DOI: 10.15827/2311-6749.19.3.3

METHODS AND MEANS OF DISTRIBUTED STORAGE SYSTEMS IMPLEMENTATION

A.V. Baranov [1]*, PhD (Engineering) Associate Professor*
S.A. Leshchev [1]*, Research Associate*

[1] *Joint Supercomputer Center of the Russian Academy of Sciences – branch of Federal State Institution «Scientific Research Institute for System Analysis of the Russian Academy of Sciences», Leninsky Ave., 32a, Moscow, 119334, Russian Federation*

One of key methods to increase the efficiency of computing resources utilization in supercomputer centers is to link them into a distributed network. For user jobs and data to be able to migrate in a supercomputer center network, there should be a distributed storage system providing a single file space for all centers.

The paper considers the existing methods and means of implementing distributed storage systems and analyzes their applicability to a supercomputer center network. It states the requirements to a distributed storage system for different aspects of its development and use: data and its metadata storage, user job life cycle support, and data security. Additionally, the paper defines possible options and techniques of implementing stated requirements into a distributed storage system of supercomputer center network.

Keywords: *high-performance computing, supercomputer center, distributed computing, distributed storage system, supercomputer job management.*

The project of a supercomputer centers (SC) distributed network is being developed in the Joint supercomputer center of the RAS. The SC distributed network is being created as a part of the digital scientific infrastructure for high-performance computing. [1]. The project implies integration of computational resources of geographically distributed SC into a single network for the benefit of scientific and higher education organizations as well as innovation organizations of the Russian Federation. The aims of such integration are to optimize computational load distribution, to simplify and improve the quality of user access to high-performance computing resources and to consolidate the functions of distributed supercomputer resources monitoring and management.

The SC distributed network project includes the development of the following systems:

- job management system [2] based on a decentralized scheme of interaction between a team of equal dispatchers;
- unified system for monitoring the state of SC network computing resources;
- unified system of user access based on the principles of federated authentication [3];
- shared distributed storage system (DSS), providing the single file space for all SCs in the network.

The paper is devoted to the study of existing methods and techniques of DSS implementation with the purpose of building a shared DSS for the created SC distributed network. Alongside with single file space, the developed DSS shall meet the requirements for the compliance with the remaining components of the SC distributed network project, primarily with the requirements of the job management system.

In terms of a management system the minimal resource unit in the SC network is a computing facility (CF), which typically means a cluster architecture supercomputer [4]. The main component of such supercomputer is the set of computing nodes (CN), united by a high-speed low-latency communication network. To access the cluster and carry out high-performance calculations, a user must prepare a computing job including an application program, resource requirements and input data. The user forms a job in a special information object (usually a set format file), called the job passport. Job passport typically includes the following information:

1) unique identifiers of the job and its owner;
2) resource requirements – the number of CNs (processor cores), the amount of RAM memory and disk space, hardware requirements for CNs (the presence and type of graphics accelerator, processor type, the minimum required amount of memory per node, etc.) and job execution time;
3) job initial data and information regarding its start on the CF (the prologue and epilogue command scripts) as well as the reference to the job body that can be either an ordinary executable or command file, or a container with the software environment required to start the job.

Every CF in the SC network is controlled by its local job management system (for example, SLURM, PBS, LSF), which distributes incoming jobs between CNs. With this purpose the local management system maintains its local job queue. Job management system of the SC network distributes jobs between CFs and maintains the global job queue [2, 5]. The team of equal dispatchers processes the global job queue. The dispatchers are situated on every CF and make a coordinated decision on job distribution between CFs. In this case, the facility that places the job in the global queue is called the «customer» and the facility to which the job is distributed for execution is called the «contractor».

The global queue jobs are stored in the information subsystem based on the distributed non-relational database management system (DDBMS) Elasticsearch [6]. The paper [2] stipulates that it is recommended to save documents with a predetermined structure in Elasticsearch DDBMS and place them in different indices. In the information subsystem of the SC network, Elasticsearch DDBMS indexes are used to store information about SC network configuration, user accounts, and jobs passports of the global queue. To transfer input and output data files of the job, data are copied between dispatchers of CFs because Elasticsearch DDBMS is document-oriented and is not intended for storing files.

