ISSN 2305-9052 (Print)
ISSN 2410-7034 (Online)

ВЕСТНИК ЮЖНО-УРАЛЬСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

BULLETIN OF THE SOUTH URAL STATE UNIVERSITY

СЕРИЯ

ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА

2022, том 11, № 1

SERIES

COMPUTATIONAL MATHEMATICS AND SOFTWARE ENGINEERING

2022, volume 11, no. 1

1943

ВЕСТНИК

ЮЖНО-УРАЛЬСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

2022
Т. 11, № 1

ISSN 2305-9052 (Print)
ISSN 2410-7034 (Online)

СЕРИЯ

«ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА»

Решением ВАК включен в Перечень научных изданий, в которых должны быть опубликованы результаты диссертаций на соискание ученых степеней кандидата и доктора наук

Учредитель — Федеральное государственное автономное образовательное учреждение высшего образования «Южно-Уральский государственный университет (национальный исследовательский университет)»

Тематика журнала:

- Вычислительная математика и численные методы
- Математическое программирование
- Распознавание образов
- Вычислительные методы линейной алгебры
- Решение обратных и некорректно поставленных задач
- Доказательные вычисления
- Численное решение дифференциальных и интегральных уравнений
- Исследование операций
- Теория игр
- Теория аппроксимации
- Информатика
- Искусственный интеллект и машинное обучение
- Системное программирование
- Перспективные многопроцессорные архитектуры
- Облачные вычисления
- Технология программирования
- Машинная графика
- Интернет-технологии
- Системы электронного обучения
- Технологии обработки баз данных и знаний
- Интеллектуальный анализ данных

BULLETIN
OF THE SOUTH URAL STATE UNIVERSITY
SERIES
"COMPUTATIONAL MATHEMATICS AND SOFTWARE ENGINEERING"

2022
Vol. 11, no. 1

ISSN 2305-9052 (Print)
ISSN 2410-7034 (Online)

Vestnik Yuzhno-Ural'skogo Gosudarstvennogo Universiteta.
Seriya "Vychislitel'naya Matematika i Informatika"

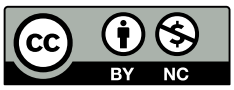
South Ural State University

The scope of the journal:

- Numerical analysis and methods
- Mathematical optimization
- Pattern recognition
- Numerical methods of linear algebra
- Reverse and ill-posed problems solution
- Computer-assisted proofs
- Numerical solutions of differential and integral equations
- Operations research
- Game theory
- Approximation theory
- Computer science
- Artificial intelligence and machine learning
- System software
- Advanced multiprocessor architectures
- Cloud computing
- Software engineering
- Computer graphics
- Internet technologies
- E-learning
- Database processing
- Data mining

Содержание

Contents



DOI: 10.14529/cmse220101

UNIFIED APPROACH FOR PROVISION OF SUPERCOMPUTER CENTER RESOURCES

© 2022 A.V. Paokin, D.A. Nikitenko

Lomonosov Moscow State University (GSP-1, Leninskie Gory 1, Moscow, 119991 Russia)

E-mail: andrejpaokin@yandex.ru, dan@parallel.ru

Received: 12.12.2021

Within one supercomputer center, there may be several computing systems with different architectures and principles of work with the end user. When organizing user access, it is necessary to fully describe the systems for the coordinated choice of tasks, software packages and hardware by the user, as well as to take into account the details of quotas, authentication, and launching applications on each of the individual machines within a single workflow. In this paper, we propose an approach to the provision of resources of a supercomputer center, where a user, using a complete description of computing systems, creates requests for access with desirable quotas. The approach describes the life cycle of access. When an access state transition occurs, it is supposed to interact with computing systems through their interfaces without deep integration. An overview of widely used approaches to quoting and organizing access is given, and the proposed approach is implemented as a software module for the Octoshell supercomputer center support system and tested on a computing system managed by the OpenNebula cloud computing platform.

Keywords: supercomputer center administering, resource provision, Octoshell.

FOR CITATION

Paokin A.V., Nikitenko D.A. Unified Approach for Provision of Supercomputer Center Resources. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2022. Vol. 11, no. 1. P. 5–14. DOI: 10.14529/cmse220101.

Introduction

A modern supercomputer center can consist of several computing systems with tens of thousands of hardware and software components which are used by thousands of people [1]. Therefore, organizing the work of a supercomputer center is a complex task.

Within one supercomputer center, in addition to supercomputers with distributed memory, there may be virtual machines. The number of virtual machines is limited only by physical resources and the desire of users to use them. The supercomputer often acts as a shared resource, while virtual machines are created for users to be used exclusively.

Therefore, resource quotas on these machines can be different. For example, in the cloud case, a virtual machine is created on the server of a certain CPU model occupying as many cores as a cloud administrator previously defined. In case of supercomputers, CPU hours or available nodes can act as similar but not the same type of resource. There are other limits that usually relate to supercomputers only: the maximum number of files for a user, the maximum number of user tasks to be executed, etc. Supercomputer limits are often correspond to different software, making the task of providing resources more difficult.

Therefore, a unified approach to the provision of resources of the supercomputer center is proposed in this paper. Its main goal is to help users choose a computing system based on its description and send a correct request to be considered by administrators. The approach will help administrators to control user access in a unified manner.

The article is organized as follows. Section 1 describes existing approaches for computer resource provision. Section 2 is devoted to the proposed approach and necessary interfaces for computing systems to be implemented. In Section 3 we describe the implementation of the approach used for cloud computing platforms. Section 4 contains performance evaluations and implementation details linked to such evaluations.

1. Existing approaches for computing resource provision

It is important to consider organizational properties of resource provision. First of all, there are project-based or user-based model of resource delegation. In the user-based model, resources are simply given to a user, but in the project-based model, resources are given to a project: a group of users united to solve one specific problem. Sometimes computing resources can be given to scientists for free. Computing resources can be given on a free basis: this can be actual for supercomputer centers connected with a government. But, in this case, the work carried on the computing systems should be monitored properly: the problem to be solved should be well described, and scientific results checked from time to time.

In this paper, we focus on computing system description, quota usage and authentication, but the resulting method should be able to integrate into any of the organizational schemes.

1.1. Supercomputer system resource provision

The widespread way of accessing supercomputer resources is SSH protocol. Using a command-line interface, users send their computing tasks to a workload manager and these tasks are executed when requested resources are ready. It is possible that a supercomputer consists of several partitions, which differ in hardware. That is why limits can correspond to one partition only or the whole supercomputer and limits can be set for user, project (group of users). Also, limits can be cumulative (CPU hours) and number of used resources should be saved and checked from time to time.

Different combinations of conditions lead to difficulties in defining and checking limits, but this is only a part of the problem: supercomputer operation usually depends on several software tools and limits are defined there in different ways. It is possible that two types of file systems exist for one supercomputer and the typical quotas on maximum hard drive space and number of files are written in different ways. The other user quotas are set on CPU hours and the number of launched and pending tasks.

1.2. Cloud computing platforms

The OpenNebula cloud computing platform will be considered further from the point of view of this work [2]. Every object in OpenNebula has its own id and type. Some of them are listed below:

- **A virtual machine** can have several real / virtual CPUs and, depending on the settings, a different amount of allocated RAM. These parameters can be changed, and only when the machine is in the “power off” state. There are more than 60 states in total: power on, power off, waiting for resource allocation, etc., as well as various transient and error states.
- **Host**, a server running OpenNebula responsible for running virtual machines.
- **Hard drive**, where virtual machine data is stored.
- **Virtual network**, designed for communication between virtual machines and with the management server.

- **Image**, containing operating system and software.
- **Template**, containing virtual machine settings and can be used for creating virtual machines.

Access privilege management in OpenNebula is similar to the UNIX file system. Each object has its owner, and you can define available actions for three categories of users in relation to the object: owner, owner's group, and other users. There are 3 available actions: use, management and administration.

OpenNebula is accessible via the command line interface and the XML-RPC interface. OpenNebula provides the reach web-interface called OpenNebula Sunstone using the XML-RPC interface above to interact with OpenNebula, available for administrators and regular users. Administrators and ordinary users (of course, respecting quotas given) can create a virtual machine from scratch or through a template. In the latter case, the user can change any item from the list above, taking into account access rights.

Administrators can set limits on the usage for users and user groups applied for different types of objects: data stores, virtual machines (to limit the overall memory, cpu or VM instances), network (to limit the number of IPs got from a given network), images (images can contain consumable resources like software licenses) [3].

OpenStack is probably a more famous cloud computing platform [4]. From a system administrator's point of view, it differs from OpenNebula. An OpenNebula cluster can be exploited by a single system administrator, which is not the same as talking about OpenStack, especially during software upgrades. But, the user level resource description is almost the same. A notable difference is that OpenStack uses flavors instead of templates. Flavor is an available hardware configuration for a server [5]. It defines parameters for a virtual server that can be launched: number of virtual CPUs, main memory and root disk sizes.

Both OpenNebula and OpenStack lack resource applications or other means of communication among ordinary users or administrators. Despite the positive experience of working with OpenNebula at the supercomputer center of Lomonosov Moscow State University, users, who were given access to OpenNebula, sometimes addressed simple tasks to administrators directly via email, and not to OpenNebula Sunstone.

2. Proposed approach

Before proceeding to the detailed description of the approach, we will describe the basic principles.

- To access computing resources, users send requests, and they should be structured: it should be clear, which system the request is for and how much resources are requested.
- It is important to fully describe computing systems in a structured manner, so that a user could choose a right computing system. Values of resources and their limits should be clear to a user.
- Thousands of users can work in a single supercomputer center. That is why, common tasks, such as granting and blocking access, should be automated.
- The software implementation of the approach should not be deeply integrated with computing systems, but communicate with them through special interfaces.

Structural description of computing systems and request, if approved, is necessary to automatically provide access with the required parameters. Parameters can be various quotas, for example, CPU hours for supercomputers or the maximum amount of used RAM for virtual

machines. Computing systems can be combined into types, and they, in turn, into a hierarchy, which makes it easier for users to find a system suitable for their problems.

The request, in addition to a complete description of the required computing resources, contains the fields for a sender and an administrator who considers it and various details related to the methods of access. A widespread option is the SSH protocol, which prompts the idea that users only need to describe in one place all the information related to access. It is allowed to submit a request for modifying an existing access. Requests can be in one of the following states: new, submitted for review, canceled by user, rejected and approved. Upon approval of the application, an administrator can change some parameters, informing the user about this and a request to grant access with the required parameters is sent.

The user should have access to the technical details of connecting to the computer and a set of possible actions: "power on", "power off", "get state" (this is relevant for virtual machines). Access can be in one of the following states:

- New. Access is in this state if the administrator has approved the user's request and is changing some parameters at the moment.
- Approved. The user is allowed to access the computing system.
- Prepared for completion by the administrator, prepared for completion. In these states, the user is denied access, but the data on the hard drives is not erased yet. These states differ only in those who initiated this process: the administrator or the user.
- Access denied, completed. In these states, all user data on hard drives is erased. In the states "Access Denied" and "Completed", the administrator transfers accesses from the states "Prepared for completion by the administrator" and "Prepared for completion" respectively.

The interface for communicating with computing systems is described further in general. So, for example, the actions to turn a supercomputer on and off by a user are completely irrelevant, but during the operation of a virtual machine, they can be useful when a virtual machine is not responding. The actions that can be performed through the interface are as follows:

- Load data about computing system, if it is relevant.
- Create and modify access parameters specified by requests.
- Load information about the current state of the computer and possible actions at the moment. For user convenience, hints and names for operations in different languages can be added.
- Perform actions to change the state of the machine available to the user: power on, power off, etc.
- Terminate user access without deleting user data on hard drives.
- Delete data on hard drives.

3. Implementation

Like any other workflow for communicating with the user at the Lomonosov Moscow State University supercomputer center, it is reasonable to implement provision of an access as a module for the Octoshell supercomputer center functioning support system accessed using web browsers [6, 7]. Also, the process of granting access also depends on the state of projects, the availability of SSH keys and the fact of passing the preliminary procedures, the logic of which has already been implemented in Octoshell and its implementation is open-source [8]. Integration with other objects in the Octoshell system is also a definite advantage.

In Octoshell, the organization of access to supercomputer systems is independent of a specific machine and is not related to the system software responsible for supercomputer management. Users are united into projects and gain or lose access to computing systems depending on events in the Octoshell system. For example, if the report on the project work gets a low mark, then this project goes into the "blocked" state and then, after synchronization, the project is denied access. Synchronization calls scripts located directly on supercomputers and thus the scripts form an interface. To work on the computing systems of Lomonosov Moscow State University, users go through preparatory procedures that are not directly related to the approach of providing resources such as preparation of required documents and their approval. Project managers apply for resource quotas and users upload the public part of their SSH keys to the Octoshell system, which are used to work with all available supercomputer systems. The resources subject to quotas are CPU hours, disk space, and GPU hours. These routines are implemented inside the Octoshell module called core.

For work on supercomputer systems, the core module has shown itself well, but it is difficult to call it a unified approach. The types of resource quotas are hardcoded right in the source code, and these types of quotas are not suitable for virtual machines. The same applies to the description of supercomputer systems themselves. The approach proposed in the article is implemented primarily in the module for interacting with cloud systems. But this does not imply module generality, because it can also work with supercomputers and cloud systems at the same time. It is convenient to describe the implementation details through entities, so they will be considered further (Fig. 1).

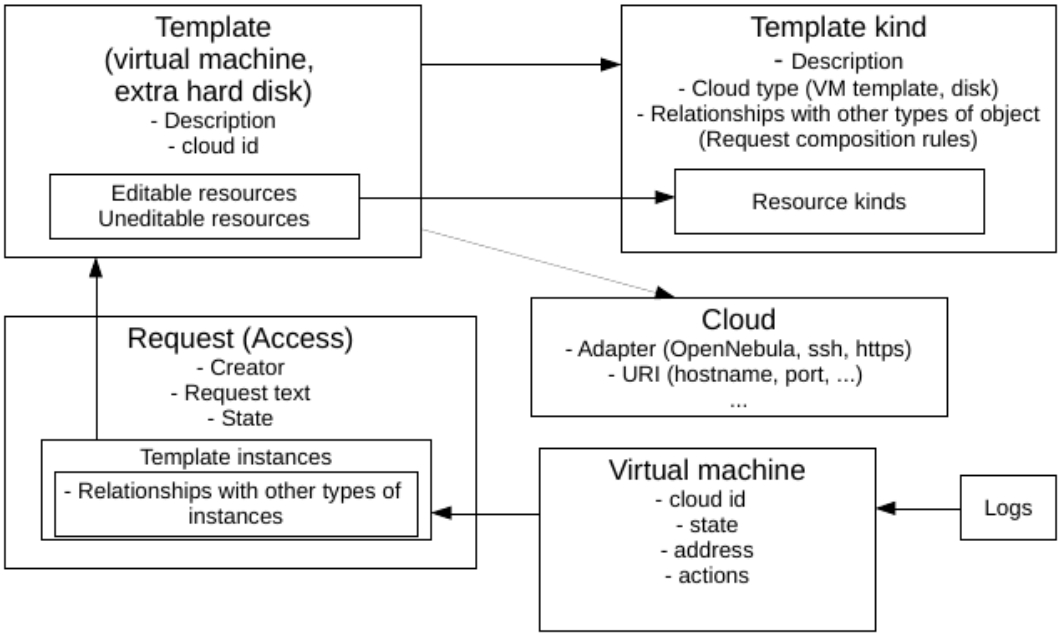


Fig. 1. The diagram of module entities

Template and resource kinds. Every cloud infrastructure object, that a user can send requests for, is represented in the module as a template. The template has its own type and you can specify links with other types. These links show how the instances in the requests and user accesses can be related to each other (the power of the connection and, in general, the ability to link the instances). Types can be combined in hierarchies, thus creating subtypes. For a template

type, the relation to a cloud computing platform can be specified. At the moment, only the type of relationship to the cloud called "virtual machine" is supported, and this can be specified only for one type of template. The template type also indicates the types of resources that will be changed in user requests or not, depending on template resource configuration. Each resource type has an identifier by which cloud platforms associate the resource type in the Octoshell module with their own. For example, the resource type "RAM size" has the "MEMORY" identifier. Currently, Octoshell has the following types of resources: the amount of RAM and disk space, the number of CPU cores and access to the Internet. The latter, from the OpenNebula point of view, means belonging to the special OpenNebula network with a dedicated Internet address.

A template has a verbal description, platform affiliation, and a cloud id that is unique on the target cloud platform. For the template, resources are specified that the user can edit or only see, depending on the settings of the resource view. There is an option not to specify resources at all: then, when creating virtual machines, only the values of the template stored on the target cloud platform will be used. The module allows you to create descriptions of a template, its type and resource types completely manually, or almost automatically by loading them from the target cloud platform: it remains only to indicate whether the resource is editable or not, as well as the limits in user requests and default values (if the resource is editable).

Requests are described by template instances, text, id of the creator and admin who reviewed the request, and state. Request and access states were described in the previous section.

Access also provides a collection of template instances, but already functioning on the template target platform, and state.

Virtual machine has its own id on the cloud and external, internal addresses. Each time the status information is updated, the possible actions for the virtual machine are also updated and saved in the Octoshell database: starting from the id of the action to be parsed by the cloud adapter and ending with prompts and action names in Russian and English. As a result, the user access page with virtual machines takes the following form in Fig. 2.

There are usually several OpenNebula XML-RPC interface commands behind each adapter action, and each of them, depending on the cloud state, may not be executed due to errors or have to wait until the virtual machine takes a suitable state (after exiting the intermediate state caused by the previous command, for example). Therefore, the result of each action is recorded in the **logs**. In case of command execution errors, administrators are notified.

4. Performance evaluation

As it has been already mentioned, the proposed approach is implemented in the Octoshell supercomputer center functioning support system and users access its functionality using web browsers. Thus, the primary goal is to ensure that pages are downloaded fast. Base functionality of all Octoshell engines (including the engine implementing the current approach) are run on the same virtual machine (on the same Linux process) sharing hardware resources equally. Technically, new engine did not operate with huge amounts of data, did not add sufficient amount of code able to disrupt the functioning of virtual machine. Reasonably implemented the previous version of the engine showed itself well, consuming less than 0.5 second to prepare page for web browser, what is the same with many other Octoshell engines.

Previous information corresponds to the base functionality, but there are long-running jobs in almost any modern web application. "Long-running" refers to processes consuming (or which can consume) significantly more than 1 second: such delays are not suitable for any modern web

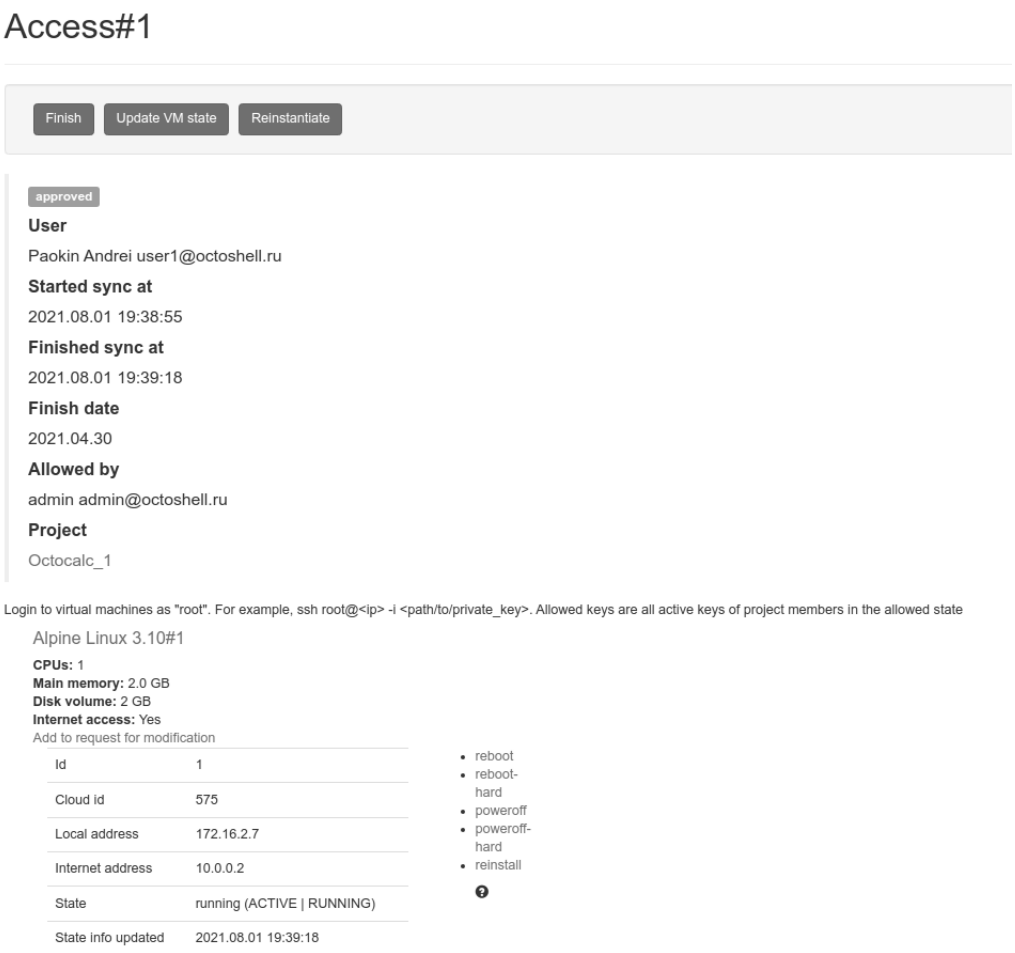


Fig. 2. User access page

application. Typical examples of such jobs in Octoshell are sending an email and interaction with computing systems, which are located on the different servers and even can be inaccessible at the moment. The typical solution is to delegate these jobs to other operating system processes, usually called background workers to be executed asynchronously. As a result, the only thing remained to be considered is efficiency of background job execution.

When talking about our approach, long-running jobs are all interactions with servers responsible for computing system functioning. Thus, these jobs are delegated to background workers and a user does not need to wait until they finish, usually receiving notifications about their final status: completed or failed. In general, adapters responsible for such communications simply send their messages via SSH and HTTPS protocols. Requests described at the end of the "Proposed approach" section do not require return value and can be executed asynchronously. Even data about the current state of a computing system does not require to be loaded immediately.

In case of OpenNebula, it has the rich API, but there are situations when one request of the Octoshell cloud engine results in nearly 10 OpenNebula API calls. A virtual machine can be in one of nearly 60 states: error, transient ("changing disk size") or normal state (power off, power on, etc.). To update parameters of a virtual machine, the Octoshell cloud engine requests can cause attachment or removal of SSH keys, network interface controllers, change of disk or main memory size, CPU resources, etc. Each of these actions has to be run in their own specific

state with their own OpenNebula API call. For us, it was reasonable to implement OpenNebula adapter directly inside Octoshell instead of creating a new subsystem to accomplish our goals. The main idea of adapter is to divide big requests like virtual machine modification, creation into small requests. Each small request is OpenNebula API call with required actions to check that a virtual machine is in a right state. In some cases, background worker has to wait until virtual machine leaves transient state and background worker performs API call to check virtual machine state. If the state is transient, it sleeps (operating system call) for 5 seconds, then sends requests again. Anyway, each large request is performed in less than 30 seconds and these background workers do not load web server resources significantly. This situation is expected to remain in the future, when more virtual machines controlled by Octoshell appear. If not, the architecture of the cloud computing engine allows to develop non-local adapters, accessible via HTTPS and SSH protocols.

Conclusions

As a result, the unified approach to the provision of resources in the supercomputer center and the module for organizing user access on cloud systems for the Octoshell have been developed. The cloud module can also be used to work with supercomputer systems.

The module improves the convenience of working with the cloud, reduces the number of manual procedures and the loss of information, which is now presented in a common place for all participants (project member, project manager and administrator). The latter is especially relevant in the upcoming tasks of creating a new information system, which, based on data from monitoring systems and Octoshell together will inform the user about the peculiarities of the task and the expected behavior on different sections of the supercomputer center.

It is especially important that such system has access to data from monitoring systems of both the user's tasks and similar users, where data integration with Octoshell is often indispensable.

The module has been successfully tested on a copy of the main Octoshell system and a server running OpenNebula [9]. It will be deployed to the main system after the next update.

The results are obtained with the financial support of the Russian Foundation for Basic Research (grant No. 20-07-00864).

References

1. 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. Supercomputing Frontiers and Innovations. 2019. Vol. 6, no. 2. P. 4–11. DOI: 10.14529/jsfi190201.
2. OpenNebula – Open Source Cloud Computing Platform. URL: https://opennebula.io/ (accessed: 12.12.2021).
3. OpenNebula Usage Quota. URL: https://docs.opennebula.io/6.0/management_and_operations/capacity_planning/quotas.html (accessed: 12.12.2021).
4. OpenStack – Open Source Cloud Computing Platform. URL: https://opennebula.io/ (accessed: 12.12.2021).

5. OpenStack flavors. URL: https://docs.openstack.org/nova/rocky/user/flavors.html (accessed: 12.12.2021).

6. Nikitenko D., Voevodin V., Zhumatiy S. Resolving Frontier Problems of Mastering Large-Scale Supercomputer Complexes. Proceedings of the ACM International Conference on Computing Frontiers. ACM, 2016. P. 349–352. DOI: 10.1145/2903150.2903481.

7. Octoshell Supercomputer Center Support Functioning System. URL: https://users.parallel.ru/ (accessed: 12.12.2021).

8. Octoshell Source Code. URL: https://github.com/octoshell/octoshell-v2 (accessed: 12.12.2021).

9. Cloud Computing Engine for the Octoshell System. URL: https://github.com/apaokin/octoshell-v2/tree/cloud_computing_engine/engines/cloud_computing (accessed: 12.12.2021).

УДК 004.457 **DOI: 10.14529/cmse220101**

УНИФИЦИРОВАННЫЙ ПОДХОД К ПРЕДОСТАВЛЕНИЮ РЕСУРСОВ СУПЕРКОМПЬЮТЕРНОГО ЦЕНТРА

© 2022 А.В. Паокин, Д.А. Никитенко

Московский государственный университет имени М.В. Ломоносова
(119991 Москва, ул. Ленинские горы, д. 1)
E-mail: andrejpaokin@yandex.ru, dan@parallel.ru
Поступила в редакцию: 12.12.2021

В рамках одного суперкомпьютерного центра могут находиться несколько вычислительных систем с разной архитектурой и принципами работы с конечным пользователем. При организации доступа пользователей необходимо полное описание систем для согласованного выбора пользователем задач, программных пакетов и аппаратуры, а также учитывать детали квотирования, аутентификации, запуска приложений на каждой из отдельных машин в рамках единого рабочего процесса. В данной работе предлагается подход к предоставлению ресурсов суперкомпьютерного центра, где пользователь, пользуясь полным описанием вычислительных систем, создает заявки на доступ с желаемыми квотами. Подход описывает жизненный цикл доступа, при изменении состояния которого предполагается взаимодействие с вычислительными системами через интерфейсы систем без глубокой интеграции. Дается обзор широкоиспользуемых подходов к квотированию и организации доступа, а предложенный подход реализован в виде программного модуля для системы поддержки функционирования суперкомпьютерных центров Octoshell и апробирован на вычислиительной системе под управлением платформы организации облачных вычислений OpenNebula.

Ключевые слова: администрирование суперкомпьютерных центров, предоставление ресурсов, Octoshell.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Paokin A.V., Nikitenko D.A. Unified Approach for Provision of Supercomputer Center Resources // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2022. Т. 11, № 1. С. 5–14. DOI: 10.14529/cmse220101.

Литература

1. 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 // Supercomputing Frontiers and Innovations. 2019. Vol. 6, no. 2. P. 4–11. DOI: 10.14529/jsfi190201.
2. OpenNebula – Open Source Cloud Computing Platform. URL: https://opennebula.io/ (дата обращения: 12.12.2021).
3. OpenNebula Usage Quota. URL: https://docs.opennebula.io/6.0/management_and_operations/capacity_planning/quotas.html (дата обращения: 12.12.2021).
4. OpenStack – Open Source Cloud Computing Platform. URL: https://opennebula.io/ (дата обращения: 12.12.2021).
5. OpenStack flavors. URL: https://docs.openstack.org/nova/rocky/user/flavors.html (дата обращения: 12.12.2021).
6. Nikitenko D., Voevodin V., Zhumatiy S. Resolving Frontier Problems of Mastering Large-Scale Supercomputer Complexes // Proceedings of the ACM International Conference on Computing Frontiers. ACM, 2016. P. 349–352. DOI: 10.1145/2903150.2903481.
7. Octoshell Supercomputer Center Support Functioning System. URL: https://users.parallel.ru/ (дата обращения: 12.12.2021).
8. Octoshell Source Code. URL: https://github.com/octoshell/octoshell-v2 (дата обращения: 12.12.2021).
9. Cloud Computing Engine for the Octoshell System. URL: https://github.com/apaokin/octoshell-v2/tree/cloud_computing_engine/engines/cloud_computing (дата обращения: 12.12.2021).

