УДК 004.75
DOI: 10.25559/SITITO.17.202101.733

Original article

# Data Visualization in Cloud Service for Scientific Computations

**N. A. Balashov*, N. A. Kutovskiy, I. A. Sokolov**
Joint Institute for Nuclear Research, Dubna, Russian Federation
6 Joliot-Curie St., Dubna 141980, Moscow region, Russian Federation
* balashov@jinr.ru

## Abstract
The saas.jinr.ru service is an attempt to simplify the usage of the JINR Multifunctional Information and Computing Complex (MICC) of the Joint Institute for Nuclear Research (JINR). It aims at providing a simple problem-oriented web-interface to help students and beginner researchers in the physics domain to abstract away the complexity of the computing infrastructure and to focus on the actual research. In this paper we show our approach to one of the problems within the scope of the project: interactive data visualization in a web-browser. When approaching this problem, we considered two major requirements to the system: first, users may not have any programming skills, so any interaction should be performed using simple visual components; second, the system must be horizontally scalable to cope with irregular user work-sessions. The paper describes how we used Bokeh and Dask for integrating our data visualization solution within the Django framework to deal with the first requirement, and the JINR cloud for service scaling. Application of cloud technologies facilitates dynamic distribution of workload across virtual machines, thus making it possible for us to control the balance between efficient hardware utilization and end-user experience. Shared in this work resulting software architecture and applied solutions, as well as some performance considerations, can be used as an example when designing other cloud-native scientific applications.

**Keywords:** cloud computing, data visualization, load balancing, virtualization.

**Funding:** This work has been supported by the grants the Russian Science Foundation, RSF 18-71-10095.

*The authors declare no conflict of interest.*

**For citation:** Balashov N.A., Kutovskiy N.A., Sokolov I.A. Data Visualization in Cloud Service for Scientific Computations. *Sovremennye informacionnye tehnologii i IT-obrazovanie* = Modern Information Technologies and IT-Education. 2021; 17(1):109-115. DOI: https://doi.org/10.25559/SITITO.17.202101.733

Оригинальная статья

# Визуализация данных в облачном сервисе для научных расчетов

**Н. А. Балашов*, Н. А. Кутовский, И. А. Соколов**
[1] Международная межправительственная организация Объединенный институт ядерных исследований, г. Дубна, Российская Федерация
141980, Российская Федерация, г. Дубна, Московская область, ул. Жолио-Кюри, д. 6
* balashov@jinr.ru

## Аннотация
Сервис saas.jinr.ru является попыткой упростить использование Многофункционального информационно-вычислительного комплекса Объединенного института ядерных исследований. Сервис saas.jinr.ru - это попытка упростить использование Многофункционального информационно-вычислительного комплекса (МИВК) Объединенного института ядерных исследований (ОИЯИ). Его цель - предоставить простой проблемно-ориентированный веб-интерфейс, который поможет студентам и начинающим исследователям в области физики абстрагироваться от сложности вычислительной инфраструктуры и сосредоточиться на реальных исследованиях. В этой статье мы показываем наш подход к одной из задач в рамках проекта: интерактивная визуализация данных в веб-браузере. Подходя к этой проблеме, мы учли два основных требования к системе: во-первых, пользователи могут не обладать какими-либо навыками программирования, поэтому любое взаимодействие должно происходить с использованием простых визуальных компонентов; во-вторых, система должна быть масштабируемой по горизонтали, чтобы справляться с нерегулярными рабочими сессиями пользователей. В статье описывается, как были использованы Bokeh и Dask для интеграции нашего решения визуализации данных в структуру Django, чтобы удовлетворить первому требованию, и облако ОИЯИ для масштабирования сервиса. Применение облачных технологий позволяет динамически перераспределять нагрузку между виртуальными машинами, что позволяет контролировать баланс между эффективным использованием оборудования и удобством для конечных пользователей. Приведенные в этой работе итоговая архитектура программного обеспечения и прикладные решения, а также некоторые оценки производительности могут быть использованы в качестве примера при разработке других облачных научных приложений.

