JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

HPC-cloud native concurrent simulation of Automotive workflows

[1]Radhika Gundavelli
[1]Engineering Manager
[1]Mechanical Engineering Department,
[1]University of Nevada, Las Vegas, USA

Abstract : In this study, we explore a novel approach that combines high-performance computing (HPC) with cloud-native frameworks to improve the efficiency of Automotive simulation based workflows. We propose a hybrid cloud architecture that leverages both HPC clusters and cloud-native frameworks.The simulation component runs on the HPC cluster, while the cloud handles CFD output processing. Microservice-like processes optimize resource utilization and minimize idle times. The hybrid model adapts dynamically to varying workloads.Scalability ensures efficient resource allocation across simulations and locations. Centralized services facilitate collaboration among users. Machine learning algorithms benefit from automatic data provision. We present a proof of concept implementation, demonstrating the framework's effectiveness.

***IndexTerms* - High Performance Computing, Automotive HPC, HPC, Cloud.**

I. INTRODUCTION

According to [1], the global "datasphere" (data created, captured and replicated) is predicted to grow to 175 zettabytes by 2025. This unprecedented amount of data is expected to drive artificial intelligence and digital transformation challenges for the benefit of both public and private sectors.

A promising field for data-driven solutions is computational fluid dynamics (CFD). Nowadays, surrogates or digital twins can be created based on CFD data to provide real-time control or optimization of complex processes [2], [3], [4]. In addition, the increase in global computing resources' performance and accessibility has allowed researchers and domain experts to use CFD models with higher temporal and spatial resolution than was possible just a few years ago. All these factors unavoidably entail an increment in the amount of data generated, processed and transferred. The upcoming challenge is to provide tools for automatic and efficient processing. For CFD data this step remains an active field of research and development; existing solutions lack scalability, portability, and interoperability for extracting and analyzing results of fluid flow simulations.

The cloud computing model with its on-demand resources model [5] is a very attractive proposition for compute and storage intensive problems like CFD and machine learning. While cloud cannot yet completely replace custom-built, on-premises HPC systems, it offers an attractive alternative for certain classes of applications [6], [7], [8]. Combining HPC and cloud systems into a hybrid cloud [5], brings together the best of both worlds — performance of on-premises HPC systems with scalability and agility of a cloud.

We can take advantage of cloud native [9] technologies, which focus on maximizing advantages of cloud's extreme agility and scalability, and modify our workflows to take full advantage of both HPC and cloud computing models. A HPC-cloud native approach offers the following advantages:

- cost efficient execution of complex workflow, with ability to match different stages of a pipeline with most appropriate resources,
- ability to overlay simulation and post-processing stages of a workflow to improve overall time to solution,

- autoscaling of cloud resources ensuring just in time scaling of post-processing stages,
- better opportunities for collaboration using as-a-service paradigm on a cloud, which is better suited for information exchange than on-premises HPC resources.

In this paper we propose an automatic, non-inclusive, on-demand and scalable framework for concurrent analysis and rendering of data generated by CFD simulations based on hybrid HPC-cloud native technologies. Our solution addresses portability and interoperability by using existing configurations, where CFD tools such as OpenFOAM [10], PyFR [11], Code_Saturne [12] or Basilisk [13] are compiled and optimized on a given HPC system, while post-processing is done on the cloud, thus allowing users of these different codes to benefit from such a common and centralized service for analyzing their data independently of the location, architecture and operating system. For instance, this service can be used for analyzing and rendering data produced by any code that can write an output with any of the formats supported by the Visualisation Toolkit (VTK) [14] readers, such as legacy and XML-based VTK, Wavefront OBJ or stereolithography files e.g. STL format amongst others. This type of set-up provides a convenient and versatile solution for feeding machine learning models with data from different sources. Finally, although we propose a CFD code running on a HPC facility and performing data analysis on the cloud, other arrangements such as cloud–cloud or desktop–cloud could be set-up depending on the needs.

