engineering proceedings

MDPI

Proceeding Paper

Providing High-Speed Data Access for Parallel Computing in the HPC Cluster †

Sergey Denisov *, Konstantin Volovich and Alexander Zatsarinny

Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, 44, Build. 2, Vavilova Str., 119333 Moscow, Russia; kvolovich@frccsc.ru (K.V.); azatsarinny@ipiran.ru (A.Z.)

* Correspondence: sdenisov@frccsc.ru; Tel.: +7-499-135-4320

† Presented at the 15th International Conference "Intelligent Systems" (INTELS'22), Moscow, Russia, 14–16 December 2022.

Abstract: The article discusses approaches to building parallel data storage systems in high-performance clusters. The features of building data structures in parallel file systems for various applied tasks are analyzed. Approaches are proposed to improve the efficiency of access to data by computing nodes of the cluster due to the correct distribution of data in parallel file storage.

Keywords: HPC cluster; parallel file system; Lustre; NVMe

1. Introduction

The growing need of scientific teams, industrial enterprises, and commercial firms in solving problems that require high-performance computing resources demands the creation of computing tools provided to users using cloud and platform technologies. At present, such computing resources are supercomputers and HPC clusters that make it possible to carry out calculations using parallel computing technologies. So, the tasks of mathematical modeling, global optimization, big data analysis, and the training of neural networks cannot be solved using a single server; powerful multi-node clusters united by high-performance computing networks are required. The workflow management in such clusters is a research task aimed at optimizing the computing resources load, minimizing the waiting and computing time, managing job priorities, and providing computing jobs with initial data.

Based on HPC clusters, it is possible to create a research infrastructure that provides tools for miscellaneous scientific calculations to teams of scientists, industry, and developers of high-tech commercial products [1]. The creation of shared research facilities on the basis of scientific centers, enterprises, organizations of the Russian Academy of Sciences and universities makes it possible to increase the efficiency of using computing facilities and expand the circle of consumers of high-performance computing resources [2]. This approach to the use of computational tools involves the parallel execution of miscellaneous types of computational tasks within a single high-performance cluster. In this case, in order to efficiently use computing resources and minimize the loss of time and money for re-configuring the HPC cluster, it is necessary to develop a technology that allows creating individual software environments for various computing tasks and ensuring their parallel execution in a HPC cluster [3], wherein each computational task obtains access to the parallel processing information means and inter-process communication. The means of HPC cluster workflow management create an individual profile for inter-process interaction in a dynamic virtual environment for each computational task.

The optimization of the loading of the shared research HPC cluster is generally aimed at ensuring the maximum loading of computing modules, reducing the waiting time for tasks in the queue, and minimizing equipment downtime.

Citation: Denisov, S.; Volovich, K.; Zatsarinny, A. Providing High-Speed Data Access for Parallel Computing in the HPC Cluster. *Eng. Proc.* **2023**, *33*, 54. https://doi.org/10.3390/engproc2023033054

Academic Editors: Askhat Diveev, Ivan Zelinka, Arutun Avetisyan and Alexander Ilin

Published: 21 July 2023

To implement such requirements for a HPC cluster, it is of particular importance to provide computing tasks with initial data. The performance of the means of providing data to computational jobs should, for example, minimize the processing downtime by waiting for data to be provided. The tasks of providing data to computational tasks in multi-node computing systems are solved by creating a specialized file storage that provides parallel access to cluster computing nodes and data. Thus, the data exchange height speed is ensured in the conditions of parallel operation of virtual software environments for the execution of miscellaneous types of computational tasks.

The classical scheme for organizing parallel file access is the use of a group of data storage nodes connected to computing nodes by a high-speed network. A parallel file system deployed on storage nodes creates a single data space, ensures information consistency, the duplication of information on different nodes, and control of access to files by computing nodes.

Both proprietary and open-source implementations of parallel file systems are available today. For example, proprietary solutions include

- General Parallel File System (GPFS)—developed by IBM;
- Google File System.