**Ключевые слова:** облачные вычисления, визуализация данных, балансировка нагрузки, виртуализация.

**Финансирование:** исследование выполнено за счет гранта Российского научного фонда (проект № 18-71-10095).

*Авторы заявляют об отсутствии конфликта интересов.*

**Для цитирования:** Балашов, Н. А. Визуализация данных в облачном сервисе для научных расчетов / Н. А. Балашов, Н. А. Кутовский, И. А. Соколов – DOI 10.25559/SITITO.17.202101.733 // Современные информационные технологии и ИТ-образование. – 2021. – Т. 17, № 1. – С. 109-115.

# 1. Introduction

The Multifunctional Information and Computing Complex (MICC) [1] of the Joint Institute for Nuclear Research is a computing infrastructure that consists of three basic components: the cloud infrastructure [2], the Complex of Information and Computing Resources (CICC) and the HybriLIT heterogeneous platform [3] (which includes "Govorun" supercomputer). It is an example of a complex scientific infrastructure each part of which is designed for solving specific types of computational tasks. This complexity implies that users need to have enough expertise not only in their research domain, but also in using the MICC to be able to choose and use the most suitable components of it for their tasks to be solved efficiently. The lack of expertise in MICC usage may lead to its inefficient utilization and waste of costly resources.

Gaining experience in MICC usage inevitably leads to the need of time-consuming training. While this is suitable for regular MICC users, in some cases users are tight in time limits, e.g. summer students. The saas.jinr.ru project [4], [5], [15], [20]-[23] is being developed for such specific cases: its goal is to give a simple graphical web-interface to quickly access and start using the MICC for running a limited number of predefined applications. In this paper we focus on the data visualization component of the project: its architecture and technologies used.

# 2. Initial assumptions

### 2.1 Software requirements

The project focuses mainly on students in the high-energy physics (HEP) domain who may not have experience with any plotting software yet. Taking into account tight time-limits of our potential users the goal was to give them a basic, simple and self-explanatory plotting tool that would be available via the web-interface of the service. Such a solution would make it possible to place data visualizations into the data analysis workflow built on the saas.jinr.ru service only, without the need to learn other more complex tools.

One of the main decisions made in the saas.jinr.ru project design is that it should be built on free and open-source software (OSS). Since visualization is an important part of any data analysis [6], numerous software solutions and libraries, both commercial and free/OSS, are available to choose from [7]-[9] and emerging. Initially, the only strict technical restriction was the ability of the visualization solution to be integrated into the Django framework (lying in the base of the project).

### 2.2 Workload expectations and requirements

The project mainly focuses on irregular users that are expected to generate rare, but significant spikes of load for relatively short periods of time (up to a few weeks). Typical scientific school (as an example of such a group of users) held by JINR lasts a week and has about 50 participants, but may vary from a few to dozens of participants. Given that the datasets may be large enough to produce significant computational workload during visualization and assuming that all of the users will work simultaneously, it is reasonable to distribute the workload across multiple computational nodes to keep the web-portal responsive. We considered the two possible ways to distribute the workload: place all the workload on the client side or on the multiple backend servers.

The client-side solution introduces additional uncertainty: hardware characteristics of client devices (which may be their personal devices) are unknown in general case and they may not be powerful enough to deal with large datasets. Additionally it requires transferring all the data to the client, even when not all of it is needed for visualization, thus introducing unwanted network channel load. To be able to better control the quality of service we chose the second approach: do most of the compute-intensive operations server-side. We didn't take into account an option to dedicate a single backend server because of the unpredictable future load (that depends on the varying number of simultaneous users and datasets size) which may exceed its capabilities.

Because of expected irregularity of the service usage and varying workload intensity it is not reasonable to dedicate a fixed amount of computational resources permanently. A better solution is to use the cloud technologies to dynamically allocate virtual resources in accordance with the current needs.