For the prototype implementation of the proposed framework, we have chosen a characteristic multiphase problem computed at the microscopic level, which provides high-resolution simulations of multiphase flow interfaces, such as those representing drops or bubbles. Many industrial processes require precise drop or bubble size control for optimization and safety evaluation. We focus our attention on the simulation of the injection of one fluid into another. Fluid injection dispersion and drop or bubble formation are encountered in chemical mixing, aeration, 3D printing, pharmaceutical manufacturing, microfluidic devices, oil and gas spill mitigation, and agricultural pesticide disposal. The challenge is typically to compute and quantify various population characteristics such as size or velocity distribution, total bubble surface, rate of break-up or coalescence. For many applications, such knowledge is of great importance for evaluating process variants prior to any industrial implementation.

Recent literature contains several examples of high-resolution simulations of dispersions. In particular, in [15] atomizing liquid sheets are simulated and demonstrate a significant difference compared with standard linear theories. A direct numerical simulation of co-current flow in [16] gives a quantitative picture of high-order statistics of turbulence and their interaction with the break-up process. Break-up in the presence of cross-flow was studied in [17]. A very important feature of the problem is the need for high temporal resolution. To correctly track the dynamics of each multiphase object, many highly correlated snapshots are required to compute particle characteristics, avoid double counting or identify parent–child relationships in break-up and coalescence events.

The presented approach does not replace, but complements existing technologies for interactive visualization or in situ post-processing [18]. The latter enables the analysis of data without writing to system and removes I/O bottlenecks. This can be achieved through a service running on dedicated hosts or processes running on the same host as the simulation. Several frameworks have appeared in recent years e.g. [19], [20] and [21] or generalization such as [22]. In contrast, our approach focuses on efficiently extracting features of interest to minimize I/O and enable a concurrent rendering and analysis on the cloud using specifically designed containerized tools. Instead of using a monolith application, we broke down the application into small microservice-like processes to take advantage of the cloud features. In this way, a given resource can be set-up and shared between different simulations, codes and locations to minimize idle times and dynamically adjust the capacity to the workload. It is worth mentioning that the memory footprint of output data for multiphase flow, scales with the square of the inverse cell width, as opposed to cube scaling of the simulation. This is because only the multiphase object surfaces are required rather than complete volumetric fields. Although our focus is on multiphase flow, such scaling also applies to other single phase flow where features of interest can be represented with isosurfaces such as velocity gradient tensor or vorticity in single or multiphase flow. The same also applies to simulations where data of interest can be represented in 2D planes.

We developed a command line interface (CLI) tool that relies on VTK components for reading, writing and processing data. The tool has been designed for processing one CFD simulation output file per job on a multithreaded single core, thus making it highly scalable and parallelizable. Special attention has also been given to the strategy for extracting the feature of interest and sending it to a cloud by the simulation workflow. Isosurfaces representing the interface of the injected fluid were calculated in parallel using a surface generation routine that provided a fast solution and reduced output sizes. Data was transferred from HPC to a cloud storage service with a multi-part upload which showed a good performance for the scenarios evaluated. Once no longer needed, data can be either purged or archived on the cloud. In addition, the cloud service used in this approach provides automatic cost optimization features, which makes the choice compatible with the scalable and cost-efficient system proposed. VTK jobs were submitted to a Kubernetes [23] cluster, which was initialized on-demand by the workflow. Cluster resources (worker nodes) were dynamically scaled up and down according to the workload demands of 7 CFD simulations representing different flow conditions.

Exascale computing (1018 floating-point operations per second) is the next major milestone in the process of exponential growth of HPC systems that occurred for over half a century (the Moore's law) [1]. One of the main concerns of the HPC community is that current high-end hardware cannot achieve this leap with reasonable power consumption (≈ 20MW) [2]. Nowadays, several public and private institutions around the world are investigating different aspects that should lead to the future generation of supercomputers, looking for innovative solutions in the way supercomputers interconnect, compute and move and store data. This set of initiatives have entitled this problem as the Exascale challenge.

The paper is structured as follows. Section II describes an example of the target Automotive use case. Section III shows the framework set-up for our hybrid HPC-cloud solution. Section IV demonstrates the use and benefits of the proposed framework. Section V provides conclusions and addresses future work.