IBM GPFS is used by the manufacturer as part of a commercial product, the elastic storage system, which is a scalable hardware and software data storage system. The Google File System is used by Google in the company's high-performance computing clusters.

Among open source parallel file systems, the Lustre project is currently actively developing, originating from Carnegie University, now supervised by Intel Corporation. Another open-source project is the Ceph parallel file system supervised by RedHat (IBM).

The direction of providing parallel access to data is currently an actively developing area of informatics [4–6]. The main efforts of developers and researchers are concentrated in the following areas:

- Maximizing the volume of data storage;
- Ensuring the required performance during data transfer, taking into account the number of computing nodes and data storage nodes;
- Development of methods for scaling the file system while maintaining or increasing performance;
- Development of methods for improving the reliability of parallel file systems.

The architectures and ways of representing data in parallel file systems are discussed below.

2. Parallel File System Architecture

Consider the architecture of a parallel file system using the Lustre project as an example.

Lustre is a high-performance file system composed of servers and storage. The Metadata Server (MDS) keeps track of metadata (such as ownership and access rights to a file or directory). Object storage servers provide file I/O services for object storage targets that host the actual data store. The storage targets are typically a single disk array. Lustre's parallel file system achieves its performance by automatically splitting data into chunks known as "stripes" and writing the stripes in a round-robin fashion across multiple storage objects. This process, called "striping", can significantly increase file I/O speed by eliminating single disk bottlenecks [7].

A parallel storage system consists of many components, including drives, storage controllers, I/O cards, storage servers, SAN switches, and related management software. Combining all of these components together and tuning them for optimal performance comes with significant challenges. Figure 1 shows the Lustre parallel file system architecture.