### 2.3 Typical data

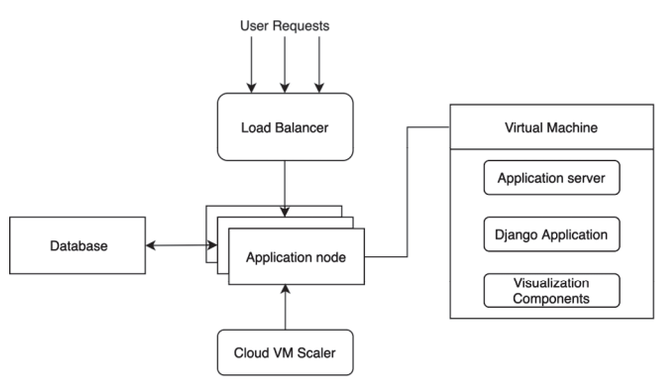
To illustrate the process of choosing the appropriate size of virtual resources we need to describe a sample dataset that we used for this purpose. It contains the charge-time dependence of the system of ten Josephson junctions [10] used to study the resonance phenomena in a model of intrinsic Josephson junctions shunted by LC-elements (L-inductance, C-capacitance). It is a relatively small, but typical dataset that is represented by a plaintext file with 3310331 rows, each of which contains 12 numeric values, the overall size of the data file is 502 MB.

The sample dataset also illustrates why we want the data visualization tools be interactive: data points density of the dataset is high enough to hide the underlying subtle patterns at a small screen of a computer, which can only be discovered by a human eye after zooming in on the plot and investigating different regions of it at a higher scale.

# 3 System Design

### 3.1 Architecture

The part of the whole system architecture involved in data visualization is simple and is depicted on Fig. 1. It consists of a load balancer routing user requests to a number of application nodes, a database node and a cloud scaling service. All the nodes are virtual machines (VM) hosted in the JINR Cloud.




F i g. 1. System architecture scheme

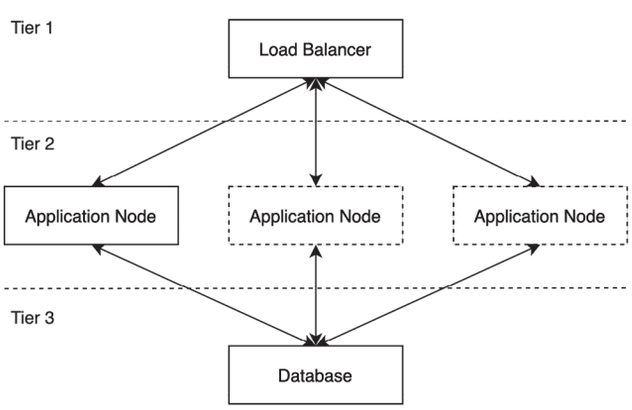
The application nodes are automatically created by the cloud service according to the number of active user requests, thus providing horizontal scalability. All of them are similar to each other and each runs an application server with its own copy of a web-portal served to a limited number of clients.

In the original architecture the database co-located with the web-application and the web-server on the same machine, but due to multiple application nodes running simultaneously in the new architecture the database was moved to a separate machine. The new load balancer component should also run on a dedicated machine because mixing it with the application server (and/or the database node) could make the whole service unresponsive for all of the users during intensive computations within the application server caused by a single user.

By simplifying the application server machines and keeping the same setup in all of them in the form of virtual machines it is possible to make the service horizontally scalable by utilizing the standard features of the JINR Cloud.

