К.Ф.Гаусс (30.04.1777-23.02.1855) – член корреспондент с 31.01.1802, почетный член с 24.03.1824; И.Ж.Бьенеме (28.08.1796-19.10.1878) – член корреспондент с 13.12.1874.

Лобачевский Николай Иванович (20.11.1792-12.02.1856) – величайший русский математик, один из первых создателей неевклидовой геометрии, выдающийся деятель университетского образования и народного просвещения. Окончив гимназию в 1806 г., Н.И.Лобачевский поступил в Императорский Казанский университет, открытый в 1805 г. По его окончании в 1811 г., получив степень магистра по физике и математике с отличием, Н.И.Лобачевский был оставлен при учеб-

ном заведении для подготовки к научной карьере, однако в течение 40 лет преподавал. Н.И.Лобачевского произвели в экстраординарные профессора – ему было 23 года! – и доверили чтение базовых математических курсов. Спустя четыре года он стал деканом физико-математического университета и был им семь лет, с 1820-го по 1827-й. Н.И.Лобачевский – первый математик – ректор университета: в 34 года избирается ректором и признан великим ректором Императорского Казанского университета (1827-1845). Членом Академии наук не был.

Брашман Николай Дмитриевич (14.06.1796-13.05.1866) – математик и механик, родился в Австро-Венгрии, образование получил в Венском политехническом институте и в Венском университете, в 1823 г. отправился в Россию, в Санкт-Петербург. С января 1824 г. преподавал математику и физику в Главном немецком училище Св. Петра (Петропавловском училище). В марте 1825 г. был определён адъюнктом физико-математических наук в Казанский университет, где преподавал «чистую математику», астрономию и механику на физико-математическом факультете, где деканом, а затем ректором был Н.И.Лобачевский. С августа 1834 г. Н.Д.Брашман – экстраординарный профессор по кафедре прикладной математики Московского университета, а с января 1835 г. – ординарный профессор по той же кафедре. В этой должности Н.Д.Брашман работал до своей отставки в 1864 г. В 1839 г. он принял присягу на подданство России. В 1841 г. на торжественном собрании университета произнёс речь «О влиянии математических наук на развитие умственных способностей». С 03.12.1855 член-корреспондент по разряду математическому Отделения физико-математических наук Академии наук – научный руководитель П.Л.Чебышёва! Н.Д.Брашман – заслуженный профессор Московского университета (1859), основатель Московского математического общества (1864) и его печатного органа – журнала «Математический сборник» (1866) – это старейший русский математический журнал.

Пафнутий Львович Чебышёв (14.05.1821-26.11.1894) – величайший русский математик XIX века, академик Петербургской Академии наук и ещё 24 Академий в разных странах мира. П.Л.Чебышев – почетный член всех российских университетов того времени. В семье Чебышевых служение Отечеству было традицией [26]. П.Л.Чебышёв окончил физико-математическое отделение философского факультета Московского Университета (1837-1846). С 16 лет он углубился в высшую математику, где, благодаря Н.Д.Брашману, изучал в оригинале работы французского инженера-изобретателя Жана-Виктора Понселе, известного в научном мире как создателя проектной геометрии и основоположника динамики машин. В 20 лет Пафнутий Чебышёв получает диплом с отличием, а в 25 лет он блестяще защитил диссертацию на степень магистра «Опыт элементарного анализа теории вероятностей». В 1847 г. П.Л.Чебышёв переехал в Петербург по приглашению академика В.Я.Буняковского и приступил к работе при АН, где вместе с В.Я.Буняковским готовил к изданию рукописные труды Л.Эйлера по теории чисел, а на кафедре Петербургского университета начал чтение лекций по алгебре и теории чисел. В 1849 г. П.Л.Чебышев защитил докторскую диссертацию, удостоенную Демидовской премии. Диссертацией служила его книга «Теория сравнений», которой затем в течение долгого времени студенты пользовались как одним из самых глубоких и серьезных руководств по теории чисел. В 1850 г. П.Л.Чебышев стал профессором Петербургского университета. В 1853 г. академики П.Н.Фусс, В.Я.Струве, Б.С.Якоби, В.Я.Буняковский представили П.Л.Чебышева к избранию в адъюнкты Императорской академии наук, особо отметив важность его работ в области практической механики.

П.Л.Чебышёв – математик, механик, адъюнкт прикладной механики по кафедре прикладной математики с 14.05.1853; экстраординарный академик с 03.08.1856 и ординарный академик с 06.02.1859 по прикладной математике на той же кафедре. В 1882 г. П.Л.Чебышев прекратил чтение лекций в Петербургском университете (в течение 35 лет!) и, выйдя в отставку, целиком занялся научной работой в АН. Фактически П.Л.Чебышёв создал и внедрил в математике самостоятельное направление, которое в настоящее время называют «прикладная математика», поскольку на протяжении всей научной деятельности основой творческого метода П.Л.Чебышёва непременно являлась связь абстрактной математической теории не только с естествознанием и техникой, но и с практикой. П.Л.Чебышёв – блестящий профессор и научный руководитель – вместе с учениками сформировал научный коллектив математиков и стал одним из главных основоположников Петербургской математической школы, а наряду с Н.И.Лобачевским – основатель русской математической школы в России. Многочисленные ученики П.Л.Чебышева внесли самостоятельный значительный вклад в науку. Среди них известные математики, механики и физики: А.В.Васильев, Г.Ф.Вороной, Д.А.Граве, Е.И.Золотарев, А.Н.Коркин, Д.А.Лачинов,

А.М.Ляпунов, А.А.Марков (старший), К.А.Поссе, И.Л.Пташицкий, П.И.Сомов, Ю.В.Сохоцкий, М.А.Тихомандрицкий. Новую теорию пространственных зубчатых механизмов создал ученик П.Л.Чебышева – Х.И.Гохман. Работы П.Л.Чебышёва по теории чисел, теории вероятностей и алгебре востребованы в XXI веке для развития фундаментальных основ «цифровой цивилизации» и новых направлений в математике, связанных с кибербезопасностью.

Следует отметить гражданскую позицию П.Л.Чебышёва: в течение сорока лет на общественном уровне (без гонорара) П.Л.Чебышёв принимал активное участие в работе военного артиллерийского ведомства: с 1855 г. действительный член Артиллерийского отделения Военно-учёного комитета, с 1859 г. действительный член Временного артиллерийского комитета и работал над усовершенствованием дальнобойности и точности артиллерийской стрельбы, применяя для обработки результатов опытных стрельб методы теории вероятностей. В курсах баллистики до наших дней сохранилась формула Чебышёва для вычисления дальности полёта снаряда в зависимости от его угла бросания, начальной скорости и сопротивления воздуха при заданной начальной скорости. Своими трудами П.Л.Чебышёв оказал большое влияние на развитие русской артиллерийской науки, на приобщение учёных-артиллеристов к математике. Это направление работ П.Л.Чебышёва в XX веке оказало значительное влияние на разработку в XXI веке ракетной проблематики, которая далее вывела человечество в космос.

Важно помнить Столетова Александра Григорьевича (29.07(10.08).1839-13(27).05.1896) – выпускника физико-математического факультета (1860) и первого физика Московского университета, получившего мировое признание [2]. А.Г.Столетов организовал первую учебно-исследовательскую физическую лабораторию и написал первый учебник «Теоретическая физика», в котором изложены и основы «математической физики». В 1892 г. А.Г.Столетов возглавил комиссию по повышению заработной платы служащим и рабочим Московского университета. В 1893 г. был выдвинут в члены АН, но получил отказ её президента великого князя К.К.Романова «как вредный крамольник». Такую репутацию заслужил за активную гражданскую позицию. Памятники А.Г.Столетову и П.Н.Лебедеву установлены на входе в физический факультет МГУ.

4. Из истории науки и математики в России XX века

Вера в могучую силу разума, в высокое и благородное предназначение науки – вот источник неиссякаемых сил УЧЁНЫХ и ЛИДЕРОВ в XX веке! На том держалась ВЕЛИКАЯ АКАДЕМИЯ НАУК СССР! Творчество, созидательный труд, культ научного подвига и всеобщая грамотность – вот основа достижений СССР в XX веке! XX век в истории земной цивилизации – это век научно-технической революции (НТР), связанной с тремя открытиями: проникновение в тайны атома и овладение ядерной энергией; покорение космического пространства и выход человека в космос; изобретение электронно-вычислительных машин (ЭВМ) и создание информационных технологий, которые стали движущей силой НТР и обеспечили успех «Трех проектов».

Увлекательное путешествие в век XX – время великих открытий, освоение атома и космоса, начало эпохи расцвета науки и образования. Символом этих перемен для отечественной и мировой науки стал Мстислав Всеволодович Келдыш. В середине XX века благодаря М.В.Келдышу фундаментальная и «прикладная математика» была поднята на такую высоту, что стала движущей силой НТР и таковой остается в XXI веке. В мировой практике эту область науки называют «computer science», занижая роль математики, без которой ни один компьютер работать и что-то считать не сможет. Единственный математик, именем которого названа «Эпоха Келдыша» [23].

Первая русская математическая школа была создана П.Л.Чебышёвым в Петрограде и Н.И.Лобачевским в Казани в середине XIX века, а в Москве – академиком (чл.-к. 15.01.1927, академик 12.01.1929) Николаем Николаевичем Лузиным (09.12.1883-28.02.1950) в первой половине XX века. Расцвет разных наук и математики наступил после 1925 г., когда была создана Академия наук СССР как высшее ученое учреждение Союза ССР [29-32]. Это было стратегическое решение. В 2024 г. грядет 300-летие основания АН и уже началась подготовка к этому юбилею – много исследований и публикаций появляется и появится, потому отложим эту тему.

Летом 1934 г. ФИАН и МИАН вместе с АН СССР переехали в Москву. ФИАН занял здание на Миуссах, которое строили с 1912 г. на пожертвования [10] для физической лаборатории Петра Николаевича Лебедева (08.03.1866-14.03.1912). Создатель первой в России научной физической школы, профессор Московского университета (1900-1911), П.Н.Лебедев, как и его коллега

А.Г.Столетов, не был членом АН, а в 1912 г. его во второй раз выдвинули на Нобелевскую премию за гениальные достижения по исследованию светового давления. Внезапная смерть (46 лет) не позволила ему получить престижную награду. С 18.12.1934 ФИАН носит имя П.Н.Лебедева.

По рекомендации [18] академика (12.01.1929) Ивана Матвеевича Виноградова (14.09.1891-20.03.1983), Дважды Героя Социалистического Труда (1945, 1971), М.В.Келдыш избран академиком в Отделении технических наук АН СССР по специальности «математика, механика». «Незаменимый» М.В.Келдыш стал лидером и «Главным математиком» страны – проводил работы и отвечал за направления по «прикладной математике». Через два дня 02.12.1946 молодого академика назначают Начальником Реактивного научно-исследовательского института (РНИИ, НИИ-1 МАП) – впервые математик стал руководителем технического института! Это был первый шаг к покорению космоса. В 1951 г. в МИАН организован первый отдел прикладной математики [4]. С 1951 г. М.В.Келдыш как Главный математик – председатель «математической секции» в НТС (Постановление СМ СССР № 1552-774 от 09.05.1951), в которую входили И.Г.Петровский, С.Л.Соболев, Н.Н.Боголюбов. А.Н.Тихонов, С.А.Лебедев и инженеры Ю.Я.Базилевский, М.А.Лесечко, – отвечал за математику, расчеты и ЭВМ в стратегических «Трех проектах» [4].

Успехи М.В.Келдыша и его коллективов были столь впечатляющими, что началась борьба разных ведомств за М.В.Келдыша при организации или института прикладной математики или вычислительного центра при Первом Главном Управлении (ПГУ) [4]. Благодаря деятельности МИАН советская математика уже занимала одно из первых мест в мировой науке [27]. Руководители АН СССР и ученые категорически возражали против изъятия указанной группы математиков из системы АН. Всю необходимую для ПГУ работу в области математических вычислений АН могла провести без разрушения МИАН. Решающим было мнение М.В.Келдыша (42 лет) и победили ученые АН СССР и МАТЕМАТИКА – после длительных согласований, чтобы не разрушать МИАН, для выполнения «Трех проектов» на основе «новых технологий» (прикладная математика, расчеты, ЭВМ) в 1953 г. было принято историческое стратегическое решение [3]: Распоряжение СМ СССР № 6111-рс об образовании Отделения прикладной математики Математического института АН СССР, г. Москва. Кремль, 18.04.1953. Сов. Секретно (рассекречено):

«1. Образовать в Математическом институте им. В.А.Стеклова Академии наук СССР Отделение прикладной математики на базе расчетно-математических бюро, руководимых академиками Петровским и Келдышем, и вычислительного бюро Геофизического института, руководимого чл.-кор. Академии наук СССР Тихоновым.

3. Назначить директором Отделения прикладной математики Математического института им. В.А.Стеклова Академии наук СССР, на правах директора института, акад. Келдыша М.В. и заместителем директора – чл.-кор. Академии наук СССР Тихонова А.Н., освободив его от работы в Геофизическом институте Академии наук СССР.

4. Обязать Первое главное управление при Совете Министров СССР (т. Завенягина) передать Академии наук СССР для Отделения прикладной математики Математического института им. В.А. Стеклова:

а) бывшее здание Физического института Академии наук СССР по 3-й Миусской улице в г. Москве с имеющимся оборудованием по состоянию на 1 апреля 1953 г., а также наличный состав работников расчетно-математических бюро;»

В 1953 г. на Миуссы переехали расчетные бюро М.В.Келдыша, И.Г.Петровского, А.Н.Тихонова, коллективы К.А.Семендяева, И.М.Гельфанда, А.А.Дородницына, А.А.Ляпунова и др. сотрудники, которые составили ядро ОПМ МИАН (секретный п/я 2287, «Институт Келдыша») [35]. В 1966 г. после смерти С.П.Королева, когда М.В.Келдыш выступил с речью на его похоронах на Красной пл. и стал всему миру известен как Главный теоретик космонавтики, в соответствии с Постановлением Президиума АН СССР от 08.07.1966 № 465-010 ОПМ МИАН СССР преобразовано в Институт прикладной математики АН СССР (ИПМ АН СССР). Указом Президиума Верховного Совета СССР от 19.04.1967 № 999-7 ИПМ АН СССР награжден орденом Ленина. В соответствии с постановлением ЦК КПСС и Совета Министров СССР от 17.07.1978 № 691 ИПМ АН СССР присвоено имя М.В. Келдыша.

М.В.Келдыш – первый математик Президент Академии наук (19.05.1961-19.05.1975) – развернул науку широким фронтом во всех областях знаний и, как следствие, математика внедрялась в разные сферы приложений. Вторым математиком и последним Президентом АН СССР

(16.10.1986-17.12.1991) был Г.И.Марчук. Третий математик Президент Ю.С.Осипов (07.07.1936) возглавлял Российскую академию наук (17.12.1991-29.05.2013) до её реформы в 2013 году.

В 1963 г. при М.В.Келдыше в АН впервые образовано Отделение математики, первым академиком-секретарём на протяжении 25 лет (1963-1988) был академик (23.10.1953) Н.Н.Боголюбов (21.08.1909-13.02.1992) – специалист в области математики, механики и теоретической физики, директор МИАН (1983-1988), Дважды Герой Социалистического Труда (1969, 1979), один из ведущих участников «Атомного проекта» – руководил расчетным бюро в КБ-11 (Саров).

7. Заключение

Настоящая публикация – это мой гражданский и профессиональный долг и дань глубочайшей благодарности моим величайшим Учителям – М.В.Келдышу и А.Н.Тихонову. М.В.Келдыш – УЧИТЕЛЬ – ОБРАЗЕЦ СЛУЖЕНИЯ НАУКЕ и Родине, МАТЕМАТИК – ЛЕГЕНДА, уникальная историческая личность ГРАЖДАНИНА и УЧЕНОГО, масштаб достижений и заслуг которого в XX веке никто в мире не смог и не сможет достичь в XXI веке, а тем более превзойти! Как и 100 и 80 лет назад брошен вызов отечественной науке – на вызов будет дан достойный ответ. Полезно перечитать «Мысли о современном значении истории знаний» (1926) и важно вспомнить актуальные слова В.И.Вернадского: «История науки является в такие моменты орудием достижения нового», а также соображения директора Государственного Радиевого института в дополнение к Отчету института (1927) о талантах: «Происходит, с моей точки зрения, безумная трата самого дорогого достояния народа – его талантов. А между тем эти таланты никогда не возобновляются непрерывно. И даже если бы оказалось, что процесс их создания в нашем народе еще длится, всегда одни личности механически не могут быть заменены другими. Надо использовать в данный момент то, что дала сейчас нам жизнь – великий дар прошлых поколений».

Создание 70 лет назад в СССР первого научного академического Института прикладной математики – это безусловно было гениальное стратегическое решение на многие века, которое позволило «прикладную математику» сформировать как самостоятельное направление в математике и вывести на уровень производительной силы: ни одна интеллектуальная проблема в «цифровой реальности» без математики не решается. И.В.Курчатов: «В любом деле важно определить приоритеты. Иначе второстепенное, хотя и нужное, отнимет все силы и не даст дойти до главного.» Объявлены новые приоритеты до 2030 года: искусственный интеллект (ИИ); современные и перспективные сети мобильной связи; квантовые вычисления; квантовые коммуникации; новое индустриальное ПО; новое общесистемное ПО; системы накопления энергии; водородная энергетика; перспективные космические системы и сервисы; технологии новых материалов и веществ. Из Распоряжения Правительства от 20.05.2023 № 1315-р «Концепция технологического развития до 2030 года»: к 2030 году национальная экономика должна обеспечивать производство высокотехнологичной продукции – чипов и другой микроэлектроники, высокоточных станков и робототехники, авиакосмической техники, беспилотников, лекарств и медицинского оборудования, телекоммуникационной техники и программного обеспечения. Необходимо обеспечить суверенитет и безопасность страны с учетом угроз «надзорного капитализма» [36].

Вспомните слова Си Цзиньпина В.В.Путину в Кремле на прощание 21.03.2023: «Мы двигаем перемены, которых не было 100 лет». В.В.Путин ответил: «Согласен». Мир сейчас переживает трансформации, которые будут влиять на следующие сто лет истории человечества и науки.

Литература

1. Указ Президента РФ от 31.03.2023 № 229 "Об утверждении Концепции внешней политики Российской Федерации". http://kremlin.ru/events/president/news/70811
2. Летопись Московского университета. http://letopis.msu.ru/
3. Распоряжение Совета Министров СССР № 6111-рс об образовании Отделения прикладной математики Математического института АН СССР от 18.04.1953 // Атомный проект СССР: документы и материалы: в 3 т. Т. 2. Атомная бомба. 1945-1954. Кн. 5. М.-Саров, 2005. С. 542-544. http://elib.biblioatom.ru/text/atomny-proekt-sssr_t2_kn5_2005/go,542/

4. Атомный проект СССР: документы и материалы: в 3 т. / Под общ. ред. Л.Д.Рябева. Федер. агентство по атом. энергии, РАН. М.-Саров: Академиздат «Наука», Изд-во МФТИ, 1998-2010. https://search.rsl.ru/ru/record/01000844024; История Росатома. Атомный проект СССР. Электронная библиотека http://elib.biblioatom.ru/sections/0201/

5. Ракетные войска стратегического назначения. Библиотека РВСН. Исторические документы: РВСН и ракетостроение (1945-1967). https://rvsn.info/library_main.html.l

6. Архивы Российской академии наук. https://arran.ru/?q=ru/aran

7. Информационная система «Архивы Российской академии наук». Персональный состав РАН. https://isaran.ru/?q=welcome; https://www.isaran.ru/?q=ru/persostav

8. Страницы памяти. ИПМ им. М.В. Келдыша РАН. https://keldysh.ru/memory/

9. Лазарев П.П. Физическій Институтъ Научнаго Института // УФН. 1918. Т. 1, вып. 1. С. 54-66. https://ufn.ru/ru/articles/1918/1/e/

10. Вавилов С.И. Физический кабинет - Физическая лаборатория - Физический институт Академии наук СССР за 220 лет // УФН. 1946. Т. 28, вып. 1. С. 1-50. / Доклад, читанный 12 июня 1945 года на торжественном заседании Учёного совета Физического института Академии Наук СССР имени П.Н.Лебедева по поводу 220-летнего юбилея Института (1725-1945). М.: Изд-во Академии наук СССР, 1945. 74 с. https://ufn.ru/ru/articles/1946/1/a/

11. Десять лет Института физики и биофизики НКЗ (1919-1929). М.: Институт физики и биофизики, 1929. 88 с. https://search.rsl.ru/ru/record/01009201043

12. Лазарев П.П. Исторический очерк развития точных наук в России в продолжение 200 лет // УФН. 1999. Т. 169, № 12. С. 1351–1361 / Речь на Торжественном заседании Академии Наук, посвященном празднованию 200-летнего юбилея Российской Академии Наук в Москве 13 сентября 1925 г. https://ufn.ru/ru/articles/1999/12/i/

13. Луначарский А.В. К 200-летию Всесоюзной Академии наук // Новый мир. 1925. № 10. С. 99-112. / Доклад на Торжественном заседании Академии Наук, посвященном празднованию 200-летнего юбилея РАН в Москве 13 сентября 1925г. («Наследие А.В.Луначарского») http://lunacharsky.newgod.su/articles/k-200-letiu-vsesouznoj-akademii-nauk/

14. Борев Ю.Б. Луначарский А.В. М.: Издательство «Молодая гвардия», 2010. 304 с. (Серия «Жизнь замечательных людей»). https://search.rsl.ru/ru/record/01004629881

15. Стеклов В.А. Математика и её значение для человечества. Берлин: Гос. Изд-во РСФСР, 1923. 137 с. https://search.rsl.ru/ru/record/01002578252

16. Исторический очерк структуры Математического института им. В.А.Стеклова РАН / К 75-летнему юбилею МИАН. Биографический словарь-справочник. М.: Янус-К, 2009. С. 10-25. https://www.mathedu.ru/text/chleny_ran_v_matematicheskom_institute_steklova_2009/p0/

17. Ченцов Н.Н. Всемирно известный, всемерно засекреченный // Наука и жизнь. 1991. № 2. С. 102-107.

18. Келдыш М.В. Творческий портрет по воспоминаниям современников. М.: Наука, 2001. 416 с. (Издано при поддержке РФФИ)

19. Сушкевич Т.А. Главный Теоретик М.В.Келдыш и Главный Конструктор космонавтики С.П.Королев – покорители космоса // Современные проблемы дистанционного зондирования Земли из космоса. 2011. Т. 8, № 1. С. 9–25. http://jr.rse.cosmos.ru/article.aspx?id=819

20. Сушкевич Т.А. УЧИТЕЛЬ – легенда-математик академик Келдыш Мстислав Всеволодович: уроки НТР в XX-м веке, авиация, космос, атом, математика и компьютеры / Презентация доклада на Международной конференции «Суперкомпьютерные дни в России», 23-24 сентября 2019 года, Москва. https://keldysh.ru/memory/keldysh/legend.pdf

21. Губарев В.С. Три звезды Героя: знания и страсти. Несколько страниц из жизни великого ученого нашей Родины М.В. Келдыша // Земля и Вселенная. 2021. № 1. С. 86-100 (начало).

https://www.elibrary.ru/item.asp?id=44870432; Земля и Вселенная. 2021. № 2. С. 79-92 (окончание). https://www.elibrary.ru/item.asp?id=44873285

22. Алферов Ж.И. Моя партия – Академия наук России // В мире науки. 2019. №4. 15.03.2023. https://scientificrussia.ru/articles/
23. Марчук Г.И., Алдошин С.М., Григорьев А.И., Козлов В.В. Эпоха М.В.Келдыша: выводы и уроки. 17.02.2011. http://sergey-sharakshane.narod.ru/Eppokha_Keldysha.pdf; https://www.ras.ru/news/shownews.aspx?id=6531c71e-d91f-44a2-bd7e-812a1405cffc
24. Вернадский В.И. Мысли о современном значении истории знаний / Доклад, прочитанный в I заседании Комиссии по истории знаний АН СССР 14.XI.26. Л.: Изд-во АН СССР, 1927. 17 с. (Труды Комиссии по истории знаний). https://search.rsl.ru/ru/record/01009016798
25. Ильин В.А., Малинецкий Г.Г., Моисеев Е.И., Попов Ю.П., Самарский А.А. Творец современной прикладной математики (К 100-летию со дня рождения академика А.Н.Тихонова) // Вестник РАН. 2006. Т. 76, № 9. С. 813-836. https://www.keldysh.ru/ANTikhonov-100/ANT100-index.html
26. Тихонов А.А. Пафнутий Львович Чебышев: человек науки на службе России (к 200-летию со дня рождения) // Чебышевский сборник. 2021. Т. 22, вып. 3. С. 405-422.
27. Лаврентьев М.А. Пути развития советской математики // Изв. АН СССР. Сер. матем. 1948. Т. 12, вып. 4. С. 411-416 / Доклад на сессии ОФМН АН СССР 28.10.1947, 30-летие Великой Октябрьской социалистической революции. http://www.mathnet.ru/php/archive.
28. Келдыш М.В. Речь на открытии Конгресса // Труды Международного Конгресса математиков. Москва, август 1966 г. М.: Издательство «Мир», 1968. С. 5-6.
29. 250 лет Академии наук СССР (1724-1974). Документы и материалы юбилейных торжеств. М.: Наука, 1977. 585 с. https://search.rsl.ru/ru/record/01007711338
30. Российская академия наук. 1724-1999 годы. Материалы юбилейных торжеств. М.: Наука, 1999. 264 с. (К 275-летию Академии наук) https://search.rsl.ru/ru/record/01000626452
31. Российская академия наук. История и современность. Краткий очерк. М.: Наука, 1999. 272 с. (К 275-летию Академии наук). https://search.rsl.ru/ru/record/01000609328
32. Уставы Российской академии наук. 1724-1999. М.: Наука, 1999. 287 с. (К 275-летию Академии наук). https://search.rsl.ru/ru/record/01004346035
33. Магницкий Л.Ф. Арифметика. М.: Синодальная типография, 1703. 662 с. arifmetika-magn1703.djvu; https://sovietime.ru/matematika/arifmetika-1703
34. Ломоносов М.В. Российская грамматика. СПб.: Тип. Императорской академии наук, 1755. 210 с. Оригинал на сайте ГПИБ России: http://elib.shpl.ru/ru/nodes/66257-lomonosov-m-v-rossiyskaya-grammatika-spb-1755; ФЭБ: http://feb-web.ru/feb/lomonos/texts/lo0/lo7/lo7-3892.htm; СПб.: Еврошкола, 2016. 210 с. https://search.rsl.ru/ru/record/01008938791
35. Попов Ю.П. Доклад на торжественном заседании, посвященном 50-летию ИПМ им. М.В.Келдыша РАН 09.10.2003. https://www.keldysh.ru/grants/rffi//50_years/report.htm; ИПМ им. М.В. Келдыша РАН 50 лет. М.: ФГУП Изд-во Известия, 2003. 62 с.; https://search.rsl.ru/ru/record/01002374119; https://keldysh.ru/pages/anniver/introduction/introduction.htm
36. Зубофф Шошана. Эпоха надзорного капитализма: битва за человеческое будущее на новых рубежах власти. М.: Издательство Института Гайдара, 2022. 781 с. (Пер. с англ.: Zuboff Shoshana. The Age of surveillance capitalism. New Yorl: Public Affairs, cop. 2019.) https://search.rsl.ru/ru/record/01010922587

Определение ходовых качеств судна с использованием современных методов численного моделирования

М.П. Лобачев, А.А. Рудниченко

ФГУП «Крыловский государственный научный центр»

В работе рассматривается расчет ходкости транспортного судна с использованием современных методов численного моделирования и суперкомпьютерных технологий. Под ходкостью судна понимается его способность двигаться в окружающей среде с заданной скоростью при определенной мощности главных двигателей и соответствующем движителе. Для прогнозирования гидродинамических характеристик корпусов судов и их гребных винтов в натурных условиях методами вычислительной гидродинамики применяется коммерческий пакет Star-CCM+. Характеристики течения вязкой жидкости вокруг корпусов судов и их гребных винтов находятся из решения методом контрольного объёма нестационарных уравнений Рейнольдса, замкнутых полуэмпирической моделью турбулентности. Использованный подход был валидирован на основе заводских ходовых испытаний судов пяти проектов, проведенных и описанных во второй половине XX века. В данной работе на примере одного из этих судов показано, что подобные расчеты возможны только с привлечением дополнительной эмпирической информации и обязательным является использование высокопроизводительных вычислительных систем. При использовании 20 процессоров Intel Xeon E5-2697 v2 время, требуемое для прогноза ходовых качеств судна на одном режиме (без учета парциальных), составляет от 6 до 12 суток. Указано только время выполнения собственно расчетов.

Ключевые слова: ходкость судна, гребной винт, уравнения Рейнольдса, суперкомпьютерные технологии.

1. Введение

Основным источником исходной информации для выполнения проектных работ в задачах корабельной гидродинамики в настоящее время по-прежнему являются экспериментальные исследования, проводимые в опытовых бассейнах, аэродинамических и кавитационных трубах. Каждый из этих экспериментов имеет свои особенности, однако, общим для них является то, что они проводятся с нарушением законов подобия, в первую очередь по числу Рейнольдса, т.е. по схемам частичного моделирования [1]. Это относится и к прогнозированию ходовых качеств судов, т.е. оценки достижимой скорости хода при заданной мощности энергетической установки. Для оценки влияния не учитываемого при проведении модельного эксперимента критерия подобия разрабатываются инженерные методики, основанные в основном на статистической обработке натурных испытаний судов. Однако эти методики не являются универсальными и включают так называемые корреляционные надбавки, которые к тому же принимаются своими в каждом гидродинамическом центре. Некоторый анализ современного состояния проблемы приведен в работе [2].

В настоящее время за счет использования современных методов численной гидродинамики, а также высокопроизводительных вычислительных технологий (суперкомпьютерных технологий), появилась возможность создания более универсальных методик. Основой таких методик является использование для моделирования течений вязкой жидкости уравнений Рейнольдса, замкнутых полуэмпирическими моделями турбулентности (в данной работе k-ω SST) [3,4]. Так как в отличие от экспериментальных исследований при численном моделировании возможно рассмотрение судна натурного масштаба, то проблема частичного моделирования исчезает. Нет так называемой проблемы масштабного эффекта при пересчете данных модельных испытаний на натурные условия. Для натурного судна расчет выполняется при одновременном моделировании как по числу Фруда, так и по числу Рейнольдса. Это дает возможность корректного прогнозиро-

вания ходовых качеств судов новых типов, для которых еще не накоплены статистические данные для разработки инженерных методик оценки масштабного эффекта. Недостатком таких методик являются только необходимость привлечения значительных вычислительных ресурсов и некоторый недостаток фактических данных по характеристикам турбулентности при больших числах Рейнольдса (10^9). Для достаточно широкого круга задач судостроения результаты расчетов для натурных условий представлены в [5].

Однако только с использованием численных методов моделирования физических процессов данная задача не может быть решена. Ситуация на самом деле такая же, как при разработке полуэмпирических моделей турбулентности: задача оказывается незамкнутой, отсюда и вытекают проблемы даже не при решении задачи, а при ее постановке. Требуется привлечение весьма специфической эмпирической информации. Именно об этих проблемах и требуемых вычислительных ресурсах при различной степени детализации геометрии, данная статья. Валидация получаемых результатов численного моделирования выполняется сопоставлением с результатами заводских ходовых испытаний.

2. Проблемы при постановке задачи о численном определении характеристик ходкости судна

Проблемы при использовании численного моделирования для определения ходкости судна делятся на три типа.

Во-первых, на различных стадиях выполнения проектных работ не вся геометрия судна может быть задана с достаточной степенью точности. Так геометрия надводной части обычно оказывается полностью известной на достаточно поздних стадиях. Сопротивление надводной части определяется в данном случае по статистическим данным с учетом площади поперечного сечения надводной части судна. Выступающие части (ВЧ) на подводной части судна: скуловые кили, рули активных успокоителей качки, кронштейны (на двухвальных судах), решетки на отверстиях подруливающих устройств устанавливаются на основе анализа обтекания подводной части корпуса, т.е. когда геометрия подводной части «голого корпуса» уже полностью определена. Эти данные могут быть получены либо по физическим экспериментам на модели корпуса судна, либо на основании расчетов. Причем часто и при испытаниях моделей в опытовом бассейне некоторые ВЧ не ставятся, а их сопротивление определяется на основе статистической информации.

Во-вторых, существуют элементы, моделирование обтекания которых требует определенных, иногда весьма больших, ресурсов, но в тоже время их вклад в сопротивление относительно невелик. К таким элементам в первую очередь относятся сварные швы. Их много, а суммарный вклад в сопротивление порядка трех процентов. Сюда же относятся такие элементы как протекторная защита, разного рода ниши, вырезы, оформление выдвижных устройств, например, лага. Для этих элементов еще и присутствует весьма большая неопределенность в геометрии и положении. Так что и при проведении физических экспериментов они не учитываются, а их сопротивление включается в «корреляционную надбавку».

В-третьих, существует проблема учета влияния шероховатости обтекаемых поверхностей на сопротивление. На начальных этапах проектирования информация о том, какова будет шероховатость обтекаемых поверхностей просто отсутствует. Какой именно краской будет окрашена поверхность судна, качество обработки поверхности гребного винта, наличие или отсутствие катодной защиты гребного винта на момент сдачи судна и т.д.? Все это порой неизвестно, хотя в ряде случаев, как например для гребных винтов, регламентируется ГОСТами. На более поздних стадиях проектирования эта информация уже появляется, однако учет шероховатости все равно возможен только с привлечением эмпирической информации.

Таким образом и при численном моделировании в ряде случаев, в том числе и на разных стадиях выполнения проектных работ, требуется внесение эмпирических поправок для учета элементов геометрии корпуса судна, которые либо еще не определены, либо их численный учет нецелесообразен. Все эти поправки могут быть приняты по материалам, приведенным в [6,7,8]. Что касается учета шероховатости, то либо соответствующие надбавки вводятся в сопротивление по процитированным выше источникам, либо возможен ее учет непосредственно при численном моделировании. Для конкретных типов технологической шероховатости поверхностей корпусов

судов и гребных винтов эмпирическая информация, достаточная для оценки влияния шероховатости на сопротивление трения, была получена в основном в начале 80-х годов прошлого столетия в ЦНИИ им. акад. А.Н. Крылова [9,10]. Но требуется корректное использование этой информации при численном моделировании, что с использованием коммерческих пакетов инженерного анализа (Fluent, CFX, Star-CCM+ и др.) весьма непростая задача. Это обусловлено тем обстоятельством, что в коммерческих кодах используется концепция так называемой «эквивалентной песочной шероховатости», которая на самом деле не описывает технологическую шероховатость. В Star CCM+ формально есть «функция шероховатости», позволяющая описывать неавтомодельную шероховатость, к которой и относится технологическая шероховатость. Но, как показывает наш опыт ее использования для описания шероховатости характерной для судостроения [9,10], она не дает правильных результатов, и эту часть в Star-CCM+ требуется исключать специальным приемом (см. раздел 4).

3. Объект исследования

При разработке «Методики расчета ходкости судов на основе суперкомпьютерных расчетов гидродинамических характеристик судов и их движителей» в рамках ОКР «Виртуал-ТК»[1] численные исследования проводились для судов пяти проектов: пр. 1552 «София», пр. 1594 «Зоя Космодемьянская», пр. 1585 «Герои Панфиловцы», пр. 12990 «Победа» пр. 13476 «Frio Las Palmas». В качестве исследуемых режимов были выбраны режимы сдаточных испытаний этих судов (заводских ходовых испытаний – ЗХИ). Совместные испытания в опытовом бассейне моделей корпуса судна и гребного винта обычно называют самоходными испытаниями, этот же термин часто используется в том числе и для численного моделирования в натурных условиях.

Данные проекты были выбраны для проведения валидации разрабатываемой методики так как хотя все эти суда достаточно старых проектов, однако по ним были полные комплекты как геометрии корпусов и гребных винтов, соответствующих построенным, так и результатов сдаточных испытаний. В настоящей работе в качестве примера будут использованы результаты расчетов для судна проекта 1594 «Зоя Космодемьянская». В рамках других работ по ОКР «Виртуал-ТК», а именно «Разработка технологий оптимизации параметрических моделей поверхности корпуса и движителя», были выполнены дополнительные расчеты и испытаны в опытовом бассейны модели корпуса судна и гребного винта этого проекта [11]. Так что данный объект в рамках темы «Виртуал-ТК» является наиболее исследованным.

На рисунке 1 показан вид на кормовую оконечность и гребной винт судна пр. 1594. Силуэт (боковая проекции CAD-модели) корпуса судна пр. 1594 представлен на рисунке 2. Основные исходные данные для моделирования самоходных испытаний, соответствующие условиям проведения сдаточных испытаний судна пр. 1594, приведены в таблице 1.

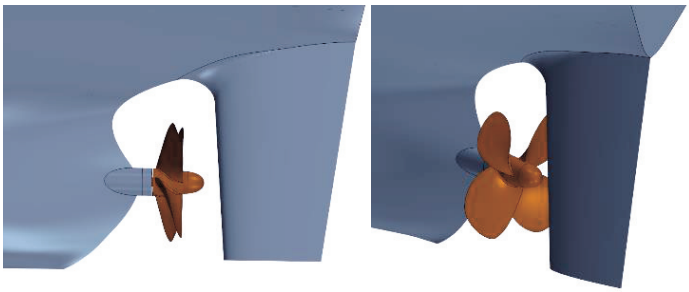


Рис. 1. CAD-модель корпуса судна и гребного винта в сборе пр. 1594

Рис. 2. CAD-модель корпуса судна пр. 1594

[1] Государственный контракт № 19411.1810190019.09.014 от 11 ноября 2019 г. Минпромторг России

Таблица 1. Исходные данные для моделирования самоходных испытаний

Величины	пр. 1594
Скорость движения судна V_S, уз.	14,75
Частота вращения ГВ N, об/мин	115,0
Плотность воды ρ, кг/м^3	1014
Динамическая вязкость воды μ, Па·с·10^{-3}	1,096
Длина по ватерлинии L, м	208,14
Осадка носом T_f, м	5,80
Осадка кормой T_a, м	9,05
Потребляемая мощность P_S, кВт	7967,7

При построении CAD-модели корпуса судна в натурном масштабе кроме руля и гребного винта иные выступающие части (ВЧ) не учитывались (скуловые кили, подруливающие устройства и пр.), т.к. в настоящее время нет возможности получения по ним полной информации. Также не моделировалась надводная часть судов. Поэтому при проведении численного моделирования величины сопротивления этих не моделируемых элементов принимались в виде надбавок к сопротивлению, как это принято в традиционной методике пересчета результатов модельных испытаний на натуру [6]. Что, впрочем, соответствует и валидации традиционной методики, представленной в [12] на основе 8 проектов, включая и 5 проектов, упомянутых в этом разделе. Оценка изменения размерности требуемой расчетной сетки и, соответственно, потребных вычислительных и временных ресурсов, при численном моделировании этих элементов выполнена на основе расчетов для других проектов с учетом аналогичных элементов.

4. Используемые численные методы

Для прогнозирования гидродинамических характеристик корпусов судов и их гребных винтов в натурных условиях методами вычислительной гидродинамики в ФГУП «Крыловский государственный научный центр» (далее КГНЦ) применяется коммерческий пакет Star-CCM+. Пакет располагает собственным сеточным генератором, позволяющим строить полиэдральные или гексаэдральные сетки с призматическими слоями вблизи границ. Характеристики течения вязкой жидкости вокруг корпусов судов и их гребных винтов находятся из решения методом контрольного объёма нестационарных уравнений Рейнольдса, замкнутых полуэмпирической моделью турбулентности. При определении положения судна относительно свободной поверхности (поверхности раздела вода-воздух) используются 2 степени свободы: всплытие и дифферент. Все задачи решаются в нестационарной постановке со схемами второго порядка для дискретизации по пространству и времени.

В изложенных далее численных исследованиях для замыкания уравнений Рейнольдса в качестве модели турбулентности использована k-ω SST модель Ментера [13] в высокорейнольдсовой постановке. Возможность и даже необходимость использования высокорейнольдсовых вариантов модели турбулентности обусловлена большими числами Рейнольдса. Так для корпусов судов характерные значения чисел Рейнольдса, построенных по длине корпуса и скорости судна, больше 10^9. В этом случае протяженность логарифмического участка в профиле скорости доходит в координатах стенки y^+ до нескольких тысяч [14]. Соответственно наиболее рациональным представляется выдерживание для первой расчетной точки при построении призматического слоя трехмерной расчетной сетки принятие для корпуса судна $500 < y^+ < 1000$. Для гребного винта и руля эти величины, в силу меньших локальных чисел Рейнольдса этих объектов, будут меньше: $40 < y^+ < 120$ для гребного винта и $80 < y^+ < 160$ для руля.

Использование полуэмпирической модели турбулентности k-ω SST для расчета интегральных характеристик как корпуса, так и гребного винта достаточно оправдано, хотя валидация их применения проводилась в основном для модельных чисел Рейнольдса (для корпуса модели судна – порядка 10^7) [3,4,15]. Для определения локальных характеристик в ряде случаев необходимо привлечение для моделирования турбулентности более ресурсоемких вихреразрешающих моделей типа DES [16].

Экспериментальные данные по замерам характеристик турбулентного пограничного слоя на шероховатых поверхностях свидетельствуют о том, что влияние шероховатости приводит к изменению распределения скоростей, особенно в зоне, прилегающей непосредственно к твердой стенке. В универсальных координатах (u+, y+) логарифмический участок профиля скорости при этом смещается вниз параллельно самому себе и может быть описан функцией вида [17]:

$$u^+ = \frac{1}{\kappa} \cdot \ln y^+ + B(h^+), \quad (1)$$

где $u^+ = u/u_\tau$; $y^+ = yu_\tau/\nu$; u–продольная составляющая скорости внутри пограничного слоя; $u_\tau = \sqrt{\tau_{w/\rho}}$–динамическая скорость; τ_w – сила трения на поверхности; ρ – плотность среды; y – координата по нормали к твердой стенке; ν – коэффициент кинематической вязкости; h – характерный высотный параметр шероховатости; $h^+ = hu_\tau/\nu$ – число Рейнольдса по высотному параметру шероховатости; κ - константа Кармана; $B(h^+)$ –функция шероховатости.

Нередко выражение (1), особенно в современной англоязычной литературе, включая документацию по Star-CCM+, представляют в несколько ином виде:

$$u^+ = \frac{1}{\kappa} \cdot \ln(E' y^+), \quad (2)$$

где: $E' = \frac{E}{f}$ и (3)

f – функция шероховатости.

При принятых в Star-CCM+ значениях константы Кармана κ=0,42 и E=9,0 (для гладкой поверхности) константа B в (1) для гладкой поверхности принимает значение $B = \frac{1}{\kappa} \cdot \ln(E) = 5{,}23149$.

Имеющиеся экспериментальные данные показывают, что зависимость функции шероховатости B (или f) от числа Рейнольдса шероховатости h+ является сугубо индивидуальной для каждого вида геометрии шероховатости [18,19].

Для учета влияния шероховатости поверхности корпуса и гребного винта в используемом для расчетов программном комплексе StarCCM+ была выполнена новая обработка испытаний круглых дисков с различными типами шероховатости, проведенных в ЦНИИ им. акад. А.Н. Крылова в период с 1979 по 1984 годы [9,10]. На основе этой обработки были получены зависимости коэффициентов в функции шероховатости, использованной в Star-CCM+ [20], от времени достройки судов для лакокрасочного покрытия корпуса, а также для катодного осадка, покрывающего поверхность гребного винта при наличии протекторной защиты.

На основании [10] для шероховатости поверхности гребных винтов принят характерный высотный параметр шероховатости Rz=18,75 мкм (Ra=3 мкм). Коэффициенты в функции шероховатости для поверхности гребного винта покрытого катодным осадком приняты: B=0,65; C=0,22. Для корпусов судов с лакокрасочными покрытиями высотный размер шероховатости зависит как от величины начальной шероховатости, так и времени пребывания в воде [9]. Для головного судна пр. 1554 «Зоя Космодемьянская» с учетом времени достройки принято Rz=107,0 мкм. Коэффициенты в функции шероховатости для поверхности корпуса судна приняты B=0,8721; C=0,0487. Для адекватного учета шероховатости, проявляющейся не в автомодельном режиме, в Star CCM+ пришлось принять следующие коэффициенты $R^+_{smooth} = 0{,}1$ и$R^+_{rough} = 1{,}0$, что существенно отличается от традиционно используемых [20]. При этом функция шероховатости якобы описывающая в Star CCM+ неавтомодельную шероховатость исключалась из рассмотрения как некорректно описывающая шероховатость поверхностей корпуса и гребного винта. Обоснование данного приема выходит за пределы настоящей статьи, однако отметим, что соответствующая работа была проведена.

5. Не моделируемые элементы

На различных этапах проектирования судна детализация его геометрии может быть совершенно различной. В этом случае сопротивление элементов, по которым нет информации можно оценивать по статистическим данным, например, по приведенным в [6,7,8]. В данной работе при анализе натурных испытаний надбавки на не моделируемые элементы принимались аналогичными тем, что использовались при пересчете модельных испытаний в опытовом бассейне при

разработке рассматриваемых проектов. Соответствующие надбавки в виде коэффициентов сопротивления приведены в таблице 2. В общем случае коэффициенты сопротивления отдельных элементов C_i рассчитываются по следующей формуле:

$$C_i = \frac{2 \cdot R_i}{\rho \cdot V_S^2 \cdot S}, \quad (4)$$

где R_i – надбавка на сопротивление, полученная по результатам модельных испытаний, н; ρ – плотность воды, кг/м^3; V_S – скорость судна, м/сек; S – смоченная поверхность голого корпуса, м^2. Для воздушного сопротивления коэффициенты приводятся к плотности воды и площади подводной части голого корпуса для единообразия при выполнении расчетов. В таблице 2 также приведены относительные вклады этих элементов в полное сопротивление $\Delta C_i = C_i/C_T \cdot 100\%$ и увеличение размерности расчетной сетки для их учета ΔN, если их учет выполнять используя численное моделирование. Номенклатура не моделируемых элементов шире, чем представленные в таблице 2. Здесь приведены только те элементы, которые были в данной задаче.

Таблица 2. Надбавки на не моделируемые геометрические элементы $C_i \cdot 10^3$ и увеличение размерности расчетной сетки ΔN

Не моделируемые в расчете элементы	$C_i \cdot 10^3$	ΔC_i, %	ΔN, млн
Воздушное сопротивление, C_{AA}	0,06	1,9	0,3
Скуловые кили, C_{AP}	0,08	2,6	4,0
Сварные швы, C_W	0,1	3,2	267,0
Шероховатость корпуса* (без сварных швов) C_{rough}	0,296	9,5	-

* В случае учета при численном моделировании через «функцию шероховатости» равна нулю.

Надбавка на сварные швы в таблице 2 выделена из общей надбавки на шероховатость, так как при численном моделировании влияние сопротивления шероховатости лакокрасочного покрытия корпуса может быть включено в расчет. В то же время сварные швы нецелесообразно учитывать при проведении численного моделирования в силу малости их размеров и большого количества. Для обычных морских транспортных судов с незначительным ледовым усилением, пр. 1594 относится именно к таким, высота сварных швов h_W не превышает 3,5 мм, а ширина – 18 мм. Число Рейнольдса по высоте шва, аналогичное тому, что построено для характерного высотного параметра шероховатости ($h_W^+ = h_W u_\tau/\nu$) $h_W^+ \approx 720 - 750$, т.е. примерно соответствует высоте первой расчетной точки (раздел 2). Таким образом первую к твердой поверхности ячейку потребуется по высоте делить в 15-20 раз, чтобы адекватно описать обтекание шва. И хотя при этом нет необходимости переходить к низкорейнольдсовым моделям турбулентности (750/20=37,5), сопряжение мелких ячеек у сварного шва как по вертикали, так и по продольному направлению, приведет к неоправданному увеличению размерности сетки. Это, соответственно, вызовет увеличение времени расчета, а также возрастание трудностей при построении расчетных сеток. Сгущение на 1 поперечный шов в районе цилиндрической вставки (шов максимальной длины) добавляет порядка 9 млн. ячеек. В таблице 2 приведено увеличение размерности расчетной сетки при учете всех поперечных сварных швов. Продольные швы не учитывались, так как их вклад в увеличение сопротивления существенно меньше, чем поперечных, за счет обтекания их в основном по направлению шва.

6. Использованные расчетные сетки

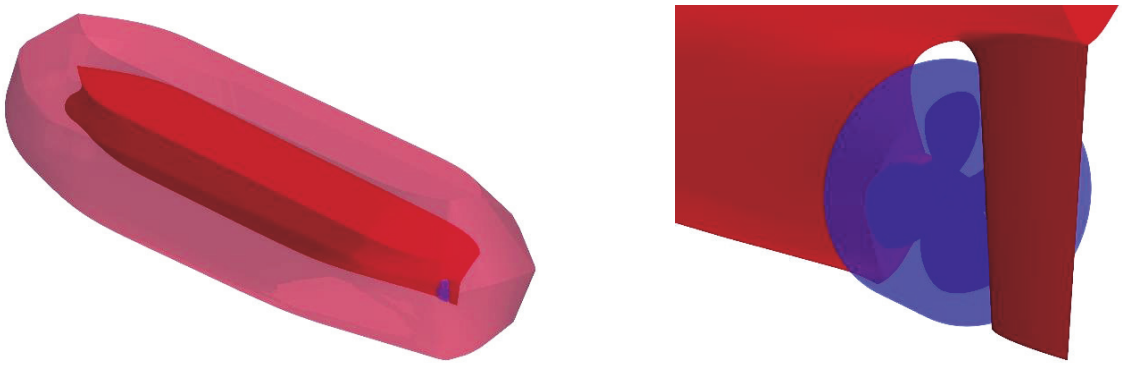
Важной частью подготовки к проведению расчетов является построение границ внешней расчетной области, построение границ внутренней расчетной области с накладывающейся сеткой (сетки химера или overset-сетки) и выделение областей (регионов) под вращающиеся элементы (ГВ). Именно на границах неподвижной внешней области в неподвижной глобальной системе координат задаются граничные условия для всей расчетной области в целом. Внутренняя расчетная область связана с корпусом судна и перемещается относительно расчетной сетки во внешней области с изменением положения судна в зависимости от сил на него действующих.

Вращающаяся сетка связана с гребным винтом и поворачивается вместе с ним. Относительно этой сетки положение гребного винта неизменно.

