УДК 004.45
DOI: 10.25559/SITITO.18.202201.176-182

Original article

Unified Software Development and Analysis Environment for MPD Experiment at NICA Collider

J. Buša Jr.[a,b], S. Hnatič[a*], V. V. Korenkov[a], O. V. Rogachevsky[a], M. Vaľa[c], J. Vrláková[c]
[a] Joint Institute for Nuclear Research, Dubna, Russian Federation
Address: 6 Joliot-Curie St., Dubna 141980, Moscow region, Russian Federation
[*] hnatics@jinr.ru
[b] Institute of Experimental Physics, Slovak Academy of Sciences, Košice, Slovak Republic
Address: 47 Watsonova St., Košice 040 01, Slovak Republic
[c] Pavol Jozef Šafárik University in Košice, Košice, Slovak Republic
Address: 2 Šrobárova St., Košice 041 80, Slovak Republic

Abstract
MPDRoot is an off-line software framework for simulation, reconstruction, and physical analyses of the simulated or experimental data for MPD experiment at NICA collider. The experiment is projected to run for few decades and to obtain $\sim 10^8$ events of heavy ion collisions for physics analysis. Hence provided software must be sufficiently flexible, resilient, robust to be used, developed, and maintainable for the full lifetime of the experiment and the analysis of its data. In this paper, we describe the effective and efficient implementation of build automatization, configuration, and installation of the software (DevOps) for the development and use of the MPDRoot, playing crucial role for the success of the whole MPD project in the future. Compared to previously existing state, the major improvement requirements were to reduce the complexity and to increase the universality of these deployment related actions for various system and hardware configurations. We show how this was achieved by the use of containers to deploy unified development and user environment with CernVM-FS service to dynamically load built modularized software from CernVM-FS server located in existing JINR infrastructure. Typical MPDRoot DevOps operations, being before heavy, cumbersome, and time consuming, are now scaled down to running few commands accompanied by short deployment guide, significantly reducing possibility of errors on end-user and developer side. Above all, the main benefit of the current implementation is its wide compatibility and full modularization making it easy to maintain, upgrade, and to identify the source of potential issues in the future.

Keywords: NICA, MPD, MPDRoot, DevOps, Cern VM-FS, docker, aliBuild

Acknowledgements: Computations were held on the basis of the HybriLIT heterogeneous computing platform (LIT, JINR) and NICA LHEP offline computing cluster (LHEP, JINR).

The authors declare no conflict of interest.

For citation: Buša J.Jr., Hnatič S., Korenkov V.V., Rogachevsky O.V., Vaľa M., Vrláková J. Unified Software Development and Analysis Environment for MPD Experiment at NICA Collider. *Sovremennye informacionnye tehnologii i IT-obrazovanie = Modern Information Technologies and IT-Education.* 2022; 18(1):176-182. doi: https://doi.org/10.25559/SITITO.18.202201.176-182

Научная статья

Единая среда разработки программного обеспечения и анализа для МПД эксперимента на НИКА коллайдере

Я. Буша мл.[1,2], С. Гнатич[1*], В. В. Кореньков[1], О. В. Рогачевский[1], М. Валя[3], Я. Врлакова[3]

[1] Международная межправительственная организация Объединенный институт ядерных исследований, г. Дубна, Российская Федерация

Адрес: 141980, Российская Федерация, г. Дубна, Московская область, ул. Жолио-Кюри, д. 6

[*] hnatics@jinr.ru

[2] Институт экспериментальной физики Словацкой академии наук, г. Кошице, Словацкая Республика