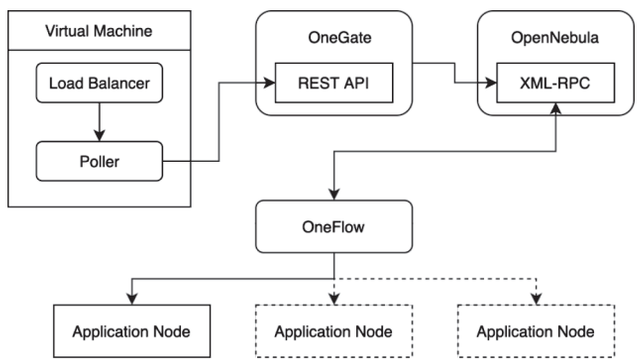
### 3.2 Implementation as a Cloud Service

The JINR Cloud is an implementation of the Infrastructure-as-a-Service model [11] which main goal is VM provisioning. It is based on the OpenNebula platform, two of which components - OneFlow and OneGate - were used to implement horizontal scaling of the service. The OneFlow component allows JINR cloud users to define multi-tiered applications (called "services" in terms of OpenNebula), composed of interconnected cloud VMs with deployment dependencies between them. The OneGate component allows Virtual Machine guests to pull from/push to OpenNebula information associated with a particular VM. We use it to perform periodic checks of the number of active client connections, which is then used by the OneFlow to make decisions on when to scale in and out the application nodes.




F i g. 2. Scheme of the architecture implementation as a OneFlow service. Arrows represent the data flow, the solid-edge application node box shows the deployed machines of a running service, the dotted-edge boxes show the scaling capability of a node role.

The schemes of the implemented OneFlow service for the saas.jinr.ru are shown on Fig. 2 and Fig. 3. The tiers on Fig. 2 show the deployment dependencies of the node roles: higher-tier levels deploy first. When starting the service the database node starts first, then at least one application node and finally the load balancer; the shutdown of the service is performed in reverse order. The logic behind this deployment order is to give users access to the system only when all of its components are ready. It is also a protection measure against the system falling into a split-brain state during the system startup/shutdown, after which it could be hard to restore the system to the normal state. Fig. 3 shows how components of the service communicate.




F i g. 3. OneFlow service components interaction scheme. Arrows show the information flow: the poller periodically pulls the number of active client connections from the Load Balancer, pushes it to the OpenNebula through the OneGate and the OneFlow then uses it for scaling the Application Nodes

### 3.3. Underlying Technologies

Three main components introduced to the system to add data visualization features are the load balancer, application server and the visualization framework. In this section we give an overview of the technologies used to implement them.

*3.3.1 Load Balancer*

We used the OSS version of the Nginx web-server which is also widely used as a high-performance load balancer in many large distributed systems. It plays two roles in the system: load balancing [12] client connections to upstream servers (application nodes) and providing information about the current number of active connections to the OneFlow service to be used for scaling.

The major limitation of the OSS version is that a list of upstream servers can not be changed dynamically (which is a feature of the paid Nginx Plus version). We overcame this limitation by just gracefully restarting the load balancer after changing the list of available upstream servers. In our case it didn't show any significant impact on service performance because of the small number of simultaneous clients, but it could be a serious limitation for large-scale services.

*3.3.2 Visualization Software*

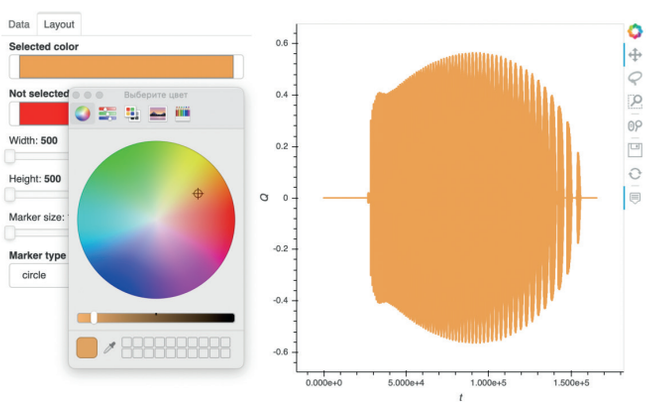
There's a number of feature-rich plotting libraries available for Python and it's really hard to choose the proper one. While studying existing solutions we discovered the HoloViz project which aims at development and support of browser-based data visualization tools and libraries written in python. One of them - hvPlot - provides a high-level plotting API built on HoloViews that implements a general and consistent API for plotting data in different formats. It supports a number of popular data types (such as Pandas and XArray) and plotting APIs (Matplotlib, Bokeh, etc.). We decided to base our development on HoloViz technologies because it provides a set of

well-integrated components with features enough for covering all of our base requirements.