В качестве внешней расчетной области используется параллелепипед такого размера, чтобы расстояние между носовой оконечностью корпуса судна и входной границей области составляло примерно 2 длины, расстояние между кормовой оконечностью судна и выходной границей расчетной области составляло примерно 3 длины, расстояния между бортами судна и боковыми границами расчетной области составляло 1,5 длины, а расстояния между ОП корпуса судна и нижней и верхней границами корпуса судна составляло 1 длину корпуса судна. На границе раздела сред вода-воздух (свободная поверхность) выполняется сгущение расчетной сетки по вертикали. Фрагмент расчетной сетки вокруг корпуса, включающий часть внешней области представлен на рисунке 3.

Рис. 3. Фрагмент расчетной сетки вокруг судна (пр. 1594)

Построение внутренней расчетной области под накладывающиеся сетки (overset) необходимо для моделирования перемещения корпуса судна в пространстве. При этом расчетная сетка из внутренней области накладывается поверх расчетной сетки из внешней области, делая наложенные ячейки расчетной сетки во внешней области неактивными. В рамках работы использовался вариант формы границ внутренней области – «форма-кокона», условно повторяющую форму корпуса судна с ВЧ и заданным расстоянием границы по нормали от элементов корпуса судна. В области вероятного нахождения свободной поверхности производится сгущение расчетной сетки по вертикали. Граница накладывающейся расчетной сетки приведена на рисунке 4 а. Для расчета обтекания гребного винта выделяется область цилиндрической формы (в виде барабана). Данная расчетная сетка вырезается из накладывающейся сетки. По поверхностям сопряжения расчетных сеток формируется интерфейс, на котором реализуется технология скользящих сеток (slidingmesh). Пример приведен на рисунке 4 б.



а б

Рис. 4. Границы накладывающейся расчетной сетки (overset) – а и барабана (slidingmesh) – б.

Размерности расчетных сеток N этих областей для рассмотренного судна приведены в таблице 3. Тогда согласно данным таблицы 2 по дополнительному увеличению размерности сеток

при учете не моделируемых элементов, при учете надводной части и скуловых килей размерность окончательной расчетной сетки увеличится всего на 18,7 %. Увеличение незначительное, но информации по этим элементам не было, поэтому их сопротивление (4,5 % от полного сопротивления) учитывалось надбавками. Другое дело сварные швы. Их учет увеличил бы использованную расчетную сетку почти в 12 раз. В задачах машиностроения точно также отбрасывают болты, заклепки и имеющиеся сварные швы.

При выполнении данной работы руль моделировался в расчетах (рисунок 1), так как он оказывает влияние не только на сопротивление, но и на работу гребного винта, которое сложно учитывать за счет статистических данных. К тому же геометрия руля была представлена в исходной информации. Увеличение размерности за счет моделирования руля составило 0,5 млн. ячеек. Только здесь эта величина входила в общую размерность расчетной сетки (внутренняя область – раздел 4). Сопротивление руля составило 2,5 % от полного сопротивления.

Таблица 3. Размерности расчетных сеток

Области	N
Все области (суммарная)	23,0
Внутренняя область (overset)	10,9
Область вокруг гребного винта (slidingmesh)	3,4

7. Определение «точки свободного самохода»

При проведении моделирования движения натурного судна под воздействием гребного винта возможно решение прямой задачи, когда задается скорость движения судна, как это было сделано, например, в работе [21], для моделирования в задачах управляемости и качки. Однако не всегда это целесообразно с точки зрения потребных для решения задачи вычислительных ресурсов. Так если, как указывалось выше, часть информации отсутствует (наводная часть судна, скуловые кили и т.п.) предпочтительным является решение обратной задачи, когда задается скорость набегающего на судно потока. В этом случае требуется для каждого значения скорости из заданного набора определить потребную мощность (частоту вращения гребного винта) при которой сумма сил, действующих на систему корпус-винт, будет равна нулю или силе, соответствующей не заданным элементам. В терминологии «теории корабля» это– «точка свободного самохода».

По результатам численного моделирования определяется суммарная тянущая сила Z (сила на гаке), действующая в продольном направлении на все, входящие в геометрическую CAD-модель, элементы системы судно – гребной винт. Обозначим через R_S все силы, действующие на корпус судна и все выступающие элементы на корпусе в проекции на продольную ось, через T_X – продольную составляющую упора, создаваемого гребным винтом. Тогда в «точке свободного самохода» для $Z=R_S+T_X$ должно выполняться равенство Z=0.

В том случае, когда не вся информация по судну задана (геометрические элементы, шероховатость), то сумма сил, действующих на систему корпус-винт, должна быть равна сумме сил R, действующих на неучтенные элементы.

$$R = \sum_{i=1}^{N} R_i, \quad (6)$$

где N – число надбавок на не моделируемые элементы. Т.е. в этом случае должно выполняться равенство Z=R.

Пересчет надбавок на сопротивление из таблицы 3 в размерные величины выполняется по формуле:

$$R_i = C_i \rho V_S S/2, \quad (7)$$

Для определения этих условий движения в режиме постоянной скорости изменяется частота вращения гребного винта. Частота вращения подбирается методом последовательного приближения упора к сопротивлению судна (с учетом сопротивления не заданных элементов).

На Рисунке 5 приведены зависимости сопротивления судна R_S, продольной составляющей упора гребного винта T_X и тянущей силы Z от времени. На данном графике знак силы R_S приведен с учетом ее направления, противоположного упору. Далее везде будем пользоваться его аб-

солютным значением, учитывая, что это сопротивление. Наличие предотрывного, а в ряде случаев отрывного течения в кормовой оконечности судна, приводит к колебаниям сопротивления судна и, как следствие, колебаниям силы Z. Приведенный на рисунке 5 временной интервал соответствует прохождению примерно 14,5 длин судна. Низкочастотные колебания сопротивления моделей судов наблюдается также и в опытовых бассейнах при проведении самоходных испытаний моделей и особенно заметно при буксировочных испытаниях. Объяснение других особенностей графика в разделе 6.

Наличие длинноволновой составляющей сил на корпусе судна увеличивает требуемое время счета как для выхода на повторяющееся решение, так и для определения средних величин, что достаточно для определения ходовых качеств судна. Практика выполнения таких расчетов свидетельствует, что без привлечения высокопроизводительных вычислительных (суперкомпьютерных) технологий решение указанного класса задач в сроки, приемлемые для практического использования, невозможно. Особенно это актуально если кроме определения ходовых качеств судна требуется анализ нестационарных сил. Потребное время расчетов в этом случае существенно возрастает из-за необходимости накопления более длительной выборки. Данные об этом представлены в разделе 6.

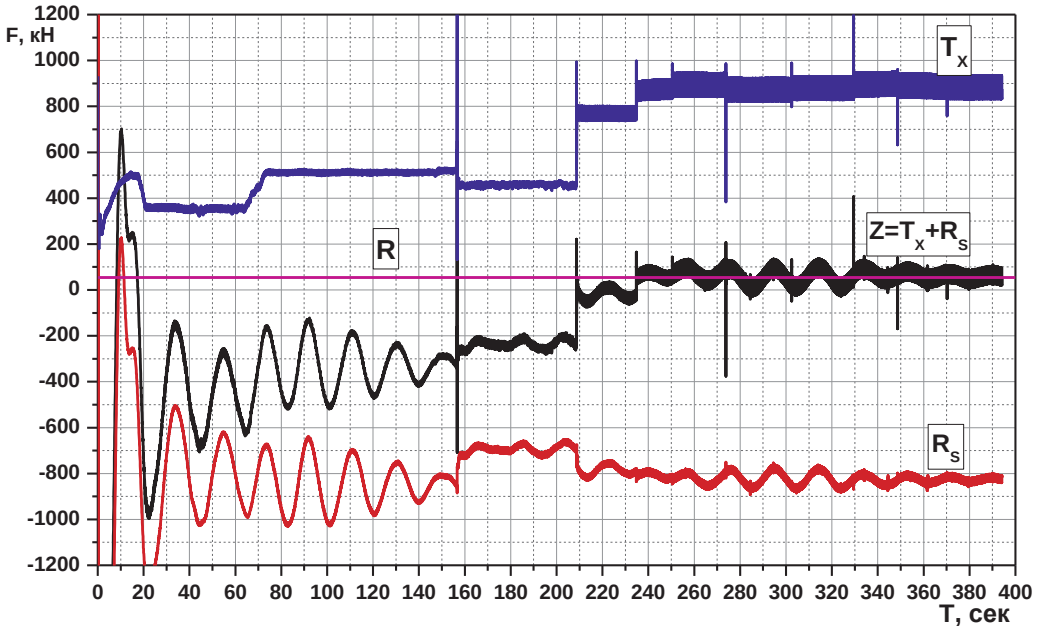


Рис. 5. Зависимость сопротивления судна R_S, упора гребного винта T и тянущей силы Z от времени.

8. Ресурсы, потребные для выполнения расчетов

Размерность расчетной сетки для пр. 1594 без учета не моделируемых элементов составила 23 млн. ячеек. Для других подобных объектов (тема «Виртуал-ТК») размерности расчетных сеток были близкими (± 2 млн.). Так что эту размерность можно считать типичной (приемлемой) для морских транспортных судов длиной 150-250 м. При выполнении расчетов включена возможность перемещения корпуса судна по двум осям, используется модель VOF для моделирования свободной поверхности.

Для расчетов этой и подобных задач обычно выделялось одно блейд-шасси, содержащее 10 двухпроцессорных узлов на процессорах AMD Opteron 6174. Частота оперативной памяти 1067 МГц. В ряде случаев использовались блейд-шасси с 10-ю двухпроцессорными узлами на Intel Xeon E5-2697 v2. Частота оперативной памяти 1600 МГц. В обоих случаях объем оперативной памяти на узел составлял 64 Гб. Разница во времени проведения расчетов на этих процессорах для данного класса задач колеблется от 2,1 до 2,6 раза. Оба сегмента объединены высокоскоростной сетью с низкими задержками Mellanox QDR InfiniBand с топологией «толстое дерево». Также выполнялись расчеты на двух блейд-шасси с 10-ю двухпроцессорными узлами на IntelXeon E5-2697 v2.

В задачах при которых требуется моделирование взаимодействия гребного винта и корпуса судна обычно для ускорения «выхода на решение» расчеты выполняются с последовательно уменьшающимся шагом по времени. Этот же прием позволяет быстрее получить правильное положение корпуса судна и волновую картину. Именно изменением шага по времени объясняются основные площадки на графиках сил рисунка 5. При правильном алгоритме уменьшения шага по времени такой прием может дать выигрыш по времени выхода на сошедшееся решение до 14 раз. В данном случае шаг по времени изменялся в процессе численного моделирования от соответствующего 45 ° поворота гребного винта до соответствующего 2 ° поворота гребного винта («2 °»). В представленных оценках ресурсов расчет, проводимый до перехода на шаг по времени «2 °» назван подготовительным. Далее проводилось моделирование с шагом по времени «2 °» в течение 40 оборотов гребного винта, затем с тем же шагом 2 ° – 40 оборотов с измененной частотой вращения («досчет»). При изменении шага по времени на одном-двух шагах наблюдается скачок сил, однако не приводящий к «раскачке» решения. Эти скачки, а также быстрый переход на другой уровень, хорошо видны на рисунке 5. Небольшая корректировка частоты вращения для выхода на требуемый уровень сил нужна практически всегда, хотя иногда удается обойтись без нее. 40 оборотов – величина характерная для судна данного проекта (пр. 1594), для других проектов эта величина может быть другой. Подготовительный расчет занимает примерно 30-35% времени на проведение численного моделирования при правильной последовательности временных шагов. Если не использовать указанный прием, то суммарное время расчета может увеличится в 4,5 – 5,5 раз, в котором доля подготовительного расчета будет составлять 80-88%. Далее при необходимости дополнительной корректировки частоты вращения ГВ осуществляется дополнительный «досчет». Сведения о потребных вычислительных ресурсах для пр. 1594 приведены в таблице 4. Из приведенных данных видно, что при увеличении числа узлов Intel Xeon в 2 раза время расчета сократилось в 1,544 раза.

Количество дополнительных шагов («досчетов») оказывается разным для различных задач и к тому же зависит от опыта расчетчика. Автоматизировать этот процесс на данный момент не представляется возможным. В таблице 4 приведены времена счета как при отсутствии необходимости выполнять корректировку частоты вращения ГВ, так и для пр. 1594, для которого потребовалось выполнить 2 корректировки. Для определения средних величин, что собственно требуется при расчете ходовых качеств, достаточно осреднения по одному периоду длинноволновых колебаний с окончательно принятой частотой вращения ГВ. Для судна пр. 1594 один период длинноволновой составляющей сопротивления корпуса составляет порядка 40 оборотов гребного винта (по первоначальной оценке, во время проведения расчетов). Однако если требуется проведение спектрального анализа нестационарных сил, то согласно [22] необходима длина выборки не менее 10 периодов самой маленькой частоты процесса. Соответственно время расчета для накопления такой выборки существенно возрастает. Последний столбец таблицы 4 как раз содержит эти времена для случая, когда дополнительная корректировка частоты вращения ГВ не требуется. Оценка периода низкочастотных колебаний в 40 оборотов гребного винта была сделана при выполнении расчетов и исходя из этого значения определялось время выполнения расчета. При обработке после выполнения расчетов было определено более точное значение периода низкочастотных колебаний. Он оказался чуть больше 35 оборотов гребного винта.

Таблица 4. Потребные вычислительные ресурсы (пр. 1594).

Количество и тип расчетных узлов	Время расчета, сутки	Время подготовительного расчета, сутки	Время дополнительного «досчета», сутки	Суммарное время расчета, сутки	Суммарное время расчета**, сутки
10 узлов на AMD Opteron 6174	15	4,7	3,5	19,7*/26,7	51,2
10 узлов на Intel Xeon E5-2697 v2	6,9	2,1	1,6	9*/12,2	23,4
20 узлов на Intel Xeon E5-2697 v2	4,5	1,4	1,0	5,9*/7,9	14,9

* – при отсутствии необходимости «досчета».

** – при расчете с установленной частотой вращения ГВ с учетом требований [22].

Вопросам эффективности распараллеливания различных задач корабельной гидродинамики посвящена работа [23]. Точный аналог рассмотренной в данной работе задачи в 2014 году не исследовался, с похожими размерностями расчетной сетки рассматривались задачи обтекания корпуса судна (3,8 млн ячеек) и поворотной колонки с работающем гребным винтом в натурных условиях (13 млн. ячеек). Для первой задачи ускорение счета по сравнению с использованием одного ядра для 24 ядер (один узел) составляло 14,8 раза, для 240 ядер (10 узлов) – 129,9 раз, для 480 ядер (20 узлов) – 206,3 раза. Для второй задачи соответственно: для 24 ядер (один узел) – 14,6 раз, для 240 ядер (10 узлов) – 132,2 раза, для 480 ядер (20 узлов) – 247,7 раза. Рассмотренная в данной работе задача распараллеливается примерно также.

Существенное увеличение времени счета по сравнению с представленными в [23] похожими задачами обусловлено длительностью процесса изменения характера обтекания корпуса судна под влиянием работы гребного винта. Причем на эффективность распараллеливания этот процесс не оказывает влияния, так как это просто последовательное повторение одного и того же расчета для постепенного «стягивания» пограничного слоя с корпуса судна. Здесь как раз на ускорение «работает» последовательность временных шагов от грубых к мелким. Данные приведены для сегмента кластера на Intel Xeon E5-2697 v2. Для организации параллельных вычислений пакет Star-CCM+, как и большинство коммерческих пакетов, использует mpi. Основным выводом работы [23] был вывод о том, что для большинства задач корабельной гидродинамики для сохранения эффективности распараллеливания не следует использовать менее 10 тыс. ячеек на одно ядро. Т.е. для данной задачи можно было бы увеличить количество используемых узлов для сокращения времени счета, но по [23] не более, чем до 96 (9,6 блейд-шасси). Формально возможно весьма значительное ускорение. Однако в реальной практике это не так. Все-таки это предельное значение, при котором увеличение количества ядер еще эффективно. Выбор количества используемых узлов определяется в основном срочностью работы, допустимым временем решения задачи, количеством лицензий на различное ПО, задействованное в конкретный момент времени на кластере и т.п. В частности, текущая загрузка при выполнении данной работы не позволяла использовать больше двух блейд-шасси с Intel Xeon E5-2697 v2 на задачу. В принципе это коррелирует с приведенными в 2016 году в докладе В.Б. Бетелина[2] данными об использовании суперкомпьютеров в компании «Airbus». Имея в 2016 году 3 суперкомпьютера в списке Топ-500 суммарной пиковой производительностью 1,5 Pflop/s большинство критически важных для «Airbus» приложений выполнялось на 256 или 512 ядрах. Хотя для решения некоторых задач как в «Airbus», так и в КГНЦ приходится использовать существенно большие ресурсы (в каждой организации по своим возможностям).

9. Результаты расчетов

В таблице 5 приведены результаты численного моделирования применительно к условиям сдаточных испытаний головного судна пр. 1594 – «Зоя Космодемьянская». Последняя строка таблицы 5 показывает погрешность, определяемую неполным соответствием заданной частоты вращения гребного винта при численном моделировании точке «свободного самохода». Задание допустимой погрешности – компромисс между требуемой точностью определения потребной мощности и затрачиваемыми на выполнение расчетов машинными и временными ресурсами. Эта величина существенно зависит от стадии выполняемых проектных работ. Представленные в таблице 5 данные – результаты осреднения за 35 оборотов гребного винта. Именно такое осреднение обеспечивает учет низкочастотных колебаний сил, реализующихся на элементах корпуса.

Потребная мощность рассчитывается на основе рассчитанного момента с учетом коэффициента полезного действия валопровода, в данном случае 98 %: $P=\frac{2\pi n \cdot Q}{0.98}$.

Здесь n – частота вращения гребного винта, 1/сек; Q – рассчитанный крутящий момент гребного винта, н·м.

[2] ЦИПР (Цифровая Индустрия Промышленной России) 07-10 июня 2016 г. г. Иннополис, секция «Аппаратные архитектуры, прикладное программное обеспечение и подготовка специалистов в области суперкомпьютерного моделирования для задач промышленности», доклад «Суперкомпьютерные технологии в промышленности России».

Таблица 5. Результаты расчета для условий сдаточных испытаний

Проект	1594
Головное судно	«Зоя Космодемьянская»
V_S, уз.	14,75
N, об/мин.	114,35
Упор ГВ T, кН	879,8
Момент ГВ Q, кН·м	648,5
Потребная мощность P, квт	7924
Продольная составляющая упора ГВ Tx, кН	878,9
Сопротивление элементов корпуса R_S, кН	825,6
Суммарная сила на гаке Z, кН	53.3
Суммарная сила от надбавок* R, кН	53.7
Z-R, кН	-0.4
Z-R, в % от T	**-0.05**

* Рассчитывается по коэффициентам таблицы 3.

На рисунке 6 представлены результаты определения силы на корпусе R_S с учетом силы на не моделируемых элементах R примерно за один период низкочастотной составляющей (R_S+R). Красной линией показаны результаты, отфильтрованные по частоте 0,5 герц.

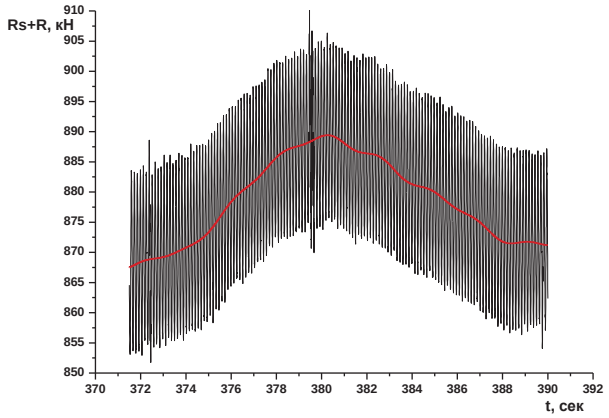


Рис. 6. Временная зависимость суммы сил, действующих на корпус судна пр. 1594 (красная линия – результат фильтрации по частоте 0,5 герц).

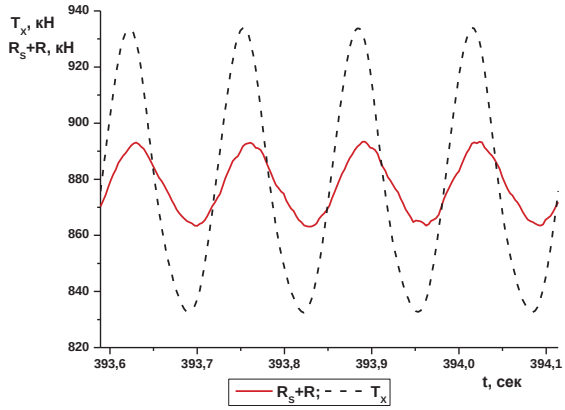


Рис. 7. Изменение сил на корпусе и гребном винте судна пр.1594

На рисунке 7 представлены колебания продольной составляющей упора и суммарной силы на корпусе за один оборот гребного винта. Четыре пика обусловлены тем обстоятельством, что на судах пр. 11594 были установлены 4-х лопастные гребные винты. Для возможности сопоставления упора и сил на корпусе приведен сумма модуля собственно рассчитанных сил на корпусе и сил от не моделируемых в расчете элементов (R_S+R). В данном случае это скуловые кили, надводная часть судна и сварные швы.

Колебания упора гребного винта (примерно ±51 кН) не имеют значимой низкочастотной составляющей и обусловлены работой гребного винта в неоднородном потоке за корпусом судна. При этом для данного проекта они больше чем колебания сил на корпусе, обусловленных взаимодействием с гребным винтом (примерно ±15 кН) и низкочастотная составляющая сил на корпусе (примерно ±24 кН). На ряде проектов размах низкочастотной составляющей сил на корпусе оказывается больше, чем колебания упора гребного винта.

На данный момент основным при собственно валидации является сопоставление с результатами сдаточных испытаний и прогнозом потребной мощности при использовании традиционных методик. Результаты такого сопоставления приведены в таблице 6 в виде так называемых корреляционных коэффициентов. Корреляционный коэффициент мощности – отношение потребной мощности P_S, полученной по результатам сдаточных испытаний, к потребной мощности, полученной расчетом P: $C_P=P_S/P$. Тоже и для частоты вращения гребного винта: $C_n=n_s/n$. Представлены результаты, полученные как с использованием численных методов, так и с использованием традиционной методики, принятой в Крыловском государственном научном центре [2]. Результаты, обозначенные как КГНЦ 1998, были выполнены с использованием традиционной методики ([2] с добавлениями по [12]) на основе старых испытаний, выполненных не с моделью спроектированного гребного винта, а с так называемым «магазинным» гребным винтом, близким по характеристикам к спроектированному. Такое ранее допускалось с целью экономии средств. Расчеты, обозначенные КГНЦ 2021 выполнялись на основе модельных испытаний по традиционной методике [2,12] с моделями корпуса судна и гребного винта, изготовленными в рамках темы «Виртуал-ТК» по данным соответствующим построенному судну.

Таблица 6. Сравнение корреляционных коэффициентов C_P и C_n

№ проекта	Расчет		КГНЦ 1998		КГНЦ 2021	
	C_P	C_n	C_P	C_n	C_P	C_n
1594	1,006	1,006	1,038	1,067	1,010	1,022

Представленные результаты свидетельствуют, что погрешности при использование численных методов для расчета ходовых качеств судов оказываются на уровне традиционных методик. При этом погрешности при использовании не спроектированной модели гребного винта, а «магазинного» гребного винта даже с близкими характеристиками, оказывается большими, чем при использовании численного моделирования.

Данная работа демонстрирует возможности современных численных методов в сочетании с высокопроизводительными вычислительными технологиями и необходимость при их использовании для решения практических задач обязательного привлечения эмпирической информации. О последнем многие расчетчики и разработчики программного обеспечения, к сожалению, забывают. Хотя без учета этого обстоятельства область возможного использования любого ПО сужается. Вопросы валидации (за исключением таблицы 6) в основном были опущены. Однако без привлечения хотя бы некоторых данных по валидации не обойтись. Поэтому в таблице 7 приводятся средние значения $\overline{C}$ и среднеквадратичные отклонения σ_C коэффициентов C_P и C_n рассчитанные для 5 судов, упомянутых в разделе 2. При расчетах по традиционной методике для всех судов, кроме пр. 1594, использовались результаты старых испытаний. Однако сама методика была использована с учетом последних поправок [12]. Детали численного моделирования ходовых качеств этих судов, включая назначение надбавок на не моделируемые в расчете элементы, не соответствуют тематике данной конференции и поэтому опущены.

Формально численное моделирование оказалось даже немного точнее, чем традиционная методика, но 5 судов – еще недостаточная выборка для категоричных утверждений. Однако уже сейчас ясно, что численное моделирование стало вполне рабочим инструментом, причем весьма перспективным инструментом. Можно считать, что задача определения ходовых качеств транспортных судов на основе численного моделирования, поставленная в КГНЦ как весьма отдаленная перспектива еще в середине 80-х годов прошлого столетия, по крайней мере в первом приближении решена. Понадобилась только смена 3,5 поколений вычислительной техники (за ½ поколения принят переход от i286 к Pentium Pro) и 4 поколений программного обеспечения, а также обязательное предметное знание: какой эмпирической информацией должно быть дополнено численное моделирование.

Таблица 7. Сравнение средних значений корреляционных коэффициентов C_P и C_n по 5 проектам

	Расчет		КГНЦ 2021	
	C_P	C_n	C_P	C_n
$\overline{C}$	1,0150	1,0010	1,0148	0,9942
σ_C	0,0193	0,0163	0,0163	0,0114
σ_C, %	1,91	1,63	1,61	1,15

10. Заключение

Представленные в данной работе результаты показывают, что современные численные методы моделирования течений вязкой жидкости позволяют решать такие практические задачи, как прогнозирование ходовых качеств судов, т.е. оценки достижимой скорости хода при заданной мощности энергетической установки. Точность получаемых результатов для судов традиционных типов сопоставима с точностью прогноза по результатам модельных испытаний и использовании соответствующих методик пересчета данных модельного эксперимента на натурные условия. Для решения поставленной задачи с использованием численного моделирования требуется привлечение эмпирической данных по сопротивлению отдельных элементов корпуса судна, которые на данном этапе проектирования еще не заданы или их численное моделирование нецелесообразно, а также данных для учета влияния реальной шероховатости поверхностей корпуса и гребного винта. При этом и для традиционной методики требуется привлечение тех же эмпирических данных с той разницей, что для ряда элементов, например, сварных швов, в модельных условиях определение их сопротивления просто невозможно и всегда используются статистические данные.

Преимущества использования численных методов для определения ходовых качеств судна:

1. Возможность получения прогноза на начальных стадиях проектирования, когда еще нерационально изготовление модели судна и проведение модельных экспериментов в опытовом бассейне.
2. Возможность корректного прогнозирования ходовых качеств судов новых типов, для которых еще не накоплены статистические данные для разработки инженерных методик пересчета данных модельных испытаний на натурные условия. Т.е. численное моделирование оказывается более универсальным (привлекаемая эмпирическая информация по сопротивлению отдельных элементов слабо зависит от типа судна).

Недостатки использования численных методов:

1. Потребность в высокопроизводительных вычислительных ресурсах.
2. Необходимость выполнения работ высококвалифицированными специалистами в области численного моделирования и обладающими к тому же предметными знаниями.

При этом использование численных методов на ранних стадиях проектирования обязательно должно сопровождаться испытанием модели окончательного варианта разрабатываемого проекта. Отказ от проведения физического моделирования, как показывает практика, может приводить к принятию ошибочных технических решений. Будущее за гибридным моделированием, сочетающим как численный, так и физический эксперимент.

Литература

1. Баренблатт Г.И. Подобие, автомодельность, промежуточная асимптотика: теория и приложения к геофизической гидродинамике. Л.: Гидрометеоиздат, 1982. 256 с.
2. Борусевич В.О., Каневский Г.И., Капранцев С.В., Клубничкин А.М., Лобачев М.П. Развитие методов прогнозирования ходовых качеств транспортных судов, Труды Крыловского государственного научного центра, выпуск 4(382), 2017, С. 21-27. DOI: 10.24937/2542-2324-2017-4-382-21-28
3. Лобачев М.П., Овчинников Н.А., Пустошный А.В. Опыт использования современных методов численной гидродинамики. Академик А.Н. Крылов. // К 150-летию со дня рождения: Сборник статей. ФГУП «Крыловский государственный научный центр», СПб, 2013, с.15-32

4. Таранов А.Е. Определение локальных и интегральных гидродинамических характеристик контейнеровоза в цифровом бассейне. // Труды Крыловского государственного научного центра. 2019; 3(389): 73-82. DOI: 10.24937/2542-2324-2019-3-389-73-82
5. Лобачев М.П., Овчинников Н.А., Таранов А.Е., Чичерин И.А. Масштабный эффект в задачах судостроения – современные возможности оценки. // Труды международной конференции «Суперкомпьютерные дни в России» 26-27 сентября 2016 г. Москва, с. 233-244. http://2016.russianscdays.org/files/pdf16/232.pdf
6. Справочник по теории корабля / Под ред. Я.И. Войткунского. Л.: Судостроение, 1985. Т 1. 768с
7. Кацман Ф.М., Пустошный А.Ф., Штумпф В.М. Пропульсивные качества морских судов. Л., Судостроение, 1972, 403 с.
8. Турбал В.К., Шпаков В.С., Штумпф В.М. Проектирование обводов и движителей морских транспортных судов. Л. Судостроение, 1983, 304 с.
9. Душина Л.Н., Каневский Г.И., Штумпф В.М., Щередин В.Н. Влияние шероховатости наружной обшивки корпуса на сопротивление транспортных судов. Сборник «Гидродинамика транспортных судов» ЦНИИ им. Крылова 1981.
10. Каневский Г.И., Лобачев М.П. Исследование влияния состояния поверхности лопастей гребных винтов на их гидродинамические характеристики. Вопросы судостроения. 1984 вып. 41, с. 57-62.
11. Лобачев М.П., Новоселов В.Н., Поляков Ю.Н., Рудниченко А.А., Сайфуллин Т.И., Таранов А.Е. Валидация методики оптимизации пропульсивного комплекса транспортного судна. Труды Крыловского государственного научного центра, 2021, 4(398), 68-80. DOI: 10.24937/2542-2324-2021-4-398-68-80
12. Александров С.А., Каневский Г.И. Исследование влияния масштабного эффекта на гидродинамические характеристики и ходовые качества двухвального грузопассажирского судна. Труды Крыловского государственного научного центра. 2022; 3(401): 19–27. DOI: 10.24937/2542-2324-2022-3-401-19-27
13. Menter F.R. Two equation eddy-viscosity turbulence models for engineering applications // AIAA Journal. 1994; 32: 1598–605.
14. Каневский Г.И., Штумпф В.М. Измерения и сопоставительный анализ характеристик пограничного слоя на корпусе крупнотоннажного танкера в натурных и модельных условиях // Труды НТО СП. Л., 1977. Вып. 249. С. 47–54.
15. Лобачев М.П, Таранов А.Е. Определение гидродинамических характеристик моделей гребных винтов с учетом ламинарно-турбулентного перехода // Труды Крыловского государственного научного центра. 2015. Вып. 90 (374). С. 47-54.
16. Лобачев М.П. Исследование особенностей течения вязкой жидкости в кормовой оконечности судов с полными обводами. // Труды Крыловского государственного научного центра. 2013; 78(362): С. 5-28.
17. Шлихтинг Г. Теория пограничного слоя. М., «Наука», 1974, 712 с.
18. Монин А.С., Яглом А.М. Статистическая гидромеханика. Ч. 1. М., Наука, 1965, 641 с.
19. Ротта И.К. Турбулентный пограничный слой несжимаемой жидкости. Л., Судостроение, 1967, 232 с.
20. Официальный сайт компании Siemens [Электронный ресурс]. URL: https://www.plm.automation.siemens.com/global/ru/products/simcenter/STAR-CCM.html
21. Блищик А.Э., Таранов А.Е. Численное моделирование динамики судна в задачах управляемости и качки. Труды Крыловского государственного научного центра. 2018; 2(384): С. 29-38. DOI: 10.24937/2542-2324-2018-2-384-29-38
22. Харкевич А.А. Спектры и анализ. Государственное издательство технико-теоретической литературы, Москва, 1957 г., 235 стр.
23. Овчинников Н.А. Эффективность использования суперкомпьютера при решении задач вычислительной гидродинамики. // Суперкомпьютерные технологии в науке, образовании и промышленности: Альманах. М.: Издательство Московского университета, 2014. С. 39-52.

Оптимизация моделирования процессов горения методом табличной аппроксимации решений уравнений химической кинетики

П.П. Введенский[1], Е.В. Михальченко[1], И.С. Яковенко[2]

[1]Московский государственный университет имени М.В. Ломоносова,
[2]Объединённый институт высоких температур РАН

В работе рассматривается подход к оптимизации расчетов химической кинетики для моделирования динамики реагирующих потоков. С целью увеличения эффективности моделирования реагирующих течений с учетом детальных схем химической кинетики предлагается использование табличной аппроксимации соответствующей системы обыкновенных дифференциальных уравнений, характеризующей химические превращения. Представлена реализация данного подхода на примере задачи моделирования динамики горения предварительно перемешанной водородно-воздушной смеси. Проведено сравнение результатов, получаемых по предложенной методике и с использованием методов прямого решения уравнений химической кинетики.

Ключевые слова: горение газовых смесей, химическая кинетика, численное моделирование.

1. Введение

Моделирование процессов горения газовых смесей крайне важно для многих отраслей промышленности. Знание фундаментальных характеристик этого процесса необходимо для создания эффективных систем пожаро- и взрывобезопасности, разработки перспективных двигателей и энергетических систем. Основу для проведения детализированных расчетов динамики горения газовых смесей составляет система уравнений Навье-Стокса для многокомпонентных газовых смесей, описывающая движение вязкой, сжимаемой среды с учетом процессов переноса тепла, диффузии отдельных компонент, кинетики химических превращений. При этом, использование детальных схем химической кинетики окисления топлив, описывающих изменение концентраций элементарных компонент, зачастую имеет принципиальную важность для получения корректных результатов моделирования процессов горения.

Необходимость в детальном воспроизведении химических превращений элементарных компонент смеси в ходе химических реакций сопряжена с высокой трудоемкостью решения задач горения с использованием вычислительных систем. Моделирование кинетики горения с использованием детальных кинетических механизмов требует решения системы обыкновенных дифференциальных уравнений в каждой ячейке пространственной расчетной сетки на каждом временном шаге. При этом количество уравнений определяется числом элементарных компонент, что значительно затрудняет возможности моделирования топлив на основе углеводородов, где детальные схемы химической кинетики могут включать превращения сотен элементарных компонент. Для проведения расчетов таких топлив используются различные методы редуцирования механизмов химической кинетики, ставящие целью уменьшить число элементарных компонент, исключив из рассмотрения химические процессы, роль которых считается не принципиальной в рамках конкретной задачи. Редуцирование необходимо применять, отталкиваясь от диапазона параметров, в которых протекает процесс горения. Соответственно, упрощенные кинетические механизмы, получаемые в результате редуцирования, обладают сравнительно узкой областью возможного применения, что ограничивает возможность их использования для моделирования сложных динамических течений, где может быть реализован широкий диапазон состояния топливной смеси. Другим перспективным методом к оптимизации моделирования химических превращений для реагирующих течений является аппроксимация решения системы уравнений химической кинетики с использованием предварительно сгенерированного набора табличных данных [1]. В рамках этого подхода, для создания массива данных могут быть использованы

детальные кинетические схемы, что позволяет сохранить их исходную точность воспроизведения параметров горения в широком диапазоне состояний топливной смеси и вместе с тем избежать необходимости проведения трудоемких расчетов соответствующей системы обыкновенных дифференциальных уравнений. В рамках настоящей работы, представлена методика для генерации набора табличных данных и их последующего применения для расчетов динамики горения водородно-воздушных смесей.

2. Математическая модель и вычислительный алгоритм

2.1 Математическая модель динамики реагирующей среды

Для описания динамики реагирующей смеси используется математическая модель, основанная на уравнениях Навье-Стокса, с учетом влияния вязкости, теплопроводности, многокомпонентной диффузии, изменения концентраций отдельных компонент и энерговыделения за счет химических реакций [2]. В общем случае горение представляет собой процесс, связанный с возникновением и распространением в среде акустических колебаний, волн сжатия, ударных волн. Однако, с учетом малых скоростей распространения волн горения в рассматриваемых смесях, в данной задаче эффектами, связанными с взаимодействием фронта пламени с волнами сжатия можно пренебречь, и для описания динамики процесса воспользоваться приближением малых чисел Маха:

Закон изменения массы отдельных компонент смеси:

$$\frac{\partial\left(\rho Y_k\right)}{\partial t}+\frac{\partial}{\partial x_i}\left(\rho\left(v_i+V_{k,i}\right)Y_k\right)=\rho\dot{\omega}_k \qquad (1)$$

Закон сохранения импульса:

$$\frac{\partial\vec{v}}{\partial t}-\vec{v}\times\vec{\omega}+gradH-\tilde{p}\,grad\left(\frac{1}{\rho}\right)=\left(\frac{1}{\rho}\right)grad\,\sigma \qquad (2)$$

Закон сохранения энтальпии:

$$\frac{d\,\rho h_s}{dt}=\frac{\partial\overline{p}}{\partial t}-\sum_{k=1}^{N_c}\dot{\omega}_k\Delta h^0_{f,k}-\frac{\partial}{\partial x_i}\left(\rho\sum_{k=1}^{N_c}h_{s,k}Y_kV_{k,i}\right)-\left(\kappa\left(T\right)\frac{\partial T}{\partial x_i}\right)+\sigma_{ij}\cdot\frac{\partial v_i}{\partial x_i} \qquad (3)$$

Условия совершенного газа:

$$\overline{p}=\rho RT\sum_{k=1}^{N_c}\frac{Y_k}{M_k} \qquad (4)$$

$$dh_s=C_p\left(Y_k,T\right)dT \qquad (5)$$

В уравнениях (1) - (5) ρ – плотность смеси, Y_k - концентрация компонента смеси, v_i - компоненты вектора массовой скорости, $V_{k,i}$ - компоненты вектора скорости диффузии *k*-го компонента, ω_k - изменение массы *k*-го компонента за счет химических реакций, σ_{ij} – компоненты тензора вязких напряжений, p – давление, $\kappa\left(T\right)$ - коэффициент теплопроводности, h_s - удельная энтальпия смеси, $h_{s,k}$ - удельная энтальпия *k*-ой компоненты, $\Delta h^0_{f,k}$ – стандартная энтальпия образования *k*-ой компоненты, T – температура смеси, $\overline{p}$ – статическое давление, M_k -

молярная масса *k*-ой компоненты, C_p – удельная теплоемкость смеси при постоянном давлении, R – универсальная газовая постоянная, $H = \frac{|v|^2}{2} + \frac{p}{\rho}$ - интеграл Бернулли.

Решение системы модельных уравнений проводилось с использованием программного комплекса NRG [3], и реализованного на основе этого комплекса алгоритма расчета реагирующих течений в приближении малых чисел Маха [4]. Алгоритм имеет второй порядок точности по времени и пространству. Размер расчетной ячейки выбирался равным 50 мкм, что соответствует области сходимости задачи. Расчеты проводились в однопоточном режиме с использованием персонального компьютера под управлением операционной системы Windows 10, оснащенного процессором Intel Core i7-4770 с тактовой частотой 3.4 ГГц.

2.2 Математическая модель химической кинетики

В ходе химических реакций в процессе окисления водорода массы компонент воздушно-водородной смеси изменяются согласно следующему дифференциальному уравнению для концентрации компонент:

$$\dot{\omega}_k(T) = \sum_{r=1}^{N_R}\left(k_{rf}(T)\left(v_{rk}^{(p)} - v_{rk}^{(r)}\right)\prod_{s=1}^{N_c}[X]_s^{v_{rs}^{(r)}} + k_{rb}(T)\left(v_{rk}^{(r)} - v_{rk}^{(p)}\right)\prod_{s=1}^{N_c}[X]_s^{v_{rs}^{(p)}} \right) \quad (6)$$

где $k_{rf}(T)$, $k_{rb}(T)$ - скорости протекания r-ой реакции в прямом и обратном направлениях соответственно, $v_{rk}^{(p)}$, $v_{rk}^{(r)}$ - стехиометрический коэффициент k-ой компоненты r-ой реакции в продуктах и реагентах соответственно, $[X]_s$ - молярная концентрация

Скорости протекания реакций определяются уравнением Аррениуса:

$$k_r(T) = A_r T^{\beta_r} \mathrm{e}^{-\frac{E_r}{RT}} \quad (7)$$

Здесь β_r - температурный показатель степени, A_r - предэкспоненциальный множитель, E_r - энергия активации *r*-ой реакции.

Для описания горения заданного состава смеси, важно учитывать детальный химический механизм. В представленной работе используется механизм окисления водородно-воздушной смеси [5], состоящий из 19 реакций для 8 элементарных компонент.

3. Постановка задачи

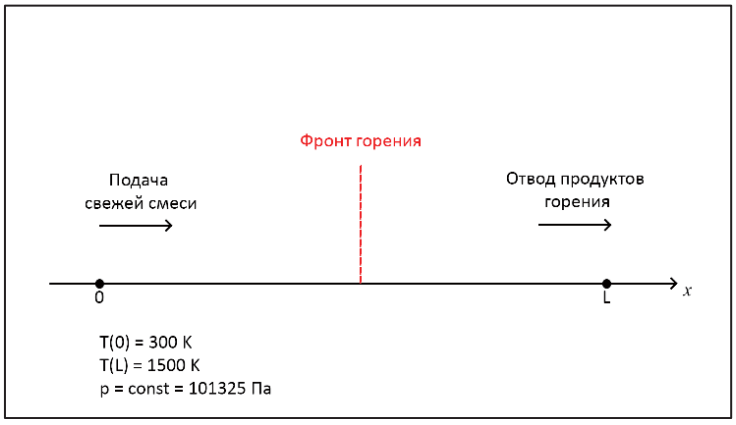


Рис. 1. Схематическое изображение постановки задачи

В задаче рассматривается одномерная камера сгорания длиной L = 50 мм с открытыми границами, изначально разделенная на две части (см. рис. 1). Левая часть заполнена свежей смесью водорода с воздухом при температуре 300 К, правая – продуктами горения при температуре 1500 К. На левой границе области поставлены условия подачи свежей смеси с начальной массовой скоростью потока 1 м/с. В начальный момент времени в области контакта между свежей смесью и продуктами горения происходит воспламенение с последующим формированием фронта горения. Изменение положения фронта пламени определяется по максимуму концентрации радикала HO_2. Скорость подачи свежей смеси динамически меняется со временем таким образом, чтобы компенсировать скорость движения фронта пламени. Таким образом подбирается значение скорости подачи свежей смеси, соответствующее нулевой скорости распространения фронта пламени. Получаемая в результате данной процедуры конфигурация фронта пламени является стационарной, и может быть использована для нахождения табличного решения для стабилизированного горения.

4. Обсуждение результатов

Моделирование выполнялось в два этапа. На первом этапе были проведены расчеты стабилизации волны горения водородно-воздушной смеси с использованием прямого метода решения уравнений химической кинетики с применением метода Гира для систем жестких дифференциальных уравнений, реализованного в библиотеке slatec. Расчёт данной задачи проводился на участке проточного реактора длиной 25 мм. Стабилизированный фронт горения использовался для создания табличных зависимостей величины ω_k от температуры. Данные зависимости применялись для генерации таблицы коэффициентов химической кинетики, которая в дальнейшем использовалась для построения аппроксимированного решения задачи стабилизированного горения на участке реактора длиной 50 мм. Ниже приведены результаты прямого вычисления параметров горения смеси с мольной долей водорода 20%, результаты табличной аппроксимации и их сравнение.

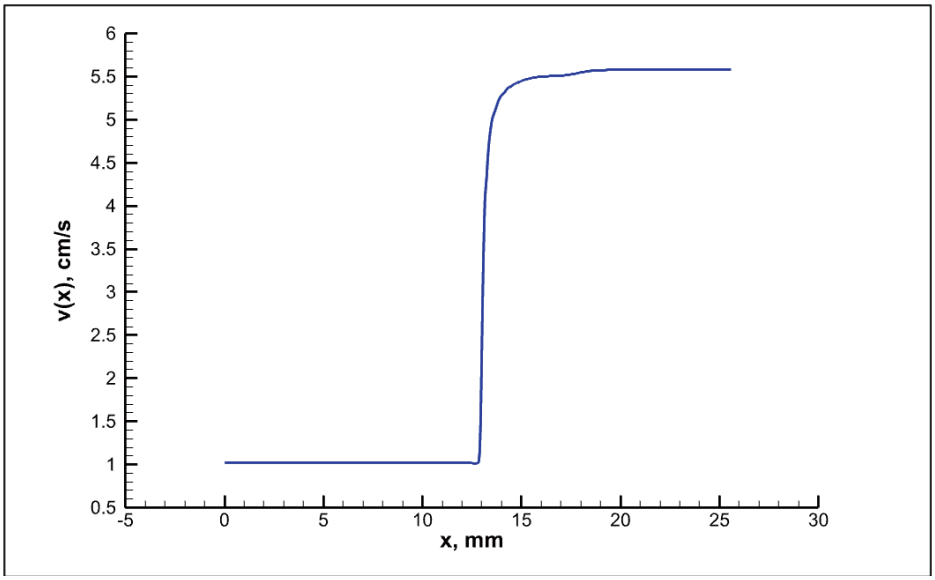


Рис. 2. Профиль массовой скорости среды в результате прямого расчета с использованием библиотеки slatec для смеси с содержанием водорода равным 20 моль-% в момент времени 1000 миллисекунд

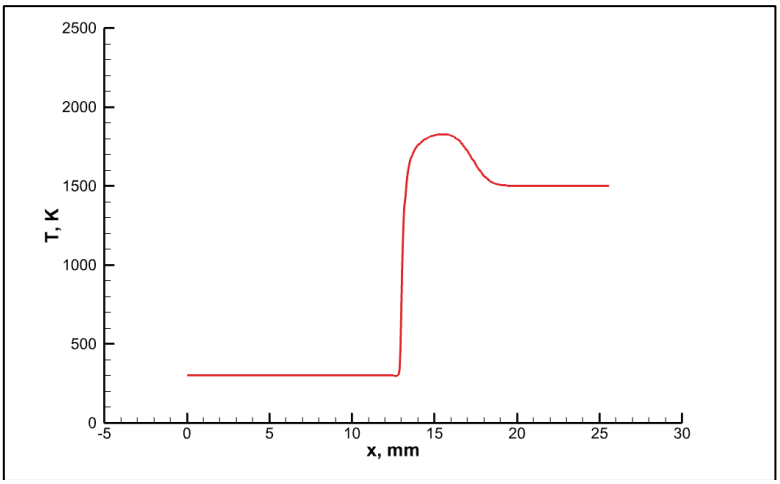


Рис. 3. Профиль температуры среды в результате прямого расчета с использованием библиотеки slatec для смеси с содержанием водорода равным 20 моль-% в момент времени 1000 миллисекунд

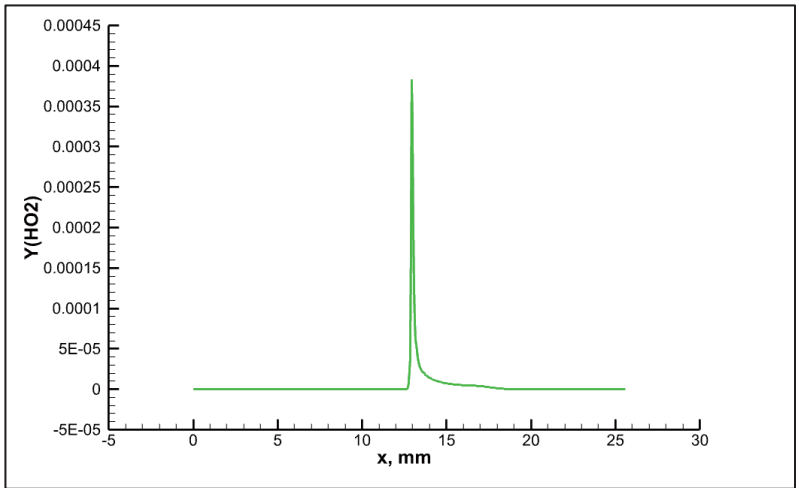


Рис. 4. Профиль распределения массовой концентрации радикала - HO_2 в результате прямого расчета с использованием библиотеки slatec для смеси с содержанием водорода равным 20 моль-% в момент времени 1000 миллисекунд

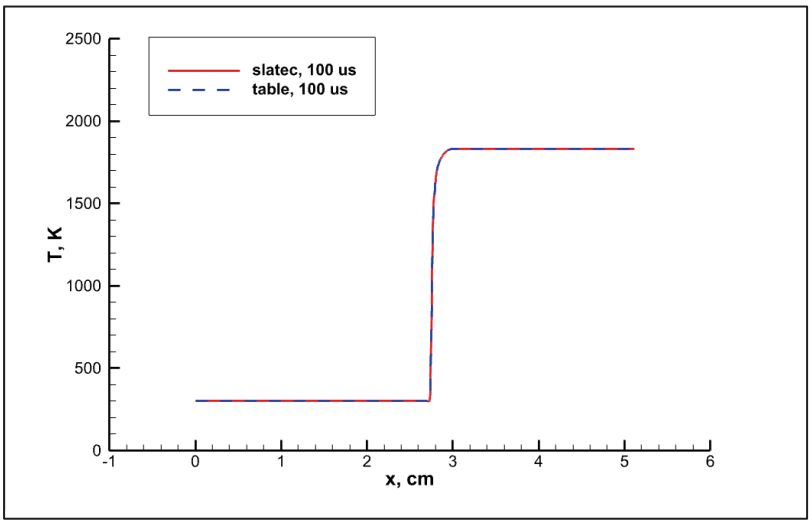


Рис. 5. Сравнение прямого (сплошные линии) и аппроксимированного (пунктирные линии) расчёта температуры среды для горения смеси с концентрацией водорода 20 моль-% в момент времени 100 микросекунд

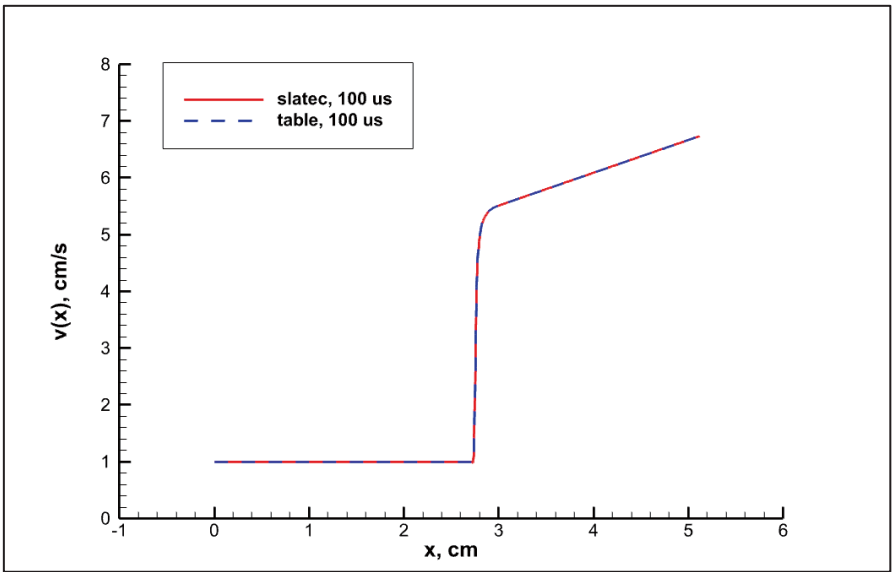


Рис. 6. Сравнение прямого (сплошные линии) и аппроксимированного (пунктирные линии) расчёта массовой скорости среды при горении смеси с концентрацией водорода 20 моль-% в момент времени 100 микросекунд

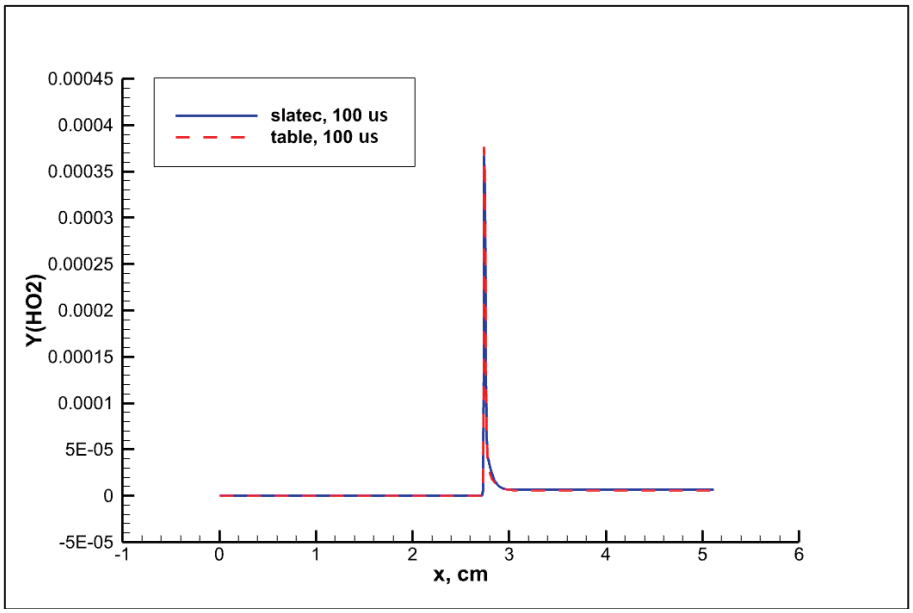


Рис. 7. Сравнение прямого (сплошные линии) и аппроксимированного (пунктирные линии) расчёта массовой концентрации радикала HO_2 при горении смеси с содержанием водорода 20 моль-% в момент времени 100 микросекунд

Оценке скорости вычислений при использовании табличной аппроксимации решения уравнений химической кинетики при расчёте стабилизированных волн горения показало высокую эффективность. Так, при прямом расчёте волны горения воздушно-водородной смеси с массовой концентрацией водорода 20% среднее время расчёта динамики процесса горения на временном интервале 100 микросекунд, составляет 8 секунд реального времени, в то время как аналогичная разница при использовании табличной аппроксимации составила 3.5 секунды, что говорит о более чем двукратном повышении вычислительной эффективности для детальной схемы окисления водородно-воздушной смеси. Важно отметить, что система уравнений химической кинетики окисления сложных углеводородных топлив может считывать более сотни элементарных компонент и соответствующих обыкновенных дифференциальных уравнений изменения концентраций компонент (6). Использование табличной аппроксимации для анализа процессов горения в подобных топливах представляется особенно перспективным.