II. Opportunities and Challenges

After years of hard work and investment, the supercomputer hardware infrastructure capability has reached the international advanced level. However, due to the lack of stable investment and long-term accumulation of scientific computing software such as basic algorithm library, parallel algorithm library, high-performance computing application middleware and application software in various disciplines, all aspects of scientific computing applications are heavily dependent on foreign countries. In particular, the use of foreign software accounts for more than 90%, which has become a "card neck" problem that restricts the further development of high-performance computing. In order to better develop scientific computing applications and strengthen R&D application software, we should strategically lay out and plan the development route and implementation plan of International scientific computing in the future, and give five suggestions[12].

- Establish a national high-performance computing software R&D center and provide long-term stable support. Scientific computing software for a number of important areas of domestic processor development; through the mining of scientific issues in a number of application areas, based on application needs, establish long-term stable scientific goals. Focus on this goal and develop long-term, continuous software development.
- Vigorously strengthen the development of middleware for high performance computing applications. In recent years, with the support of the national “863” plan and the key research and development plan “High Performance Computing”, researchers have successfully developed the three-dimensional parallel structure adaptive software framework and the three-dimensional parallel adaptive finite element software platform. Researchers at the Computer Network Information Center of the International Academy of Sciences, with the support of the Center for Computational Science Applied Research, are developing the parallel computing framework. Through framework support, parallel computing details can be shielded from applied scientific computing researchers, allowing them to focus on physical model and computational method innovation and accelerate the integration of computing programs with new and new models, ultimately enabling rapid implementation of massively parallel computing applications. Development.
- Further strengthen the planning and development of scientific computing application software at the national strategic level. Scientific computing applications are the product of cross-integration in the fields of computational science and applied science. It is far from enough to rely solely on the spontaneous or fragmented R&D of researchers in institutions of higher learning and research institutes. It should be based on the needs of national strategic and scientific issues, demand-driven, and the need to solve problems. It takes only 10 years or more of planning and firm implementation to see results.
- Improve the basic research level of scientific computing applications. Scientific computing capabilities include computer software hardware, supporting software, and the ability to implement algorithms. Only by improving the basic research level of scientific computing applications can higher requirements be placed on the hardware and

III. Example of Automotive Use Case

Application Virtualization, which includes simulation, simulation, and interpretation techniques. Java virtual machines are typically virtualized at the application layer. Based on the application layer virtualization technology, the user's personalized computing environment can be reproduced on any computer by saving the configuration information of the user's personalized computing environment. Service virtualization is a hot topic in recent years. Service virtualization enables business users to quickly build applications on demand. Through service aggregation, the complexity of service resource usage can be shielded, making it easier for users to directly map business requirements to virtual. Service resources[5].

Application virtualization is a layer of technology in a large family of virtualization that provides multi-user remote access to centralized application resources to deliver applications as a service to users. In the traditional application usage mode, users need to install each application and prepare enough computing and storage space to maintain the software running. The daily backup maintenance and upgrade management of the software is troublesome and the data security is poor[6].

The basic principle of application virtualization is to separate the application's computational logic and display logic, that is, interface abstraction, without installing software on the client computer. When the user accesses the virtualized application through the network, the client computer sends the accessed data request to the virtualization management server, and the server needs to run the application to establish a separate session and perform logical operations, and then the generated result is generated. Delivered to the client computer desktop, users can transparently use the virtualized application on the server to get the same experience running the application locally[7].

The core technology in cloud computing is virtualization. It can be said that virtualization is an important feature that distinguishes cloud computing from traditional computing models. Virtualization can transfer the entire execution environment of the application to other nodes in the cloud computing platform in a packaged manner, which realizes the isolation between the execution environment of the program and the physical environment, making the environment change of the application easy to implement. It is because of the maturity and wide application of virtualization technology that computing, storage, applications and services in cloud computing have become resources. These resources can be dynamically expanded and configured, and cloud computing can ultimately be logically in a single whole form. Presented. With the continuous development of virtualization technology, the concept of virtualization has been extended to multiple levels of IT technology. The current virtualization technologies mainly include: full virtualization, paravirtualization, operating system layer virtualization, hardware virtualization, application virtualization. And so on.