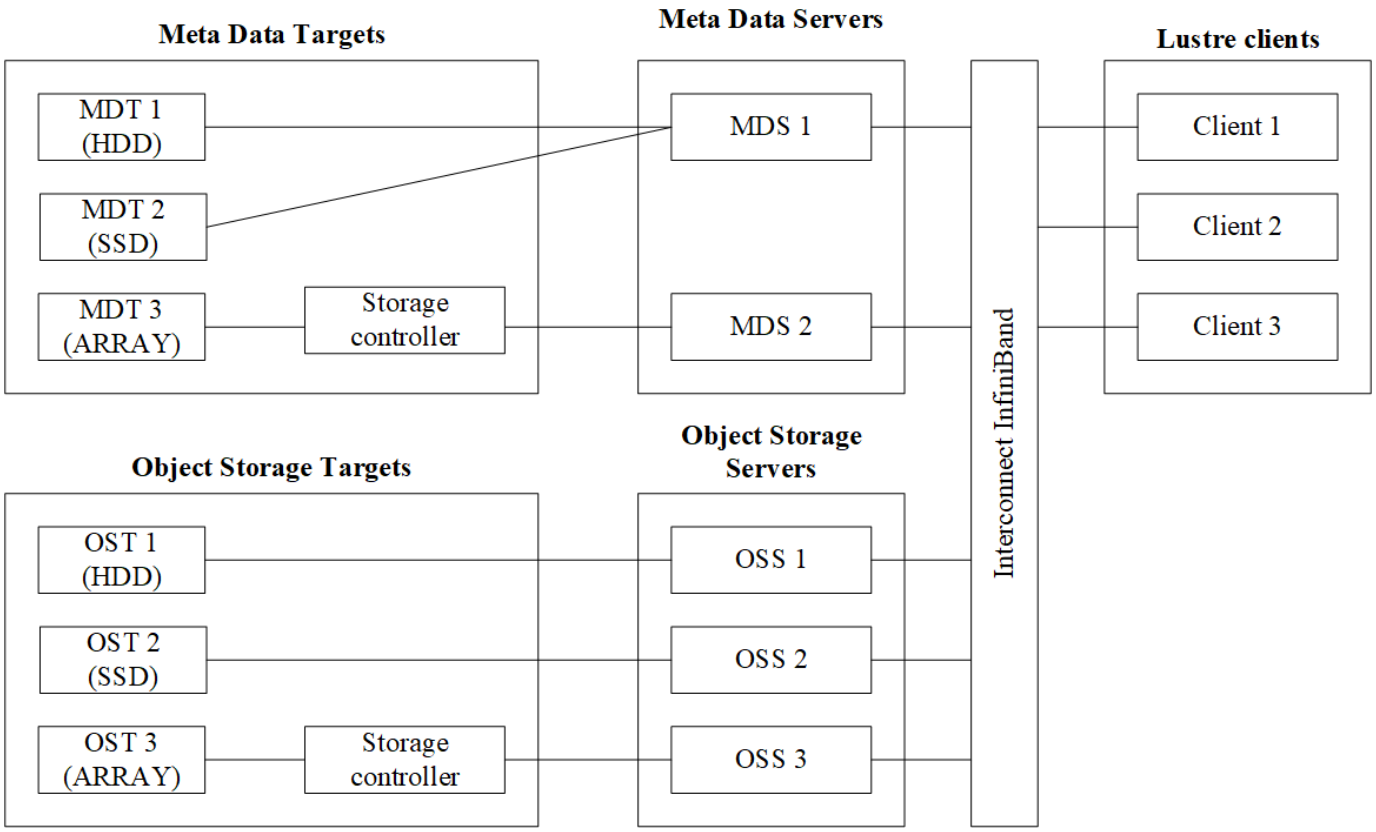

Figure 1. Lustre parallel file system architecture.

The elements of architecture are as follows:

- Control nodes—meta data servers (MDSs);
- Data storage and provision nodes—object storage servers (OSSs);
- Data storage elements—object storage target (OST);
- Metadata storage elements—meta data targets (MDTs);
- High-speed data-processing network.

Metadata servers (MDSs) are the control components of a parallel file system that store information about all the data in the system, as well as serving client requests for access to data. Metadata are stored on information resources called metadata targets (MDTs), implemented either as physical disks or as block devices in a data storage system. To ensure fault tolerance, metadata servers can be duplicated.

Access to parallel file system clients data is carried out through the metadata server. When servicing requests, the metadata server identifies the client and selects one of the nodes for storing and providing data—OSS. The choice is made based on meta-information about the availability of the requested data on storage elements available to this OSS, as well as based on information about OSS loading by requests from other clients. After that, the interaction between the client and the selected OSS is carried out directly through a high-performance data network. This solves the problem of the formation of "bottlenecks" in the access of computing nodes to a single data storage [8].

Data storage elements are disk drives directly connected to the OSS as well as logical drives formed by storage systems based on disk arrays. Data storage and provision nodes provide the caching of information contained on disks and high-speed exchange with the data transmission network.

Thus, the architecture of a parallel file system makes it possible to provide data for a group of computational tasks due to the parallel operation of a group of storage nodes.

3. Data Storage Structure in A Parallel File System

As noted above, the parallel file storage architecture allows you to provide computing nodes with access to different storage nodes, which allows you to avoid "bottlenecks" and prevent performance degradation.

Note that different application tasks may have different requirements for the data structure stored in the file system.

For big data-processing tasks—telemetry streams, the analysis of accumulated information arrays in order to obtain new knowledge (data mining)—where parallel processes of one computing task performed on different computing nodes require access to independent data arrays, it is enough to place one application data copy on each storage node and ensure the interaction of each pair of "computing node–storage node" over a high-speed data transmission network.

For data-intensive tasks (e.g., preparing training sets for neural networks, and training neural networks [9]), a more complex structure data storage is required, allowing different computing nodes to access the same file from several computing nodes. This feature is supported by the parallel file system by fragmenting files into separate blocks (stripes) [10].