Адрес: 040 01, Словацкая Республика, г. Кошице, ул. Ватсонова, д. 47

[3] Университет Павла Йозефа Шафарика, г. Кошице, Словацкая Республика

Адрес: 041 80, Словацкая Республика, г. Кошице, ул. Шробарова, д. 2

Аннотация

MPDRoot является оффлайн программной средой для моделирования, реконструкции и физического анализа смоделированных или экспериментальных данных для эксперимента MPD на коллайдере NICA. Предполагается, что эксперимент продлится несколько десятилетий и позволит получить $\sim 10^8$ событий столкновений тяжелых ионов для физического анализа. Следовательно, предоставляемое программное обеспечение должно быть достаточно гибким, отказоустойчивым, надежным, чтобы его можно было использовать, разрабатывать и поддерживать в течение всего времени проведения эксперимента и анализа его данных. В этой статье мы описываем эффективную реализацию автоматизации сборки, настройки и установки программного обеспечения (DevOps) для разработки и использования MPDRoot, играющего решающую роль в успехе всего проекта MPD в будущем. По сравнению с ранее существовавшим состоянием основные требования к улучшению заключались в снижении сложности и повышении универсальности действий, связанных с развертыванием на различных системных и аппаратных конфигурациях. Это было достигнуто за счет использования контейнеров для развертывания единой среды для разработчиков и пользователей с сервисом CernVM-FS для динамической загрузки модульного программного обеспечения с сервера, расположенного в существующей инфраструктуре ОИЯИ. Типичные операции MPDRoot devOps, которые ранее были громоздкими и трудоемкими, теперь сокращены до выполнения нескольких команд, сопровождаемых кратким руководством по развертыванию, что значительно снижает вероятность ошибок на стороне конечного пользователя и разработчика. Основным преимуществом текущей реализации является ее широкая совместимость и полная модульность, упрощающая обслуживание, обновление и выявление источника потенциальных проблем в будущем.

Ключевые слова: НИКА, МПД, ПО для МПД, CernVM-FS, докер, алибилд

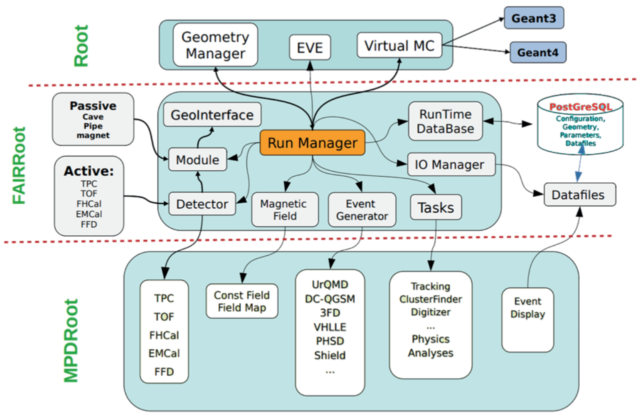
Благодарности: Расчеты проводились на базе гетерогенной вычислительной платформы HybriLIT (ЛИТ, ОИЯИ) и автономного вычислительного кластера NICA ЛФВЭ (ЛФВЭ, ОИЯИ).

Авторы заявляют об отсутствии конфликта интересов.

Для цитирования: Единая среда разработки программного обеспечения и анализа для МПД эксперимента на НИКА коллайдере / Я. Буша мл. [и др.] // Современные информационные технологии и ИТ-образование. 2022. Т. 18, № 1. С. 176-182. doi: https://doi.org/10.25559/SITITO.18.202201.176-182

1 Introduction

The MPDRoot [1] is a framework for simulation, reconstruction and data analysis of the MPD experiment at NICA [2; 3]. It is built on top of ROOT [4; 5] and FairRoot [6; 7] frameworks having complex multi-layered structure, which contains



F i g. 1. Architecture of MPDRoot

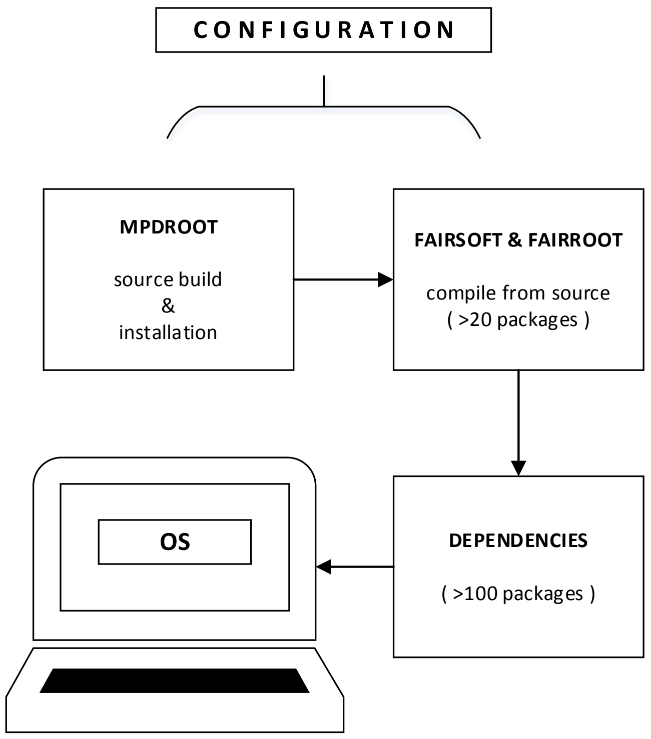
necessary tools to simulate, run, and analyze physics experiment in general, such as virtual Monte Carlo interface [8] implemented in Geant, interfaces to different event generators, interfaces to detectors and their geometry, simulation and analysis tasks, event display, and other software entities. These are in general implemented as interfaces, which at the core are handled by a Run Manager (Fig. 1).