Паокин Андрей Викторович, аспирант, кафедра суперкомпьютеров и квантовой информатики, Московский государственный университет имени М.В. Ломоносова (Москва, Российская Федерация)

Никитенко Дмитрий Александрович, к.ф.-м.н., с.н.с., лаборатория параллельных информационных технологий, Научно-исследовательский вычислительный центр, Московский государственный университет имени М.В. Ломоносова (Москва, Российская Федерация)

УДК 004.272, 004.9 **DOI: 10.14529/cmse220102**

ВЫСОКОПРОИЗВОДИТЕЛЬНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ РЕСУРСЫ ЮЖНО-УРАЛЬСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

© 2022 Р.В. Биленко, Н.Ю. Долганина, Е.В. Иванова, А.И. Рекачинский

Южно-Уральский государственный университет
(454080 Челябинск, пр. им. В.И. Ленина, д. 76)
E-mail: bilenkorv@susu.ru, dolganinani@susu.ru,
elena.ivanova@susu.ru, alexander.rekachinsky@susu.ru

Поступила в редакцию: 20.02.2022

В настоящее время в Южно-Уральском государственном университете достигнуты значительные результаты в области суперкомпьютерного моделирования, искусственного интеллекта и больших данных. ЮУрГУ обладает энергоэффективным суперкомпьютером «Торнадо ЮУрГУ», который занимает 15е место в рейтинге самых мощных суперкомпьютеров СНГ ТОП50 (сентябрь 2021). Для исследований в области искусственных нейронных сетей в ЮУрГУ был установлен специализированный многопроцессорный комплекс «Нейрокомпьютер». «Нейрокомпьютер» использует мощные передовые графические ускорители для обучения нейронных сетей. Суперкомпьютер «Торнадо ЮУрГУ» и комплекс «Нейрокомпьютер» находятся в центре научной жизни Университета, позволяя производить сложнейшие вычисления для расчетов в области инжиниринга, естественных наук, наук о человеке и искусственного интеллекта. Вычислительные ресурсы ЮУрГУ используются в образовании и в коммерческих целях для расчетов задач партнеров Университета. В работе описываются характеристики высокопроизводительного оборудования ЮУрГУ, доступное системное и прикладное параллельное программное обеспечение, приведены сведения о решенных научных и инженерных задачах.

Ключевые слова: суперкомпьютер, нейрокомпьютер, параллельная система хранения данных, администрирование суперкомпьютеров, суперкомпьютерное моделирование, нейронные сети.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Биленко Р.В., Долганина Н.Ю., Иванова Е.В., Рекачинский А.И. Высокопроизводительные вычислительные ресурсы Южно-Уральского государственного университета // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2022. Т. 11, № 1. С. 15–30. DOI: 10.14529/cmse220102.

Введение

В Южно-Уральском государственном университете достигнуты значительные результаты в области создания цифровой индустрии. Активно развиваются исследования с применением суперкомпьютерного моделирования, искусственного интеллекта и больших данных. В настоящее время ЮУрГУ обладает энергоэффективным суперкомпьютером «Торнадо ЮУрГУ», который занимает 15 место в рейтинге самых мощных суперкомпьютеров СНГ ТОП50 (сентябрь 2021). Задачи в области искусственного интеллекта требуют высокого параллелизма на общей памяти, но суперкомпьютер с кластерной архитектурой этого не обеспечивает. Для создания искусственных нейронных сетей ЮУрГУ был приобретен специализированный многопроцессорный комплекс «Нейрокомпьютер». Нейрокомпьютер использует мощные передовые графические ускорители для обучения нейронных сетей. Высокопроизводительными вычислительными ресурсами ЮУрГУ пользуются более 500 человек, это не только сотрудники и студенты Южно-Уральского государственного университета, но сотрудники внешних образовательных, научных и производственных орга-

низаций (промышленные предприятия, университеты, институты РАН). В ЮУрГУ создан научно-образовательный центр «Искусственный интеллект и квантовые технологии» (НОЦ ИИКТ) [1], сотрудники которого занимаются администрированием высокопроизводительных ресурсов ЮУрГУ, научными исследованиями в области суперкомпьютерных технологий, обеспечивают поддержку пользователей. В данной работе выполнен обзор высокопроизводительных ресурсов ЮУрГУ, системного и прикладного программного обеспечения.

Остаток статьи организован следующим образом. В разделе 1 описаны основные высокопроизводительные ресурсы ЮУрГУ, установленные в НОЦ ИИКТ, такие как суперкомпьютер «Торнадо ЮУрГУ», комплекс «Нейрокомпьютер», системы хранения данных Panasas ActiveStor 11, OceanStor Dorado 3000 V6, Huawei OceanStor 5300 V5. Раздел 2 посвящен обзору системного программного обеспечения, используемого в НОЦ ИИКТ. Описано применение систем мониторинга и управления оборудованием и программными системами в НОЦ ИИКТ. В разделе 3 представлено прикладное программное обеспечение, доступное пользователям высокопроизводительных ресурсов ЮУрГУ, выполнен обзор решаемых научных и инженерных задач. Заключение содержит итоговые выводы.

1. Высокопроизводительные ресурсы

1.1. Суперкомпьютер «Торнадо ЮУрГУ»

Суперкомпьютер «Торнадо ЮУрГУ» представляет собой вычислительный комплекс с полным жидкостным охлаждением производительностью 473.6 Терафлопс, который в настоящее время занимает 15 место в рейтинге самых мощных суперкомпьютеров СНГ ТОП50 (сентябрь 2021). Использование жидкостного охлаждения влечет за собой увеличение энергоэффективности системы (экономия 40–50% электроэнергии по сравнению с системами с воздушным охлаждением), а также максимизацию плотности упаковки электроники, что позволяет избавиться от подвижных частей в вычислителе, шума и вибрации, повышая тем самым надежность и эргономику установки [2]. Технические характеристики суперкомпьютера «Торнадо ЮУрГУ» представлены в табл. 1.

Таблица 1. Технические характеристики суперкомпьютера «Торнадо ЮУрГУ»

Характеристика	Значение
Число выч. узлов/процессоров/ сопроцессоров/процессорных ядер	480/960/384/29184
Тип процессора	Intel Xeon X5680 (Gulftown, 6 ядер по 3.33 GHz)
Тип сопроцессора	Intel Xeon Phi SE10X (61 ядро по 1.1 GHz)
Оперативная память	16.9 Тбайт
Дисковая память	204 Тбайт, Panasas ActiveStor 11; 700 Тбайт, Huawei OceanStor 5300 V5
Тип системной сети	InfiniBand QDR (40 Гбит/с)
Тип управляющей сети	Gigabit Ethernet
Пиковая производительность	473.6 Терафлопс
Операционная система	Linux CentOS 6.2

Суперкомпьютер «Торнадо ЮУрГУ» для хранения исходных данных и результатов расчетов пользователей оснащен высокопроизводительной параллельной системой хране-

ния данных Panasas ActiveStor 11 с пиковой производительностью 30 886 операций ввода-вывода в секунду, при этом скорость записи составляет 2 402 МБ/с, а скорость чтения — 3 239 МБ/с. В настоящее время хранилище успешно используют более 500 пользователей суперкомпьютера, система используется непрерывно с момента установки в 2013 году и показала себя отказоустойчивой и надежной.

Система хранения данных Panasas ActiveStor 11 состоит из пяти полок. Четыре полки содержат по 10 узлов для хранения данных (StorageBlade) объемом 4 Тбайт и по одному узлу управления (DirectorBlade), пятая — состоит из 11 узлов хранения данных. Объем хранилища составляет 204 Тбайт, часть которого отведена под репликацию данных. Узлы управления выполняют задачу хранения метаданных, а также обеспечивают доступ к данным по таким протоколам как NFS и CIFS. В системе настроено 7 виртуальных узлов горячего резерва (hot spare), позволяющих достигнуть устойчивости работы при отказе до семи дисков включительно.

Полки в Panasas ActiveStor 11 имеют поддержку только интерфейса 10 Gigabit Ethernet. Для подключения системы к сети Infiniband QDR со скоростью 40 Гбит/с используются три Ininiband роутера Panasas, которые маршрутизируют пакеты из сети Infiniband QDR в сеть СХД. Балансировка нагрузки сети выполняется на вычислительных узлах посредством маршрутов до сети системы хранения с одинаковой метрикой.

1.2. Комплекс «Нейрокомпьютер»

Архитектура комплекса «Нейрокомпьютер» основывается на разнородных графических ускорителях, что позволяет гибко выбрать подходящее оборудование для максимально эффективного расчета любой задачи связанной с нейронными сетями. Комплекс состоит из шести серверов, объединенных общей очередью задач, с помощью которой пользователь получает доступ к серверу с требуемой для его задачи архитектурой. Характеристики комплекса «Нейрокомпьютер» представлены в табл. 2.

Архитектура комплекса «Нейрокомпьютер» представлена на рис. 1 и состоит из двух GPU-серверов Dell PowerEdge R750 на базе NVIDIA Ampere A100, трех GPU-серверов Dell PowerEdge R750 на базе NVIDIA Ampere A30, одного GPU-сервера HPE Apollo ProLiant XL270d Gen10 на основе NVIDIA Tesla V100 и трех управляющих серверов Dell PowerEdge R640.

GPU-сервер Dell PowerEdge R750 на базе передовых на сегодняшний день графических процессоров NVIDIA Ampere A100 имеет два таких процессора с объемом видеопамяти 80 Гбайт, два процессора Intel Xeon Silver 4314, оперативную память объемом 192 Гбайт и постоянную память на твердотельных накопителях объемом 1.9 Тб. GPU-сервер Dell PowerEdge R750 на базе NVIDIA Ampere A30 состоит из двух графических процессоров, имеющих объем видеопамяти 24 Гбайт, двух процессоров Intel Xeon Silver 4314, оперативной памяти объемом 192 Гбайт и твердотельных дисках объемом 1.9 Тбайт. Данные серверы наилучшим образом подходят для требовательных к видеопамяти задач и задач, которым ресурсы необходимы в монопольный доступ.

GPU-сервер HPE Apollo ProLiant XL270d Gen10 представляет собой сервер с восемью графическими процессорами NVIDIA Tesla V100 SXM2 (32 Гбайт видеопамяти), объединенных сетью NVLink, двумя процессорами Intel Xeon Gold 6254, оперативной памятью объемом 192 Гбайт и твердотельными дисками общим размером 7.68 Тбайт. Сервер позволя-

Таблица 2. Технические характеристики комплекса «Нейрокомпьютер»

Характеристика	Значение
Число графических процессоров/ ядер CUDA	18/91432
Типы графических процессоров	NVIDIA Ampere A100 80 GB PCI-E — 4 шт. NVIDIA Ampere A30 24 GB PCI-E — 6 шт. NVIDIA Tesla V100 SXM2 — 8 шт.
Число вычислительных процессоров/ процессорных ядер	18/268
Типы вычислительных процессоров	Intel Xeon Gold 6254 (Cascade Lake, 18 ядер по 4 GHz) — 2 шт.; Intel Xeon Silver 4314 (Ice Lake, 16 ядер по 3.4 GHz) — 10 шт.; Intel Xeon Silver 4214 (Cascade Lake, 12 ядер по 3.2 GHz) — 6 шт.
Оперативная память	1920 Гбайт
Хранилище данных	700 Тбайт, Huawei OceanStor 5300 V5; 46 Тбайт, система хранения данных на основе твердотельных накопителей Huawei OceanStor Dorado 3000 V6
Коммуникационная сеть	Mellanox Infiniband QSFP28, Mellanox Infiniband SFP28, Gigabit Ethernet
Пиковая производительность	276.4 Терафлопс
Операционная система	Linux Centos 7.8

ет достигнуть максимальной эффективности при распараллеливании задач на нескольких графических ускорителях.

Также в комплекс входят три управляющих сервера Dell PowerEdge R640, необходимых для организации работы комплекса. Каждый управляющий сервер содержит два процессора Intel Xeon Silver 4214 и 256 Гбайт оперативной памяти.

Для обучения нейронных сетей используются наборы данных больших объемов, которые в большинстве случаев состоят из множества мелких файлов (изображения, аудио и видео файлы). Для эффективного обучения нейронных сетей к комплексу «Нейрокомпьютер» подключена система хранения данных на основе твердотельных накопителей Huawei OceanStor Dorado 3000 V6, которая обеспечивает максимально доступную производительность при работе с данными типами файлов. Данная система выполняет чтение и запись на два порядка быстрее по сравнению с жесткими дисками и, в отличие от классических систем хранения, не теряет производительность при работе с большим количеством мелких файлов.

Архитектура Huawei OceanStor Dorado 3000 V6 представляет собой систему, содержащую два контроллера, которые могут сменять друг друга в случае отказа одного из них, обеспечивая тем самым бесперебойную работу хранилища. В каждый контролер встроен один процессор Kungpeng 920, разработанный компанией Huawei на базе архитектуры ARM, с 96 Гбайт кэш-памяти и сопроцессор Ascend 310 для поддержания сервисных нейронных

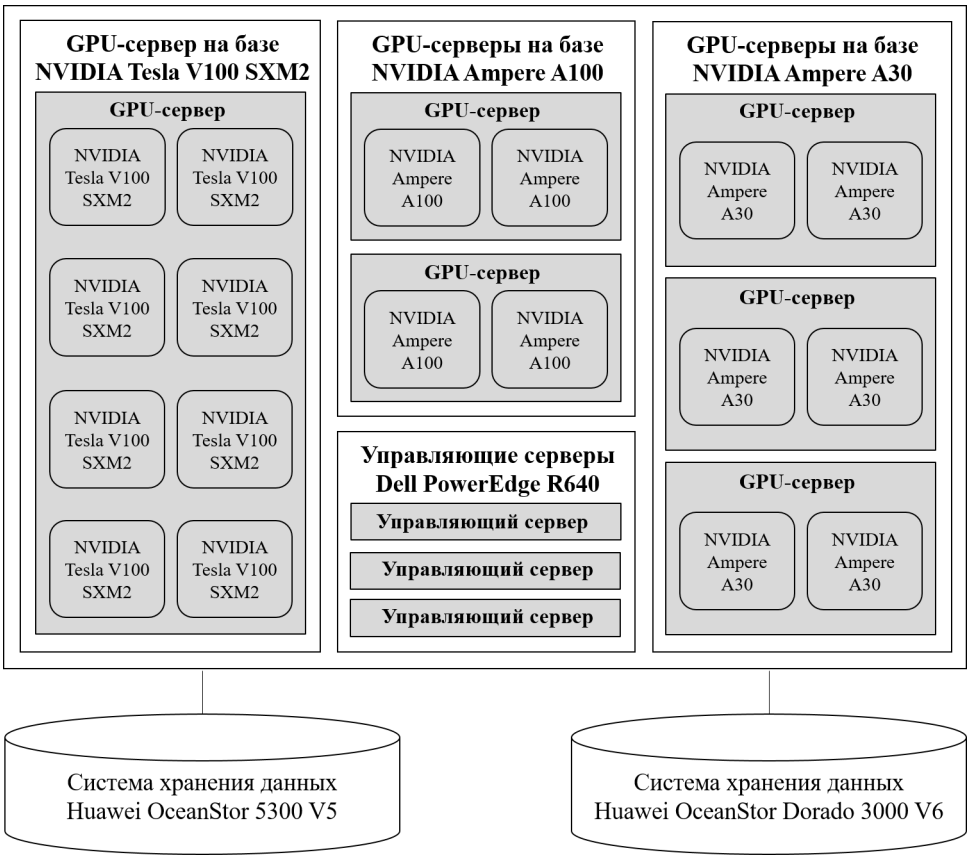


Рис. 1. Архитектура комплекса «Нейрокомпьютер»

сетей, встроенных в хранилище. Система включает в себя 12 твердотельных накопителей Enterprise SSD по 3.84 Тбайт каждый. Все компоненты хранилища являются собственной разработки компании Huawei. Объем хранилища составляет 46.08 Тбайт с кэшем 192 Гбайт. Эффективный объем хранилища, который могут использовать пользователи для своих данных, составляет не менее 35 Тбайт, остальное пространство используется для обеспечения отказоустойчивости хранилища.

Для долговременного хранения данных пользователей и результатов вычисления в комплексе «Нейрокомпьютер» используется система хранения данных Huawei OceanStor 5300 V5 с объемом памяти 700 Тбайт. Архитектура СХД Huawei OceanStor содержит два контролера, в каждый контролер встроен один процессор Kungpeng 920 с 64 Гбайт кэш-памяти, зеркалирование данных между контролерами осуществляется через сеть 100 Гбит/с. Система включает в себя жесткие диски NL-SAS 50 шт. по 14 Тбайт каждый. Поддержка большинства популярных протоколов, таких как NFS, CIFS, iSCSI и др., позволяет использовать данное хранилище как для комплекса «Нейрокомпьютер», так и для суперкомпьютера «Торнадо ЮУрГУ».

2. Системное программное обеспечение

На каждый узел суперкомпьютера «Торнадо ЮУрГУ» установлена операционная система CentOS 6.2. Используются компиляторы Intel Compiler (C/C++, Fortran 77, Fortran 90), GCC, библиотека параллельного программирования MPI2 (Intel MPI, OpenMPI, MVAPICH), позволяющие реализовывать пользователям собственные приложения для решения своих задач.

В комплексе «Нейрокомпьютер» установлена операционная система CentOS 7.8. Используются несколько разных версий компилятора GCC, библиотеки CUDA, NCCL и CUDNN для работы с графическими ускорителями, библиотека параллельного программирования OpenMPI. Для конфигурации пользовательского окружения Python-программы (установка необходимой версии Python, установка сопутствующих библиотек, обеспечивающих взаимодействие с искусственными нейронными сетями, Keras [3], Tensorflow [4] и др.) в комплексе «Нейрокомпьютер» установлена система Anaconda.

Для совместного эффективного использования ресурсов большим количеством пользователей на каждом вычислительном комплексе в НОЦ ИИКТ установлена отказоустойчивая и масштабируемая система управления кластерами и планирования заданий SLURM [5]. На суперкомпьютере «Торнадо» установлена SLURM версии 2.5.3, в комплексе «Нейрокомпьютер» используется SLURM версии 20.02.4.

2.1. Системы мониторинга

Одной из основных задач системного администратора является обеспечение корректной и бесперебойной работы оборудования, например, Infiniband и Ethernet сетей, систем хранения данных и др. Для системного мониторинга в НОЦ ИИКТ используются системы Nagios и Zabbix. В НОЦ ИИКТ также была разработана собственная система мониторинга загрузки суперкомпьютеров, которая позволяет формировать отчеты о загрузке и деятельности пользователей из структурных подразделений университета [6].

Система Nagios предоставляет информацию о работоспособности оборудования и программных сервисов и формирует сообщение об изменении состояния, отправляемое по электронной почте администратору [7]. Основной подход к написанию проверок в Nagios — описание проверок в виде скриптов с помощью самостоятельно написанного кода, а также с помощью стандартного набора базовых проверок, которые не подходят для организации мониторинга всех компонентов ресурсов НОЦ ИИКТ. За счет возможности самостоятельного написания простых проверок Nagios позволяет гибко выполнять проверки различных сервисов, но это также является серьезным недостатком. Необходимо самостоятельно описывать все нестандартные проверки, что требует проанализировать SNMP команды оборудования и описать их обработчики. Оповещение через электронную почту также неудобный механизм, зачастую письма попадают в спам либо доходят позднее, чем необходимо для быстрого реагирования на проблемы. В НОЦ ИИКТ Nagios используется для проверки состояния сервисов, таких как почта, очереди задач, и сетевого, коммуникационного и других видов оборудования (рис. 2).

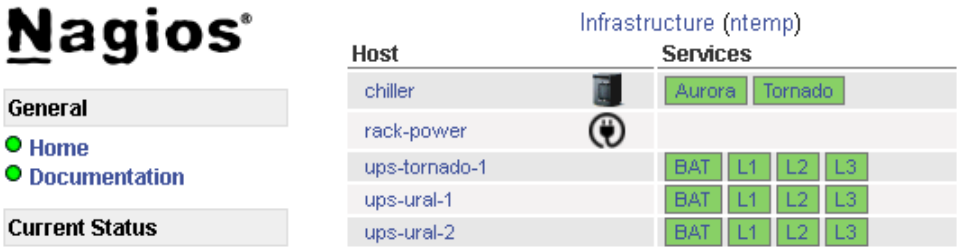


Рис. 2. Мониторинг состояния оборудования с помощью веб-интерфейса системы Nagios

Система мониторинга Zabbix [8] появилась в НОЦ ИИКТ позднее, чем Nagios, но благодаря возможностям активного и пассивного мониторинга систем, установки на Windows

и Linux системы используется более широко. В Zabbix предусмотрено создание графиков для отдельных групп оборудования (рис. 3), большая пополняемая база стандартных проверок для оборудования (например, для системы хранения Dorado и др.), что позволяет настроить мониторинг нового оборудования в кратчайшие сроки. Полезным инструментом Zabbix также являются графики, они позволяют сравнивать параметры, мониторинг которых проводился в различные временные промежутки, выявлять зависимости и проблемы. В дополнение к перечисленному существенным преимуществом системы Zabbix является возможность интеграции с популярными мессенджерами и корпоративными системами, что позволяет достигнуть практически мгновенного реагирования на возникшее изменение состояния оборудования или программного сервиса. В настоящее время в НОЦ ИИКТ активно внедряется Zabbix, заменяя большую часть функционала Nagios, но полностью покрыть функциональность Nagios система Zabbix пока не может. Схема инфраструктуры НОЦ ИИКТ в системе мониторинга Zabbix представлена на рис. 4.

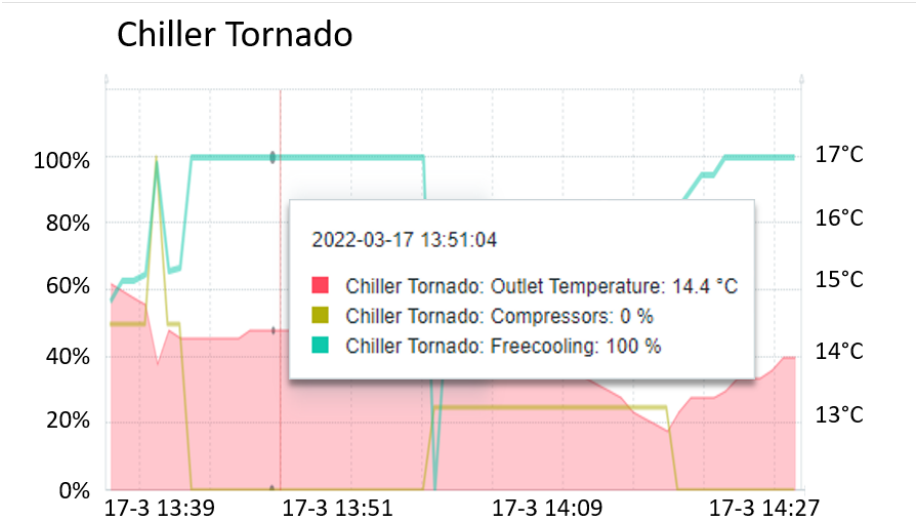


Рис. 3. Мониторинг состояния чиллера суперкомпьютера Торнадо с помощью веб-интерфейса системы Zabbix

2.2. Системы управления

Для установки и конфигурирования программного обеспечения большого количества вычислительных узлов, как на суперкомпьютере «Торнадо ЮурГУ», в НОЦ ИИКТ используются специализированные программные системы управления xCAT и Puppet.

xCAT (Extreme Cluster Administration Tool) — масштабируемый инструментарий для развертывания и сопровождения больших кластеров [9]. xCAT предоставляет унифицированный интерфейс для управления аппаратным оборудованием, обнаружением и развертыванием diskful/diskless операционных систем. Все команды являются клиент-серверными, поддерживают аутентификацию, протоколируются и управляются политиками. xCAT поддерживает разграничение прав на основе политик доступа. В клиент-серверном приложении xCAT весь поток между клиентом и сервером контролируется службой xcatd (xCAT daemon) на управляющем узле (Management Node). Когда служба xcatd получает упакованную как XML команду, она проверяет полномочия отправителя, сверяясь со списками

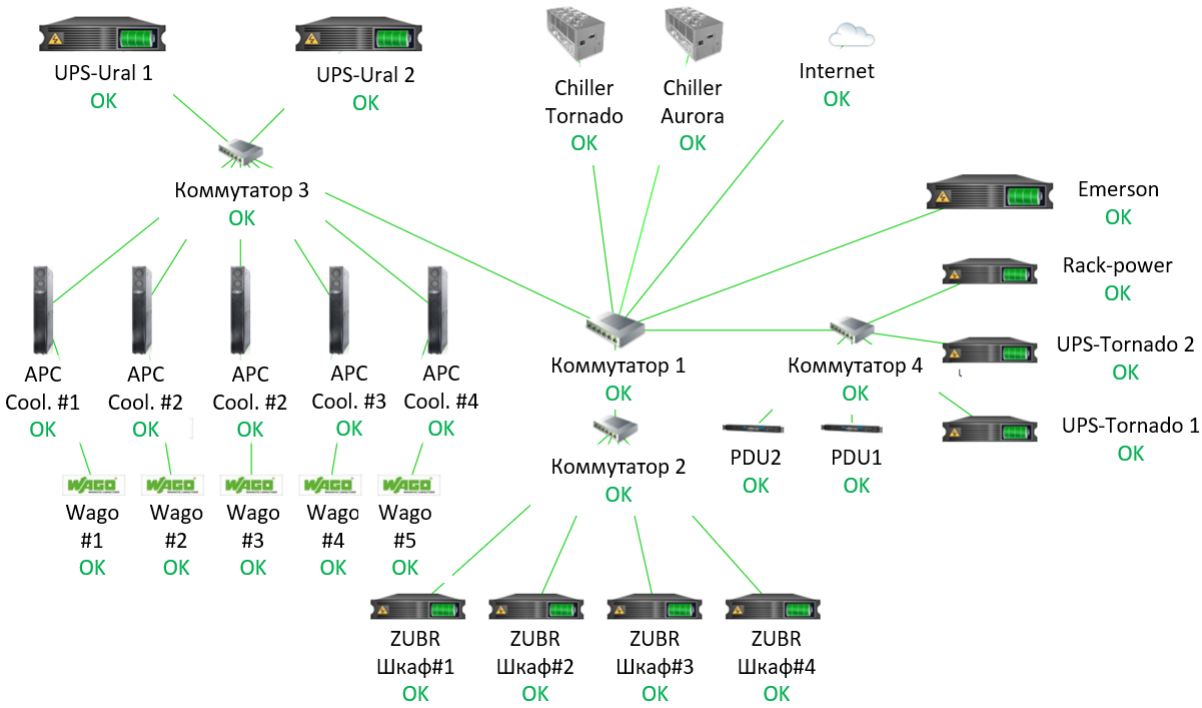


Рис. 4. Схема инфраструктуры НОЦ ИИКТ в системе мониторинга Zabbix

контроля доступа ACL в таблице политик. Также служба получает информацию о состоянии и статусе узлов с момента начала их работы. xCAT спроектирован для масштабирования очень больших кластеров. Благодаря поддержке иерархии, один управляющий узел может иметь любое количество stateless или statefull сервисных узлов, что повышает производительность и позволяет управлять очень большими кластерами. В НОЦ ИИКТ xCAT используется для установки кластерной операционной системы на вычислительные узлы, посредством PXE загрузки по DHCP, с первоначальной установкой и конфигурацией операционной системы и запуском фонового процесса системы Puppet для дальнейшей конфигурации.

Puppet представляет собой систему управления конфигурацией и язык для описания задач конфигурирования. Puppet используется системными администраторами для эффективного управления большим количеством систем и обеспечения единой конфигурации. В НОЦ ИИКТ с помощью Puppet настраивается вся конфигурация узла после его базовой установки с помощью xCAT, а именно, установка очереди задач SLURM, пакетов для работы с высокоскоростной сетью Infiniband, хранилищем и системных пакетов. Такая организация установки системы позволяет гибко вносить изменения в конфигруацию системы в случае необходимости.