As a plotting backend we used Bokeh as it can be embedded in Django and provides the required basic interactivity features for the plots (zooming, panning, selecting data points and exporting the resulting plot as a picture). Some of these features require asynchronous communications of the client web-browser with the backend server, which is handled by the application server described in the following section.

As a data container we chose Dask DataFrame. Dask DataFrame is a large parallel DataFrame composed of many smaller Pandas DataFrames, split along the index [13], [14], [16]-[19]. This makes it possible to process large datasets, especially those that don't fit in memory, in parts.

The currently implemented interface allows one to choose a file to load the data from, preview a few lines of it to help choose the columns to use for plotting, and tune some layout parameters of the resulting plot: plot size, data point marker parameters (type, size and color) and others. An example of the layout is shown on Fig. 4.




F i g. 4. Example layout of the developed visualization system. The panel on the left allows users to interactively set properties applied to the plot shown on the right.

*3.3.3 Application Server*

Linking Python applications (in our case based on Django framework) with the web-server is traditionally done via the Web Server Gateway Interface (WSGI), which bridges the HTTP requests from the web-server to the backend python application to be processed by it. WSGI is supported by any modern web-server, but it lacks support for asynchronous communication between the client and the application. To provide the necessary interactivity in our application we used the Asynchronous Server Gateway Interface (ASGI), which is a newer interface specification intended to provide asynchronous operations support while providing WSGI backwards-compatibility. To integrate the ASGI in Django we used the Channels package, which makes it possible to process both the HTTP and the WebSocket traffic, support for which is needed for communication with Bokeh. As an application server we used Daphne as the officially supported and recommended ASGI server by the Channels project [24], [25].

### 3.4 Performance Testing

We implemented two plot view modes: standard and rasterized. These modes differ in performance and feature set. The purpose of the rasterized mode is to offload computations from the client when handling datasets that are too large for the client device to keep the user-interface responsive. In standard mode the plot provides a full interactivity feature set, while in rasterized mode it lacks many data layout properties (e.g. marker type, its size, etc) due to server-side rasterization.

The sample dataset defined in Section 2.3 was used to test the performance of both modes. As an application server a single-core VM with 4 GB of RAM was used. Adding more cores and memory to the server VM didn't show any impact on the performance with the sample dataset, so only results for the abovementioned configuration are shown below.

As a client machine the laptop with the following characteristics was used: Intel Core i7-6500U CPU, 8 GB DDR3 RAM. While the laptop CPU has 2 physical cores, 4 logical cores are actually addressed by the operating system (OS) because of the Hyper-Threading technology. To carefully interpret the measurements it is important to understand that 25 % CPU usage on the client equals 1 fully utilized core (out of 4 virtual cores reported by the client OS) and is the same as 100 % CPU usage reported by the server.

To compare the performance of the two modes we measured CPU load of the client and the application server machines when loading the sample dataset defined in Section 2.3. As a server we used a VM with a single-core CPU, and a laptop with a 4-core CPU (2 physical/4 virtual cores) as a client.


(a)


(b)

F i g. 5. CPU load profiles of the client and server during loading visualization of the sample dataset in two modes: standard
(a) and rasterized
(b) in both cases the same machines were used: the single-core VM as the server and the 4-core laptop as the client.