Due to this heavy complexity, the MPDRoot has large multi-level dependency base. Therefore the scale of related DevOps operations has significant impact on the overall productivity of users and developers. Of crucial importance are the implementations of processes of automatization, build, and installation of the software. This work describes the redesign of these processes and its implementation, making them simple, compact, and unified, substantially reducing associated support and maintenance costs and thereby improving overall productivity.

2 Previous state disadvantages

The process of MPDRoot deployment on the local machines was, up to minor details, identical for users and developers, performed in the following sequence (Fig. 2):

1. Installing FairSoft dependencies together with MPDRoot dependencies for the target linux distribution
2. Cloning FairSoft from repository, patching, building from sources, configuring related variables
3. Cloning FairRoot from repository, building from source, configuring related variables
4. Cloning MPDRoot dev version from repository, building from source, configuring, installing



F i g. 2. Schematic representation of legacy method of MPDRoot software delivery to the local machines

This approach had significant disadvantages:

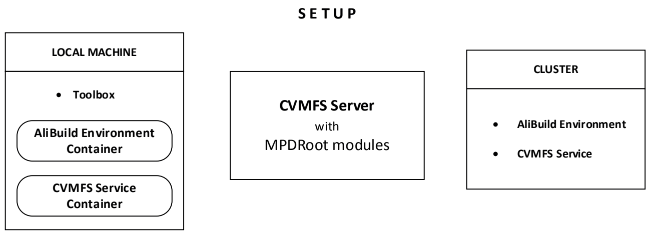
- Installation of more than 100 package dependencies varying in versions for different Linux distributions being the source of potential compatibility problems.
- Whole procedure was heavy, involving many steps by typing multiple commands, configuring system variables, increasing the probability of something going wrong. If the error was made, usually the whole procedure had to be repeated from scratch.
- Source build taking many hours for each installation.
- The step-by-step installation procedure varied between different Linux distributions. Support and maintenance costs grew considerably with each added OS and OS version. It was necessary to support Linux versions that passed their EOL since some users were not willing to upgrade their base OS.
- Required technical skill level for software deployment was the same for user and developer. This in combination with difficulty of overall procedure was adding additional workload on developers who had to supervise user installations.
- This caused lack of support staff resulting in poor functionality on the cluster with outdated software versions or software on cluster not working at all.
- Development CI pipeline not guaranteeing 100% compatibility of new commits on all used OS environments, possibly introducing new compatibility related defects.
- Complexity of software deployment causing migration of the procedure to the updated versions of FairRoot and FairSoft requiring months of effort.

In terms of project effort cost [9] these disadvantages were so severe[1], that we decided it was necessary to rework the whole build and deployment system from scratch.

3 Architecture of the new deployment

To address the issues from previous section we were guided by the following considerations:

- For multi-compatible MPDRoot software delivery the environment needed to build all of MPDRoot's dependencies must have same versions of packages and must be separated from the infrastructure [10]. This can be achieved by having this environment encapsulated in its own OS in a docker [11]. The docker image is then used in an OS-layer, a container [12; 13].
- For convenience, the choice of docker OS is the same as already present in the existing JINR's cluster infrastructure [14].
- The MPDRoot's dependencies, different for its release and development versions, are then built in this environment as modules and stored on the external server for distribution along with MPDRoot's software releases.
- Currently, the best tool tailored for physics experiments to build such modularized environment is aliBuild used for ALICE software in CERN [15]. It can be used to build modular dependencies and software itself.
- The service chosen for software distribution is CernVM-FS [16; 17] already present on JINR's clusters. For the simplicity of local deployment, it is convenient to encapsulate this service in docker/container.
- CernVM-FS server to host MPDRoot's release and development environment modules.