5. Вывод

В настоящей работе проведена серия расчетов процесса распространения пламени для водородно-воздушной смеси с использованием детальной схемы химической кинетики и применением как прямого подхода к решению системы уравнений химической кинетики, так и метода табличной аппроксимации. Проведено сравнение параметров среды, полученных с использованием прямого решения и табличного. Показано, что использование таблиц, основанных на предварительно рассчитанной структуре стабилизированного фронта горения для конкретной смеси, без значительной потери точности позволяет значительно повысить эффективность расчётов распространения волн горения в одномерной камере сгорания. В дальнейшем планируется усовершенствование механизма поиска аппроксимированного решения для повышения его точности.

Литература

1. J.A. van Oijen, A. Donini, R.J.M. Bastiaans, J.H.M. ten Thije Boonkkamp, L.P.H. de Goey. State-of-the-art in premixed combustion modeling using flamelet generated manifolds. 2016. Progress in Energy and Computer Science, V. 57, pp. 30-74.
2. Kenneth K. Kuo, Ragini Acharya «Fundamentals of turbulent and multiphase combustion»
3. https://github.com/yakovenko-ivan/NRG/
4. Kevin McGrattan, Simo Hostikka, Jason Floyd, Randall McDermott, Marcos Vanella. Fire Dynamics Simulator Technical Reference Guide Volume 1: Mathematical Model. DOI: NIST.SP.1018.
5. A. Kéromnès, et al. An experimental and detailed chemical kinetic modeling study of hydrogen and syngas mixture oxidation at elevated pressures. 2013. Combust. Flame, V. 160, pp. 995–1011.

Оптимизация численного моделирования переноса пассивной примеси на графических ускорителях

Е.М. Гащук[1,2,3], А.А. Ежкова[2], В.А. Оноприенко[2], А.В. Дебольский[3,4], Е.В. Мортиков[3,2]

[1]Московский государственный университет им. М.В.Ломоносова, Механико-математический факультет
[2]Институт вычислительной математики им. Г.И. Марчука РАН
[3]Московский государственный университет им. М.В. Ломоносова, Научно-исследовательский вычислительный центр
[4]Институт физики атмосферы им. А.М. Обухова РАН

Моделирование переноса большого числа примесей в атмосфере и океане требует значительных вычислительных ресурсов, поэтому ускорение расчетов остается актуальной задачей на сегодняшний день. В данной работе рассматривается эффективность использования GPU (Graphics Processing Unit) в сравнении с CPU (Central Processing Unit) для реализации адвективных схем, которые наиболее часто используются в моделях общей циркуляции океана. Также мы представляем результаты по применению ряда методов оптимизации для повышения эффективности вычислений на GPU в случае грубого пространственного разрешения, характерного для климатических расчетов. Однако, при использовании множества вычислительных процессов возникает необходимость в обменах данными, а при использовании GPU появляются дополнительные накладные расходы, вызванные копированием данных с памяти GPU на CPU и обратно. По этой причине в данной работе мы также представляем результаты оптимизации обмена данными между графическими процессорами.

Ключевые слова: графические ускорители, моделирование переноса пассивной примеси в модели общей циркуляции океана, вычислительная мощность

1. Введение

Численные модели общей циркуляции океана являются важным инструментом в исследованиях Земной системы и необходимы для уточнения нашего понимания влияния океана на погоду и климат. Они являются неотъемлемой частью современных моделей прогноза погоды и климата.

Проведение численных экспериментов по моделированию океана с высоким разрешением по-прежнему требует огромных вычислительных ресурсов. С учетом чувствительности результатов моделирования климата (см., например, [12]) к пространственному разрешению и возрастающей сложности океанических моделей (некоторые из которых теперь дополнены моделями биохимии океана, что требует решения большого числа уравнений типа адвекции-диффузии-реакции [11]), ясно, что повышение вычислительной эффективности алгоритмов, используемых в моделях, является одной из приоритетных задач.

Повышение вычислительной эффективности океанических моделей может быть достигнуто путем разработки новых алгоритмов и адаптации их для выполнения на параллельных вычислительных системах различной архитектуры. Особый интерес представляют HPC-системы (High-Performance Computing), использующие графические процессоры. Новейшие архитектуры GPU имеют пиковую производительность в несколько раз выше, чем последние поколения CPU, причем разрыв в производительности с развитием технологий увеличивается [7, 9].

Алгоритмы, используемые в моделировании океана, могут быть многократно ускорены за счет использования GPU. В статье [14] представлена новая версия модели океана

POM (Princeton Ocean Model) с поддержкой вычислений на GPU. Эта версия демонстрирует производительность на сервере с четырьмя GPU, сравнимую с кластером из 408 ядер CPU, что соответствует почти семикратному снижению энергопотребления. В работе [15] также обсуждаются методы оптимизации алгоритмов на GPU, используемых в Princeton Ocean Model, направленные на более эффективное использование иерархии памяти процессора для подобных приложений, скорость выполнения которых определяется в основном объемом свободной памяти, необходимой для хранения используемых данных. Тесты показали, что предложенные подходы позволили ускорить модель в 5.8-16.7 раз, в зависимости от используемой архитектуры GPU и CPU.

Однако, за счет распараллеливания на GPU возникают дополнительные накладные расходы из-за необходимости обмена данными, расположенными в памяти графических процессоров. Современные реализации программного интерфейса MPI пока не приспособлены к эффективному обмену такими массивами. Для этого были разработаны специализированные технологии передачи данных на параллельных вычислительных системах, такие как NVIDIA Collective Communication Library (NCCL) и Inter-Process Communication (IPC). Так, например, в работе [8] получено, что для сообщений размером 4МБ использование IPC может позволить уменьшить задержку двусторонней и односторонней связи между GPU на 79% и 74% соответственно, а также увеличить пропускную способность в 4 раза.

Целью данного исследования было изучение возможности ускорения вычислений для двух- и трехмерной моделей переноса пассивной примеси в океане путем переноса реализации на GPU. В данной работе мы представляем сравнение вычислительной производительности реализаций различных схем адвекции на GPU и CPU. Помимо этого, проводится сравнение времени выполнения обмена данными на GPU в зависимости от реализации: исходной MPI-реализации с IPC в случае, если обмены происходят внутри узла, иначе – с NCCL-реализацией.

2. Численная реализация

Уравнение переноса примеси $\varphi_k(\mathbf{x}, t),\ k = 1, \ldots, N_c$ представлено следующим образом:

$$\frac{\partial \varphi_k}{\partial t} + \nabla_x(u\varphi_k) + \nabla_y(v\varphi_k) + \nabla_z(w\varphi_k) = Q_k, \tag{1}$$

где $\mathbf{x} = (x, y, z)$ – произвольная ортогональная криволинейная z-система координат на сфере, u, v и w обозначают компоненты вектора скорости $\mathbf{u}$, Q_k – вклад процессов, отличных от адвекции, и t обозначает время. Q_k может соответствовать горизонтальной или вертикальной диффузии, газообмену с атмосферой, химическим реакциям и другим процессам. Предполагается, что поле скорости бездивергентно:

$$\nabla_x u + \nabla_y v + \nabla_z w = 0.$$

Мы рассматриваем уравнение (1) как для трехмерной задачи, так и для двумерной, где вертикальный перенос отсутствует, а поле скорости и скалярные величины зависят от горизонтальных координат x и y. В данной работе мы обсуждаем реализацию схем адвекции и, следовательно, задаем $Q_k \equiv 0$, но рассматриваемые подходы могут быть применены и в более общем случае.

2.1. Численные схемы

В данной работе исследуются следующие численные схемы для решения уравнения переноса примеси: схемы upwind первого (UW) и третьего (UW3) порядка точности, центрально-разностная схема 2-го порядка точности, записанная в дивергентной (DIV) и кососимметричной (SKEW) формах, схема Лакса–Вендроффа с ограничителем потока (LAX) [5] и алгоритм Flux-Corrected Transport (FCT) [16].

В случае схем LAX и FCT для интегрирования уравнения (1) по времени мы использовали метод Эйлера. С остальными схемами был использован метод Адамса-Бэшфорта третьего порядка точности.

3. Программная реализация

В этом разделе описаны детали реализации для двумерной задачи, поскольку они легко обобщаются на случай трехмерной задачи.

Численная модель [4, 6, 10] реализована с использованием языка программирования C/C++ на основе подхода MPI-OpenMP-CUDA, что обеспечивает поддержку вычислений на гибридных архитектурах современных HPC-систем. Решение уравнения переноса (1) реализовано так, что все расчеты можно проводить на GPU. Мы рассмотрели следующие методы оптимизации для реализации переноса:

- Shared memory (разделяемая память)

  Использование shared memory может ускорить вычисления за счет сокращения обращений к глобальной памяти, которые часто возникают при реализации ограниченных по памяти сеточных операторов (ядро – функция, которая будет запущена в нескольких экземплярах, каждый из которых будет выполняться на своём ядре графической карты).

- Kernel fusion (слияние ядер)

  Если между вызовами последовательности ядер нет точек синхронизации, то можно объединить последовательность в одно ядро, избегая многократных вызовов.

  Численный алгоритм решения (1) с использованием явной по времени схемы Адамса-Бэшфорта третьего порядка точности можно представить в виде следующих шагов:

$$(\mathrm{ADV})_k^n = -\left[\nabla \cdot (\mathbf{u}\varphi_k)\right]_h^n, \tag{2.а}$$

$$(\mathrm{RHS})_k^n = \frac{23}{12}(\mathrm{ADV})_k^n - \frac{4}{3}(\mathrm{ADV})_k^{n-1} + \frac{5}{12}(\mathrm{ADV})_k^{n-2}, \tag{2.б}$$

$$\varphi_k^{n+1} = \varphi_k^n + (\mathrm{RHS})_k^n \cdot \Delta t, \tag{2.в}$$

  где $[\cdot]_h^n$ обозначает пространственную аппроксимацию члена адвекции на вычислительной сетке с использованием значений на n-м временном шаге. В таком виде алгоритм состоит из расчета адвекции (2.а), интегрирования по времени (2.б) и обновления значений концентрации φ_k для k-го вещества (2.в) на следующем $(n+1)$ шаге по времени. Мы объединили шаги (2.а) и (2.б) в одно ядро в случае схем DIV, SKEW и UW3.

- Scalar fusion

  При рассмотрении $N_c \in \mathbb{N}$ примесей вычисления могут быть объединены для улучшения эффективности обращения к памяти. В результате каждый шаг интегрирования по времени выполняется для всего набора (или подмножества) примесей вместо N_c отдельных вычислений для каждой концентрации скаляра φ_k. В частности, такая модификация алгоритма может позволить оптимизировать обращение к памяти для поля скорости, которое одинаково для всех примесей. Для реализации такого подхода (*scalar fusion*) мы объединили массивы, используемые для хранения значений концентрации φ_k в единый вектор состояния полной системы Φ. Заметим, что расположение данных концентрации в Φ может быть различным. Этот метод был реализован двумя способами в зависимости от размещении памяти k-ой примеси в Φ:

1. $[k][x][y]$ порядок: $\varphi_k[m] = \Phi[k \cdot N_g + m]$,
2. $[x][y][k]$ порядок: $\varphi_k[m] = \Phi[m \cdot N_c + k]$,

где порядок задан для k-го вещества, $k = 1, \ldots, N_c$, и m-го индекса ячейки вычислительной сетки (сетка состоит из $N_g = N_x \cdot N_y$ ячеек), $m = 1, \ldots, N_g$. В первом подходе сохраняется локальность ячеек сетки, а во втором скалярные величины для каждой ячейки хранятся последовательно в пространстве памяти.

Как уже ранее упоминалось, при вычислениях на нескольких GPU появляются дополнительные накладные расходы, вызванные копированием данных с памяти GPU на CPU и обратно при выполнении MPI-обменов. В связи с этим мы провели тесты по передаче данными напрямую между GPU с помощью технологии NCCL [1], когда необходимо произвести обмен между процессами, находящимися на разных узлах. В случае, когда обмен проводится между процессами на одном вычислительном узле, была использована технология IPC [2].

4. Постановка задачи

Для оценки эффективности реализации схем переноса рассматривалась постановка задачи, предложенная в работе [13] для тестирования моделей атмосферы и океана.

Численные эксперименты проводились на равномерной географической сетке. Для изучения зависимости эффективности от размера сетки мы проводили расчеты с различным горизонтальным разрешением: $\Delta_{\lambda,\theta} \in [2^\circ, 1^\circ, 0.8^\circ, 0.4^\circ, 0.2^\circ]$. Для двумерного случая это соответствует следующим размерностям задачи: $N_x \times N_y \in \{180 \times 90,\ 360 \times 180,\ 450 \times 225,\ 900 \times 450,\ 1800 \times 900\}$. В трехмерных экспериментах число вертикальных уровней полагалось фиксированным и равным 48. Здесь, для эквивалентного горизонтального разрешения, сетка состоит из $N_x \times N_y \times 48$ ячеек. Из-за ограниченного размера оперативной памяти тесты с $\Delta_{\lambda,\theta} = 0.2^\circ$ для трехмерного случая не проводились.

Время интегрирования составляет 12 дней, а шаг по времени варьируется от разрешения вычислительной сетки. Для экспериментов на самой грубой сетке $\Delta_{\lambda,\theta} = 2.0^\circ$ шаг по времени равен 600 секундам. В экспериментах с высоким разрешением шаг по времени уменьшается так, чтобы число Куранта $\mathrm{CFL} = \max\limits_{i,j}[CFL_{i,j}]$, где $CFL_{i,j}$ – значение числа Куранта в i, j ячейке, оставалось приблизительно постоянным (≈ 0.1).

5. Результаты численных эксперментов

5.1. Эффективность базовой реализации

На Рисунке 1 показаны результаты ускорения двумерной модели при переносе вычислений на графические процессоры. Получено, что расчеты переноса пассивной примеси, выполненные на одном A100, оказались быстрее максимум в 80 раз по сравнению с одним CPU[1]-ядром[2] (Рисунок 1a) и максимум в 6 раз по сравнению с одним узлом из 18 ядер CPU (Рисунок 1b) на точных сетках, в то время как для тестов на грубых сетках наблюдаются противоположные результаты: и ядро CPU, и узел оказались более эффективны, чем один GPU. Для трехмерной модели расчеты на A100 в среднем в 30 раз быстрее одного ядра CPU и в среднем в 2.5 раза быстрее одного узла CPU в тестах с высоким разрешением. Для сеток с грубым разрешением результаты аналогичны 2D случаю.

[1]Intel Xeon Gold 6240

[2]Под CPU-ядром подразумевается физическое ядро процессора

5.2. Shared memory

Получено, что оптимизация, основанная на использовании shared памяти графических устройств, не обеспечивает значительного ускорения и может даже увеличить время моделирования переноса как на V100, так и на A100 GPU. В случае трехмерной модели наблюдается схожая зависимость. Эти результаты согласуются с выводами в работе [3], где данный подход был использован для оптимизации алгоритма фильтрации изображений с наблюдаемым снижением производительности на 20% на графических процессорах архитектур Volta и Ampere.

5.3. Kernel Fusion

Метод был применен к численному алгоритму решения (1) при использовании схем адвекции DIV, SKEW и UW3 с явной по времени схемой Адамса-Бэшфорта третьего порядка точности. Для двумерной модели метод повысил скорость вычислений для всех тестируемых разрешений, за исключением схемы UW3 на V100 с замедлением реализации вплоть до 20% на самой точной сетке ($\Delta_{\lambda,\theta} = 0.2^\circ$). Для трехмерного случая этот метод обеспечил в лучшем случае незначительный выигрыш, а в среднем оказался неэффективным.

5.4. Scalar fusion

Результаты оптимизации вычислений за счет использования *scalar fusion* подхода для $N_c = 15$ показаны на Рисунке 2. Было обнаружено, что как для двух-, так и для трехмерной модели этот подход обеспечил ускорение во всех проведенных экспериментах. Для двумерной модели оптимизация дала ускорение примерно в 5 раз на V100 (Рисунок 2b) и примерно в 9 раз на A100 (Рисунок 2a) на самой грубой сетке. Ускорение уменьшается с увеличением горизонтального разрешения для всех реализованных схем адвекции, так что время вычислений становится близким к времени выполнения для базовой реализации при $\Delta_{\lambda,\theta} = 0.2^\circ$. Среднее ускорение в 1.5 раза достигается в тестах с грубым разрешением на V100 и A100 для трехмерного случая. В отличие от двумерного случая, зависимость ускорения от разрешения не одинакова для всех схем адвекции.

Рисунок 3 демонстрирует ускорение реализации $[x][y][k]$ по сравнению с $[k][x][y]$ для тестов двумерной модели в случае схем DIV и UW3. Для обеих схем мы видим, что ускорение достигает своего предела в экспериментах с высоким разрешением. Максимальное достигнутое ускорение составило 1.5 раза для DIV и 1.4 раза для UW3 на V100.

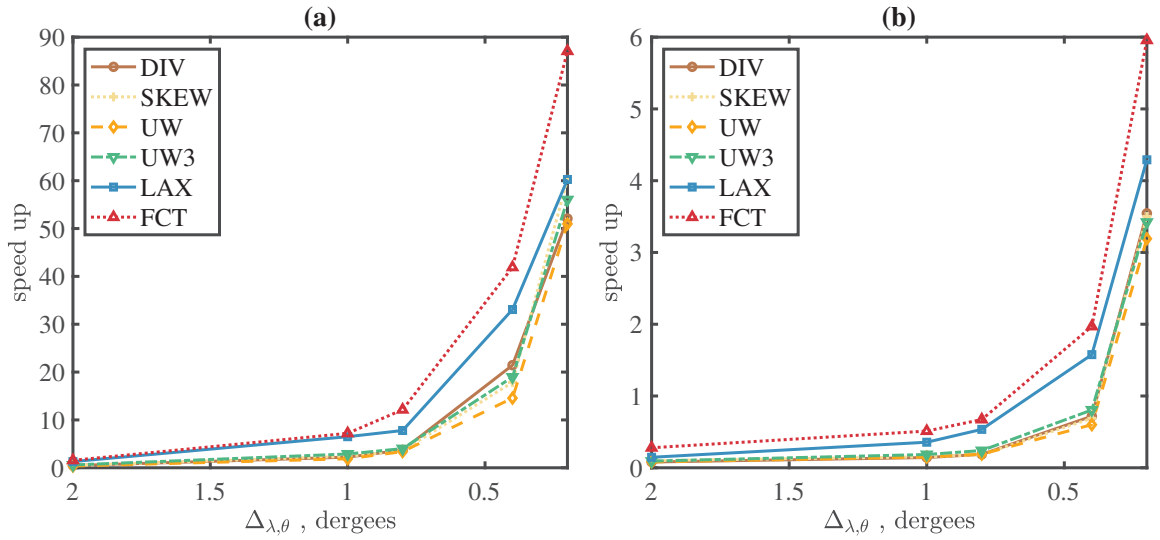


Рис. 1: Ускорение выполнения расчетов переноса примеси на A100 GPU по отношению к CPU-ядру **(a)** и по отношению к CPU-узлу **(b)** в зависимости от схемы и горизонтального разрешения $\Delta_{\lambda,\theta}$ для двумерной модели.

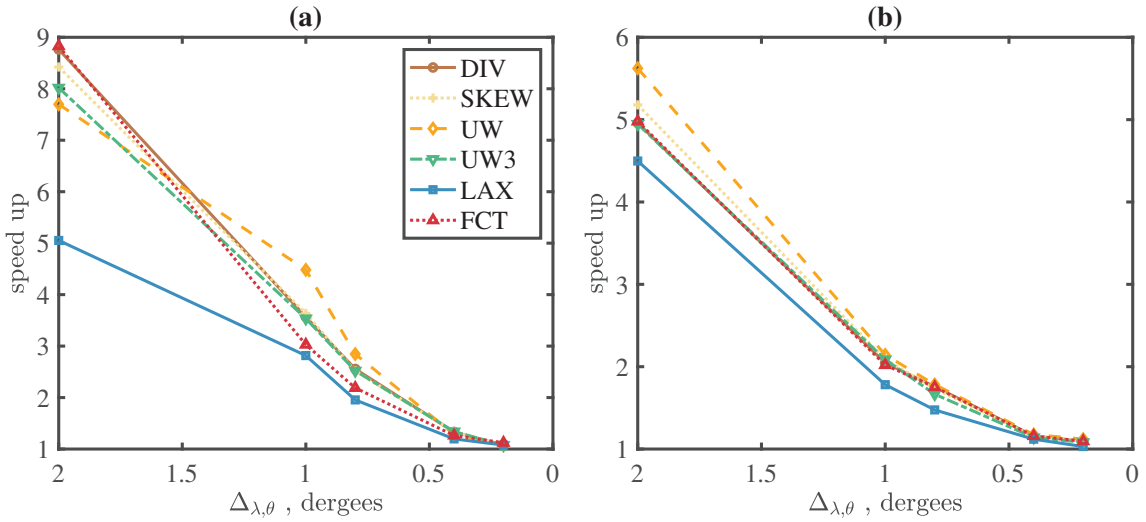


Рис. 2: Ускорение метода scalar fusion по отношению к базовой реализации для двумерной модели на A100 **(a)** и V100 **(b)** GPUs.

5.5. Обмен данными

На рисунке 4 показано ускорение выполнения обмена данными при использовании IPC (Рис. 4a) и NCCL (Рис. 4b) по отношению к базовой реализации посредством библиотеки MPI. Тесты проводились на двумерной MPI-сетке для 4 процессов, где каждый обменивался данными со всеми своими соседями.

Показано, что для обмена между GPU, находящимися на одном узле, во всех случаях использование IPC намного эффективнее MPI, а максимальное ускорение достигает около 40 раз для сообщений размера 1МБ. В случае, когда графические карты расположены на разных узлах, зависимость иная: для сообщений размером менее 0.26МБ базовая MPI-реализация выполняет обмены быстрее, однако, для сообщений большего размера использование NCCL оказывается эффективнее – наблюдается ускорение в 2.5 раза для сообщений

размера 1МБ.

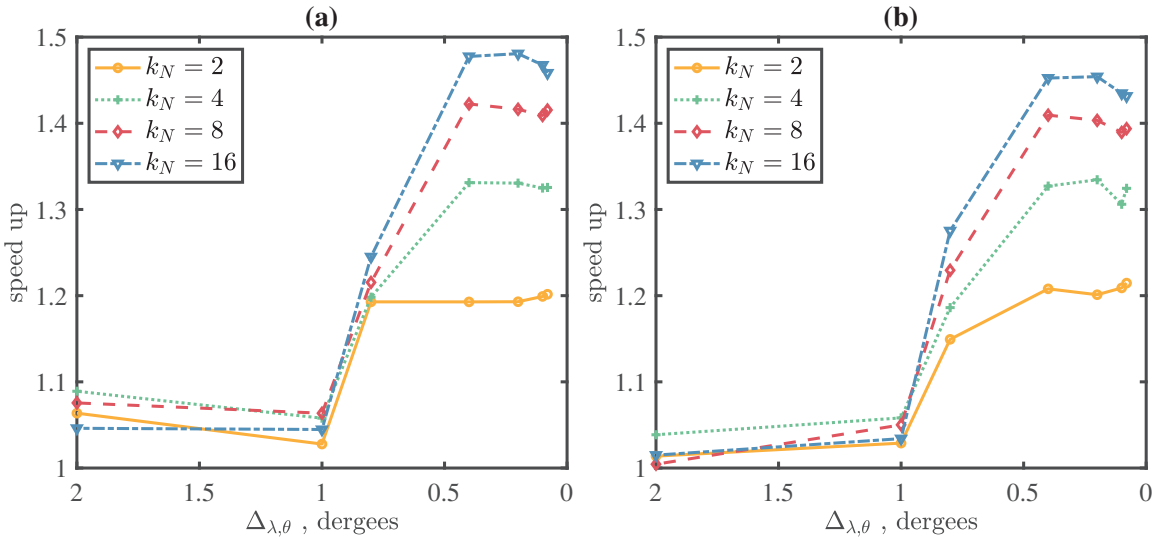


Рис. 3: Ускорение метода scalar fusion с переупорядочиванием по отношению к базовой реализации в случае двумерной модели для схем DIV **(a)** и UW3 **(b)** на V100 GPU.

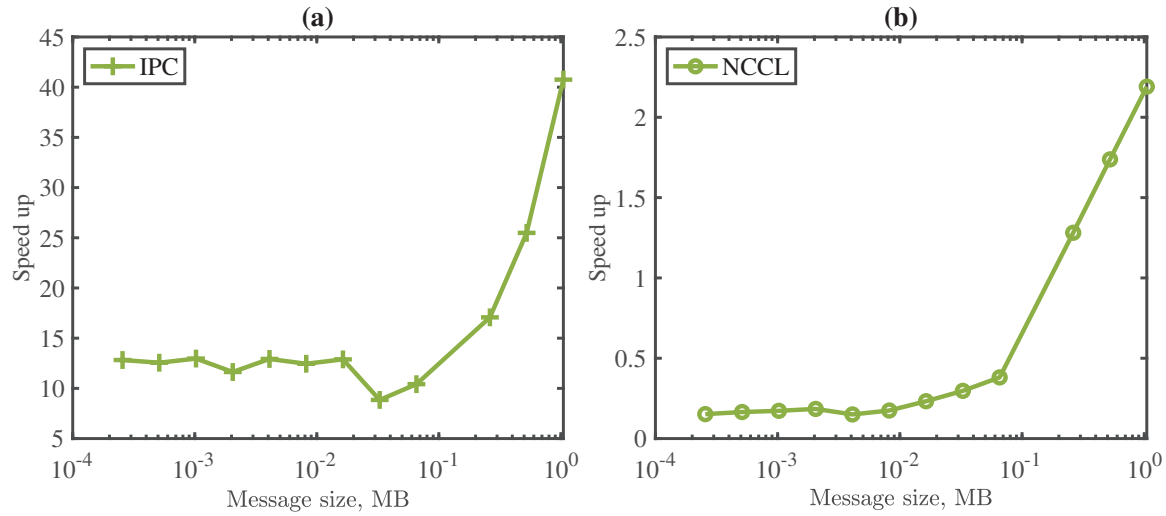


Рис. 4: Ускорение реализации обмена с применением IPC **(a)** и NCCL **(b)** на V100 GPU в зависимости от размера пересылаемого сообщения.

6. Заключение

В данной работе рассматривалась задача ускорения расчета переноса пассивной примеси в моделях океана на графических процессорах.

Численные эксперименты для оценки производительности показывают, что для двумерного случая базовая реализация на одном GPU до 80 раз быстрее, чем вычисления на одном CPU, и время выполнения расчетов на одном графическом процессоре сопоставимо с расчетами на 18 ядрах центрального процессора на точных сетках в отличие от экспериментов с грубым разрешением, где расчеты на CPU-узле оказались эффективнее.

Для повышения эффективности GPU-реализации на сетках грубого разрешения был исследован ряд методов оптимизации вычислений. Было обнаружено, что использование

разделяемой памяти для реализации схем адвекции не дает значительного ускорения, а результаты применения *kernel fusion* показывают, что данный метод не всегда является эффективным.

В случае оптимизации *scalar fusion* показано, что прямой подход к объединению расчета переноса скаляров повысил эффективность реализации во всех проведенных тестах. Для двумерной модели метод обеспечил ускорение примерно в 5 раз на V100 и примерно в 9 раз на A100 на самой грубой сетке. Более того, мы можем повысить эффективность метода *scalar fusion* для экспериментов с высоким разрешением, переупорядочив расположение массивов скаляров (вместо подхода $[k][x][y]$ используется $[x][y][k]$) в едином непрерывном пространстве памяти. *Scalar fusion* с переупорядочиванием обеспечил увеличение скорости примерно в 1.5 раза на V100 по сравнению с прямым методом для двумерного случая при проведении экспериментов на точных сетках для схем DIV и UW3.

В результате проведенных тестов по ускорению выполнения обмена данными напрямую между GPU получено, что использование IPC ускоряет обмен во всех экспериментах (максимальное ускорение ≈ 40 раз для сообщений размером 1МБ). Применение NCCL позволяет получить выигрыш в скорости проведения обмена только в случае сообщений размером более 0.26МБ (максимальное ускорение ≈ 2.5 раза для сообщений размером 1МБ).

Код модели, использованной в данной работе, доступен на GitLab сервере `http://tesla.parallel.ru` по запросу.

7. Благодарности

Работа выполнена при частичной поддержке гранта РНФ № 21-71-30023 (разработка реализации численной модели переноса пассивной примеси в океане), гранта РНФ № 21-71-30003 (исследование методов оптимизации для архитектуры GPU) и гранта молодежной лаборатории «Суперкомпьютерные технологии математического моделирования Земной системы» № 075-03-2023-509/1. Тесты производительности проводились с использованием оборудования Центра коллективного пользования сверхвысокопроизводительными вычислительными ресурсами МГУ имени М.В. Ломоносова и с использованием ресурсов Центра коллективного пользования “Центр обработки и хранения научных данных ДВО РАН”, поддерживаемого Министерством науки и высшего образования РФ (проект № 075-15-2021-663).

Литература

1. *Corporation NVIDIA*. NVIDIA Collective Communication Library (NCCL) Documentation. URL: `https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/index.html`. 2020.

2. *Corporation NVIDIA*. CUDA Runtime API. URL: `https://docs.nvidia.com/cuda/cuda-runtime-api/group__CUDART__DEVICE.html`. 2023.

3. *Gambrych J.* Influence of optimization techniques on software performance for subsequent generations of CUDA architecture // 2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom). 2021. 1002–1009.

4. *Gladskikh D.S., Stepanenko V.M., Mortikov E.V.* The effect of the horizontal dimensions of inland water bodies on the thickness of the upper mixed layer // Water Resources. Vol. 48. P. 226–234. 2021. DOI: 10.1134/S0097807821020068.

5. *LeVeque R.J.* High-Resolution Conservative Algorithms for Advection in Incompressible Flow // SIAM Journal on Numerical Analysis. Vol. 33, No. 2. P. 627–665. 1996. DOI: 10.1137/0733033.

6. *Mortikov E.V., Debolskiy A.V.* Direct Numerical Simulation of Stratified Turbulent Flows and Passive Tracer Transport on HPC Systems: Comparison of CPU Architectures // Supercomputing Frontiers and Innovations. Vol. 8, No. 4. P. 50–68. 2021. DOI: 10.14529/jsfi210405.

7. *Pi Puig M., De Giusti L., Naiouf M.* Are GPUs Non-Green Computing Devices? // Journal of Computer Science and Technology. Vol. 18. P. e17. 10 2018. DOI: 10.24215/16666038.18.e17.

8. *Potluri S., Wang H., Bureddy D., Singh A.K., Rosales C., Panda Dhabaleswar K.* Optimizing MPI Communication on Multi-GPU Systems Using CUDA Inter-Process Communication // 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum. 2012. 1848–1857.

9. *Sun Y., Agostini N.B., Dong S., Kaeli D.R.* Summarizing CPU and GPU Design Trends with Product Data // ArXiv. Vol. abs/1911.11313. 2019. DOI: 10.48550/ARXIV.1911.11313.

10. *Tkachenko E.V., Debolskiy A.V., Mortikov E.V.* Intercomparison of subgrid scale models in large-eddy simulation of sunset atmospheric boundary layer turbulence: Computational aspects // Lobachevskii Journal of Mathematics. Vol. 42, No. 7. P. 1580–1595. 2021. DOI: 10.1134/S1995080221070234.

11. *Watanabe S., Hajima T., Sudo K., Nagashima T., Takemura T., Okajima H., Nozawa T., Kawase H., Abe M., Yokohata T., Ise T., Sato H., Kato E., Takata K., Emori S., Kawamiya M.* MIROC-ESM 2010: model description and basic results of CMIP5-20c3m experiments // Geoscientific Model Development. Vol. 4, No. 4. P. 845–872. 2011. DOI: 10.5194/gmd-4-845-2011.

12. *Wickramage C., Köhl A., Jungclaus J., Stammer D.* Sensitivity of MPI-ESM Sea Level Projections to Its Ocean Spatial Resolution // Journal of Climate. Vol. 36, No. 6. P. 1957 – 1980. 2023. DOI: 10.1175/JCLI-D-22-0418.1.

13. *Williamson D.L., Drake J.B., Hack J.J., Jakob R., Swarztrauber P.N.* A standard test set for numerical approximations to the shallow water equations in spherical geometry // Journal of computational physics. Vol. 102, No. 1. P. 211–224. 1992. DOI: 10.1016/S0021-9991(05)80016-6.

14. *Xu S., Huang X., Oey L.-Y., Xu F., Fu H., Zhang Y., Yang G.* POM.gpu-v1.0: a GPU-based Princeton Ocean Model // Geoscientific Model Development. Vol. 8, No. 9. P. 2815–2827. 2015. DOI: 10.5194/gmd-8-2815-2015.

15. *Xu S., Huang X., Zhang Y., Hu Y., Yang G.* A customized GPU acceleration of the Princeton ocean model // 2014 IEEE 25th International Conference on Application-Specific Systems, Architectures and Processors. 2014. 192–193.

16. *Zalesak S.T.* Fully multidimensional flux-corrected transport algorithms for fluids // Journal of computational physics. Vol. 31, No. 3. P. 335–362. 1979. DOI: 10.1016/0021-9991(79)90051-2.

Опыт разработки и преподавания учебного курса по распределенным вычислениям

А.Н. Свистунов, Н.В. Шестакова

Нижегородский государственный университет им. Н.И. Лобачевского

В статье представлен учебный курс «Методы и средства построения распределенных программных систем с использованием технологии Java», разработанный и реализуемый с 2003 года в ННГУ им. Н. И. Лобачевского. Актуальность курса связана с тем, что он направлен на изучение технологий, позволяющих создавать распределенные системы, которые все более широко применяются в различных областях человеческой деятельности. В курсе происходит знакомство слушателя с основными концепциями, применяемыми при проектировании такого рода систем. Описаны концепция курса, его структура, категории слушателей, которым он может быть интересен, и варианты построения курса в зависимости от уровня подготовки аудитории**.** Рассмотрены предпосылки создания курса, проведен анализ многолетнего обучения по материалам курса, обобщен опыт его использования. Сформулированы принципы успешного проектирования и развития дисциплины в области информационных технологий.

Ключевые слова: распределенные системы, язык программирования Java, профессиональные компетенции.

1. Введение

Область информационных технологий стремительно развивается, в связи с чем потребность в высококвалифицированных ИТ-кадрах стабильно возрастает. На государственном уровне это находит отражение в развитии федеральных государственных стандартов, что в свою очередь ставит перед вузами задачу непрерывного обновления образовательных программ как по составу входящих в них дисциплин, так и по содержанию отдельных курсов. Вопросы реализации такой актуализации на основании профессиональных стандартов и с учетом мнения работодателей освещены в работах [1–5]. В работе [6] отмечается, что участие представителей ИТ-индустрии в формировании профессиональных компетенций ИТ-выпускников может происходить в разных формах и в целом повышает привлекательность таких специалистов на рынке труда. В данной статье обобщен опыт разработки и создания учебного курса в тесном сотрудничестве с представителями индустрии и получены методические рекомендации по проектированию и внедрению курсов в области формирования профессиональных компетенций ИТ-специалистов.

Практически все современные программные системы, с которыми мы взаимодействуем являются распределенными. Банкоматы, почтовые клиенты в наших смартфонах, веб-сайты магазинов и государственных структур, даже большинство современных автомобилей – все это примеры того, где используются распределенные программные системы. Как следствие этого, любой выпускник вуза, специализирующийся в области разработки программного обеспечения, практически наверняка столкнется либо с задачей проектирования, либо с задачей разработки, либо с задачей эксплуатации такого рода систем (либо со всеми такими задачами сразу). При проектировании, разработке и эксплуатации таких систем возникают проблемы, с которыми может не справиться специалист, не имеющий соответствующего опыта. Именно поэтому все ведущие университеты имеют в своей программе курсы, так или иначе посвященные проблематике распределенных систем. Поскольку рассматриваемая область очень объемная, курсы могут быть как общетеоретическими [7, 8] так и более прикладными, рассматривающими более подробно какие-то отдельные аспекты и технологии [9, 10]. Следует отметить, что все указанные задачи в общем случае являются более сложными, чем такие же задачи, но применительно к «локальному» (монолитному) приложению [11].

Учебный курс «Методы и средства построения распределенных программных систем с использованием технологии Java» призван решить две задачи:

1. Познакомить слушателя с основными проблемами и задачами, возникающими при проектировании, разработке, и эксплуатации распределенных систем. Дать для наиболее часто встречающихся из них типичные решения (шаблоны).
2. Дать обзор имеющихся технологий, используемых при создании распределенных систем. Привести примеры и сформировать у слушателя практические навыки их использования.

Все примеры, которыми иллюстрируется материал курса, выполнены на языке программирования Java. Это создает определенные трудности при иллюстрации некоторых разделов курса (например, при иллюстрации проблем, возникающих в гетерогенных средах при обмене данными). Второй проблемой, которая может возникнуть, является незнание слушателями этого языка программирования. Также следует учитывать, что программа курса включена в общую образовательную программу обучения в институте. То есть, во-первых, некоторые важные для понимания материала курса сведения считаются уже известными слушателям. Во-вторых, некоторые важные архитектурные шаблоны и технологии, безусловно относящиеся к области создания распределенных приложений, согласно учебной программе оказываются прочитанными в рамках других, читаемых в институте курсов, и не включаются в данный курс. Ярким примером является популярнейший архитектурный шаблон REST (и соответствующие способы его применения), рассматриваемый в курсе, посвященном Web-разработке.

В связи с вышеизложенными факторами, структура курса имеет модульный характер, и в зависимости от категории слушателей те или иные модули могут быть либо исключены из аудиторных часов и перенесены в материалы для самостоятельной работы студентов или добавлены. Ниже представлена программа курса, так, как он читается в институте информационных технологий, математики и механики ННГУ (ИИТММ ННГУ). Дополнительные модули, которые могут быть добавлены при необходимости, выделены отдельно.

Дальнейшее изложение в статье построено следующим образом. В разделе 2 изложены требования к уровню знаний слушателя курса, как обязательные, так и желательные. В разделе 3 описаны категории слушателей, которым рассматриваемый курс может быть полезен/интересен. В разделе 4 описана структура курса, в разделе 5 обобщен опыт его чтения, раздел 6 содержит заключительные выводы.

2. Предварительные требования к уровню слушателей курса

Как уже упоминалось выше, все примеры, а также все лабораторные и отчетные работы курса выполняются на языке программирования Java. Это не является проблемой для слушателей, являющихся студентами направлений подготовки 02.03.02 Фундаментальная информатика и информационные технологии (ФИИТ) и 09.03.04 Программная инженерия (ПИ) ИИТММ ННГУ, поскольку абсолютное большинство из них изучает соответствующий спецкурс по этому языку, но может стать проблемой как для студентов других специальностей, так и (при чтении курса за рамками ННГУ) для слушателей, базовыми знаниями Java не обладающими. Для решения этой проблемы в курс введен дополнительный модуль, посвященный знакомству с Java в том объеме, который требуется для усвоения курса. Кроме того, весь программный код примеров намеренно выполнен с использованием базового синтаксиса, упрощенных конструкций и с отказом от использования каких-либо дополнительных библиотек и фреймворков везде, где это возможно.

Поскольку распределенное приложение предполагает наличие обмена данными, слушатели также должны обладать некоторыми базовыми знаниями в области компьютерных сетей передачи данных. Студенты ИИТММ слушают соответствующий спецкурс и соответствующими знаниями обладают, в случае необходимости чтения вне рамок ННГУ в курс включен дополнительный раздел с краткими сведениями в этой области, необходимыми для успешного понимания материала курса.

Таким образом, для успешного освоения основной части материала слушатели курса должны обладать следующими навыками:

1. владеть базовыми знаниями по основам алгоритмизации, модульному программированию, алгоритмам и структурам данных, а также иметь опыт применения этих знаний, как минимум, на учебных задачах;
2. владеть базовыми знаниями об основах функционирования компьютерных сетей, быть знакомыми с сетевыми протоколами семейства TCP/IP;
3. владеть базовыми знаниями языка программирования Java.

3. Категории слушателей

Курс предоставляет базовые сведения из области проектирования и разработки распределенных систем и предназначен для слушателей либо не имеющих соответствующего опыта, либо имеющих минимальный опыт в этой области. В первую очередь, это студенты бакалавриата, уже освоившие объектно-ориентированное программирование и имеющие опыт написания учебных программ в рамках базовых курсов по программированию. Вторая категория слушателей, это интерны и молодые специалисты ИТ-компаний, нуждающиеся в быстром освоении соответствующих технологий и инструментов разработки распределенных приложений. Каждой из этих категорий слушателей требуется разная степень детальности изложения материала, кроме того они обладают разными входными знаниями и опытом. По этой причине структура курса построена в виде отдельных модулей, часть из которых может быть опущена и/или сокращена при соответствующем уровне подготовки слушателей.

4. Структура курса

Курс состоит из 7 основных модулей:

1. Введение в распределенные системы. Базовые примеры архитектур распределенных систем.
2. Прямое использование протоколов семейства TCP/IP для взаимодействия компонентов распределенного приложения. Примеры приложений.
3. Концепция RMI. Java RMI как как одна из реализаций концепции RMI. Примеры использования.
4. WebServices (SOAP) как пример реализации RMI с использованием открытых протоколов и стандартов. Примеры использования.
5. gRPC – реализация RMI от компании Google. Примеры использования.
6. Распределенная система как совокупность компонентов, обменивающихся сообщениями. Брокеры сообщений. JMS как стандарт промежуточного ПО для рассылки сообщений. Примеры использования.
7. Потоковые платформы обработки данных. Apache Kafka. Примеры использования.

и 3 дополнительных (читаются в случае, если слушатели не обладают необходимыми базовыми знаниями):

8. Обзор технологии JVM и языка программирования Java (базовый синтаксис, типы данных, ООП в Java, ввод-вывод, использование многопоточности и блокировки, рефлексия).
9. Краткие сведения о компьютерных сетях передачи данных, базовые сведения о протоколах семейства TCP/IP (TCP и UDP).
10. Использование современных средств для автоматизации развертывания и управления приложениями в средах с поддержкой контейнеризации, контейнеризатор приложений. Краткая справка по использованию Docker и Kubernetes.

Кроме заданий, рассматриваемых в ходе практических занятий, на протяжении курса слушатели выполняют две самостоятельные работы, которые являются итоговыми.

Следует отметить, что при чтении курса за пределами ННГУ практическая часть курса часто расширяется примерами на других языках программирования, отличных от Java (C++, Kotlin).

4.1 Модуль «Введение в распределенные системы. Базовые примеры архитектур распределенных систем»

Продолжительность модуля – две лекции.

Предварительные требования к освоению модуля отсутствуют.

В первой части модуля дается обзор типичных проблем, которые возникают при проектировании и разработке распределенного приложения, а также описываются некоторые стандартные пути их решения. Во второй части модуля дается обзор некоторых распространенных шаблонов, используемых при построении распределенных приложений, описываются их области применения.

4.2 Модуль «Прямое использование протоколов семейства TCP/IP для взаимодействия компонентов распределенного приложения. Примеры приложений»

Продолжительность модуля – две лекции, одна практика.

Предварительные требования к освоению модуля: базовые знания об особенностях использования протоколов TCP и UDP семейства TCP/IP.

В модуле дается обзор возможностей, которые предоставляются типичными API при использовании сетевых протоколов TCP и UDP. Рассматриваются некоторые шаблоны использования этих протоколов. Обсуждается возможность построения собственных прикладных протоколов поверх ненадежных транспортных. Приводятся примеры приложений, использующих рассмотренные подходы.

4.3 Модуль «Концепция RMI. Java RMI как как одна из реализаций концепции RMI. Примеры использования»

Продолжительность модуля – две лекции, одна практика.

Предварительные требования к освоению модуля: базовые знания из области структурного, модульного и объектно-ориентированного программирования. Знание языка программирования Java.

В модуле рассматривается общая концепция взаимодействия компонентов распределенной системы через вызовы удаленных методов и ее реализация в языке программирования Java (Java RMI). Обсуждается структура сервисов, методы определения интерфейсов и разрешения имен с помощью различных механизмов. Рассматриваются особенности передачи разных типов данных. Приводятся примеры приложений.

4.4 Модуль «WebServices (SOAP) как пример реализации RMI с использованием открытых протоколов и стандартов. Примеры использования»

Продолжительность модуля – две лекции, одна практика.

Предварительные требования к освоению модуля: базовые знания из области структурного, модульного и объектно-ориентированного программирования. Знание языка программирования Java.

В модуле рассматриваются ключевые элементы, особенности функционирования, области применения и способы использования технологии. Обсуждаются различные способы взаимодействия компонентов – с использованием кодогенератора и без. Приводятся примеры развертывания в разных средах – развертывание и функционирование под управлением сервера приложений (Apache Tomcat, Glassfish, ...), а также развертывание в виде самостоятельного приложения. Приводятся примеры приложений для всех рассмотренных вариантов.

4.5 Модуль «gRPC – реализация RMI от компании Google. Примеры использования»

Продолжительность модуля – две лекции, одна практика.

Предварительные требования к освоению модуля: базовые знания из области структурного, модульного и объектно-ориентированного программирования. Знание языка программирования Java.

В модуле рассматривается технология gRPC – современная технология RMI, обладающая рядом интересных возможностей. Рассматривается стандартный для gRPC способ кодирования данных (protobuf) и язык определения интерфейса сервиса. Обсуждаются способы генерации шаблонного кода для сервиса и клиента и способы коммуникации (синхронные и асинхронные вызовы, стриминг). Приводятся примеры приложений для иллюстрации всех рассмотренных возможностей технологии.

4.6 Модуль «Распределенная система как совокупность компонентов, обменивающихся сообщениями. Брокеры сообщений. JMS как стандарт промежуточного ПО для рассылки сообщений. Примеры использования»

Продолжительность модуля – две лекции, одна практика.

Предварительные требования к освоению модуля: базовые знания из области структурного, модульного и объектно-ориентированного программирования. Знание языка программирования Java.

В модуле рассматривается альтернативный способ построения распределенной системы – как совокупности независимых компонентов, обменивающихся сообщениями. Приводится мотивация для выбора подобного способа. Рассматриваются некоторые типичные шаблоны, применяемые при построении таких приложений. Обсуждается роль брокера сообщений. Рассматривается один из стандартов (JMS), применяемых при обмене сообщениями. Приводятся примеры приложений для иллюстрации рассмотренных подходов.

4.7 Модуль «Потоковые платформы обработки данных. Apache Kafka. Примеры использования»

Продолжительность модуля – две лекции, одна практика.

Предварительные требования к освоению модуля: базовые знания из области структурного, модульного и объектно-ориентированного программирования. Знание языка программирования Java.

В модуле рассматривается один из популярных подходов к построению распределенных приложений специального вида – приложений, предназначенных для потоковой обработки данных. В качестве иллюстрации используется продукт Apache Kafka. Рассматривается архитектура Apache Kafka, обсуждаются способы реализации компонентов для генерации сообщений и их обработки. Приводится мотивация для использования такого рода систем, обсуждаются области их применимости. Приводятся примеры приложений для иллюстрации рассмотренных подходов.