Plots on Fig. 5 show the client and server CPU load profiles when loading the sample dataset in standard (a) and raserized modes (b). These plots show the dramatic difference in overall rendering time: 38 seconds in standard mode against less than 6 seconds in rasterized mode. It can also be clearly seen that in the rasterized mode the client CPU was involved in computation much less time compared to the standard mode.

## 4 Conclusions

In this paper we showed how modern software technologies can be used to build interactive web-applications for scientific and educational usage. It demonstrates how cloud technologies help implement scalable services, which becomes a necessity due to increasing data volumes used in analysis, rapidly developing information technologies and emerging new data analysis tools. Presented architecture, despite its simplicity and targeting a small number of clients, has good scalability potential (although may require further decoupling other components of the system to cope with higher loads). The resulting data visualization system showed decent performance, enough not only for educational purposes, but also suitable for real scientific visual analyses of not very large datasets. In our future studies we plan to investigate the ways to improve performance of the system to make it more suitable for handling larger datasets.

## References

[1] Dolbilov A., Kashunin I., Korenkov V., Kutovskiy N., Mitsyn V., Podgainy D., Stretsova O., Strizh T., Trofimov V., Vorontsov A. Multifunctional Information and Computing Complex of JINR: Status and Perspectives. *CEUR Workshop Proceedings: Proc. of 27th International Symposium NEC-2019 (Budva, Montenegro).* 2019; 2507:16-22. Available at: http://ceur-ws.org/Vol-2507/16-22-paper-3.pdf (accessed 24.02.2021). (In Eng.)

[2] Balashov N.A., Baranov A.V., Kutovskiy N.A., Makhalkin A.N., Mazhitova Ye.M., Pelevanyuk I.S., Semenov R.N. Present Status and Main Directions of the JINR Cloud Development. *CEUR Workshop Proceedings: Proc. of 27th International Symposium NEC-2019 (Budva, Montenegro).* 2019; 2507:185-189. Available at: http://ceur-ws.org/Vol-2507/185-189-paper-32.pdf (accessed 24.02.2021). (In Eng.)

[3] Adam Gh., Bashashin M., Belyakov D., Kirakosyan M., Matveev M., Podgainy D., Sapozhnikova T., Streltsova O., Torosyan Sh., Vala M., Valova L., Vorontsov A., Zaikina T., Zemlyanaya E., Zuev M. IT-ecosystem of the HybriLIT heterogeneous platform for high-performance computing and training of IT-specialists. *CEUR Workshop Proceedings.* 2018; 2267:638-644. Available at: http://ceur-ws.org/Vol-2267/638-644-paper-122.pdf (accessed 24.02.2021). (In Eng.)

[4] Balashov N., Bashashin M., Kuchumov R., Kutovskiy N., Sokolov I. JINR Cloud Service for Scientific and Engineering Computations. *Sovremennye informacionnye tehnologii i IT-obrazovanie* = Modern Information Technologies and IT-Education. 2018; 14(1):61-72. (In Eng.) DOI: https://doi.org/10.25559/SITITO.14.201801.061-072

[5] Balashov N., Kutovskiy N., Priakhina D., Sokolov I. Evolution and Perspectives of the Service for Parallel Applications Running at JINR Multifunctional Information and Computing Complex. *EPJ Web of Conferences.* 2020; 226:03002. (In Eng.) DOI: https://doi.org/10.1051/epjconf/202022603002

[6] Idreos S., Papaemmanouil O., Chaudhuri S. Overview of Data Exploration Techniques. *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD '15).* Association for Computing Machinery, New York, NY, USA; 2015. p. 277-281. (In Eng.) DOI: https://doi.org/10.1145/2723372.2731084

[7] Raghav R.S., Pothula S., Vengattaraman T., Ponnurangam D. A survey of data visualization tools for analyzing large volume of data in big data platform. *2016 International Conference on Communication and Electronics Systems (ICCES).* Coimbatore, India; 2016. p. 1-6. (In Eng.) DOI: https://doi.org/10.1109/CESYS.2016.7889976