The term "number of stripes" refers to the number of fragments into which a file is divided; in other words, the number of OSTs that are used to store the file. So, each stripe of the file will be in a different OST. The "stripe size" refers to the size of a stripe recorded as a single block in the OST.

The benefits of striping include the following:

- Increased I/O throughput due to multiple file areas being read or written in parallel;
- Helping to balance the use of the OST pool.

However, striping has disadvantages if performed incorrectly, such as increased overhead due to internal network operations and contention between servers, and throughput degradation due to inappropriate striping settings.

The default number and sizes of stripes are chosen to balance the I/O performance needs of multiple parallel execution scales and file sizes. Small files should be striped at a value of 1. However, setting the stripe number too low can degrade I/O performance for large files and parallel I/O. Thus, the user must carefully select strip specifications according to application data.

The striping must be compatible with the application's I/O strategy and output size. The increase in the number of stripes and/or the size of the stripes should be proportional to the number of nodes used for I/O. As a general rule, an application should try to use as many OSTs as possible. Thus, when writing a large single file in parallel, the maximum allowable value for the stripe counter is set. Alternatively, when writing a large number of small files in parallel, set the interleave counter to 1. The intermediate number of concurrent output files may work better if the number of stripes is greater than 1. An experimental estimate of the number of stripes is advisable for best performance. Note that for a number of tasks, the file size correlates with the number of computing nodes that perform parallel writing to it. Therefore, adjusting interleaving based on file size is usually sufficient, and a simpler starting point for estimating the number of lanes for subsequent processing optimization.

Figure 2 shows an example of file distribution across six storage elements.