This section describes our proof of concept implementation of the proposed HPC-cloud native framework. Fig. 4 shows an overview of the infrastructure implemented in this contribution. We describe a hybrid HPC-cloud solution with a CFD Cloud Analyzer and Renderer Tool, hereon in referred to as CFD Cloud Analyzer and Renderer Tool, located on the HPC side where the data is generated and controlling the workflow and communication with the cloud services. In this section, we discuss the system model of task offloading in a VEC network. The system consists of multiple RSUs surrounding vehicular users, as depicted in tasks to the nearest available RSU, i.e., an RSU that covers the user and has a moderate computational load. As a user moves, he or she exits the coverage area of the current RSU and enters that of another RSU. QoS concerns oblige the system to offload the unfinished tasks of that user to the new RSU. In addition, when the user moves, there are multiple candidates for the new RSU. The choice of the new RSU is based on the server availability and the estimated computation delay. Most navigation systems predetermine the path for each user from the starting point to the destination. We let that path pass through various coverage circles, as shown in Fig. 2. In every coverage circle, the load of each RSU may range from light (green) to moderate (red/green gradient) or heavy (red). MPI technology allows you to organize various schemes for the parallel execution of computational processes aimed at solving one scientific problem. Process execution can be organized on one or several nodes of a high performance computing cluster. In this case, the exchange of data between processes is organized through the MPI interface. The use of container technology introduces an additional level of nesting when distributing application processes across computing components. The following distribution variants for MPI processes are possible when using containers in a high-performance computing cluster.

Variant 1. Execution of several processes in one container by executing mpirun in the same container.
The advantage of this method is the absence of the need for additional configuration of the software environment of the computing cluster for MPI processes. To implement this option, it is enough to deploy MPI libraries in the container. The user has the ability to perform a group of parallel processes inside the container and get the resources of the computing node on which the container operates.

The disadvantage of this method is the inability to run this option on a group of nodes connected by an interconnect network. Thus, this scheme can be implemented in order to debug application software or perform scientific calculations that do not require the resources of a whole cluster and which can be performed on a single computing node.

With this method of execution, containers are created on the computing nodes by the number of processes being executed, and then the mpirun utility creates one process in each container that is designed to solve an applied scientific problem. Parallel processes can exchange messages and data via the MPI interface using shared memory of a computing node or a high-performance interconnect network. The mpirun utility is executed outside the containers with application processes; it can be performed either on one of the computing nodes of the cluster or in a specialized container.

Variant 2. Running one process in one container by running mpirun in an external environment.
The advantage of this method is the ability to perform parallel processes on a group of nodes of a high performance computing cluster using standard MPI mechanisms for interaction between parallel processes. The disadvantage of this method is the need to dynamically form a pool of containers (based on their IP addresses) for transferring to mpirun utility as source data for creating parallel processes.

These shortcomings are not critical, the problem of creating a dynamic pool of container addresses is solved by developing special scripts to create a container environment and execute parallel processes. The problem of container multiplicity is not critical, because, unlike virtual machines, the overhead of container operation is small compared to operating system processes.

Variant 3. The combination of variants 1 and 2.
With a combination of variants 1 and 2, it is possible to execute processes on a group of nodes of a high performance cluster. At the same time, one container is created on each cluster node to solve a specific application, in which a group of parallel processes is performed. Utility execution should also be performed outside containers intended for the application.

The advantages of this option include the reduction in the number of containers operating on one node of a highperformance cluster.

The disadvantages are similar to variant 2 and are associated with the need to create a functioning environment using special software.

Note that for the application of MPI technologies and containers in a hybrid high-performance cluster, it is necessary to ensure their integration with the workload management system. It is required to create a mechanism that allows the control system to create a group of containers intended for the application, to create a list of ip-addresses of the nodes to be transmitted to the mpirun utility and to execute the utility on one of the cluster nodes or in a separate container.