[8] Caldarola E.G., Rinaldi A.M. Big Data Visualization Tools: A Survey. *Proceedings of the 6th International Conference on Data Science, Technology and Applications (DATA 2017).* SCITEPRESS – Science and Technology Publications, Lda, Setubal, PRT; 2017. p. 296-305. (In Eng.) DOI: https://doi.org/10.5220/0006484102960305

[9] Qin X., Luo Y., Tang N., Li G. Making data visualization more efficient and effective: a survey. *The VLDB Journal.* 2020; 29(1):93-117. (In Eng.) DOI: https://doi.org/10.1007/s00778-019-00588-3

[10] Shukrinov Yu.M. et al. Modeling of *LC*-shunted intrinsic Josephson junctions in high-$T_c$ superconductors. *Superconductor Science and Technology*. 2016; 30(2):024006. (In Eng.) DOI: https://doi.org/10.1088/1361-6668/30/2/024006

[11] Bokhari M.U., Shallal Q.M., Tamandani Y.K. Cloud computing service models: A comparative study. *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom).* IEEE, New Delhi, India; 2016. p. 890-895. (In Eng.)

[12] Cardellini V., Colajanni M., Yu P.S. Dynamic load balancing on Web-server systems. *IEEE Internet Computing*. 1999; 3(3):28-39. (In Eng.) DOI: https://doi.org/10.1109/4236.769420

[13] Rocklin M. Dask: Parallel Computation with Blocked algorithms and Task Scheduling. In: K. Huff, J. Bergstra (Eds.) *Proceedings of the 14th Python in Science Conference (SciPy 2015).* Austin, Texas; 2015. p. 126-132. (In Eng.) DOI: https://doi.org/10.25080/Majora-7b98e3ed-013

[14] Bird I. Computing for the Large Hadron Collider. *Annual Review of Nuclear and Particle Science*. 2011; 61:99-118. (In Eng.) DOI: https://doi.org/10.1146/annurev-nucl-102010-130059

[15] Baranov A.V., Balashov N.A., Kutovskiy N.A., Semenov R.N. JINR cloud infrastructure evolution. *Physics of Particles and Nuclei Letters*. 2016; 13(5):672-675. (In Eng.) DOI: https://doi.org/10.1134/S1547477116050071

[16] Greenfeld D.R., Greenfeld A.R. Two Scoops of Django: Best Practices for Django 1.8. 3rd ed. Two Scoops Press. Publ.; 2015. (In Eng.)

[17] Thain D., Tannenbaum T., Livny M. Distributed Computing in Practice: The Condor Experience. *Concurrency and Computation: Practice and Experience*. 2005; 17(2-4):323-356. (In Eng.) DOI: https://doi.org/10.1002/cpe.938

[18] Adamson P. et al. First measurement of electron neutrino appearance in NovA. *Physical Review Letters*. 2016; 116(15):151806. (In Eng.) DOI: https://doi.org/10.1103/PhysRevLett.116.151806

[19] Moreno-Vozmediano R., Montero R.S., Llorente I.M. IaaS Cloud Architecture: From Virtualized Datacenters to Federated Cloud Infrastructures. *IEEE Computer*. 2012; 45(12):65-72. (In Eng.) DOI: https://doi.org/10.1109/MC.2012.76

[20] Balashov N.A. et al. JINR Member States cloud infrastructure. *CEUR Workshop Proceedings*. 2017; 2023:202-206. Available at: http://ceur-ws.org/Vol-2023/122-128-paper-19.pdf (accessed 24.02.2021). (In Eng.)