3. Прикладное программное обеспечение

На высокопроизводительных вычислительных ресурсах ЮУрГУ установлено прикладное программное обеспечение как проприетарное, так и свободное [10], в т.ч.:

– многоцелевой конечно-элементный пакет для проведения анализа в широком круге инженерных дисциплин ANSYS [11];

– многоцелевой конечно-элементный комплекс разработки для анализа высоконелинейных и быстротекущих процессов в задачах механики твердого и жидкого тела LS-DYNA [12];
– комплексное решение в области моделирования трехмерных турбулентных течений жидкости и газа FlowVision [13];
– специализированный инженерный программный комплекс, предназначенный для анализа процессов обработки металлов давлением, термической и механической обработки SFTC DEFORM [14];
– специализированный пакет для решения инженерных, научно-технических и экономических задач MathWorks MATLAB [15];
– открытая интегрируемая платформа для численного моделирования задач механики сплошных сред OpenFOAM [16].

На вычислительных ресурсах НОЦ ИИКТ ЮУрГУ ежегодно выполняется более 250 научных задач из различных областей — искусственный интеллект, машиностроение, металлургия и металлообработка, топливно-энергетический комплекс, легкая промышленность, производство суперкомпьютеров и программного обеспечения, в т.ч.:

– мониторинг выбросов загрязняющих веществ от автотранспорта в режиме реального времени [17]
– изучение реакции нормализованного вегетационного индекса (NDVI) ландшафтов в нижнем бассейне Тигра на текущую глобальную и региональную изменчивость климата [18]
– моделирование сжимаемости изоструктурных галогенсодержащих кристаллов на макро- и микроуровнях[19],
– изучение квантового электронного давления и сжимаемости кристаллов для диборида магния при моделируемом сжатии [20],
– механизм реакции разложения Тетракиса с помощью реактивной динамики [21],
– первопринципное компьютерное моделирование в системах Fe-C [22],
– моделирование полосно-пропускающих фильтров на основе многослойной технологии [23],
– валидатор решений задач линейного программирования VaLiPro [24],
– параллельный подход для поиска аномалий в больших временных рядах [25],
– использование концепции матричного профиля в реляционную СУБД для интеллектуального анализа временных рядов [26],
– поверхностная обработка арамидной ткани и ее влияние на механику фрикционного взаимодействия нитей [27],
– разработка суперкомпьютерной модели иглопробивного войлока [28],
– пластическая деформация при динамическом уплотнении нанопорошка алюминия: моделирование молекулярной динамики и механическая модель [29],
– компьютерное моделирование поведения представительного объема порошкового материала [30],
– PaaS-решения для туманных вычислений [31],
– концепция и классификация платформ для поддержки распределенных вычислительных систем для туманных вычислений [32],
– обеспечение безопасности уязвимых участников дорожного движения [33],
– синтез речи на русском языке [34],

– система учета посещаемости студентов на основе методов компьютерного зрения [35],
– оценка транспортного потока на основе данных с камеры видеонаблюдения [36].

Заключение

В настоящее время в Южно-Уральском государственном университете активно развиваются исследования, связанные с суперкомпьютерными технологиями. Созданный в ЮУрГУ научно-образовательный центр «Искусственный интеллект и квантовые технологии» предоставляет доступ к высокопроизводительным ресурсам более 500 пользователям, которые являются студентами и сотрудниками ЮУрГУ и внешних образовательных, научных и производственных организаций. Суперкомпьютер «Торнадо ЮУрГУ», комплекс «Нейрокомпьютер», системы хранения данных Panasas ActiveStor 11, OceanStor Dorado 3000 V6, Huawei OceanStor 5300 V5 администрируются сотрудниками НОЦ ИИКТ: выполняется поддержание работоспособности очереди задач SLURM, обеспечение корректной и бесперебойной работы Infiniband и Ethernet сетей, хранилищ и другого оборудования, с помощью специализированных систем мониторинга и управления. На высокопроизводительных ресурсах ЮУрГУ установлено современное параллельное программное обеспечение, позволяющее выполнять научно-исследовательские и опытно-конструкторские работы из разных областей знаний.

Исследование выполнено при финансовой поддержке Министерства науки и высшего образования РФ (государственное задание FENU-2020-0022) и Российского фонда фундаментальных исследований (грант № 20-07-00140).

Литература

1. Научно-образовательный центр искусственный интеллект и квантовые технологии ЮУрГУ. URL: `https://supercomputer.susu.ru/` (дата обращения: 11.03.2022).
2. Абрамов С.М., Заднепровский В.Ф., Лилитко Е.П. Суперкомпьютеры «СКИФ» ряда 4 // Информационные технологии и вычислительные системы. 2012. № 1. С. 3—16.
3. Keras. Documentation. URL: `https://keras.io/guides/` (дата обращения: 10.02.2022).
4. Tensorflow. Documentation. URL: `https://www.tensorflow.org/` (дата обращения: 10.02.2022).
5. Slurm. Documentation. URL: `https://slurm.schedmd.com/documentation.html` (дата обращения: 10.02.2022).
6. Kostenetskiy P., Semenikhina P. SUSU Supercomputer Resources for Industry and fundamental Science // 2018 Global Smart Industry Conference (GloSIC), Chelyabinsk, Russia, November 13–15, 2018. IEEE, 2018. P. 1–7. DOI: `10.1109/GloSIC.2018.8570068`.
7. Borghesi A., Molan M., Milano M., Bartolini A. Anomaly Detection and Anticipation in High Performance Computing Systems // IEEE Transactions on Parallel and Distributed Systems. 2022. Vol. 33, no. 4. P. 739–750. DOI: `10.1109/TPDS.2021.3082802`.
8. Борисов С.Н., Зима А.М., Дьяченко Р.А., Елизаров П.В. Обзор современных информационных систем мониторинга сетей передачи данных // Современная наука: Актуальные проблемы теории и практики. Серия: Естественные и технические науки. 2019. № 5. С. 29—34.

9. Lascu O., Brindeyev A., Quintero D.E. и др. xCAT 2 Guide for the CSM System Administrator. 2008. URL: https://www.redbooks.ibm.com/redpapers/pdfs/redp4437.pdf (дата обращения: 27.02.2022).

10. НОЦ ИИКТ ЮУрГУ. Прикладное программное обеспечение. URL: http://supercomputer.susu.ru/users/simulation/.

11. ANSYS. URL: http://ansys.com (дата обращения: 24.02.2022).

12. LSTC LS-DYNA. URL: http://www.ls-dyna.com/ (дата обращения: 24.02.2022).

13. FlowVision. URL: https://tesis.com.ru/own_design/flowvision/ (дата обращения: 24.02.2022).

14. SFTC DEFORM. URL: https://tesis.com.ru/cae_brands/deform/ (дата обращения: 24.02.2022).

15. MATLAB. URL: https://www.mathworks.com/ (дата обращения: 24.02.2022).

16. OpenFOAM. URL: https://www.openfoam.com/ (дата обращения: 24.02.2022).

17. Shepelev V., Zhankaziev S., Aliukov S., *et al.* Forecasting the Passage Time of the Queue of Highly Automated Vehicles Based on Neural Networks in the Services of Cooperative Intelligent Transport Systems // Mathematics. 2022. Vol. 10, no. 2. Article 282. DOI: 10.3390/math10020282.

18. Alhumaima A., Abdullaev S. Tigris basin landscapes: Sensitivity of vegetation index ndvi to climate variability derived from observational and reanalysis data // Earth Interactions. 2020. Vol. 24, no. 7. P. 1–18. DOI: 10.1175/EI-D-20-0002.1.

19. Bartashevich E., Sobalev S., Matveychuk Y., Tsirelson V. Simulation of the compressibility of isostructural halogen containing crystals on macro- and microlevels // Journal of Structural Chemistry. 2021. Vol. 62. P. 1607–1620. DOI: 10.1134/S0022476621100164.

20. Matveychuk Y.V., Bartashevich E.V., Skalyova K.K., Tsirelson V.G. Quantum electronic pressure and crystal compressibility for magnesium diboride under simulated compression // Materials Today Communications. 2021. Vol. 26. P. 101952. DOI: 10.1016/j.mtcomm.2020.101952.

21. Zybin S.V., Morozov S.I., Prakash P., *et al.* Reaction Mechanism and Energetics of Decomposition of Tetrakis(1,3-dimethyltetrazol-5-imidoperchloratomanganese(II)) from Quantum-Mechanics-based Reactive Dynamics // Journal of the American Chemical Society. 2021. Vol. 143, no. 41. P. 16960–16975. DOI: 10.1021/jacs.1c04847.

22. Mirzoev A.A., Ridnyi Y.M. Ab initio calculation of total energy of a bcc iron cell containing three dissolved carbon atoms, and internal friction in Fe–C solid solutions // Journal of Alloys and Compounds. 2021. Vol. 883. P. 160850. DOI: 10.1016/j.jallcom.2021.160850.

23. Fomin D.G., Dudarev N.V., Darovskikh S.N. Duplexer Based on Volumetric Modular Technology // 2021 IEEE 22nd International Conference of Young Professionals in Electron Devices and Materials (EDM), Souzga, the Altai Republic, Russia, June 30–July 4, 2021. IEEE, 2021. P. 97–100. DOI: 10.1109/EDM52169.2021.9507637.

24. Sokolinsky L.B., Sokolinskaya I.M. VaLiPro: Linear Programming Validator for Cluster Computing Systems // Supercomputing Frontiers and Innovations. 2021. Vol. 8, no. 3. P. 51–61. DOI: 10.14529/jsfi210303.

25. Zymbler M., Grents A., Kraeva Y., Kumar S. A Parallel Approach to Discords Discovery in Massive Time Series Data // Computers, Materials and Continua. 2021. Vol. 66, no. 2. P. 1867–1876. DOI: 10.32604/cmc.2020.014232.

26. Zymbler M., Ivanova E. Matrix Profile-Based Approach to Industrial Sensor Data Analysis Inside RDBMS // Mathematics. 2021. Vol. 9, no. 17. Article 2146. DOI: 10.3390/math9172146.

27. Ignatova A.V., Dolganina N.Y., Sapozhnikov S.B., Shabley A.A. Aramid fabric surface treatment and its impact on the mechanics of yarn's frictional interaction // PNRPU Mechanics Bulletin. 2017. No. 4. P. 121–137.

28. Dolganina N.Y., Teleshova E.A., Semenikhina P.N. Development of Supercomputer Model of Needle-Punched Felt // 2020 Global Smart Industry Conference (GloSIC), Chelyabinsk, Russia, November 17–19, 2020. IEEE, 2020. P. 1–6. DOI: 10.1109/GloSIC50886.2020.9267856.

29. Mayer A.E., Ebel A.A., Al-Sandoqachi M.K. Plastic deformation at dynamic compaction of aluminum nanopowder: Molecular dynamics simulations and mechanical model // International Journal of Plasticity. 2020. Vol. 124. P. 22–41. DOI: https://doi.org/10.1016/j.ijplas.2019.08.005.

30. Иванов В.А. Микромеханическая модель представительного объема порошковых материалов // Вестник Южно-Уральского государственного университета. Серия: Металлургия. 2021. Т. 21, № 3. С. 67—81. DOI: 10.14529/met210308.

31. Vetoshkin N., Radchenko G. Towards the Fog Computing PaaS Solution // 2020 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT), Yekaterinburg, Russia, May 14–15, 2020. IEEE, 2020. P. 0516–0519. DOI: 10.1109/USBEREIT48449.2020.9117791.

32. Kirsanova A.A., Radchenko G.I., Tchernykh A.N. Fog Computing State of the Art: Concept and Classification of Platforms to Support Distributed Computing Systems // Supercomputing Frontiers and Innovations. 2021. Vol. 8, no. 3. P. 17–50. DOI: 10.14529/jsfi210302.

33. Pustokhina I.V., Pustokhin D.A., Vaiyapuri T., *et al.* An automated deep learning based anomaly detection in pedestrian walkways for vulnerable road users safety // IEEE Transactions on Parallel and Distributed Systems. 2021. Vol. 142, no. 4. P. 105356. DOI: 10.1016/j.ssci.2021.105356.

34. Kuzmin A.D., Ivanov S.A. Transfer Learning for the Russian Language Speech Synthesis // 2021 International Conference on Quality Management, Transport and Information Security, Information Technologies (IT QM IS), Yaroslavl, Russian Federation, September 6–10, 2021. IEEE, 2021. P. 507–510. DOI: 10.1109/ITQMIS53292.2021.9642715.

35. Strueva A.Y., Ivanova E.V. Student Attendance Control System with Face Recognition Based on Neural Network // 2021 International Russian Automation Conference (RusAutoCon), Sochi, Russian Federation, September 5–11, 2021. IEEE, 2021. P. 929–933. DOI: 10.1109/RusAutoCon52004.2021.9537386.

36. Fedorov A., Nikolskaia K., Ivanov S., *et al.* Traffic flow estimation with data from a video surveillance camera // Journal of Big Data. 2019. Vol. 6. Article 73. DOI: 10.1186/s40537-019-0234-z.

Биленко Роман Владимирович, лаборант НОЦ ИИКТ, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Долганина Наталья Юрьевна, к.т.н., директор НОЦ ИИКТ, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Иванова Елена Владимировна, к.ф.-м.н., руководитель лаборатории суперкомпьютерного моделирования НОЦ ИИКТ, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Рекачинский Александр Игоревич, директор суперкомпьютерного центра НОЦ ИИКТ, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse220102

HIGH-PERFORMANCE COMPUTING RESOURCES OF SOUTH URAL STATE UNIVERSITY

© 2022 R.V. Bilenko, N.Yu. Dolganina, E.V. Ivanova, A.I. Rekachinsky

South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)

E-mail: bilenkorv@susu.ru, dolganinani@susu.ru,
elena.ivanova@susu.ru, alexander.rekachinsky@susu.ru

Received: 20.02.2022

Currently, South Ural State University has achieved significant results in the field of supercomputer modeling, artificial intelligence and big data. SUSU has an energy-efficient supercomputer "Tornado SUSU", which ranks 15th in the ranking of the most powerful supercomputers of the CIS TOP50 (September 2021). For research in artificial neural networks, a specialized multiprocessor "Neurocomputer" complex was installed at SUSU. The "Neurocomputer" uses powerful advanced graphics accelerators to train neural networks. The supercomputer and the "Neurocomputer" are at the center of the scientific life of the University, allowing for the most complex calculations in the field of engineering, natural sciences, human sciences and artificial intelligence. SUSU computing resources are used in education and for commercial purposes to calculate the tasks of the University's partners. The paper describes the characteristics of high-performance SUSU equipment, available system and application parallel software, provides information about solved scientific and engineering tasks.

Keywords: supercomputer, neurocomputer, parallel storage system, supercomputer administration, supercomputer modeling, neural networks.

FOR CITATION

Bilenko R.V., Dolganina N.Yu., Ivanova E.V., Rekachinsky A.I. High-performance Computing Resources of South Ural State University. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2022. Vol. 11, no. 1. P. 15–30. (in Russian) DOI: 10.14529/cmse220102.

References

1. SUSU Research and Educational Center for Artificial Intelligence and Quantum Technologies. URL: https://supercomputer.susu.ru/ (accessed: 11.03.2022).
2. Abramov S.M., Zadneprovskiy V.F., Lilitko E.P. Supercomputers "SKIF" series 4. Information technologies and computing systems. 2012. No. 1. P. 3–16.
3. Keras. Documentation. URL: https://keras.io/guides/ (accessed: 10.02.2022).
4. Tensorflow. Documentation. URL: https://www.tensorflow.org/ (accessed: 10.02.2022).
5. Slurm. Documentation. URL: https://slurm.schedmd.com/documentation.html (accessed: 10.02.2022).
6. Kostenetskiy P., Semenikhina P. SUSU Supercomputer Resources for Industry and fundamental Science. 2018 Global Smart Industry Conference (GloSIC), Chelyabinsk, Russia, November 13–15, 2018. IEEE, 2018. P. 1–7. DOI: 10.1109/GloSIC.2018.8570068.
7. Borghesi A., Molan M., Milano M., Bartolini A. Anomaly Detection and Anticipation in High Performance Computing Systems. IEEE Transactions on Parallel and Distributed Systems. 2022. Vol. 33, no. 4. P. 739–750. DOI: 10.1109/TPDS.2021.3082802.
8. Borisov S.N., Zima A.M., Dyachenko R.A., Elizarov P.V. Review of modern information monitoring systems for data networks. Modern science: Actual problems of theory and practice. Series: Natural and technical sciences. 2019. No. 5. P. 29–34. (in Russian).
9. Lascu O., Brindeyev A., Quintero D.E., *et al.* xCAT 2 Guide for the CSM System Administrator. 2008. URL: https://www.redbooks.ibm.com/redpapers/pdfs/redp4437.pdf (accessed: 27.02.2022).
10. SUSU REC AIQT. Application software. URL: http://supercomputer.susu.ru/users/simulation/ (in Russian).
11. ANSYS. URL: http://ansys.com (accessed: 24.02.2022).
12. LSTC LS-DYNA. URL: http://www.ls-dyna.com/ (accessed: 24.02.2022).
13. FlowVision. URL: https://tesis.com.ru/own_design/flowvision/ (accessed: 24.02.2022).
14. SFTC DEFORM. URL: https://tesis.com.ru/cae_brands/deform/ (accessed: 24.02.2022).
15. MATLAB. URL: https://www.mathworks.com/ (accessed: 24.02.2022).
16. OpenFOAM. URL: https://www.openfoam.com/ (accessed: 24.02.2022).
17. Shepelev V., Zhankaziev S., Aliukov S., *et al.* Forecasting the Passage Time of the Queue of Highly Automated Vehicles Based on Neural Networks in the Services of Cooperative Intelligent Transport Systems. Mathematics. 2022. Vol. 10, no. 2. Article 282. DOI: 10.3390/math10020282.
18. Alhumaima A., Abdullaev S. Tigris basin landscapes: Sensitivity of vegetation index ndvi to climate variability derived from observational and reanalysis data. Earth Interactions. 2020. Vol. 24, no. 7. P. 1–18. DOI: 10.1175/EI-D-20-0002.1.
19. Bartashevich E., Sobalev S., Matveychuk Y., Tsirelson V. Simulation of the compressibility of isostructural halogen containing crystals on macro- and microlevels. Journal of Structural Chemistry. 2021. Vol. 62. P. 1607–1620. DOI: 10.1134/S0022476621100164.

20. Matveychuk Y.V., Bartashevich E.V., Skalyova K.K., Tsirelson V.G. Quantum electronic pressure and crystal compressibility for magnesium diboride under simulated compression. Materials Today Communications. 2021. Vol. 26. P. 101952. DOI: https://doi.org/10.1016/j.mtcomm.2020.101952.

21. Zybin S.V., Morozov S.I., Prakash P., *et al.* Reaction Mechanism and Energetics of Decomposition of Tetrakis(1,3-dimethyltetrazol-5-imidoperchloratomanganese(II)) from Quantum-Mechanics-based Reactive Dynamics. Journal of the American Chemical Society. 2021. Vol. 143, no. 41. P. 16960–16975. DOI: 10.1021/jacs.1c04847.

22. Mirzoev A.A., Ridnyi Y.M. Ab initio calculation of total energy of a bcc iron cell containing three dissolved carbon atoms, and internal friction in Fe–C solid solutions. Journal of Alloys and Compounds. 2021. Vol. 883. P. 160850. DOI: https://doi.org/10.1016/j.jallcom.2021.160850.

23. Fomin D.G., Dudarev N.V., Darovskikh S.N. Duplexer Based on Volumetric Modular Technology. 2021 IEEE 22nd International Conference of Young Professionals in Electron Devices and Materials (EDM), Souzga, the Altai Republic, Russia, June 30–July 4, 2021. IEEE, 2021. P. 97–100. DOI: 10.1109/EDM52169.2021.9507637.

24. Sokolinsky L.B., Sokolinskaya I.M. VaLiPro: Linear Programming Validator for Cluster Computing Systems. Supercomputing Frontiers and Innovations. 2021. Vol. 8, no. 3. P. 51–61. DOI: 10.14529/jsfi210303.

25. Zymbler M., Grents A., Kraeva Y., Kumar S. A Parallel Approach to Discords Discovery in Massive Time Series Data. Computers, Materials and Continua. 2021. Vol. 66, no. 2. P. 1867–1876. DOI: 10.32604/cmc.2020.014232.

26. Zymbler M., Ivanova E. Matrix Profile-Based Approach to Industrial Sensor Data Analysis Inside RDBMS. Mathematics. 2021. Vol. 9, no. 17. Article 2146. DOI: 10.3390/math9172146.

27. Ignatova A.V., Dolganina N.Y., Sapozhnikov S.B., Shabley A.A. Aramid fabric surface treatment and its impact on the mechanics of yarn's frictional interaction. PNRPU Mechanics Bulletin. 2017. No. 4. P. 121–137. URL: https://ered.pstu.ru/index.php/mechanics/article/view/118.

28. Dolganina N.Y., Teleshova E.A., Semenikhina P.N. Development of Supercomputer Model of Needle-Punched Felt. 2020 Global Smart Industry Conference (GloSIC), Chelyabinsk, Russia, November 17–19, 2020. IEEE, 2020. P. 1–6. DOI: 10.1109/GloSIC50886.2020.9267856.

29. Mayer A.E., Ebel A.A., Al-Sandoqachi M.K. Plastic deformation at dynamic compaction of aluminum nanopowder: Molecular dynamics simulations and mechanical model. International Journal of Plasticity. 2020. Vol. 124. P. 22–41. DOI: https://doi.org/10.1016/j.ijplas.2019.08.005.

30. Ivanov V.A. Micromechanical model of representative volume of powders material. Bulletin of the South Ural State University. Series: Metallurgy. 2021. Vol. 21, no. 3. P. 67–81. DOI: 10.14529/met210308.

31. Vetoshkin N., Radchenko G. Towards the Fog Computing PaaS Solution. 2020 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT), Yekaterinburg, Russia, May 14–15, 2020. IEEE, 2020. P. 0516–0519. DOI: 10.1109/USBEREIT48449.2020.9117791.

32. Kirsanova A.A., Radchenko G.I., Tchernykh A.N. Fog Computing State of the Art: Concept and Classification of Platforms to Support Distributed Computing Systems. Supercomputing Frontiers and Innovations. 2021. Vol. 8, no. 3. P. 17–50. DOI: 10.14529/jsfi210302.

33. Pustokhina I.V., Pustokhin D.A., Vaiyapuri T., *et al.* An automated deep learning based anomaly detection in pedestrian walkways for vulnerable road users safety. IEEE Transactions on Parallel and Distributed Systems. 2021. Vol. 142, no. 4. P. 105356. DOI: 10.1016/j.ssci.2021.105356.

34. Kuzmin A.D., Ivanov S.A. Transfer Learning for the Russian Language Speech Synthesis. 2021 International Conference on Quality Management, Transport and Information Security, Information Technologies (IT QM IS), Yaroslavl, Russian Federation, September 6–10, 2021. IEEE, 2021. P. 507–510. DOI: 10.1109/ITQMIS53292.2021.9642715.

35. Strueva A.Y., Ivanova E.V. Student Attendance Control System with Face Recognition Based on Neural Network. 2021 International Russian Automation Conference (RusAutoCon), Sochi, Russian Federation, September 5–11, 2021. IEEE, 2021. P. 929–933. DOI: 10.1109/RusAutoCon52004.2021.9537386.

36. Fedorov A., Nikolskaia K., Ivanov S., *et al.* Traffic flow estimation with data from a video surveillance camera. Journal of Big Data. 2019. Vol. 6. Article 73. DOI: 10.1186/s40537-019-0234-z.

УДК 519.852, 004.032.26 **DOI: 10.14529/cmse220103**

ВИЗУАЛЬНОЕ ПРЕДСТАВЛЕНИЕ МНОГОМЕРНЫХ ЗАДАЧ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

© 2022 Н.А. Ольховский, Л.Б. Соколинский

Южно-Уральский государственный университет
(454080 Челябинск, пр. им. В.И. Ленина, д. 76)
E-mail: olkhovskiina@susu.ru, leonid.sokolinsky@susu.ru

Поступила в редакцию: 10.03.2022

В статье строится n-мерная математическая модель визуального представления задачи линейного программирования. Эта модель позволит использовать аппарат искусственных нейронных сетей для решения многомерных задач линейной оптимизации, допустимая область которых является ограниченным непустым множеством. Для визуализации задачи линейного программирования вводится целевая гиперплоскость, ориентация которой определяется градиентом линейной целевой функции: градиент является нормалью к целевой гиперплоскости. В случае поиска максимума целевая гиперплоскость располагается таким образом, чтобы значение целевой функции во всех ее точках превосходило значение целевой функции во всех точках допустимой области, представляющей собой ограниченный выпуклый многогранник. Для произвольной точки целевой гиперплоскости определяется целевая проекция на многогранник: чем ближе точка целевой проекции к целевой гиперплоскости, тем больше значение целевой функции в этой точке. На основе целевой гиперплоскости строится конечное регулярное множество точек, называемое рецептивным полем. С помощью целевых проекций строится образ многогранника, включающий в себя точки рецептивного поля и расстояния до соответствующих точек поверхности многогранника. На основе предложенной модели строится параллельный алгоритм визуализации задачи линейного программирования. Дается аналитическая оценка его масштабируемости. Приводятся сведения о программной реализации и результаты масштабных вычислительных экспериментов, подтверждающие эффективность предложенных подходов.

Ключевые слова: линейное программирование, n-мерная визуализация, математическая модель, параллельный алгоритм, BSF-каркас.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

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.

Введение

Быстрое развитие технологий накопления и обработки больших данных [1, 2] привело к появлению оптимизационных математических моделей в виде сверхбольших задач линейного программирования (ЛП) [3]. Такие задачи возникают в индустрии, экономике, логистике, статистике, квантовой физике и других областях [4—8]. Классическое программное обеспечение во многих случаях не позволяет решить подобные масштабные задачи ЛП за приемлемое время [9]. Вместе с тем в ближайшие 2–3 года появятся вычислительные системы экзафлопсного уровня производительности [10], потенциально способные решать подобные задачи. В соответствии с этим актуальной является задача разработки новых эффективных методов для решения сверхбольших задач ЛП с помощью экзамасштабных вычислительных систем.

До настоящего времени одним из самых распространенных способов решения задачи ЛП являлся класс алгоритмов, предложенных и разработанных Данцигом на основе симплекс-метода [11]. Симплекс-метод оказался эффективным для решения большого клас-

са задач ЛП. Однако симплекс-метод имеет некоторые фундаментальные особенности, ограничивающие его применение для больших задач ЛП. Во-первых, в определенных случаях симплекс-методу приходится перебирать все вершины симплекса, что соответствует экспоненциальной временной сложности [12]. Во-вторых, симплекс-метод в большинстве случаев удовлетворительно решает задачи ЛП, содержащие до 50000 переменных, однако на больших задачах часто наблюдается потеря точности, которая не может быть компенсирована даже путем применения таких ресурсоемких процедур, как «аффинное масштабирование» или «итерационное уточнение» [13]. В-третьих, в общем случае симплекс метод плохо масштабируется на многопроцессорных системах с распределенной памятью. Были предприняты многочисленные попытки построить масштабируемую реализацию симплекс-метода, однако они не увенчались успехом [14]. Кармаркар предложил метод внутренних точек [15], способный решать сверхбольшие задачи ЛП с миллионами переменных и миллионами уравнений [16]. Более того, алгоритмы, основанные на методе внутренних точек, являются самокорректирующимися, и поэтому обеспечивают высокую точность вычислений. В качестве недостатка метода внутренних точек следует отметить тот факт, что для начала работы алгоритма необходимо иметь точку, удовлетворяющую всем ограничениям задачи ЛП. Нахождение такой внутренней точки может сводиться к решению дополнительной задачи ЛП. В качестве альтернативы можно указать метод псевдопроекций, основанный на использовании фейеровских отображений [17]. Еще одним существенным недостатком метода внутренних точек является его плохая масштабируемость применительно к многопроцессорным системам с распределенной памятью. Было сделано несколько попыток построить параллельную реализацию для частных случаев (см., например, [18, 19]), но эффективную параллельную реализацию для экзамасштабных многопроцессорных систем в общем случае построить не удалось. В соответствии с этим является актуальным направление исследований, связанное с поиском новых масштабируемых методов решения задач ЛП.

Возможной эффективной альтернативой классическим методам ЛП являются методы оптимизации, основанные на нейросетевых технологиях. Искусственные нейронные сети [20, 21] являются одним из самых обещающих и быстро развивающихся направлений современных информационных технологий. Нейронные сети представляют собой универсальный инструмент, способный решать задачи практически во всех областях. Особенно большие успехи достигнуты в распознавании и анализе изображений с помощью сверточных нейронных сетей [22]. Однако, в научной периодике практически отсутствуют работы, посвященные применению сверточных нейронных сетей для решения задач линейной оптимизации [23]. Это связано с тем, что сверточные нейронные сети ориентированы на распознавание и анализ изображений, но в научной литературе отсутствуют работы по визуальному представлению многомерных задач линейного программирования. Таким образом, вопрос разработки новых нейросетевых моделей и методов применительно к линейной оптимизации остается открытым.