F i g. 3. Architecture for fast delivery of MPDRoot in the unified environment

Fig. 3 shows the working setup with clusters connected to already existing CernVM-FS server. Toolbox package is the only dependency required to deliver MPDRoot to the local machine.

The clusters used to run MPDRoot are NICA LHEP cluster[2] and HybriLIT [18; 19] , which is a part of Multifunctional Information and Computing Complex of JINR [20].

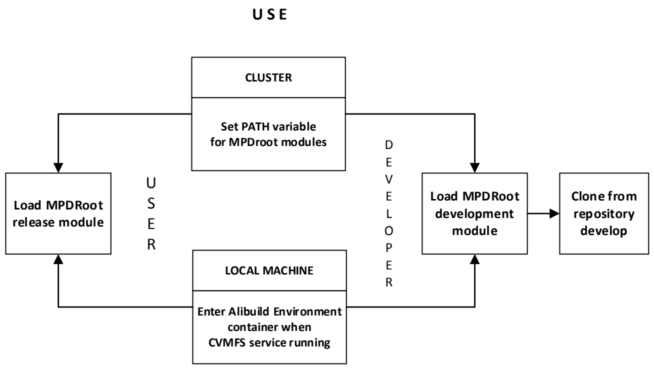
The idea of having packages deployed as modules, allowing multiple module versions to coexist, making the system portable, reproducible, and avoiding so-called "dependency hell" is not new. It is implemented by purely functional software deployment model, where software modules are installed into unique directories generated by crypto hashes, while all module dependencies are stored in the hash. For example, it is used for over a decade in the Nix package manager, on which the NixOS Linux distribution is based [21].

In the core of our modular package architecture is the similar concept used in nicadist project[3] – a partial fork of alidist project[4], based on alibuild. We created it to implement the modular MPDRoot build system. The main purpose of nicadist is to build software packages from source and keep track of their dependencies. If package version changes, or some build parameters change, then all its dependent packages are rebuilt.

The repository consists of mainly 3 types of files. Recipes, which are rules, parameters and configuration to build certain software package. Defaults file, containing rules, parameters and configuration on how to build larger project with desired package versions. Scripts needed to build packages, which are then placed onto CernVM-FS server. In this way, new dependencies can be added simply by writing a recipe, multiple combinations of software versions can be built and distributed, ensuring their proper separation. The speed of software distribution is ensured by aggressive caching and reduction of latency implemented in CernVM-FS [22].

4 Usability improvements, benefits

The installation of the unified environment on a local machine is now automatic with script and compatible with any mainstream RedHat and Debian based distributions. This is substantial simplification compared to the previous method shown in Fig. 2. The local installation process takes few minutes instead of hours. MPDRoot's use on the cluster is now "ready-to-go" and involves typing two commands.


USE
CLUSTER
Set PATH variable for MPDroot modules
Load MPDRoot release module
USER
DEVELOPER
Load MPDRoot development module
Clone from repository develop
LOCAL MACHINE
Enter Alibuild Environment container when CVMFS service running

F i g. 4. Schematic representation of the implemented MPDRoot state-of-the-art software delivery using unified environment for users and developers

Fig. 4 illustrates current use of MPDRoot. One enters the unified environment by setting the module path when working on the cluster or entering the environment container via toolbox when working locally. Then the desired main module is loaded, which automati-

[1] CHAOS Report 2015. The Standish Group International, Inc.; 2015. 13 p. Available at: https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf (accessed 24.01.2022). (In Eng.)

[2] The NICA LHEP offline computing cluster [Electronic resource]. WebNCX; 2022. Available at: https://webncx.jinr.ru (accessed 24.01.2022). (In Eng.)

[3] Buša J.Jr. NICA [Electronic resource]. GitLab; 2022. Available at: https://git.jinr.ru/nica/nicadist (accessed 24.01.2022). (In Eng.)