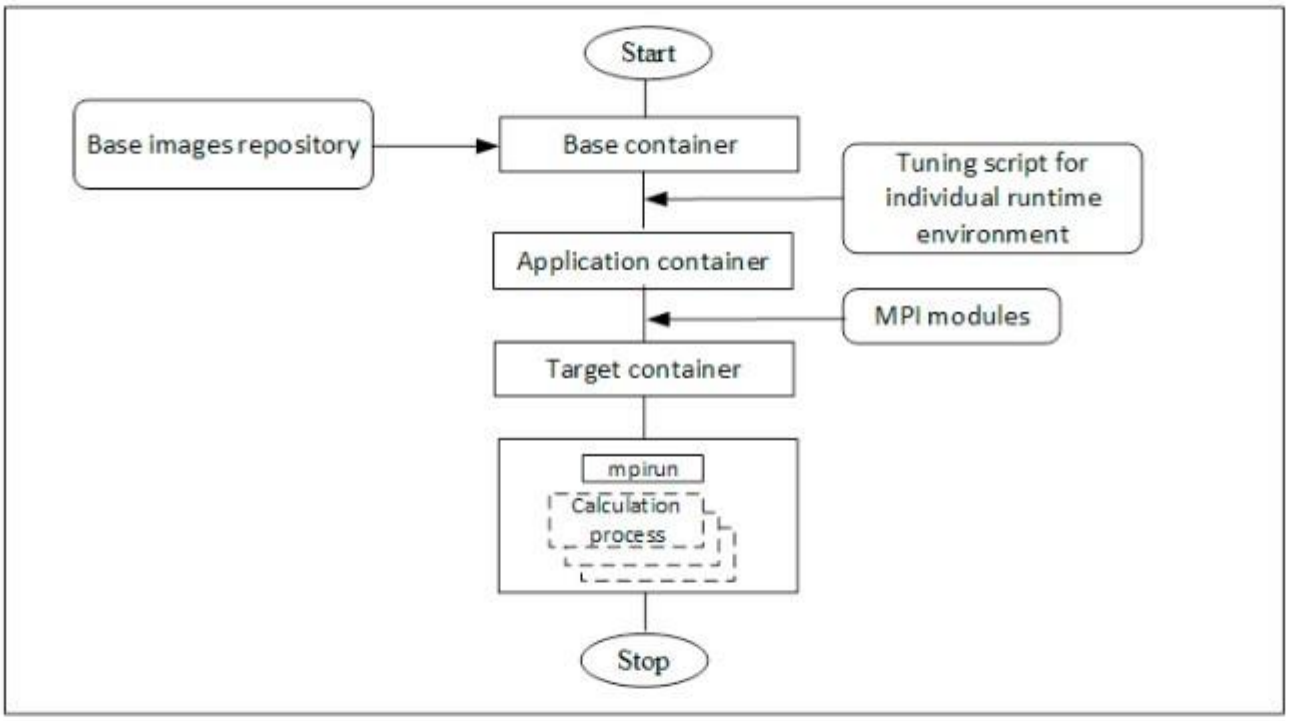
When performing such actions, the system should keep track of the resources allocated to each parallel process, allow managing the execution of a group of containers, provide prioritization, queuing, and monitoring of resource use. Also, using the control system or additional software integrated with it, "garbage collection" should be carried out in case of program failures and errors. In the event of a malfunction in any parallel process running in the container, leading to the need to complete the program, all containers related to this program must be unloaded.

Management of jobs using GPU resources imposes additional requirements on the operation of the workload management system. Based on the fact that switching the accelerator between computational tasks is a lengthy process, the control system should allocate GPU resources exclusively for the computational process. In the presence of parallel processes, it is optimal to allocate to each of them one or more graphics accelerators for the entire duration of the calculations. Below are the algorithms for the formation of an individual software environment using container technology, focused on the use of MPI and allowing the use of computing resources of graphic accelerators of calculations.

IV. Framework Setup

When deploying containers according to option 1, you must perform the following steps: Using the docker virtualization environment, create a container from the base image; Configure the individual execution environment to solve an applied scientific problem [9]; Install MPI libraries into the container (if they are not in the base image); Execute mpirun in the container indicating the number of parallel processes on the local node involved in solving the scientific problem.

Figure 1 shows the Framework Setup



Note that when using parallel graphics accelerator resources, it is preferable to proceed from the rule of exclusive allocation of the accelerator to the process in order to avoid task switching and performance degradation.