В этой статье мы предприняли попытку построить n-мерную математическую модель визуального представления задачи ЛП, которая позволит использовать аппарат искусственных нейронных сетей для решения многомерных задач линейной оптимизации, допустимая область которых является ограниченным непустым множеством. Метод визуализации, основанный на описываемой модели, обладает высокой вычислительной сложностью. Поэтому мы предлагаем его реализацию в виде параллельного алгоритма, ориентированного на кластерные вычислительные системы. Статья организована следующим образом. Раз-

дел 1 посвящен построению математической модели визуального представления многомерной задачи ЛП. В разделе 2 описывается реализация предложенного метода визуализации в виде параллельного алгоритма и дается аналитическая оценка его масштабируемости. В разделе 3 приводится информация о программной реализации описанного параллельного алгоритма и обсуждаются результаты масштабных вычислительных экспериментов на кластерной вычислительной системе. В заключении суммируются полученные результаты и приводятся направления дальнейших исследований.

1. Математическая модель визуального представления задачи ЛП

Рассмотрим задачу ЛП в следующей форме:

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

где $c, b \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$ и $c \neq \mathbf{0}$. Здесь $\langle\cdot,\cdot\rangle$ обозначает скалярное произведение двух векторов. Мы предполагаем, что ограничение $x \geqslant \mathbf{0}$ также включено в систему $Ax \leqslant b$ в виде неравенств

$$\begin{array}{ccccccccccc}
-x_1 & + & 0 & + & \cdots & \cdots & \cdots & + & 0 & \leqslant & 0;\\
0 & - & x_2 & + & 0 & + & \cdots & + & 0 & \leqslant & 0;\\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots\\
0 & + & \cdots & \cdots & \cdots & + & 0 & - & x_n & \leqslant & 0.
\end{array}$$

Вектор c является градиентом линейной целевой функции

$$f(x) = c_1x_1 + \ldots + c_nx_n. \tag{2}$$

Обозначим через M допустимую область задачи (1):

$$M = \left\{x \in \mathbb{R}^n | Ax \leqslant b\right\}. \tag{3}$$

Везде далее мы предполагаем, что M является непустым ограниченным множеством. Это означает, что M представляет собой выпуклый замкнутый многогранник в пространстве $\mathbb{R}^n$, и множество решений задачи (1) не является пустым.

Пусть $\tilde{a}_i \in \mathbb{R}^n$ — вектор, образованный элементами i-той строки матрицы A. Тогда матричное неравенство $Ax \leqslant b$ можно представить в виде системы неравенств

$$\langle\tilde{a}_i, x\rangle \leqslant b_i, i = 1, \ldots, m. \tag{4}$$

Везде далее мы будем предполагать, что для всех $i = 1, \ldots, m$

$$\tilde{a}_i \neq \mathbf{0}. \tag{5}$$

Обозначим через H_i гиперплоскость, задаваемую уравнением

$$\langle\tilde{a}_i, x\rangle = b_i \ (1 \leqslant i \leqslant m). \tag{6}$$

Таким образом,

$$H_i = \left\{x \in \mathbb{R}^n | \langle\tilde{a}_i, x\rangle = b_i\right\}. \tag{7}$$

Определение 1. Под полупространством H_i^+, *порождаемым гиперплоскостью* H_i, будем понимать полупространство, определяемое формулой

$$H_i^+ = \{x \in \mathbb{R}^n | \langle \tilde{a}_i, x \rangle \leqslant b_i\}. \tag{8}$$

Везде далее мы будем предполагать, что задача (1) является невырожденной, то есть

$$\forall i \neq j : H_i \neq H_j \ (i, j \in \{1, \ldots, m\}). \tag{9}$$

Определение 2. Полупространство H_i^+, порождаемое гиперплоскостью H_i, является *рецессивным* относительно вектора c, если

$$\forall x \in H_i, \forall \lambda \in \mathbb{R}_{>0} : x - \lambda c \in H_i^+ \wedge x - \lambda c \notin H_i. \tag{10}$$

Другими словами, луч, исходящий из гиперплоскости H_i в направлении противоположном вектору c, полностью лежит в H_i^+, но не в H_i.

Утверждение 1. Необходимым и достаточным условием рецессивности полупространства H_i^+ относительно вектора c является условие

$$\langle \tilde{a}_i, c \rangle > 0. \tag{11}$$

Доказательство. Докажем сначала необходимость. Пусть выполняется условие (10). Из (7) следует

$$x = \frac{b_i \tilde{a}_i}{\|\tilde{a}_i\|^2} \in H_i. \tag{12}$$

Из (5) следует

$$\lambda = \frac{1}{\|\tilde{a}_i\|^2} \in \mathbb{R}_{>0}. \tag{13}$$

Сопоставляя (10),(12) и (13) получаем

$$\frac{b_i \tilde{a}_i}{\|\tilde{a}_i\|^2} - \frac{1}{\|\tilde{a}_i\|^2} c \in H_i^+;$$
$$\frac{b_i \tilde{a}_i}{\|\tilde{a}_i\|^2} - \frac{1}{\|\tilde{a}_i\|^2} c \notin H_i.$$

С учетом (7), (8) отсюда следует, что

$$\left\langle \tilde{a}_i, \tfrac{b_i \tilde{a}_i}{\|\tilde{a}_i\|^2} - \tfrac{1}{\|\tilde{a}_i\|^2} c \right\rangle < b_i. \tag{14}$$

Выполнив несложные преобразования неравенства (14), получаем (11).

Доказательство достаточности проведем от противного. Предположим, что имеет место (11) и существуют $x \in H_i$ и $\lambda > 0$ такие, что

$$x - \lambda c \notin H_i^+ \vee x - \lambda c \in H_i.$$

В соответствии с (7), (8) это означает

$$\langle \tilde{a}_i, x - \lambda c \rangle \geqslant b_i,$$

что равносильно

$$\langle \tilde{a}_i, x \rangle - \lambda \langle \tilde{a}_i, c \rangle \geqslant b_i.$$

Так как $\lambda > 0$, то в силу (11) отсюда получаем

$$\langle \tilde{a}_i, x \rangle > b_i,$$

что противоречит предположению $x \in H_i$. □

Определение 3. Зафиксируем точку $z \in \mathbb{R}^n$ такую, что полупространство

$$H_c^+ = \{x \in \mathbb{R}^n | \langle c, x - z \rangle \leqslant 0\} \tag{15}$$

включает в себя многогранник M:

$$M \subset H_c^+.$$

Полупространство H_c^+ в этом случае будем называть *целевым полупространством*, а гиперплоскость H_c, определяемую уравнением

$$H_c = \{x \in \mathbb{R}^n | \langle c, x - z \rangle = 0\}, \tag{16}$$

будем называть *целевой гиперплоскостью*.

Обозначим через $\pi_c(x)$ ортогональную проекцию точки x на целевую гиперплоскость H_c:

$$\pi_c(x) = x - \frac{\langle c, x - z \rangle}{\|c\|^2} c. \tag{17}$$

Здесь $\| \cdot \|$ обозначает евклидову норму. Определим *расстояние* $\rho_c(x)$ от точки $x \in H_c^+$ до целевой гиперплоскости H_c следующим образом:

$$\rho_c(x) = \|\pi_c(x) - x\|. \tag{18}$$

Сопоставляя (15), (17) и (18) находим, что в этом случае расстояние $\rho_c(x)$ может быть вычислено следующим образом:

$$\rho_c(x) = \frac{\langle c, z - x \rangle}{\|c\|}. \tag{19}$$

Справедливо следующее утверждение.

Утверждение 2. Для любых $x, y \in H_c^+$

$$\rho_c(x) \leqslant \rho_c(y) \Leftrightarrow \langle c, x \rangle \geqslant \langle c, y \rangle.$$

Доказательство. Используя (19), получаем

$$\begin{aligned}
&\rho_c(x) \leqslant \rho_c(y) \Leftrightarrow \frac{\langle c, z-x\rangle}{\|c\|} \leqslant \frac{\langle c, z-y\rangle}{\|c\|} \\
&\Leftrightarrow \langle c, z-x\rangle \leqslant \langle c, z-y\rangle \\
&\Leftrightarrow \langle c, z\rangle + \langle c, -x\rangle \leqslant \langle c, z\rangle + \langle c, -y\rangle \\
&\Leftrightarrow \langle c, -x\rangle \leqslant \langle c, -y\rangle \\
&\Leftrightarrow \langle c, x\rangle \geqslant \langle c, y\rangle .
\end{aligned}$$

□

В соответствии с утверждением 2 задача (1) эквивалентна следующей задаче:

$$\bar{x} = \arg\min \{\rho_c(x) | x \in M\} . \tag{20}$$

Определение 4. Пусть полупространство H_i^+ является рецессивным относительно вектора c. *Целевой проекцией* $\gamma_i(x)$ точки $x \in \mathbb{R}^n$ на рецессивное полупространство H_i^+ называется точка, определяемая формулой

$$\gamma_i(x) = x - \sigma_i(x)c, \tag{21}$$

где

$$\sigma_i(x) = \min \left\{\sigma \in \mathbb{R}_{\geqslant 0} \;\middle|\; x - \sigma c \in H_i^+\right\} .$$

Примеры целевых проекций в $\mathbb{R}^2$ приведены на рис. 1.

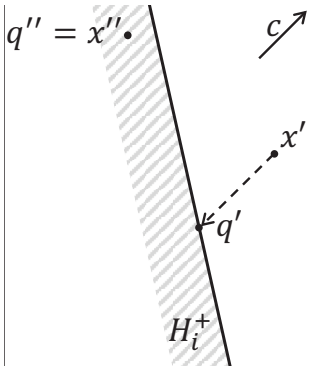


Рис. 1. Целевые проекции в пространстве $\mathbb{R}^2$: $\gamma_i(x') = q'$; $\gamma_i(x'') = q'' = x''$

Следующее утверждение предоставляет формулу для вычисления целевой проекции на полупространство, рецессивное относительно вектора c.

Утверждение 3. Пусть полупространство H_i^+, определяемое неравенством

$$\langle \tilde{a}_i, x\rangle \leqslant b_i, \tag{22}$$

является рецессивным относительно вектора c. Пусть

$$g \notin H_i^+. \tag{23}$$

Тогда

$$\gamma_i(g) = g - \frac{\langle \tilde{a}_i, g\rangle - b_i}{\langle \tilde{a}_i, c\rangle} c. \tag{24}$$

Доказательство. В соответствии с определением 4 имеем

$$\gamma_i(g) = g - \sigma_i(g)c,$$

где

$$\sigma_i(x) = \min\left\{\sigma \in \mathbb{R}_{\geqslant 0} \;\middle|\; x - \sigma c \in H_i^+\right\}.$$

Таким образом, нам необходимо показать, что

$$\frac{\langle \tilde{a}_i, g\rangle - b_i}{\langle \tilde{a}_i, c\rangle} = \min\left\{\sigma \in \mathbb{R}_{\geqslant 0} \;\middle|\; x - \sigma c \in H_i^+\right\}. \tag{25}$$

Рассмотрим прямую L, задаваемую параметрическим уравнением

$$L = \{g + \tau c | \tau \in \mathbb{R}\}.$$

Пусть точка q является пересечением прямой L с гиперплоскостью H_i:

$$q = L \cap H_i. \tag{26}$$

Тогда q должна удовлетворять уравнению

$$q = g + \tau' c \tag{27}$$

при некотором $\tau' \in \mathbb{R}$. Подставим правую часть уравнения (27) в уравнение (6) вместо x:

$$\left\langle \tilde{a}_i, g + \tau' c\right\rangle = b_i.$$

Отсюда

$$\begin{aligned} &\langle \tilde{a}_i, g\rangle + \tau' \langle \tilde{a}_i, c\rangle = b_i, \\ &\tau' = \frac{b_i - \langle \tilde{a}_i, g\rangle}{\langle \tilde{a}_i, c\rangle}. \end{aligned} \tag{28}$$

Подставив вместо τ' правую часть уравнения (28) в формулу (27) получаем

$$q = g + \frac{b_i - \langle \tilde{a}_i, g\rangle}{\langle \tilde{a}_i, c\rangle} c,$$

что эквивалентно

$$q = g - \frac{\langle \tilde{a}_i, g\rangle - b_i}{\langle \tilde{a}_i, c\rangle} c. \tag{29}$$

Так как $q \in H_i$ в соответствии с (26), формула (25) будет иметь место, если мы покажем, что

$$\forall \sigma \in \mathbb{R}_{>0} : \sigma < \frac{\langle \tilde{a}_i, g\rangle - b_i}{\langle \tilde{a}_i, c\rangle} \Rightarrow g - \sigma c \notin H_i^+. \tag{30}$$

Предположим противное, то есть существует $\sigma' > 0$ такое, что

$$\sigma' < \frac{\langle \tilde{a}_i, g\rangle - b_i}{\langle \tilde{a}_i, c\rangle} \tag{31}$$

и

$$g - \sigma' c \in H_i^+. \tag{32}$$

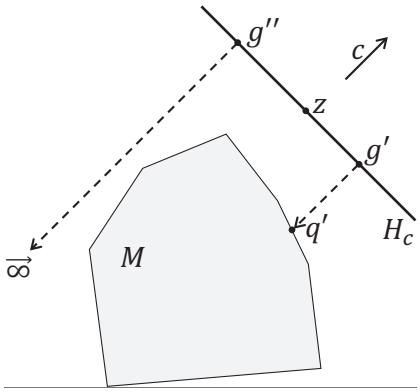


Рис. 2. Целевые проекции на многогранник M в пространстве $\mathbb{R}^2$: $\gamma_M(g') = q'$; $\gamma_M(g'') = \vec{\infty}$

Тогда из (22) и (32) следует

$$\left\langle \tilde{a}_i, g - \sigma' c \right\rangle \leqslant b_i,$$

что равносильно

$$\langle \tilde{a}_i, g \rangle - b_i \leqslant \sigma' \langle \tilde{a}_i, c \rangle . \tag{33}$$

В силу утверждения 1 имеем $\langle \tilde{a}_i, c \rangle > 0$. Поэтому (33) равносильно

$$\frac{\langle \tilde{a}_i, g \rangle - b_i}{\langle \tilde{a}_i, c \rangle} \leqslant \sigma'.$$

Получили противоречие с (31). □

Определение 5. Пусть $g \in H_c$. *Целевой проекцией* $\gamma_M(g)$ точки g на многогранник M называется точка, определяемая формулой

$$\gamma_M(g) = g - \sigma_M(g)c, \tag{34}$$

где

$$\sigma_M(g) = \min \{\sigma \in \mathbb{R}_{\geqslant 0} | g - \sigma c \in M\} .$$

Если

$$\neg\exists\, \sigma \in \mathbb{R}_{\geqslant 0} : g - \sigma c \in M,$$

полагаем $\gamma_M(g) = \vec{\infty}$ — точка, бесконечно удаленная от M.

Примеры целевых проекций на многогранник M в $\mathbb{R}^2$ приведены на рис. 2.

Определение 6. *Рецептивным полем* $\mathfrak{G}(z, \eta, \delta) \subset H_c$ плотности $\delta \in \mathbb{R}_{>0}$ с центром в точке $z \in H_c$ и рангом $\eta \in \mathbb{N}$ будем называть конечное упорядоченное множество точек, удовлетворяющих следующим условиям:

$$z \in \mathfrak{G}(z, \eta, \delta); \tag{35}$$

$$\forall g \in \mathfrak{G}(z, \eta, \delta) : \|g - z\| \leqslant \eta\delta\sqrt{n}; \tag{36}$$

$$\forall g', g'' \in \mathfrak{G}(z, \eta, \delta) : g' \neq g'' \Rightarrow \|g' - g''\| \geqslant \delta; \tag{37}$$

$$\forall g' \in \mathfrak{G}(z, \eta, \delta)\ \exists g'' \in \mathfrak{G}(z, \eta, \delta) : \|g' - g''\| = \delta; \tag{38}$$

$$\forall x \in \mathrm{Co}(\mathfrak{G}(z, \eta, \delta))\ \exists g \in \mathfrak{G}(z, \eta, \delta) : \|g - x\| \leqslant \tfrac{1}{2}\delta\sqrt{n}. \tag{39}$$

Здесь $\mathrm{Co}(X)$ обозначает выпуклую оболочку конечного множества точек $X = \left\{x^{(1)}, \ldots, x^{(K)}\right\} \subset \mathbb{R}^n$:

$$Co(X) = \left\{ \sum_{i=1}^{K} \lambda_i x^{(i)} \middle| \lambda_i \in \mathbb{R}_{\geqslant 0}, \sum_{i=1}^{K} \lambda_i = 1 \right\}.$$

Точки рецептивного поля будем называть *рецептивными точками.*

Формула (35) в определении 6 означает, что точка, находящаяся в центре рецептивного поля, принадлежит этому полю. Из формулы (36) следует, что любая точка g рецептивного поля отстоит от центральной точки z на расстояние не более $\eta\delta\sqrt{n}$. В соответствии с формулой (37) для любых двух различных точек $g' \neq g''$ рецептивного поля расстояние между ними не может быть меньше δ. Формула (38) говорит, что для любой точки g' рецептивного поля найдется точка g'', принадлежащая этому же полю, такая, что расстояние между g' и g'' будет равно δ. Формула (39) означает, что для любой точки x, принадлежащей выпуклой оболочке рецептивного поля, в этом поле найдется точка g, отстоящая от x на расстояние не более $\frac{1}{2}\delta\sqrt{n}$. Пример рецептивного поля в пространстве $\mathbb{R}^3$ приведен на рис. 3.

Опишем конструктивный метод построения рецептивного поля. Без ограничения общности мы можем предполагать, что $c_n \neq 0$. Построим в $\mathbb{R}^n$ следующий ортогональный базис, включающий в себя вектор c:

$$\begin{aligned}
&c^{(0)} = c = (c_1, c_2, c_3, c_4, \ldots, c_{n-1}, c_n);\\
&c^{(1)} = \begin{cases} \left(-\frac{1}{c_1}\sum_{i=2}^{n} c_i^2, c_2, c_3, c_4, \ldots, c_{n-1}, c_n\right), \text{ если } c_1 \neq 0; \\ (1, 0, \ldots, 0), \text{ если } c_1 = 0; \end{cases}\\
&c^{(2)} = \begin{cases} \left(0, -\frac{1}{c_2}\sum_{i=3}^{n} c_i^2, c_3, c_4, \ldots, c_{n-1}, c_n\right), \text{ если } c_2 \neq 0; \\ (0, 1, 0, \ldots, 0), \text{ если } c_2 = 0; \end{cases}\\
&c^{(3)} = \begin{cases} \left(0, 0, -\frac{1}{c_3}\sum_{i=4}^{n} c_i^2, c_4, \ldots, c_{n-1}, c_n\right), \text{ если } c_3 \neq 0; \\ (0, 0, 1, 0, \ldots, 0), \text{ если } c_3 = 0; \end{cases}\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&c^{(n-2)} = \begin{cases} \left(0, \ldots, 0, -\frac{1}{c_{n-2}}\sum_{i=n-1}^{n} c_i^2, c_{n-1}, c_n\right), \text{ если } c_{n-2} \neq 0; \\ (0, \ldots, 0, 1, 0, 0), \text{ если } c_{n-2} = 0; \end{cases}\\
&c^{(n-1)} = \begin{cases} \left(0, \ldots, 0, -\frac{c_n^2}{c_{n-1}}, c_n\right), \text{ если } c_{n-1} \neq 0; \\ (0, \ldots, 0, 0, 1, 0), \text{ если } c_{n-1} = 0. \end{cases}
\end{aligned}$$

Непосредственно проверяется, что

$$\forall i, j \in \{0, 1, \ldots, n-1\}, i \neq j : \left\langle c^{(i)}, c^{(j)} \right\rangle = 0.$$

В частности

$$\forall i = 1, \ldots, n-1 : \left\langle c, c^{(i)} \right\rangle = 0. \tag{40}$$

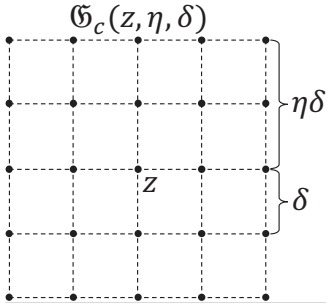


Рис. 3. Рецептивное поле в пространстве $\mathbb{R}^3$

Следующее утверждение показывает, что линейное подпространство размерности $(n-1)$, порождаемое ортогональными векторами $c_1, \ldots, c_{n-1}$, является гиперплоскостью, параллельной гиперплоскости H_c.

Утверждение 4. Определим линейное подпространство $S_c \subset \mathbb{R}^n$ размерности $n-1$:

$$S_c = \left\{ \sum_{i=1}^{n-1} \lambda_i c^{(i)} \middle| \lambda_i \in \mathbb{R} \right\}. \tag{41}$$

Тогда

$$\forall s \in S_c : s + z \in H_c. \tag{42}$$

Доказательство. Пусть $s \in S_c$, то есть

$$s = \lambda_1 c^{(1)} + \ldots + \lambda_{n-1} c^{(n-1)}.$$

Тогда

$$\langle c, (s+z) - z \rangle = \lambda_1 \left\langle c, c^{(1)} \right\rangle + \ldots + \lambda_{n-1} \left\langle c, c^{(n-1)} \right\rangle.$$

Отсюда в силу (40) получаем

$$\langle c, (s+z) - z \rangle = 0.$$

Сопоставляя это с (16), имеем $s + z \in H_c$. □

Определим векторы

$$e^{(i)} = \frac{c^{(i)}}{\|c^{(i)}\|} \ (i = 1, \ldots, n-1), \tag{43}$$

образующие ортонормированный базис подпространства S_c.

Процедура построения рецептивного поля представлена в виде алгоритма 1. Этот алгоритм строит рецептивное поле $\mathfrak{G}(z, \eta, \delta)$, состоящее из

$$K_{\mathfrak{G}} = (2\eta + 1)^{n-1} \tag{44}$$

точек, расположенных в узлах регулярной решетки, имеющей форму гиперквадрата (гиперкуба размерности $n-1$) с длиной ребра, равной $2\eta\delta$. Длина ребра ячейки гиперквадрата равна δ. В соответствии с шагом 13 алгоритма 1 и утверждением 4 этот гиперквадрат лежит в гиперплоскости H_c и имеет центр в точке z. Недостаток алгоритма 1 состоит в том, что количество вложенных циклов **for** зависит от размерности пространства. Эту проблему можно решить с помощью функции G, вычисляющей точку рецептивного множества по ее

Алгоритм 1 Построение рецептивного поля $\mathfrak{G}(z, \eta, \delta)$

Require: $z \in H_c$, $\eta \in \mathbb{N}$, $\delta \in \mathbb{R}_{>0}$

$\mathfrak{G} := \emptyset$
for $i_{n-1} = 0 \dots 2\eta$ **do**
  $s_{n-1} := i_{n-1}\delta - \eta\delta$
  for $i_{n-2} = 0 \dots 2\eta$ **do**
    $s_{n-2} := i_{n-2}\delta - \eta\delta$
    $\dots$
    for $i_1 = 0 \dots 2\eta$ **do**
      $s_1 := i_1\delta - \eta\delta$
      $s := \mathbf{0}$
      for $j = 1 \dots n-1$ **do**
        $s := s + s_j e^{(j)}$
      end for
      $\mathfrak{G} := \mathfrak{G} \cup \{s + z\}$
    end for
  end for
end for

Алгоритм 2 Функция G вычисляет точку рецептивного поля по ее номеру k

Require: $z \in H_c$, $\eta \in \mathbb{N}$, $\delta \in \mathbb{R}_{>0}$

function $\mathrm{G}(k, n, z, \eta, \delta)$
  for $j = (n-1) \dots 1$ **do**
    $l_j := \lfloor k/(2\eta+1)^{j-1} \rfloor$
    $k := k \mod (2\eta+1)^{j-1}$
  end for
  $g := z$
  for $j = 1 \dots (n-1)$ **do**
    $g := g + (l_j\delta - \eta\delta)e^{(j)}$
  end for
  $\mathrm{G} := g$
end function

порядковому номеру (нумерация начинается с нуля; порядок определяется алгоритмом 1). Реализация функции G представлена в виде алгоритма 2. Следующее утверждение 5 дает оценку временной сложности алгоритма 2.

Утверждение 5. Алгоритм 2 допускает реализацию с временной сложностью[1]

$$c_G = 4n^2 + 5n - 9, \tag{45}$$

[1]Под временной сложностью здесь понимается количество арифметических операций и операций сравнения, необходимых для выполнения алгоритма.

где n — размерность пространства.

Доказательство. Рассмотрим низкоуровневую реализацию алгоритма 2, представленную в виде алгоритма 3.

Алгоритм 3 Низкоуровневая реализация алгоритма 2

1: $p := 2\eta + 1$; $r := \eta\delta$; $h := p^{n-2}$; $g := z$
2: $j := n - 1$
3: **repeat**
4: $\quad l_j := \lfloor k/h \rfloor$
5: $\quad k := k \mod h$
6: $\quad h := h/p$
7: $\quad j := j - 1$
8: **until** $j = 0$
9: $j := 1$
10: **repeat**
11: $\quad w_j := l_j\delta - r$
12: $\quad i := 1$
13: $\quad$ **repeat**
14: $\quad\quad g_i := g_i + w_j e_i^{(j)}$
15: $\quad\quad i := i + 1$
16: $\quad$ **until** $i > n$
17: $\quad j := j + 1$
18: **until** $j = n$

Значения, вычисляемые на шагах 1–2 алгоритма 3, не зависят от номера рецептивной точки k и поэтому могут считаться константами. Цикл **repeat**/**until** на шагах 3–8 выполняется $(n-1)$ раз и требует $c_{3:8} = 5(n-1)$ операций. Вложенный цикл **repeat**/**until** на шагах 13–16 выполняется n раз и требует $c_{13:16} = 4n$ операций. Внешний цикл **repeat**/**until** на шагах 10–18 выполняется $(n-1)$ раз и требует $c_{10:18} = (4 + c_{13-16})(n-1) = 4(n^2-1)$ операций. В сумме получаем

$$c_G = c_{3:8} + c_{10:18} = 4n^2 + 5n - 9.$$

□

Следствие 1. Временная сложность алгоритма 2 может быть оценена как $O(n^2)$.

Определение 7. Пусть $z \in H_c$. Зафиксируем $\eta \in \mathbb{N}$, $\delta \in \mathbb{R}_{>0}$. *Образом* $\mathfrak{I}(z, \eta, \delta)$, порожденным рецептивным полем $\mathfrak{G}(z, \eta, \delta)$, будем называть упорядоченное множество вещественных чисел

$$\mathfrak{I}(z, \eta, \delta) = \{\rho_c(\gamma_M(g)) | g \in \mathfrak{G}(z, \eta, \delta)\}. \tag{46}$$

Порядок чисел в образе определяется порядком соответствующих точек рецептивного поля.

Функцию построения образа $\mathfrak{I}(z, \eta, \delta)$ в виде списка чисел представлена в виде алгоритма 4. Здесь [] обозначает пустой список, а $+\!\!+$ обозначает операцию конкатенации списков.

Алгоритм 4 Построение образа $\mathfrak{I}(z, \eta, \delta)$

Require: $z \in H_c$, $\eta \in \mathbb{N}$, $\delta \in \mathbb{R}_{>0}$

function $\mathfrak{I}(z, \eta, \delta)$
  $\mathfrak{I} := [\,]$
  for $k = 0 \ldots ((2\eta+1)^{n-1} - 1)$ **do**
    $g_k := \mathrm{G}(k, n, z, \eta, \delta)$
    $\mathfrak{I} := \mathfrak{I} +\!\!+ [\rho_c(\gamma_M(g_k))]$
  end for
end function

Покажем как полученный образ может быть использован для решения задачи ЛП. Пусть $\langle \tilde{a}_i, c\rangle > 0$. Это означает, что полупространство H_i^+ является рецессивным относительно вектора c (см. утверждение 1). Пусть имеется точка $u \in H_i \cap M$. Допустим, что нам удалось создать *искусственную нейронную сеть* DNN, получающую на входе образ $\mathfrak{I}(\pi_c(u), \eta, \delta)$ окрестности точки u, и выдающую на выходе точку u' такую, что

$$u' = \arg\min \{\rho_c(x) | x \in H_i \cap M\}.$$

Тогда мы можем построить алгоритм 5, решающий задачу линейного программирования (20) с помощью DNN.