4.8 «Обзор технологии JVM и языка программирования Java (базовый синтаксис, типы данных, ООП в Java, ввод-вывод, использование многопоточности и блокировки, рефлексия)»

Продолжительность модуля – шесть лекций, шесть практик.

Предварительные требования к освоению модуля: базовые знания из области структурного, модульного и объектно-ориентированного программирования. Знание какого-либо объектно-ориентированного языка программирования (например, C++).

Модуль предназначен для ознакомления слушателей, незнакомых с языком программирования Java, с базовым синтаксисом языка, а также с некоторыми особенностями,

важными для понимания кода программ, приводимых в курсе. Так, разбираются некоторые вопросы, связанные с использованием многопоточности (в частности, синхронизация и использование потокобезопасных коллекций), стандартная сериализация Java (используется в Java RMI), особенности реализации ООП в Java, использование механизма аннотаций (используется в курсе для кодогенерации), рефлексия.

4.9 Модуль «Краткие сведения о компьютерных сетях передачи данных, базовые сведения о протоколах семейства TCP/IP (TCP и UDP)»

Продолжительность модуля – одна лекция.

Предварительные требования к освоению модуля отсутствуют.

В модуле приводятся краткие сведения о базовых аспектах функционирования компьютерных сетей передачи данных и приводятся практические примеры настройки, диагностики работы протоколов семейства TCP/IP в разных операционных системах.

4.10 Модуль «Использование современных средств для автоматизации развертывания и управления приложениями в средах с поддержкой контейнеризации, контейнеризатор приложений. Краткая справка по использованию Docker и Kubernetes»

Продолжительность модуля – одна лекция, одна практика.

Предварительные требования к освоению модуля: базовые знания из области структурного, модульного и объектно-ориентированного программирования. Знание языка программирования Java.

В модуле рассматривается современный подход к развертыванию приложений с использованием технологии контейнеризации. Даются примеры как использования готовых образов для разворачивания сторонних приложений, так и создания собственных образов, предназначенных для развертывания собственных приложений. Обсуждаются вопросы создания взаимозависимых групп сервисов и управления ими.

5. Апробация

История курса начинается в 2003-м году, когда в ННГУ совместно с одной из крупных Нижегородских ИТ-компаний была создана лаборатория, в рамках которой проводилось дополнительное обучение слушателей. Они набирались из числа студентов Нижегородских вузов и одновременно принимались в качестве интернов в указанную ИТ-компанию. Таким образом, перечень преподаваемых в лаборатории курсов и в целом программа обучения формировались исходя из производственной потребности, которая существовала в компании на тот момент.

Именно этот фактор сформировал основную направленность курса – формирование у слушателей практических навыков применения рассматриваемых в рамках курса технологий. Вторая важная особенность курса – использование языка программирования Java в качестве базового языка курса, также связана с имеющейся в тот момент потребностью в именно таких специалистах.

В виде курса, читаемого только в лаборатории, курс просуществовал недолго – уже начиная с 2005 года он начал читаться в виде спецкурса для студентов, обучающихся на кафедре Математического обеспечения ЭВМ факультета ВМК ННГУ, а начиная с 2010 года как спецкурс для направления подготовки ФИИТ.

В это же время материалы курса были опубликованы на электронной площадке Национального Открытого Университета «ИНТУИТ» [12], а затем были изданы в виде учебника [13].

В настоящее время курс включен в учебную программу направлений ФИИТ и ПИ ИИТММ ННГУ, кроме того в различных модификациях читался в Нижегородских ИТ-компаниях в рамках программ переподготовки, летних и зимних школ, и т.д.

Также на протяжении двух лет (2021, 2022) курс читался в совместном университете МГУ и Пекинского политехнического института (ППИ) в Шэньчжэне (КНР).

Конечно, со временем программа курса претерпевала значительные изменения. Поскольку курс отличается четко выраженной практической направленностью, а область информационных технологий крайне динамична, какие-то из рассматриваемых в рамках курса технологий теряли актуальность, и наоборот, возникали и набирали популярность новые технологии.

Так, например, одна из базовых технологий, вокруг которой строился курс в 2003 – 2005 гг (CORBA) в настоящее время уже не может считаться актуальной для новых проектов. Крайне популярная еще 15 лет назад Java EE с технологией EJB также сейчас все чаще заменяется другими подходами.

С другой стороны, получая обратную связь от слушателей, реагируя на запросы со стороны ИТ-компаний, в курс добавлялись и добавляются новые разделы, модифицировались приводимые в лекциях примеры, изменялись итоговые задания и способы контроля знаний.

Так, в связи с высокой востребованностью в индустрии, были добавлены разделы, посвященные технологии gRPC, и платформе Apache Kafka. С широким распространением микросервисного подхода к построению распределенных систем, модифицировался вводный модуль курса – в него добавлялись соответствующие архитектурные шаблоны. Поскольку с приходом контейнеров совершенно изменились подходы к развертыванию и эксплуатации распределенных приложений, в курс был добавлен соответствующий дополнительный модуль, и в настоящее время слушателям предлагается реализовывать отчетные задания в виде связанного набора сервисов.

В настоящее время курс продолжает развиваться: запланирована очередная переработка практической части с переходом на последние версии Java, внедрение в лабораторный практикум современных инструментов тестирования и оценки производительности.

За время своего существования курс был прочитан более 50 раз и прослушан в общей сложности более 3000 слушателей. Было получено много положительных отзывов как от представителей отрасли, так и от слушателей-студентов.

В ходе курса слушатели выполняют две отчетные самостоятельные работы, по итогам которых можно судить об успешности усвоения материала. Как и следовало ожидать, результаты сильно зависят от уровня подготовки и мотивации слушателей. Наилучшие результаты по усвоению материала демонстрировали интерны и молодые специалисты, уже работающие или стажирующиеся в ИТ-компаниях, наихудшие – слушатели, не имеющие профильного базового образования (зачастую в рамках переподготовки сотрудников компании-работодатели отправляли таких слушателей на этот курс). При чтении курса в ННГУ процент студентов, успевающих выполнить все отчетные задания в срок, составляет в среднем 70-75%.

6. Заключение

Опыт реализации данного курса, а также нескольких других ИТ-дисциплин [14, 15], позволяет сформулировать методические рекомендации по проектированию учебных курсов в области формирования профессиональных компетенций ИТ-специалистов. Использование следующих принципов позволяет создать востребованный образовательный контент:

1. тесное взаимодействие с представителями работодателей при формировании программы курса, выборе тематики лабораторных работ и рецензировании отчетных работ слушателей;
2. модульное построение и плановое проведение входного контроля для формирования индивидуальных образовательных траекторий в зависимости от исходных навыков слушателей;
3. наличие практико-ориентированных лабораторных работ для мотивации слушателей к активному обучению;
4. наличие материалов для обеспечения самостоятельной работы студентов.

Литература

1. Рязапова Л.З., Китаева Л.А. Профессиональные стандарты как детерминирующая база проектирования основных образовательных программ // Управление устойчивым развитием. 2019. № 3(22). С. 101–105.
2. Кузенков О.А., Грезина А.В., Шестакова Н.В., Карпенко С.Н. Опыт разработки образовательных стандартов (в соответствии с ФГОС 3++) // Образовательные технологии и общество. 2020. Т. 23. № 1. С. 159–169.
3. Каракозов С.Д., Худжина М.В., Петров Д.А. Проектирование содержания профессиональных компетенций образовательного стандарта ит-специалиста на основе требований профессиональных стандартов и работодателей // Информатика и образование. 2019. №7(306). С. 7–16. DOI: 10.32517/0234-0453-2019-34-7-7-16.
4. Фионова Л.Р. Управление подготовкой специалистов сферы информационных технологий на основе профессиональных стандартов // Прикаспийский журнал: управление и высокие технологии. 2018. №3(43). С. 47–59.
5. Гаврилов А.В., Куликова С.В., Голкина Г.Е. Повышение уровня подготовки IT-специалистов на основе анализа требований рынка труда // Открытое образование. 2019. Т.23. №6. С. 30–40.
6. Климова Ю.О. Анализ соответствия уровня компетенций выпускников ит-специальностей требованиям работодателей // Вопросы территориального развития. 2021. Т.9. № 1. С. 5. DOI: 10.15838/tdi.2021.1.56.5.
7. Distributed Systems.CS244B. Stanford School of Engineering. URL:https://online.stanford.edu/courses/cs244b-distributed-systems (дата обращения: 24.03.2023).
8. Distributed Computer Systems Engineering. MIT OpenCourseWare. URL:https://ocw.mit.edu/courses/6-824-distributed-computer-systems-engineering-spring-2006/ (дата обращения: 24.03.2023).
9. Distributed Systems Fundamentals. Columbia University Course COMS 4113. URL:https://systems.cs.columbia.edu/ds1-class/01-lectures/ (дата обращения: 24.03.2023).
10. Карпов Л.Е. Архитектура распределённых систем программного обеспечения (Distributed software systems' architecture). МГУ. URL:http://sp.cmc.msu.su/courses/sdpi/ (дата обращения: 24.03.2023).
11. Мартен ван С., Таненбаум Э.С. Распределенные системы. Москва: Изд-во ДМК Пресс, 2021. 584 с.
12. Свистунов А.Н. Построение распределенных систем на Java. Национальный Открытый Университет «ИНТУИТ». URL:https://intuit.ru/studies/courses/633/489/info (дата обращения: 24.03.2023).
13. Свистунов А.Н. Построение распределенных систем на Java : учебное пособие. Москва: Интернет-Университет Информационных Технологий: БИНОМ. Лаборатория знаний, 2011. 279 с.
14. Сысоев А.В., Горшков А.В., Волокитин В.Д., Шестакова Н.В., Мееров И.Б. Учебный курс «Программирование с использованием модели oneAPI» // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2022. Т. 11, № 3. С. 45–58. DOI: 10.14529/cmse220303.
15. Сысоев А.В., Шестакова Н.В., Пирова А.Ю. Магистерская программа «Вычислительная математика и суперкомпьютерные технологии»: опыт перехода на стандарт 3++ // Образовательные технологии и общество. 2019. Т. 22. № 4. С. 172-185.

Поиск аномалий в больших временных рядах на кластере с GPU узлами

Я.А. Краева, М.Л. Цымблер

Южно-Уральский государственный университет

В настоящее время обнаружение аномалий во временных рядах является одной из наиболее актуальных исследовательских проблем и возникает в широком спектре предметных областей: цифровая индустрия, здравоохранение, моделирование климата и прогноз погоды, финансовая аналитика и др. Одной из наиболее часто используемых на практике формализаций понятия аномалии является концепция диссонанса. Диссонанс определяется как подпоследовательность ряда, которая имеет расстояние до своего ближайшего соседа, не превышающее наперед заданного аналитиком порога. Ближайшим соседом подпоследовательности является та подпоследовательность ряда, которая не пересекается с данной и имеет минимальное расстояние до нее. В предыдущих исследованиях авторы разработали параллельный алгоритм поиска диссонансов, имеющих длину в заданном диапазоне длин, на графическом процессоре. Однако данный алгоритм ограничивает длину ряда размером оперативной памяти GPU. В статье представлено продолжение указанного исследования, в котором предлагается алгоритм поиска диссонансов временного ряда на вычислительном кластере, каждый узел которого оснащен графическим процессором. Вычислительные эксперименты с временными рядами из реальных предметных областей показывают высокую масштабируемость разработанного алгоритма.

Ключевые слова: временной ряд, поиск аномалий, диссонанс, параллельный алгоритм, вычислительный кластер, графический процессор, CUDA, DRAG, MERLIN, PD3, PALMAD.

Введение

В настоящее время обнаружение аномалий во временных рядах является одной из наиболее актуальных исследовательских проблем и возникает в широком спектре предметных областей: цифровая индустрия, здравоохранение, моделирование климата и прогноз погоды, финансовая аналитика и др. [1]. Наиболее востребованной и сложной здесь является задача поиска аномальных подпоследовательностей, состоящих из последовательных во времени значений, коллективное поведение которых необычно, хотя каждое из них в отдельности не обязательно является выбросом (аномалией) [1].

Концепция диссонанса временного ряда является в настоящее время одним из наиболее перспективных подходов к формализации понятия аномальной подпоследовательности [2, 3]. Диссонанс [4] определяется как подпоследовательность ряда, которая имеет максимальное расстояние до своего ближайшего соседа. Ближайшим соседом подпоследовательности является та подпоследовательность ряда, которая не пересекается с данной и имеет минимальное расстояние до нее.

Недавно разработанный алгоритм MERLIN [5] обеспечивает эффективный поиск диссонансов временного ряда, имеющих длину в заданном диапазоне. Авторами настоящей статьи в предыдущих исследованиях разработан алгоритм PALMAD [6], распараллеливающий MERLIN на платформе графического процессора. Однако MERLIN и PALMAD ограничивают длину временного ряда размером оперативной памяти GPU. В настоящей статье авторы продолжают свои исследования и представляют алгоритм поиска диссонансов временного ряда, имеющих длину в заданном диапазоне, на вычислительном кластере, каждый узел которого оснащен графическим процессором.

Статья имеет следующую структуру. В разделе 1 приводится обзор работ по темати-

ке исследования. Раздел 2 содержит нотацию и формальные определения, а также краткое описание последовательных и параллельных алгоритмов поиска диссонансов, задействованных в данном исследовании. В разделе 3 представлен новый алгоритм поиска диссонансов на вычислительном кластере с GPU-узлами. Результаты вычислительных экспериментов по исследованию эффективности предложенного алгоритма описаны в разделе 4. Заключение подводит итоги исследования и описывает направление будущих работ.

1. Обзор связанных работ

Концепция диссонансов была предложена в работе [4] одновременно с алгоритмом HOTSAX (Heuristically Ordered Time series using Symbolic Aggregate ApproXimation) для их поиска. Диссонанс определяется как подпоследовательность ряда, которая имеет максимальное расстояние до своего ближайшего соседа. Ближайшим соседом подпоследовательности является та подпоследовательность ряда, которая не пересекается с данной и имеет минимальное расстояние до нее. Однако HOTSAX является приближенным алгоритмом, поскольку основан на сжатии подпоследовательностей исходного временного ряда с помощью символьной агрегатной аппроксимации [7]. В работе [8] была предложена схема распараллеливания данного алгоритма для многоядерных процессоров Intel и графических процессоров NVIDIA. Общей особенностью указанных алгоритмов является ограничение длины временного ряда размером, допускающим размещение исходных данных в оперативной памяти.

В работе [9] предложена концепция диапазонных диссонансов и алгоритм DRAG (Discord Range Aware Gathering) для их обнаружения, предполагающий хранение временного ряда на диске, а не в оперативной памяти. Диапазонный диссонанс представляет собой подпоследовательность ряда, расстояние от которой до ее ближайшего соседа не ниже r, где r – заданный порог. DRAG предполагает два сканирования ряда с помощью скользящего окна размером, равным длине искомых диссонансов. На первом проходе выполняется отбор подпоследовательностей-кандидатов в диссонансы, на втором – очистка полученного множества кандидатов от ложноположительных образцов. Позднее в работе [10] те же авторы предложили схему распараллеливания DRAG на основе парадигмы MapReduce, однако в вычислительных экспериментах ограничились симуляцией выполнения алгоритма на распределенной памяти.

Авторами настоящей статьи в работе [11] предложен алгоритм PD3 (Parallel DRAG-based Discord Discovery), распараллеливающий DRAG на платформе графического процессора. В работе [12] авторы настоящей статьи предложили схему распараллеливания DRAG для кластера с многоядерными процессорами, которая состоит из следующих шагов. На первом шаге формируются множества локальных кандидатов в диссонансы путем выполнения на каждом отдельном узле кластера процедур отбора и очистки для соответствующих фрагментов ряда. Далее один из узлов кластера выступает в роли сборщика и формирует множество глобальных кандидатов, объединяя множества локальных кандидатов, полученные от остальных узлов кластера. Затем узел-сборщик рассылает полученное им множество всем узлам кластера, каждый из которых выполняет процедуру очистки глобальных кандидатов для своего фрагмента, формируя тем самым множество локальных диссонансов. Наконец, узел-сборщик формирует итоговый ответ, выполняя пересечение множеств локальных диссонансов, получаемых от всех узлов кластера. В данной схеме применены технологии OpenMP для реализации процедур отбора и очистки на одном узле кластера и MPI для обменов данными между узлами кластера соответственно. В экспериментах данный алгоритм показал существенно бóльшую производительность, чем алгоритмы DDD (Distributed Discord Discovery) [13] и PDD (Parallel Discord Discovery) [14], которые применяют иные схемы распараллеливания, предполагающие интенсивные обмены данными между узлами кластера.

Тем не менее, указанные выше алгоритмы обеспечивают обнаружение диссонансов лишь

только одной (заданной исследователем) длины. Прямолинейным обобщением данного решения на случай диапазона длин диссонансов был бы циклический запуск алгоритма DRAG для каждой из длин в заданном диапазоне. Однако на практике это решение, как правило, является вычислительно невозможным, поскольку требует заново подбирать параметр r на каждом шаге такого цикла. Предложенный авторами алгоритма DRAG новый последовательный алгоритм MERLIN [5] снимает указанное выше ограничение. MERLIN многократно вызывает DRAG и адаптивно подбирает параметр r. В экспериментах MERLIN обнаруживает диссонансы любой длины из заданного диапазона длин, опережая конкурентов как по точности, так и по производительности [5]. Авторами настоящей статьи предложен алгоритм PALMAD (Parallel Arbitrary Length MERLIN-based Anomaly Detection) [6], в котором вычислительная схема MERLIN распараллеливается для графического процессора. PALMAD многократно вызывает PD3 и сокращает объем вычислений за счет применения выведенных авторами рекуррентных формул расчета среднего и стандартного отклонения соседних подпоследовательностей при нахождении расстояний.

Подводя итоги обзора, можно заключить, что в настоящее время MERLIN является одним из наиболее эффективных последовательных алгоритмов поиска аномальных подпоследовательностей временного ряда. Однако вместе с тем пока не предложен подход к распараллеливанию MERLIN на платформе вычислительного кластера с узлами на базе графических процессоров, который мог бы существенно повысить эффективность поиска аномалий больших временных рядов.

2. Предварительный теоретический базис

2.1. Формальные определения и обозначения

Временной ряд (time series) T представляет собой последовательность хронологически упорядоченных вещественных значений:

$$T = (t_1, \ldots, t_n), \quad t_i \in \mathbb{R}. \tag{1}$$

Число n обозначается $|T|$ и называется длиной ряда.

Подпоследовательность (subsequence) $T_{i,\,m}$ временного ряда T представляет собой непрерывный промежуток из m элементов, начиная с позиции i:

$$T_{i,m} = (t_i, \ldots, t_{i+m-1}), \quad 1 \leqslant m \ll n, \quad 1 \leqslant i \leqslant n - m + 1. \tag{2}$$

Множество всех подпоследовательностей ряда T, имеющих длину m, обозначим как S_T^m, а мощность такого множества за N, $N = |S_T^m| = n - m + 1$.

Подпоследовательности $T_{i,\,m}$ и $T_{j,\,m}$ ряда T называются *непересекающимися (non-self match)*, если $|i - j| \geqslant m$. Подпоследовательность, которая является непересекающейся к данной подпоследовательности C, обозначим как M_C.

Подпоследовательность D ряда T является *диапазонным диссонансом (range discord)*, если

$$\min_{M_D \in T} \big(\mathrm{Dist}(D, M_D)\big) \geqslant r, \tag{3}$$

где $\mathrm{Dist}(\cdot\,,\cdot\,)$ представляет собой неотрицательную симметричную функцию, порог расстояния r – наперед заданный параметр. Другими словами, некая подпоследовательность ряда является диапазонным диссонансом, если ее ближайший сосед (ближайшая и не пересекающаяся с ней подпоследовательность) находится на расстоянии не менее чем r. Далее для краткости мы будем использовать термин “диссонанс”, подразумевая диапазонный диссонанс, если не указано обратное.

Рассматриваемые далее последовательные и параллельные алгоритмы поиска диссонансов предполагают, что обрабатываемые подпоследовательности временного ряда предварительно подвергнуты z-нормализации. *Z-нормализация* подпоследовательности (ряда) T

представляет собой подпоследовательность (ряд) $\hat{T} = (\hat{t}_1, \ldots, \hat{t}_m)$, элементы которого вычисляются следующим образом:

$$\hat{t}_i = \frac{t_i - \mu}{\sigma}, \quad \mu = \frac{1}{m}\sum_{i=1}^{m} t_i, \quad \sigma^2 = \frac{1}{m}\sum_{i=1}^{m} t_i^2 - \mu^2. \tag{4}$$

Для обеспечения лучшей производительности в качестве функции расстояния $\mathrm{Dist}(\cdot,\cdot)$ в нашем исследовании применяется квадрат z-нормализованного евклидова расстояния, эффективное вычисление которого производится по следующей формуле [15]:

$$\mathrm{Dist}(X,\, Y) = 2m\left(1 - \frac{X\cdot Y - m\cdot \mu_X \cdot \mu_Y}{m\cdot \sigma_X \cdot \sigma_Y}\right), \tag{5}$$

где $X\cdot Y$ обозначает скалярное произведение векторов $X, Y \in \mathbb{R}^m$.

2.2. Последовательные алгоритмы DRAG и MERLIN

Алгоритм DRAG [9] предполагает две фазы: отбор и очистка кандидатов, где выполняется соответственно поиск потенциальных диссонансов и отбрасывание ложноположительных экземпляров. На первой фазе множество кандидатов в диссонансы $\mathcal{C}$ инициализируется первой подпоследовательностью ряда, имеющей заданную длину m. Далее DRAG просматривает временной ряд T и для каждой подпоследовательности $s \in S_T^m$ проверяет, что каждый кандидат $c \in \mathcal{C}$ является диссонансом в соответствии с определением (3). Если кандидат c не проходит проверку, то он удаляется из $\mathcal{C}$. В конце концов, s либо добавляется в множество кандидатов как потенциальный диссонанс, либо удаляется из него. На второй фазе алгоритм инициализирует расстояния всех кандидатов до их ближайших соседей значением $+\infty$ и сканирует ряд, вычисляя расстояние между каждой подпоследовательностью $s \in S_T^m$ и каждым кандидатом c. Если расстояние меньше r, то кандидат является ложноположительным и навсегда удаляется из $\mathcal{C}$. Если вышеупомянутое расстояние меньше текущего лучшего (минимального) расстояния до ближайшего соседа, то обновляется текущее лучшее расстояние до ближайшего соседа.

Алгоритм MERLIN [5] предписывает следующую процедуру подбора параметра r. Поиск диссонансов выполняется последовательно для каждого значения длины диссонанса в диапазоне $minL..maxL$. На каждом шаге MERLIN вычисляет среднее арифметическое μ и стандартное отклонение σ расстояний от последних пяти найденных диссонансов до их ближайших соседей, а затем вызывает алгоритм DRAG, передавая ему параметр $r = \mu - 2\sigma$. Если DRAG не обнаружил диссонанс, то σ вычитается из r до тех пор, пока DRAG не завершится успешно (т.е. пока не будет найден диссонанс). Для первых пяти длин диссонансов параметр r устанавливается следующим образом. Для диссонансов длины $minL$ берется значение $r = 2\sqrt{minL}$, поскольку это максимально возможное расстояние между любой парой подпоследовательностей длины $minL$, и тогда r уменьшается вдвое до тех пор, пока DRAG с таким параметром не завершится успешно. Чтобы обработать следующие четыре длины диссонанса, DRAG принимает расстояние от диссонанса до его ближайшего соседа, полученное на предыдущем шаге, за вычетом небольшого значения, равного 1%. Вычитание дополнительного 1% продолжается до тех пор, пока обнаружение диссонанса с таким параметром не приведет к успеху.

2.3. Параллельные алгоритмы PD3 и PALMAD

Алгоритм PD3 [11] распараллеливает каждую из фаз отбора и очистки алгоритма DRAG на платформе GPU на основе концепции параллелизма по данным. Временной ряд разбивается на *сегменты равной длины*, каждый из которых обрабатывается отдельным блоком нитей GPU. *Параллельный отбор кандидатов* в диссонансы организуется следую-

щим образом. Блок нитей полагает подпоследовательности сегмента (локальными) кандидатами и выполняет обработку тех из них, которые не пересекаются с кандидатами и расположены справа от данного сегмента, следующим образом. Если расстояние от кандидата до подпоследовательности меньше порога r, то кандидат и подпоследовательность исключаются из дальнейшей обработки как заведомо не являющиеся диссонансами. Если все локальные кандидаты отброшены, блок досрочно завершает работу. Блок нитей обрабатывает подпоследовательности ряда порциями, количество элементов в которых равно длине сегмента. Используя формулу (5), блок вычисляет расстояния от всех подпоследовательностей своего сегмента до всех подпоследовательностей текущей порции следующим образом. Сначала нити блока вычисляют скалярные произведения между первой подпоследовательностью сегмента и всеми подпоследовательностями текущей порции и сохраняют результаты в массиве, хранящемся в разделяемой памяти GPU. Затем они вычисляют скалярные произведения между первой подпоследовательностью текущей порции и всеми подпоследовательностями сегмента, сохраняя результат в другом массиве, также хранящемся в разделяемой памяти. Далее на основе полученных результатов вычисляются расстояния между первой подпоследовательностью порции и всеми подпоследовательностями сегмента. С помощью вычисленного расстояния отбрасываются бесперспективные кандидаты в сегменте и текущей порции. Если отброшены все кандидаты сегмента, блок заканчивает работу. После этого нити выполняют аналогичные действия над оставшимися подпоследовательностями текущей порции. *Параллельная очистка кандидатов* сходна с описанной выше процедурой параллельного отбора. В очистке участвуют те сегменты ряда, множество локальных кандидатов которых не пусто. Очистка кандидатов сегмента заключается в сканировании и обработке подпоследовательностей ряда, которые не пересекаются с кандидатами и расположены слева от данного сегмента. Если расстояние от кандидата до подпоследовательности меньше порога r, то кандидат отбрасывается.

Алгоритм PALMAD [6] в целом повторяет вычислительную схему оригинального алгоритма MERLIN на графическом процессоре, многократно вызывая алгоритм PD3. Перед первым вызовом PD3 выполняется параллельное вычисление двух векторов, хранящих соответственно средние значения и стандартные отклонения всех подпоследовательностей ряда, имеющих длину, равную заданной минимальной длине диссонанса. Далее указанные векторы обеспечивают сокращение объема вычислений за счет применения следующих выведенных авторами рекуррентных формул расчета среднего и стандартного отклонения соседних подпоследовательностей при нахождении расстояний:

$$\mu_{T_{i,\,m+1}} = \frac{1}{m+1}\big(m\mu_{T_{i,\,m}} + t_{i+m}\big), \tag{6}$$

$$\sigma^2_{T_{i,\,m+1}} = \frac{m}{m+1}\Big(\sigma^2_{T_{i,\,m}} + \frac{1}{m+1}\big(\mu_{T_{i,\,m}} - t_{i+m}\big)^2\Big). \tag{7}$$

3. Поиск диссонансов на кластере с GPU-узлами

3.1. Общая схема вычислений

Рис. 1 отражает общую схему вычислений предлагаемого алгоритма. Новый алгоритм модифицирует схему вычислений, реализованную авторами ранее в алгоритме PALMAD [6], применяя концепцию параллелизма по данным. В модифицированной схеме ряд разбивается на фрагменты, распределяемые по узлам вычислительного кластера, и каждый узел выполняет многократные вызовы алгоритма PD3 [11] для обработки собственного фрагмента на графическом процессоре. При этом выполнении процедуры подбора параметра r требуются обмены данными между узлами: для каждого фиксированного значения r каждому узлу кластера необходимо проверить, что найденные в текущем фрагменте кандидаты в диссонансы являются таковыми для фрагментов всех остальных узлов.

Для описания предложенного алгоритма введем следующие обозначения. Пусть име-

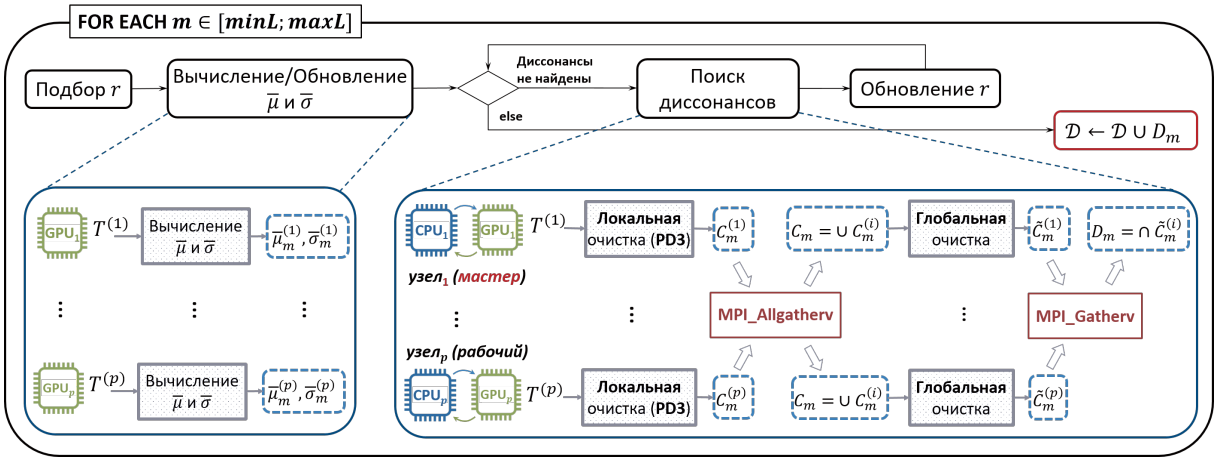


Рис. 1. Общая схема вычислений

ется временной ряд T, $|T| = n$, и требуется найти все его диссонансы, имеющие длину в диапазоне $minL..maxL$ ($3 \leqslant minL \leqslant maxL \ll n$). Пусть алгоритму выделено p узлов вычислительного кластера, каждый из которых оснащен центральным и графическим процессорами.

Временной ряд разбивается на фрагменты равной длины по числу доступных алгоритму узлов кластера, обозначаемые как $T^{(1)}, \ldots, T^{(p)}$. При этом в конец каждого фрагмента $T^{(i)}$ (за исключением последнего $T^{(p)}$) добавляется $maxL - 1$ элементов, взятых из начала следующего фрагмента $T^{(i+1)}$, для предотвращения потери результатов на стыке фрагментов.

Далее выполняется выравнивание каждого фрагмента для обеспечения баланса загрузки нитей графического процессора, обрабатывающих фрагмент. Длина фрагмента устанавливается кратной размеру варпа графического процессора. Если таковая кратность не имеет место, то фрагмент дополняется справа фиктивными элементами, имеющими значение $+\infty$. Мы предполагаем, что в итоге описанных преобразований фрагмент ряда может быть целиком размещен в оперативной памяти как центрального, так и графического процессоров.

Для поиска множества диссонансов $\mathcal{D}_m$, имеющих длину m ($minL \leqslant m \leqslant maxL$) каждый шаг подбора параметра r на вычислительном кластере с графическими процессорами выполняется по следующей схеме. Сперва каждый i-й узел выполняет *локальный отбор*: с помощью алгоритма PD3 находит во фрагменте $T^{(i)}$ множество $\mathcal{C}_m^{(i)}$ локальных кандидатов в диссонансы. Узлы кластера осуществляют обмен найденными локальными кандидатами по принципу "каждый с каждым", после чего на каждом узле формируется множество глобальных кандидатов в диссонансы $\mathcal{C}_m = \cup_{i=1}^{p} \mathcal{C}_m^{(i)}$. Далее каждый узел выполняет *глобальную очистку* множества $\mathcal{C}$ от ложноположительных диссонансов (см. ниже раздел 3.2) и формирует множество локальных диссонансов $\tilde{\mathcal{C}}_m^{(i)}$. После этого один из узлов вычислительного кластера (например, нулевой узел) объявляется мастером, остальные – рабочими. Каждый узел-рабочий отправляет узлу-мастеру свое множество локальных диссонансов, после чего мастер формирует результирующее множество диссонансов заданной длины как $\mathcal{D}_m = \cap_{i=1}^{p} \tilde{\mathcal{C}}_m^{(i)}$. Пороговое расстояние вычисляется как $r = \max\limits_{c \in \mathcal{D}_m} c.nnDist$, где запись $c.nnDist$ означает расстояние от кандидата в диссонансы c до его ближайшего соседа.

По завершении шага подбора параметра r узел-мастер пополняет итоговое множество диссонансов исходного ряда $\mathcal{D}$, добавляя в него только что найденное множество $\mathcal{D}_m$. Таким образом, итоговое множество диссонансов вычисляется как $\mathcal{D} = \cup_{m=minL}^{maxL} \mathcal{D}_m$.

В описанной выше схеме мы используем функции стандарта MPI `MPI_Allgatherv` и `MPI_Gatherv` для реализации приема-передачи локальных кандидатов и локальных диссонансов соответственно.

3.2. Распараллеливание глобальной очистки кандидатов

Глобальная очистка заключается в проведении каждым узлом кластера очистки множества кандидатов $\mathcal{C}_m$ для множества подпоследовательностей соответствующего фрагмента $S^m_{T^{(i)}}$ в соответствии с процедурой, предусмотренной в алгоритме DRAG (см. раздел 2.2). В соответствии с этим нам необходимо вычислить расстояния от каждого кандидата $c \in \mathcal{C}_m$ до каждой подпоследовательности $s \in S^m_{T^{(i)}}$, используя формулу (5).

В памяти графического процессора узла кластера множества подпоследовательностей-кандидатов $\mathcal{C}_m$ и подпоследовательностей фрагмента $S^m_{T^{(i)}}$ представляются в виде матриц $C \in \mathbb{R}^{|\mathcal{C}_m| \times m}$ и $S \in \mathbb{R}^{N \times m}$ соответственно. Для хранения средних арифметических и стандартных отклонений подпоследовательностей-кандидатов предусматриваются векторы $\overline{\mu}, \overline{\sigma} \in \mathbb{R}^{|\mathcal{C}_m|}$ соответственно. Для хранения вычисленных расстояний используется массив $nnDist \in \mathbb{R}^{|\mathcal{C}_m|}$.

Результатом умножения матриц S и C является матрица $DP \in \mathbb{R}^{N \times |\mathcal{C}_m|}$, каждый элемент которой представляет скалярное произведение соответствующих строк матрицы S (подпоследовательностей) и столбцов транспонированной матрицы C^{T} (кандидатов). Для выполнения указанного умножения на графическом процессоре формируется двумерная сетка нитей, состоящая из блоков нитей размера $block \times block$, где параметр $block$ (количество нитей в блоке) выбирается кратным размеру варпа и не превышает максимальное количество нитей в блоке. Каждый блок нитей вычисляет одну подматрицу (тайл) $Tile_{\mathrm{DP}} \in \mathbb{R}^{block \times block}$, а нить блока отвечает за вычисление одного элемента тайла.

Далее каждая нить блока, имеющая координаты (i, j), используя формулу (5), вычисляет расстояние между кандидатом $C(i, \cdot)$ и подпоследовательностью $S(j, \cdot)$ как

$$dist = 2m \left(1 - \frac{DP(i, j) - m \cdot \overline{\mu}(i) \cdot \overline{\mu}(j)}{m \cdot \overline{\sigma}(i) \cdot \overline{\sigma}(j)}\right).$$

Затем, используя операцию атомарного минимума (выполняемого над данными в приватной памяти текущей нити), блок нитей вычисляет расстояние от кандидата $C(i, \cdot)$ до его ближайшего соседа как минимум всех вышеуказанных расстояний, помещая результат в $nnDist(i)$.

На финальном шаге глобальной очистки, используя подсчитанные выше расстояния до ближайших соседей кандидатов, множество локальных диссонансов формируется как $\tilde{\mathcal{C}}^{(i)}_m = \{c \in \mathcal{C}^{(i)}_m \mid c.nnDist \geqslant r\}$.

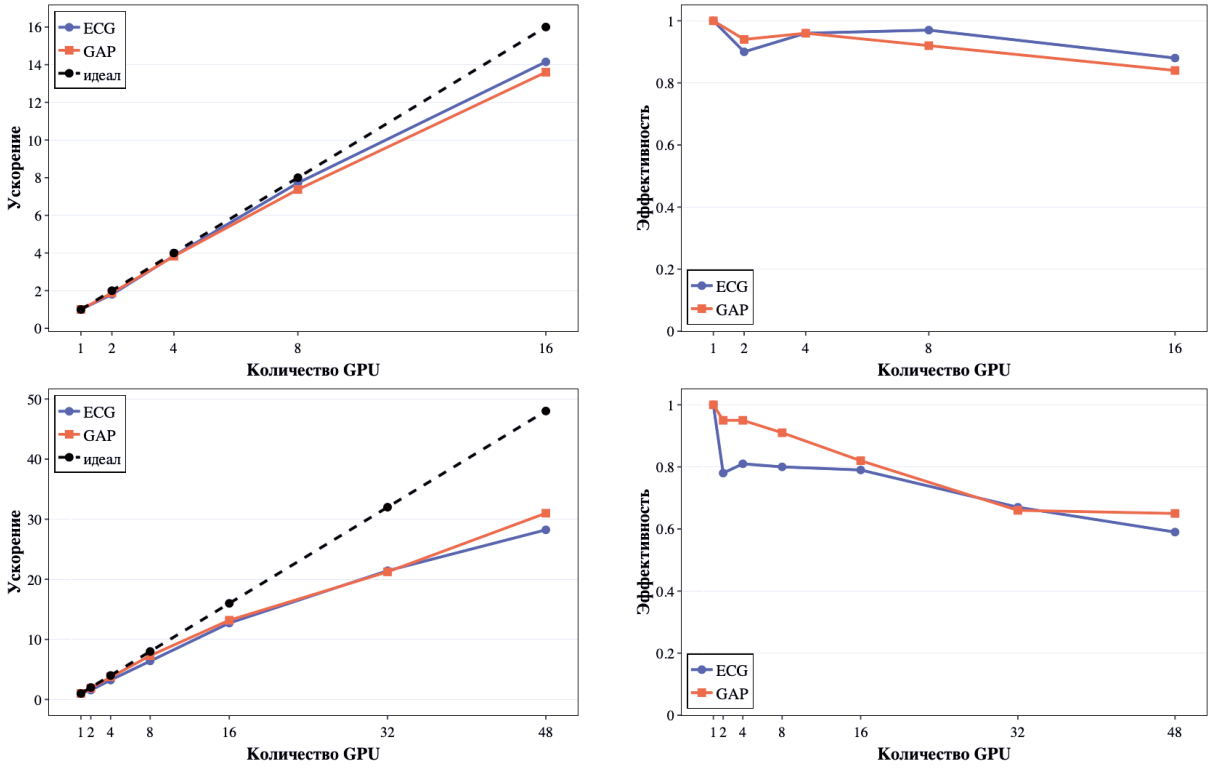
4. Вычислительные эксперименты

Для исследования эффективности разработанного алгоритма нами были проведены вычислительные эксперименты на платформе суперкомпьютера Ломоносов-2 [16] для следующих двух конфигураций вычислительного кластера с узлами на базе графического процессора. *Конфигурация 16×K40* задействовала 16 узлов раздела Compute суперкомпьютера, каждый из которых оснащен графическим процессором NVIDIA Tesla K40 (2 880 ядер @745 МГц, пиковая производительность 1.682 TFLOPS для арифметики двойной точности). *Конфигурация 48×P100* использовала 48 узлов раздела Pascal суперкомпьютера, на каждом из которых установлен графический процессор NVIDIA Tesla P100 (3 584 ядер @1 328 МГц, пиковая производительность 5.3 TFLOPS для арифметики двойной точности).

В экспериментах исследовались производительность, ускорение и эффективность распараллеливания алгоритма, определяемые следующим образом. *Производительность* представляет собой усредненное по 10 запускам время работы алгоритма, не включающее затраты на ввод исходных данных и вывод полученных результатов. *Ускорение* алгоритма $s(p)$ на p узлах кластера вычисляется как $s(p) = \frac{t_1}{t_p}$, где t_1 и t_p – производительность соответственно ранее разработанного алгоритма PALMAD на одном графическом процессоре узла кластера и алгоритма, разработанного в рамках данной статьи, на p узлах кластера.

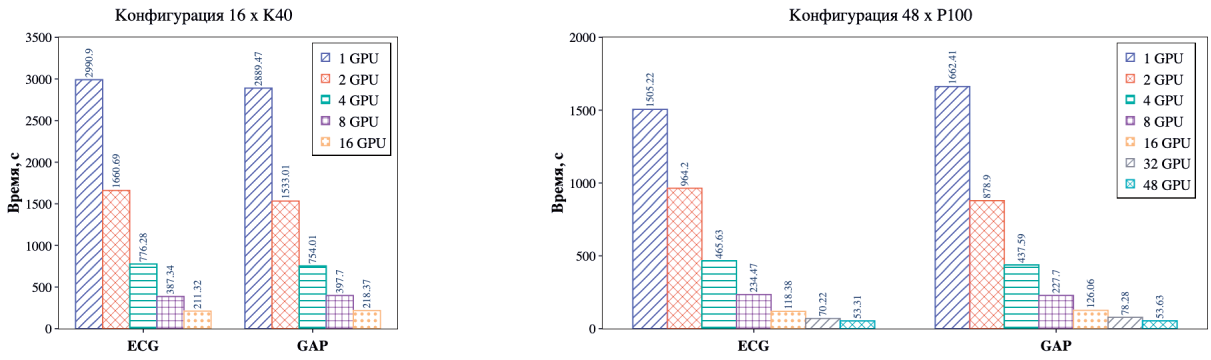
Эффективность распараллеливания алгоритма $e(p)$ на p узлах кластера определяется как $e(p) = \frac{s(p)}{p}$.

В экспериментах использовались следующие *наборы данных*. Ряд ECG [17] содержит показания электрокардиограммы взрослого пациента и имеет длину 2 000 000 точек. Ряд GAP [18] представляет собой поминутные показатели общего энергопотребления частного дома во Франции в период 2006–2010 гг. и содержит 2 000 000 точек. Для обоих рядов поиск диссонансов осуществлялся в диапазоне длин 64..128.



(a) Ускорение

(b) Эффективность

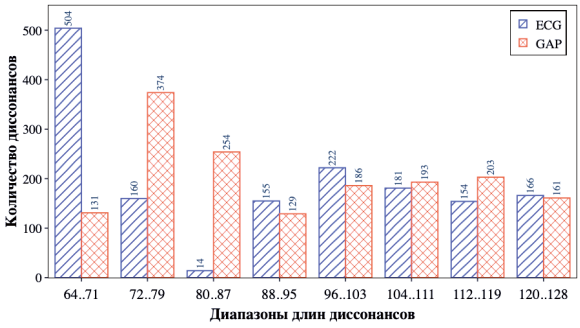


(c) Производительность

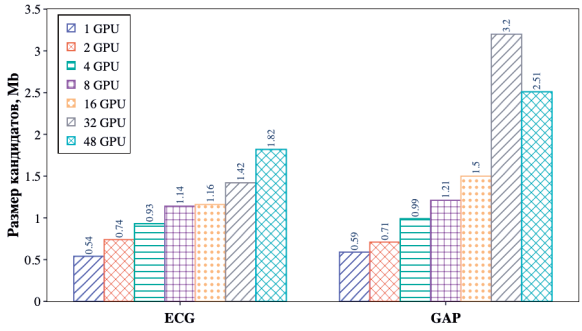
Рис. 2. Масштабируемость алгоритма на различных конфигурациях вычислительного кластера

Графики, представленные на рис. 2, отражают масштабируемость алгоритма в проведенных экспериментах. Можно видеть, что для конфигурации 16×K40 при поиске диссонансов в обоих рядах разработанный алгоритм демонстрирует близкие к линейным ускорение и эффективность (см. верхние графики рис. 2a и 2b соответственно). В случае конфигурации 48×P100 ускорение и эффективность для обоих рядов становятся сублинейными (см.

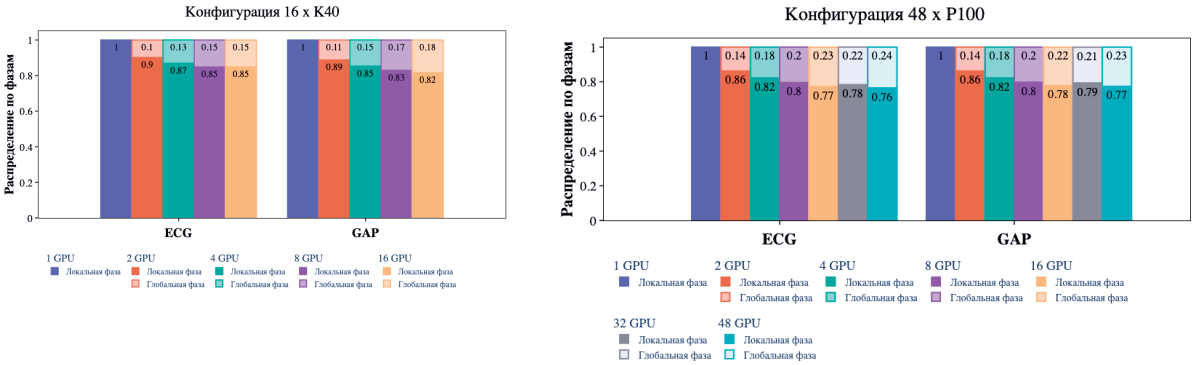
нижние графики рис. 2a и 2b соответственно). Производительность алгоритма в случае конфигурации 16×K40 ожидаемо ниже, чем для конфигурации 48×P100, которая задействует более мощные графические процессоры в бóльшем количестве (см. графики слева и справа рис. 2c соответственно).



(a) Количество найденных диссонансов



(b) Размер обработанных кандидатов



(c) Распределение времени по фазам алгоритма

Рис. 3. Статистические итоги работы алгоритма

Снижение масштабируемости алгоритма при переходе к конфигурации с бóльшим количеством узлов объясняется возрастающими при этом накладными расходами на обмены кандидатами между узлами. Тем не менее, масштабируемость алгоритма сохраняет линейный характер и ее деградация отсутствует. Отметим также, что для решения задачи поиска диссонансов на параллельных системах с распределенной памятью обмены кандидатами неизбежны, поскольку применение парадигмы MapReduce приведет к неадекватному результату: кандидаты в диссонансы, найденные в одном фрагменте ряда, но очищенные в рамках того же фрагмента, очевидно, не обязаны являться диссонансами ряда в смысле определения (3).

Статистические итоги поиска диссонансов (см. рис. 3) показывают следующее. В обоих рядах распределение найденных диссонансов по длинам примерно одинаковое (см. рис. 3a), а общее количество найденных диссонансов не превышает 0.08% от общего числа подпоследовательностей ряда с искомыми длинами, что согласуется с интуитивными представлениями об аномалиях.

Размер кандидатов, пересылаемых между узлами кластера (см. рис. 3b), определяет распределение времени работы алгоритма по его фазам. Локальная фаза подразумевает вычисления на графических процессорах узлов, необходимые для отбора и локальной очистки кандидатов. Глобальная фаза включает в себя пересылки кандидатов между узлами и вычисления на узле-мастере, необходимые для получения результирующего множества диссонансов. Бóльшее время выполнение глобальной фазы соответствует меньшей масштабируемости (см. рис. 3c и рис. 2a).

Заключение

В статье рассмотрена проблема обнаружения аномалий во временных рядах, которая в настоящее время является актуальной в широком спектре предметных областей: цифровая индустрия, здравоохранение, моделирование климата и прогноз погоды, финансовая аналитика и др. Для формализации понятия аномалии применяется концепция диссонанса [9] временного ряда. Подпоследовательность ряда является диссонансом, если ее ближайший сосед находится на расстоянии не менее чем заданный аналитиком порог r; ближайшим соседом называют подпоследовательность, которая не пересекается с данной и находится на минимальном расстоянии от нее. Алгоритм DRAG [9] реализует последовательный поиск диссонансов фиксированной длины. Алгоритм MERLIN [5] обеспечивает последовательный поиск диссонансов, имеющих длину в заданном диапазоне длин, и применяет многократные вызовы DRAG. В предыдущих исследованиях авторы настоящей статьи разработали параллельные версии вышеуказанных алгоритмов для графического процессора, соответственно PD3 [11] и PALMAD [6], где PALMAD применяет многократные вызовы PD3. Однако PALMAD ограничивает длину ряда размером оперативной памяти GPU. В настоящей статье предложен новый алгоритм поиска диссонансов временного ряда, имеющих длину в заданном диапазоне длин, на вычислительном кластере, каждый узел которого оснащен графическим процессором.

Предложенный алгоритм модифицирует схему вычислений PALMAD следующим образом. Временной ряд разбивается на фрагменты, распределяемые по узлам вычислительного кластера, и каждый узел выполняет многократные вызовы алгоритма PD3 для обработки собственного фрагмента на графическом процессоре. Алгоритм предусматривает цикл по диапазону длин диссонансов, где на каждом шаге подбор параметра r выполняется по следующей схеме. Сперва каждый узел выполняет локальный отбор: с помощью алгоритма PD3 находит во фрагменте множество локальных кандидатов в диссонансы. Далее с помощью технологии MPI узлы кластера осуществляют обмен полученными результатами по принципу “каждый с каждым”, и на каждом узле формируется множество глобальных кандидатов как объединение всех локальных кандидатов. Затем каждый узел выполняет глобальную очистку: из множества глобальных кандидатов удаляются ложноположительные диссонансы, формируя тем самым множество локальных диссонансов. Процедура глобальной очистки распараллеливается на основе блочного умножения матрицы подпоследовательностей-кандидатов и матрицы подпоследовательностей фрагмента. После этого один из узлов кластера объявляется мастером, остальные – рабочими. Каждый рабочий отправляет мастеру свое множество локальных диссонансов, после чего мастер формирует результирующее множество диссонансов рассматриваемой длины как пересечение множеств, полученных от рабочих и определяет порог как максимальное расстояние до ближайшего соседа среди только что найденных диссонансов.

Вычислительные эксперименты, проведенные на платформе суперкомпьютера Ломоносов-2 с временными рядами из реальных предметных областей, показывают близкие к линейным ускорение и эффективность разработанного алгоритма.

В будущих исследованиях мы планируем расширить предложенный алгоритм на случай, когда узел вычислительного кластера оснащается несколькими графическими процессорами.

Благодарности

Работа выполнена при финансовой поддержке Российского научного фонда (грант № 23-21-00465).

В исследованиях использовано оборудование Центра коллективного пользования сверхвысокопроизводительными вычислительными ресурсами МГУ имени М.В. Ломоносова.