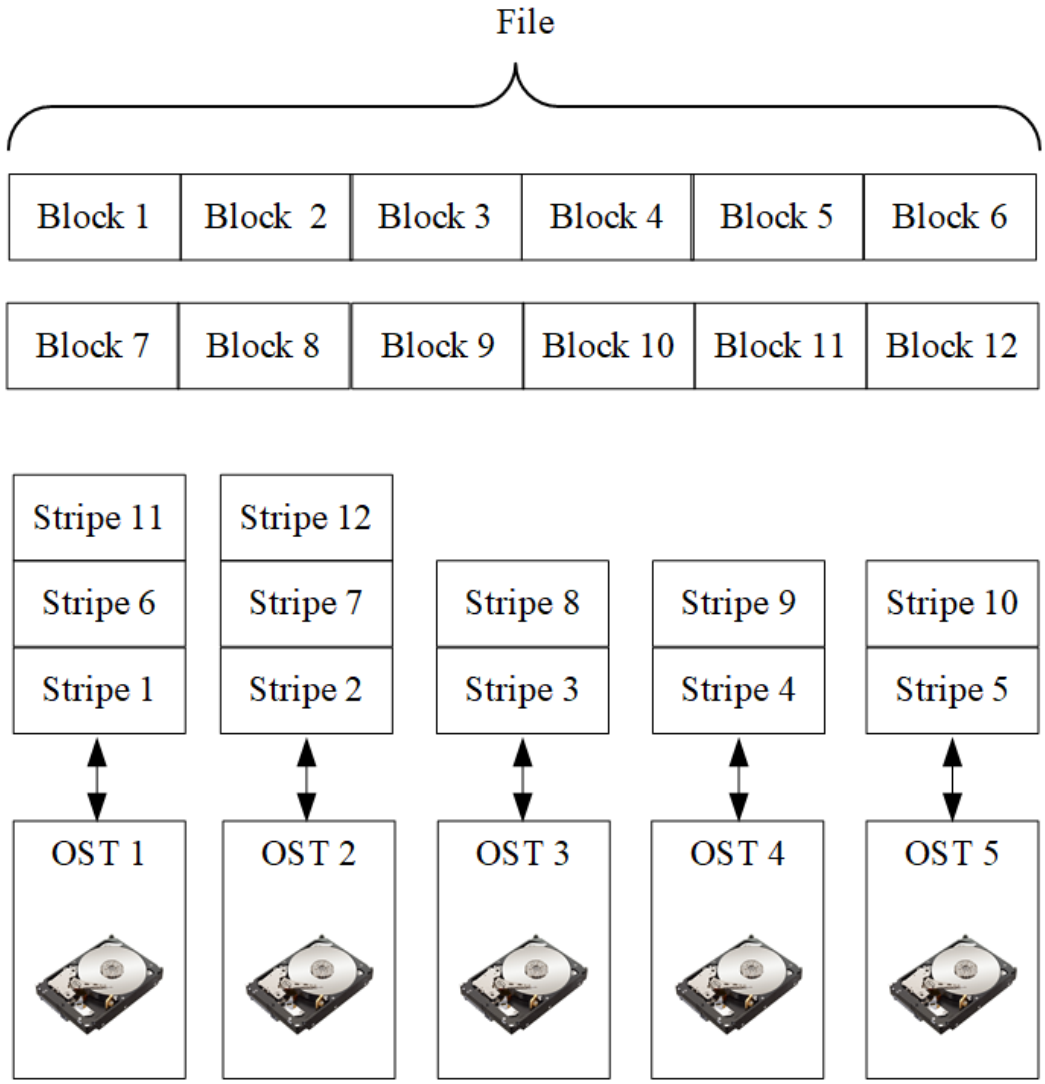

Figure 2. The file distribution example.

4. Experience of Application of Parallel File Systems in The Shared Research Facilities "Informatics"

To conduct research in the field of application of parallel file systems, taking into account the diversity of tasks solved by a HPC cluster, the parallel file storage stand was created using the infrastructure of the Shared Research Facilities "High Performance Computing and Big Data" (CKP "Informatics") of FRC CSC RAS (Moscow, Russia), which allows you to provide the data for different types of computational tasks functioning on the basis of virtual computing environments [11].

During the creation of the stand, the Lustre file system version 2.14.50 was deployed on six servers of the ARM architecture manufactured by Huawei.

During the experiments carried out on the created stand, the following scientific and practical work was carried out:

- Deployment of the Lustre parallel file system by compiling open-source software;
- Assessment of the performance and quality of functioning of the parallel file system in the base case when performing computational tasks of various types;
- Implementation of the developed storage system architecture, data models, access scenarios by setting up various data storage scenarios;
- Evaluation of performance and quality of functioning of a parallel file system for test computing tasks that have different requirements for data storage (size and number of files, multiple access to one file by a group of computing nodes);

- Design of recommendations on "fine tuning" of Lustre file storage in high-performance computing systems of scientific organizations when solving applied problems of various types.

The basic approach for configuration parallel data storage is the approach in which the user creates his own structure of files and directories to store files of various sizes and maps certain storage patterns to these directories that describe the number and size of stripes. Tools for such configuration are provided to the user in the form of executable scripts.

Users have the ability to customize the size and number of stripes for any file they use or create during a computational job. Determining the best settings sometimes requires experimentation, but there are general rules of thumb.

Let us consider the basic process of writing a file to a Lustre parallel file system as an example.

Let us assume that 200 MB is to be written to a file that is created with 10 stripes and a 1 MB stripe. When the file is initially recorded, 10 1 MB blocks will be simultaneously written to 10 different OSTs. Once those 10 blocks are full, Lustre writes another 10 1 MB blocks to those 10 OSTs. This process is repeated a total of 20 times until the entire file is written. When completed, the file will exist as 20 1 MB data blocks in each of 10 separate OSTs.

One of the key factors behind the high performance of Lustre file systems is the ability to stripe data across multiple storage targets (OSTs) in a round robin fashion. Essentially, files can be divided into multiple fragments, which are then stored in different OSTs in the Lustre system.