Алгоритм 5 Линейное программирование с использованием DNN

Require: $u^{(1)} \in H_i \cap M$, $\langle \tilde{a}_i, c\rangle > 0$, $z \in H_c$; $\eta \in \mathbb{N}$, $\delta \in \mathbb{R}_{>0}$

$k := 1$
repeat
  $\mathcal{I} := \mathfrak{I}(u^{(k)}, \eta, \delta)$
  $u^{(k+1)} := \mathrm{DNN}(\mathcal{I})$
  $k := k + 1$
until $u^{(k)} \neq u^{(k-1)}$
$\bar{x} := u^{(k)}$
stop

2. Параллельный алгоритм построения образа задачи ЛП

При решении задач ЛП большой размерности и с большим количеством ограничений алгоритм 4 построения образа задачи ЛП может потребовать значительных временных затрат. В этом разделе приводится его параллельная версия, позволяющая существенно сократить время решения задачи ЛП с помощью алгоритма 5. Параллельная версия алгоритма 4 строится на основе модели параллельных вычислений BSF [24, 25], ориентированной на кластерные вычислительные системы. Модель BSF использует парадигму мастер–рабочие и требует представление алгоритма в форме операций над списками с использованием функций высшего порядка *Map* и *Reduce*, определенных в формализме Бёрда—Миртенса (Bird–Meertens formalism) [26]. Модель BSF также предоставляет метрику для аналитиче-

ской оценки масштабируемости параллельного алгоритма, удовлетворяющего указанным требованиям.

Представим алгоритм 4 в форме операций над списками с использованием функций высшего порядка *Map* и *Reduce*. В качестве списка, обрабатываемого функцией высшего порядка *Map*, возьмем список номеров неравенств системы (4):

$$\mathcal{L}_{map} = [1, \ldots, m]\,. \tag{47}$$

Обозначим $\mathbb{R}_\infty = \mathbb{R} \cup \{\infty\}$. Определим следующим образом параметризованную функцию $\mathrm{F}_k : \{1, \ldots, m\} \to \mathbb{R}_\infty$, являющуюся первым параметром функции высшего порядка *Map*:

$$\mathrm{F}_k(i) = \begin{cases} \rho_c\left(\gamma_i(g_k)\right), \text{ если } \langle \tilde{a}_i, c\rangle > 0 \text{ и } \gamma_i(g_k) \in M; \\ \infty, \text{ если } \langle \tilde{a}_i, c\rangle \leqslant 0 \text{ или } \gamma_i(g_k) \notin M, \end{cases} \tag{48}$$

где $g_k = G(k, n, z, \eta, \delta)$ вычисляется с помощью алгоритма 2, а $\gamma_i(g_k)$ вычисляется по формуле (24). С неформальной точки зрения функция F_k отображает номер полупространства H_i^+ в расстояние от целевой проекции до целевой гиперплоскости, если H_i^+ является рецессивным относительно c (см. утверждение 1) и целевая проекция принадлежит M. В противном случае F_k возвращает специальное значение ∞.

Функция высшего порядка *Map* преобразует список $\mathcal{L}_{map}$ в список $\mathcal{L}_{reduce}$ путем применения функции F_k к каждому элементу списка $\mathcal{L}_{map}$:

$$\mathcal{L}_{reduce} = \mathrm{Map}\left(\mathrm{F}_k, \mathcal{L}_{map}\right) = \left[\mathrm{F}_k(1), \ldots, \mathrm{F}_k(m)\right] = [\rho_1, \ldots, \rho_m]\,.$$

Определим бинарную ассоциативную операцию $\Downarrow : \mathbb{R}_\infty \to \mathbb{R}_\infty$ следующим образом:

$$\begin{aligned} \infty \Downarrow \infty &= \infty; \\ \forall \alpha \in \mathbb{R} : \alpha \Downarrow \infty &= \alpha; \\ \forall \alpha, \beta \in \mathbb{R} : \alpha \Downarrow \beta &= \min(\alpha, \beta). \end{aligned}$$

С неформальной точки зрения операция $\Downarrow$ вычисляет минимальное из двух чисел.

Функция высшего порядка *Reduce* преобразует список $\mathcal{L}_{reduce}$ в атомарное значение $\rho \in \mathbb{R}_\infty$ путем последовательного применения операции $\Downarrow$ ко всему списку:

$$\mathrm{Reduce}(\Downarrow, \mathcal{L}_{reduce}) = \rho_1 \Downarrow \rho_2 \Downarrow \ldots \Downarrow \rho_m = \rho.$$

Построение образа $\mathfrak{I}$ задачи ЛП с использованием функций высшего порядка *Map* и *Reduce* представлено в виде алгоритма 6. Параллельная версия алгоритма 6 строится на основе алгоритмического шаблона 2 из [24]. Результат представлен в виде алгоритма 7. Прокомментируем параллельный алгоритм 7. Для простоты мы будем предполагать, что количество ограничений m кратно количеству рабочих L и нумерация неравенств начинается с нуля. Параллельный алгоритм включает в себя $L + 1$ процесс: один процесс-мастер (кратко — мастер) и L процессов-рабочих (кратко — рабочие). Мастер управляет вычислениями. Первоначально в качестве образа $\mathfrak{I}$ берется пустой список (шаг 2 мастера). Текущий номер k полагается равным нулю (шаг 3 мастера). На шагах 4–15 мастер организует цикл **repeat**/**until**, в котором строится образ $\mathfrak{I}$ задачи ЛП. На шаге 5 мастер посылает номер очередной рецептивной точки k всем рабочим. На шаге 8 мастер ожидает получение

Алгоритм 6 Построение образа $\mathfrak{I}$ с использованием *Map* и *Reduce*

Require: $z \in H_c$, $\eta \in \mathbb{N}$, $\delta \in \mathbb{R}_{>0}$

1: **input** $n, m, A, b, c, z, \eta, \delta$
2: $\mathfrak{I} := [\,]$
3: $\mathcal{L}_{map} := [1, \ldots, m]$
4: **for** $k = 0 \ldots ((2\eta+1)^{n-1} - 1)$ **do**
5: $\quad \mathcal{L}_{reduce} := \mathrm{Map}(\mathrm{F}_k, \mathcal{L}_{map})$
6: $\quad \rho := \mathrm{Reduce}(\Downarrow, \mathcal{L}_{reduce})$
7: $\quad \mathfrak{I} := \mathfrak{I} +\!\!+ [\rho]$
8: **end for**
9: **output** $\mathfrak{I}$
10: **stop**

Алгоритм 7 Параллельный алгоритм построение образа $\mathfrak{I}$ задачи ЛП

Мастер	**Рабочий (l=0,. . . ,L-1)**
1: **input** n	1: **input** $n, m, A, b, c, z, \eta, \delta$
2: $\mathfrak{I} := [\,]$	2: $L := \mathrm{NumberOfWorkers}$
3: $k := 0$	3: $\mathcal{L}_{map(l)} := [lm/L, \ldots, ((l+1)m/L) - 1]$
4: **repeat**	4: **repeat**
5: $\quad$ **SendToWorkers** k	5: $\quad$ **RecvFromMaster** k
6:	6: $\quad \mathcal{L}_{reduce(l)} := \mathrm{Map}\left(\mathrm{F}_k, \mathcal{L}_{map(l)}\right)$
7:	7: $\quad \rho_l := \mathrm{Reduce}\left(\Downarrow, \mathcal{L}_{reduce(l)}\right)$
8: $\quad$ **RecvFromWorkers** $[\rho_0, \ldots, \rho_{L-1}]$	8: $\quad$ **SendToMaster** ρ_l
9: $\quad \rho := \mathrm{Reduce}\,(\Downarrow, [\rho_0, \ldots, \rho_{L-1}])$	9:
10: $\quad \mathfrak{I} := \mathfrak{I} +\!\!+ [\rho]$	10:
11: $\quad k := k + 1$	11:
12: $\quad exit := \left(k \geqslant (2\eta+1)^{n-1}\right)$	12:
13: $\quad$ **SendToWorkers** $exit$	13: $\quad$ **RecvFromMaster** $exit$
14: **until** $exit$	14: **until** $exit$
15: **output** $\mathfrak{I}$	15:
16: **stop**	16: **stop**

частичных результатов от всех рабочих. Эти результаты редуцируются в одно значение, которое добавляется в образ $\mathfrak{I}$ (шаги 9–10 мастера). Шаг 11 увеличивает счетчик итераций k на единицу. На шаге 12 мастер вычисляет критерий завершения в виде логического выражения $\left(k \geqslant (2\eta+1)^{n-1}\right)$, значение которого присваивается булевой переменной *exit*. В соответствии с формулой (44) значение true будет означать, что точки рецептного поля закончились. На шаге 13 мастер посылает значение булевой переменной *exit* всем рабочим. Если обработаны не все точки рецептивного поля, то на шаге 14 происходит переход к выполнению следующей итерации цикла **repeat**/**until**. В противном случае мастер выводит построенный образ $\mathfrak{I}$ (шаг 15) и завершает свою работу (шаг 16).

Каждый l-тый рабочий выполняет общую последовательность действий, но над своей частью $\mathcal{L}_{map(l)}$ списка $\mathcal{L}_{map}$, которая определяется на шаге 3. На шаге 4 рабочий входит в цикл **repeat/until**. На шаге 5 он получает номер очередной точки k, принадлежащей рецептивному полю. На шаге 6 рабочий обрабатывает свой подсписок $\mathcal{L}_{map(l)}$, используя функцию высшего порядка *Map*, которая для каждого элемента подсписка выполняет параметризованную функцию F_k, определенную с помощью формулы (48). В результате получается подсписок $\mathcal{L}_{reduce(l)}$, содержащий расстояния $\mathrm{F}_k(i)$ от целевой гиперплоскости H_c до целевых проекций рецептивной точки g_k на гиперплоскости H_i для всех i из подсписка $\mathcal{L}_{map(l)}$. На шаге 7 рабочий с помощью функции высшего порядка *Reduce* редуцирует подсписок $\mathcal{L}_{reduce(l)}$ в атомарное значение ρ_l, используя ассоциативную бинарную операцию $\textcircled{\downarrow}$, вычисляющую минимальное расстояние. Полученный частный результат пересылается мастеру (шаг 8 рабочего). На шаге 13 рабочий ожидает получение от мастера значения булевой переменной *exit*. Если получено значение false, рабочий продолжает выполнение цикла **repeat/until** (шаг 14 рабочего). В противном случае процесс рабочего завершается на шаге 16.

Дадим *аналитическую оценку границы масштабируемости* параллельного алгоритма 7 с использованием стоимостной метрики модели параллельных вычислений BSF [24]. Под границей масштабируемости здесь понимается число рабочих, на котором достигается максимум ускорения. Стоимостная метрика модели BSF включает в себя следующие стоимостные параметры для цикла **repeat/until** (шаги 4–14) параллельного алгоритма 7:

- m : длина списка $\mathcal{L}_{map}$;
- D : латентность (время пересылки одно байта от мастера к рабочему);
- t_c : время, затрачиваемое мастером на пересылку одному рабочему координат рецептивной точки и получения от него расстояния от этой точки до целевой проекции (включая латентность);
- t_{Map} : время, затрачиваемое одним рабочим на выполнение функции высшего порядка *Map* для всего списка $\mathcal{L}_{map}$;
- t_a : время, затрачиваемое на выполнение одной бинарной операции $\textcircled{\downarrow}$.

В соответствии с формулой (14) из [24] граница масштабируемости параллельного алгоритма 7 может быть оценена следующим образом:

$$L_{max} = \frac{1}{2}\sqrt{\left(\frac{t_c}{t_a \ln 2}\right)^2 + \frac{t_{Map}}{t_a} + 4m} - \frac{t_c}{t_a \ln 2}. \tag{49}$$

Вычислим оценку для временных параметров формулы (49). Для этого ведем следующие обозначения в рамках одной итерации цикла **repeat/until** (шаги 4–14) параллельного алгоритма 7:

- c_c : количество числе, пересылаемых от мастера к рабочему и обратно в рамках одной итерации;
- c_{Map} : количество арифметических операций и операций сравнения, выполняемых на шаге 5 последовательного алгоритма 6;
- c_a : количество операций, необходимых для выполнения бинарной операции $\textcircled{\downarrow}$.

В начале каждой итерации мастер посылает каждому рабочему номер рецептивной точки. В ответ рабочий посылает расстояние от этой точки до целевой проекции. Следовательно

$$c_c = 2. \tag{50}$$

В контексте алгоритма 6

$$c_{Map} = (c_G + c_{F_k})\, m, \tag{51}$$

где c_G — количество операций, необходимых для вычисления координат точки g_k, c_{F_k} — количество операций, необходимых для вычисления функции $\mathrm{F}_k(i)$ по формуле (48) в предположении, что координаты точки g_k уже вычислены. Оценка значения c_G дана в предложении 5. Оценим значение c_{F_k}. В соответствии с (24) вычисление целевой проекции $\gamma_i(g_k)$ требует $(6n-2)$ арифметических операций. Из (19) следует, что вычисление $\rho_c(x)$ составляет $(5n-1)$ арифметических операций. Проверка условия $x \in M$ в соответствии с (4) потребует $m(2n-1)$ арифметических операций и m операций сравнения. Таким образом

$$c_{F_k} = 2mn + 11n - 3. \tag{52}$$

Подставив в (51) правые части формул (45) и (52), получаем

$$c_{Map} = 4n^2m + 2m^2n + 16nm - 12m. \tag{53}$$

Для выполнения бинарной операции $\textcircled{\downarrow}$ необходимо выполнить одну операцию сравнения:

$$c_a = 1. \tag{54}$$

Пусть τ_{op} — среднее время выполнения арифметических операций и операций сравнения, τ_{tr} — среднее время для пересылки одного вещественного числа без учета латентности. Тогда, используя формулы (50), (53) и (54), получаем

$$t_c = c_c\tau_{tr} + 2D = 2(\tau_{tr} + D); \tag{55}$$

$$t_{Map} = c_{Map}\tau_{op} = (4n^2m + 2m^2n + 16nm - 12m)\tau_{op}; \tag{56}$$

$$t_a = c_a\tau_{op} = \tau_{op}. \tag{57}$$

Подставив в (49) правые части формул (55)–(57), получим следующую оценку границы масштабируемости параллельного алгоритма 7:

$$L_{max} = \frac{1}{2}\sqrt{\left(\frac{2(\tau_{tr}+D)}{\tau_{op}\ln 2}\right)^2 + 4n^2m + 2m^2n + 16nm - 12m} - \frac{2(\tau_{tr}+D)}{\tau_{op}\ln 2}.$$

Для больших значений m и n это эквивалентно

$$L_{max} \approx O(\sqrt{2n^2m + m^2n + 8nm - 6m}). \tag{58}$$

Если предположить, что $m = O(n)$, то из (58) следует

$$L_{max} \approx O(n\sqrt{n}), \tag{59}$$

Таблица 1. Характеристики кластера «Торнадо ЮУрГУ»

Параметр	Значение
Количество процессорных узлов	480
Процессоры	Intel Xeon X5680 (6 ядер, 3.33 GHz)
Количество процессоров в узле	2
Оперативная память узла	24 GB DDR3
Соединительная сеть	InfiniBand QDR (40 Gbit/s)
Операционная система	Linux CentOS

где n — размерность пространства. Оценка (59) позволяет сделать вывод, что параллельный алгоритм 7 демонстрирует превосходную масштабируемость[2]. В следующем разделе мы проверим аналитическую оценку (59) путем проведения масштабных вычислительных экспериментов на реальной кластерной вычислительной системе.

3. Вычислительные эксперименты

Нами была выполнена параллельная реализация алгоритма 7 в виде программы ViLiPP (Visualization of Linear Programming Problem) на языке C++ с использованием программного BSF-каркаса [27, 28]. BSF-каркас базируется на модели параллельных вычислений BSF и инкапсулирует все аспекты, связанные с распараллеливанием программы с использованием библиотеки MPI [29] и программного интерфейса OpenMP [30]. Исходные коды программы ViLiPP свободно доступны в сети Интернет по адресу `https://github.com/nikolay-olkhovskiy/LP-visualization-MPI`. С использованием параллельной программы ViLiPP мы провели эксперименты по исследованию масштабируемости алгоритма 7 на кластерной вычислительной системе «Торнадо ЮУрГУ» [31], характеристики которой приведены в таблице 1.

С помощью генератора задач «FRaGenLP» [32, 33] для проведения вычислительных экспериментов были сгенерированы три случайные задачи ЛП, параметры которых приведены в таблице 2. Количество ненулевых значений матрицы A задачи (1) всех случаях составило 100%. Для всех задач ранг рецептивного поля η полагался равным 2. В соответствии с формулой (44) мощность рецептивного поля демонстрировала экспоненциальный рост с увеличением размерности пространства.

Результаты вычислительных экспериментов приведены в таблице 3 и на рис. 4. Во всех запусках каждому рабочему выделялся отдельный процессорный узел. Еще один дополнительный процессорный узел выделялся для работы мастера. Вычислительные эксперименты показали, что с ростом размерности задачи наблюдается рост границы масштабируемости программы ViLiPP: для LP5 максимум кривой ускорения достигается в районе 190 узлов, для LP6 максимум располагается в районе 260 узлов, а для LP7 он приблизительно равен 326 узлам. При этом наблюдается экспоненциальный рост времени решения задачи: образ задачи LP5 на 11 процессорных узлах строится за 10 сек., а построение образа задачи LP7 на таком же количестве узлов требует уже 5 мин. Дополнительный вычислительный

[2]Пусть $L_{max} = O(n^{\alpha})$. Алгоритм демонстрирует *превосходную масштабируемость*, если $\alpha > 1$; алгоритм демонстрирует *хорошую масштабируемость*, если $\alpha = 1$; алгоритм демонстрирует *ограниченную масштабируемость*, если $0 < \alpha < 1$; алгоритм *не масштабируется*, если $\alpha = 0$.

Таблица 2. Параметры тестовых задач ЛП

Идентификатор задачи	Число переменных	Число ограничений	Процент ненулевых значений в A	Мощность рецептивного поля
$LP7$	7	4016	100%	15 625
$LP6$	6	4014	100%	3 125
$LP5$	5	4012	100%	625

Таблица 3. Время построения образа задач ЛП (сек.)

Число процессорных узлов	LP5	LP6	LP7
11	9.81	54.45	303.78
56	1.93	10.02	59.43
101	1.55	6.29	33.82
146	1.39	4.84	24.73
191	1.35	4.20	21.10
236	1.38	3.98	19.20
281	1.45	3.98	18.47
326	1.55	4.14	18.30

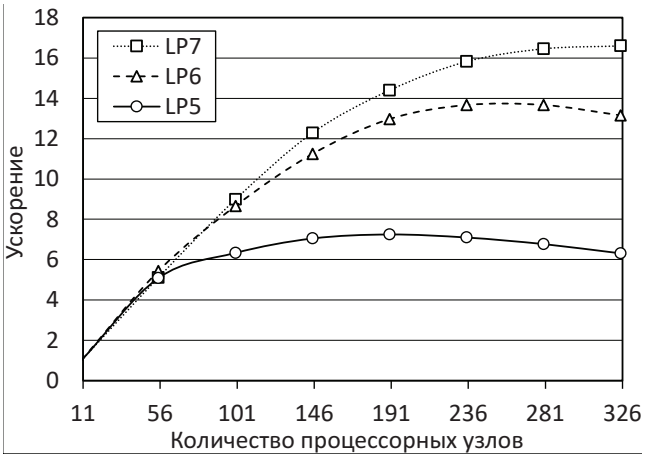


Рис. 4. Графики ускорения параллельной программы ViLiPP для задач ЛП различного размера

эксперимент показал, что построение образа задачи при $n = 9$ на 11 процессорных узлах занимает 1.5 часа.

Проведенные эксперименты позволяют сделать вывод, что при современном уровне развития вычислительной техники применение искусственных нейронных сетей для решения задач ЛП на основе предложенного метода визуализации может быть эффективным для задач размерности, не превышающей 100, с количеством ограничений до 100 000.

Заключение

Основным результатом, полученным в данной работе, является математическая модель визуального представления многомерной задачи линейного программирования по поиску в допустимой области точки максимума линейной целевой функции. Основным элементом модели является рецептивное поле, представляющее собой регулярное множество точек, располагающихся в узлах решетки, построенной внутри гиперкуба. Все точки рецептивного поля лежат в целевой гиперплоскости, ортогональной вектору $c = (c_1, \ldots, c_n)$, составленному из коэффициентов линейной целевой функции. Целевая гиперплоскость располагается так, что для любой точки x из допустимой области и любой точки z целевой гиперплоскости выполняется неравенство $\langle c, x\rangle < \langle c, z\rangle$. Можно сказать, что рецептивное поле является многомерным абстрактным аналогом светочувствительной матрицы цифрового фотоаппарата. Из каждой точки рецептивного поля в направлении допустимой области строится луч, параллельный вектору c. Точка, в которой луч касается допустимой области, называется целевой проекцией. Образ задачи линейного программирования представляет собой матрицу положительных вещественных чисел размерности $(n-1)$, в которой каждый элемент является расстоянием от точки рецептивного поля до соответствующей точки целевой проекции.

В статье описан алгоритм вычисления координат точки рецептивного поля по ее порядковому номеру. Показано, что временная сложность этого алгоритма может быть оценена как $O(n^2)$, где n — размерность пространства. Приведено общее описание алгоритма решения задачи линейного программирования с помощью искусственной нейронной сети на основе анализа построенных образов. Предложен параллельный алгоритм построения образа задачи линейного программирования, ориентированный на кластерные вычислительные системы. Этот алгоритм основывается на модели параллельных вычислений BSF, предполагающей использование парадигмы мастер–рабочие и представление алгоритма в виде операций над списками с использованием функций высшего порядка *Map* и *Reduce*. Показано, что для границы масштабируемости параллельного алгоритма справедлива оценка $O(n\sqrt{n})$. Это означает, что алгоритм демонстрирует хорошую масштабируемость.

Выполнена реализация параллельного алгоритма построения образа задачи линейного программирования на языке C++ с использованием параллельного программного BSF-каркаса, инкапсулирующего все аспекты, связанные с распараллеливанием на основе библиотеки MPI и программного интерфейса OpenMP. С использованием этой программной реализации на вычислительном кластере «Торнадо ЮУрГУ» проведены масштабные вычислительные эксперименты по построению образов для случайных многомерных задач линейного программирования с большим числом ограничений. Проведенные эксперименты подтверждают корректность и эффективность предложенных подходов. Вместе с тем следует отметить, что время построение образа растет экспоненциально с увеличением размерности пространства. Поэтому предложенный метод применим для задач с числом переменных, не превышающим 100. При этом количество ограничений теоретически может быть неограниченным.

В качестве направлений дальнейших исследований можно выделить следующие.

1. Разработать метод решения задачи линейного программирования на основании анализа ее образов и доказать его сходимость.
2. Разработать и реализовать метод построения обучающих множеств для создания нейронной сети, решающей задачи линейного программирования путем анализа их образов.

3. Разработать и обучить искусственную нейронную сеть для решения многомерные задачи линейного программирования.
4. Разработать и реализовать на вычислительном кластере параллельную программу, строящую многомерные образы задачи линейного программирования и получающую ее решение с помощью искусственной нейронной сети.

Исследование выполнено при финансовой поддержке РФФИ (грант 20-07-00092 а) и Министерства науки и высшего образования РФ (государственное задание FENU-2020-0022).

Литература

1. Jagadish H.V., Gehrke J., Labrinidis A., *et al.* Big data and its technical challenges // Communications of the ACM. 2014. Vol. 57, no. 7. P. 86–94. DOI: 10.1145/2611567.
2. Hartung T. Making Big Sense From Big Data // Frontiers in Big Data. 2018. Vol. 1. P. 5. DOI: 10.3389/fdata.2018.00005.
3. Соколинская И., Соколинский Л. О решении задачи линейного программирования в эпоху больших данных // Параллельные вычислительные технологии (ПаВТ'2017). Короткие статьи и описания плакатов. Челябинск: Издательский центр ЮУрГУ, 2017. С. 471—484. URL: http://omega.sp.susu.ru/pavt2017/short/014.pdf.
4. Chung W. Applying large-scale linear programming in business analytics // 2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM). IEEE, 2015. P. 1860–1864. DOI: 10.1109/IEEM.2015.7385970.
5. Gondzio J., Gruca J.A., Hall J.A.J., *et al.* Solving large-scale optimization problems related to Bell's Theorem // Journal of Computational and Applied Mathematics. 2014. Vol. 263. P. 392–404. DOI: 10.1016/j.cam.2013.12.003.
6. Sodhi M.S. LP modeling for asset-liability management: A survey of choices and simplifications // Operations Research. 2005. Vol. 53, no. 2. P. 181–196. DOI: 10.1287/opre.1040.0185.
7. 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.
8. Дышаев М.М., Соколинская И.М. Представление торговых сигналов на основе адаптивной скользящей средней Кауфмана в виде системы линейных неравенств // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. 2014. Т. 2, № 4. С. 103—108. DOI: 10.14529/cmse130408.
9. Bixby R.E. Solving Real-World Linear Programs: A Decade and More of Progress // Operations Research. 2002. Vol. 50, no. 1. P. 3–15. DOI: 10.1287/opre.50.1.3.17780.
10. Dongarra J., Gottlieb S., Kramer W.T.C. Race to Exascale // Computing in Science and Engineering. 2019. Vol. 21, no. 1. P. 4–5. DOI: 10.1109/MCSE.2018.2882574.
11. Dantzig G.B. Linear programming and extensions. Princeton, N.J.: Princeton university press, 1998. 656 p.
12. Zadeh N. A bad network problem for the simplex method and other minimum cost flow algorithms // Mathematical Programming. 1973. Vol. 5, no. 1. P. 255–266. DOI: 10.1007/BF01580132.

13. Tolla P. A Survey of Some Linear Programming Methods // Concepts of Combinatorial Optimization / ed. by V.T. Paschos. 2nd ed. Hoboken, NJ, USA: John Wiley, Sons, 2014. Chap. 7. P. 157–188. DOI: 10.1002/9781119005216.ch7.

14. Mamalis B., Pantziou G. Advances in the Parallelization of the Simplex Method // Algorithms, Probability, Networks, and Games. Lecture Notes in Computer Science, vol. 9295 / ed. by C. Zaroliagis, G. Pantziou, S. Kontogiannis. Cham: Springer, 2015. P. 281–307. DOI: 10.1007/978-3-319-24024-4_17.

15. Karmarkar N. A new polynomial-time algorithm for linear programming // Combinatorica. 1984. Vol. 4, no. 4. P. 373–395. DOI: 10.1007/BF02579150.

16. Yuan Y. Implementation tricks of interior-point methods for large-scale linear programs // Proc. SPIE, vol. 8285. International Conference on Graphic and Image Processing (ICGIP 2011). International Society for Optics, Photonics, 2011. DOI: 10.1117/12.913019.

17. Ершова А.В., Соколинская И.М. О сходимости масштабируемого алгоритма построения псевдопроекции на выпуклое замкнутое множество // Вестник Южно-Уральского государственного университета. Серия: Математическое моделирование и программирование. 2011. № 37(254). С. 12—21. URL: https://mmp.susu.ru/article/ru/127.

18. Hafsteinsson H., Levkovitz R., Mitra G. Solving large scale linear programming problems using an interior point method on a massively parallel SIMD computer // Parallel Algorithms and Applications. 1994. Vol. 4, no. 3-4. P. 301–316. DOI: 10.1080/10637199408915470.

19. Karypis G., Gupta A., Kumar V. A parallel formulation of interior point algorithms // Proceedings of the 1994 ACM/IEEE conference on Supercomputing (Supercomputing'94). Los Alamitos, CA, USA: IEEE Computer Society Press, 1994. P. 204–213. DOI: 10.1109/SUPERC.1994.344280.

20. 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.

21. Schmidhuber J. Deep learning in neural networks: An overview // Neural Networks. 2015. Vol. 61. P. 85–117. DOI: 10.1016/j.neunet.2014.09.003.

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

23. 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.

24. 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.

25. Ежова Н.А., Соколинский Л.Б. Модель параллельных вычислений BSF-MR // Системы управления и информационные технологии. 2019. № 3(77). С. 15—21.

26. Bird R.S. Lectures on Constructive Functional Programming // Constructive Methods in Computing Science. NATO ASI Series F: Computer and Systems Sciences, vol. 55 / ed. by M. Broy. Berlin, Heidelberg: Springer, 1988. P. 151–216.

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

28. Sokolinsky L.B. BSF-skeleton. 2019. URL: https://github.com/leonid-sokolinsky/BSF-skeleton (дата обращения: 24.01.2022).