Литература

1. Blázquez-Garcıa A., Conde A., Mori U., Lozano J.A. A review on outlier/anomaly detection in time series data // ACM Comput. Surv. 2021. Vol. 54, no. 3. 56:1–56:33. DOI: 10.1145/3444690.
2. Chandola V., Banerjee A., Kumar V. Anomaly detection: A survey // ACM Comput. Surv. 2009. Vol. 41, no. 3. 15:1–15:58. DOI: 10.1145/1541880.1541882.
3. Chandola V., Cheboli D., Kumar V. Detecting anomalies in a time series database. Retrieved from the University of Minnesota Digital Conservancy. 2009. Accessed: 2022-04-12. https://hdl.handle.net/11299/215791.
4. Lin J., Keogh E.J., Fu A.W., Herle H.V. Approximations to magic: Finding unusual medical time series // 18th IEEE Symposium on Computer-Based Medical Systems (CBMS 2005), 23-24 June 2005, Dublin, Ireland. IEEE Computer Society, 2005. P. 329–334. DOI: 10.1109/CBMS.2005.34.
5. Nakamura T., Imamura M., Mercer R., Keogh E.J. MERLIN: parameter-free discovery of arbitrary length anomalies in massive time series archives // 20th IEEE International Conference on Data Mining, ICDM 2020, Sorrento, Italy, November 17-20, 2020 / ed. by C. Plant, H. Wang, A. Cuzzocrea, *et al.* IEEE, 2020. P. 1190–1195. DOI: 10.1109/ICDM50108.2020.00147.
6. Zymbler M., Kraeva Y. High-performance Time Series Anomaly Discovery on Graphics Processors // CoRR. 2023. Vol. abs/2304.01660. arXiv: 2304.01660. URL: https://arxiv.org/abs/2304.01660.
7. Lin J., Keogh E.J., Lonardi S., Chiu B.Y. A symbolic representation of time series, with implications for streaming algorithms // Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery, DMKD 2003, San Diego, California, USA, June 13, 2003 / ed. by M.J. Zaki, C.C. Aggarwal. ACM, 2003. P. 2–11. DOI: 10.1145/882082.882086.
8. Zymbler M. A parallel discord discovery algorithm for time series on many-core accelerators // Numerical Methods and Programming. 2019. Vol. 20, no. 3. P. 211–223. (in Russian) DOI: 10.26089/NumMet.v20r320.
9. Yankov D., Keogh E.J., Rebbapragada U. Disk aware discord discovery: Finding unusual time series in terabyte sized datasets // Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007), October 28-31, 2007, Omaha, Nebraska, USA. IEEE Computer Society, 2007. P. 381–390. DOI: 10.1109/ICDM.2007.61.
10. Yankov D., Keogh E.J., Rebbapragada U. Disk aware discord discovery: Finding unusual time series in terabyte sized datasets // Knowl. Inf. Syst. 2008. Vol. 17, no. 2. P. 241–262. DOI: 10.1007/s10115-008-0131-9.
11. Kraeva Y., Zymbler M. A parallel discord discovery algorithm for a graphics processor // Pattern Recognition and Image Analysis. 2023. Vol. 33, no. 2. P. 101–113. DOI: 10.1134/S1054661823020062.
12. Zymbler M., Grents A., Kraeva Y., Kumar S. A parallel approach to discords discovery in massive time series data // Computers, Materials & Continua. 2021. Vol. 66, no. 2. P. 1867–1878. DOI: 10.32604/cmc.2020.014232.

13. Wu Y., Zhu Y., Huang T., *et al.* Distributed discord discovery: Spark based anomaly detection in time series // 17th IEEE International Conference on High Performance Computing and Communications, HPCC 2015, 7th IEEE International Symposium on Cyberspace Safety and Security, CSS 2015, and 12th IEEE International Conference on Embedded Software and Systems, ICESS 2015, New York, NY, USA, August 24-26, 2015. IEEE, 2015. P. 154–159. DOI: 10.1109/HPCC-CSS-ICESS.2015.228.

14. Huang T., Zhu Y., Mao Y., *et al.* Parallel discord discovery // Advances in Knowledge Discovery and Data Mining - 20th Pacific-Asia Conference, PAKDD 2016, Auckland, New Zealand, April 19-22, 2016, Proceedings, Part II. Vol. 9652 / ed. by J. Bailey, L. Khan, T. Washio, *et al.* Springer, 2016. P. 233–244. Lecture Notes in Computer Science. DOI: 10.1007/978-3-319-31750-2_19.

15. Mueen A., Nath S., Liu J. Fast approximate correlation for massive time-series data // Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010 / ed. by A.K. Elmagarmid, D. Agrawal. ACM, 2010. P. 171–182. DOI: 10.1145/1807167.1807188.

16. Voevodin V.V., Antonov A.S., Nikitenko D.A., *et al.* Supercomputer Lomonosov-2: large scale, deep monitoring and fine analytics for the user community // Supercomput. Front. Innov. 2019. Vol. 6, no. 2. P. 4–11. DOI: 10.14529/jsfi190201.

17. Goldberger A.L., Amaral L.A.N., Glass L., *et al.* PhysioBank, PhysioToolkit, and PhysioNet components of a new research resource for complex physiologic signals // Circulation. 2000. Vol. 101, no. 23. P. 215–220. DOI: 10.1161/01.CIR.101.23.e215.

18. Individual household electric power consumption. 2023. Accessed: 2023-04-18. https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption/.

Расчеты барьерных свойств полимерных материалов с использованием пакета MULTICOMP

А.А. Книжник [1,2], П.В. Комаров [3,4], А.С. Синица[1,2], Д.Б. Ширабайкин [1], С.В. Трепалин [1], Б.В. Потапкин [1,2]

[1] ООО "Кинтех Лаб",
[2] Национальный исследовательский центр “Курчатовский институт”,
[3] Институт элементоорганических соединений РАН,
[4] Тверской государственный университет

Описано расширение функционала программного комплекса MULTICOMP за счет добавления расчетного модуля «Permeability», предназначенного для оценки газопроницаемости термопластических полимеров. Модуль «Permeability» построен на основе методов полноатомной молекулярной динамики и Монте-Карло для большого канонического ансамбля. В ходе тестовых расчетов проверена возможность выполнения распределенных вычислений по клиент-серверной технологии с переносом ресурсоёмких вычислений на суперкомпьютер. Для выбранных полимеров (полиэтилен, полиэтилентерефталат, политетрафторэтилен и поливинилиденфторид) получено согласие расчетных значений коэффициента проницаемости по кислороду с экспериментом. На основе полученных результатов сделан вывод о том, что MULTICOMP, дополненный модулем «Permeability», применим к решению задач анализа газопроницаемости полимеров и создания новых мембранных и барьерных материалов.

Ключевые слова: полимеры, мембраны, барьерные материалы, газопроницаемость, диффузия, цифровое материаловедение.

1. Введение

Газопроницаемость является одним из основных свойств при оценке эксплуатационных характеристик мембранных, упаковочных и барьерных материалов, предназначенных для обеспечения длительного времени хранения пищевых продуктов и лекарственных препаратов, продления сроков эксплуатации компонентов электронных устройств, фильтрации, газоразделения и др. [1]. Одной из основных проблем, связанных с сохранностью продуктов питания и лекарственных препаратов, является их относительно быстрая деградация из-за контакта с молекулами кислорода и водяного пара. Это объясняется тем, что пищевые продукты и биологически активные компоненты лекарств содержат много химически активных групп, легко вступающих в реакции окисления и гидрирования. Поэтому контакт даже с небольшим количеством кислорода или воды может значительно снизить их потребительские свойства и постепенно привести к полной порче [2,3].

За последние годы широкое распространение получили барьерные материалы на основе гибких многослойных полимерных пленок. Они удобны как непосредственно для производства упаковок скоропортящихся продуктов, так и для тарных пакетов и контейнеров для хранения молока, молодого вина, лекарств и др.). Как правило, коммерческие многослойные пленки состоят из 3-9 слоев, содержащих различные полимеры с дополнительными слоями из алюминия для достижения ультранизких значений газопроницаемости [4,5]. Также в настоящее время активно изучается возможность использования биоразлагаемых полимерных пленок в тандеме с биологически инертными наполнителями из алюмосиликатов, чтобы соответствовать современным тенденциям по сокращению полимерных отходов [6]. При этом использование прослойки из алюминия увеличивает загрязняющий эффект на окружающую среду и снижает возможности по переработки изделий на такой основе.

Дальнейшее совершенствование упаковочных материалов [7-10] требует интенсивных исследований, объединяющих усилия больших коллективов химиков и инженеров, а также исполь-

зования сложного лабораторного оборудования. При этом отдельно стоит задача выбора подходящих полимерных материалов каждого слоя для достижения целевой газопроницаемости. В этом случае применение компьютерного моделирования может помочь уменьшить общие затраты времени на проведение исследований. Для обеспечения таких работ мы расширили функционал программного комплекса MULTICOMP [11], который разрабатывается нами как открытая платформа, интегрирующая набор универсальных инструментов для молекулярного моделирования полимеров наполненных наночастицами. С помощью MULTICOMP пользователь может создавать гибкие сценарии расчета различных свойств полимерных материалов. В данной работе мы представляем новый модуль «Permeability» (проницаемость), добавленный в состав комплекса MULTICOMP. «Permeability» выполняет оценки газопроницаемости полимерных материалов с использованием персональных и суперкомпьютерных вычислительных систем. При разработке модуля использована методология из работ [12-16] в которых исследовались вопросы адсорбции, растворимости, проницаемости и сепарации для таких газов как H_2, He, N_2, O_2, CH_4 и CO_2.

2. Организация модуля Permeability и использованная методология

Модуль «Permeability» ориентирован на проведение быстрых количественных оценок барьерных свойств полимерных материалов. На вход модуля передается атомистическая структура образца исследуемого материала, подготовленная на основе информации о его химическом строении. Расчет проницаемости реализован как комбинация двух методов - молекулярной динамики (МД) и Монте-Карло для большого канонического ансамбля (Grand Canonical Monte Carlo, МКБКА) [12-14]. Логическая организация реализованной расчетной схемы показана на рис. 1. При проведении вычислений пользователи имеют возможность применять такие валентно-силовые поля (ВСП), как PCFF [17], DREIDING [18] и COMPASS II [19], и гибко настраивать параметры обоих методов. В данной работе все расчёты проводились с использованием ВСП PCFF.

На первом шаге расчетной схемы модуля «Permeability» происходит оптимизация и предварительный отжиг геометрии исследуемого материала (абсорбента). Затем в подготовленный образец с помощью метода МКБКА случайно внедряются или удаляются молекулы газов (пенетранты). На следующем шаге производится оптимизация геометрии абсорбента с пенетрантом. Готовые модели молекулярных структур используются для расчета коэффициентов и растворимости S и диффузии D. Полученные значения используются для вычисления коэффициента проницаемости [13,14]:

$$P = D \cdot S \,. \tag{1}$$

Таким образом, в модуле «Permeability» последовательно проводятся два независимых процесса моделирования: (I) расчёт коэффициента растворимости с использованием методов МД + МКБКА и (II) МД-расчет коэффициента диффузии, которые были реализованы на основе интегрированного в программный комплекс MULTICOMP кода LAMMPS [20].

Коэффициент диффузии D вычисляется по формуле Эйнштейна [13,14] на основе вычисления среднеквадратичного смещения молекул-пенетранта:

$$D = \frac{1}{6N} \lim_{t\to\infty} \frac{d}{dt} \sum_{i=1}^{N} \left\langle (\mathbf{r}_i(t) - \mathbf{r}_i(0))^2 \right\rangle , \tag{2}$$

здесь $\mathbf{r}_i(t)$ и $\mathbf{r}_i(0)$ радиус векторы мгновенных и начальных положений центра масс молекул пенетранта, N – число диффундирующих молекул. На больших временах МД моделирования, если среднеквадратичное смещение можно аппроксимировать прямой (выход на диффузионный режим), значение коэффициента диффузии дается следующей упрощенной зависимостью: $D = k/6$, где k - тангенс угла наклона линейного участка среднеквадратичного смещения.

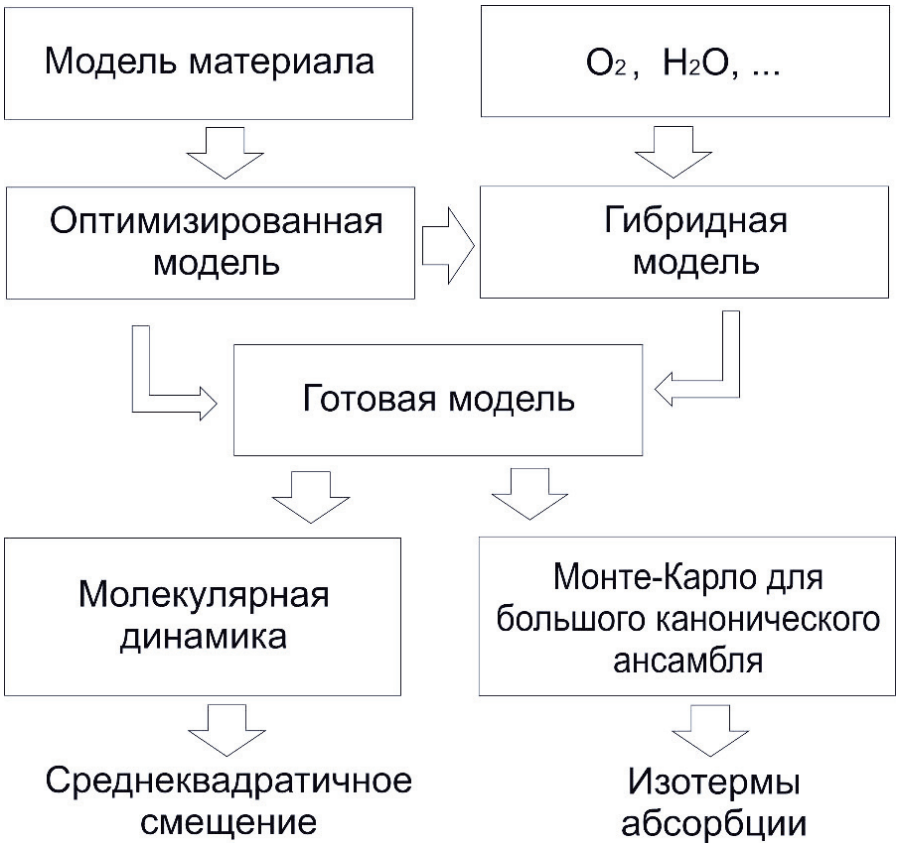


Рис. 1. Общая схема методики расчета проницаемости полимерных материалов, реализованная в рамках модуля «Permeability».

Коэффициент растворимости оценивается в рамках метода МКБКА. В этом случае производится расчет кривых изотерм адсорбции при фиксированной температуре T и объеме ячейки моделирования V. Предполагается, что число молекул в системе меняется при изменении давления газа-пенетранта P_{gas} приблизительно линейно, в соответствии с законом Генри $S = k_G P_{gas}$ (k_G - константа Генри). Поэтому коэффициент растворимости вычисляется посредством аппроксимации изотермы адсорбции с помощью прямой линии $N_{cell}(P_{gas})$ (где N_{cell} – рассчитанное равновесное число частиц в ячейке моделирования для заданного давления P_{gas}) и вычислением тангенса угла наклона построенной прямой при стремлении давления к нулю.

3. Подготовка образцов полимерных материалов и методика расчетов

Для проведения тестовых расчетов мы выбрали четыре варианта крупнотоннажных термопластов, состоящих из линейных молекул, такие как полиэтилен $[C_2H_4]_n$ (ПЭ), полиэтилентерефталат (ПЭТ) и два фторсодержащих полимера, политетрафторэтилен $[C_2F_4]_n$ (ПТФЭ) и поливинилиденфторид $[C_2H_2F_2]_n$ (ПВДФ). Поскольку основная цель разработки нашей модели - проведение быстрых оценок, для верификации разработанного модуля использовались сравнительно небольшие образцы систем из ~2000 атомов, а длительность тестовых расчетов составляла ~10 нс.

Для построения образцов полимерных материалов были созданы атомистические модели выбранных мономеров с помощью модуля визуализации MULTICOMP. Дальнейшая подготовка систем происходила с помощью инструментальных средств комплекса MULTICOMP в автоматическом режиме по схеме: «модель мономера» → «модель полимерной цепи» → «модель полимерной матрицы» (низкая плотность) → «сжатие системы» → «релаксация». Данная схема была нами неоднократно опробована при моделировании различных полимерных материалов, см.

например, работу [2]. Следует отметить, что автоматизация проведения расчетов является одним из преимуществ комплекса MULTICOMP. Она реализуется посредством построения пользователем гибких сценариев управляющих работой комплекса.

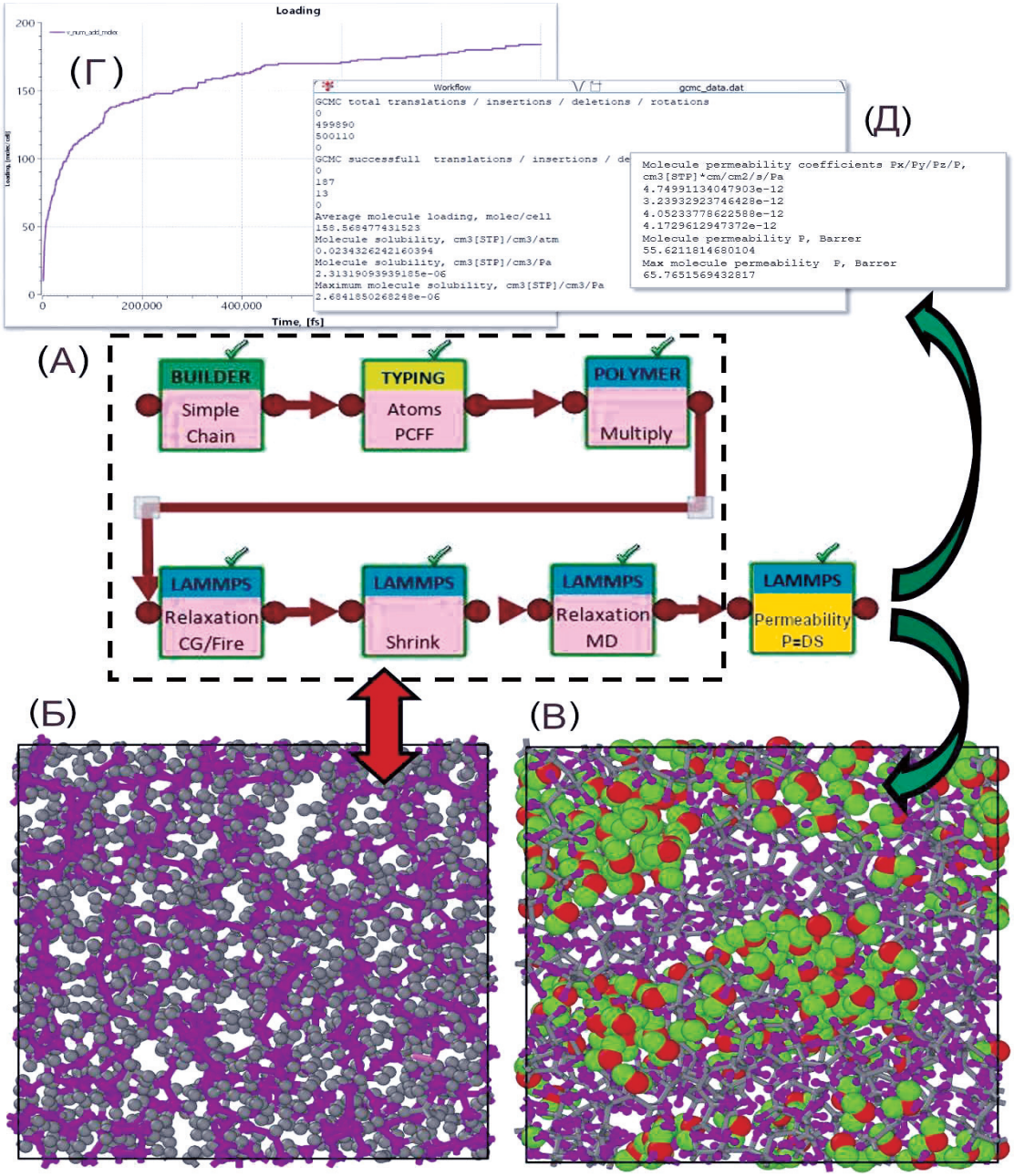


Рис. 2. Пример расчёта оценки проницаемости политетрафторэтилена (ПТФЭ) молекулами воды: (А) сценарий моделирования проницаемости в MULTICOMP. Примеры выходных данных модуля «Permeability»: (Б) сгенерированная матрица ПТФЭ, (В) визуализация конечной структуры ПТФЭ с растворенными молекулами воды, (Г) временная зависимость числа растворенных молекул, демонстрирующая выход структуры на равновесие в результате МД+ МКБКА расчёта, (Д) выходные файлы с информацией о коэффициентах: диффузии, растворимости и проницаемости. Атомы C и F окрашены серым и фиолетовым цветом, атомы H и O растворенных молекул воды - зеленым и красным, соответственно.

Пример использованного сценария для расчета барьерных свойств матрицы ПТФЭ показан на рис. 2. В нем задействовано 6 специализированных модулей. Построение линейных полимерных цепей было выполнено с помощью модуля «Polymer chain», после чего атомам были присвоены типы согласно их определению в ВСП PCFF с помощью модуля «Typing: Atoms PCFF». Степень полимеризации цепей была установлена $n = 33$ для ПЭ, ПЭТ, ПВДФ и 11 в случае ПЭТ.

Длина полимерных цепей была выбрана из расчета, что каждая цепь будет содержать порядка 200 атомов. Данное значение является достаточным для оценочных предсказаний транспортных свойств полимеров [22,23]. Построенные цепи первоначально имеют стержнеобразную конформацию. Затем с использованием модулей «Polymer/Composite Constructor» были сконструированы полимерные матрицы из 10 полимерных цепей. Для этого цепи помещаются в пустую кубическую ячейку большого объема чтобы избежать близких контактов атомов соседних молекул (система имеет низкую плотность). Далее для привидения подготовленных образцов в равновесное состояние были последовательно выполнены: оптимизация геометрии, сжатие образца до целевой плотности (модуль «Shrink», температура $T = 600\text{K}$) и релаксация при постоянной температуре и объеме (модули «Relaxation CG/Fire» и «Relaxation MD», температура $T = 300\text{K}$). В ходе процесса релаксации полимерные цепи приобретают конформации близкие к равновесным. Все подготовленные таким образом образцы материалов имеет аморфную структуру (которая вносит основной вклад в формирование транспортных свойств полимеров) с плотностью близкой к экспериментальной.

Для расчета барьерных свойств подготовленные системы передавались в модуль «Permeability», который выполняет оценку коэффициентов растворимости и диффузии для выбранной в настройках модуля молекулы. На первой стадии работы модуля моделируется процесс растворения в большом каноническом ансамбле (μVT) с периодическими граничными условиями. При этом матрица полимера остается подвижной, что позволяет более точно воспроизвести этот процесс. В более ранних работах аналогичное моделирование с использованием LAMMPS МКБКА проводилось только с жестко зафиксированными матрицами [24-29]. В ходе моделировании абсорбции чередуются циклы МКБКА и МД. Для вычисления электростатических взаимодействий между частицами используется солвер particle−particle and particle−mesh (PPPM) [30] с точностью 1×10^{-4} и радиусом отсечки 10 Å. Интегратор с методом Верле применяется для расчета скорости и положения частиц, шаг по времени был выбран равным 1 фс. Температура $T = 300\text{K}$ поддерживалась с помощью термостата Берендсена [31].

Через заданное число шагов по времени (один из параметров модуля, по умолчанию - 100 шагов) вызывается процедура МКБКА. Она реализует внедрение или удаление из системы молекул пенетранта. Как и в классическом методе Монте-Карло, одновременно можно задать: выполнение трансляции, и вращения молекул пенетранта. Перед каждым МКБКА-циклом происходит промежуточный МД-отжиг моделируемой системы. Данная процедура используется для получения состояния с равновесной концентрацией растворенных молекул при определенной температуре и химическом потенциале/давлении в среде, на основе которой рассчитывается коэффициент растворимости S . После прохождения заданного числа шагов МКБКА для равновесного числа растворенных молекул выполняется МД-расчет коэффициента диффузии D (вторая стадия работы модуля). После завершения всех этапов расчетной схемы модуля происходит обработка полученных результатов и вычисляется коэффициент проницаемости P по формуле (1).

4. Результаты расчетов

Пример оценки значений коэффициентов диффузии (*D*), растворимости (*S*) и проницаемости (*P*) в случае кислорода собраны в таблице 1. Коэффициенты проницаемости для всех выбранных полимеров в пределах порядка величины находятся в хорошем согласии с экспериментом. Следует отметить, что полученный разброс результатов для коэффициента диффузии и растворимости характерен и для экспериментальных работ, где эти характеристики также определяются с точностью до одного-двух порядков измеряемой величины [32].

Следует отметить, что для коэффициентов растворимости и проницаемости молекул кислорода модуль «Permeability» достаточно хорошо предсказывает общий тренд взаимного расположения выбранных полимеров: ПТФЭ > ПЭ ≈ ПВДФ > ПЭТ, что находится в качественном согласии с экспериментом. Также из таблицы 1 видно, что, хотя полученные значения *P* по порядку величины близки к экспериментальным результатам, коэффициенты растворимости, как правило, оказываются несколько ниже, а коэффициенты диффузии — выше экспериментальных значений. Это можно объяснить относительно небольшим размером построенных образцов систем

(~2000 атомов). При этом необходимо иметь в виду следующее. Во-первых, использование относительно малых образцов систем ограничивает число возможных траекторий движения отдельной молекулы, что приводит к увеличению коэффициента диффузии D. Во-вторых, для данного размера систем растворимость порядка $S \approx 10^{-7}$ см^3(СУ) / см^{3}Па соответствует всего 1-10 молекулам газа-пенетранта в ячейке моделирования (для рассмотренного интервала $P_{gas} \approx 10 - 1000$ атм, $T = 300$K).

Таблица 1. Коэффициенты диффузии D, растворимости S и проницаемости P полиэтилентерефталата (ПЭТ), полиэтилена (ПЭ), поливинилиденфторида (ПВДФ) и политетрафторэтилена (ПТФЭ) по воде и кислороду вычисленная использованием ВСП PCFF.

	$D \times 10^{-7}$ (см^2 / с)		$S \times 10^{-7}$ (см^3(СУ) / см^{3}Па)		P (Баррер)	
	Расчет	Эксперимент [32]	Расчет	Эксперимент [32]	Расчет	Эксперимент [32]
Кислород						
ПЭТ	0.18-0.23	0.03-0.8	0.2-0.5	6-10	0.01-0.02	0.013-5
ПЭ	12-27	0.2-12	0.32-0.8	2-4	0.12-0.31	0.04-2.3
ПВДФ	1.5-2.3	1.7	0.25-0.35	3.6	0.1-0.25	0.02-1.8
ПТФЭ	20-26	1.5	4.9-6.5	6-9	17-22	2.5-5
Вода						
ПЭ	10-20	0.6-1.3	8	0.58	15-20	74
ПТФЭ	20-25	1.5-12	40	-	55-65	8

При низких давлениях молекулы газа, как правило, занимают небольшое число наиболее доступных для растворения положений (адсорбционных сайтов) в полимерной матрице. При увеличении давления молекулы начинают проникать также и в небольшие поры с плохой связностью. При этом, из-за значительного сближения между ними начинает возникать сильное взаимодействие. В этом случае молекулы начинают формировать кластеры забивающие поры (как это показано на рис. 2В). Поэтому в расчетах при высоких давлениях $P_{gas} > 1000$ атм данный процесс может приводить к сильной нелинейности зависимости $S(P_{gas})$. Таким образом, полученное завышение коэффициента S по отношению к экспериментальным значениям можно объяснить возникновением сильного взаимодействия между внедренными молекулами газов в условиях ограниченного пространства. Тем не менее в пределах порядка расчетных величин также можно говорить и о согласии с экспериментом. Мы считаем, что более аккуратный выбор параметров моделирования (размер системы, ВСП и др.) и увеличение объема статистики позволит значительно увеличить точность и, соответственно, предсказательные возможности разработанной модели.

Таким образом, полученные тестовые результаты по проницаемости молекул кислорода и воды для четырёх выбранных материалов (ПЭТ, ПЭ, ПВДФ и ПТФЭ) находятся в качественном согласии с экспериментом. О количественном согласии пока можно говорить в пределах порядка величины. Тем не менее, это позволяет заключить, что использованная в модуле «Permeability» методика расчетов дает неплохие оценочные тренды растворимости малых молекул в полимерных материалах, что является важным при выборе барьерных покрытий с наилучшими свойствами.

5. Характеристики параллельности и масштабируемости

Важной практической характеристикой программных продуктов, ориентированных на реализацию высокопроизводительных расчетов, является их производительность и возможность масштабирования на массивно-параллельных вычислительных системах. Это особенно важно для модуля расчета проницаемости полимеров «Permeability», для которого существенна возможность расчета больших атомистических систем одновременно с достижением интервалов времени большой длительности, когда полученные результаты перестают зависеть как от раз-

мера системы, так и времени моделирования. Необходимое время определяется скоростью процесса релаксации в системе абсорбент-пенетрант и может составлять ~ 1 мкс даже для систем из 20 000 атомов при длине полимерной цепи ~ 8-160 мономеров (в зависимости от химического строения полимера).

Используемые в модуле «Permeability» чередующиеся циклы МКБКА и МД имеют разные вычислительные затраты и возможности масштабирования. Однако, для типичных условий расчета коэффициента проницаемости лимитирующей является стадия молекулярной динамики. Программный комплекс MULTICOMP может функционировать на высокопроизводительных архитектурах параллельных вычислений, включая гетерогенные вычислительные системы. При использовании современных межатомных потенциалов, разработанных для моделирования полимеров, таких как ВСП класса II, результирующая нормированная оценка производительности для моделирования молекулярной динамики составляет порядка 30 000 (шаг × атом) / (ядро ЦП × секунда) для процессоров Intel семейства Xeon второго поколения и новее (тестирование проводилось на модели Intel Xeon Gold 6226R). При такой производительности моделирование структуры, состоящей из 100 000 атомов, за 1 миллион шагов (что при шаге интегрирования уравнений движения в 1 фс соответствует 1 нс моделирования) на системе со 100 ядрами ЦП занимает около 10 часов.

В случае использования вычислительных ускорителей на основе GPGPU (графических карт) оценка производительности по молекулярной динамике составляет примерно 200 (шаг × атом) / (ядро GPU × секунда), что для ускорителя GPGPU с числом ядер > 2000 позволяет достичь производительности 100 000 атомов на 1 нс за 100 часов (тестирование проводилось на ускорителях Nvidia GeForce RTX 3090). Следует отметить, что лимитирующей стадией в данном случае является расчет парного межмолекулярного Ван-дер-Ваальсова взаимодействия, а не расчет дальнодействующего многочастичного кулоновского взаимодействия, что обеспечивает практически линейную масштабируемость молекулярной динамики, на основе которой можно пропорционально увеличить размер рассматриваемой системы за счет увеличения количества вычислительных ресурсов. Линейное масштабирование проверялось при использовании многопроцессорных CPU систем с общим количеством физических ядер до 1024 штук и применением технологии MPI для распараллеливания процессов (исследование показало, что применение гибридной параллелизации на основе комбинации многопоточности и многопроцессности с использованием технологий OpenMP + MPI не дает существенного увеличения производительности вычислений для моделирования полимерных систем в программе LAMMPS по сравнению с использованием только MPI). Это позволяет существенно повысить производительность вычислений коэффициента диффузии и коэффициентов растворимости и проводить предсказательные расчеты проницаемости новых полимерных материалов на масштабе времени порядка одного дня.

6. Выводы

В работе представлено описание модуля «Permeability» расширения функционала программного комплекса MULTICOMP для расчета свойств проницаемости полимерных материалов на основе информации о их химическом строении. Получены значения коэффициентов растворимости для полиэтилена, полиэтилентерефталата, политетрафторэтилена и поливинилиденфторида по кислороду. Так же проведено исследование масштабируемости при расчете на вычислительном кластере ООО «Кинтех Лаб» и НИЦ «Курчатовский институт». Выполненные тестовые расчеты показали, что комплекс MULTICOMP, дополненный модулем «Permeability», позволяет получать результаты относительной проницаемости полимерных в хорошем согласии с экспериментом. Точность расчетов можно повысить за счет увеличения размера моделируемых систем, построения набора статистически независимых образцов и увеличения длительности расчетов. Также разработанный модуль может использоваться для оценки проницаемости гибридных органо-неорганических материалов.

Таким образом, расширенная функциональная возможность комплекса MULTICOMP позволяет использовать его для дизайна покрытий на основе полимерных материалов в задачах, где необходимо оптимизировать газопроницаемость барьерных покрытий.

Работа была выполнена с использованием оборудования центра коллективного пользования «Комплекс моделирования и обработки данных исследовательских установок мега-класса» НИЦ

«Курчатовский институт», http://ckp.nrcki.ru/. П.В. Комаров проводил работы по разработке расчетной схемы проницаемости полимерных материалов в рамках Государственного задания №075-03-2023-642 Министерства науки и высшего образования Российской Федерации.

Литература

1. Church N. Developments in modified-atmosphere packaging and related technologies. // Trends Food Sci. Technol. 1994. Vol. 5. P. 345–352. DOI: 10.1016/0924-2244(94)90211-9.
2. Lau O. W., Wong S. K. Contamination in food from packaging material // J. Chromatogr. A. 2000. Vol. 882. P. 255–270. DOI: 10.1016/S0021-9673(00)00356-3.
3. Han J.-W., Ruiz-Garcia L., Qian J.-P., Yang X.-T. Food Packaging: A Comprehensive Review and Future Trends // Compr. Rev. Food Sci. Food Saf. 2018. Vol. 17. P. 860–877. DOI: 10.1111/1541-4337.12343.
4. Bekhta P., Lyutyy P., Hiziroglu S., Ortynska G. Properties of composite panels made from tetra-Pak and polyethylene waste material // J. Polym. Environ. 2016. Vol. 24. P. 159–165. DOI: 10.1007/s10924-016-0758-7.
5. Kopacic S., Walzl A., Zankel A., Leitner E., Bauer W. Alginate and chitosan as a functional barrier for paper-based packaging materials // Coat. World. 2018. Vol. 8. P. 235. DOI: 10.3390/coatings8070235.
6. Chung D., Papadakis S. E., Yam K. L. Simple models for evaluating effects of small leaks on the gas barrier properties of food packages // Packag. Technol. Sci. 2003/ Vol. 16. P. 77–86. DOI: 10.1002/pts.616.
7. Cheng H., Chen L., McClements M. J., Yang T. et al. Starch-based biodegradable packaging materials: A review of their preparation, characterization and diverse applications in the food industry // Trends Food Sci. Technol. 2021. Vol. 114. P. 70–82. DOI: 10.1016/j.tifs.2021.05.017.
8. Shaikh S., Yaqoob M., Aggarwal P. An overview of biodegradable packaging in food industry // Curr Res Food Sci. 2021. Vol. 4. P. 503–520. DOI: 10.1016/j.crfs.2021.07.005.
9. Wang Q., Chen W., Zhu W. et al. A review of multilayer and composite films and coatings for active biodegradable packaging // NPJ Sci Food. 2022. Vol. 6. P. 18. DOI: 10.1038/s41538-022-00132-8.
10. Gupta P., Toksha B., Rahaman M. A Review on Biodegradable Packaging Films from Vegetative and Food Waste // Chem. Rec. 2022. Vol. 22. P. e202100326. DOI: 10.1002/tcr.202100326.
11. Akhukov M. A., Chorkov V. A., Gavrilov A. A. et al. MULTICOMP package for multilevel simulation of polymer nanocomposites // Comput. Mater. Sci. 2023. Vol. 216. P. 111832. DOI: 10.1016/j.commatsci.2022.111832.
12. Tylianakis E., Froudakis G. E. Grand canonical Monte Carlo method for gas adsorption and separation // J. Comput. Theor. Nanosci. 2009. Vol. 6. P. 335–348. DOI: 10.1166/jctn.2009.1040.
13. Khosravanian A., Dehghani M., Pazirofteh M. et al. Grand canonical Monte Carlo and molecular dynamics simulations of the structural properties, diffusion and adsorption of hydrogen molecules through poly(benzimidazoles)/nanoparticle oxides composites // Int. J. Hydrogen Energy. 2018. Vol. 43. P. 2803–2816. DOI: 10.1016/j.ijhydene.2017.12.122.
14. Harami H. R., Amirkhani F., Khadem S. A. et al. Mass transfer through PDMS/zeolite 4A MMMs for hydrogen separation: Molecular dynamics and grand canonical Monte Carlo simulations // Int. Commun. Heat Mass Transfer. 2019. Vol. 108. P. 104259. DOI: 10.1016/j.icheatmasstransfer.2019.05.005.
15. Volgin I. V., Andreeva M. V., Larin S. V. et al. Transport Properties of Thermoplastic R-BAPB Polyimide: Molecular Dynamics Simulations and Experiment // Polymers 2019. Vol. 11, P. 1775. DOI:10.3390/polym11111775.

16. Wu S., Yi J., Zhang L., Zhang L. The study on the gas permeabilities of the ethylene/1-hexene copolymer by molecular dynamics simulation // International Journal of Modern Physics B Vol. 22, Nos. 31 & 32 (2008) 5859–5864 10.1142/S0217979208051285.

17. Sun H., Mumby S. J., Maple J. R., Hagler A. T. An ab initio CFF93 all-atom force field for polycarbonates // J. Am. Chem. Soc. 1994. Vol. 116. P. 2978–2987. DOI: 10.1021/ja00086a030.

18. Mayo S. L., Olafson B. D., Goddard W. A. DREIDING: a generic force field for molecular simulations // J. Phys. Chem. 1990. Vol. 94. P. 8897–8909. DOI: 10.1021/j100389a010.

19. Sun H. COMPASS: An ab initio force-field optimized for condensed-phase Applications Overview with details on alkane and benzene compounds // J. Phys. Chem. B. 1998. Vol. 102. P. 7338–7364. DOI: 10.1021/jp980939v.

20. Plimpton S. Fast parallel algorithms for short-range molecular dynamics // J. Comput. Phys. 1995. Vol. 117. P. 1–19. DOI: 10.1006/jcph.1995.1039.

21. Komarov P. V., Mikhailov I. V., Chiu Y.-T., Chen S.-M., Khalatur P. G. Molecular dynamics study of interface structure in composites comprising surface-modified SiO2 nanoparticles and a Polyimide Matrix // Macromol. Theory Simul. 2013. Vol. 22. P. 187–197. DOI: 10.1002/mats.201200063.

22. Yampolskii Y., Shishatskii S., Alentiev A., Loza K. Group contribution method for transport property predictions of glassy polymers: focus on polyimides and polynorbornenes // J. Memb. Sci. 1998. Vol. 149. P. 203–220. DOI: 10.1016/S0376-7388(98)00152-5.

23. Ryzhikh V., Tsarev D., Alentiev A., Yampolskii Y. A novel method for predictions of the gas permeation parameters of polymers on the basis of their chemical structure // J. Memb. Sci. 2015. Vol. 487. P. 189–198. DOI: 10.1016/j.memsci.2015.03.055.

24. Cozmuta I., Blanco M., Goddard W. A. 3rd. Gas sorption and barrier properties of polymeric membranes from molecular dynamics and Monte Carlo simulations // J. Phys. Chem. B. 2007. Vol. 111. P. 3151–3166. DOI: 10.1021/jp062942h.

25. Sava D. F., Rodriguez M. A., Chapman K. W. et al. Capture of volatile iodine, a gaseous fission product, by zeolitic imidazolate framework-8 // J. Am. Chem. Soc. 2011. Vol. 133. P. 12398–12401. DOI: 10.1021/ja204757x.

26. Sava D. F. Chapman K. W., Rodriguez M. A. et al. Competitive I2sorption by cu-BTC from humid gas streams // Chem. Mater. 2013. Vol. 25. P. 2591–2596. DOI: 10.1021/cm401762g.

27. Zhang J., Clennell M. B., Dewhurst D. N. Liu K. Combined Monte Carlo and molecular dynamics simulation of methane adsorption on dry and moist coal // Fuel. 2014. Vol. 122. P. 186–197. DOI: 10.1016/j.fuel.2014.01.006.

28. Zhou W., Wang H., Zhang Z., Chen H., Liu X. Molecular simulation of CO/CH/HO competitive adsorption and diffusion in brown coal // RSC Adv. 2019. Vol. 9. P. 3004–3011. DOI: 10.1039/C8RA10243K.

29. Li J., Wang Y., Chen Z., Rahman S. S. Insights into the Molecular Competitive Adsorption Mechanism of CH/CO in a Kerogen Matrix in the Presence of Moisture, Salinity, and Ethane // Langmuir. 2021. Vol. 37. P. 12732–12745. DOI: 10.1021/acs.langmuir.1c02274.

30. Hockney R. W., Eastwood J. W. Computer simulation using particles. Hilger, Bristol, 1988. 540 p.

31. Berendsen H. J. C., Postma J. P. M., van Gunsteren W. F., DiNola A. R., Haak J. Molecular dynamics with coupling to an external bath // J. Chem. Phys. 1984. Vol. 81. P. 3684. DOI: 10.1063/1.448118.

32. The NIMS Materials Database (MatNavi) https://mits.nims.go.jp/en/ (дата обращения: 12.04.2023).

Совместная модель ПЛАВ-NEMO-SI3

Р.Ю. Фадеев[1,2,5], Ю.Д. Реснянский[2,5], Б.С. Струков[2], А.А. Зеленько[2], И.Н. Смирнов[3,5], К.П. Беляев[4,5], А.А. Кулешов[5]

[1]Институт вычислительной математики им. Г.И. Марчука РАН,
[2]ФГБУ «Гидрометцентр России»,
[3]Московский государственный университет имени М.В. Ломоносова,
[4]Институт океанологии им. П.П. Ширшова РАН,
[5]Институт прикладной математики им. М.В. Келдыша РАН.

ПЛАВ-NEMO-SI3 является совместной моделью атмосферы, океана и морского льда, которая разрабатывается для целей численного среднесрочного прогноза погоды. Для объединения ПЛАВ и NEMO в рамках единой модели применяется программный комплекс OASIS3-MCT. Модель SI3 функционирует в рамках NEMO. В работе обсуждаются архитектура и особенности реализации совместной модели. Особое внимание в работе уделяется первым результатам по воспроизведению атмосферной циркуляции совместной моделью на компьютерах с массивно-параллельной архитектурой.

Ключевые слова: совместная модель, численный прогноз погоды, параллельная масштабируемость

1. Введение

Разработка совместных моделей динамики океана и атмосферы началась в 90-х годах прошлого века. Наверное, исторически первой совместной моделью, получившей международное признание, была модель Н.Н. Моисеева и В.В. Александрова (ВЦ РАН) 1970 года, предсказавшая так называемую «ядерную зиму» [1]. Одной из первых геофизических глобальных моделей была модель COLA (Center Ocean-Land-Atmosphere, Maryland, USA) [2]. Эта модель достаточно успешно предсказала появление феномена El Niño в 1997 году. Далее, с развитием вычислительной техники и методов параллельного программирования совместные модели стали разрабатываться и появляться в различных научных центрах и университетах, например, модель GFDL's CM2, разработанная в Университете Аризоны в 2006 году [3], Модель Европейского центра среднесрочных прогнозов погоды ECMWF (European Centre for Medium-Range Weather Forecasts) [4].

Сейчас программные комплексы на основе совместных моделей широко применяются для оценки климатических изменений и изучения климата Земли в прошлом. Для целей долгосрочного прогноза погоды в практически всех ведущих метеорологических центрах ВМО применяются системы на основе совместных моделей: GloSea [5] в Английской метеослужбе, SYSTEM5 [6] во Франции, CFSv2 [7] в США, CanSIPS [8] в метеорологическом центре Канады и т.д. Характерное разрешение таких моделей составляет от 25 до 100 км.

В последние годы в ведущих метеорологических центрах ВМО изучается возможность применения совместных моделей для целей среднесрочного прогноза погоды до 10 дней. Например, в работе [9] показано статистически значимое улучшение качества среднесрочного прогноза погоды по объединенным в единый программный комплекс оперативным прогностическим моделям атмосферы GEM [10] и океана GIOPS (основана на NEMO) в сравнении с текущей моделью оперативного прогноза метеорологического центра Канады. В [11] обсуждается актуальные результаты по точности предсказания интенсивности и траекторий тропических циклонов на основе совместной модели атмосферы и океана Европейского центра среднесрочных прогнозов погоды. Отметим, что, например, для прогноза траектории движения и интенсивности тропических циклонов в Атлантическом океане и Восточной части Тихого океана в США используется совместная модель COAMPS-TC [12]. Для

изучения и прогноза тайфунов в Юго-Восточной Азии (Maritime Continent) разработана совместная модель [13], состоящая из глобальной моделли атмосферы MetUM (в версии на основе EndGame [14]) Английской метеослужбы и модели Мирового океана NEMO, которые объединены в единый программный комплекс с помощью блока OASIS. Пространственное разрешение этой модели составляет 4.5 км.

В настоящее время в России существует только одна совместная гидродинамическая модель атмосферы и океана INMCM [15] (разработана в ИВМ РАН), которая участвует в международном проекте по сравнению климатических моделей (CMIP) проводимой Межправительственной группой экспертов по изменению климата (МГЭИК). Следует отметить, что в Главной геофизической обсерватории им. А.И. Воейкова (ГГО) ведутся разработки совместной модели атмосферы и океана для долгосрочного прогноза погоды [16]. В качестве атмосферного компонента здесь используется разработанная в ГГО спектральная модель атмосферы [17], которая применяется в настоящее время в Гидрометцентре России в качестве одного из компонентов для вероятностного долгосрочного прогноза аномалий погоды. В качестве модели океана здесь используется разработанная в ИВМ РАН модель, та же, что и в INMCM. Обмен данными между моделями атмосферы и океана осуществляется напрямую без использования системы для совместного моделирования. Еще одной разработкой, нацеленной на практическое применение в области долгосрочного прогноза аномалий погоды, является совместная модель [19], объединяющая модель атмосферы ПЛАВ (ИВМ РАН и Гидрометцентр России) [18], модель Мирового океана ИВМИО (ИВМ РАН и ИО РАН) [25] и заимствованную модель динамики морского льда CICE [26]. Объединение в совместную модель здесь осуществлено с применением оригинальной системы для совместного моделирования CMF [27]. Однако, разработка этой совместной модели, как и системы усвоения данных океанографических наблюдений (измерений) для нее, к настоящему моменту не завершены.

В данной работе рассматривается новая совместная модель атмосферы, океана и морского льда ПЛАВ-NEMO-SI3.

2. Архитектура программного комплекса ПЛАВ-NEMO-SI3

На рис. 1 представлена принципиальная схема работы совместной модели ПЛАВ-NEMO-SI3. Основными компонентами этой модели являются модель общей циркуляции атмосферы ПЛАВ (на рис. 1 - аббревиатура на английском языке - SLAV), модель Мирового океана NEMO (Nucleus for European Modelling of the Ocean) [3] и модель морского льда SI3. Модели объединены системой совместного моделирования OASIS3-MCT, а программное обеспечение XIOS, необходимое для функционирования NEMO, выполняет работу с файловой системой. Блок описания процессов на поверхности суши и в почве является частью модели атмосферы ПЛАВ (Land surface model на рис. 1). Блоки CLIRAD [22, 23] и RRTMG-LW [24] являются неотъемлемыми частями ПЛАВ и отвечают за расчет распространения коротко- и длинноволновой радиации в атмосфере Земли с учетом ее поглощения, переизлучения и отражения как поверхностью, так и облачностью и атмосферными аэрозолями. Библиотеки NetCDF и HDF5 являются заимствованным и свободно распространяемым программным обеспечением, к функциям которого относится работа с файловой системой в массивно-параллельном режиме. Следует отметить, что в рамках OASIS используются библиотеки PSMILE и SCRIP, реализующие математические алгоритмы поиска ближайших соседей к точке на поверхности сферы с известными координатами и интерполяцию данных с одной расчетной сетки на другую с учетом маски.

Основные уравнения NEMO и SI3 дискретизируются на одной и той же триполярной сетке типа ORCA. Реализовано две версии модели ПЛАВ-NEMO-SI3, отличающихся пространственным разрешением моделей NEMO и SI3: в версии модели с высоким пространственным разрешением применяется сетка ORCA025 с шагом сетки 0.25°, в версии с

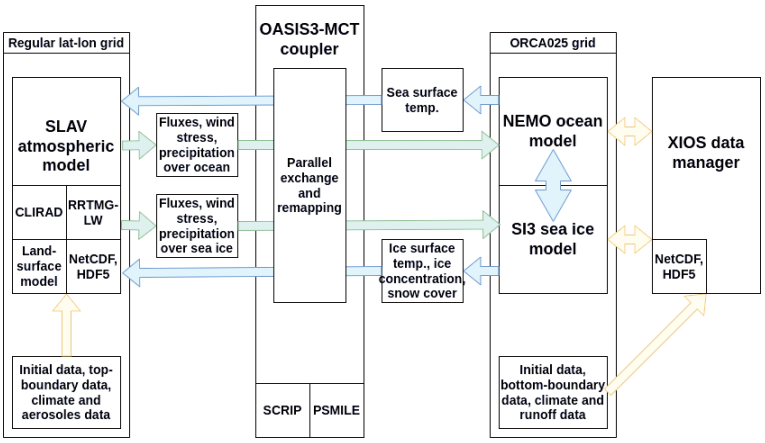


Рис. 1. Принципиальная схема программного комплекса совместной модели ПЛАВ-NEMO-SI3.

грубым разрешением - ORCA1 с шагом сетки 1°. Число уровней по вертикали в океане - 75. Уравнения гидротермодинамики атмосферы, записанные в гидростатическом приближении, дискретизируются в ПЛАВ на регулярной широтно-долготной сетке с шагом 0.9° по долготе, 0.72° по широте и 96 уровнями по вертикали. Отметим, что большое число уровней по вертикали в ПЛАВ позволяет детально описывать пограничный слой атмосферы (нижняя тропосфера) и динамику стратосферы - проводника крупномасштабных волн Россби. Использование в ПЛАВ и NEMO/SI3 отличающихся сеток приводит к необходимости использования системы для совместного моделирования (т.н. каплер), к задачам которого относится:

- синхронизация времени моделей-компонентов совместной модели;
- организация параллельного обмена данными между моделями-компонентами;
- интерполяция пересылаемых между моделями-компонентами данных в случае, если сетки моделей отправителя и получателя отличаются друг от друга.