When deploying containers according to variant 2, you must perform the following steps:

- Using the docker virtualization environment, create a group of containers from the base image according to the number of planned parallel processes on one or more nodes of the computing cluster;
- Algorithm for executing a task in a single container;
- In each container, configure the individual execution environment. Note that at high costs for the dynamic creation of an individual environment, it is preferable to create and save it in the base container in advance. In this case, you can skip the data step;
- Install MPI libraries in containers (if not in the base image);

V. Uses and Benefits of the Proposed Framework

Providing the resources of a hybrid high-performance cluster can be considered as a cloud service of a digital platform [10]. The formation of a repository of basic images intended for applied research is one of the main tasks of the competence center, which is part of the unit that operates the computing cluster and provides the provision of cloud services. The hybrid high-performance computing cluster of the CKP «Computer science» of FRCCSC RAS [11] in the repository of basic images contains software environments designed to solve applied problems in the areas of artificial intelligence, mathematical modeling, materials science, and biomedical chemistry. The presence of such images allows us to provide PaaS-type cloud services to clients to solve applied problems in these fields of science [12-15].

In the general case, the problems of managing computing jobs in a hybrid high-performance computing cluster boils down to distributing computing resources between tasks, managing job queues, priorities, and accounting for resources used. Open source and proprietary workload management systems, such as, for example, SLURM, IBM Spectrum LSF, successfully cope with these tasks. Support for MPI technology is provided in these computing process control systems by default. However, when using this technology in combination with container virtualization technology, it creates problems related to the creation and management of virtual environments and the execution of applied computing tasks in them.

- Integration through the development of middleware (scripts), which acts as an aggregator of information about containers and nodes, allows you to perform the following actions necessary to execute custom applications:
- Allocate the necessary computing resources to a group of physical nodes of a hybrid high-performance computing cluster; Create the necessary number of containers or pseudo-hosts;
- Configure the individual execution environment;
- Perform automatic (dynamic) and predefined (static, by template) allocation of the processes of one applied task on the hosts and pseudo-hosts of the hybrid cluster;
- To manage the execution of the computational task on all nodes and pseudo nodes of the cluster in accordance with the policies for servicing users of the computing cluster.

When allocating computing resources to increase the efficiency of using the cluster, the workload managemen system should proceed from the rule that GPU resources should be allocated exclusively to each parallel proces within the MPI framework to avoid loss of time for switching GPUs. CPU resources should be allocated by cores t ensure parallel execution of processes that do not require GPU resources. Features of reserving CPU and GP resources when performing one or more tasks using MPI. Suc a resource management policy makes it possible to increase the efficiency of servicing user tasks, but does not affect the efficiency of executing the application themselves, which depends on the quality of the program code. Approaches to evaluating the effectiveness of program code using CPUs and GPUs are associated with profiling user applications when working with central processes and computing accelerators [7].

IV. CONCLUSION

The parallel execution of MPI in the cloud infrastructure using container virtualization requires the development of algorithms for deploying containers of individual runtime environments and their inclusion in the general field of the MPI task.To provide a cloud service, the creation of specialized containers is required, including software libraries for working with MPI and interconnect, as well as integrated packages that are configured to interact between parallel processes.

For the high-quality provision of a PaaS-type cloud service, the competence center must create and keep up to date a repository of container images focused on solving applied problems in various fields of science and technology. The development of middleware for managing tasks oriented to parallel execution under MPI control will allow controlling the computing process in a hybrid cluster using standard control systems, taking into account resources, ensuring cluster loading and priority task execution.

References

[1] Abramov, S.M. and Lilitko, E.P. (2012). The status and development prospects of computing systems of super-high performance. Information Technologies and Computing Systems, 2, pp.6-22.

[2] Bryukhov, D.O., Vovchenko, A.E., Zakharov, V.N., Zhelenkova, O.P., Kalinichenko, L.A., Martynov, D.O., Skvortsov, N.A. and Stupnikov, S.A. (2008). The middle ware architecture of the subjectmediators for problemsolving over a set of integrated heterogeneous distributed information resources in the hybrid grid-infrastructure of virtual observatories. Informatics and applications, 2(1), pp.2-34.