29. Gropp W. MPI 3 and Beyond: Why MPI Is Successful and What Challenges It Faces // Recent Advances in the Message Passing Interface. EuroMPI 2012. Lecture Notes in Computer Science, vol. 7490 / ed. by J. Traff, S. Benkner, J. Dongarra. Berlin, Heidelberg: Springer, 2012. P. 1–9. DOI: 10.1007/978-3-642-33518-1_1.

30. Kale V. Shared-memory Parallel Programming with OpenMP // Parallel Computing Architectures and APIs. Boca Raton: Chapman, Hall/CRC, 2019. Chap. 14. P. 213–222. DOI: 10.1201/9781351029223-18/SHARED-MEMORY-PARALLEL-PROGRAMMING-OPENMP-VIVEK-KALE.

31. Kostenetskiy P., Semenikhina P. SUSU Supercomputer Resources for Industry and fundamental Science // Proceedings - 2018 Global Smart Industry Conference, GloSIC 2018. IEEE, 2018. Article 8570068. P. 1–7. DOI: 10.1109/GloSIC.2018.8570068.

32. 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, № 1. С. 38—52. DOI: 10.14529/cmse210103.

33. Соколинский Л.Б. Свидетельство о государственной регистрации программы для ЭВМ № 2021619526 Российская Федерация. Генератор случайных задач линейного программирования FRaGenLP : № 2021618165 : заявл. 28.05.2021 : опубл. 10.06.2021.

Ольховский Николай Александрович, аспирант, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Соколинский Леонид Борисович, д.ф.-м.н., профессор, проректор по информатизации, заведующий кафедрой системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация

DOI: 10.14529/cmse220103

VISUAL REPRESENTATION OF MULTIDIMENSIONAL LINEAR PROGRAMMING PROBLEMS

© 2022 N.A. Olkhovsky, L.B. Sokolinsky

South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)

E-mail: olkhovskiina@susu.ru, leonid.sokolinsky@susu.ru

Received: 10.03.2022

The article proposes an n-dimensional mathematical model of the visual representation of a linear programming problem. This model makes it possible to use artificial neural networks to solve multidimensional linear optimization problems, the feasible region of which is a bounded non-empty set. To visualize the linear programming problem, an objective hyperplane is introduced, the orientation of which is determined by the gradient of the linear objective function: the gradient is the normal to the objective hyperplane. In case of searching a maximum, the objective hyperplane is positioned in such a way that the value of the objective function at all its points exceeds the value of the objective function at all points of the feasible region, which is a bounded convex polytope. For an arbitrary point of the objective hyperplane, the objective projection onto the polytope is determined: the closer the objective projection point is to the objective hyperplane, the greater the value of the objective function at this point. Based on the objective hyperplane, a finite regular set of points is constructed, called the retina. Using objective projections, an image of a polytope is constructed. This image includes the points of the retina and the distances to the corresponding points of the polytope surface. Based on the proposed model, parallel algorithms for visualizing a linear programming problem are constructed. An analytical estimation of its scalability is performed. Information about the software implementation and the results of large-scale computational experiments confirming the efficiency of the proposed approaches are presented.

Keywords: linear programming, n-dimensional visualization, mathematical model, parallel algorithm, BSF-skeleton.

FOR CITATION

Olkhovsky N.A., Sokolinsky L.B. Visual Representation of Multidimensional Linear Programming Problems. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2022. Vol. 11, no. 1. P. 31–56. (in Russian) DOI: 10.14529/cmse220103.

References

1. Jagadish H.V., Gehrke J., Labrinidis A., *et al.* Big data and its technical challenges. Communications of the ACM. 2014. Vol. 57, no. 7. P. 86–94. DOI: 10.1145/2611567.
2. Hartung T. Making Big Sense From Big Data. Frontiers in Big Data. 2018. Vol. 1. P. 5. DOI: 10.3389/fdata.2018.00005.
3. Sokolinskaya I.M., Sokolinsky L.B. On solving linear programming problems in the era of big data. Parallel Computing Technologies (PCT'2017). Short articles and posters. Chelyabinsk: SUSU Publishing Center, 2017. P. 471–484. (in Russian).
4. Chung W. Applying large-scale linear programming in business analytics. 2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM). IEEE, 2015. P. 1860–1864. DOI: 10.1109/IEEM.2015.7385970.

5. Gondzio J., Gruca J.A., Hall J.A.J., *et al.* Solving large-scale optimization problems related to Bell's Theorem. Journal of Computational and Applied Mathematics. 2014. Vol. 263. P. 392–404. DOI: 10.1016/j.cam.2013.12.003.
6. Sodhi M.S. LP modeling for asset-liability management: A survey of choices and simplifications. Operations Research. 2005. Vol. 53, no. 2. P. 181–196. DOI: 10.1287/opre.1040.0185.
7. 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.
8. Dyshaev M.M., Sokolinskaya I.M. Representation of trading signals based on the Kaufman's Adaptive Moving Average in the form of a system of linear inequalities. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2014. Vol. 2, no. 4. P. 103–108. (in Russian) DOI: 10.14529/cmse130408.
9. Bixby R.E. Solving Real-World Linear Programs: A Decade and More of Progress. Operations Research. 2002. Vol. 50, no. 1. P. 3–15. DOI: 10.1287/opre.50.1.3.17780.
10. Dongarra J., Gottlieb S., Kramer W.T.C. Race to Exascale. Computing in Science and Engineering. 2019. Vol. 21, no. 1. P. 4–5. DOI: 10.1109/MCSE.2018.2882574.
11. Dantzig G.B. Linear programming and extensions. Princeton, N.J.: Princeton university press, 1998. 656 p.
12. Zadeh N. A bad network problem for the simplex method and other minimum cost flow algorithms. Mathematical Programming. 1973. Vol. 5, no. 1. P. 255–266. DOI: 10.1007/BF01580132.
13. Tolla P. A Survey of Some Linear Programming Methods. Concepts of Combinatorial Optimization / ed. by V.T. Paschos. 2nd ed. Hoboken, NJ, USA: John Wiley, Sons, 2014. Chap. 7. P. 157–188. DOI: 10.1002/9781119005216.ch7.
14. Mamalis B., Pantziou G. Advances in the Parallelization of the Simplex Method. Algorithms, Probability, Networks, and Games. Lecture Notes in Computer Science, vol. 9295 / ed. by C. Zaroliagis, G. Pantziou, S. Kontogiannis. Cham: Springer, 2015. P. 281–307. DOI: 10.1007/978-3-319-24024-4_17.
15. Karmarkar N. A new polynomial-time algorithm for linear programming. Combinatorica. 1984. Vol. 4, no. 4. P. 373–395. DOI: 10.1007/BF02579150.
16. Yuan Y. Implementation tricks of interior-point methods for large-scale linear programs. Proc. SPIE, vol. 8285. International Conference on Graphic and Image Processing (ICGIP 2011). International Society for Optics, Photonics, 2011. DOI: 10.1117/12.913019.
17. Ershova A.V., Sokolinskaya I.M. On the convergence of a scalable algorithm for constructing a pseudoprojection on a convex closed set. Bulletin of the South Ural State University, Series: Mathematical Modelling, Programming and Computer Software. 2011. No. 37(254). (in Russian).
18. Hafsteinsson H., Levkovitz R., Mitra G. Solving large scale linear programming problems using an interior point method on a massively parallel SIMD computer. Parallel Algorithms and Applications. 1994. Vol. 4, no. 3-4. P. 301–316. DOI: 10.1080/10637199408915470.
19. Karypis G., Gupta A., Kumar V. A parallel formulation of interior point algorithms. Proceedings of the 1994 ACM/IEEE conference on Supercomputing (Supercomputing'94). Los Alamitos, CA, USA: IEEE Computer Society Press, 1994. P. 204–213. DOI: 10.1109/SUPERC.1994.344280.

20. 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.

21. Schmidhuber J. Deep learning in neural networks: An overview. Neural Networks. 2015. Vol. 61. P. 85–117. DOI: 10.1016/j.neunet.2014.09.003.

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

23. 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.

24. 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.

25. Ezhova N.A., Sokolinsky L.B. Parallel computation model BSF-MR. Control systems and information technologies. 2019. No. 3(77). P. 15–21. (in Russian).

26. Bird R.S. Lectures on Constructive Functional Programming. Constructive Methods in Computing Science. NATO ASI Series F: Computer and Systems Sciences, vol. 55 / ed. by M. Broy. Berlin, Heidelberg: Springer, 1988. P. 151–216.

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

28. Sokolinsky L.B. BSF-skeleton. 2019. URL: https://github.com/leonid-sokolinsky/BSF-skeleton (accessed: 24.01.2022).

29. Gropp W. MPI 3 and Beyond: Why MPI Is Successful and What Challenges It Faces. Recent Advances in the Message Passing Interface. EuroMPI 2012. Lecture Notes in Computer Science, vol. 7490 / ed. by J. Traff, S. Benkner, J. Dongarra. Berlin, Heidelberg: Springer, 2012. P. 1–9. DOI: 10.1007/978-3-642-33518-1_1.

30. Kale V. Shared-memory Parallel Programming with OpenMP. Parallel Computing Architectures and APIs. Boca Raton: Chapman, Hall/CRC, 2019. Chap. 14. P. 213–222. DOI: 10.1201/9781351029223-18/SHARED-MEMORY-PARALLEL-PROGRAMMING-OPENMP-VIVEK-KALE.

31. Kostenetskiy P., Semenikhina P. SUSU Supercomputer Resources for Industry and fundamental Science. Proceedings - 2018 Global Smart Industry Conference, GloSIC 2018. IEEE, 2018. Article 8570068. P. 1–7. DOI: 10.1109/GloSIC.2018.8570068.

32. Sokolinsky L.B., Sokolinskaya I.M. On generation of random linear programming problems on cluster computing systems. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2021. Vol. 10, no. 1. P. 38–52. (in Russian).

33. Sokolinsky L.B. Certificate of state registration of computer program no. 2021619526 Russian Federation. Generator of random linear programming problems FRaGenLP : no. 2021618165 : application 28.05.2021 : publ. 10.06.2021. (in Russian).

УДК 519.642.6, 004.021, 004.67 **DOI: 10.14529/cmse220104**

РАЗРАБОТКА КОМПЛЕКСА ПАРАЛЛЕЛЬНЫХ ПРОГРАММ РЕШЕНИЯ ОБРАТНЫХ ЗАДАЧ ГРАВИМЕТРИИ И МАГНИТОМЕТРИИ ДЛЯ СЕТОК БОЛЬШОЙ РАЗМЕРНОСТИ

© 2022 А.И. Третьяков

Институт математики и механики им. Н.Н. Красовского УрО РАН
(620990 Екатеринбург, ул. Софьи Ковалевской, д. 16)
E-mail: fr1z2rt@gmail.com

Поступила в редакцию: 29.10.2021

В настоящее время важнейшими задачами при исследовании структуры земной коры являются обратные задачи гравиметрии и задачи магнитометрии о нахождении поверхностей раздела сред на основе данных о гравитационном и магнитном поле, измеренном на некоторой площади земной поверхности. В основе методов решения этих задач лежат идеи итеративной регуляризации. После дискретизации эти задачи сводятся к системам нелинейных уравнений большой размерности, решение которых требует большого количества вычислительных ресурсов. Необходимость в повышении точности решения требует дополнительных вычислений, что влечет за собой увеличение времени счета. В работе описывается система удаленных вычислений и интегрированного в нее комплекса программ для графических ускорителей, реализующих наиболее быстрые и экономичные по памяти из разработанных ранее итерационных алгоритмов на основе градиентных методов. Эта система представляет собой веб-портал, являющийся универсальным решением для запуска задач на удаленных кластерах. Важнейшим преимуществом такого портала является его простота использования: при подключении к кластеру для осуществления вычислений более не требуется производить установку дополнительного ПО на самом кластере, также не требуется наличие привилегированной учетной записи для работы с кластером. Все что требуется — действующая учетная запись на исполняемом кластере, остальную работу по коммуникации с центром обработки данных (ЦОД) берет на себя портал. Портал может легко масштабироваться при росте количества пользователей, которые могут загружать необходимые алгоритмы и выполнять вычисления с помощью ЦОД.

Ключевые слова: обратная задача гравиметрии, обратная задача магнитометрии, веб-портал.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Третьяков А.И. Разработка комплекса параллельных программ решения обратных задач гравиметрии и магнитометрии для сеток большой размерности // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2022. Т. 11, № 1. С. 57–78. DOI: 10.14529/cmse220104.

Введение

При исследовании структуры земной коры особо важными являются обратная задача гравиметрии [1] о нахождении поверхностей раздела сред постоянной плотности по известным скачкам плотности и гравитационному полю, измеренному на некоторой площади земной поверхности, задача магнитометрии [2] о нахождении поверхностей раздела сред постоянной вертикальной намагниченности по известным скачкам намагниченности и вертикальной компоненте магнитного поля, измеренной на некоторой площади земной поверхности, и задача магнитометрии о нахождении поверхностей раздела сред постоянной произвольно направленной намагниченности по известным скачкам намагниченности и магнитному полю, измеренному на некоторой площади земной поверхности. Задачи описываются нелинейными интегральными уравнениями Фредгольма первого рода. При разработке методов

решения задач используются идеи итеративной регуляризации [3]. После дискретизации эти задачи сводятся к системам нелинейных уравнений большой размерности. Повышение точности решения задач достигается за счет использования более мелких сеток, однако это приводит к увеличению затрат на вычислительные ресурсы.

Решение геофизических задач — трудоемкий процесс, требующий использования сложного математического аппарата, а также передовых компьютерных технологий. При решении задач требуется хранение и обработка большого объема информации. Кроме того, алгоритмы обладают высокой вычислительной сложностью: увеличение объема входных данных существенно увеличивает время вычислений. Решение таких задач на персональном компьютере может занимать от нескольких часов до нескольких дней, поэтому целесообразно использовать параллельные вычисления как на графических ускорителях, так и на многопроцессорной системе. Использование графических ускорителей при этом является наиболее перспективным, так как позволяет сократить время решения задач на несколько порядков. Несмотря на значительное сокращение времени использование параллельных вычислений порождает ряд проблем. Во-первых, исследователю нужно быть не только специалистом в предметной области, но и в области вычислительной математики. Кроме этого, ему также необходимо владеть технологиями параллельного программирования. Во-вторых, вычислительные ресурсы, необходимые для исследований, имеют высокую стоимость и доступны далеко не каждому исследователю. Таким образом, получение результатов является сложной и трудоемкой работой, требующей не только мощных вычислительных ресурсов, но и больших затрат человеческого труда и финансов. Вследствие этого появляется необходимость в создании специализированного веб-портала для удаленных вычислений, который позволил бы существенно облегчить и удешевить эти процессы.

Целью данной работы является реализация разработанных ранее [4–7] быстрых и экономичных по памяти итерационных алгоритмов решения нелинейных обратных задач гравиметрии и магнитометрии о нахождении поверхностей раздела сред на основе градиентных методов в виде комплекса программ для графических ускорителей, разработка системы удаленных вычислений и интеграция в нее рассмотренных алгоритмов. Веб-портал позволяет запускать программы в удобном графическом интерфейсе (через веб-сервис) из любой точки мира при наличии сети интернет, не прибегая к использованию командной строки.

В работе [4] описано построение модифицированного метода наискорейшего спуска, линеаризованного метода сопряженных градиентов и его модифицированного варианта для решения задач гравиметрии и магнитометрии в многослойной среде. Разработаны алгоритмы для многоядерных процессоров и графических ускорителей. В работе [5] предлагается регуляризованный линеаризованный вариант нелинейного метода сопряженных градиентов для решения систем нелинейных уравнений, а также построены регуляризованные линеаризованные методы: модифицированный метод сопряженных градиентов и гибридный метод сопряженных градиентов. На основе методов разработаны алгоритмы для графических ускорителей Nvidia, которые используют свойство блочно-теплицевости матрицы производных для ее эффективного хранения в памяти. В работе [6] предложен модифицированный вариант покомпонентного градиентного метода для решения систем нелинейных уравнений. Модифицированный метод показал свою эффективность для больших углов намагниченности в задаче магнитометрии. В работе [7] разработаны параллельные алгоритмы для графических ускорителей на основе модифицированных аналогов α-процессов [8] для

решения задачи гравиметрии. Под α-процессом подразумевается рекуррентная формула $z^{k+1} = z^k - \frac{\langle A^\alpha S_n, S_n\rangle}{\langle A^{\alpha+1} S_n, S_n\rangle} S_n$, где α — некоторое фиксированное вещественное число.

В работе [9] описывается первый подход к системе удаленных вычислений. На данном этапе веб-портал позволяет пользователю запускать задание на вычисление решения обратной задачи гравиметрии и задачи Дирихле с использованием предложенных методов на МВС-1000/32. Всего на портале предустановлено 8 алгоритмов для решения данного типа задач. Пользователь может выбрать необходимый алгоритм, ввести входные параметры, выбрать количество процессоров и запустить задачу. После завершения расчетов у пользователя появляется возможность посмотреть изображение решения и скачать результаты. В работе [10] предложен ряд важных улучшений для веб-портала: система аутентификации пользователя, возможность использования графических ускорителей, выбор выходных файлов, по которым осуществляется построение изображений результатов расчета. Система аутентификации пользователя проверяет учетную запись пользователя на суперкомпьютере МВС-ИММ. Для обеспечения безопасности добавлено SSL-шифрование. Также добавлена возможность выбора планировщика вычислительных ресурсов. В работе [11] добавлена возможность работы с суперкомпьютером УРАН. Также имеются альтернативные наработки в данной отрасли. В работе [12] описывается библиотека для удаленного запуска задач, без интерфейса и хранилища алгоритмов. Данная библиотека отлично подходит для автоматизации запуска задач на удаленном кластере. В работе [13] описывается веб-портал для запуска распределенных задач на вычислительном кластере. Особенностью портала является фиксированный список решаемых задач и работа с конкретным вычислительным кластером. Также существует продукт UNICORE [14] — сложная многомодульная система. Для работы требуется установка и настройка нескольких модулей, клиента для работы с порталом. Создание и загрузку задач требуется производить каждому пользователю. На базе этого продукта есть реализованные решения запуска собственных веб-порталов. В работе [15] на базе UNICORE предлагается реализация веб-портала для высокопроизводительных вычислений в области медицины. Инструмент достаточно гибкий, но при этом необходимо установить или разработать некоторое количество модулей, для того, чтобы поддержать свой процесс запуска задач. Как правило, модули необходимо устанавливать на вычислительные узлы, находящиеся в одной сети с кластером.

Как правило, запросы на выполнение вычислений к веб-порталам осуществляются посредством различных методов API, представляющих собой комплексное решение, одно из которых описано в работе [16]. Работа представляет собой FirecREST API, обеспечивающий доступ к ресурсам высокопроизводительного компьютерного центра, что позволяет научным веб-порталам запускать задачи на удаленном кластере и впоследствии загружать на него (или скачивать с него) данные. Также имеется альтернативное решение в виде NEWT API, описанного в работе [17]. Это API поддерживает аутентификацию пользователей и их работу с очередью задач на вычисления, с файлами, папками и другими хранимыми данными.

В данной работе предлагается развитие специализированного веб-портала, описанного в работах [9–11], включающее в себя идеи из альтернативных разработок и оптимизации существующих наработок. Одна из целей — сделать инструмент простым для установки, настройки и использования. Веб-портал позволяет пользователям загружать алгоритмы для решения своих собственных задач и делится ими с другими пользователями системы. Так же появляется возможность подключения к различным вычислительным кластерам.

Статья имеет следующую структуру. В разделе 1 описываются постановки структурных обратных задач гравиметрии и магнитометрии. В разделе 2 описывается схема аппроксимации интегрального оператора в задаче магнитометрии и приводятся методы решения задач. В разделе 3 приводится описание комплекса программ, архитектура универсального веб-портала и описание численных экспериментов для модельной задачи гравиметрии. Заключение содержит краткую сводку результатов, полученных в работе, с указанием направления дальнейших исследований.

1. Структурные обратные задачи гравиметрии и магнитометрии

1.1. Постановка обратной задачи гравиметрии

Рассмотрим постановку обратной задачи гравиметрии для нескольких поверхностей раздела.

Введем трехмерную декартову систему координат, в которой плоскость $x0y$ совпадает с земной поверхностью, а ось z направлена вертикально вниз. Предполагается, что нижнее полупространство состоит из нескольких слоев постоянной плотности, разделенных искомыми поверхностями $S_l (l = 1, \ldots, L)$, где L — априорно известное число границ раздела (рис. 1).

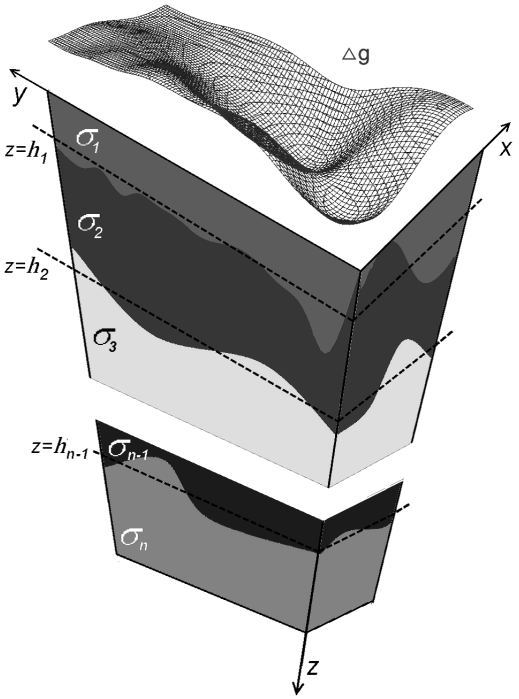


Рис. 1. Модель многослойной среды в задаче гравиметрии

Пусть поверхности раздела задаются уравнениями $\zeta_l = \zeta_l(x, y)$, скачки плотности на них равны $\Delta\sigma_1$, поверхности имеют горизонтальные асимптотические плоскости $\zeta_l = h_l$, т.е. $\lim_{|x|+|y|\to\infty} |\zeta_l (x, y) - h_l| = 0$.

Поле от суперпозиции границ с точностью до постоянного слагаемого равно

$$f \sum_{l=1}^{L} \Delta\sigma_l \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \left[\frac{1}{\sqrt{(x-x')^2+(y-y')^2+\zeta_l{}^2(x,y)}} - \frac{1}{\sqrt{(x-x')^2+(y-y')^2+h_l{}^2}} \right] dxdy = \Delta g\,(x', y', 0)\,, \qquad (1)$$

где f — гравитационная постоянная, $\Delta\sigma_l$ — скачок плотности на границе, $\zeta_l(x,y)$ — искомые поверхности.

Обратная задача состоит в определении формы поверхностей $\zeta_l(x,y)$ по данным об изменении силы тяжести на поверхности Земли $\Delta g(x',y',0)$. Исходными данными считаются плотности слоев σ_1, $\sigma_2..\sigma_n$ и глубины залегания плоскостей h_l.

После дискретизации уравнения (1) на сетке $n = M \times N$ (рис. 2) с шагами Δx, Δy, где задана правая часть $\Delta g(x,y,0)$ и аппроксимации интегрального оператора $A(z)$, имеем вектор правой части $F = (F_1, F_2, \ldots, F_n)$ размерности n, вектор решения $z = z_1 \oplus z_2 \oplus \ldots \oplus z_L$ (где $\oplus$ — конкатенация векторов) размерности L_n и систему n нелинейных уравнений $A(z) = F$.

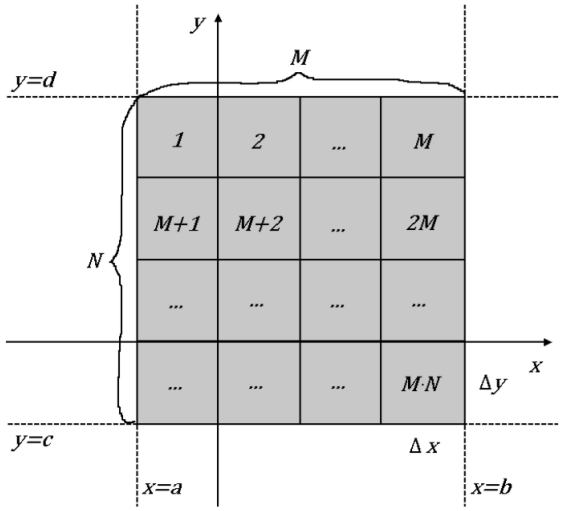


Рис. 2. Распределение индексов по сетке

1.2. Постановка обратной задачи магнитометрии

Предполагается, что нижнее полупространство состоит из нескольких слоев постоянной вертикально направленной намагниченности $J_l = J_l^z (l = 1, \ldots, L)$, разделенных искомыми поверхностями $S_l (l = 1, \ldots, L)$, где L — априорно известное число границ раздела (рис. 3). Магнитный эффект от такого полупространства равен сумме магнитных эффектов от всех поверхностей раздела.

Пусть поверхности раздела задаются уравнениями $\zeta_l = \zeta_l\ (x,\ y)$, скачки модулей векторов намагниченности на них равны ΔJ_l, поверхности имеют горизонтальные асимптотические плоскости $\zeta_l = h_l$, т.е. $\lim_{|x|+|y|\to\infty} |\zeta_l(x,y) - h_l| = 0$. Поле от суперпозиции границ с точностью до постоянного слагаемого равно

$$\sum_{l=1}^{L} \frac{\Delta J_l}{4\pi} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \left[\frac{\zeta_l(x,y)}{\left((x-x')^2 + (y-y')^2 + {\zeta_l}^2(x,y)\right)^{3/2}} - \right.$$
$$\left. - \frac{h}{\left((x-x')^2 + (y-y')^2 + {h_l}^2\right)^{3/2}} \right] dxdy = \Delta Z(x',y',0), \tag{2}$$

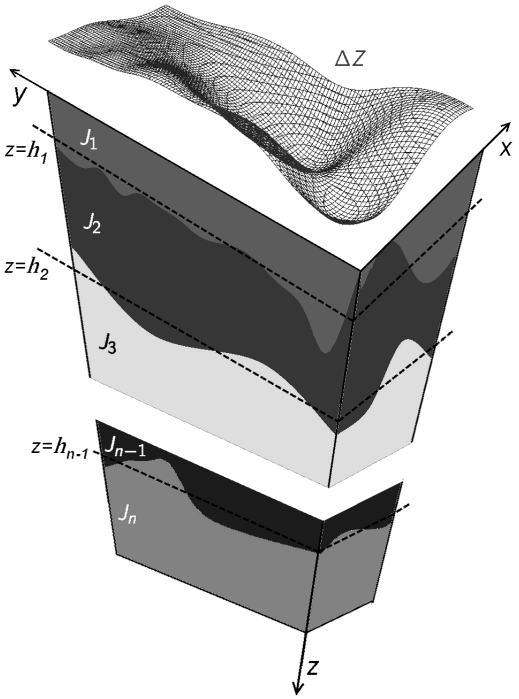


Рис. 3. Модель многослойной среды в задаче магнитометрии

где ΔJ — разность векторов намагниченности слоев, $\zeta_l(x,y)$ — искомые поверхности.

После дискретизации уравнения (2) на сетке $n = M \times N$ (рис. 2) с шагами Δx, Δy, где задана правая часть $\Delta Z(x,y,0)$ и аппроксимации интегрального оператора $B(z)$, имеем вектор правой части $F = (F_1, F_2, \ldots, F_n)$ размерности n, вектор решения $z = z_1 \oplus z_2 \oplus \ldots \oplus z_L$ (где $\oplus$ — конкатенация векторов) размерности L_n и систему n нелинейных уравнений $B[z] = F$.

1.3. Постановка обратной задачи магнитометрии для случая произвольно направленной суммарной намагниченности

Предполагается, что нижнее полупространство состоит из нескольких слоев постоянной произвольнонаправленной намагниченности $J_l = J_l^z (l = 1, \ldots, L)$, разделенных искомыми поверхностями (рис. 4). Магнитный эффект от такого полупространства равен сумме магнитных эффектов от всех поверхностей раздела.