В рамках совместной модели ПЛАВ-NEMO-SI3 для перечисленных целей было выбрано программное обеспечение OASIS3-MCT [28], разрабатываемое в CERFACS (Тулуза, Франция) и Национальном центре научных исследований (Centre National de la Recherche Scientifique, Париж, Франция). Выбор данной системы для совместного моделирования обусловлен тем, что модели NEMO и SI3 уже имеют готовые интерфейсы для работы с OASIS. Конфигурация моделей NEMO и SI3 в совместной модели соответствовала версиям, применяемым в оперативной технологии Гидрометцентра России, и дополненная блоками, допускающими применение в рамках системы для совместного моделирования OASIS. Для модели NEMO/SI3 были подготовлены контрольные точки, содержащие согласованную информацию о состоянии вод Мирового океана и морского льда.

Совместная модель ПЛАВ-NEMO-SI3 собирается в виде двух исполняемых файлов. Один соответствует ПЛАВ, второй - NEMO и SI3. Оба файла исполняются на вычислительной системе в параллельном режиме, что позволяет настроить для каждого из них собственные параметры окружения. Такой подход обоснован применением технологии OpenMP в модели ПЛАВ.

Параллельная конфигурация совместной модели ПЛАВ-NEMO-SI3 в версии с высоким пространственным разрешением на 1008 процессорных ядер следующая: 768 вычислительных ядра отводится моделям NEMO и SI3, в то время как для работы ПЛАВ достаточно 240 процессорных ядер (60 MPI процессов дополняются 4 OpenMP нитями). Время расчета одного модельного дня в такой конфигурации составляет величину чуть менее трех минут

на вычислительной системе Cray XC-40, установленной в Главном вычислительном центре (ГВЦ) Росгидромета. Отметим, что указанная производительность соответствует версии совместной модели с достаточно большим объемом выдачи, необходимым для ее отладки, настройки и проверки. Исследование оптимальных характеристик совместной модели, соответствующих максимальной скорости счета в зависимости от задействованного объема вычислительных ресурсов, способа их распределения между моделями и периода времени между обменами данными, является предметом дальнейшей работы.

3. Особенности реализации совместной модели

Температура поверхности океана и морского льда задаются в качестве граничных условий для модели глобальной атмосферы на нижней границе расчетной области. В исследовательских задачах, где предметом изучения являются атмосферные процессы и явления, произошедшие в прошлом, температура поверхности океана и морского льда (и ее эволюция) задается по данным наблюдений и реанализа. В задаче оперативного прогнозирования, обычно, применяется методика экстраполяции аномалий температуры поверхности океана по отношению к осредненным за много лет значениям. В рамках совместной модели поверхностные характеристики океана и морского льда необходимые атмосфере являются расчетными. В то же время, модель атмосферы является поставщиком данных для граничных условий для моделей океана и морского льда. На начальном этапе реализации совместной модели ПЛАВ-NEMO-SI3 после ряда экспериментов были определены основные поля, передаваемые между компонентами. В частности, из ПЛАВ в NEMO и SI3 пересылаются следующие поля:

1. зональный и меридиональный компоненты касательного напряжения силы трения ветра на поверхности;
2. потоки (Вт/м^2) коротковолнового излучения на поверхности Земли (с учетом альбедо поверхности) - отдельно для моделей океана и морского льда;
3. суммарный поток теплового излучения поверхности (с учетом переизлучения и отражения облаками, аэрозолями и пр.), потоков скрытого и явного тепла - отдельно для моделей океана и морского льда;
4. накопленное за период между обменами количество влаги на поверхности, обусловленное выпадением осадков. В модели океана и морского льда влага передается с учетом ее фазового состояния: твердая (снег, град) и жидкая (капли).

Расчет перечисленных полей выполняется в блоках параметризованного описания процессов подсеточного масштаба модели ПЛАВ. В частности, зональный и меридиональный компоненты касательного напряжения силы ветра на поверхности, потоки скрытого и явного тепла на поверхности рассчитываются в блоке описания пограничного слоя атмосферы, основанного на работе [20]. Крупномасштабные осадки в ПЛАВ вычисляются на основе подхода, предложенного в работе [21]. Для расчета потоков солнечного и теплового излучения на поверхности Земли применяются заимствованные пакеты CLIRAD и RRTMG-LW.

Из NEMO и SI3 в ПЛАВ, в свою очередь, пересылаются следующие поля:

1. температура поверхности океана;
2. доля расчетной ячейки, занятая морским льдом;
3. температура поверхности морского льда (снежного покрова на морском льду, при наличии);
4. температура морского льда на глубине, известной в ПЛАВ;

5. толщина морского льда;

6. толщина снежного покрова на морском льду.

Реализация совместной модели на основе OASIS3-MCT позволяет относительно легко изменять характер передаваемых между моделями полей, периодичность обмена и метод интерполяции. Для этих целей в системе совместного моделирования предусмотрен конфигурационный файл. Однако, для использования данного функционала и разработки совместной модели в целом потребовалось провести ряд работ по модификации программного кода глобальной модели атмосферы ПЛАВ. Так, например, в ПЛАВ был реализован управляющий файл, в рамках которого выполняются следующие процедуры:

- регистрация модели ПЛАВ в системе совместного моделирования OASIS;
- передача необходимой OASIS информации о характеристиках модели ПЛАВ: шаг по времени, конфигурации расчетной сетки, параллельная декомпозиция и пр.
- передача OASIS указателей на массивы принимаемых и отправляемых полей другим моделям.
- основной цикл ПЛАВ по времени, выполнение одного шага по времени, обработка передаваемых и получаемых моделью атмосферы данных.

В рамках самой модели ПЛАВ был реализован ряд методов для расчета передаваемых другим моделям данных о состоянии нижней атмосферы и потоках на поверхности. Для этого в ПЛАВ были созданы производные типы данных (модель ПЛАВ реализована на языке Фортран), содержащие информацию о внешних по отношению к ПЛАВ моделях, количестве и характеристиках получаемых и передаваемым этим моделям данных, массивы данных и, наконец, флаги, указывающие на наличие обновленных граничных условиях. На начальном этапе разработки была выбрана стратегия минимального промежутка времени между обменами, информацией между моделями атмосферы, океана и морским льдом. Это время составило 1440 с., что соответствует одному шагу по времени в модели ПЛАВ и двум шагам в моделях NEMO и SI3. Частый обмен данными между моделями позволяет избежать проблемы задания/интерпретации граничных условий в моменты времени между обменами, сосредоточившись на технологической отладке совместной модели и анализе ошибок воспроизведения физических процессов в атмосфере и океане.

4. Предварительные результаты моделирования

Совместная модель ПЛАВ-NEMO-SI3 была проверена на серии численных экспериментов с максимальной продолжительностью интегрирования не превышающей 30 дней. Стартовая дата во всех расчетах соответствовала 00 ВСВ 11 ноября 2021 года. Начальные состояния океана и морского льда являются согласованными друг с другом, но не согласованными с состоянием атмосферы, поскольку подготовка начальных данных о состоянии указанных сред осуществляется по разным и независимым друг от друга технологиям. Рассогласование состояний нижней тропосферы и поверхности океана, обычно, приводит к так называемому эффекту spin-up, выражающемуся в возникновении в совместной системе процессов приспособления. Отметим, что уменьшение ошибок моделирования, обусловленных эффектами приспособления, является предметом дальнейшей работы.

На рис. 2 слева представлена температура поверхности после 14 дней интегрирования совместной модели (соответствует 00 ВСВ 25 ноября 2021 г.) Обмен данными между моделями-компонентами в этом эксперименте осуществлялся на основе консервативной интерполяции. На рис. 2 справа иллюстрируется разница этого решения по отношению к полю температуры поверхности, полученной с помощью той же версии модели, но с билинейной интерполяцией передаваемых полей. Можно видеть, что за 14 дней интегрирования

численные решения успевают довольно сильно разойтись, вследствие нелинейности моделируемых процессов. Наиболее значимые отличия характерны для поверхности морского льда и суши, обладающими меньшей, по сравнению с верхним перемешенным слоем океана, теплоемкостью.

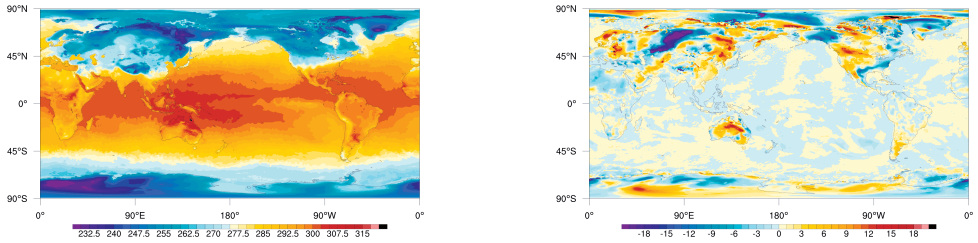


Рис. 2. Температура поверхности после 14 дней интерирования ПЛАВ-NEMO-SI3 с консервативной интерполяцией передаваемых между моделями-компонентами полей (слева) и отклонение этого решения от полученного с помощью той же версии модели, но с билинейной интерполяцией передаваемых полей.

Результаты воспроизведения общей циркуляции атмосферы и океана на 10 дней были сравнены с начальным состоянием атмосферы, океана и морского льда, соответствующими 00 ВСВ 21 ноября 2021 г. На рис. 3 иллюстрируется разность между температурой поверхности после 10 дней интегрирования совместной модели ПЛАВ-NEMO-SI3 и соответствующих 00 ВСВ 21 ноября 2021 г начальных данных. Можно видеть, что наиболее значимые отличия достигаются на суше и на поверхности морского льда. В то же время, ошибка около двух градусов для океана является значимой и, по всей видимости, является следствием нескольких факторов: эффектами приспособления упомянутых выше, необходимостью настройки моделей (например, с целью увеличения балла облачности верхнего яруса в западной части Тихого океана), проверки технологии каплинга и более тщательной диагностики обратных связей.

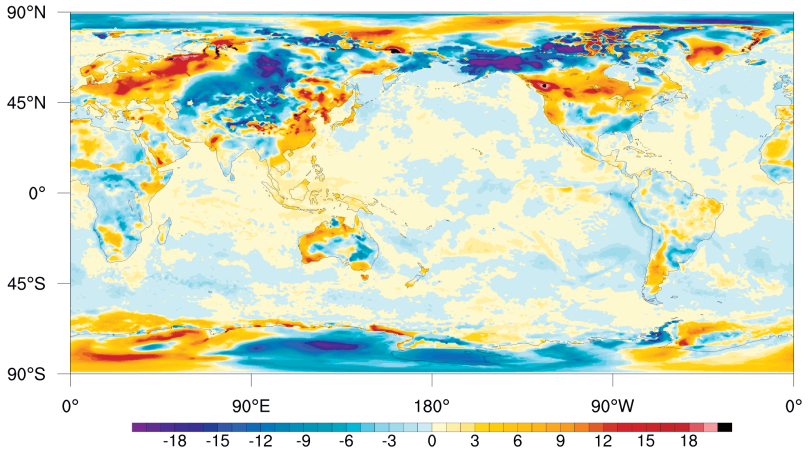


Рис. 3. Отклонение прогнозируемой по ПЛАВ-NEMO-SI3 поля приземной температуры от анализа Гидрометцентра России на тот же момент времени. Заблаговременность прогноза - 14 суток.

5. Заключение

Совместные модели, объединяющие модели атмосферы, океана и морского льда, являются одним из основных инструментов прогноза аномалий погоды на субсезонном (до одного месяца) и долгосрочном (от одного до четырех месяцев) масштабах времени. В работе

рассматривается совместная модель, которая после доработки и тщательного тестирования может быть применена в рамках технологии среднесрочного прогнозирования.

Совместная модель ПЛАВ-NEMO-SI3 была отлажена и проверена на серии численных экспериментов с продолжительною интегрирования до 30 дней. Результаты экспериментов показали устойчивую работу совместной модели, отсутствие систематических крупномасштабных смещений основных метеорологических полей в атмосфере по сравнению с реанализом ERA5, адекватную эволюцию характеристик морского льда и поверхности океана. Результаты предварительного сравнения результатов моделирования с заблаговременностью до 14 суток на основе совместной модели ПЛАВ-NEMO-SI3 с анализом Гидрометцентра России показали значимые отличия в поле приземной температуры. Такое отличие можно объяснить рассогласованием начальных состояний атмосферы и океана, а также большим сроком интегрирования модели. Отличия результатов моделирования от анализа под водной поверхностью не столь существенны и достигаются, главным образом, в зоне интенсивного перемешивания, где дуют сильные ветра, приводящие к образованию мезомасштабных вихрей на поверхности океана. Более детальный анализ качества воспроизведения атмосферной и океанической циркуляции в сравнении с данными наблюдений является предметом дальнейшей работы.

Работа выполнена при поддержке Российского научного фонда (РНФ), проект No 22-11-00053.

Литература

1. Моисеев Н.Н., Александров В.В., Тарко А.М. Опыт системного анализа и эксперименты с моделями // Человек и биосфера, М.: Наука, 1985.

2. Chen D., Zebiak S.E., Busalacchi A.J., Cane M.A. An improved procedure for El Niño forecasting: Implications for predictability // Science. 1995. Vol. 269. P. 1699–702.

3. Delworth T.L., Broccoli A.J., Rosati A. et al. GFDL's CM2 global coupled climate models—Part 1: Formulation and simulation characteristics // J. Climate. 2006. Vol. 19 No. 5. P. 643–674.

4. Bechtold P. et al. Advances in simulating atmospheric variability with the ECMWF model: From synoptic to decadal time-scales // Q. J. Roy. Meteorol. Soc. 2008. Vol. 134. P. 1337–1351.

5. MacLachlan C., Arribas A., Peterson K.A., et al. : Global seasonal forecast system version 5 (GloSea5): a high-resolution seasonal forecast system, Quart. J. Roy. Meteor. Soc. 2015. Vol. 141. P. 1072-1084. DOI: 10.1002/qj.2396

6. Voldoire A., Sanchez-Gomez E., Salas y Melia D., et al. The CNRM-CM5.1 global climate model: description and basic evaluation // Climate dynamics. 2013. Vol. 40. P. 2091–2121. DOI: 10.1007/s00382-011-1259-y

7. Saha S., Moorthi S., Wu X., et al. The NCEP climate forecast system version 2 // Journal of Climate. 2014. Vol. 27. P. 2185–2208. DOI: 10.1175/JCLI-D-12-00823.1

8. von Salzen K., Scinocca J.F., McFarlane N.A., Lazare M., et al. The Canadian fourth generation atmospheric global climate model (CanAM4). Part I: physical processes. // Atmosphere–Ocean. 2013. Vol. 51. P. 104–125. DOI: 10.1080/07055900.2012.755610

9. Smith G.C., Bélanger J., Roy F., et al. Impact of Coupling with an Ice–Ocean Model on Global Medium-Range NWP Forecast Skill // Monthly Weather Review. 2018. Vol. 146. No. 4. P. 1157-1180.

10. Global Deterministic Prediction System (GDPS): Update from version 4.0.1 to version 5.0.0. // Canadian Meteorological Centre Tech. 2015. Note 59.
URL: http://collaboration.cmc.ec.gc.ca/cmc/cmoi/product_guide/docs/tech_notes/technote_gdps-500_20151215_e.pdf

11. Magnusson, L., Bidlot, J.-R., Bonavita, M., et al. ECMWF activities for improved hurricane forecasts // Bulletin of the American Meteorological Society. 2019. Vol. 100. No.3. 445-458.

12. Doyle, J.D., R.M. Hodur, S. Chen, Y. Jin, J.R. Moskaitis, S. Wang, E.A. Hendricks, H. Jin, and T.A. Smith. // Tropical cyclone prediction using COAMPS-TC // Oceanography. 2014. Vol. 27. No. 3. P. 104–115. DOI: 10.5670/oceanog.2014.72

13. Thompson, B., Sanchez, C., Heng, B. C. P., Kumar, R., Liu, J., Huang, X.-Y., and Tkalich, P. Development of a MetUM (v 11.1) and NEMO (v 3.6) coupled operational forecast model for the Maritime Continent – Part 1: Evaluation of ocean forecasts // Geosci. Model Dev. 2021. Vol. 14. P. 1081–1100. DOI: 10.5194/gmd-14-1081-2021

14. Wood N., Staniforth A., White A., et al. An inherently mass-conserving semi-implicit semi-Lagrangian discretization of the deep-atmosphere global non-hydrostatic equations // Q. J. R. Meteorol. Soc. 2014. Vol. 140. P. 1505-1520. DOI: 10.1002/qj.2235

15. Volodin E.M., Mortikov E.V., Kostrykin S.V. et al. Simulation of the present day climate with the climate model INMCM5 // Clim. Dyn. 2017. Vol. 49. P. 3715-3734.

16. Мирвис В.М., Мелешко В.П., Львова Т.Ю. и др. Прогностические эксперименты на основе совместной модели океан-атмосфера ГГО // Труды Главной геофизической обсерватории им. А.И. Воейкова. 2016. № 583. С. 129-148.

17. Мелешко В.П., Матюгин В.А., Спорышев П.В. и др. Модель общей циркуляции атмосферы ГГО (версия MGO-03 T63L25) // Труды Главной геофизической обсерватории им. А.И. Воейкова. 2014. № 571. С. 5-87.

18. Толстых М.А., Фадеев Р.Ю., Шашкин В.В. и др. Многомасштабная глобальная модель атмосферы ПЛАВ: результаты среднесрочных прогнозов погоды // Метеорология и гидрология. 2018. № 11. С. 90-99.

19. Fadeev, R.Yu., Ushakov, K.V., Tolstykh, M.A., Ibrayev, R.A. Design and development of the SLAV-INMIO-CICE coupled model for seasonal prediction and climate research // Russian Journal of Numerical Analysis and Mathematical Modelling. 2018. Vol. 33. No. 6. P. 333-340.

20. Ďurán I. B., Geleyn J-F., Váňa F. A Compact Model for the Stability Dependency of TKE Production–Destruction–Conversion Terms Valid for the Whole Range of Richardson Numbers // J. Atmos. Sci. 2014. Vol. 71. P. 3004–3026.

21. Gerard L., Piriou J., Brožková R., Geleyn J-F., Banciu D. Cloud and Precipitation Parameterization in a Meso-Gamma-Scale Operational Weather Prediction Model // Mon. Wea. Rev. 2009. Vol. 137. P. 3960–3977.

22. M.-D. Chou and M. J. Suarez A solar radiation parameterization (CLIRAD-SW) for atmospheric studies // NASA Tech. Memo. 10460. 1999. No. 15.
URL: https://ntrs.nasa.gov/api/citations/19990060930/downloads/19990060930.pdf (accessed: 09.10.2023).

23. T. Tarasova and B. Fomin The Use of New Parameterizations for Gaseous Absorption in the CLIRAD-SW Solar Radiation Code for Models. J. Atmos. and Oceanic Tech. 2007. Vol. 24. No. 6. P. 1157-1162. DOI: 10.1175/JTECH2023.1

24. E. J. Mlawer, S. J. Taubman, P. D. Brown, M. J. Iacono and S. A. Clough RRTM, a validated correlated-k model for the longwave. J. Geophys. Res. 1997. Vol. 102. No. 16. P. 663-682. DOI: 10.1029/97JD00237

25. Ибраев Р.А., Хабеев Р.Н., Ушаков К.В. Вихреразрешающая 1/10° модель Мирового океана // Известия РАН. Физика атмосферы и океана. 2012. № 48 (1). С. 45–55.

26. Hunke E.C., Lipscomb W.H., Turner A.K., Jeffery N., Elliott S. CICE: the Los Alamos Sea Ice Model documentation and software user's manual, version 5.1 // Technical report LA-CC-06-012. Los Alamos National Laboratory: Los Alamos, NM. 2015. URL: https://svn-ccsm-models.cgd.ucar.edu/cesm1/alphas/branches/cesm1_5_alpha04c_timers/components/cice/src/doc/cicedoc.pdf (accessed: 09.10.2023).

27. Kalmykov V.V., Ibrayev R.A., Kaurkin M.N., Ushakov K.V. Compact modeling framework v3.0 for high-resolution global ocean–ice–atmosphere models // Geoscientific model development. 2018. Vol. 11. P. 3983-3997. DOI: 10.5194/gmd-11-3983-2018

28. Valcke S.: The OASIS3 coupler: a European climate modelling community software. Geosci. Model Devel. 2013. Vol. 6. P. 373-388. DOI: 10.5194/gmd-6-373-2013

Фреймворк методов интеллектуальной эвристической оптимизации iOpt

А.В. Сысоев, Е.А. Козинов, К.А. Баркалов, И.Г. Лебедев, Д.А. Карчков, Д.М. Родионов

Нижегородский государственный университет им. Н.И. Лобачевского

В работе представлен фреймворк с открытым исходным кодом iOpt, предназначенный для выбора оптимальных в заданной метрике значений параметров сложных объектов и процессов, например, методов машинного обучения или эвристической оптимизации. Фреймворк разработан на языке Python и позволяет проводить точную настройку параметров как моделей, так и методов, используемых в прикладных исследованиях в различных научных областях. При решении задач выбора оптимальных параметров для сложных моделей и методов возникают проблемы (1) частичной определенности и многоэкстремальности критериев, невыпуклости функциональных ограничений, дискретности варьируемых параметров и (2) отсутствия формульного описания исследуемой модели (модель вида «черный ящик»). Методы, реализованные в iOpt, способны адаптироваться под различные типы задач оптимизации (задачи с дискретными параметрами, невыпуклыми ограничениями, многоэкстремальными критериями) и работать с обобщенными моделями критериев вида «черный ящик». В статье представлено описание архитектуры фреймворка, реализованных в нем методов, типовых сценариев использования, схемы параллельного исполнения на высокопроизводительных системах с общей памятью. Приведены результаты вычислительных экспериментов при решении прикладных задач.

Ключевые слова: высокопроизводительные вычисления, параллельное программирование, эвристическая оптимизация, многоэкстремальные задачи оптимизации, настройка параметров, фреймворк, Python

1. Введение

Фреймворк iOpt ориентирован на решение задач выбора оптимальных значений параметров широкого класса процессов и алгоритмов. Характерными примерами задач, решаемых фреймворком, являются задачи настройки гиперпараметров методов машинного обучения, а также методов эвристической оптимизации. Например, на качество классификации объектов с помощью метода опорных векторов, которое традиционно измеряется метрикой F_1 [1], существенным образом влияют два параметра метода: коэффициент регуляризации и коэффициент ядра. Подбор указанных параметров может значительно повысить значение целевой метрики качества.

Типичной ситуацией для многих эвристических методов решения задач оптимизации, а также методов машинного обучения является наличие лишь нескольких (порядка пяти) настраиваемых параметров, которые определяют качество работы алгоритма.

В силу этого методы фреймворка ориентированы на решение задач не более чем с десятью параметрами и десятью функциональными ограничениями. При этом время проведения одного поискового испытания (т.е. работы настраиваемого алгоритма с одним набором параметров) должно составлять величину порядка минут, а не часов (иначе настройка займет неприемлемое время).

Задача выбора оптимальных параметров рассматривается как задача липшицевой глобальной оптимизации вида [2-6]

$$\varphi(y^*) = min\{\varphi(y), y \in Q\},$$

где $y \in R^N$ – вектор варьируемых параметров, $\varphi(y)$ – целевая функция, Q – область допустимых сочетаний параметров. Предполагается, что в области Q целевая функция $\varphi(y)$ является многоэкстремальной, а сама область Q может быть невыпуклой и неодносвязной.

Методы решения задач липшицевой оптимизации фреймворк реализует, используя:

- схему редукции размерности на основе кривых, заполняющих пространство (кривых Пеано);
- способ эффективного (по времени доступа) хранения накопленной поисковой информации;
- правила выбора перспективных и отсеивания бесперспективных подобластей поиска;
- процедуры локального уточнения найденной оценки глобального оптимума;
- визуализацию процесса решения поставленной задачи оптимизации.

2. Структура фреймворка

Реализация фреймворка iOpt выполнена на принципах объектно-ориентированного программирования в виде системы взаимосвязанных классов.

Взаимодействие пользователя с фреймворком начинается с подготовки класса, описывающего постановку задачи глобальной оптимизации в виде потомка от базового класса **`Problem`**.

Класс **`Problem`** предоставляет единственный публичный метод:

- **`Calculate`** – вычислить значение заданного функционала (ограничения или критерия) в заданной точке.

Класс **`Problem`** и пример класса-потомка с постановкой задачи показан на рис. 1.

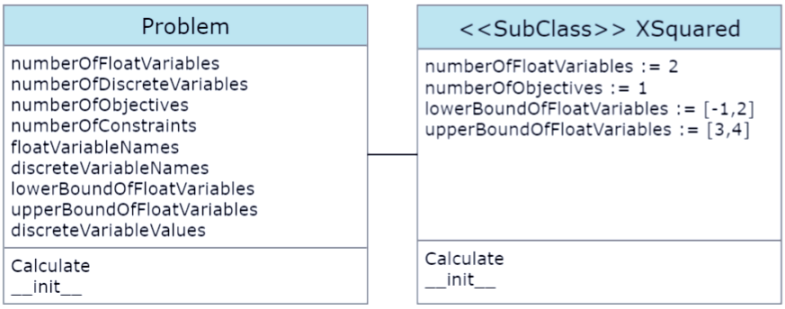


Рис. 1. Постановка задачи оптимизации с целевой функцией параболоидом

Второй шаг, который необходимо выполнить пользователю, для получения глобально-оптимального решения поставленной задачи оптимизации, – создать решатель в виде экземпляра класса **`Solver`** и настроить его параметры (класс **`SolverParameters`**).

Схемы указанных классов с принципиальными полями и методами представлены на рис. 2.

Рис. 2. Классы «Решатель», «Параметры решателя»

Класс **`Solver`** предоставляет следующие публичные методы:

- **`DoGlobalIteration`** – выполнить одну итерацию поиска глобального оптимума задачи оптимизации,
- **`DoLocalRefinement`** – выполнить уточнение текущей оценки глобального оптимума локальным методом оптимизации,
- **`GetResults`** – вернуть текущие результаты решения,

- **`LoadProgress`** – загрузить ранее сохраненное состояние процесса поиска для продолжения счета,
- **`SaveProgress`** – сохранить текущее состояние процесса поиска,
- **`Solve`** – решить задачу оптимизации до достижения заданных параметров (числа испытаний и/или точности).

Параметры класса **`SolverParameters`** представлены следующими полями:

- **`eps`** – точность решения задачи, значение по умолчанию – 0.01,
- **`epsR`** – параметр метода (эпсилон-резервирование), значение по умолчанию – 0.001,
- **`evolventDensity`** – точность (плотность) построения развертки, значение по умолчанию – 10,
- **`itersLimit`** – максимальное число испытаний для решения задачи оптимизации, значение по умолчанию – 20 000,
- **`r`** – параметр метода (надежность), значение по умолчанию – 2.0,
- **`refineSolution`** – необходимость уточнять оценку глобального оптимума после остановки процесса решения, значение по умолчанию – **`false`**,
- **`startPoint`** – начальная точка в области поиска решения, с которой стартует процесс, значение по умолчанию – не задана.
- **`numberOfParallelPoints`** – число параллельно проводимых испытаний.

3. Алгоритм функционирования фреймворка

На рис. 2 приведена блок-схема общего алгоритма функционирования фреймворка.

На вход алгоритма подается объект класса на языке Python с унифицированной моделью задачи оптимизации.

Унифицированная модель включает следующие элементы:

- Область поиска решения. Включает в себя:
  - размерность модели,
  - параметры модели,
  - диапазоны изменения параметров.
- Функциональные ограничения (при наличии), заданные в виде функций на языке Python.
- Целевые функции, заданные в виде функций на языке Python.

Дополнительно входные данные могут содержать:

- Начальное приближение – начальные значения параметров модели.
- Поисковую информацию, накопленную на предыдущих запусках фреймворка iOpt при решении той же задачи.

При наличии во входной информации начального приближения оно задается в качестве стартовой точки для индексного метода оптимизации. При отсутствии начального приближения метод выбирает точку самостоятельно.

При наличии накопленной ранее поисковой информации база поисковой информации инициализируется на ее основе, что дает возможность не решать задачу оптимизации «с нуля», а продолжить процесс поиска глобального оптимума, возможно с новыми настройками метода или другими условиями останова.

На основе поданного на вход алгоритма объекта класса на языке Python с унифицированной моделью задачи оптимизации выполняется инициализация решаемой задачи оптимизации, включающая создание и настройку внутренних структур данных фреймворка.

На следующем этапе выполняется настройка метода решения сформированной задачи оптимизации. Метод решения в рамках фреймворка iOpt представляет собой объект класса с унифицированной моделью метода оптимизации, основанной на характеристической представимости алгоритмов глобального поиска.

Модель метода включает следующие элементы:

- параметры метода,
- диапазоны изменения параметров,

- значения по умолчанию для параметров метода,
- правило вычисления характеристики в виде формулы или функции / фрагмента кода на языке Python,
- правило останова процесса поиска, который выполняется при достижении:
  - заданной точности,
  - максимального числа итераций.

На следующем этапе с учетом всех выполненных ранее подготовительных шагов, выполняется запуск индексного алгоритма. В рамках индексного алгоритма реализована параллельная схема проведения испытаний. Алгоритм позволяет выбрать несколько перспективных сочетаний параметров и провести вычисления для них независимо [2,3,5].

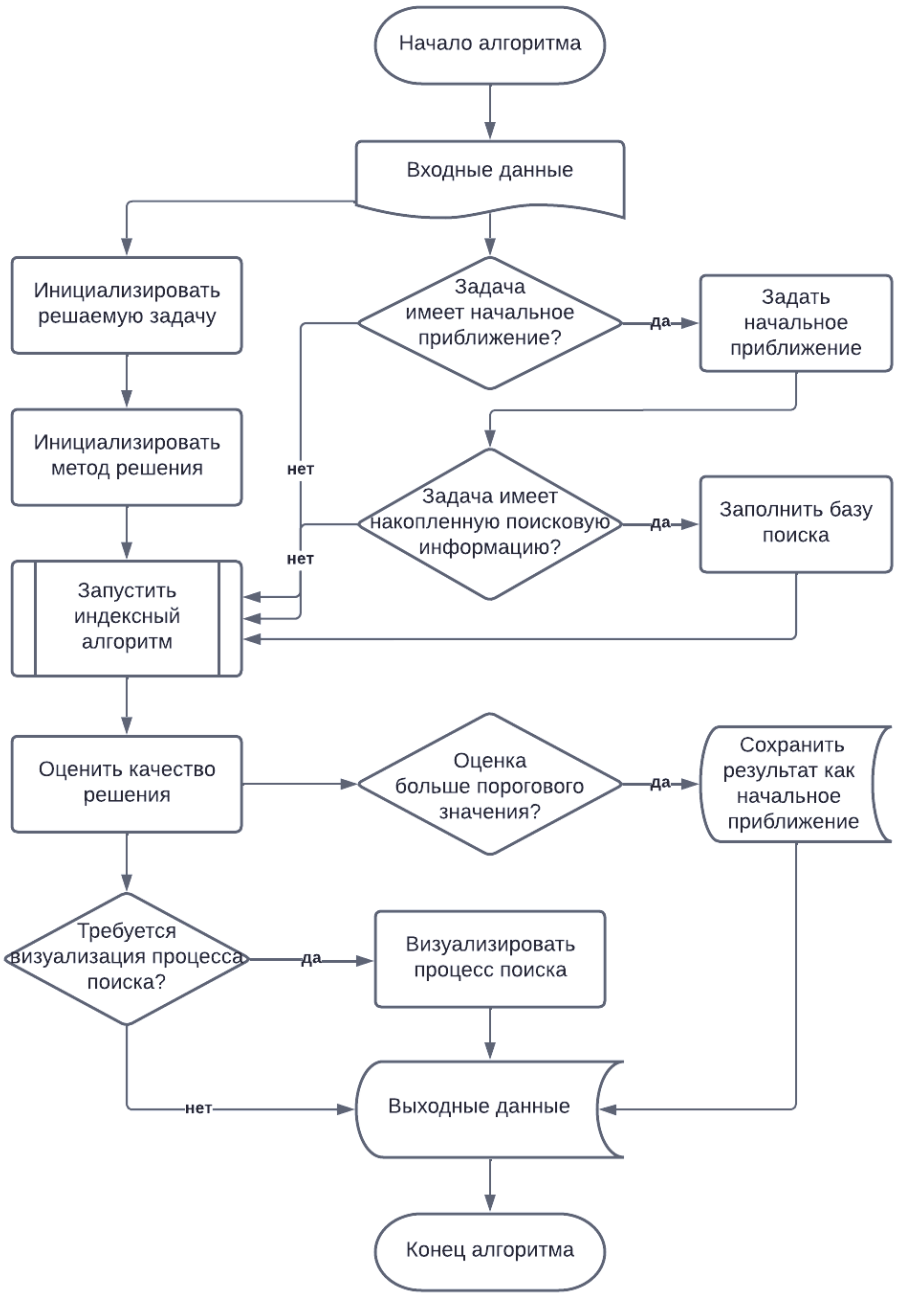


Рис. 3. Блок-схема общего алгоритма функционирования фреймворка

После завершения работы индексного алгоритма, выполняется оценка качества полученного решения. Если останов индексного алгоритма произошел по максимальному числу итераций,

т.е. требуемая точность решения не достигнута, текущая оценка глобального оптимума сохраняется как начальное приближение для последующего запуска общего алгоритма фреймворка.

При необходимости процесс поиска может быть визуализирован с использованием предусмотренных во фреймворке графических индикаторов.

По результатам работы индексного алгоритма формируются выходные данные в виде:

- достигнутая точность,
- выполненное число итераций,
- полученная оценка глобального минимума:
  - значения параметров модели,
  - значение целевой функции,
- поисковая информация в виде таблицы точек, в которых проводились испытания.

4. Результаты вычислительных экспериментов

Вычислительные эксперименты проводились на суперкомпьютере «Лобачевский» Нижегородского государственного университета. В экспериментах использовался вычислительный узел с двумя 64-ядерными процессорами AMD EPYC 7742 (всего 128 вычислительных ядра). На узле установлено 512 Gb оперативной памяти, операционная система CentOS 7.

Задачи выбора оптимальных значений параметров сложных объектов, процессов, алгоритмов возникают во многих прикладных областях. Эффективность параллельной реализации фреймворка iOpt проверялась на решении двух задач: настройка параметров алгоритма машинного обучения для построения модели предсказания рака молочной железы, а также настройка параметров генетического алгоритма при решении задачи коммивояжёра.

Для построения модели предсказания рака молочной железы в рамках исследования использовался известный набор данных [7]. Набор данных включает в себя 569 примеров, в каждом из которых по 30 числовых характеристик. Характеристики рассчитываются по оцифрованному изображению тонкоигольной аспирации (ТАБ) массы молочной железы. Они описывают характеристики ядер клеток, присутствующих на изображении. Распределение по классам следующее: 212 злокачественных, 357 доброкачественных опухолей. Модель для предсказания класса строилась с использованием метода опорных векторов (SVC) [8]. С использованием iOpt выбирались два вещественных параметра: коэффициент регуляризации (C, диапазон выбора $[10^1..10^6]$), коэффициент ядра (gamma, диапазон выбора $[10^{-7}..10^{-3}]$). В качестве критерия качества использовалась метрика F_1 [1].

На рис. 4 представлен график изменения качества модели в зависимости от настраиваемых параметров. Из графика видно, что зависимость от параметров является не линейной, а выбор параметров может сильно повлиять на качество предсказания.

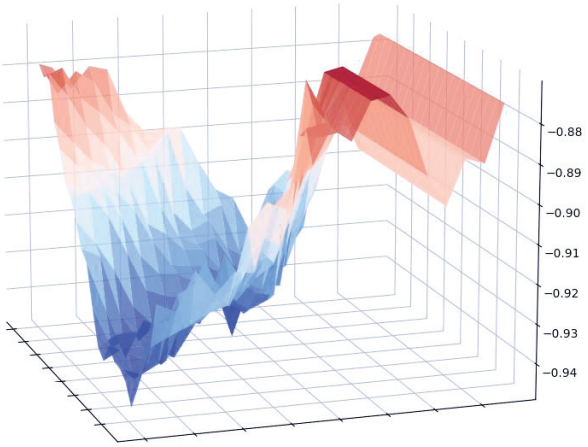


Рис. 4. Точность предсказания рака молочной железы по метрике F_1
(ось Z, больше значение лучше) в зависимости от параметров метода SVC

Задача решалась с использованием scikit-learn с параметрами по умолчанию, а также после настройки параметров методами iOpt. С параметрами по умолчанию удалось решить задачу со значением $F_1 = 0{,}87$. Для настройки параметров методами iOpt выполнялось 1000 вычислений значения F_1 при разном сочетании параметров C и gamma. Результат настройки параметров в зависимости от числа используемых параллельных процессов приведены в таблице 1.

Таблица 1. Решение задачи предсказания рака молочной железы после настройки параметров метода SVC

Число процессов	F_1	Время настройки параметров	Ускорение
1	0,975	31,199	1
5	0,975	7,876	4,0
10	0,975	4,775	6,5
20	0,975	3,251	9,6
40	0,975	2,002	15,6
80	0,975	1,616	19,3

Исходя из таблицы 1 можно увидеть, что iOpt позволили повысить качество предсказания модели. Применение параллельных методов позволило настроить метод построения модели с существенным ускорением.

Из таблицы 1 можно заметить, что ускорение является не линейным. Отсутствие линейного ускорения связано с относительно малым временем обучения модели при заданных параметрах C и gamma. Для демонстрации эффективности предлагаемого фреймворка iOpt рассматривалась задача большей вычислительной сложности – настройка параметров генетического алгоритма при решении задачи коммивояжёра.

Содержательную постановку задачи коммивояжёра можно сформулировать следующим образом: «Для заданного списка городов и расстояний между каждой парой городов, найти кратчайший возможный маршрут, который проходит через каждый город ровно один раз и возвращается в родной город». Для получения оптимального расстояния, которое необходимо пройти коммивояжёру, существует большое количество алгоритмов: полный перебор, случайный перебор, метод ближайшего соседа, метод дешёвого включения, метод минимального остовного дерева, метод ветвей и границ.

В рамках исследования применялся генетический алгоритм, реализованный в библиотеке scikit-opt [9,10]. Средствами iOpt подбирались два параметра алгоритма – вероятность мутации (диапазон выбора $[0..1]$), а также размер популяции (диапазон выбора $[0..1]$). Для поиска параметров выполнялось 400 испытаний. В качестве задачи использовался набор данных «a280» из базы [11]. Результаты вычислительных экспериментов представлены в таблице 2.

Таблица 2. Решение задачи коммивояжёра генетическим алгоритмом с настройкой параметров

Число процессов	длина пути (лучше меньше)	время настройки параметров	ускорение
1	16040,52	1162,5	1
5	16223,25	316,0	3,7
10	16011,50	164,3	7,1
20	16089,76	90,25	12,9
40	16216,21	54,69	21,3
80	15910,54	34,86	33,3

Исходя из таблицы 2, можно видеть, что iOpt позволил решить задачу коммивояжёра со сравнимым качеством. Также при сравнении результатов таблиц 1 и 2 легко заметить, что при решении более сложных задач эффективность (масштабируемость) параллельной версии алгоритма повысилась.

Заключение

В статье представлен фреймворк с открытым исходным кодом iOpt. Фреймворк разработан на языке Python и позволяет решать задачи поиска глобально-оптимальных решений, включая настройку параметров методов машинного обучения или эвристической оптимизации.

Методы, реализованные в iOpt, способны адаптироваться под различные типы задач оптимизации (задачи с дискретными параметрами, невыпуклыми ограничениями, многоэкстремальными критериями) и работать с обобщенными моделями критериев вида «черный ящик». В статье представлено описание архитектуры фреймворка, реализованных в нем методов, типовых сценариев использования, схемы параллельного исполнения на высокопроизводительных системах с общей памятью. Приведены результаты вычислительных экспериментов при решении прикладных задач.

Литература

1. Flach P.A., Kull M. Precision-Recall-Gain Curves: PR Analysis Done Right // NIPS, 2015.
2. Strongin R.G., Sergeyev Y.D. Global Optimization with Non-Convex Constraints. Sequential and Parallel Algorithms. – The Netherlands, Dordrecht: Kluwer Academic Publishers, 2000. – 728 p.
3. Стронгин Р.Г., Гергель В.П., Гришагин В.А., Баркалов К.А. Параллельные вычисления в задачах глобальной оптимизации. – М.: Издательство Московского университета. 2013. – 280 с.
4. Сергеев Я.Д., Квасов Д. Диагональные методы глобальной оптимизации. – ФИЗМАТЛИТ. – 2008. – 352 с.
5. Sergeyev Y.D., Grishagin V.A. Parallel Asynchronous Global Search and the Nested Optimization Scheme // J. Comput. Anal. Appl. – 2001. – V. 3(2). – P. 123–145.
6. Sergeyev Y.D., Strongin R.G., Lera D. Introduction to global optimization exploiting space filling curves. – Verlag New York:Springer. – 2013. – 125 p.
7. Breast Cancer Wisconsin (Diagnostic) Data Set. URL: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)
8. Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines // ACM Transactions on Intelligent Systems and Technology, 2011. №27. V. 2. P.1-27.
9. scikit-opt URL: https://github.com/guofei9987/scikit-opt
10. Yu, J., He, Y., Yan, Q., & Kang, X. SpecView: Malware Spectrum Visualization Framework With Singular Spectrum Transformation // IEEE Transactions on Information Forensics and Security, 2021. №16. P. 5093-5107.
11. TSPLIB is a library of sample instances for the TSP (and related problems) from various sources and of various types. URL: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

Экспериментальная оценка результатов внедрения технологии NVIDIA GPUDirect на суперкомпьютере НИУ ВШЭ*

Р.А. Чулкевич, В.И. Козырев, П.С. Костенецкий, А.А. Раимова

Национальный исследовательский университет «Высшая школа экономики»

Оптимизация использования вычислительных ресурсов на высокопроизводительных кластерах является важной задачей в условиях высокой загрузки. Одним из способов такой оптимизации является применение современных технологий. В то же время, на разных серверных архитектурах поведение технологий может отличаться. В частности, влияние оказывает то, как именно осуществляется взаимодействие компонентов аппаратной архитектуры (например, между GPU и InfiniBand адаптером). В данной статье анализируется применение технологий NVIDIA GPUDirect RDMA и NVIDIA GPUDirect Copy на различных архитектурах вычислительных узлов суперкомпьютерного комплекса cHARISMa. Рассматривается изменение задержки и скорости передачи данных между GPU на разных вычислительных узлах при различных комбинациях задействованных технологий. В лучших случаях задержка при передаче данных уменьшилась в 7.8 раза, а увеличение пропускной способности составило до 286%. Полученные результаты показывают, что применение технологий GPUDirect Copy и GPUDirect RDMA с учетом аппаратной архитектуры может значительно ускорять выполнение задач, как использующих частые обмены с памятью GPU в рамках одного узла, так и выполняющих обмены между GPU на нескольких вычислительных узлах.

Ключевые слова: GPUDirect RDMA, GDR, GPUDirect Copy, GDRC, вычислительный кластер, NUMA, InfiniBand.

1. Введение

С увеличением числа пользователей и научных проектов, выполняемых на суперкомпьютерном комплексе *cHARISMa* НИУ ВШЭ, нагрузка на его вычислительные ресурсы существенно возросла. Возникла потребность в увеличении эффективности использования имеющихся вычислительных ресурсов, путем применения новых программных инструментов и оптимизаций. Одним из таких инструментов стала разработанная в НИУ ВШЭ система HPC TaskMaster, позволяющая автоматически находить и отменять некорректно запущенные и неэффективные задачи пользователей [6]. Другим инструментом для увеличения эффективности суперкомпьютера стала оптимизация системного программного обеспечения. Так обновление программного обеспечения системы хранения данных суперкомпьютера позволило повысить скорость записи данных на 32.8% [1], что значительно ускорило задачи, активно использующие файловое хранилище.

Данная работа посвящена новому этапу оптимизации программной конфигурации суперкомпьютерного комплекса *cHARISMa*, ускорению выполнения вычислительных задач, использующих графические процессоры, путем изменения процесса обмена данными с памятью GPU. Весьма важным и значимым в этой области является семейство технологий *NVIDIA GPUDirect*.

Технологии *NVIDIA GPUDirect* внедряются на многих суперкомпьютерных комплексах [3, 9, 12]. Например, в статье [3] приводятся данные о латентности в типовых задачах машинного обучения и анализируются преимущества внедрения аппаратного ускорения передачи данных. Авторы получили результат, что GPUDirect RDMA может сократить задержки передачи данных от 15 до 50%, что существенно ускоряет задачи машинного обучения.

* Исследование выполнено с использованием суперкомпьютерного комплекса НИУ ВШЭ [5].

Эффективность технологий *GPUDirect* напрямую связана с аппаратной архитектурой вычислительных узлов кластера, поэтому важно сравнивать производительность до и после внедрения технологий и использовать их в случае получения положительного эффекта. В этой статье рассказывается о результатах применения на всех типах вычислительных узлов суперкомпьютера НИУ ВШЭ двух технологий *GPUDirect Remote Direct Memory Access* и *GPUDirect Copy*.

Технология *GPUDirect Remote Direct Memory Access* (далее – *GDR*) предназначена для исключения центрального процессора и оперативной памяти из процесса обмена данных между GPU на разных вычислительных узлах. GDR основана на возможности GPU-ускорителей соотносить (*expose*) часть памяти GPU с областью памяти PCIe-устройства, известной также как базовые адресные регистры *BAR (Base Address Register*). В этом случае PCIe-устройства получают прямой доступ к памяти GPU через Peer-to-Peer-соединение, исключая необходимость в использовании CPU и промежуточных буферов оперативной памяти.

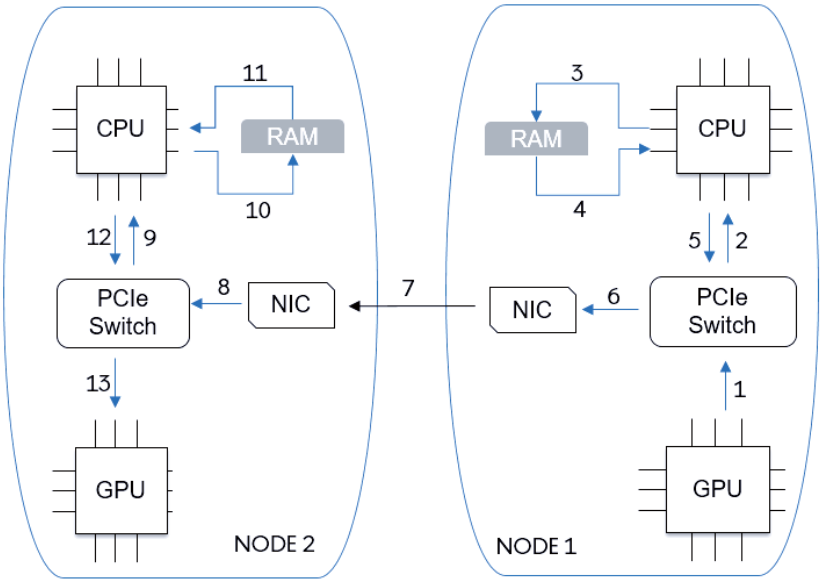


Рис. 1. Обмен данными между GPU без применения GDR

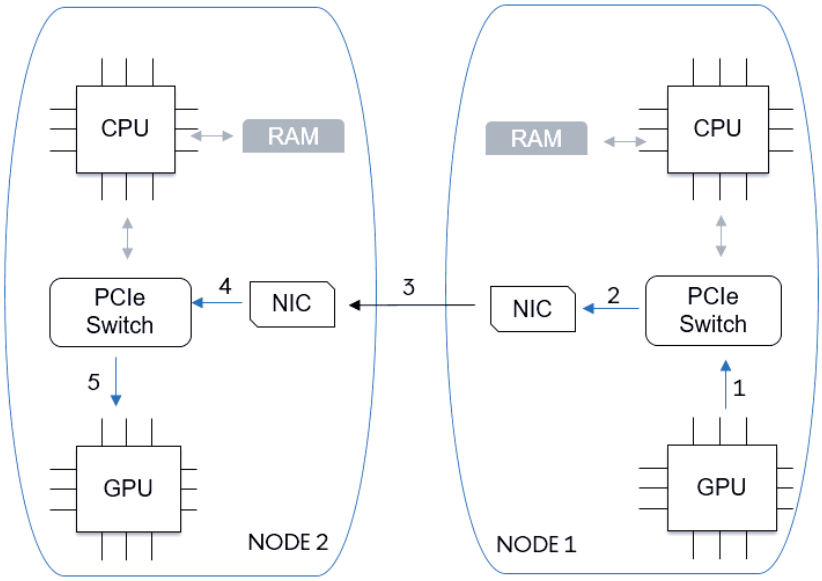


Рис. 2. Обмен данными между GPU с применением GDR

На рис. 1 и рис. 2 показано сравнение процесса передачи данных между двумя GPU, расположенными на разных вычислительных узлах, в стандартном варианте и с применением

технологии. Без GDR данные сначала попадают в оперативную память с использованием CPU, затем передаются через сетевой адаптер в оперативную память другого вычислительного узла и в конечном итоге размещаются в оперативной памяти графического ускорителя. При использовании GDR процесс обмена данными выполняется через сетевые адаптеры напрямую, уменьшая задержки и увеличивая скорость передачи. Если в обычном режиме передача выполняется за 13 этапов, то при использовании GDR, количество этапов сокращается до 5.

Если основной целью GDR является реализация прямого доступа в память GPU с различных устройств, то технология *GPUDirect Copy* (далее – *GDRC*) предназначена для создания сопоставлений памяти GPU с CPU (*CPU Mapping of the GPU Memory*). В результате, в пространстве пользователя создается сопоставление (mapping) памяти GPU, которое может быть использовано, как если бы это была обычная оперативная память вычислительного узла. Это позволяет снизить задержки и накладные расходы (overhead) при передаче данных из/в GPU.

На рис. 3. показано сравнение использования стандартного механизма работы с памятью через cudaMemcpy и механизмов из библиотеки GDRCopy. CudaMemcpy использует DMA Engine API для перемещения данных между CPU и памятью GPU [2]. Это приводит к задержкам и накладным расходам при передаче при передаче данных. GDRCopy позволяет CPU напрямую обращаться к памяти GPU через сопоставления BAR (Base Address Registers), значительно уменьшая задержку при передаче.