[21] Balashov N.A. et al. Service for parallel applications based on JINR cloud and HybriLIT resources. *EPJ Web of Conferences*. 2019; 214:07012. (In Eng.) DOI: https://doi.org/10.1051/epjconf/201921407012

[22] Balashov N. et al. Creating a Unified Educational Environment for Training IT Specialists of Organizations of the JINR Member States in the Field of Cloud Technologies. In: V. Sukhomlin, E. Zubareva (Eds.) *Modern Information Technology and IT Education. SITITO 2018. Communications in Computer and Information Science*, vol. 1201. Springer, Cham; 2020. p. 149-162. (In Eng.) DOI: https://doi.org/10.1007/978-3-030-46895-8_12

[23] Goncharov P., Ososkov G., Nechaevskiy A., Uzhinskiy A. Architecture and basic principles of the multifunctional platform for plant disease detection. *CEUR Workshop Proceedings*. 2018; 2267:200-206. Available at: http://ceur-ws.org/Vol-2267/200-206-paper-37.pdf (accessed 24.02.2021). (In Eng.)

[24] Weil S.A., Brandt S.A., Miller E.L., Long D.D.E., MaltzahnC. Ceph: a scalable, high-performance distributed file system. *Proceedings of the 7th symposium on Operating systems design and implementation (OSDI'06).* USENIX Association, Berkeley, CA, USA; 2006. p. 307-320. Available at: https://www.ssrc.ucsc.edu/papers/weil-osdi06.pdf (accessed 24.02.2021). (In Eng.)

[25] Massie M., Chun B., Culler D. The Ganglia Distributed Monitoring System: Design, Implementation, and Experience. *Parallel Computing*. 2004; 30:817-840. (In Eng.) DOI: https://doi.org/10.1016/j.parco.2004.04.001

*Submitted 24.02.2021; approved after reviewing 30.03.2021; accepted for publication 02.04.2021.*
*Поступила 24.02.2021; одобрена после рецензирования 30.03.2021; принята к публикации 02.04.2021.*

About the authors:

**Nikita A. Balashov**, Software engineer of the Laboratory of Information Technologies, Joint Institute for Nuclear Research (6 Joliot-Curie St., Dubna 141980, Moscow region, Russian Federation), **ORCID:** http://orcid.org/0000-0002-3646-0522, balashov@jinr.ru
**Nikolay A. Kutovskiy**, Senior Researcher of the Laboratory of Information Technologies, Joint Institute for Nuclear Research (6 Joliot-Curie St., Dubna 141980, Moscow region, Russian Federation), Ph.D. (Phys.-Math.), **ORCID:** http://orcid.org/0000-0002-2920-8775, kut@jinr.ru
**Ivan A. Sokolov**, Software engineer of the Laboratory of Information Technologies, Joint Institute for Nuclear Research (6 Joliot-Curie St., Dubna 141980, Moscow region, Russian Federation), **ORCID:** http://orcid.org/0000-0003-0295-5372, sokolov@jinr.ru

*All authors have read and approved the final manuscript.*

Об авторах:

**Балашов Никита Александрович**, инженер-программист Лаборатории информационных технологий, Международная межправительственная организация Объединенный институт ядерных исследований (141980, Российская Федерация, Московская область, г. Дубна, ул. Жолио-Кюри, д. 6), **ORCID:** http://orcid.org/0000-0002-3646-0522, balashov@jinr.ru
**Кутовский Николай Александрович**, старший научный сотрудник Лаборатории информационных технологий, Международная межправительственная организация Объединенный институт ядерных исследований (141980, Российская Федерация, Московская область, г. Дубна, ул. Жолио-Кюри, д. 6), кандидат физико-математических наук, **ORCID:** http://orcid.org/0000-0002-2920-8775, kut@jinr.ru
**Соколов Иван Александрович**, инженер-программист Лаборатории информационных технологий, Международная межправительственная организация Объединенный институт ядерных исследований (141980, Российская Федерация, Московская область, г. Дубна, ул. Жолио-Кюри, д. 6), **ORCID:** http://orcid.org/0000-0003-0295-5372, sokolov@jinr.ru

*Все авторы прочитали и одобрили окончательный вариант рукописи.*