[3] Budzko, V.I., Bryukhov, D.O., Devyatkin, D.A., Skvortsov, N.A., Smetanin, N.N., Stupnikov, S.A. and Shelmanov, A.O. (2017). Scalable architecture of a system for extracting information from data on the Arctic zone. In: Collection of Information Technologies and Mathematical Modeling of Systems 2017. Proceedings of the international scientific and technical conference. pp.90-94.

[4] Zatsarinny, A.A., Gorshenin, A.K., Kondrashev, V.A., Volovich, K.I. and Denisov, S.A. (2019). Toward high performance solutions as services of research digital platform. In: Procedia Computer Science. vol.150, pp.622-627.

[5] Gorchakov, A.Y. (2019). K-frontal method of nonuniform coverings. International Journal of Open Information Technologies, 7(8), pp.65-69.

[6] Abramov, S.M. (2018). Analysis of supercomputer cyber infrastructures of the leading countries of the world. In: Supercomputer technologies. SKT-2018. Materials of the 5th All-Russian Scientific and Technical Conference. Rostov-on-Don, pp.11-18.
[7] Volovich, K.I., Denisov, S.A., Shabanov, A.P. and Malkovsky, S.I. (2019). Aspects of the assessment of the quality of loading hybrid high-performance computing cluster. In: CEUR Workshop Proceedings. vol.2426, pp.7-11.
[8] Volovich, K.I. and Denisov, S.A. (2019). The main scientific and technical problems of using hybrid HPC clusters in materials science. In: Materials of the I international conference “Mathematical modeling in materials science of electronic components. MMMEC-2019. Moscow, pp.15-18.
[9] Volovich, K.I., Denisov, S.A. and Malkovsky, S.I. (2019). The formation of an individual modeling environment in a hybrid high performance computing cluster. News of higher educational institutions. Materials of electronic equipment, 22(3), pp.197-201.
[10] Zatsarinny, A.A., Kondrashev, V.A. and Suchkov, A.P. (2019). The system of scientific services as a relevant component of scientific research. Systems and Means of Informatics, 29(1), pp.25-40.
[11] FRC CSC (2020). The regulation CKP «Informatics». [online] Available at: http://www.frccsc.ru/ckp [Accessed 23 Jun. 2020].
[12] Mikurova, A.V., Skvortsov, V.S. and Raevsky, O.A. (2018). Computer assessment of the selectivity of inhibition of muscarinic receptors M1-M4. Biomedical Chemistry: Research and Methods, 1(3), pp.1-9.
[13] Abgaryan, K.K., Zhuravlev, A.A. and Reviznikov, D.L. (2017). Parallel data processing in computer modeling problems of high-speed interaction of solids. In: Materials of the XX International Conference on Computational Mechanics and Modern Applied Software Systems. VMSPSPS 2017. Alushta, pp.27-28.
[14] Yadrintsev, V.V., Klyubina, K.V., Tikhomirov, I.A. and Gershelman, A.F. (2018). Choosing a server solution for a digital text search and
analysis platform. Systems and Means of Informatics,28(3), pp.26-38.
[15] Goloviznin, V.M., Gorchakov, A.Yu., Zalesny, V.B, Mayorov, P.A., Mayorov, P.A., Semenov, E.V. and Soloviev, A.V. (2019). A numerical model for solving equations of geophysical hydrodynamics using a new class of conservative difference schemes that preserve angular momentum on computational grids. In: Seas of Russia: Fundamental and Applied Research Abstracts of the All-Russian Scientific Conference.
[16] Zatsarinny, A.A., Gorshenin, A.K., Volovich, K.I., Kolin, K.K., Kondrashev, V.A. and Stepanov, P.V. (2017). Management of scientific services as the basis of the national digital platform "Science and Education". Strategic priorities, 2(14), pp.103-113.
[17] Kondrashev, V.A., Volovich, K.I. Service management of a digital platform on the example of high-performance computing services. In: Materials of the International scientific conference. Voronezh, September 3-6. pp.217-223.