[4] von Haller B. ALICE Software [Electronic resource]. GitHub, Inc.; 2022. Available at: https://github.com/alisw/alidist (accessed 24.01.2022). (In Eng.)

cally loads all module dependencies. Users load MPDRoot release module, all paths are already pre-configured and there is no additional configuration needed. Developers load MPDRoot development module, then clone and build latest development version from the git repository, export the installation path variable, source the configuration script and are all set.

The possibility to build full local environment, identical to remote build, without CernVM-FS service is also present. This option is very time consuming and not recommended for users and developers of MPDRoot. It is mainly for the purposes of development of the nicadist project.

The MPDRoot's release versions, as well as basic OS images, are now separated and tagged, having unique build configuration and metadata, which is necessary for the exact reproduction of analysis results and quick bug tracing. This allows to build even after many years exactly the same combination of packages and redo the calculations if necessary.

The illustrative example of benefit of new unified build system is the implementation of automatic code formatting using the clang-format package. Proper code formatting style is essential feature of a well-designed code[5] [23; 24], necessary for MPDRoot's optimal performance [25]. Clang-format, although widely used, has well-known problem of no guaranteed auto-formatted code output compatibility between different package releases. Various Linux distributions have different clang-format package versions making implementation of automatic code formatting at the project level overly complex. The output from different clang-format releases throws errors in CI checks. The commonly used workaround is to support one specific release of clang-format for bulk of Linux distributions, for example as attempted by CBM team by releasing their own clang-format package version[6]. Such heavy solution has its own compatibility issues and adds considerable support and maintenance burden. However, in the MPDRoot build implementation presented in this work the problem does not exist by definition. The environment is unified, all users and developers load the same built clang-format release version module located on the CernVM-FS server.

As this implementation is encapsulated using dockers/containers separating it from the infrastructure, it is expected to work out of the box for new releases of Linux distributions. To prove this, we tested it to work on unreleased Ubuntu Linux 22.04 daily build as of 17.2.2022 - one week from its feature freeze stage and two months from its projected release date.

5 Conclusions and Future Perspective

In this paper we showed how we implemented the new modern build system to develop and use MPDRoot software for MPD experiment at NICA collider. It is accompanied by the interactive how-to on our website[7] implemented in asciinema terminal emulator[8], which is user-friendly and easy to follow. Apart from initial user complaints mainly related to transition to a completely new deployment system, most of the feedback is overwhelmingly positive. The main issues this new implementation solves are:

- all users and developers are now working with the same package versions of dependencies built in the same unified environment;
- quick, user-friendly installation;
- no build and configuration required for regular (non-developer) users at all;
- nothing but the build of the fresh development branch from git repository is required for developers with much less configuration than previously;
- lower probability of installation errors, less time spent on support;
- easy updates, less maintenance required;
- new commits guaranteed to work on other machines after passing the CI pipeline, eliminates the need for multi-OS testing.

In the future we plan to distribute MPDRoot software for use on local machines by docker releases, furthermore simplifying the installation process for users and reducing efforts needed for support and maintenance.

References

[1] Rogachevsky O.V., Bychkov A.V., Krylov A.V., et al. Software Development and Computing for the MPD Experiment. *Physics of Particles and Nuclei*. 2021; 52(4):817-820. (In Eng.) doi: https://doi.org/10.1134/S106779621040523

[2] Golovatyuk V., Kekelidze V., Kolesnikov V., Rogachevsky O., Sorin A. Multi-Purpose Detector to study heavy-ion collisions at the NICA collider. *Nuclear Physics A*. 2019; 982:963-966. (In Eng.) doi: https://doi.org/10.1016/j.nuclphysa.2018.10.082

[3] Kekelidze V., Kolesnikov V., Matveev V., Sorin A. Status and Prospects at NICA. The 18th International Conference on Strangeness in Quark Matter (SQM 2019). *Springer Proceedings in Physics.* 2019; 250:503-508. (In Eng.) doi: https://doi.org/10.1007/978-3-030-53448-6_79

[4] Brun R., Rademakers F. ROOT – An Object Oriented Data Analysis Framework. *Nucl. Inst. & Meth. in Phys. Res. A.* 1997; 389:81-86. (In Eng.) doi: https://doi.org/10.1016/S0168-9002(97)00048-X

[5] Bourque P., Fairley R.E., eds. SWEBOK 3.0: Guide to the Software Engineering Body of Knowledge [Electronic resource]. IEEE Computer Society Press, Los Alamitos, CA, USA; 2014. Available at: https://cs.fit.edu/~kgallagher/Schtick/Serious/SWEBOKv3.pdf (accessed 24.01.2022). (In Eng.)