In order to ensure scalability and performance of the SC network job management system, the design and implementation of the information subsystem complied with the limitations defined in the REST [7] architecture, with some exceptions. For example, in case the customer dispatcher that placed a job passport in a global queue is unavailable at the time of the job startup for some reason, the job input data files become unavailable as well. Formally, the requirement of the REST architecture (all client's requests should be prepared so that the server receives necessary information to fulfill a request) will be met at the time the job is queued. When placing a job in a global queue, the reference to the job input data location is sent. However, this data is not being enough when the job is being fulfilled by the contractor at the moment of either the dispatcher or customer communication channel malfunction. It is possible to avoid such situation by creating backup copies in the SC network: backup copies of the job input data when it is queued at the customer and backup copies of the job output data when it is over at the contractor. For these purposes, it is proposed to use DSS with its elements located on the disk space allocated by the supercomputer centers participating in the project.

The purpose of this paper is to analyze architectural peculiarities of the required DSS and to determine the requirements for it enabling further selection and development of DSS implementation methods and means.

Data Storage in a Supercomputer Center

Data storage in supercomputer centers is organized on a hierarchical basis [1]. As in other data centers, it is possible to differentiate three levels of storage, as shown in the Figure 1.

Information of the first level represents the so called "hot" data – intermediate results of calculations (temporary files), placed on local high-speed storage devices of the supercomputer nodes. This level is often considered to include arrays on fast disks (SAS), Intel Optane SSD [8] disks or modern flash-memory modules with the NVMe [9] interface of relatively small capacity, but with high speed and I/O performance. In such arrays it is possible to place checkpoints of jobs, temporary aborted and returned to the queue.

The second (the main) level is comprised by different block devices providing both long-term storage of user data and real-time access to it.

The third level is usually represented by tape libraries and is used for long-term storage of the so called "cold" (rarely used and/or archived) data. It should be noted that there is a tendency to blur the boundaries between traditional storage levels today. Thus, an increase in the areal density on SATA disks with a simultaneous decrease in their cost led to the fact that the cost of storing even "cold" data on disk arrays is almost the same as the cost of tape library storage. The decrease in the cost of solid state drives gradually led to the fact that they began to replace SAS disks in disk arrays of the second storage level. In general, SC can implement both all listed storage levels as well as only some of them.

The paper [10] discusses the evolution of supercomputer center storage systems and shows that due to the increase in computing nodes location density and their subsequent miniaturization, the external memory devices were also moved beyond the CF perimeter alongside with power and cooling subsystems. Thus, external (relative to computing nodes and management servers) storage systems of the second and the third levels became typical for the SC structure represented by:

- file servers with the file system both on their own block devices (HDD) and on external direct-attached disk shelves (DAS), as well as on logical disks (LUN) in the enterprise storage network (SAN) (/home1 … /home3 in the Figure 2);
- network-attached storages (NAS) – /home4 in the Figure 2;
- parallel file systems (Lustre, GPFS and others); – /home5 in the Figure 2;
- long term storages based on tape libraries – /home6 in the Figure 2.

One of peculiarities of working with SC data is massive parallel file reading/writing with simultaneous access to these files from many different computing nodes. Technically, such access can be implemented either using file protocols, or using parallel file systems clients embedded in node operating systems that use their own specialized data access protocols over a local network. As a result, CF nodes and control servers usually mount into their file system one or more volumes with project directories exported by storage devices as is shown in the Figure 2.

It is known that the main supercomputer operating system (OS) is Linux of various distributions. Based on IO-500 benchmark results [11], which appeared in 2017, it is possible to point out the following. The most part of the list are parallel file systems (Lustre, Spectrum scale, BeeGFS, GPFS, CephFS). All other items in the list are various versions of the NFS file protocol or NAS systems that export data volumes using the NFS protocol.

The NFS version 3 protocol appeared in June 1995 [12]. Although it has substantial security vulnerability (the procedure of authorizing user access to data is delegated to a client device), nevertheless it is still used. In December 2000, the version 4 protocol appeared [13], allowing to implement the user authorization mechanism on the server side as well as extended access control lists (Access Control List, ACL) that have more capabilities than ACL POSIX [14]. In 2010, Parallel NFS specification was introduced into the NFS v4.1 [15]. It allowed placing different files of the same network directory in different storage servers in the way transparent for a user, thereby making access to these files parallel. IO-500 [11] list analysis shows that Parallel NFS was not widely distributed due to the widespread use of other parallel file systems.

