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).

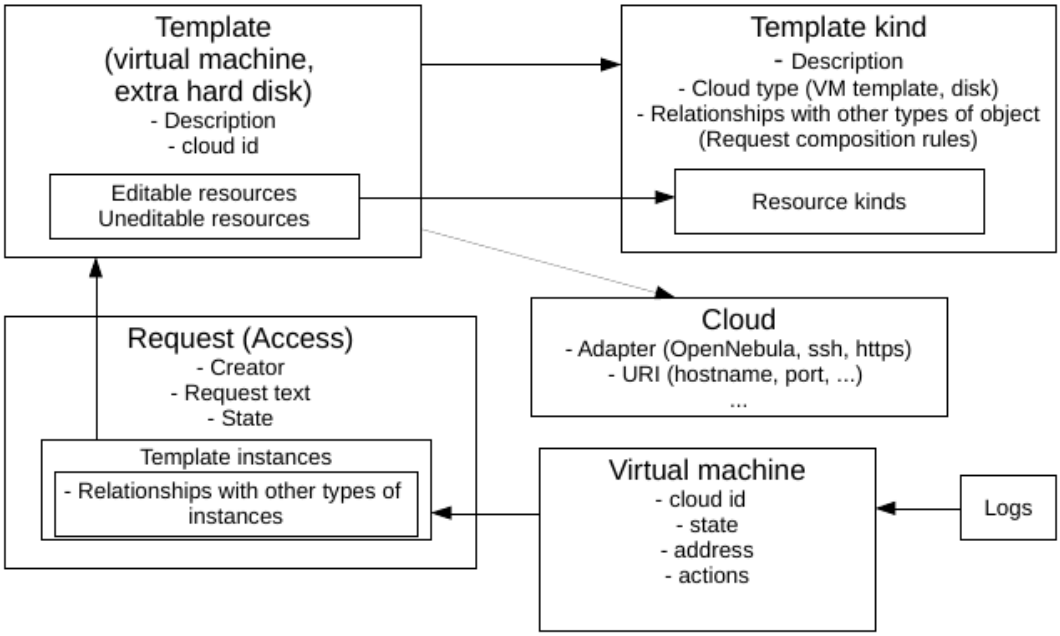

Fig. 1. The diagram of module entities

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

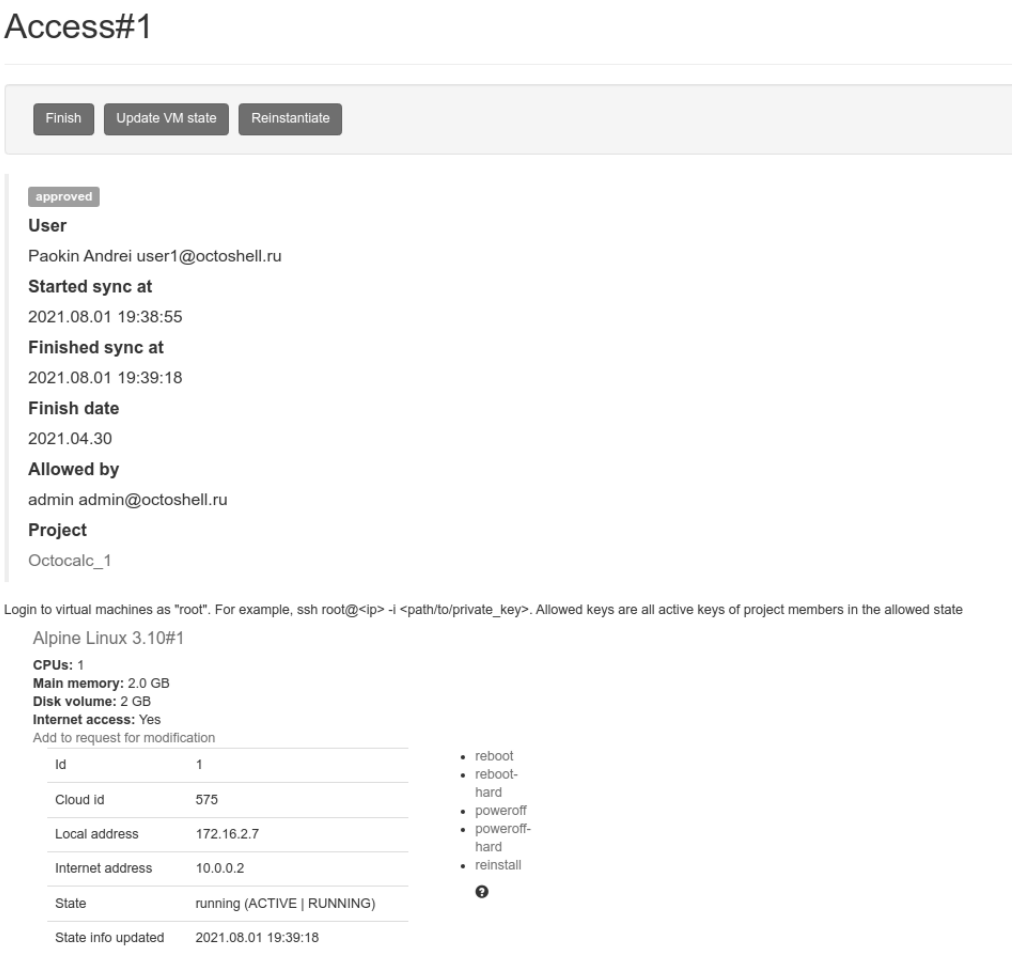

Fig. 2. User access page

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)