[6] Clang-format [Electronic resource]. CBM Redmine; 2022. Available at: https://redmine.cbm.gsi.de/projects/cbmroot/wiki/Clang-format (accessed 24.01.2022). (In Eng.)

[7] Running MPDRoot locally using CVMFS [Electronic resource]. MPDRoot; 2022. Available at: http://mpdroot.jinr.ru/running-mpdroot-on-local-machine-using-cvmfs (accessed 24.01.2022). (In Eng.)

[8] Asciinema [Electronic resource]. Available at: https://asciinema.org (accessed 24.01.2022). (In Eng.)

[5] Antcheva I., et. al. ROOT – A C++ framework for petabyte data storage, statistical analysis and visualization. *Computer Physics Communications*. 2011; 180(12):2499-2512. (In Eng.) doi: https://doi.org/10.1016/j.cpc.2009.08.005

[6] Al-Turany M., Uhlig F. FairRoot Framework. *12th International Workshop on Advanced Computing and Analysis Techniques in Physics Research. PoS (ACAT08)*. Vol. 70. Erice, Italy; 2009. p. 048. (In Eng.) doi: https://doi.org/10.22323/1.070.0048

[7] Al-Turany M., Bertini D., Karabowicz R., Kresan D., Malzacher P., Stockmanns T., Uhlig F. The FairRoot Framework. *Journal of Physics: Conference Series.* 2012; 396:022001. (In Eng.) doi: https://doi.org/10.1088/1742-6596/396/2/022001

[8] Hrivnacova I., Adamova D., Berejnoi V., Brun R., Carminati F., Fasso A., Futo E., Gheata A., Gonzalez Caballero I., Morsch A. The Virtual Monte Carlo. *Proceedings of the 13th International Conference on Computing in High-Enery and Nuclear Physics (CHEP 2003). eConf.* 2003; C0303241 (2003) THJT006. La Jolla, California. (In Eng.) doi: https://doi.org/10.48550/arxiv.cs/0306005

[9] Jorgensen M., Molokken-Ostvold K. How large are software cost overruns? A review of the 1994 CHAOS report. *Information and Software Technology.* 2006; 48(4):297-301. (In Eng.) doi: https://doi.org/10.1016/j.infsof.2005.07.002

[10] Dijkstra E.W. On the Role of Scientific Thought. In: *Selected Writings on Computing: A personal Perspective. Texts and Monographs in Computer Science.* Springer, New York, NY; 1982. p. 60-66. (In Eng.) doi: https://doi.org/10.1007/978-1-4612-5695-3_12

[11] Boettinger C. An introduction to Docker for reproducible research, with examples from the R environment. *ACM SIGOPS Operating Systems Review, Special Issue on Repeatability and Sharing of Experimental Artifacts.* 2015; 49(1):71-79. (In Eng.) doi: https://doi.org/10.1145/2723872.2723882

[12] Abraham S., Paul A., Khan R., Butt. A. On the Use of Containers in High Performance Computing Environments. *IEEE 13th International Conference on Cloud Computing (CLOUD)*. IEEE Press, Beijing, China; 2020. p. 284-293. (In Eng.) doi: https://doi.org/10.1109/CLOUD49709.2020.00048

[13] Gantikow H., Walter S., Reich C. Rootless Containers with Podman for HPC. In: Jagode H., Anzt H., Juckeland G., Ltaief H. (eds.). *High Performance Computing. ISC High Performance 2020. Lecture Notes in Computer Science.* Vol. 12321. Springer, Cham; 2020. p. 343-354. (In Eng.) doi: https://doi.org/10.1007/978-3-030-59851-8_23

[14] Korenkov V., Dolbilov A., Mitsyn V., Kashunin I., Kutovskiy N., Podgainy D., Streltsova O., Strizh T., Trofimov V., Zrelov P. The JINR distributed computing environment. *EPJ Web of Conferences*. 2019; 214:03009. (In Eng.) doi: https://doi.org/10.1051/epjconf/201921403009