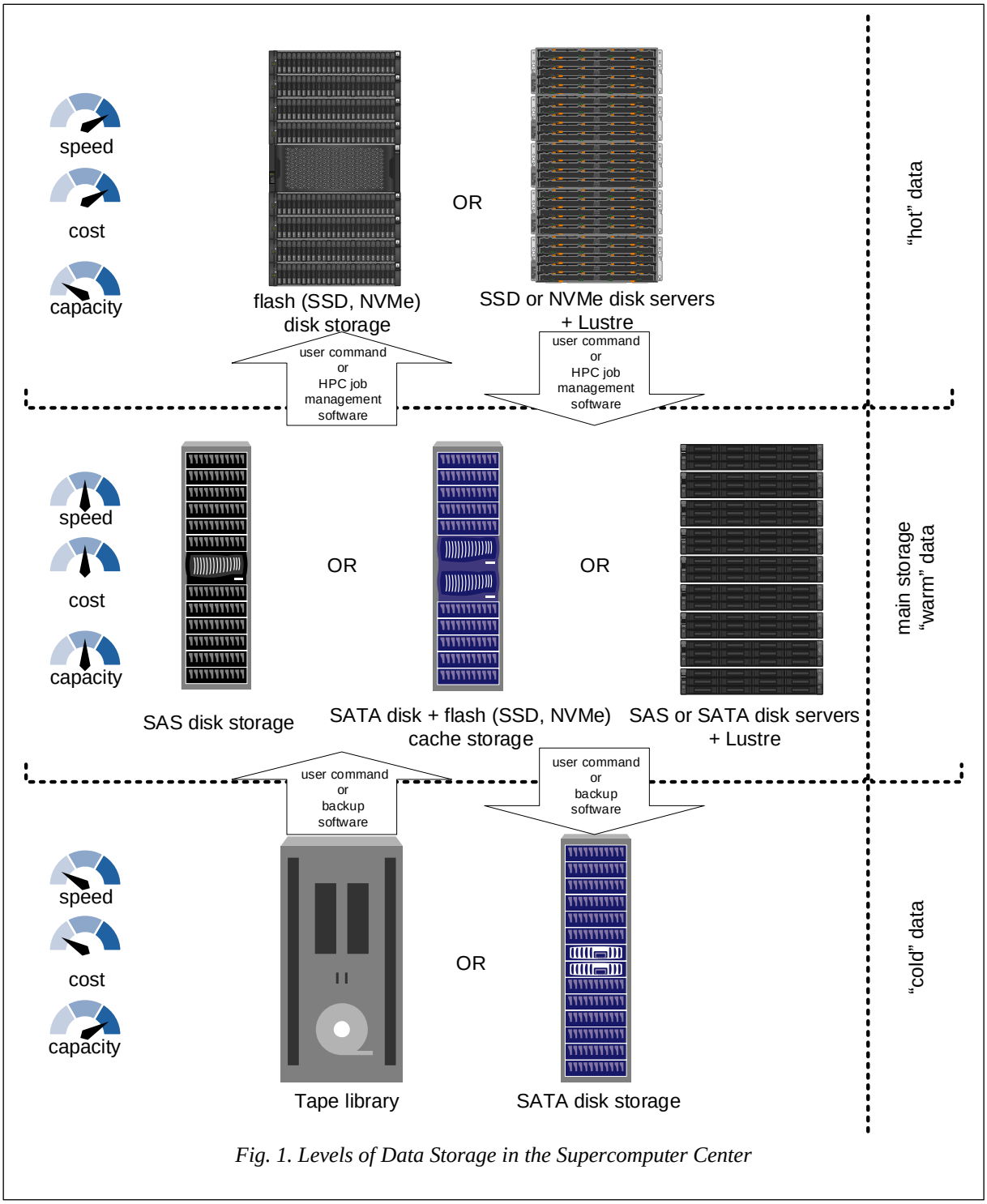

Fig. 1. Levels of Data Storage in the Supercomputer Center

The NFS protocol substantial feature is that it is abstracted from both server and client file system types. NFS access makes it possible to avoid making changes in SC file systems when implementing the DSS. Thus, we can formulate the first requirements for the created DSS:

1) implementation of all of its functioning components in Linux OS;
2) access to the stored data using the NFS version 3 and 4 protocols.