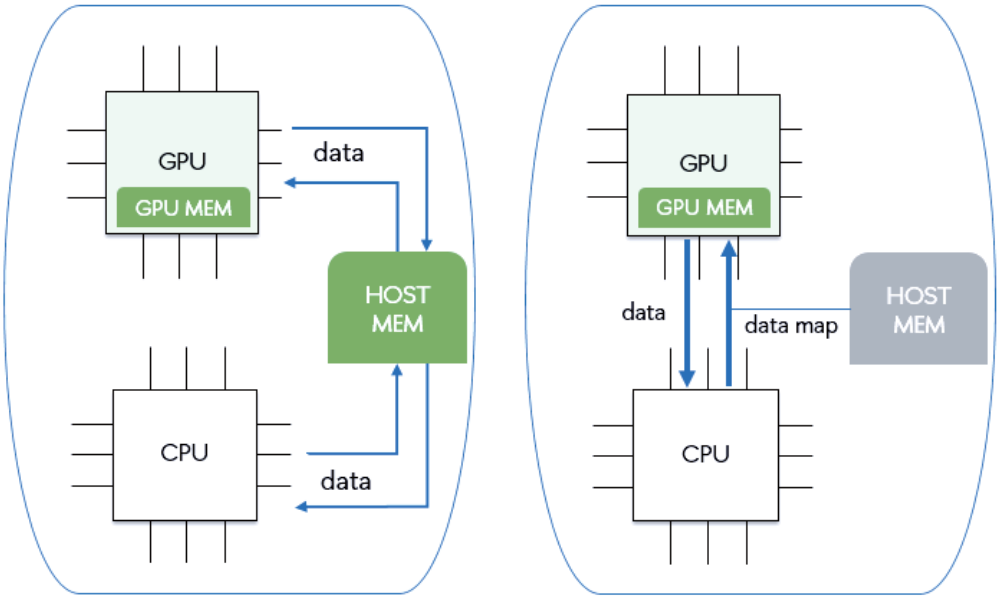


Рис. 3. Разница обмена данными с памятью GPU:
слева – по умолчанию cudaMemcpy, справа – с применением GDRC

Поддержка GDR и GDRC реализована во всех современных коммуникационных библиотеках (OpenMPI, MVAPICH2, NVIDIA HPC-X и др.). Это позволяет задействовать новые технологии во многих вычислительных задачах, путем подключения обновленных библиотек.

2. Архитектура вычислительных узлов

В суперкомпьютере НИУ ВШЭ *cHARISMa* есть три типа вычислительных узлов с GPU, обладающих различной аппаратной архитектурой.

1. *Tun A/B*: Dell PowerEdge C4140K, 2 x Intel Xeon Gold 6152, 768/1536 ГБ ОЗУ, 4 x NVIDIA Tesla V100 32 ГБ NVLink, 2 x InfiniBand EDR – см. Рис 4.
2. *Tun C*: Dell PowerEdge C4140M, 2 x Intel Xeon Gold 6240R, 768 ГБ ОЗУ, 4 x NVIDIA Tesla V100 32 ГБ NVLink, 2 x InfiniBand EDR – см. Рис 4.
3. *Tun E*: HPE XL675d Gen10+, 2 x AMD EPYC 7702, 1024 ГБ ОЗУ, 8 x NVIDIA Tesla A100 80 ГБ NVLink, 2 x InfiniBand EDR – см. Рис 5.

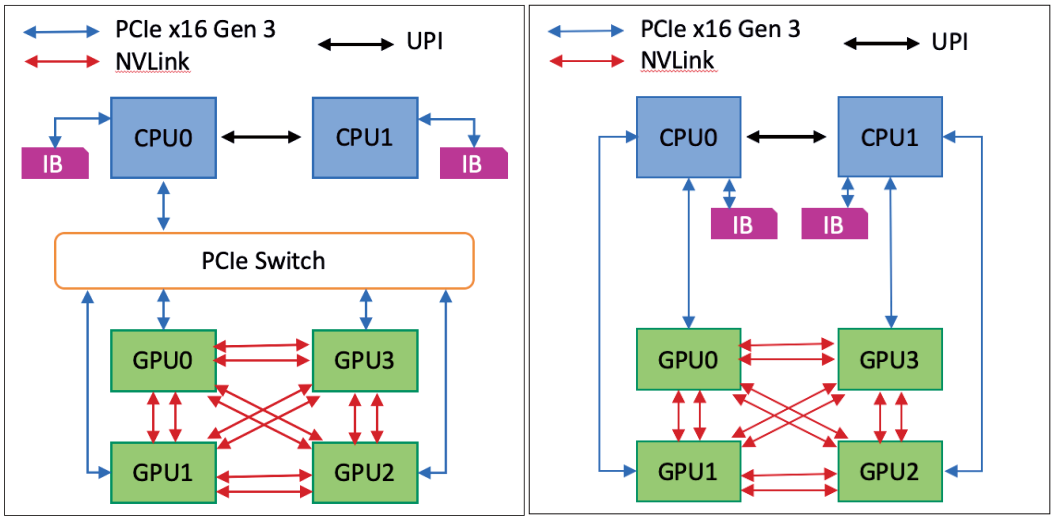


Рис. 4. Аппаратная архитектура вычислительных узлов типов A, B (слева) и C (справа)

На вычислительных узлах типов A/B и C доступ GPU-ускорителей к InfiniBand адаптеру производится через PCIe-мост и центральный процессор. Дополнительно, на вычислительных узлах типа A и B, доступ к модулю GPU-ускорителей с хоста осуществляется через внутренний PCIe-коммутатор, что усложняет процесс передачи данных и немного уменьшает производительность в пиковых режимах.

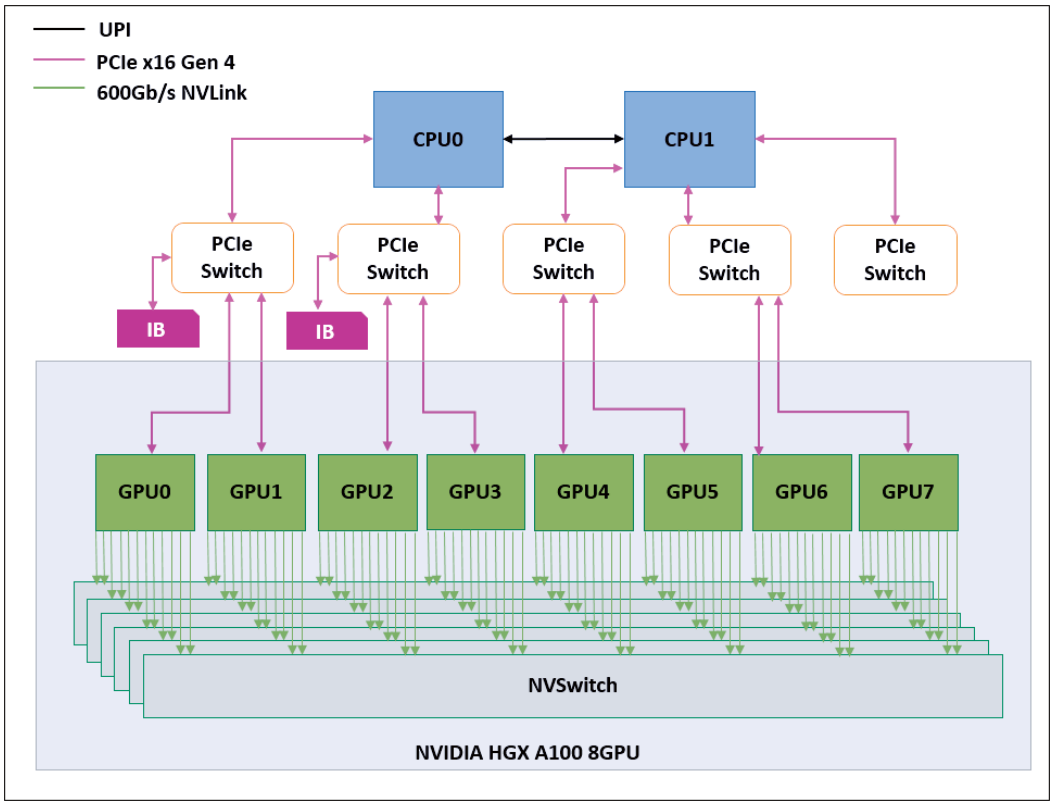


Рис. 5. Аппаратная архитектура вычислительных узлов типа E

Вычислительные узлы типа E сконструированы по подобию «архитектурного эталона» от NVIDIA, который был реализован в NVIDIA DGX A100. Здесь все GPU ускорители размещены на отдельной плате NVIDIA HGX и объединены скоростными внутренними коммутациями NVSwitch. В данной архитектуре доступ GPU к InfiniBand-адаптеру и NVMe-диску может происходить напрямую через PCIe-коммутатор, не требуя предварительной передачи данных в оперативную память узла.

В каждом вычислительном узле, независимо от его типа, установлено два сетевых адаптера InfiniBand EDR 100 Гб/с, подключенных к разным сетевым коммутаторам Mellanox.

Конфигурация вычислительных узлов сильно влияет на эффективность использования технологий GDR и GDRC. Скорость передачи данных при использовании GPUDirect RDMA также снижается при наличии архитектурных особенностей PCIe и NUMA-эффектов [2]. Стоит также отметить, что модуль ядра GDRC может быть задействован в системном ПО UCX, необходимом для эффективной работы более крупного системного ПО – OpenMPI (GPU-Aware), которое, в свою очередь, является фундаментом для сборок многих прикладных пакетов и может оказывать влияние на их производительность при расчетах.

3. Конфигурация программного обеспечения

Базовая операционная система кластера *cHARISMa* – Linux Centos 7.9 [1].

В процессе внедрения технологий NVIDIA GPUDirect была выполнена сборка и установка следующего системного программного обеспечения:

1) CUDA v12;
2) библиотека для высокоскоростных и низко-латентных сетей UCX v1.14.1 с параметрами *--with-cuda* и *--with-gdrcopy*;
3) библиотека для выполнения глобальных операций UCC v1.2.0 с параметрами *--with-cuda* и *--with-ucx*;
4) OpenMPI v4.1.5 с параметрами *--with-cuda, --with-ucx* и *--with-ucc*.

4. Тестирование

В рамках данного исследования на всех типах вычислительных узлов замерялись две величины: задержка и пропускная способность при двунаправленной передаче данных из GPU одного узла в GPU другого узла. Замеры выполнялись с использованием набора тестов OSU Micro-Benchmarks v7.0.1 [7], собранного с указанным выше набором ПО.

Тестирование задержки при передаче данных производилось по принципу «пинг-понг». Узел-отправитель посылает сообщение определенного размера и ожидает ответа от узла-получателя. Тот, в свою очередь, получает сообщение и отправляет его обратно с тем же объемом данных. В процессе тестирования проводится несколько итераций и в качестве итогового результата выбирается средняя величина односторонней задержки.

Тестирование пропускной способности осуществлялось путем отправки узлом-отправителем фиксированного количества сообщений одного размера, следующих друг за другом. Узел-получатель отправлял ответ только после получения всех этих сообщений. Производилось несколько итераций этого процесса, и пропускная способность рассчитывалась на основе прошедшего времени (с момента отправки первого сообщения до момента получения ответа от узла-получателя) и количества байт, отправленных узлом-отправителем. Затем узлы менялись ролями, и отправка сообщений производится в обратную сторону. Итоговый результат является суммарной пропускной способностью в две стороны.

Для чистоты экспериментов, тестирование проводилось во время профилактических работ, когда на суперкомпьютере отсутствовала какая-либо посторонняя вычислительная нагрузка. На выбранных вычислительных узлах были использованы одни и те же ядра центрального процессора, области оперативной памяти, GPU-ускоритель и сетевой адаптер, объединенные в один *узел NUMA* (*NUMA node*). Известно, что влияние *NUMA* (*Non-Uniform Memory Access*) может быть существенно [4, 11]. Корректная привязка MPI-процессов к правильному узлу NUMA во время тестирования обеспечивалась с помощью утилиты *numactl* [8].

При выполнении тестирования учитывался тот факт, что GDRCopy в отношении задержек на передачу данных эффективна для небольших сообщений [10] – опытным путем было установлено, что независимо от типа узла, для cHARISMa это размер до 8КБ: именно на этом размере задержка превышает стандартную (без применения технологий), а затем GDRCopy просто не оказывает эффекта. Для GPUDirect RDMA наоборот более интересна пропускная способность, оценка которой не столь значима на пакетах с маленьким размером. Поэтому для наглядности мы приводим графики задержки для сообщений с размером до 8КБ, а графики

пропускной способности – свыше 8КБ. Результаты тестирования для каждого типа GPU-узлов суперкомпьютера *cHARISMa* приведены на рис. 6–11.

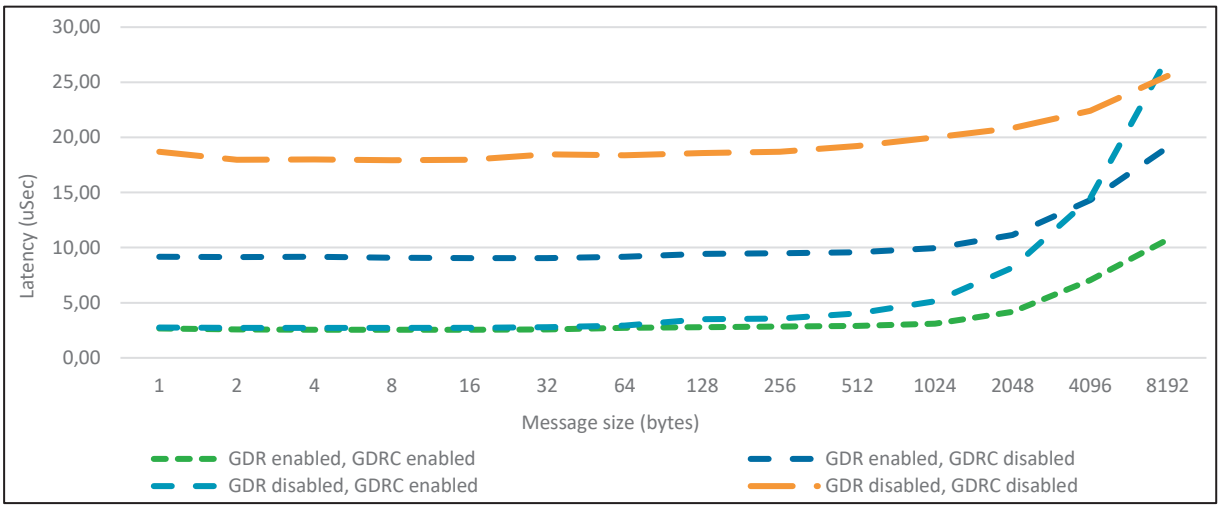


Рис. 6. Задержка передачи на узлах типа A/B

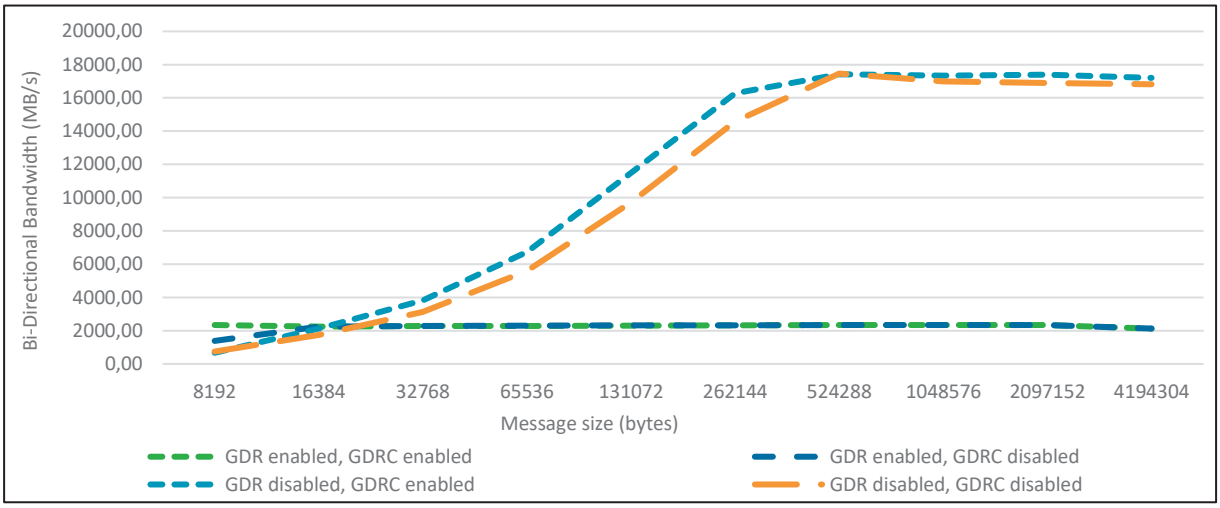


Рис. 7. Пропускная способность на узлах типа A/B

На узлах типа A/B задержка при передаче данных уменьшилась в 7 раз. Наибольший прирост эффективности этого показателя заметен при передаче небольших пакетов данных, размером до 32 байт. Включение GDR на таких узлах дает прирост пропускной способности на пакетах данных, размером до 16 КБ, но при увеличении размера пакета GDR значительно замедляет передачу данных.

На вычислительных узлах типа A/B и C заметно влияние различающейся аппаратной архитектуры и, конкретно, необходимость взаимодействия с PCIe-мостом при передаче данных (см. рис. 4). Так на узлах типа C задержка при передаче данных изначально ниже из-за более эффективной аппаратной архитектуры. После включения GDR и GDRC задержка уменьшилась в 5.8 раз. Наибольшее уменьшение задержки видно на небольших пакетах данных, размером до 32 байт. В то же время, пропускная способность существенно ускоряется при размере пакетов от 64 КБ.

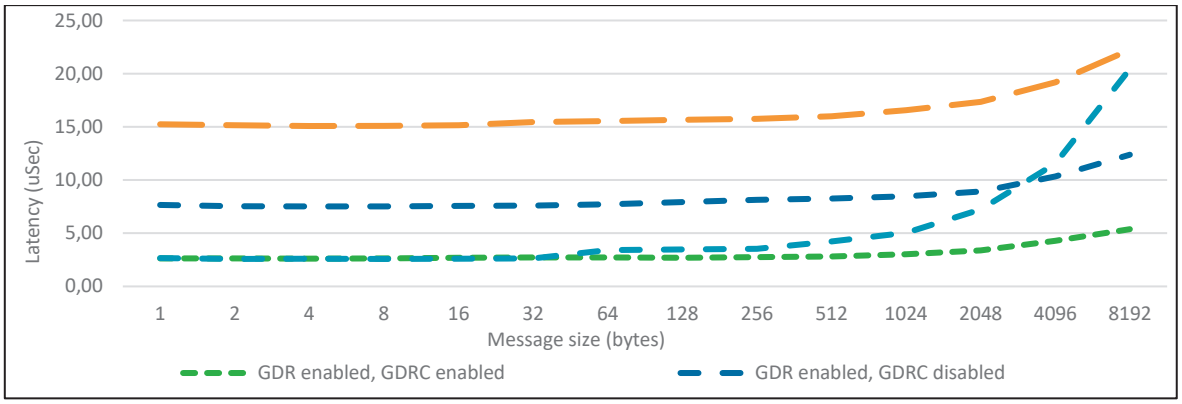


Рис. 8. Задержка передачи на узлах типа C

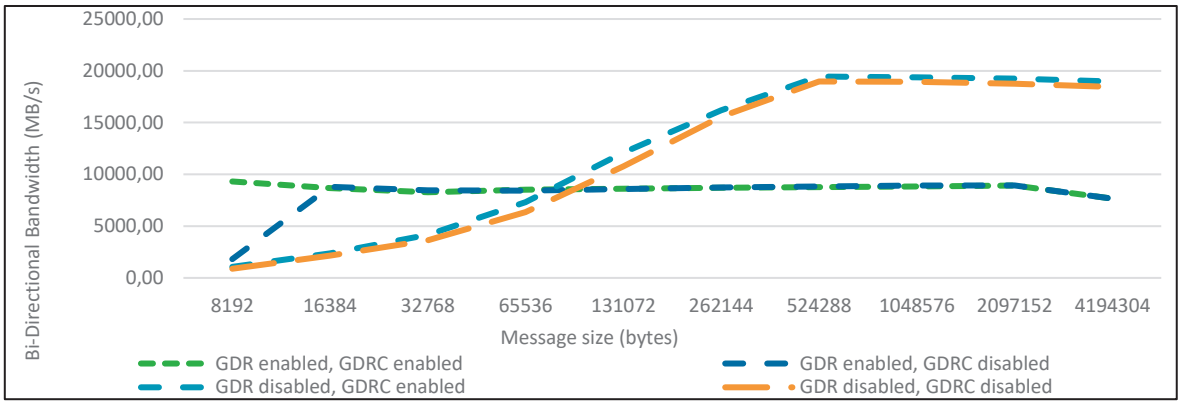


Рис. 9. Пропускная способность на узлах типа C

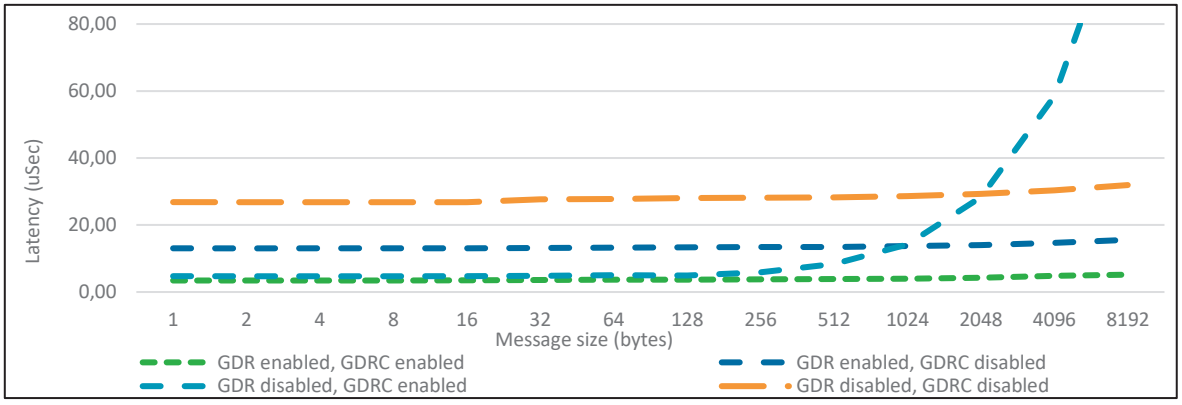


Рис. 10. Задержка на узлах типа E

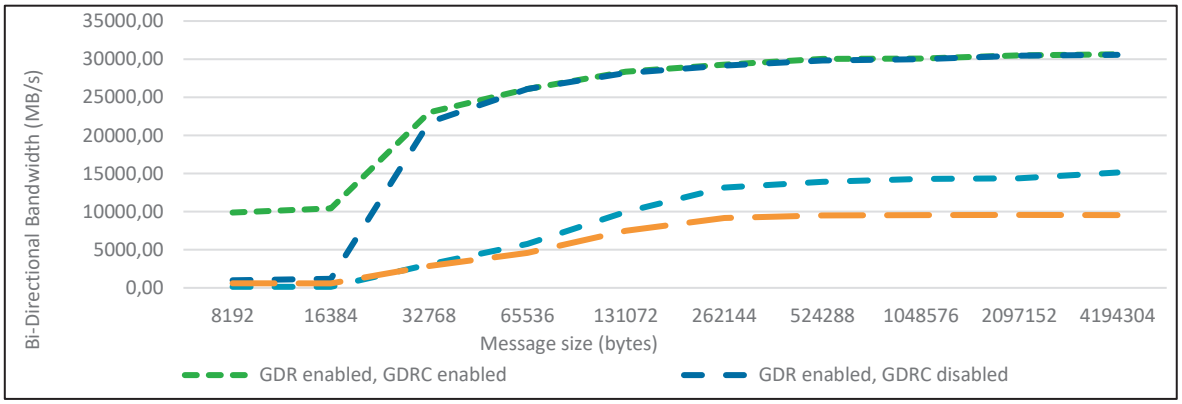


Рис. 11. Пропускная способность на узлах типа E

Наилучшие результаты применения технологий GDR и GDRC были получены на наиболее современных вычислительных узлах типа E. Задержка при передаче данных уменьшилась в 7.8 раз. Увеличение пропускной способности составило до 286% на небольших пакетах данных и до 100% на пакетах данных размером более 64 КБ. Пиковая скорость передачи данных достигла 30 400 МБ/с (243.2 Гигабит/с). Негативного влияния от включения сразу двух технологий (GDR и GDRC) на вычислительных узлах типа E замечено не было, поэтому по умолчанию они будут включены обе.

5. Выводы

Оптимальным вариантом для снижения задержки на передачу данных в память GPU для всех типов узлов стало совместное применением обеих технологий: GDR и GDRC. Что касается пропускной способности, то здесь архитектурные особенности сыграли ключевую роль, так лучшим вариантом для узлов типов A/B, C оказался вариант с применением GDRC, но с отключением GDR. Для наиболее современных вычислительных узлов Типа E – наибольший эффект дало включение обеих технологий.

Технологии GDR и GDRC доступны для использования на суперкомпьютере cHARISMa НИУ ВШЭ. Пользователи могут активировать их использование путем установки переменных окружения во время постановки задачи в очередь. В дальнейшем планируется реализовать более удобный автоматический механизм применения технологий, исключающий человеческие ошибки.

Внедрение современных коммуникационных технологий положительно сказывается на производительности суперкомпьютера cHARISMa в условиях высокой нагрузки. В результате внедрения и настройки технологии GPUDirect на суперкомпьютере cHARISMa НИУ ВШЭ производительность ПО, использующего GPU, повысилась на 3% по данным системы HPC TaskMaster [6] (на данный момент это общая цифра сводной статистики – в дальнейшем планируется оценить влияние на различные прикладные пакеты и процессы обучения искусственных нейронных сетей). Исходя из статистики расчетов с применением GPU на данном суперкомпьютере, такой прирост можно оценить как высвобождение до 37 000 GPU-часов машинного времени в год. Дополнительное машинное время позволяет выполнять большее количество научных проектов без затрат на покупку новых вычислительных узлов.

Литература

1. Чулкевич Р.А., Козырев В.И., Шамсутдинов А.Б., Костенецкий П.С. Сравнение производительности параллельной СХД суперкомпьютера с разными версиями файловой системы Lustre // Russian Supercomputing Days, 2022.
2. Ammendola R. et al. GPU peer-to-peer techniques applied to a cluster interconnect // 2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum, Cambridge, MA, USA, 2013, pp. 806-815. DOI: 10.1109/IPDPSW.2013.128.
3. Hanafy W. et. al. Understanding the Benefits of Hardware-Accelerated Communication in Model-Serving Applications // arXiv, 2023/
4. Hollowell C. et. al. The Effect of NUMA Tunings on CPU Performance // Journal of Physics: Conference Series. 2015. Vol. 664, P. 092010. DOI: 10.1088/1742-6596/664/9/092010.
5. Kostenetskiy P.S., Chulkevich R.A., Kozyrev V.I. HPC Resources of the Higher School of Economics // Journal of Physics: Conference Series. 2021. Vol. 1740, No. 1. P. 012050. DOI: 10.1088/1742-6596/1740/1/012050.
6. Kostenetskiy P.S., Chulkevich R.A., Kozyrev V.I., Shamsutdinov A.B. HPC TaskMaster - Task Efficiency Monitoring System for the Supercomputer Center // Parallel Computational Technologies. PCT 2022. Communications in Computer and Information Science, vol 1618. Springer, Cham. DOI: 10.1007/978-3-031-11623-0_2.

7. OSU Micro-Benchmarks. The Ohio State University. URL: http://mvapich.cse.ohio-state.edu/benchmarks/ (дата обращения: 01.07.2023).
8. RedHat Performance Tuning Guide. URL: https://access.redhat.com/documentation/ru-ru/red_hat_enterprise_linux/7/html/performance_tuning_guide/sect-red_hat_enterprise_linux-performance_tuning_guide-tool_reference-numactl (дата обращения: 01.07.2023).
9. Rossetti D. Benchmarking GPUDirect RDMA on Modern Server Platforms. URL: http://devblogs.nvidia.com/benchmarking-gpudirect-rdma-on-modern-server-platforms (дата обращения: 01.07.2023).
10. Shi R., et.al. Designing efficient small message transfer mechanism for inter-node MPI communication on InfiniBand GPU clusters // 21st International Conference on High Performance Computing, HiPC 2014. 10.1109/HiPC.2014.7116873.
11. Spafford K., Meredith J., Vetter J. Quantifying NUMA and contention effects in multi-GPU systems // Proceedings of the Fourth Workshop on General Purpose Processing on Graphics Processing Units (GPGPU-4), 2011. Article 11, 1–7. DOI: 10.1145/1964179.1964194.
12. Venkatesh A., Subramoni H., et.al. A high performance broadcast design with hardware multicast and GPUDirect RDMA for streaming applications on InfiniBand clusters // 21st International Conference on High Performance Computing (HiPC), Goa, India, 17-20 Dec. 2014, Proceedings. P. 1-10, 2014 doi: 10.1109/HiPC.2014.7116875.

Аннотации стендовых докладов

Диалект MLIR для общего тайлинга циклов

А.В. Левченко

Суперкомпьютерный центр СПбПУ

Актуальность работы обусловлена необходимостью обобщения операций тайлинга циклов в рамках диалекта многоуровневого промежуточного представления (Multi-Level Intermediate Representation, MLIR) библиотеки LLVM [1]. Особенностью MLIR стала возможность полиэдральной оптимизации циклов с использованием диалекта Affine. Это позволяет моделировать трансформации циклов в задачах выпуклой оптимизации, в том числе и тайлинг, а именно разбиение пространства итераций цикла на атомарно выполняемые операции, тайлы. Новый диалект для тайлинга в библиотеке MLIR позволит существующим диалектам улучшить локальность обращений к глубокой иерархической памяти высокопроизводительных систем. Кроме того, в свете развития фронтенда C и C++ для MLIR [2] такой диалект представляется особенно полезным для оптимизации суперкомпьютерных программ класса трафаретных вычислений (stencils), реализованных на этих языках.

Вкладом данной работы является диалект Common Loop Tiling (CLT). Макроструктура взаимодействия CLT и уже существующих диалектов MLIR продемонстрирована на рис. 1.

Рис. 1. Схема взаимодействия CLT с некоторыми диалектами инфраструктуры MLIR

Диалект CLT объединяет базовые операции, атрибуты и типы, которые требуются для осуществления тайлинга на уровне MLIR. На рис. 1 показан многоуровневый конвейер компиляции, который использует диалекты CLT, Linalg и Affine для генерации тайлового кода. *Входными данными* конвейера является код MLIR вышестоящих предметно-ориентированных диалектов (Domain-Specific Dialects, DSD). Диалект CLT использует операции диалекта Affine для осуществления анализа зависимостей и трансформаций циклов в рамках подхода полиэдральной компиляции. При этом возможно использование библиотеки быстрой арифметики Пресбургера (Fast Presburger Library, FPL) [3]. Диалект CLT решает проблему параметризации операций тайлинга значениями размерности тайлов, предварительно полученными посредством алгоритма поиска размера тайла (Tile Size Selection, TSS) для адаптации диалекта к особенностям целевой архитектуры. *Выходными данными* конвейера является тайловый код для GPU, полученный на основе подхода, ранее предложенного в работе [4]. Диалект определяет операцию `clt.launch{dev,tss,cache}` для создания региона и генерации тайлового кода для целевого устройства через диалекты LLVM и GPU.

Экспериментальные результаты ускорения тайлового кода относительно версии без тайлинга получены с использованием AMD EPYC 7763 и NVIDIA RTX A2000 и представлены на рис. 2. Использованы репрезентативные задачи класса трафаретных вычислений (stencils): FDTD-1D/2D, Jacobi-1D/2D, Seidel-2D. Представлены результаты ускорения тайлового кода только для CPU, полученного через диалект LLVM, а также кода для GPU, полученного через диалекты GPU и NVVM. Результаты параллельных циклов с использованием операции `clt.for` сопоставлены с результатами параметрического тайлинга, где дополнительно уточнены размеры тайлов, заранее полученные из алгоритма TSS, в рамках которого решена задача геометрического программирования для случая многоуровневого тайлинга. Показано, что во всех случаях применение операции `clt.for` с результатами алгоритма TSS привело к заметному ускорению по сравнению с версией кода без тайлинга.

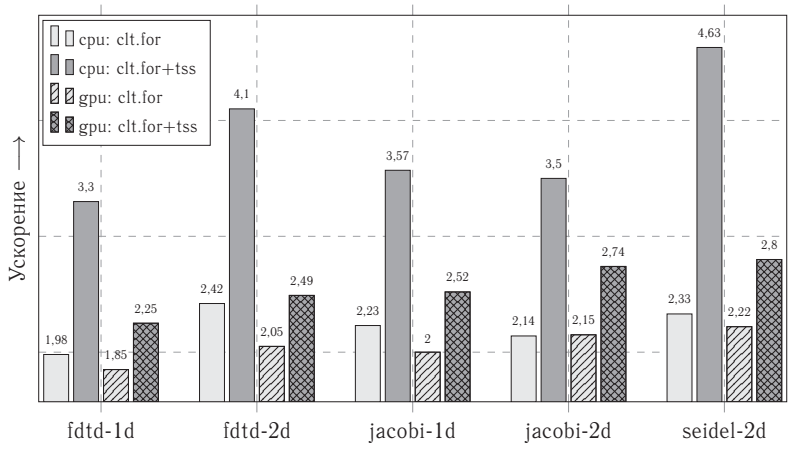


Рис. 2. Ускорение тайлового кода тестовых программ относительно первоначального кода

Выводы о практической значимости предложенного диалекта обоснованы возможностью использования его операций, атрибутов и типов для повышения производительности существующих и будущих языковых и предметно-ориентированных диалектов MLIR. Будущая работа будет сфокусирована на разработке операций, не требующих явного переключения метода тайлинга для получения параллельного тайлового кода для CPU и GPU.

Литература

1. Lattner C., Amini M., Bondhugula U., Cohen A., Davis A., Pienaar J., Riddle R., Shpeisman T., Vasilache N., Zinenko O. MLIR: Scaling Compiler Infrastructure for Domain Specific Computation // Proceedings of the 2021 IEEE/ACM International Symposium on Code Generation and Optimization. Seoul, Korea. February 27 – March 3, 2021. P. 2–14. DOI: 10.1109/CGO51591.2021.9370308

2. Moses W., Chelini L., Zhao R., Zinenko O. Polygeist: Raising C to Polyhedral MLIR // 30th International Conference on Parallel Architectures and Compilation Techniques. Atlanta, USA. September 26–29, 2021. P. 45–59. DOI: 10.1109/PACT52795.2021.00011

3. Pitchanathan A., Ulmann C., Weber M., Hoefler T., Grosser T. FPL: Fast Presburger Arithmetic through Transprecision // Proceedings of the ACM on Programming Languages. October, 2021. Vol. 5. Issue OOPSLA. Article 162. DOI: 10.1145/3485539

4. Gysi T., Müller C., Zinenko O., Herhut S., Davis E., Wicky T., Fuhrer O., Hoefler T., Grosser T. Domain-Specific Multi-Level IR Rewriting for GPU: The Open Earth Compiler for GPU-accelerated Climate Simulation // ACM Transactions on Architecture and Code Optimization. September, 2021. Vol. 18. Issue 4. Article 51. DOI: 10.1145/3469030

Исследование ячеистой структуры детонационной волны в смеси водород-воздух

М.А. Швецова, М.Ю. Мальсагов, Е.В. Михальченко

Московский государственный университет имени М.В. Ломоносова

Детонационная волна в газовых смесях является многомерным комплексом, формируемым ведущей ударной волной и следующей за ней зоной химической реакции [1-2]. По своей природе детонационная волна не является устойчивой, что ограничивает возможности контролируемого использования детонационного горения. Исследование структуры детонационной волны важно при изучении вопросов безопасности, например в водородной энергетики, а так же при разработке новых типов силовых установок. Особенности развития неустойчивости детонационной волны во многом определяются химической кинетикой реагирующей смеси. Корректное воспроизведение параметров горения смеси возможно с применением детальных схем химической кинетики. При этом, численное моделирование с использованием детальных кинетических механизмов требует значительных вычислительных ресурсов, что ограничивает возможности их использования для расчетов на масштабах реальных технических систем.

В данной работе изучается возможность моделирования процессов химической кинетики в химически активных газовых смесях с применением искусственных нейронных сетей. С помощью классических численных методов и детальной схемы окисления смеси водород-воздух GRIMech 3.0[3] были построены наборы обучающих данных. Численная реализация основана на методе Новикова [4] из класса методов Розенброка для жестких систем обыкновенных дифференциальных уравнений. Архитектура нейронной сети представлена на рисунке 1. Входом сети является вектор размерности 11, т.е. состояние химической системы (температура и молярные плотности веществ). Дальше вход передается через несколько одинаковых блоков, состоящих из нескольких слоев. Затем результат по компонентно складывается с входом и передается на выход сети. Каждый блок состоит из трех полносвязных слоев, в первом слое 256 нейронов, второй состоит из 128 нейронов, а третий по размерности входного вектора из 11 нейронов. В качестве функции активации используется одна из параметрических версий ReLU – LeakyReLU, с параметром a = 0.15. Выход третьего полносвязного слоя по компонентно складывается с входом всего блока. Результат передается дальше по основной сети.

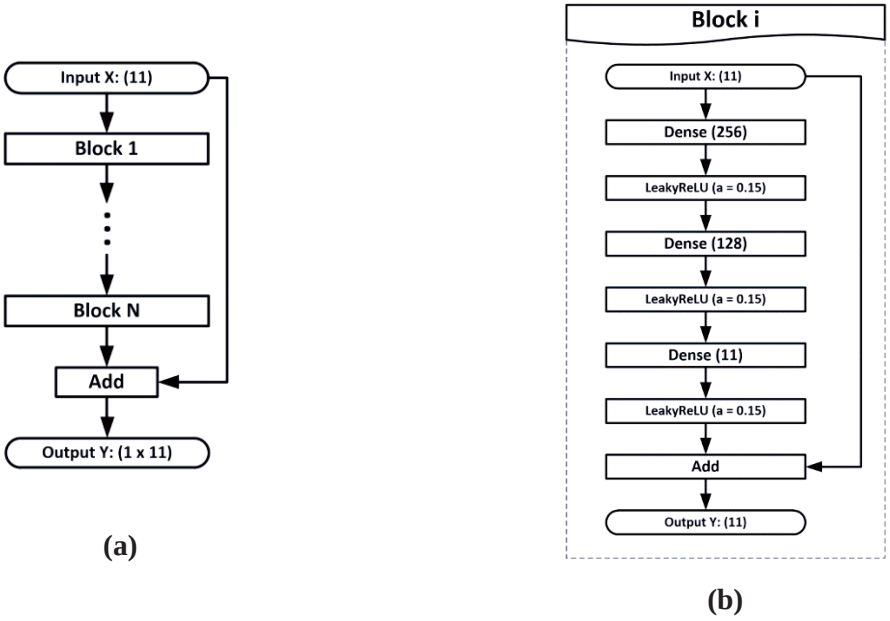


Рис. 1. Архитектура сети: (а) основная модель; (б) одноблочная структура.

Таким образом, получилась нейронная многослойная сеть прямого распространения с обходными связями. Число блоков можно менять, чтобы добиться большей точности и надежности либо скорости работы. Вычислительная сложность блока составляет порядка 74 000 операций. В результате своей работы данная нейронная сеть предсказывает состояние химической системы по предыдущему. Поэтому, если требуется получить динамику развития процесса во времени, необходимо циклично обновлять вход сети, подавая полученный выход на вход. Основным недостатком такого алгоритма является нарастание ошибки. Сеть на каждом шагу выдает результат с некоторым отклонением, неточный результат. Какой бы маленькой ни была бы ошибка, она приводит к тому, что на следующем шаге на вход сети подается искаженный вход. А, значит, выход сети с каждым циклом будет все больше отклоняться от идеального результата. Увеличение числа блоков в сети и остаточная связь позволяют частично бороться с этой проблемой и получать качественный результат приблизительно на 1000 шагов. Полученная нейронная сеть была встроена в расчет задачи моделирования ячеистой структуры детонации в канале.

Благодарности

Работа выполнена за счет субсидии, выделенной ФГУ ФНЦ НИИСИ РАН на выполнение государственного задания № 1021061509701-5-1.2.1 «Разработка алгоритмической компоновки и программ для расчета многомасштабных процессов и горения» (FNEF-2022-0021).

Литература

1. White, D. (1961). Turbulent structure of gaseous detonation. Phys. Fluids 4, 465–480.
2. Denisov, Y. N., and Y. K. Troshin (1959). Pulsating and spinning detonation of gaseous mixtures in tubes. Dokl. Akad. Nauk. 125, 110–113.
3. GRI-Mech Version 3.0 7/30/99 CHEMKINII format, at http://www. me. berke-ley.edu/gri_mech/.
4. E.A. Novikov. Investigation of (m,2)-methods of stiff systems calculation. Vychislitel'nye tekhnologii (Calculation technologies), Vol. 12, 5(2007) 103 - 115

Моделирование вытеснения вязкой жидкости из пористой среды с учетом мелкомасштабной неустойчивости

А.Г. Бароян, Е.И. Скрылева

Московский государственный университет имени М.В. Ломоносова

Для добычи углеводородных полезных ископаемых сегодня используются различные методы, самым распространенным из которых является заводнение - вытеснение углеводорода путем нагнетания в скважину вытесняющих агентов, повышения тем самым градиента давления. При этом на границе раздела фаз вытеснения развивается неустойчивость: вытесняющая жидкость стремится прорваться сквозь слой вытесняемой, формируя в ней каналы, называемые «вязкими пальцами», что существенно сказывается на качестве нефтедобычи. Одним из наиболее эффективных методов изучения фильтрации является численный эксперимент, поскольку уравнения, описывающие фильтрацию, в большинстве случаев не разрешимы аналитическими и приближенными методами. Данный подход намного дешевле и доступнее по сравнению с натурным экспериментом. Основной недостаток известных методов моделирования нелинейных процессов вытеснения углеводородов - моделирование на макроуровне без учета мелкомасштабной неустойчивости, возникающей на фронте вытеснения углеводорода из пласта, из-за чего снижается точность расчетов и увеличивается время их проведения при построении реальной модели прогнозирования протекания процессов в нефтесодержащих пластах, при этом расчёты с достаточно большим разрешением невозможны даже с использованием современной вычислительной техники.

В данной работе описан метод, позволяющий учесть подсеточную неустойчивость, развивающуюся на мелком масштабе при моделировании вытеснения на крупном масштабе. Данный метод позволяет учесть неустойчивость вытеснения даже при одномерном моделировании. Суть метода заключается в разбиении исследуемой геологической структуры на крупномасштабные блоки (КМБ), а затем в разбиении КМБ на мелкомасштабные блоки (ММБ) и вычислении значения физических свойств для ММБ. После чего путем ремасштабирования осуществляют вычисление значения физических свойств для КМБ. Технический результат заключается в изменении модели на макроуровне за счет дополнительных потоков в уравнениях в зависимости от физических свойств, полученных при исследовании процесса на микроуровне.

Для описания используется двухфазная модель флюида в пористой среде, где фазы считаются несмешивающимися и пористость постоянна, уравнение баланса массы для каждой из фаз, закон Дарси, а также уравнение связи между давлениями в фазах через капиллярное давление и дополнительные алгебраические соотношения. Уравнения глобальной системы приведены в размерном виде. Расчеты производились с использованием технологии OpenMP на одном узле (на АПК-5). Всего 12 узлов, на узле 2 процессора по 10 ядер.

В настоящей работе проведена серия численных экспериментов на трехмерной программе для ММБ и выведено осредненное значение насыщенности для разных параметров и разных моментов времени.

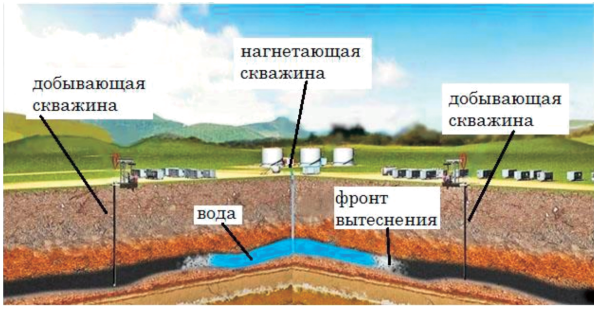


Рис. 1. Технология нефтедобычи: вытеснение нефти водой.

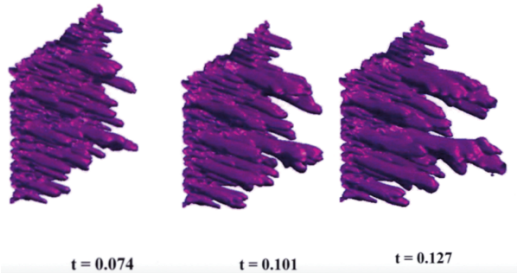


Рис. 2. Численное моделирование в пористой среде.

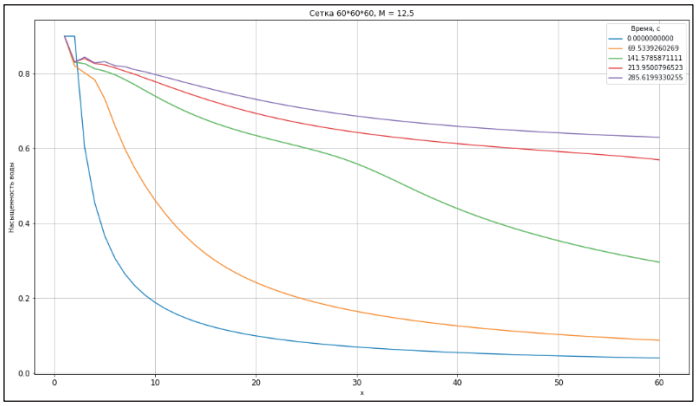


Рис. 3. Осредненное по сечениям значение насыщенности при отношении вязкостей M=12,5.

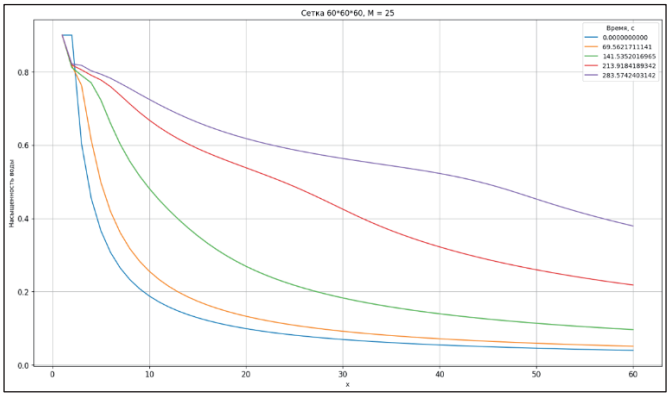


Рис. 4. Осредненное по сечениям значение насыщенности при отношении вязкостей M=25.

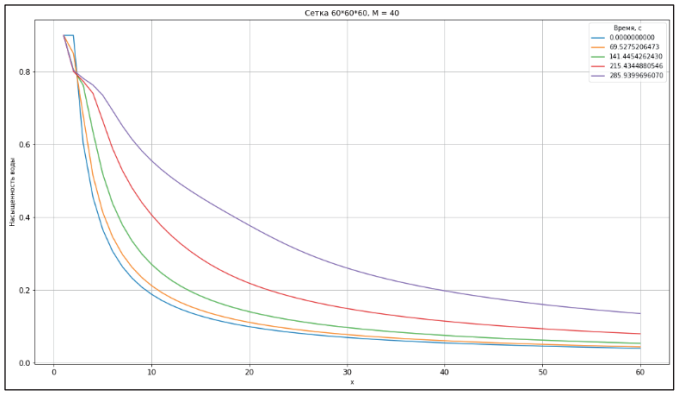


Рис. 5. Осредненное по сечениям значение насыщенности при отношении вязкостей M=40.

Литература

1. Бетелин В.Б., Смирнов Н.Н., Никитин В.Ф., Стамов Л.И., Михальченко Е.В., Тюренкова В.В., Скрылева Е.И. Способ многомасштабного моделирования нелинейных процессов подземной гидродинамики // Патент №2670174 — 18 октября 2018

https://yandex.ru/patents/doc/RU2670174C1_20181018
2. Kaviany, M.: Principles of Heat Transfer in Porous Media. Second Ed. // Springer-Verlag, New York – 1995.
3. Saffman, P.G., Taylor, G.J. The penetration of a fluid into a porous medium of Hele-Shaw cell containing a more viscous fluid // Proc. R. Foc. Zond. – 1958 – A 245,312 http://kinampark.com/PL/files/Saffman%201958,%20The%20penetration%20of%20a%20fluid%20into%20a%20porous%20medium%20or%20Hele-Shaw%20cell%20containing%20more%20viscous%20liquid.pdf

Моделирование структуры подвижных участков белка с помощью GPU-ускоренной метадинамики с коллективными переменными на основе вариационных автоэнкодеров

К.Е. Копылов[1], Е.М. Кирилин[1,2], В.К. Швядас[1,3]

[1]Научно-исследовательский вычислительный центр МГУ имени М.В. Ломоносова,
[2]НИИ ФХБ имени А.Н. Белозерского,
[3]Факультет биоинженерии и биоинформатики МГУ имени М.В. Ломоносова

Фермент 2-гидроксибифенил-3-монооксигеназа [1] из почвенной бактерии *Pseudomonas azelaica* (HbpA) относится к классу А флавин-зависимых монооксигеназ. Изучение этих ферментов представляет фундаментальный интерес из-за особенностей их структуры и механизма действия [2], а также имеет практическую значимость с точки зрения их использования в тонком органическом синтезе для региоспецифического гидроксилирования различных фенольных субстратов [1,3]. Наличие подвижных участков в субстрат-связывающем домене HbpA затрудняет расшифровку его полноатомной структуры методами рентгеноструктурного анализа. Некоторые из этих участков расположены в непосредственной близости от активного центра и могут играть ключевую роль в распознавании, доставке и стабилизации реакционноспособного положения субстрата. В субстрат-связывающем домене HbpA в кристаллографической структуре 5BRT из банка данных PDB [4] описаны подвижные участки, положение которых не удавалось расшифровать методами рентгеноструктурного анализа. Для установления структуры участков Tyr256-Ile266 и Arg228-Val236 предлагается использовать комбинацию методов молекулярного моделирования метадинамики и искусственного интеллекта.

С помощью нейросети Alphafold [5], было предсказано 5 структур с альтернативными положениями петель в активном центре. Также ранее с помощью метода Monte-Carlo моделирования Rosetta [6] были предсказаны варианты положения петель, существенно отличающиеся от полученных с помощью Alphafold. Для расширения репрезентативности набора конформаций петель была запущена «разведывательная» метадинамика из структур Rosetta, обнаружены 273 локальных минимума. Для дальнейшего выяснения энергетического профиля среди полученных конформаций изучаемых участков (рис.1A) был использован метод “хорошо темперированной” метадинамики [7,8], подразумевающий исследование поведения системы путём запуска нескольких параллельных молекулярных динамик с общим потенциалом, которые помогают системе преодолеть потенциальные барьеры за счет постепенного наращивания дополнительного потенциала от выбранных коллективных переменных. Изучаемые участки состоят из 24 остатков (13 остатков одной, 11 остатков другой смежной петли), определяя 48 переменных двугранных углов основной цепи. Исследовать пространство такой большой размерности невозможно в силу ограниченности вычислительных ресурсов.