[15] Berzano D., Krzewicki M., The ALICE Software Release Validation cluster. *Journal of Physics: Conference Series*. 2015; 664(2):022006. (In Eng.) doi: https://doi.org/10.1088/1742-6596/664/2/022006

[16] Blomer J., et. al. Distributing LHC application software and condition databases using the CernVM file system. *Journal of Physics: Conference Series*. 2011; 331(4):042003. (In Eng.) doi: https://doi.org/10.1088/1742-6596/331/4/042003

[17] Blomer J., Buncic P., Fuhrmann T. CernVM-FS: Delivering scientific software to globally distributed computing resources. *Proceedings of the first international workshop on Network-aware data management (NDM '11).* ACM, New York, NY, USA; 2011. p. 49-56. (In Eng.) doi: https://doi.org/10.1145/2110217.2110225

[18] 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. ITecosystem of the HybriLIT heterogeneous platform for highperformance computing and training of ITspecialists. *CEUR Workshop Proceedings.* 2018; 2267:638-644. Available at: http://ceur-ws.org/Vol-2267/638-644-paper-122.pdf (accessed 24.01.2022). (In Eng.)

[19] Korenkov V.V., Podgainy D.V., Streltsova O.I. Educational Program on HPC Technologies Based on the Heterogeneous Cluster HybriLIT (LIT JINR). *Sovremennye informacionnye tehnologii i IT-obrazovanie* = Modern Information Technologies and IT-Education. 2017; 13(4):141-146. (In Russ., abstract in Eng.) doi: https://doi.org/10.25559/SITITO.2017.4.633

[20] Korenkov V. The JINR Multifunctional Information and Computing Complex. *2020 International Scientific and Technical Conference Modern Computer Network Technologies (MoNeTeC).* IEEE Press, Moscow, Russia; 2020. p. 1-4. (In Eng.) doi: https://doi.org/10.1109/MoNeTeC49726.2020.9258311

[21] Dolstra E., Löh A. NixOS: a purely functional Linux distribution. *ACM SIGPLAN Notices*. 2008; 43(9):367-378. (In Eng.) doi: https://doi.org/10.1145/1411203.1411255

[22] Blomer J., Fuhrmann T. A Fully Decentralized File System Cache for the CernVM-FS. *2010 Proceedings of 19th International Conference on Computer Communications and Networks.* IEEE Press, Zurich, Switzerland; 2010. p. 1-6. (In Eng.) doi: https://doi.org/10.1109/ICCCN.2010.5560054

[23] Dolbilov A., Kashunin I., Korenkov V., Kutovskiy N., et al. Multifunctional Information and Computing Complex of JINR: Status and Perspectives. *CEUR Workshop Proceedings*. 2019; 2507:16-22. Available at: http://ceur-ws.org/Vol-2507/16-22-paper-3.pdf (accessed 24.01.2022). (In Eng.)

[24] Hunt A., Thomas D. The Art of Enbugging. *IEEE Software.* 2003; 20(1):10-11. (In Eng.) doi: https://doi.org/10.1109/MS.2003.1159022

[25] Buša J.Jr., Hnatič S., Rogachevsky O.V. Performance Analysis and Optimization of MPDRoot. *CEUR Workshop Proceedings.* 2021; 3041:75-79. Available at: http://ceur-ws.org/Vol-3041/75-79-paper-13.pdf (accessed 24.01.2022). (In Eng.)

Submitted 24.01.2022; approved after reviewing 28.02.2022; accepted for publication 02.03.2022.
Поступила 24.01.2022; одобрена после рецензирования 28.02.2022; принята к публикации 02.03.2022.