Larger files benefit from more stripes. By distributing a large file across multiple OSTs, the system's throughput for accessing the file increases, which improves efficiency when multiple processes are working on the same file in parallel. Conversely, a very large file that is interleaved in only one or two OSTs can degrade the performance of the entire Lustre system due to unnecessary OST padding. It is good practice to have dedicated directories with lots of stripes for writing very large files to.

Note that the following method of dividing a file into stripes should be avoided: dividing small files with a large number of stripes. This can negatively impact performance due to the unnecessary overhead of communicating with multiple OSTs. In this case, it is good practice to write small files to a directory with stripes equal to 1, i.e., without striping.

Thus, to configure striping, its main advantages and disadvantages should be considered:

- Advantage 1: increased throughput because multiple processes can access the same file at the same time;
- Advantage 2: the ability to store large files that take up more space than one OST;
- Disadvantage 1: increased overhead due to network operations and server conflicts;
- Disadvantage 2: increased risk of file corruption due to hardware failure.

Methods for improving reliability are the use of RAID of various levels in data storage nodes (OSS). At the same time, RAID can be implemented both by means of storage systems that provide block disk devices to storage nodes, and by means of RAID controllers located directly on storage nodes. Both methods improve the reliability and fault tolerance of the parallel file system. At the same time, as the experience of using such configurations in the CKP "Informatics" shows, the bandwidth for accessing drives increases compared to accessing single disks due to parallel access to drives.

A separate task is to ensure the reliability and fault tolerance of NVMe SSD drives connected directly to the PCI bus. The absence of a RAID controller in this case makes it necessary to take special measures to build fault-tolerant arrays. One such measure is the use of software RAID by means of the operating system. Another method is to use specialized storage systems that include NVMe SSD drives and RAID controllers that combine these drives into arrays of various levels.

Note that due to the use of specialized controllers, creating RAID using storage systems is more productive than using software RAID.

Experiments in the CKP "Informatics" using Huawei OceanStore Dorado 3000 v6 storage systems showed that the speed of access to block devices on RAID5 based on NVMe SSD, provided via the iSCSI interface over a 100 Gb Ethernet network, is about 2 GBps. The access speed to a similar SSD NVMe drive installed in the OSS server is about 1.2–1.4 GBps.

Thus, the use of high-speed storage systems equipped with SSD NVMe drives and RAID controllers as OST allows you to simultaneously increase the throughput of the Lustre parallel file system and improve reliability and fault tolerance.

5. Conclusions

A parallel file system is an integral part of a high-performance computing system designed to solve a wide range of scientific and scientific–practical problems. The parallel file system effectively prevents the formation of "bottlenecks" that reduce the performance of the HPC clusters due to delays in disk operations. Its application in the shared research facilities, providing high-performance computing services, allows to provide sufficient throughput of the disk subsystem for all scientific tasks performed in parallel by the HPC cluster.

The tasks of processing big data, extracting knowledge, and mathematical modeling impose different requirements on the organization of data exchange with disk storage when performed on a group of computing nodes.

At the same time, in order to optimize and increase the efficiency of the functioning of the HPC cluster as a whole, it is necessary to take measures to adapt data storage patterns in a parallel file system to the specifics of the applied problems being solved.

An analysis of computational tasks from various fields of science and technology shows that the requirements for the throughput of access to the file system for various applied tasks differ significantly. So the tasks of optimization, quantum mechanical calculations, aerodynamics do not require significant resources of the file system while loading the computing unit.

The tasks of training neural networks are more demanding on the performance of the file system, which ensures the timely loading of training data on computing nodes, but the ratio between data volumes and calculations is not the maximum.

The greatest requirements for the performance of the file system are observed in the tasks of preparing data for training neural networks and in the tasks of extracting knowledge (data mining). These tasks require a high-speed exchange of large data arrays between a large number of data storage and processing nodes.

Depending on the task type and file storage characteristics, a data storage template is selected. In general, maximizing file system performance requires maximizing the number of storage objects used.

Thus, when deploying projects designed to solve computing problems that require high disk efficiency, you should allocate the maximum number of storage elements, mainly NVMe SSD.