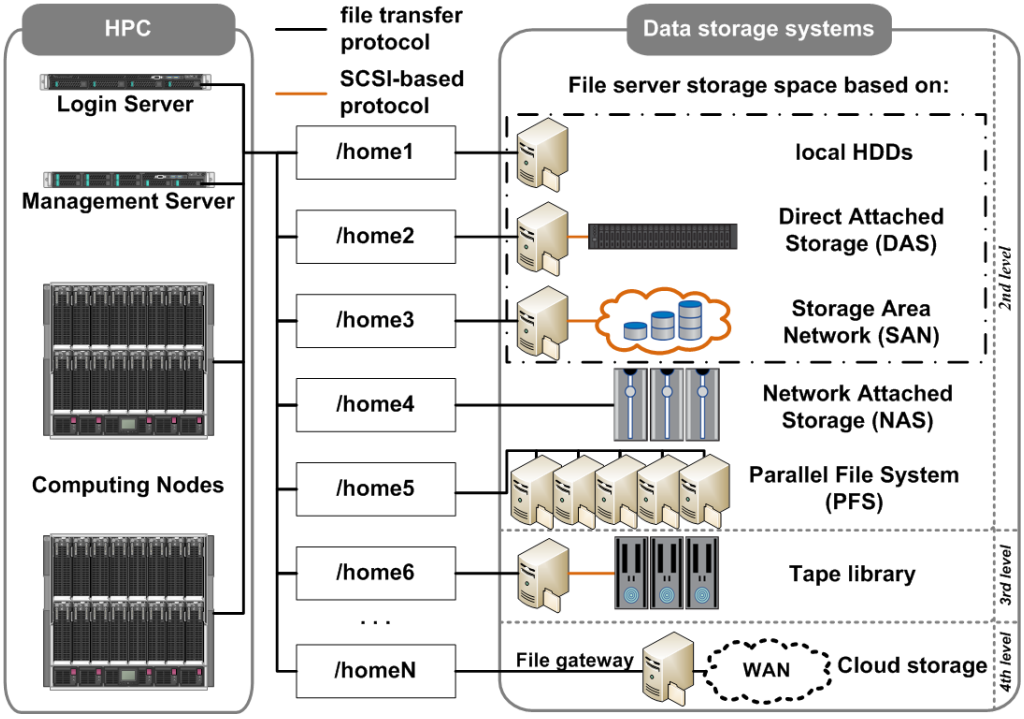

Fig. 2. DSS in a Supercomputer Center

Job Data Life Cycle in the SC Network

The main events of a computing job are its submission, startup, completion, deletion from the queue without start [16]. These events correspond to the main life cycle phases of the job placed into the global queue. In turn, three basic data operations correspond to the job life cycle:

- job data placement (input data – at the customer side, output data – at the contractor side);
- job data accessing (input data – at the contractor side, output data – at the customer side);
- job data deletion (after job completion or deletion from the queue).

Apparently, to fulfill a job in the SC network its data shall be transferred between participating CFs in both directions. In the case of transferring the output data, the contractor and customer are already known, and the data transmission may be started at the job completion moment. The delay is determined by parameters of communication channels between CFs and will not be critical for the job execution process.

The input data access delay during the job startup on the contractor leads to ineffective use of CF resources and unreasonable spending of user machine time. Let us consider possible input data access options.

1) Data transmission begins at the moment of accessing data on the contractor. This method is used when an another way implementation is impossible since the delay in this case is maximal and is determined by the customer and contractor communication channel speed.

2) Data transmission begins at the moment the job is submitted to the local contractor's queue. To transfer data this method uses job waiting time in the local queue [2]. The job data transfer is impossible during the job waiting time in the global queue because the contractor is not yet defined at the job submitting moment.

3) Data transmission begins at the moment the job is placed in the global queue; possible destinations are determined using proactive data prefetching algorithms.