Пусть поверхности раздела задаются уравнениями $\zeta_l = \zeta_l\ (x,\ y)$, скачки модулей векторов намагниченности на них равны ΔJ_l, поверхности имеют горизонтальные асимптотические плоскости $\zeta_l = h_l$, т.е. $\lim_{|x|+|y|\to\infty} |\zeta_l(x,y) - h_l| = 0$. Поле от суперпозиции границ с точностью до постоянного слагаемого равно

$$\frac{1}{4\pi}\int\limits_{-\infty}^{\infty}\int\limits_{-\infty}^{\infty}\left[\frac{\Delta J_x(x-x') + \Delta J_y(y-y') - \Delta J_z h}{\left((x-x')^2 + (y-y')^2 + h^2\right)^{3/2}} - \right.$$
$$\left. - \frac{\Delta J_x(x-x') + \Delta J_y(y-y') - \Delta J_z \zeta_l(x,y)}{\left((x-x')^2 + (y-y')^2 + \zeta^2(x,y)\right)^{3/2}}\right] dxdy = \Delta Z(x',y',0), \tag{3}$$

где $\Delta J = (\Delta \mathrm{J_x}, \Delta \mathrm{J_y}, \Delta \mathrm{J_z})$ — скачок намагниченности слоев.

После дискретизации уравнения (3) получаем систему нелинейных уравнений $B[z] = F$.

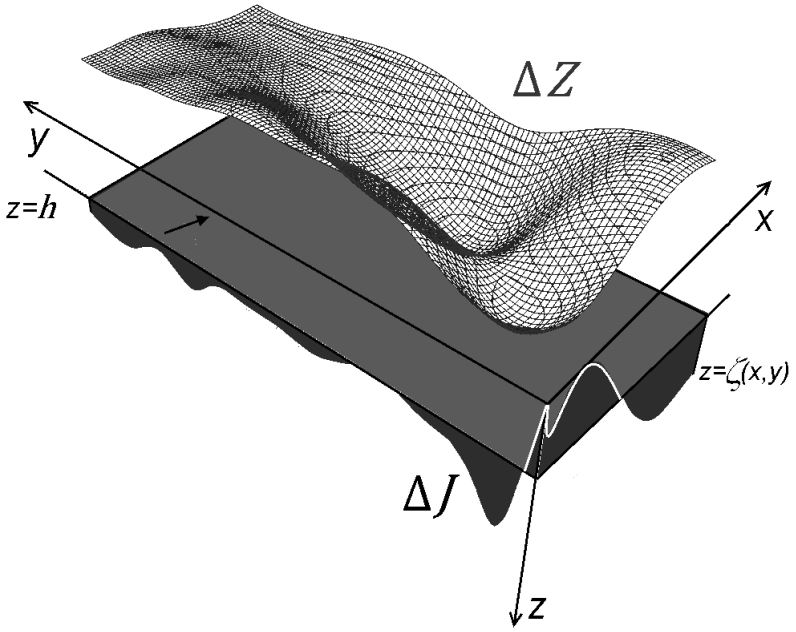


Рис. 4. Модель многослойной среды в задаче магнитометрии для случая произвольно направленной суммарной намагниченности

Обратные задачи гравиметрии и магнитометрии являются существенно некорректными задачами, решение которых обладает сильной чувствительностью к погрешностям правых частей, полученных в результате измерений и предварительной обработки геофизических данных. Поэтому при решении задач используются методы итеративной регуляризации.

2. Методы решения обратных задач гравиметрии и магнитометрии

2.1. Схема аппроксимации интегрального оператора в задаче магнитометрии

Для каждой ячейки сетки с индексами (i, j) строится элементарный параллелепипед высотой $|\Delta Z|$, $\Delta Z = (Z(x_i, y_j) - h)$, имеющий намагниченность $J = (J_x, J_y, J_z) = \text{sign}(\Delta Z)\Delta J$. Поле от криволинейной поверхности будет равным сумме полей от всех элементов разбиения:

$$\Delta Z(x, y) \approx \sum_{ij} \Delta Z_{ij}.$$

Найдем выражение для поля прямоугольного параллелепипеда со сторонами, параллельными координатным осям. Для этого воспользуемся теоремой Пуассона

$$\Delta Z = -\frac{1}{4\pi}\frac{\partial}{\partial z}\langle J, \nabla V\rangle = -\frac{1}{4\pi}\left(J_x V_{xz} + J_y V_{yz} + J_z V_{zz}\right).$$

Выражения для вторых производных V_{xz} и V_{yz} потенциала приводятся в [18]

$$V_{xz} = \left|\left|\left|(\eta - y) + \sqrt{(\xi - x)^2 + (\eta - y)^2 + (\zeta - z)^2}\right|_{\xi_1}^{\xi_2}\right|_{\eta_1}^{\eta_2}\right|_{\zeta_1}^{\zeta_2},$$

$$V_{yz} = \left|\left|\left|(\xi - x) + \sqrt{(\xi - x)^2 + (\eta - y)^2 + (\zeta - z)^2}\right|_{\xi_1}^{\xi_2}\right|_{\eta_1}^{\eta_2}\right|_{\zeta_1}^{\zeta_2}.$$

Получим выражение для V_{zz}. Производная потенциала однородного тела произвольной формы:

$$V_{zz} = 3\int_V \frac{(\zeta - z)^2}{\left((\xi - x)^2 + (\eta - y)^2 + (\zeta - z)^2\right)^{5/2}} d\xi d\eta d\zeta.$$

Подставляем пределы и поочередно интегрируем. Получаем

$$V_{zz} = \left|\left|\left|\tan^{-1}\left(\frac{(\xi - x)(\eta - y)}{(\zeta - z)\sqrt{(\xi - x)^2 + (\eta - y)^2 + (\zeta - z)^2}}\right)\right|_{\zeta_1}^{\zeta_2}\right|_{\eta_1}^{\eta_2}\right|_{\xi_1}^{\xi_2}.$$

Данная схема позволяет достичь более высокого уровня точности при меньших затратах вычислительных ресурсов.

2.2. Градиентные методы

Для решения обратных задач гравиметрии и магнитометрии для случая двуслойной среды в работе используются следующие методы:

• регуляризованный линеаризованный метод сопряженных градиентов (РЛМСГ) [5, 6]

$$z^{k+1} = z^k - \psi\frac{\left\langle p^k, S_\alpha\left(z^k\right)\right\rangle}{\left\|A'\left(z^k\right)p^k\right\|^2 + \alpha\left\|p^k\right\|^2}p^k, p^k = S_\alpha\left(z^k\right) + \beta^k p^{k-1}, p^0 = S_\alpha\left(z^0\right),$$

$$\beta^k = max\left\{0, \frac{\left\langle S_\alpha\left(z^k\right), S_\alpha\left(z^k\right) - S_\alpha\left(z^{k-1}\right)\right\rangle}{\left\|S_\alpha\left(z^{k-1}\right)\right\|^2}\right\}, \tag{4}$$

$$S_\alpha(z) = A'(z)^T\left(A(z) - F\right) + \alpha\left(z - z^0\right),$$

где z^0 — начальное приближение, z^k — приближенное решение на k-ой итерации, α — параметр регуляризации, ψ — демпфирующий множитель. В качестве условия останова используется (5) при достаточно малом ε.

$$\frac{\|A(z) - F\|}{\|F\|} < \varepsilon \tag{5}$$

• покомпонентный градиентный метод (ПГМ) [6]

$$z_i^{k+1} = z_i^k - \psi\frac{A_i\left(z^k\right) - F_i + \alpha\left\|z^k - z^0\right\|^2}{\left\|\nabla A_i\left(z^k\right)\right\|^2}\left(\frac{\partial A_i\left(z^k\right)}{\partial z_i}\right), \tag{6}$$

где z_i — i компонента приближенного решения, $i = 1..n$, $k \in \mathbb{N}$, ψ — демпфирующий множитель. В данном методе меньше действий по сравнению с методом сопряженных градиентов, но больше чувствительность к возмущенным данным.

• модифицированный регуляризованный линеаризованный метод сопряженных градиентов (МРЛМСГ) [5]

$$z^{k+1} = z^k - \psi \frac{\left\langle p^k, S_\alpha^0\left(z^k\right)\right\rangle}{\left\|A'\left(z^0\right)p^k\right\|^2 + \alpha\left\|p^k\right\|^2} p^k, p^k = S_\alpha^0\left(z^k\right) + \beta^k p^{k-1}, p^0 = S_\alpha^0\left(z^0\right),$$

$$\beta^k = max\left\{0, \frac{\left\langle S_\alpha^0\left(z^k\right), S_\alpha^0\left(z^k\right) - S_\alpha^0\left(z^{k-1}\right)\right\rangle}{\left\|S_\alpha^0\left(z^{k-1}\right)\right\|^2}\right\}, \tag{7}$$

$$S_\alpha^0(z) = A'\left(z^0\right)^T\left(A(z) - F\right) + \alpha\left(z - z^0\right),$$

где z^0 — начальное приближение, z^k — приближенное решение на k-ой итерации, α — параметр регуляризации, ψ — демпфирующий множитель. В качестве условия останова используется (5) при достаточно малом ε.

• модифицированные α-процессы [7]:

метод минимальной невязки (ММН)

$$z^{k+1} = z^k - \psi \frac{\left\langle (A'(z^0)+\overline{\alpha}I)\overline{S_\alpha}(z^k), \overline{S_\alpha}(z^k)\right\rangle}{\left\|(A'(z^0)+\overline{\alpha}I)\overline{S_\alpha}(z^k)\right\|^2}\overline{S_\alpha}\left(z^k\right); \tag{8}$$

метод наискорейшего спуска (МНС)

$$z^{k+1} = z^k - \psi \frac{\left\|\overline{S_\alpha}(z^k)\right\|^2}{\left\langle (A'(z^0)+\overline{\alpha}I)\overline{S_\alpha}(z^k), \overline{S_\alpha}(z^k)\right\rangle}\overline{S_\alpha}\left(z^k\right); \tag{9}$$

метод минимальной ошибки (ММО)

$$z^{k+1} = z^k - \psi \frac{\left\langle \left(A'\left(z^0\right) + \overline{\alpha}I\right)^{-1}\overline{S_\alpha}\left(z^k\right), \overline{S_\alpha}\left(z^k\right)\right\rangle}{\left\|\overline{S_\alpha}\left(z^k\right)\right\|^2}\overline{S_\alpha}\left(z^k\right), \tag{10}$$

где z^0 — начальное приближение, z^k — приближенное решение на k-ой итерации, α — параметр регуляризации, ψ — демпфирующий множитель. В качестве условия останова используется (5) при достаточно малом ε.

Можно заметить, что для модифицированных методов матрица производных интегрального оператора вычисляется в начальной точке и не пересчитывается в дальнейшем процессе. Матрица $A'\left(z^0\right)$ при $z^0 = \text{const}$ является блочно-теплицевой. На рис. 5 изображена структура матрицы производных интегрального оператора для случая двухслойной среды, где k, $l = 1..M$ — индекс блока, p, $q = 1..N$ — индекс внутри блока. Блоки с одинаковыми индексами совпадают, элементы внутри блока с одинаковыми индексами имеют одно и то же значение. На рис. 6 изображена структура матрицы производных интегрального оператора для случая многослойной среды. Она состоит из L больших блоков, где L — число границ раздела. Каждый такой блок имеет блочно-теплицеву структуру. Для хранения такой матрицы достаточно хранить одну строку, для вычисления любого элемента в матрице достаточно применить соответствующий сдвиг относительно сохраненной строки. Это позволяет сократить время счета и затраты на оперативную память.

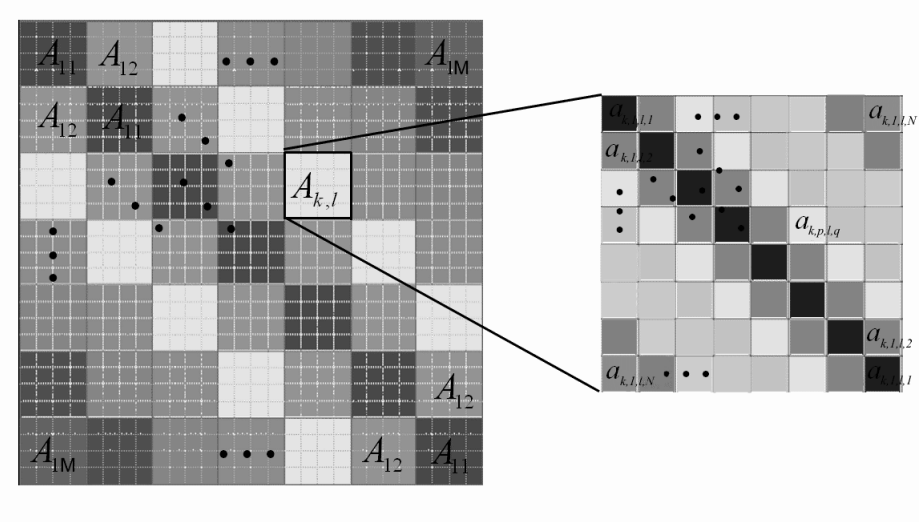


Рис. 5. Структура матрицы производных интегрального оператора для случая двухслойной среды

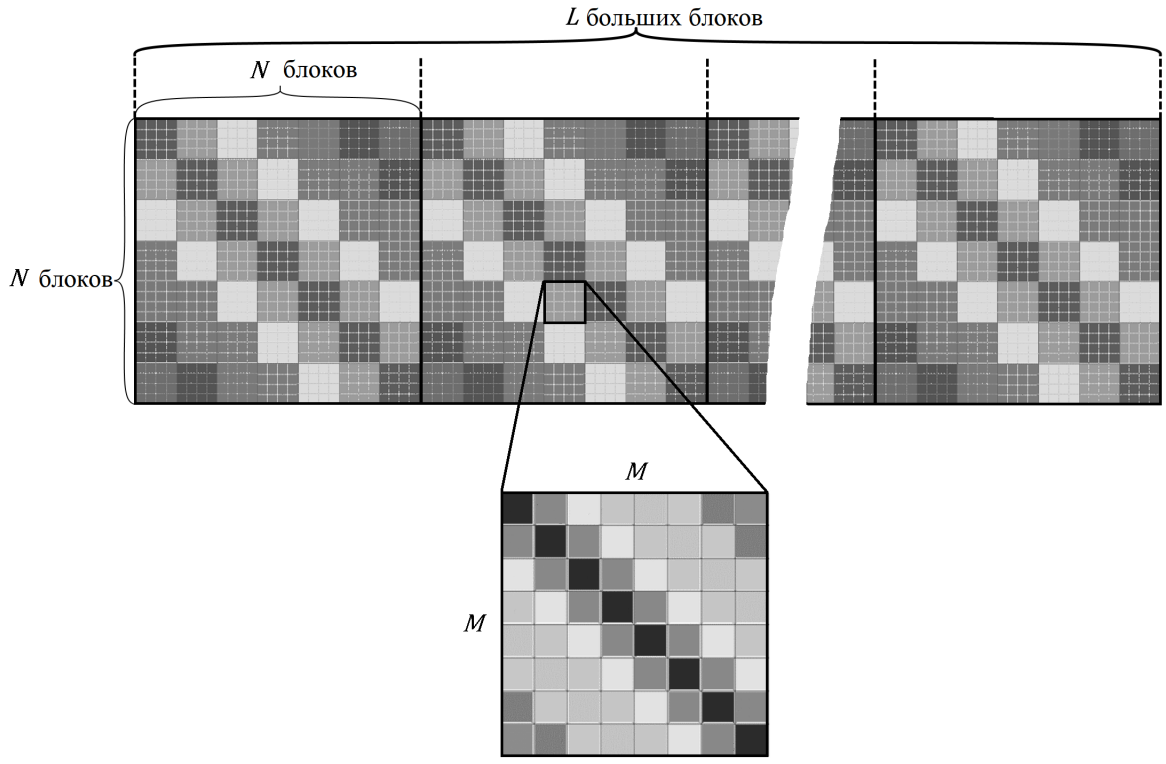


Рис. 6. Структура матрицы производных интегрального оператора для случая многослойной среды

Для решения обратных задач гравиметрии и магнитометрии для случая многослойной среды используются:

• линеаризованный метод сопряженных градиентов с весовыми множителями (ЛМСГВМ) [5]

$$
\begin{aligned}
&z^{k+1} = z^k - \psi \tfrac{\langle p^k, S_\alpha(z^k)\rangle}{\|A_k p^k\|^2 + \alpha\|p^k\|^2} p^k, \\
&p^k = v^k + \beta^k p^{k-1}, p^0 = v^0, v^0 = \Lambda S_\alpha\left(z^k\right), \\
&\beta^k = max\left\{0, \tfrac{\langle v^k, v^k - v^{k-1}\rangle}{\|v^{k-1}\|^2}\right\}, \\
&S_\alpha(z) = A_k^T\left(A(z) - F\right),
\end{aligned}
\tag{11}
$$

где z^0 — начальное приближение, z^k — приближенное решение на k-ой итерации, α — параметр регуляризации, ψ — демпфирующий множитель, Λ — диагональная матрица, состоящая из весовых множителей. В качестве условия останова используется (5) при достаточно малом ε.

• модифицированный линеаризованный метод сопряженных градиентов с весовыми множителями (ММСГВМ) [5]

$$
\begin{aligned}
&z^{k+1} = z^k - \psi \frac{\langle p^k, S_0(z^k)\rangle}{\|A'(z^0)p^k\|} p^k, \\
&p^k = v^k + \beta^k p^{k-1}, \quad p^0 = v^0, \\
&\beta^k = \max\{\frac{\langle v^k, v^k - v^{k-1}\rangle}{\|v^{k-1}\|}, 0\}, \\
&v = \Lambda S(z), \\
&S_0(z) = A'(z^0)^*(A(z) - F),
\end{aligned}
\tag{12}
$$

где z^0 — начальное приближение, z^k — приближенное решение на k-ой итерации, α — параметр регуляризации, ψ — демпфирующий множитель, Λ — диагональная матрица, состоящая из весовых множителей. В качестве условия останова используется (5) при достаточно малом ε.

3. Комплекс программ и вычислительные эксперименты

3.1. Описание комплекса программ

На основе вышеописанных методов для решения структурных обратных задач гравиметрии и магнитометрии разработаны алгоритмы для графических ускорителей, входящие в состав комплекса программ. Алгоритмы встроены в систему удаленных вычислений. Описание входных и выходных данных, а также рекомендации по использованию также загружены на веб-портал. Для запуска задачи необходимо загрузить файл с аномальным полем, указать глубину и количество графических ускорителей. После окончания работы программы на выходе будет файл с восстановленной поверхностью и ее изображение.

На данный момент в системе удаленных вычислений доступны следующие методы, реализованные для графических ускорителей (рис. 7):

- Задача гравиметрии для одной границы:
  - регуляризованный линеаризованный МСГ (4);
  - модифицированный регуляризованный линеаризованный МСГ (7);
  - гибридный МСГ [6];

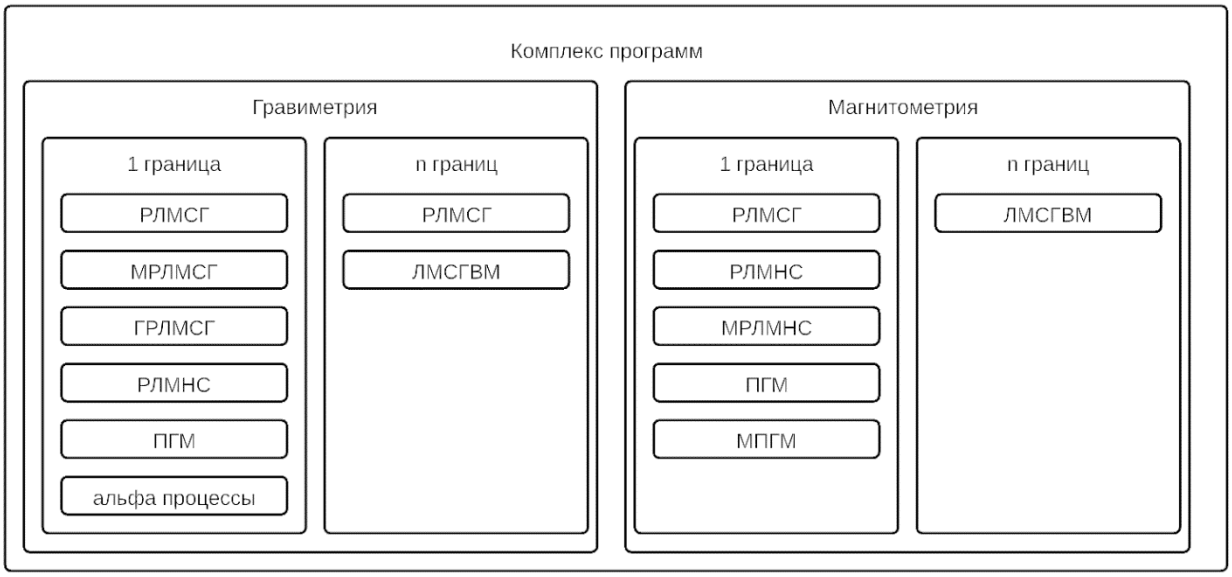


Рис. 7. Структура комплекса программ

  - линеаризованный МНС [6];
  - покомпонентный градиентный метод (6);
  - модифицированные альфа процессы (8)–(10).
- Задача гравиметрии для нескольких границ:
  - линеаризованный МСГ с весовыми множителями (11);
  - модифицированный МСГ с весовыми множителями (12).
- Задача магнитометрии для одной границы:
  - регуляризованный линеаризованный МСГ (4);
  - покомпонентный градиентный метод (6);
  - модифицированный покомпонентный градиентный метод [6];
  - линеаризованный МНС [7];
  - модифицированный МНС [7].
- Задача магнитометрии для нескольких границ:
  - линеаризованный МСГ с весовыми множителями (11).

Рекомендации по использованию методов:

– для зашумленных данных в задачах гравиметрии и магнитометрии лучше подходят модифицированные α-процессы;
– для модельных задач и слабозашумленных данных рекомендуется покомпонентный метод;
– для задач с несколькими границами необходимо использовать МСГ с весовыми множителями.

Алгоритмы решения обратной задачи гравиметрии на основе градиентных методов численно реализованы на суперкомпьютере кластерного типа «Уран». Алгоритмы реализованы на видеоускорителях NVIDIA Tesla с помощью технологии CUDA и библиотеки Cublas.

На первом шаге на всех графических ускорителях выделяется память под вектора и матрицы, копируется вектор правой части уравнения. Затем на каждом устройстве запускается набор ядерных функций. Данные функции реализованы таким образом, чтобы минимизировать необходимость синхронизации данных между различными ускорителями.

Например, в алгоритме, реализующем РЛМСГ, имеется 3 ядерных функции, которые выполняются в рамках каждой итерации:

1. На данном этапе вычисляются элементы матрицы A', компоненты векторов S_α, p^k, β^k. После завершения функции CalculateStep1 на центральном процессоре собирается коэффициент β^k и передается на все графические ускорители.
2. Затем вычисляется числитель и знаменатель основного уравнения. После завершения функции CalculateStep2 на центральном процессоре производится деление числителя на знаменатель.
3. Полученные вычисления используются для окончательного расчета вектора z^{k+1} в функции CalculateZ.

Предполагается, что ускорители установлены на 1 вычислительном узле — это позволяет минимизировать сетевые расходы. Центральный процессор в такой схеме занимается диспетчеризаций видеокарт. Для сетки 512×512 хранение матрицы производных требует 512 ГБ. Поэтому элементы матрицы производной оператора вычисляются «на лету» во время выполнения векторно-матричных операций. Используется способ автоматической настройки параметров выполнения подпрограмм на GPU: найден оптимальный размер блока для эталонного размера сетки 256×256, для другого размера сетки параметры пропорционально пересчитываются.

3.2. Описание веб-портала

В данной работе предлагается веб-портал, в котором учтены недостатки работ [9–11] и используются удачные идеи из работ [12, 14]. Из работы [12] используется идея общения с кластером с использованиям личных учетных данных пользователя, с учетом этого появляется возможность работы с произвольным кластером. Идея о возможности самостоятельной загрузки алгоритмов пользователями взята из [14].

Предлагаемый веб-портал является универсальным решением для запуска задач на удаленных кластерах. При подключении удаленного кластера не требуется производить установку дополнительного ПО на самом кластере, также нет необходимости в привилегированной учетной записи. Общение с кластером ведется через ssh-туннель. Пользователю, для работы с сервисом, требуется действующая учетная запись. Взаимодействие с центром обработки данных (ЦОД) берет на себя портал, пользователям нет необходимости в прямом подключении к кластеру.

Преимуществами веб-портала являются:

- Нет привязки к конкретному кластеру и его ПО.
- Нет ограничений на список алгоритмов/задач. Пользователи сами могут загружать алгоритмы, делать их публичными на всех/определенную группу людей.
- Гибкая настройка пред- и постобработки задач.

Важной особенностью является то, что работа по планированию (учет времени и приоритизация) пользовательских задач остается на стороне кластера, что позволяет сильно упростить логику взаимодействия с удаленными узлами и не конфликтовать с пользователями, использующими кластер напрямую.

Входные данные и результаты работы хранятся непосредственно на кластере — это позволяет уменьшить требования к вычислительным ресурсам и объему хранилища. В базе данных хранятся учетные данные пользователей, авторизационная информация и список задач. Система управления базой данных (СУБД) может быть установлена вместе с основ-

Рис. 8. Структура веб-портала

ным веб-приложением на одних и тех же серверах, т.к. объем хранимых данных невелик. При необходимости СУБД может быть вынесена на отдельные серверы.

Портал может масштабироваться при росте количества пользователей и задач за счет реплицирования основного приложения и шардирования БД по пользователям. Так же за счет реплицирования повышается надежность и отказоустойчивость.

Веб-портал имеет следующую структуру (рис. 8):

- Веб-интерфейс — написан с использованием технологии blazor [19] (реализация web assembler под .net core). Blazor — перспективная технология, позволяющая писать код для браузера на языке C#. Впоследствии код компилируется веб-ассемблером в двоичный код для последующего исполнения браузером. Данный подход позволяет писать достаточно эффективный код, т.к. не происходит преобразований в javascript-код и последующей интерпретации. Общение с серверной частьб происходит по протоколу http.
- Серверная часть — использует ASP.NET [20] core фреймворк и язык C#, для взаимодействия с базой данных используется EF core [21].
- СУБД — может использоваться любая реляционная СУБД, для которой реализован драйвер под EF core.

На рис. 9 представлена структура БД. Таблицы с префиксом AspNet созданы фреймворком ASP.NET для корректной работы авторизации. В таблице ssh_user_settings хранится информация об учетных данных пользователей, необходимых для подключения к кластеру. В таблице programs содержится информация о загруженных алгоритмах и скриптах для предобработки входных данных и постобработки результатов вычислений. Таблица tasks хранит информацию о запущенных и завершенных заданиях на расчеты.

Логические модули:

- Раздел администрирования. Здесь ведется учет пользователей и групп.
- Программы. В данном разделе производится загрузка новых программ, редактирования существующих и запуск.
- Задания. В данном разделе находится список запущенных задач. Для исполняющихся задач можно посмотреть время запуска и статус. Для готовых есть возможность скачать архив с выходными данными.
- Настройки. Здесь описываются настройки для подключения к вычислительным кластерам: ip-адрес кластера, ssh-ключи для подключения (так же есть возможность воспользоваться логином и паролем). Использование ssh-ключей позволяют сделать взаимодействие с кластером более безопасным, т.к. не требуется передачи и хранения логина и пароля, а также появляется возможность отзыва скомпрометированных ключей.