For tasks with medium and low disk intensity, it is enough to allocate a small number of SAS or SATA storage elements.

Ensuring reliability and fault tolerance through the use of a redundant storage architecture is provided on data object storage servers or storage systems.

Author Contributions: Conceptualization, S.D., K.V. and A.Z.; methodology, S.D., K.V. and A.Z.; validation, S.D.; formal analysis, K.V.; investigation, S.D., K.V. and A.Z.; writing—original draft preparation, S.D. and K.V.; writing—review and editing, S.D. and K.V.; supervision, A.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The research was carried out using the infrastructure of the Shared Research Facilities "High Performance Computing and Big Data" (CKP "Informatics") of FRC CSC RAS (Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, Moscow).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zatsarinnyy, A.A.; Abgaryan, K.K. Current problems of creation of research infrastructure for synthesis of new materials in the framework of the digital transformation of society. In Proceedings of the II International Conference Mathematical Modeling in Materials Science of Electronic Components, Online, 19–20 October 2020; pp. 8–13.
2. Zatsarinny, A.A.; Volovich, K.I.; Denisov, S.A.; Ionenkov, Y.S.; Kondrashev, V.A. Methodological approaches to evaluating the effectiveness of the center collective use "Informatics". *Highly Available Syst.* **2020**, *16*, 44–51.
3. Volovich, K.; Zatsarinnyy, A.; Frenkel, S.; Denisov, S. High Performance Computing in a Shared Virtual Infrastructure. In Proceedings of the VI International Conference on Information Technologies and High-Performance Computing (ITHPC 2021), Khabarovsk, Russia, 14–16 September 2021; Volume 2930, pp. 38–46.
4. Kartsev, A.; Malkovsky, S.; Volovich, K.; Sorokin, A. Study of the performance and scalability of the Quantum ESPRESSO package in the study of low-dimensional systems on hybrid computing systems. In Proceedings of the I International Conference Mathematical Modeling in Materials Science of Electronic Components, Moscow, Russia, 21–23 October 2019; pp. 18–21.
5. Abgaryan, K.K. Information technology is the construction of multi-scale models in problems of computational materials science. *Highly Available Syst.* **2018**, *14*, 9–15.
6. Abgaryan, K.K.; Gavrilov, E.S.; Marasanov, A.M. Multiscale modeling for composite materials computer simulation support. *Int. J. Open Inf. Technol.* **2017**, *5*, 24–28.
7. Kokorev, A.; Belyakov, D.; Lyubimova, M. Data storage systems of "hybrilit" heterogeneous computing platform for scientific research carried out in JINR: Filesystems and raids performance research CEUR Workshop Proceedings. In Proceedings of the 9th International Conference "Distributed Computing and Grid Technologies in Science and Education" (GRID'2021), Dubna, Russia, 5–9 July 2021; Volume 3041, pp. 296–303.
8. Seiz, M.; Offenhäuser, P.; Andersson, S.; Hötzer, J.; Hierl, H.; Nestler, B.; Resch, M. Lustre I/O performance investigations on Hazel Hen: Experiments and heuristics. *J. Supercomput.* **2021**, *77*, 12508–12536. [CrossRef]
9. Tipu, A.J.S.; Conbhuí, P.Ó.; Howley, E. Applying neural networks to predict HPC-I/O bandwidth over seismic data on lustre file system for ExSeisDat. *Clust. Comput.* **2022**, *25*, 2661–2682. [CrossRef] [PubMed]
10. Rybintsev, V. Optimizing the parameters of the Lustre-file-system-based HPC system for reverse time migration. *J. Supercomput.* **2020**, *76*, 536–548. [CrossRef]
11. Volovich, K. Estimation of the workload of a hybrid computing cluster in tasks of modeling in materials science. In Proceedings of the II International Conference Mathematical Modeling in Materials Science of Electronic Components, Online, 19–20 October 2020; pp. 30–33.