About the authors:

Ján Buša Jr., Senior Research Scientist of the Meshcheryakov Laboratory of Information Technologies, Joint Institute for Nuclear Research (6 Joliot-Curie St., Dubna 141980, Moscow region, Russian Federation); Researcher of the Institute of Experimental Physics, Slovak Academy of Sciences (47 Watsonova St., Košice 040 01, Slovak Republic), PhD in Mathematics, **ORCID: https://orcid.org/0000-0003-3807-0373,** busa@jinr.ru

Slavomír Hnatič, Lead Scientist of the Meshcheryakov Laboratory of Information Technologies, Joint Institute for Nuclear Research (6 Joliot-Curie St., Dubna 141980, Moscow region, Russian Federation), Dr. rer. nat., **ORCID: https://orcid.org/0000-0002-1674-7403,** hnatics@jinr.ru

Vladimir V. Korenkov, Director of the Meshcheryakov Laboratory of Information Technologies, Joint Institute for Nuclear Research (6 Joliot-Curie St., Dubna 141980, Moscow region, Russian Federation), Dr. Sci. (Tech.), Professor, **ORCID: https://orcid.org/0000-0002-2342-7862,** korenkov@jinr.ru

Oleg V. Rogachevsky, Head of Department of the Meshcheryakov Laboratory of Information Technologies, Joint Institute for Nuclear Research (6 Joliot-Curie St., Dubna 141980, Moscow region, Russian Federation), Cand. Sci. (Phys.-Math.), **ORCID: https://orcid.org/0000-0002-1539-8096,** rogachevsky@jinr.ru

Martin Vaľa, Senior Research Scientist, Pavol Jozef Šafárik University in Košice (2 Šrobárova St., Košice 041 80, Slovak Republic), PhD in Mathematics, **ORCID: https://orcid.org/0000-0003-1965-0516,** martin.vala@upjs.sk

Janka Vrláková, Head of Department, Pavol Jozef Šafárik University in Košice (2 Šrobárova St., Košice 041 80, Slovak Republic), PhD in Mathematics, Associate Professor, **ORCID: https://orcid.org/0000-0002-5846-8496,** janka.vrlakova@upjs.sk

All authors have read and approved the final manuscript.

Об авторах:

Буша Ян мл., старший научный сотрудник Лаборатории информационных технологий имени М.Г. Мещерякова, Международная межправительственная организация Объединенный институт ядерных исследований (141980, Российская Федерация, г. Дубна, Московская область, ул. Жолио-Кюри, д. 6); исследователь, Институт экспериментальной физики Словацкой академии наук (040 01, Словацкая Республика, г. Кошице, ул. Ватсонова, д. 47), кандидат физико-математических наук, **ORCID: https://orcid.org/0000-0003-3807-0373,** busa@jinr.ru

Гнатич Славомир, ведущий научный сотрудник Лаборатории информационных технологий имени М.Г. Мещерякова, Международная межправительственная организация Объединенный институт ядерных исследований (141980, Российская Федерация, г. Дубна, Московская область, ул. Жолио-Кюри, д. 6), кандидат физико-математических наук, **ORCID: https://orcid.org/0000-0002-1674-7403,** hnatics@jinr.ru

Кореньков Владимир Васильевич, директор Лаборатории информационных технологий имени М.Г. Мещерякова, Международная межправительственная организация Объединенный институт ядерных исследований (141980, Российская Федерация, г. Дубна, Московская область, ул. Жолио-Кюри, д. 6), доктор технических наук, профессор, **ORCID: https://orcid.org/0000-0002-2342-7862,** korenkov@jinr.ru

Рогачевский Олег Васильевич, начальник сектора Лаборатории информационных технологий имени М.Г. Мещерякова, Международная межправительственная организация Объединенный институт ядерных исследований (141980, Российская Федерация, г. Дубна, Московская область, ул. Жолио-Кюри, д. 6), кандидат физико-математических наук, **ORCID: https://orcid.org/0000-0002-1539-8096,** rogachevsky@jinr.ru

Валя Мартин, старший научный сотрудник, Университет Павла Йозефа Шафарика (041 80, Словацкая Республика, г. Кошице, ул. Шробарова, д. 2), кандидат физико-математических наук, **ORCID: https://orcid.org/0000-0003-1965-0516,** martin.vala@upjs.sk

Врлакова Янка, заведующий кафедрой, Университет Павла Йозефа Шафарика (041 80, Словацкая Республика, г. Кошице, ул. Шробарова, д. 2), кандидат физико-математических наук, доцент, **ORCID: https://orcid.org/0000-0002-5846-8496,** janka.vrlakova@upjs.sk

Все авторы прочитали и одобрили окончательный вариант рукописи.