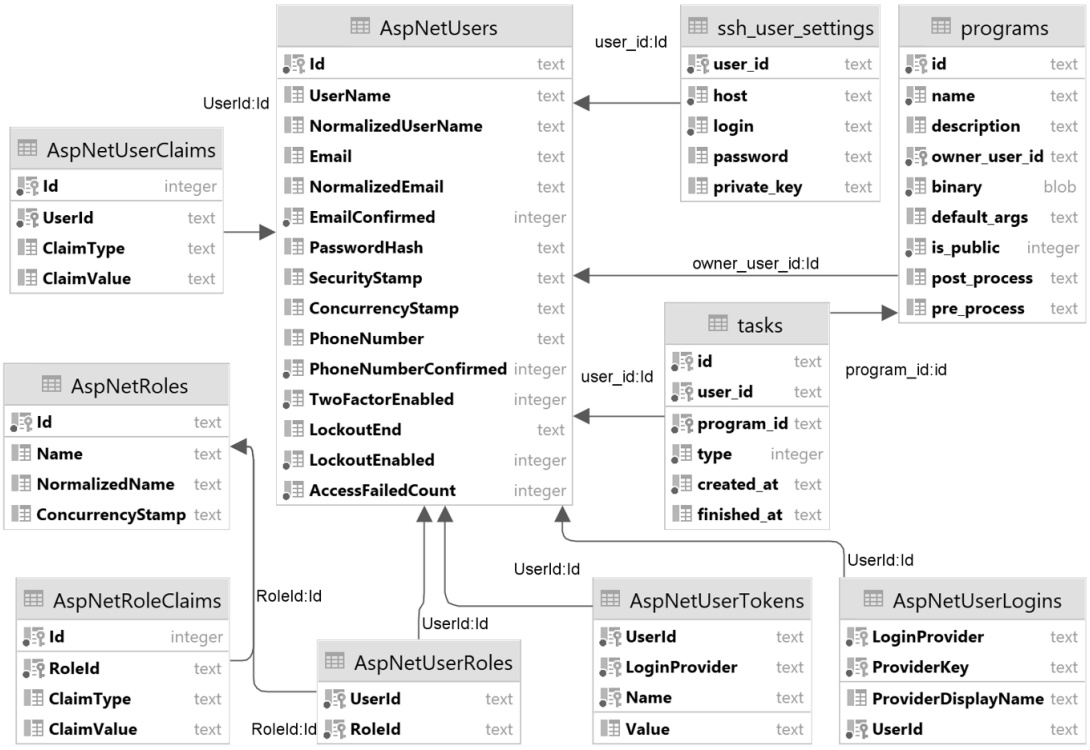


Рис. 9. Структура базы данных

На рис. 10 представлена схема взаимодействия пользователя с веб-порталом. При первом входе на веб-портал пользователю необходимо пройти регистрацию и указать учетные данные для подключения к вычислительному кластеру. Далее он может создать и настроить алгоритмы решения необходимых ему задач (загруженные программы могут использоваться в личных целях, либо могут быть доступны всем пользователям). Загружать можно как готовый исполняемый файл, так и исходный код (при наличии соответствующего компилятора или интерпретатора). После загрузки формируется шаблон для запуска задачи, в котором описываются параметры, необходимы для запуска и требуемые шаги для предобработки входных данных и постобработки выходных файлов. После этого пользователь выбирает алгоритм, загружает входные данные и ставит задание в очередь на выполнение. После завершения выполнения задания происходит постобработка выходных данных и появляется возможность скачать результаты работы.

Предварительно на портал загружен набор алгоритмов для решения обратных задач гравиметрии и магнитометрии. Запуск этих задач возможен на многопроцессорных вычислительных узлах, либо на узлах, оборудованных графическими ускорителями.

При запуске задачи создается соответствующая запись в БД. После чего производится копирование входных данных на выбранный кластер и запуск задачи с переданными параметрами.

Далее, до тех пор, пока задача не завершится, производится периодический опрос статуса. После завершения работы в БД появляется отметка об окончании и ссылка на архив с выходными данными.

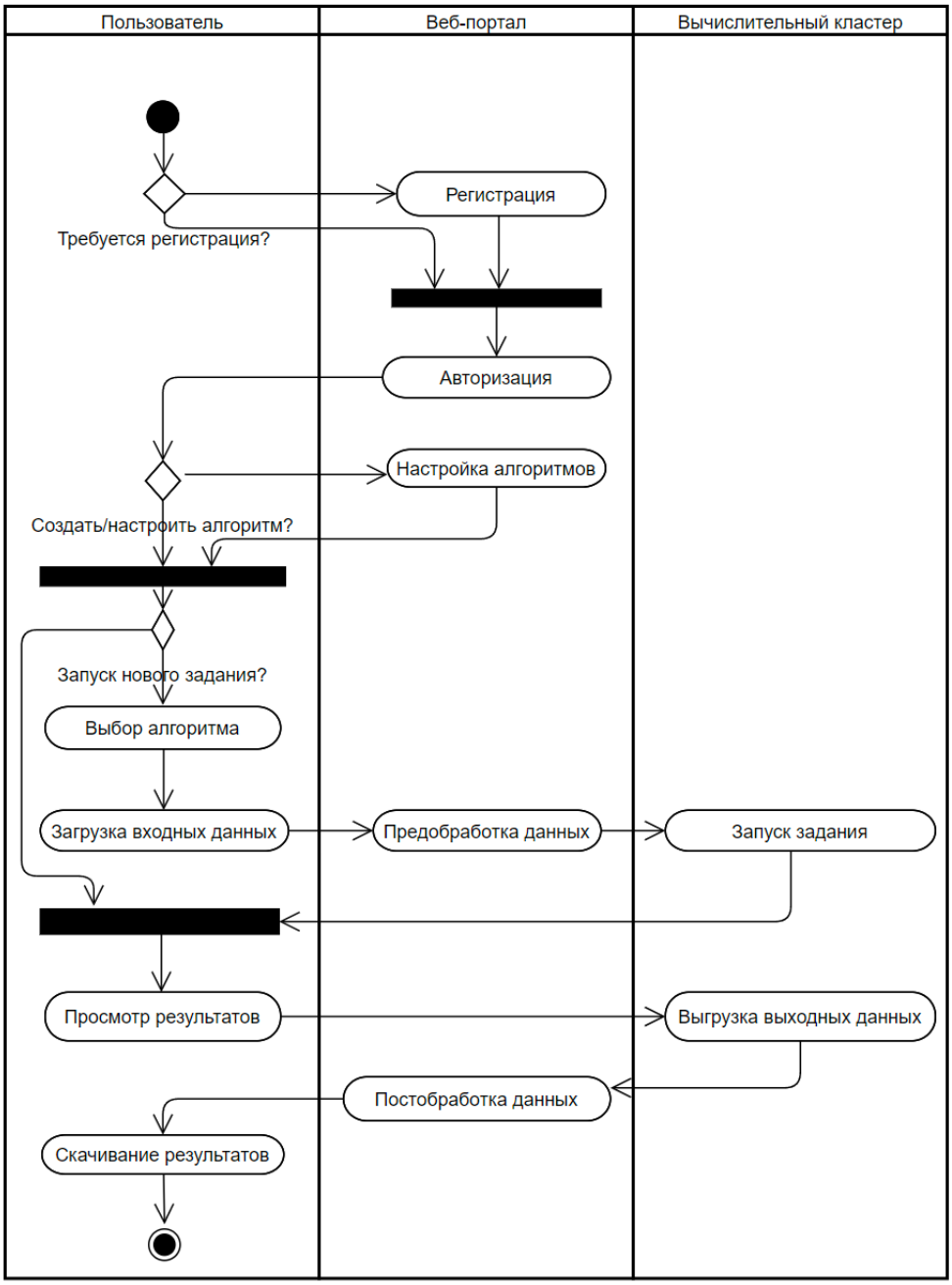


Рис. 10. Схема взаимодействия пользователя с веб-порталом

3.3. Численные эксперименты

Численные эксперименты проводились на суперкомпьютере УРАН с использованием графических ускорителей M2090 с помощью разработанного веб-портала.

Целью эксперимента является сравнение методов ЛМСГВМ и ММСГВМ решения обратной задачи гравиметрии для нескольких границ по времени счета с помощью разработанного веб-портала. Сравнение эффективности методов для одной границы приводятся в работах [5–7].

В экспериментах использовалась квазиреальная модель 4-слойной среды размерностью $2^9 \times 2^9$. На рис. 11 представлено суммарное гравитационное поле, полученное путем решения прямой задачи с тремя модельными поверхностями S_1, S_2, S_3 (рис. 12) с асимптотическими

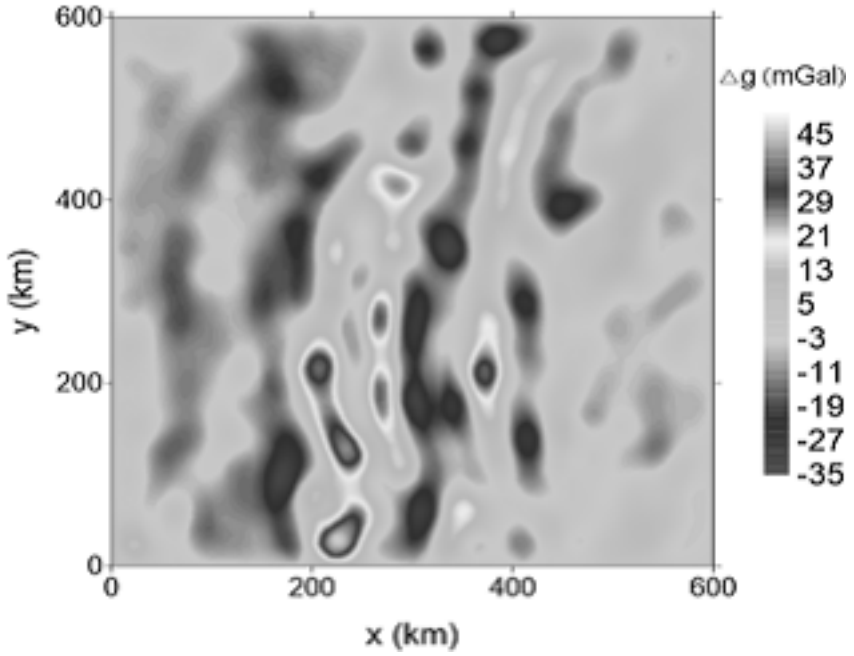


Рис. 11. Суммарное гравитационное поле

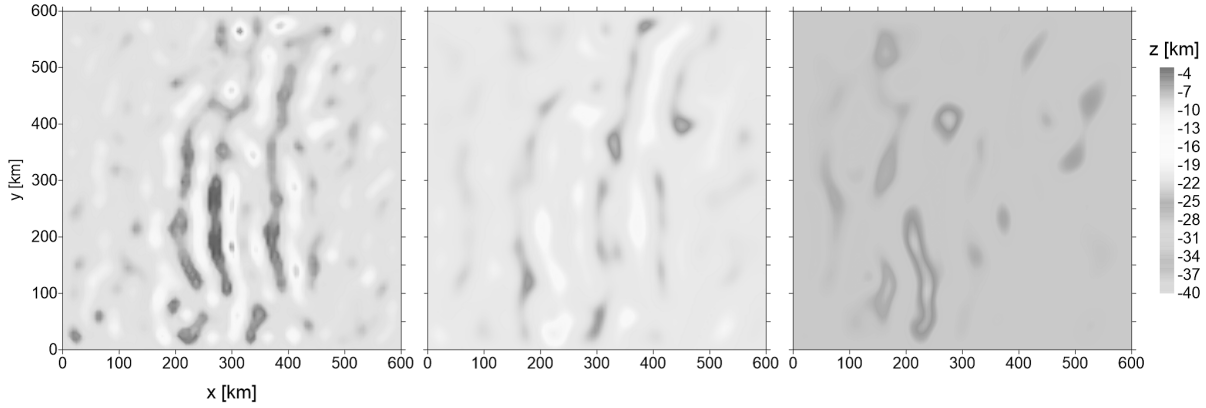


Рис. 12. Модельные поверхности S_1, S_2, S_3

плоскостями $H_1 = 10$ км, $H_2 = 20$ км, $H_3 = 30$ км и скачками плотности $\sigma_1 = \sigma_2 = \sigma_3 = 0.2$ г/см^3.

При решении задачи использовались следующие параметры: $\psi = 1$, $\alpha = 0.1$. Значения были подобраны экспериментальным путем. Условием останова было выбрано $\frac{\|A(z)-F\|}{\|F\|} < \varepsilon,\ \varepsilon = 0.01$. Относительная погрешность при этих условиях составляет $\delta = \frac{\|z-z^*\|}{\|z^*\|} < 0.01$.

Запуск расчетов производится на странице списка загруженных программ. Результаты вычислений можно посмотреть на странице списка задач. Для приведенных алгоритмов для постобработки результатов используется модуль взаимодействия с Surfer. Это позволяет вывести изображения поверхностей на странице результатов.

Для сравнения времени счета решения задачи используются формулы ускорения и эффективности параллельных алгоритмов:

$$S_m = \frac{T_1}{T_m}, \quad E_m = \frac{S_m}{m},$$

Таблица 1. Результаты численных экспериментов

Алгоритм	Время (1 gpu), мин	Время (8 gpu), мин	Ускорение, S	Эффективность, E
ЛМСГВМ	12	1.73	6.93	0.86
ММСГВМ	5.3	0.74	7.16	0.89

где S_m — ускорение для m графических ускорителей, T_1 — время счета на 1 графическом ускорителе, T_m — время счета на m графических ускорителях, E_m — эффективность для m графических ускорителей.

В таблице представлены результаты эксперимента для ММСГ, МСГ с весовыми множителями:

Можно заметить, что методы показали высокую эффективность: 86% и 89% соответственно. Для ММСГВМ время счета на 8 графических ускорителей составляет менее 1 минуты для модельной задачи на сетке $2^9 \times 2^9$.

Заключение

В статье затронута проблема высокоточного исследования структуры земной коры, для преодоления которой требуется наиболее эффективно уметь решать обратные задачи гравиметрии и задачи магнитометрии о нахождении поверхностей раздела сред на основе данных о гравитационном и магнитном поле, измеренном на некоторой площади земной поверхности. Для задачи магнитометрии для случая произвольно направленной суммарной намагниченности построена схема аппроксимации интегрального оператора. Предложенная схема при меньших затратах вычислительных ресурсов позволяет достичь более высокого уровня точности. На основе метода решения данной задачи, а также на базе других методов градиентного типа разработан комплекс параллельных программ решения обратных задач гравиметрии и магнитометрии для сеток большой размерности для графических ускорителей. Также в статье поднимается проблема обмена перечисленным комплексом программ для дальнейших вычислений всеми заинтересованными лицами. Для ее решения была разработана система удаленных вычислений или веб-портал, разрешающий совместное использование комплексом программ, встроенным в него. Также веб-портал позволяет загружать пользовательские алгоритмы, что помогает распространять и делиться другим пользователям своими наработками. Система удаленных вычислений проста в установке и настройке и при необходимости может быть установлена на другие серверы. Для задачи гравиметрии проведены численные эксперименты на суперкомпьютере УРАН для методов ЛМСГВМ и ММСГВМ с использованием графических ускорителей M2090. ММСГВМ показал хорошую эффективность порядка 89%. Время счета на 8 графических ускорителях составляет менее 1 минуты против 5.3 минут на 1 ускорителе. В качестве одного из перспективных улучшений веб-портала может быть добавлена возможность автоматизировано проводить серию экспериментов над входными данными, применяя разные алгоритмы из комплекса программ, что позволит построить наиболее оптимальное решение задачи гравиметрии и задачи магнитометрии.

Литература

1. Нумеров Б.В. Интерпретация гравитационных наблюдений в случае одной контактной поверхности // Докл. АН СССР. 1930. № 21. С. 569–574.

2. Малкин Н.Р. О решении обратной магнитометрической задачи для случая одной контактной поверхности (случай пластообразно залегающих масс) // Докл. АН СССР. Сер. А. 1931. № 9. С. 232–235.

3. Bakushinskiy A., Goncharsky A. Ill-posed problems: theory and applications. Springer Science & Business Media, 1994. DOI: 10.1007/978-94-011-1026-6.

4. Акимова Е.Н., Мисилов В.Е., Скурыдина А.Ф., Третьяков А.И. Градиентные методы решения структурных обратных задач гравиметрии и магнитометрии на суперкомпьютере «Уран» // Вычислительные методы и программирование. 2015. Т. 16, № 1. С. 155–164. DOI: 10.26089/NumMet.v16r116.

5. Akimova E.N., Misilov V.E., Tretyakov A.I. Optimized algorithms for solving structural inverse gravimetry and magnetometry problems on GPUs // Parallel Computational Technologies. Vol. 753. Springer, 2017. P. 144–155. Communications in Computer and Information Science. DOI: 10.1007/978-3-319-67035-5_11.

6. Akimova E.N., Misilov V.E., Tretyakov A.I. Modified Componentwise Gradient Method for Solving Structural Magnetic Inverse Problem // Parallel Computational Technologies. Vol. 910. Springer, 2018. P. 162–173. Communications in Computer and Information Science. DOI: 10.1007/978-3-319-99673-8_12.

7. Akimova E.N., Misilov V.E., Tretyakov A.I. Using Multicore and Graphics Processors to Solve the Structural Inverse Gravimetry Problem in a Two-Layer Medium by Means of α-Processes // Parallel Computational Technologies. Vol. 1063. Springer, 2019. P. 285–296. Communications in Computer and Information Science. DOI: 10.1007/978-3-030-28163-2_20.

8. Васин В.В. Основы теории некорректных задач. Новосибирск: Изд-во СО РАН, 2020. 313 с.

9. Акимова Е.Н., Гемайдинов Д.В. Параллельные алгоритмы решения обратной задачи гравиметрии и организация удаленного взаимодействия между МВС-1000 и пользователем // Вычислительные методы и программирование. 2008. Т. 9. С. 129–140.

10. Акимова Е.Н., Белоусов Д.В., Мисилов В.Е. Алгоритмы решения обратных геофизических задач на многопроцессорных вычислительных системах // Сибирский журнал вычислительной математики. 2013. Т. 16, № 2. С. 107–121.

11. Akimova E.N., Misilov V.E., Skurydina A.F., Martyshko M.P. Specialized web portal for solving problems on multiprocessor computing systems // CEUR Workshop Proceedings. 2015. Vol. 1513. P. 123–129.

12. Tsidaev A. .NET Library for Seamless Remote Execution of Supercomputing Software // CEUR-WS. 2017. Vol. 1990. P. 79–83.

13. Kuklin E., Pravdin S. A Web-based System for Launching Large Experiment Series on Supercomputers // CEUR-WS. 2018. Vol. 2281. P. 136–145.

14. Erwin D.W. UNICORE — a grid computing environment // Concurrency and Computation: Practice and Experience. 2002. Vol. 14. P. 1395–1410. DOI: 10.1002/cpe.691.

15. Grosch A., Waldmann M., Göbbert J.H., Lintermann A. A Web-Based Service Portal to Steer Numerical Simulations on High-Performance Computers // 8th European Medical and Biological Engineering Conference. Vol. 80. Springer, 2020. P. 57–65. DOI: 10.1007/978-3-030-64610-3_8.

16. Cruz F.A., Dabin A.J., Dorsch J.P., *et al.* FirecREST: a RESTful API to HPC systems // IEEE/ACM International Workshop on Interoperability of Supercomputing and Cloud Technologies (SuperCompCloud). 2020. P. 21–26. DOI: 10.1109/SuperCompCloud51944.2020.00009.

17. Cholia S., Skinner D., Boverhof J. NEWT: A RESTful service for building High Performance Computing web applications // Gateway Computing Environments Workshop (GCE). 2010. P. 1–11. DOI: 10.1109/GCE.2010.5676125.

18. Шванк О.А., Люстих Е.Н. Интерпретация гравитационных наблюдений. Л.: Гостоптехиздат, 1947. 400 с.

19. Introduction to ASP.NET Core Blazor. Microsoft. 2021. URL: `https://docs.microsoft.com/en-us/aspnet/core/blazor` (дата обращения: 29.10.2021).

20. Документация по ASP.NET. Microsoft. 2021. URL: `https://docs.microsoft.com/ru-ru/aspnet/core` (дата обращения: 29.10.2021).

21. Entity Framework Core. Microsoft. 2021 URL: `https://docs.microsoft.com/ru-ru/ef/core` (дата обращения: 29.10.2021).

Третьяков Андрей Игоревич, старший программист Института математики и механики им. Н.Н. Красовского Уральского отделения РАН (Екатеринбург, Российская Федерация)

DOI: 10.14529/cmse220104

DEVELOPMENT OF THE PARALLEL PROGRAMS COMPLEX FOR SOLVING THE INVERSE GRAVIMETRIC AND MAGNETOMETRY PROBLEMS FOR LARGE GRIDS

© 2022 A.I. Tretyakov

N.N. Krasovskii Institute of Mathematics and Mechanics, UrB RAS
(S. Kovalevskaya st. 16, Ekaterinburg, 620990 Russia)
E-mail: fr1z2rt@gmail.com

Received: 29.10.2021

The most important problems in studying of the structure of the earth's crust are the inverse problems of gravimetry and the problems of magnetometry. The problem is in finding interfaces between layers with different constant densities using known gravitational data. The methods for solving these problems are based on the ideas of iterative regularization. After discretization, these are reduced to systems of nonlinear functions of large dimensions. The need to improve the accuracy of the results of solving problems entails an increase in the computation time. The paper describes a remote computing system and an integrated program complex for graphics accelerators that implement the fastest and most memory-efficient algorithms developed earlier and based on gradient methods. The remote computing system considered in the work is a web portal that is a universal solution for launching tasks on remote clusters. The most important advantage of the portal is the simplicity of its use: when connecting to a cluster to carry out calculations, you no longer need to install additional software on the cluster itself, and you do not need a privileged account to work with the cluster. All that is required is a valid account on the remote cluster, the rest of the work on communication with the data processing center (DPC) is taken over by the portal. The portal can easily scale with the growth of the number of users who can download the necessary algorithms and perform calculations using the data center.

Keywords: inverse gravimetry problem, inverse magnitometry problem, web-portal.

FOR CITATION

Tretiakov A.I. Development of the Parallel Programs Complex for Solving the Inverse Gravimetric and Magnetometry Problems for Large Grids. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2022. Vol. 11, no. 1. P. 57–78. (in Russian) DOI: 10.14529/cmse220104.

References

1. Numerov B.V. Interpretation of gravity observations for one contact surface. Proc. USSR Acad. Sci. 1930. No. 21. P. 569–574. (in Russian)

2. Malkin N.R. On solution of inverse magnetic problem for one contact surface (the case of layered masses). Proc. USSR Acad. Sci. 1931. No. 9. P. 232–235. (in Russian)

3. Bakushinskiy A., Goncharsky A. Ill-posed problems: theory and applications. Springer Science & Business Media, 1994. DOI: 10.1007/978-94-011-1026-6.

4. Akimova E.N., Misilov V.E., Skurydina A.F., Tretyakov A.I. Gradient methods for solving inverse gravimetry and magnetometry problems on the Uran supercomputer. Numerical Methods and Programming. 2015. Vol. 16. P. 155–164. (in Russian) DOI: 10.26089/NumMet.v16r116.

5. Akimova E.N., Misilov V.E., Tretyakov A.I. Optimized algorithms for solving structural inverse gravimetry and magnetometry problems on GPUs. Parallel Computational Technologies. Vol. 753. Springer, 2017. P. 144–155. Communications in Computer and Information Science. DOI: 10.1007/978-3-319-67035-5_11.

6. Akimova E.N., Misilov V.E., Tretyakov A.I. Modified Componentwise Gradient Method for Solving Structural Magnetic Inverse Problem. Parallel Computational Technologies. Vol. 910. Springer, 2018. P. 162–173. Communications in Computer and Information Science. DOI: 10.1007/978-3-319-99673-8_12.

7. Akimova E.N., Misilov V.E., Tretyakov A.I. Using Multicore and Graphics Processors to Solve the Structural Inverse Gravimetry Problem in a Two-Layer Medium by Means of α-Processes. Parallel Computational Technologies. Vol. 1063. Springer, 2019. P. 285–296. Communications in Computer and Information Science. DOI: 10.1007/978-3-030-28163-2_20.

8. Vasin V.V. Fundamentals of the theory of ill-posed problems. Novosibirsk, SB RAS, 2020. 313 p.

9. Akimova E.N., Gemaidinov D.V. Parallel Algorithms for Solving the Inverse Gravimetry Problem and Organization of Remote Communication between PCS1000 and the User. Numerical Methods and Programming. 2008. Vol. 9, no. 1. P. 129–140. (in Russian)

10. Akimova E.N., Belousov D.V., Misilov V.E. Algorithms for solving inverse geophysical problems on parallel computing systems. Numerical Analysis and Applications. 2013. Vol. 6, no. 2. P. 98–110. DOI: 10.1134/S199542391302002X.

11. Akimova E.N., Misilov V.E., Skurydina A.F., Martyshko M.P. Specialized web portal for solving problems on multiprocessor computing systems. CEUR Workshop Proceedings. 2015. Vol. 1513. P. 123–129.

12. Tsidaev A. .NET Library for Seamless Remote Execution of Supercomputing Software. CEUR-WS. 2017. Vol. 1990. P. 79–83.

13. Kuklin E., Pravdin S. A Web-based System for Launching Large Experiment Series on Supercomputers. CEUR-WS. 2018. Vol. 2281. P. 136–145.

14. Erwin D.W. UNICORE — a grid computing environment. Concurrency and Computation: Practice and Experience. 2002. Vol. 14. P. 1395–1410. DOI: 10.1002/cpe.691.

15. Grosch A., Waldmann M., Göbbert J.H., Lintermann A. A Web-Based Service Portal to Steer Numerical Simulations on High-Performance Computers. 8th European Medical and Biological Engineering Conference. Vol. 80. Springer, 2020. P. 57–65. DOI: 10.1007/978-3-030-64610-3_8.

16. Cruz F.A., Dabin A.J., Dorsch J.P., *et al.* FirecREST: a RESTful API to HPC systems. IEEE/ACM International Workshop on Interoperability of Supercomputing and Cloud Technologies (SuperCompCloud). 2020. P. 21–26. DOI: 10.1109/SuperCompCloud51944.2020.00009.

17. Cholia S., Skinner D., Boverhof J. NEWT: A RESTful service for building High Performance Computing web applications. Gateway Computing Environments Workshop (GCE). 2010. P. 1–11. DOI: 10.1109/GCE.2010.5676125.

18. Schwank O.A., Lustikh E.N. Interpretation of gravitational observations. Theory and practice of solving the direct and inverse problem of gravimetric reconnaissance. Moscow-Leningrad, SRTI of oil and mountain-fuel literature, 1947. 400 p.

19. Introduction to ASP.NET Core Blazor. Microsoft. 2021. URL: `https://docs.microsoft.com/en-us/aspnet/core/blazor` (accessed: 29.10.2021).

20. ASP.NET documentation. Microsoft. 2021. URL: `https://docs.microsoft.com/ru-ru/aspnet/core` (accessed: 29.10.2021).

21. Entity Framework Core. Microsoft. 2021 URL: `https://docs.microsoft.com/ru-ru/ef/core` (accessed: 29.10.2021).

СВЕДЕНИЯ ОБ ИЗДАНИИ

Научный журнал «Вестник ЮУрГУ. Серия «Вычислительная математика и информатика» основан в 2012 году.

Учредитель — Федеральное государственное автономное образовательное учреждение высшего образования «Южно-Уральский государственный университет» (национальный исследовательский университет).

Главный редактор — Л.Б. Соколинский.

Свидетельство о регистрации ПИ ФС77-57377 выдано 24 марта 2014 г. Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций.

Журнал включен в Реферативный журнал и Базы данных ВИНИТИ; индексируется в библиографической базе данных РИНЦ. Журнал размещен в открытом доступе на Всероссийском математическом портале MathNet. Сведения о журнале ежегодно публикуются в международной справочной системе по периодическим и продолжающимся изданиям «Ulrich's Periodicals Directory».

Решением Президиума Высшей аттестационной комиссии Министерства образования и науки Российской Федерации журнал включен в «Перечень рецензируемых научных изданий, в которых должны быть опубликованы основные научные результаты на соискание ученой степени кандидата наук, на соискание ученой степени доктора наук» по научным специальностям и соответствующим им отраслям науки: 05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей (физико-математические науки), 05.13.17 – Теоретические основы информатики (физико-математические науки).

Подписной индекс научного журнала «Вестник ЮУрГУ», серия «Вычислительная математика и информатика»: 10244, каталог «Пресса России». Периодичность выхода — 4 выпуска в год.

Адрес редакции, издателя: 454080, г. Челябинск, проспект Ленина, 76, Издательский центр ЮУрГУ, каб. 32.

ПРАВИЛА ДЛЯ АВТОРОВ

1. Правила подготовки рукописей и пример оформления статей можно загрузить с сайта серии `http://vestnikvmi.susu.ru`. **Статьи, оформленные без соблюдения правил, к рассмотрению не принимаются.**
2. Адрес редакционной коллегии научного журнала «Вестник ЮУрГУ», серия «Вычислительная математика и информатика»:

   Россия 454080, г. Челябинск, пр. им. В.И. Ленина, 76, ЮУрГУ, кафедра СП,

   ответственному секретарю Цымблеру М.Л.
3. Адрес электронной почты редакции: `vestnikvmi@susu.ru`
4. **Плата с авторов за публикацию рукописей не взимается, и гонорары авторам не выплачиваются.**

ВЕСТНИК
ЮЖНО-УРАЛЬСКОГО
ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА
Серия
«ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА»
Том 11, № 1
2022

16+

Техн. редактор *А.В. Миних*

Издательский центр Южно-Уральского государственного университета

Подписано в печать 28.02.2022. Дата выхода в свет 08.04.2022. Формат 60×84 1/8. Печать цифровая.
Усл. печ. л. 9,30. Тираж 500 экз. Заказ 42/91. Цена свободная.

Отпечатано в типографии Издательского центра ЮУрГУ.
454080, г. Челябинск, проспект Ленина, 76.