В качестве разумной модели предложено понижение размерности пространства коллективных переменных (рис.1B) до трёхмерного с помощью вариационного автоэнкодера [10,11], обученного на датасете траекторий «разведывательной» метадинамики данного фермента и набора предсказанных конформаций Alphafold и Rosetta (примерно 2 мкс траекторий). Для обучения была использована метрика ошибки вариационного автоэнкодера, состоящая из суммы кумулятивной функции реконструкции данных (косинусное расстояние) и дивергенции Кульбака-Лейблера апостериорного распределения из предположения гиперсферической природы латентного пространства [12]. Модель PyTorch была присоединена к Plumed с помощью интерфейса [13], GPU-ускоренная метадинамика запущена в пространстве эмбеддингов для исследования поверхности потенциальной энергии, действуя на исходное пространство путём вычисления градиентов с помощью функции PyTorch autograd. В результате вычислений удалось выявить наиболее стабильные конформации петель в структуре фермента и

определить их динамическое поведение, а также предположить роль в механизме открытия/закрытия активного центра.

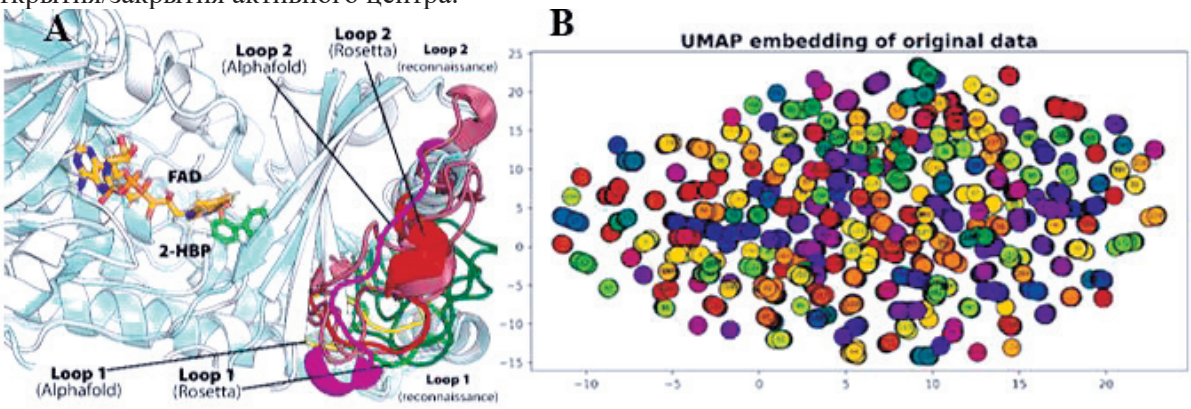


Рис. 1. (**A**) Положения подвижных петель активного центра, предсказанные различными моделями (**B**) Визуализация пространства коллективных переменных (двугранных углов) с помощью UMAP [9]. Раскраска соответствует различным кластерам – локальным минимумам.

Работа Е.М.К. и В.К.Ш. выполнена при финансовой поддержке РНФ (грант № 21-71-30003) и Некоммерческого Фонда развития науки и образования «Интеллект» (стипендия К.Е.К. № 4/1-12-НС-СОТ-11/2022).

Литература

1. Bregman-Cohen A. et al. Altering 2-Hydroxybiphenyl 3-Monooxygenase Regioselectivity by Protein Engineering for the Production of a New Antioxidant: 6 // ChemBioChem. 2018. Vol. 19, № 6. P. 583–590.
2. Kopylov K., Kirilin E., Švedas V. Conformational transitions induced by NADH binding promote reduction half-reaction in 2-hydroxybiphenyl-3-monooxygenase catalytic cycle // Biochem. Biophys. Res. Commun. 2023. Vol. 639. P. 77–83.
3. Lutz J. et al. Preparative application of 2-hydroxybiphenyl 3-monooxygenase with enzymatic cofactor regeneration in organic-aqueous reaction media // J. Mol. Catal. B Enzym. 2002. Vol. 19. P. 177–187.
4. Kanteev M. et al. A crystal structure of 2-hydroxybiphenyl 3-monooxygenase with bound substrate provides insights into the enzymatic mechanism: 12 // Biochim. Biophys. Acta BBA-Proteins Proteomics. 2015. Vol. 1854, № 12. P. 1906–1913.
5. Jumper J. et al. Highly accurate protein structure prediction with AlphaFold // Nature. 2021. Vol. 596, № 7873. P. 583–589.
6. Stein A., Kortemme T. Improvements to robotics-inspired conformational sampling in rosetta: 5 // PLoS One. 2013. Vol. 8, № 5. P. e63090.
7. Tribello G.A. et al. PLUMED 2: New feathers for an old bird: 2 // Comput. Phys. Commun. 2014. Vol. 185, № 2. P. 604–613.
8. Drobot V.V. et al. PLUMED plugin integration into high performance pmemd program for enhanced molecular dynamics simulations // Supercomput. Front. Innov. 2021. Vol. 8, № 4. P. 94–99.
9. McInnes L., Healy J., Melville J. Umap: Uniform manifold approximation and projection for dimension reduction // ArXiv Prepr. ArXiv180203426. 2018.
10. Alam F.F., Shehu A. Variational Autoencoders for Protein Structure Prediction // Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics. 2020. P. 1–10.
11. Alam F.F., Shehu A. Data Size and Quality Matter: Generating Physically-Realistic Distance Maps of Protein Tertiary Structures // Biomolecules. 2022. Vol. 12, № 7. P. 908.
12. Davidson T.R. et al. Hyperspherical variational auto-encoders // ArXiv Prepr. ArXiv180400891. 2018.

13. Bonati L., Rizzi V., Parrinello M. Data-driven collective variables for enhanced sampling // J. Phys. Chem. Lett. 2020. Vol. 11, № 8. P. 2998–3004.

Модификация и оптимизация библиотеки обмена данными на параллельных вычислительных системах для моделей Земной системы

О.А. Имеев[1,2], Е.М. Гащук[2,3,4], А.В. Дебольский[4,5], Е.В. Мортиков[4,2]

[1]Московский государственный университет им. М.В.Ломоносова, факультет Вычислительной математики и кибернетики,
[2]Институт вычислительной математики им. Г.И. Марчука РАН,
[3]Московский государственный университет им. М.В.Ломоносова, Механико-математический факультет,
[4]Московский государственный университет им. М.В. Ломоносова, Научно-исследовательский вычислительный центр,
[5]Институт физики атмосферы им. А.М. Обухова РАН

Климатическое моделирование является эффективным инструментом для изучения процессов, определяющих климат Земли и погодных явлений. Проведение подобных численных экспериментов является вычислительно сложной задачей, требующей решения большого числа систем уравнений, которые представляют процессы и взаимодействия Земной системы. Они включают в себя общую циркуляцию атмосферы и океана, тепло- и массо- обмен с сушей, покрытые льдом регионы планеты. Для решения уравнений используются трехмерные расчетные сетки, число ячеек в которых может достигать десятков или сотен миллионов, причем данные вычисления проводятся многократно, что может занимать десятки и сотни расчетных часов. Таким образом, климатическое моделирование является сложной задачей для одного ядра CPU (Central Processing Unit), в связи с чем для проведения таких расчетов используются массивно-параллельные вычислительные системы. Также проведение численного моделирования осложняется необходимостью множественного обмена данными между MPI-процессами, как внутри программных блоков, отвечающих за отдельные процессы Земной системы, так и между ними, что может сильно повлиять на время расчетов. В данной работе мы представляем описание программной реализации библиотеки обмена данными Parlib [1] на параллельных вычислительных системах для климатической модели INMCM, разработанной в Институте Вычислительной Математики (ИВМ) РАН [2].

Библиотека написана на языке C, но существует поддержка вызова функций с помощью Fortran. На данный момент она поддерживает два вида функций: обмен граничными условиями (необходимо при параллельной реализации конечно-разностных методов) и транспонирование данных (перераспределение данных между MPI-процессами, используемое для реализации быстрого преобразования Фурье).

Для упрощения работы с программной реализацией и внедрения нового функционала ядро библиотеки будет реализовано на языке C++. Первым этапом модификаций является адаптация библиотеки для использования гибридного MPI-OpenMP подхода с поддержкой вычислений на графических процессорах. На данный момент в общей модели Земной системы только ядро океана поддерживает использование технологии OpenMP, а вся модель не поддерживает вычисления на GPU. С учетом того, что планируется адаптация модели Земной системы под вычисления на HPC-системах гибридной архитектуры, в рамках обновления библиотеки будет добавлен соответствующий функционал. Кроме того, будут применены некоторые техники оптимизации, в частности, методы сжатия данных, что является распространенной практикой в решении задачи ускорения обменов данными между MPI-процессами. Так, в работе [3] продемонстрирована техника по оптимизации MPI-обменов, которая использует интерфейс PMPI профилирования MPI для применения сжатия (и распаковки) без потерь во время выполнения, уменьшая объем сообщений, что дает максимальное ускорение в 1.7 раз всей численной модели расчета свойств полупроводниковых элементов [4]. В случае вычислений на

GPU (Graphics Processing Unit) накладные расходы на копирование данных между CPU и GPU могут значительно увеличить время обмена. В новейших архитектурах графических ускорителей NVIDIA возможно реализовать передачу данных напрямую между GPU с помощью таких технологий, как NVLink, PCIe и GPU Direct Peer to Peer. Таким образом, еще один подход к ускорению, который будет реализован, заключается в исключении CPU из выполнения обмена данными, когда вычисления проводятся на графических ускорителях.

Исследование выполнено в рамках гранта молодежной лаборатории «Суперкомпьютерные технологии математического моделирования Земной системы» (Соглашение № 075-03-2023-509/1 с Минобрнаукой России).

Литература

1. Gloukhov, V. "Parallel Computations in Problems of Climate Modeling." Parallel Computational Fluid Dynamics 2003. Elsevier. 301-308 (2004).
2. Volodin, E.M., Mortikov, E.V., Kostrykin, S.V. et al. “Simulation of modern climate with the new version of the INM RAS climate model.” Izv. Atmos. Ocean. Phys. 53, 142–155 (2017)
3. Filgueira, R., Atkinson, M., Nuñez, A., Fernández, J. An Adaptive, Scalable, and Portable Technique for Speeding Up MPI-Based Applications. In: Kaklamanis, C., Papatheodorou, T., Spirakis, P.G. (eds) Euro-Par 2012 Parallel Processing. Euro-Par 2012. Lecture Notes in Computer Science, vol 7484. Springer, Berlin, Heidelberg. (2012).
4. Loureiro, A., Gonz´alez, J., Pena, T.F. "A parallel 3D semiconductor device simulator for gradual heterojunction bipolar transistors." Int. Journal of Numerical Modelling: Electronic Networks, Devices and Fields 16, 53–66 (2003).

Обобщаемость нейросетевых моделей для предсказания энергии атомизации молекул

В.М. Волохов[1], И.И. Акостелов[1], Е.С. Амосова[1], Д.Б. Лемперт[1], В.В. Парахин[2], А.С. Лазаренко[1]

[1]ФИЦ ПХФ и МХ РАН
[2]ИОХ РАН

Установление связи между структурой и свойством является одной из наиболее важных тем химии. Наиболее общее решение этой задачи дает квантовая механика, но численное решение уравнения Шредингера для большинства химических систем требует слишком большого количества вычислительных ресурсов. Поэтому, в последние годы всё большую популярность приобретают подходы, направленные на предсказание различных квантовомеханических характеристик молекулярных систем с помощью графовых нейронных сетей. [1]

Несмотря на значительное увеличение точности таких моделей за последнее время, их использование для предсказания свойств новых молекул всё еще затруднительно. В большинстве работ, описывающих новые архитектуры нейронных сетей, обучение и тестирование моделей проводится на наборе данных QM9 [2-3], а точность на других наборах данных не исследуется. Это представляет проблему, так как QM9 не является достаточно хорошим представлением пространства химических веществ; например, в QM9 отсутствуют соединения с алифатическими нитрогруппами.

В представленной работе исследовалась зависимость обобщаемости нейросетевых моделей от набора данных, на котором проходило обучение. В качестве архитектуры нейронной сети была выбрана PaiNN [4], обучение моделей проводилось на наборах данных QM9 [2-3], OD9 [5], PC9 [6], Alchemy [7], а также нескольких их подмножествах и объединениях. Предсказывающая способность полученных моделей была протестирована на четырех разнообразных небольших группах молекул, подобранных вручную.

Таблица 1. Точность предсказания энергии атомизации на валидационном наборе (средняя абсолютная ошибка в ккал\моль) в зависимости от тренировочного набора данных

PC9	QM9	Alchemy	ALQM9	OD9	OD9 (0)	OD9 (1)
0,286	0,169	0,326	0,297	0,312	0,218	0,454

Таблица 2. Точность предсказания энергии атомизации различных групп веществ (средняя абсолютная ошибка в ккал\моль) в зависимости от тренировочного набора данных

	Азокси	Высокоэнерг.	Нейротрансм.	Сахара
PC9	3,76	3,03	0,40	0,29
QM9	23,70	21,28	9,80	6,66
Alchemy	95,21	66,59	55,86	48,13
ALQM9	190,74	64,93	66,18	56,26
OD9	65,76	1,53	0,96	0,38
OD9 (0)	51,14	26,24	6,49	13,33
OD9 (1)	181,0	78,63	1,75	18,84

Точность модели при обучении и валидации на QM9 составляет 0,169 ккал/моль, но при этом на тестовых группах эта модель демонстрирует ошибки примерно в 100 раз больше. Точность модели при обучении и валидации на PC9 составляет примерно 0,286 ккал\моль и этот результат более удовлетворительно согласуется с ошибками на тестовых наборах. Остальные наборы данных демонстрируют меньшую точность.

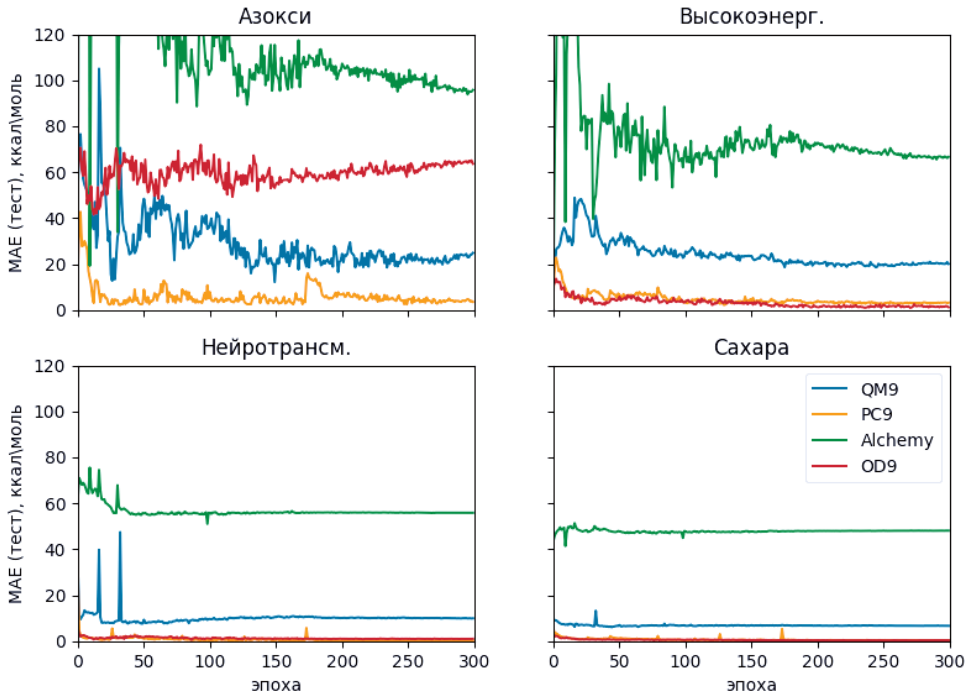


Рис. 1. Кривые точности предсказания энергии атомизации различных групп веществ в зависимости от тренировочного набора данных.

Детальное описание методов и тестовых наборов данных будет опубликовано в дальнейшем. Во всех моделях в качестве предсказываемого параметра использовалась энергия атомизации без учета колебательной энергии нулевого уровня, что связано с отсутствием вычислений частот колебаний в PC9. ALQM9 является объединением Alchemy и QM9. Описание подмножеств OD9 (0) и OD9 (1) приведено в оригинальной работе.

Проведенное исследование можно дополнить, исследовав, как зависит обобщаемость от архитектуры нейронной сети. Также, важным направлением работы является развитие разнообразных тестовых наборов молекул для оценки обобщаемости моделей.

В работе показано, что обучение на PC9 позволяет получить наиболее переносимую модель. Этот набор данных может быть использован для обучения нейросетевых моделей, способных к предсказанию свойств новых молекул. Перспективным является перерасчёт его более точным квантовохимическим методом (например, G4MP2), что позволит использовать его для точного предсказания энтальпии образования, востребованного в области высокоэнергетических веществ.

Исследование выполнение при финансовой поддержке РНФ в рамках научного проекта №23-71-00005.

Литература

1. J. Gilmer, S. S. Schoenholz, P. F. Riley et al. Neural message passing for quantum chemistry // International Conference on Machine Learning, PMLR 70:1263-1272, 2017.
2. L. Ruddigkeit, R. van Deursen, L. C. Blum, J.-L. Reymond, Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17, J. Chem. Inf. Model. **52**, 2864–2875, 2012. DOI: 10.1021/ci300415d
3. R. Ramakrishnan, P. O. Dral, M. Rupp, O. A. von Lilienfeld, Quantum chemistry structures and properties of 134 kilo molecules, Scientific Data 1, 140022, 2014. DOI: 10.1038/sdata.2014.22
4. K. T. Schütt, O. T. Unke, M. Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra // International Conference on Machine Learning, PMLR 139:9377-9388, 2021.

5. Leguy, J., Glavatskikh, M., Cauchy, T. et al. Scalable estimator of the diversity for de novo molecular generation resulting in a more robust QM dataset (OD9) and a more efficient molecular optimization. J Cheminform 13, 76 (2021). DOI: 10.1186/s13321-021-00554-8.
6. Glavatskikh, M., Leguy, J., Hunault, G. et al. Dataset's chemical diversity limits the generalizability of machine learning predictions. *J Cheminform* 11, 69 (2019). DOI: 10.1186/s13321-019-0391-2
7. Chen, G., Chen, P., Hsieh, C.Y et al. (2019). Alchemy: A Quantum Chemistry Dataset for Benchmarking AI Models. *arXiv preprint arXiv:1906.09427*.

Параллельный алгоритм глобальной оптимизации с монотонным преобразованием целевой функции*

К.А. Баркалов, А.С. Кудрявцев

Нижегородский государственный университет им. Н.И. Лобачевского

В работе рассматривается параллельный алгоритм для минимизации многоэкстремальных функций, удовлетворяющих условию Липшица с априори неизвестной константой. Предположение липшицевости целевой функции является типичным для многих подходов к разработке алгоритмов глобальной оптимизации [1, 2]. В отличие от широко распространенных метаэвристических алгоритмов, методы липшицевой глобальной оптимизации обладают детерминированным поведением и обеспечивают более быструю сходимость к глобальному минимуму [3].

Задача рассматривается в следующей постановке:

$$\varphi(y^*) = \min_{y \in D} \varphi(y), \qquad D = \{y \in R^N : a_i \le y_i \le b_i,\ i = \overline{1,N}\}.$$

Используемый в работе подход основан на редукции исходной многомерной задачи к эквивалентной ей системе одномерных подзадач с последующим решением одномерных задач эффективными методами оптимизации функций одной переменной. Подобная редукция может быть выполнена в соответствии с соотношением

$$\min_{y \in D} \varphi(y) = \min_{y_1 \in [a_1,b_1]} \min_{y_2 \in [a_2,b_2]} \dots \min_{y_N \in [a_N,b_N]} \varphi(y_1, \dots, y_N),$$

порождающим множество рекурсивно связанных одномерных подзадач.

Ранее был предложен подход адаптивного выбора решаемой в данный момент подзадачи [4, 5]. Для этого каждой подзадаче присваивается некоторое числовое значение, называемое *характеристикой*. Считается, что чем выше значение характеристики, тем более «перспективной» является подзадача для продолжения вычислений, и поэтому на каждой итерации выбирается подзадача с максимальной характеристикой для проведения в ней очередного *испытания* (вычисления значения целевой функции). Это испытание либо вычисляет значение целевой функции $\varphi(y)$ (если выбранная «лучшая» подзадача принадлежала уровню $j = N$), либо порождает новые подзадачи согласно при $j \le N - 1$. В последнем случае новые порожденные задачи добавляются к множеству подзадач, вычисляются их характеристики и процесс повторяется. Завершение многомерной оптимизации происходит, когда в корневой задаче уровня $j = 1$ выполняется условие остановки.

Одним из «узких мест» методов глобальной оптимизации является накопление точек поисковых испытаний в окрестности локальных экстремумов, если целевая функция в них обладает гладкостью. Снизить число поисковых испытаний позволяет следующий прием. Вместо исходной целевой функции $\varphi(y)$ будем минимизировать преобразованную функцию

$$\phi(y) = \sqrt{1 - \left(\frac{z_k^+ - \varphi(y)}{z_k^+ - z_k^-}\right)^2},$$

где $z_k^+ = \max_{1 \le i \le k} \varphi(y^i)$, $z_k^- = \min_{1 \le i \le k} \varphi(y^i)$. Указанное преобразование, изменяющееся в процессе работы алгоритма, меняет вид целевой функции, но сохраняет расположение точки решения задачи. При этом алгоритм приобретает свойства *почти монотонной сходимости* к точке глобального минимума в ее малой окрестности.

Распараллеливание метода было организовано следующим образом: на каждой итерации алгоритма определяется p лучших подзадач (в соответствии с их характеристиками), в каждой из которых параллельно проводится одно поисковое испытаний. Указанный способ ориентирован на распараллеливание самой трудоемкой части – вычисления значений целевой функции.

* Исследование выполнено при поддержке гранта Российского научного фонда № 21-11-00204, https://rscf.ru/project/21-11-00204/

Программная реализация алгоритма выполнена на C++ с использованием OpenMP. Проведены эксперименты на суперкомпьютере «Лобачевский», наглядно демонстрирующие эффект ускорения сходимости алгоритма. В экспериментах был использован узел суперкомпьютера с процессором AMD EPYC 7742 2.25 GHz (64 ядра).

Были решены 200 тестовых многоэкстремальных задач классов *Simple* и *Hard*, которые для имитации вычислительной трудоемкости, присущих прикладным задачам, были усложнены дополнительными вычислениями, не меняющими характер поведения функции и точку ее глобального минимума (умножением матриц размера 256×256). Вычисления, связанные с матричными преобразованиями, являются типичными для многих прикладных задач.

Таблица 1. Среднее число итераций K_{av}, время работы T_{av} и ускорение S_{av} параллельного алгоритма при решении задач классов *Simple* и *Hard*.

p	Класс *Simple*			Класс *Hard*		
	K_{av}	T_{av}	S_{av}	K_{av}	T_{av}	S_{av}
1	302.4	6.61	1.0	702.1	15.25	1.0
4	80.6	1.85	3.6	181.6	4.17	3.7
8	47.3	1.11	6.0	96.7	2.26	6.8
32	20.0	0.52	12.8	41.8	1.07	14.3
64	12.2	0.36	18.6	22.3	0.63	24.3

Результаты, приведенные в таблице, показывают наличие хорошего ускорения при небольшом числе потоков (до 8). При увеличении числа потоков наблюдается снижение эффективности распараллеливания. Одновременно с этим при решении сложных задач достигается большее ускорение, чем при решении простых задач.

Литература

1. Evtushenko Y.G., Posypkin M.A. A Deterministic Approach to Global Box-Constrained Optimization // Optim. Letters. 2013. vol. 7(4), P. 819–829. DOI: 10.1007/s11590-012-0452-1
2. Квасов Д.Е., Сергеев Я.Д. Методы липшицевой глобальной оптимизации в задачах управления // Автоматика и телемеханика. 2013. № 9, С. 3–19.
3. Sergeyev Y., Kvasov D., Mukhametzhanov M. On the Efficiency of Nature-Inspired Metaheuristics in Expensive Global Optimization with Limited Budget // Scientific Reports. vol. 8(1). Art. No. 435. DOI: 10.1038/s41598-017-18940-4.
4. Gergel V., Grishagin V., Gergel A. Adaptive nested optimization scheme for multidimensional global search // J. Glob. Optim., 2016. vol. 66(1), P. 35–51. DOI: 10.1007/s10898-015-0355-7.
5. Стронгин Р.Г., Гергель В.П., Баркалов К.А. Адаптивная глобальная оптимизация на основе блочно-рекурсивной схемы редукции размерности // Автоматика и телемеханика. 2020. № 8, 136–148. DOI: 10.31857/S0005231020080103.

Тестирование производительности процессоров архитектуры RISC-V*

В.Д. Волокитин, Е.А. Козинов, В.Д. Кустикова, А.В. Линев, И.Б. Мееров

Нижегородский государственный университет им. Н. И. Лобачевского

Разработка новых процессорных архитектур является одним из двигателей прогресса в области вычислительной техники. Казалось бы, зачем нужны новые архитектуры, если сообщество располагает широким спектром разных решений, к тому же достаточно давно прошедших сложный путь от первых прототипов до полноценно работающих серийных изделий? Этот вопрос допускает разные ответы. Во-первых, необходимо принимать во внимание, что принципиальные прорывы редко происходят в результате инкрементальных улучшений в рамках старых идей и парадигм. Во-вторых, закрытость и коммерческая принадлежность имеющихся архитектур (x86, ARM и др.) ведет к определенным сложностям и рискам. Поэтому представленный более 10 лет назад в Калифорнийском университете в Беркли проект новой, основанной на концепции RISC (reduced instruction set computer) [1] открытой и расширяемой архитектуры RISC-V [2, 3] представляет интерес.

Всего за 12 лет разработчикам процессоров и системного программного обеспечения удалось пройти большой путь от первых инженерных образцов до пока еще маломощных, но вполне работоспособных процессоров, выпускаемых серийно и доступных для приобретения и использования. Конечно, производительность существующих RISC-V процессоров далека не только от серверных, но даже от мобильных CPU архитектур x86 и ARM, однако прогресс в данной области идет значительными темпами. Вряд ли кто-либо возьмет на себя смелость предсказать, когда будет создан первый высокопроизводительный RISC-V процессор, но перспективы выглядят вполне реальными, что подтверждается и текущими анонсами разработчиков, и вложениями мировых лидеров индустрии (например, Intel), и возрастающим интересом научного сообщества [4].

В данном постерном докладе мы анализируем производительность двух доступных на текущий момент RISC-V процессоров при решении задач, в которых работа с памятью является главным фактором, влияющим на время работы программы. Наш основной интерес состоит в том, чтобы оценить текущие возможности и будущие перспективы и дать ответ на следующие основные вопросы:

1. Какие существуют возможности по адаптации имеющегося системного программного обеспечения для работы на процессорах RISC-V архитектуры и какие усилия необходимы для реализации этих возможностей?
2. Какие показатели производительности применительно к подсистеме памяти доступных RISC-V устройств (пропускная способность) достижимы на стандартных тестах (бенчмарках), широко применявшихся ранее на архитектурах x86 и ARM?
3. Как соотносятся достижимые показатели производительности RISC-V устройств с пиковыми характеристиками, указанными производителями оборудования?
4. В какой степени стандартные, хорошо себя зарекомендовавшие техники оптимизации работы с памятью применимы для повышения производительности программ на доступных RISC-V устройствах?

Для получения первых ответов на поставленные вопросы мы провели эксперименты на следующих RISC-V устройствах:

1. MangoPi MQ-Pro (D1) с процессором Allwinner D1 (1xXuanTie C906, 1ГГц) и 1ГБ оперативной памяти DDR3L. ОС Ubuntu 22.10 (RISC-V) и компилятор GCC версии 12.2.

* Исследование выполнено при поддержке программы академического лидерства ННГУ "Приоритеты-2030".

2. StarFive VisionFive (v1) с процессором StarFive JH7100 (2xStarFive U74, 1ГГц) и 8ГБ LPDDR4. ОС Ubuntu 22.10 (RISC-V) и компилятор GCC версии 12.2.

Для того, чтобы иметь возможность сопоставлять результаты, мы провели тот же набор экспериментов на устройстве Raspberry Pi и серверном процессоре Intel Xeon со следующей конфигурацией окружения:

1. Raspberry Pi 4 model B с процессором Broadcom BCM2711 (4xCortex-A72, до 1.8ГГц) и 4ГБ LPDDR4. В качестве операционной системы использовалась ОС Ubuntu 20.04, код собран компилятором GCC версии 9.4.
2. Сервер с 2 x Intel Xeon 4310T (2x10 ядер Ice Lake, до 3.4ГГц) с 64ГБ оперативной памяти (DDR4). В качестве операционной системы использовалась CentOS 7, а также компилятор GCC версии 9.5. Использовались только ядра одного процессора во избежание возникновения NUMA-эффектов.

В начале мы убедились в работоспособности программно-аппаратного окружения. В целом надо отметить, что несмотря на пока еще имеющиеся сложности с документацией, надежностью доступного ПО и отсутствием большого сообщества разработчиков, способного быстро дать ответ на любой возникающий вопрос, существующие возможности сборки и запуска ПО достаточны, по крайней мере, для опытных специалистов.

Далее мы собрали и запустили STREAM-бенчмарк, повсеместно используемый для определения пропускной способности подсистемы памяти. Результаты экспериментов ожидаемо показали значительное отставание от рассматриваемых устройств на ARM и x86_64. Так, на устройствах MangoPi MQ-Pro и StarFive VisionFive получилось достигнуть 2.5ГБ/сек и 550МБ/сек соответственно, в том время как на Raspberry Pi удалось показать 4ГБ/сек, а на сервере с процессорами Intel Xeon – 35ГБ/сек. В случае же скорости работы кэша текущие устройства на RISC-V показали отставание до 5 раз от аналога на ARM при расчете производительности на ядро. Полученные результаты хотя и не являются конкурентными в сравнении с ARM и, тем более, серверным x86, однако показывают хороший потенциал для дальнейшего развития.

В заключение мы изучили влияние подходов к повышению производительности программ при помощи более рациональной работы с памятью на время вычислений в алгоритме транспонирования плотной матрицы. Были рассмотрены следующие редакции алгоритма: «наивный» алгоритм, параллельная версия, блочная версия и вариант с ручным кэшированием. Эксперименты показали, что все оптимизации работы с памятью, ранее успешно опробованные на x86, работают на представленных устройствах на архитектуре RISC-V. При этом самые оптимальные варианты программы на устройстве StarFive VisionFive показывает большую *относительную производительность* (по отношению *к достигнутой пропускной способности оперативной памяти*), чем у архитектуры ARM, а сами показатели *относительной производительности* сопоставимы с представленным сервером на Intel Xeon.

В постерном докладе будут представлены и подробно прокомментированы результаты описанных выше экспериментов. В целом, основной вывод на текущий момент заключается в перспективности открытой архитектуры RISC-V для дальнейшего развития в сторону высокопроизводительных вычислений, что соответствует текущим результатам других исследователей, запланировавших проведении первого Воркшопа по HPC на RISC-V на самой крупной европейской конференции по суперкомпьютерным технологиям ISC High Performance (https://riscv.epcc.ed.ac.uk/community/isc23-workshop).

Литература

1. Furber S. B. VLSI RISC architecture and organization. – CRC Press, 1989. – Т. 56.
2. Asanović K., Patterson D. A. Instruction sets should be free: The case for risc-v //EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-146. – 2014.
3. Waterman A. et al. The RISC-V instruction set manual //Volume I: User-Level ISA', version. – 2014. – Т. 2.
4. Asanović K. Advancing HPC with RISC-V // Invited talk at Supercomputing, 15.11.2022.

Ускорение численного моделирования электромагнитных каскадов при моделировании плазмы методом частиц в ячейках*

В.Д. Волокитин[1], А.В. Башинов[2], А.А. Муравьев[2], Е.С. Ефименко[2], И.Б. Мееров[1]

[1] Нижегородский государственный университет им. Н. И. Лобачевского
[2] Институт прикладной физики РАН

Численное моделирование квантово-электродинамических (КЭД) процессов в сверхсильных электромагнитных полях [1] является важным этапом на пути к реальным физическим экспериментам на современных сверхмощных лазерных комплексах, которые планируется ввести в действие в ближайшие годы. КЭД-процессы характеризуются существенной нелинейностью, поэтому их моделирование является сложной задачей, требующей эффективного использования суперкомпьютеров. Метод частиц в ячейках является классическим методом моделирования физики плазмы, широко применяющимся для разработки программных комплексов, позволяющих учитывать КЭД процессы в задачах взаимодействия лазерного излучения с веществом. При решении вычислительно трудоемких трехмерных задач данный метод оперирует двумя наборами данных: значениями электромагнитного поля, заданного на трехмерной прямоугольной сетке, и ансамблем макрочастиц, характеризующихся своими параметрами – координатами, импульсами и статистическими весами макрочастиц. Шаг по времени в базовом цикле метода устроен следующим образом. В результате интегрирования уравнений Максвелла вычисляются сеточные значения электромагнитного поля в следующий момент времени. Эти значения интерполируются в соответствии с координатами расположения макрочастиц, далее для каждой из макрочастиц интегрируются уравнения движения, вычисляются новые координаты и скорости. После этого производится вычисление токов, возникающих в результате движения заряженных частиц, и интерполяция подсчитанных значений токов на узлы сетки. Далее этапы базового цикла повторяются. Отсутствие непосредственного взаимодействия между макрочастицами и пространственная локальность взаимодействий «частица–сетка» обуславливают потенциал распараллеливания данного метода, многократно подтвержденный при использовании программных комплексов EPOCH, OSIRIS, PICADOR, VLPL, WARP-X и др.

Одним из наиболее трудоемких для моделирования КЭД-процессов является КЭД-каскад – цепная реакция, включающая излучение гамма-фотонов электронами (позитронами) и распад гамма-фотонов на электрон-позитронные пары в сверхсильном лазерном поле [9] (рис. 1). Моделирование КЭД-каскадов имеет ряд особенностей, влияющих на ход расчетов и балансировку нагрузки. В связи с вероятностной природой квантовых процессов и нелинейной зависимостью вероятности процессов от величины и структуры полей необходимо учитывать возможность рождения новых частиц преимущественно в областях высоких значений интенсивности электромагнитного поля (на рис. 1 электроны и позитроны сконцентрированы в областях наиболее сильного электрического поля, где гуще и ярче представлены силовые линии). Это может привести к экспоненциальному росту числа частиц в локальных регионах моделирования, и, как следствие, к истощению оперативной памяти и к дисбалансу вычислительной нагрузки. Первая проблема решается за счет объединения или прореживания ансамбля макрочастиц [2], а вторая – за счет использования специальных схем подразбиения ячеек [3]. Однако, существует третья проблема – значительная разница временных масштабов плазменных процессов и КЭД-процессов. В используемых сейчас схемах моделирования КЭД-каскадов [1, 4, 5, 6] приходится сильно подразбивать шаг по времени, что существенно замедляет расчёт.

* Исследование выполнено при поддержке Министерства науки и высшего образования России, проект № FSWR-2023-0034. Авторы благодарят Ю. Магнуссона за помощь в апробации и А. Гоноскова за полезные обсуждения и внимание к работе. Эксперименты проведены на СК МСЦ РАН и ННГУ.

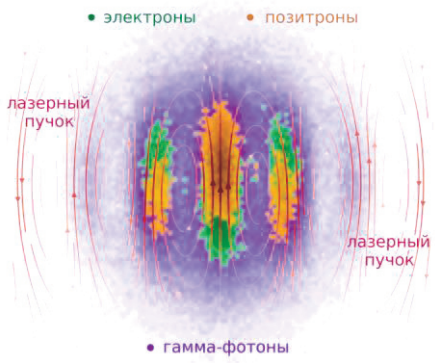


Рис. 1. Лавинообразное рождение электрон-позитронных пар и гамма-фотонов в результате развития КЭД-каскада в сверхсильном лазерном поле. Электрическое поле лазерного излучения представлено в форме силовых линий.

Ранее мы, отталкиваясь от результатов, полученных в работе [4], разработали несколько способов ускорения расчетов. Были улучшены аналитические оценки величин, используемых при анализе вероятностей рождения новых макрочастиц, а также предложены более эффективные реализации аппроксимаций синхротронных функций [7]. Это привело к значительному ускорению, но не решило принципиальную проблему необходимости подразбиения временного шага. В данном докладе рассказывается о новом методе решения этой проблемы. Суть метода заключается в том, что в отличие от традиционных подходов, вычисляющих вероятность рождения макрочастиц на каждом временном шаге, мы определяем момент наступления этого события аналитически с использованием численных аппроксимаций. Выигрыш достигается за счет того, что в этом случае исключается необходимость подразбиения временного шага и последующих расчетов с малым шагом по времени, а применение аппроксимаций исключает необходимость многократного вычисления синхротронных функций. Результаты вычислений с помощью нового метода согласуются с результатами, полученными ранее нами и другими исследователями, при этом вычисления могут ускоряться на порядок в зависимости от задачи.

Аналитические выкладки, описание численных аппроксимаций, результаты тестирования и апробации разработанного подхода при решении задач изложены нами в препринте [7]. В рамках данного постерного доклада мы концентрируемся на анализе производительности ПО, разработанного на языке C++ с использованием технологий OpenMP и MPI, включенного в программный комплекс PICADOR и открыто доступного в репозитории проекта hi-χ [8].

Литература

1. Gonoskov A. et al. Extended particle-in-cell schemes for physics in ultrastrong laser fields: Review and developments // Physical review E. – 2015. – Т. 92. – №. 2. – С. 023305.
2. Muraviev A. et al. Strategies for particle resampling in PIC simulations // Computer Physics Communications. – 2021. – Т. 262. – С. 107826.
3. Meyerov I. et al. Exploiting Parallelism on Shared Memory in the QED Particle-in-Cell Code PICADOR with Greedy Load Balancing // PPAM 2019, Bialystok, Poland, Revised Selected Papers, Part I. – Cham: Springer International Publishing, 2020. – С. 335-347.
4. Volokitin V. et al. Optimized routines for event generators in QED-PIC codes // Journal of Physics: Conference Series. – IOP Publishing, 2020. – Т. 1640. – №. 1. – С. 012015.
5. Montefiori S., Tamburini M. SFQEDtoolkit: a high-performance library for the accurate modeling of strong-field QED processes in PIC and Monte Carlo codes // arXiv preprint arXiv:2301.07684. – 2023.
6. Fedeli L. et al. PICSAR-QED: a Monte Carlo module to simulate strong-field quantum electrodynamics in particle-in-cell codes for exascale architectures //New Journal of Physics. – 2022. – Т. 24. – №. 2. – С. 025009.
7. Volokitin V. et al. Optimized event generator for strong-field QED simulations within the hi-χ framework // arXiv preprint arXiv:2303.00648. – 2023.
8. Репозиторий проекта hi-χ. URL: https://github.com/hi-chi/pyHiChi

9. Bell A. R., Kirk J. G. Possibility of prolific pair production with high-power lasers // Physical review letters. – 2008. – Т. 101. – №. 20. – С. 200403.

Экспериментальная оценка немарковости кубита 5-кубитного квантового компьютера

П.Е. Ведруков, А.В. Линев, И.Б. Мееров, Т. Лаптева

Нижегородский государственный университет им. Н. И. Лобачевского

Введение

Обработка измерений специальных квантовых схем [1] методом гармонической инверсии [2] позволяет восстановить набор собственных значений, образующих диаграмму, качественно аналогичную полному спектру лиувиллиана открытой квантовой системы, представленному в [3]. Метод томографии Линдблада [4] позволяет оценить ошибки подготовки начального состояния и измерения (SPAM), операторы Крауса, меру немарковости, гамильтониан и операторы Линдблада, описывающие эволюцию открытой квантовой системы. Мы оценили SPAM ошибки, восстановили эволюцию с помощью операторов Крауса для дискретных моментов времени и оценили немарковость первого кубита квантового компьютера ibmq_belem на платформе IBM Quantum Computing.

Методика проведения экспериментов

1. Кубит инициализируется в состоянии $\rho_0 = |0\rangle$, после чего к нему применяется один из шести однокубитных поворотов $R_s = \{I, X_{\pi}, Y_{\frac{\pi}{2}}, Y_{-\frac{\pi}{2}}, X_{-\frac{\pi}{2}}, X_{\frac{\pi}{2}}\}$.

2. К кубиту применяется выжидающая схема $\tilde{I}(t) = U_1^{\dagger}U_2^{\dagger} \dots U_t^{\dagger}U_t \dots U_2U_1$.

3. Кубит измеряется в одном из базисов Паули (x, y, z) посредством применения соответствующего однокубитного поворота $R_b = \{Y_{-\frac{\pi}{2}}, X_{\frac{\pi}{2}}, I\}$.

Данные шаги применяются для всех комбинаций начальных поворотов, выжидающих схем $t_i \in \{0, \dots, 99\}$ и базисов измерения. Мы выполнили 20 000 измерений для каждой из 1 800 квантовых схем на квантовом компьютере ibmq_belem.

Анализ экспериментальных данных

1. Неидеальное начальное состояние описывается матрицей плотности квантовой системы ρ_0, положительно полуопределенной и имеющей единичный след. Ошибка измерения описывается положительной операторной мерой (POVM). POVM для одного кубита состоит из двух операторов, представляемых матрицами $M_0, M_1 \in \mathbb{C}^{2\times 2}: M_0 + M_1 = I$. Вероятность измерения в основном состоянии $p_0 = Tr(\rho_0 M_0)$, в возбужденном состоянии $p_1 = Tr(\rho_0 M_1)$. Для нахождения оценки подготовленного начального состояния ρ_0 и операторов M_0, M_1 мы использовали метод максимального правдоподобия с функцией стоимости следующего вида: $ln(\mathcal{L}_{\mathcal{SPAM}}) = \sum_{b,s} f(s,b) ln(Tr[\rho_s M_b]) + \overline{f}(s,b) ln(Tr[\rho_s (I - M_b)])$, где $b \in \{z, x, y\}$ – базис измерения, $s \in (|0\rangle, |1\rangle, |+\rangle, |-\rangle, |i\rangle, |-i\rangle)$ – повороты начального состояния, $f(s,b)$ и $\overline{f}(s,b)$ – количество измерений кубита в состояниях $|0\rangle, |1\rangle$ соответственно, $M_b = R_b M_0 R_b^{\dagger}$, $\rho_s = R_s \rho_0 R_s^{\dagger}$.

2. Любое сочетание квантовых операций может быть описано с помощью операторов Крауса: эволюция системы описывается как $\rho = \sum_j \mathcal{K}_j \rho_0 \mathcal{K}_j^{\dagger}$ с выполнением условия $\sum_j \mathcal{K}_j^{\dagger}\mathcal{K}_j = I$. Для восстановления операторов Крауса для всех временных задержек $t_i \in \{1, \dots, 99\}$ мы использовали следующую функцию стоимости $ln\big(\mathcal{L}_{\mathcal{K}}(t_i)\big) = \sum_{b,s} f(s,b,i) ln(Tr[\rho_s(t_i) M_b]) + \overline{f}(s,b,i) ln(Tr[\rho_s(t_i)(I - M_b)])$, где $\rho_s(t_i) = \sum_j \mathcal{K}_j \rho_s \mathcal{K}_j^{\dagger}$.

3. Для любого квантового процесса, который может быть описан уравнением Линдблада $\dot{\rho}(t) = -\frac{i}{\hbar}[H(t), \rho(t)] + \sum_i \gamma_i \left(L_i(t)\rho(t)L_i^{\dagger}(t) - \frac{1}{2}\{L_i^{\dagger}(t)L_i(t), \rho(t)\}\right)$, $\gamma_i > 0$, следовое расстояние

$D\big(\rho_1(0), \rho_2(0)\big)$ между состояниями двух систем со временем может только уменьшаться, а его увеличение означает наличие немарковской динамики. На основании этого наблюдения в работе [5] предложена следующая оценка немарковости процесса $N_{markov} = max_{\rho_1(0),\rho_2(0)} \int_{\sigma>0} \sigma\big(t, \rho_1(0), \rho_2(0)\big)dt$, где $\sigma\big(t, \rho_1(0), \rho_2(0)\big) = \frac{d}{dt}\Big(D\big(\rho_1(t), \rho_2(t)\big)\Big)$.

Результаты работы

Были выполнены две серии экспериментов.

1. Воздействие применялось к первому кубиту, к другим операции не применялись.

2. Воздействие применялось к первому кубиту, к остальным кубитам перед запуском задерживающей последовательности применялся гейт X.

Таблица 1. Матрицы начального состояния, POVM и оценка немарковости

Эксперимент 1	Эксперимент 2
$\rho_0 = \begin{pmatrix} 0.9955 & -0.0061 + 0.0067i \\ -0.0061 - 0.0067i & 0.0045 \end{pmatrix}$	$\rho_0 = \begin{pmatrix} 0.9962 & -0.0018 + 0.0018i \\ -0.0018 - 0.0018i & 0.0038 \end{pmatrix}$
$M_0 = \begin{pmatrix} 0.9995 & -0.017 + 0.0124i \\ -0.017 - 0.0124i & 0.0307 \end{pmatrix}$	$M_0 = \begin{pmatrix} 0.9992 & -0.0228 + 0.0128i \\ -0.0228 - 0.0128i & 0.0293 \end{pmatrix}$
$N_{markov} = 0.30799$	$N_{markov} = 0.21096$

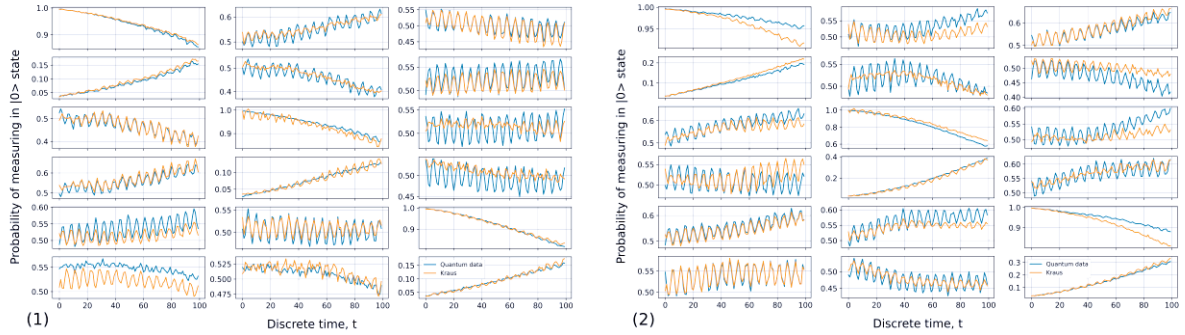


Рис. 1. Восстановление дискретной динамики операторами Крауса для экспериментов (1) и (2)

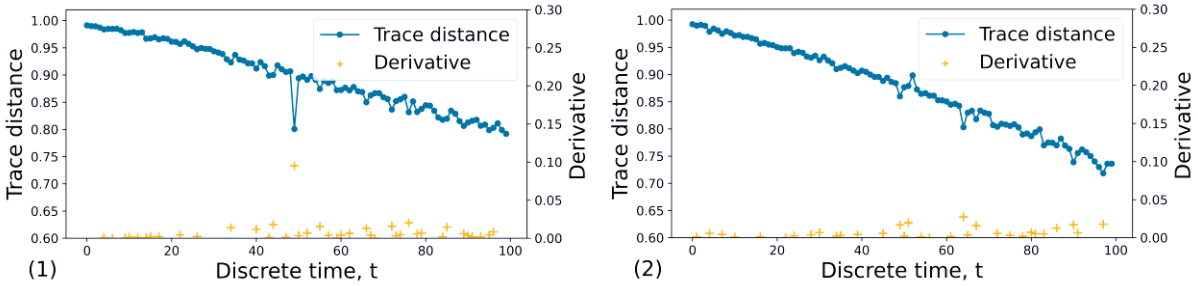


Рис. 2. Изменение следового расстояния для экспериментов (1) и (2)

Полученные результаты показывают низкую степень немарковости, к тому же, проявляющейся в виде «выбросов», что может быть связано с ошибками при выполнении экспериментов. Также они позволяют рассчитывать на возможность описания наблюдаемых процессов в виде не зависящего от времени уравнения Линдблада, что является нашей следующей задачей наряду с проведением экспериментов с двухкубитым взаимодействием.

Работа поддержана научно-образовательным математическим центром «Математика технологий будущего» (Соглашение 075-02-2023-945 от 16.02.2023 г.).

Авторы выражают благодарность С. Денисову за многолетнее сотрудничество.

Литература

1. Sommer O. E., Piazza F., Luitz D. J. Many-body hierarchy of dissipative timescales in a quantum computer //Physical Review Research. – 2021. – Т. 3. – №. 2. – С. 023190.
2. Mandelshtam V. A., Taylor H. S. Harmonic inversion of time signals and its applications //The Journal of chemical physics. – 1997. – Т. 107. – №. 17. – С. 6756-6769.
3. Wang K., Piazza F., Luitz D. J. Hierarchy of relaxation timescales in local random Liouvillians //Physical Review Letters. – 2020. – Т. 124. – №. 10. – С. 100604.
4. Samach G. O. et al. Lindblad tomography of a superconducting quantum processor //Physical Review Applied. – 2022. – Т. 18. – №. 6. – С. 064056.
5. Breuer H. P., Laine E. M., Piilo J. Measure for the degree of non-Markovian behavior of quantum processes in open systems //Physical review letters. – 2009. – Т. 103. – №. 21. – С. 210401.

Содержание

Научное издание

СУПЕРКОМПЬЮТЕРНЫЕ ДНИ В РОССИИ
Труды международной конференции
25–26 сентября 2023 г. Москва

Издательство «МАКС Пресс»
Главный редактор: *Е. М. Бугачева*
Обложка: *А. В. Кононова*

Напечатано с готового оригинал-макета

Подписано в печать 16.11.2023 г. Формат 60x90 1/8.
Усл.печ.л. 28,0. Тираж 8 экз. Изд. №. 161.

Издательство ООО “МАКС Пресс”. Лицензия ИД N 00510 от 01.12.99 г.
119992, ГСП-2, Москва, Ленинские горы, МГУ им. М.В. Ломоносова,
2-й учебный корпус, 527 к. Тел. 8(495) 939-3890/91. Тел./Факс 8(495) 939-3891.

Отпечатано в полном соответствии с качеством
предоставленных материалов в ООО «Фотоэксперт»
109316, г. Москва, Волгоградский проспект, д. 42,
корп. 5, эт. 1, пом. I, ком. 6.3-23Н