Let us consider the last method. One of the first papers regarding disk data prefetching in RAM memory appeared in 1993 [17]. As geographically distributed networks evolve, it was proposed to use intelligent prefetching strategy that can recognize the current access pattern by analyzing the access trace [18], to use the methods of cluster [19] and statistical [20] analysis of file access history. Due to the limited bandwidth of communication channels in WAN networks, there were the attempts to combine prefetching algorithms according to data access patterns with on-demand fetching algorithms [21, 22]. Algorithms based on a decentralized approach for data prefetching in desktop grid network for Bag-Of-Tasks and DAG applications were proposed [23]. Best known

is the paper [24], in which Iranian scientists have developed the idea [25] that the members in a virtual organization of the distributed network have similar interests in files and membership in a virtual organization can determine a data access pattern and predict access to the files.

Data prefetching may be used in the developed DSS to reduce the delay of accessing the job input data. Prefetching algorithms in distributed computing networks rest predominantly upon the methods for recognition of data access [26] patterns based on file access history. One of distinctive peculiarities of the prefetching algorithm proposed for DSS is the development of a forecast model based on accumulated information regarding distribution of jobs from the global queue. The forecast model is dynamically adjusted based on the real time input stream of processed jobs. The output of such model is a probability of assigning the job to one or another contractor. As DSS fault tolerance engine uses the data backup principle, it is possible to select such number of simultaneously created backups, so that the job input data copy had already been on the contractor DSS site at the job launching time with the required probability.

Maintaining the User Job Life Cycle in DSS

In accordance with [1] DSS unites the resources provided by supercomputer centers participating in the project. Private cloud storages use such approach to store and access data. To ensure scalability the cloud storage software is designed using the principle of modularity, when one or another type of service is implemented as a separate daemon or server. Metadata servers, data servers, backup servers, user authorization servers and others are implemented separately in cloud storages alongside with gateways for various data access protocols. Part of SC software and hardware resources, which DSS daemons and servers are located, is called the DSS site.

DSS sites are territorially located in different SC and use special data transfer protocols (S3, WebDAW, SMB 3.0, HTTPS, etc.) to interact via WAN. SC storage devices of the second level are located in the local SC network. In order to preserve the existing SC architecture of user data access, the so-called cloud storage gateway is organized in the local SC network. This is a dedicated server interacting with DSS infrastructure via WAN-protocols or via special application programing interface (API) and simultaneously exporting DSS file space as a separate volume with project directories (/homeN in the Figure 2) to SC servers via NFS.

To ensure fault tolerance of cloud storage, the information placed on metadata servers due to its small (relative to stored data) size is usually replicated between metadata servers of all DSS sites. Backup dispatchers automatically create backups for the data on storage servers of several sites. The main backup is on the site where data were placed into DSS.

Let us consider work peculiarities of cloud storage components during the job life cycle.

1) During the job placement phase (when the job enters the system) its input data are placed in DSS by the customer dispatcher; at the same time data files are copied through the cloud storage gateway to the data server and metadata are recorded to the metadata server of the customer local DSS site.

2) During the job startup phase the contractor dispatcher, which selected the job from the global queue, accesses the necessary data through the cloud storage gateway that finds the data server with the main copy of the required files based on the local metadata server information and copies data via WAN to computing nodes assigned for the job.

3) During the job completion phase the contractor dispatcher places job output data to the DSS. Output data placing into DSS is similar to input data placing into the global queue, however the placing CF changes. After output data is placed into the DSS, the customer dispatcher copies it from DSS to the job results directory (the process is similar to accessing the input job data by contractor at the job startup phase, i.e. goes via WAN). Then it removes all job data from DSS through the cloud storage gateway. When the job data are removed from DSS, corresponding metadata changes (files and their backups are marked for deletion) are saved to the DSS site metadata server of the nearest to customer. After metadata replication, backup dispatchers of all DSS sites that contain copies of removed files delete them from their data servers.

4) During the phase of removing a job from the queue, the customer dispatcher deletes job data files; this time its actions are similar to the last operation of the job completion phase.

It is easy to see that in terms of speed, the worst case is data access via WAN (shown by dotted arrows in the Figure 3); the best one is if by the beginning of the data access operation one of their copies is placed on the CF site data server, from where the data is accessed, and this copy is marked as the main one in the metadata.

To implement the best case the backup dispatcher should create backups on those CF sites that most likely will contain the job selected from the global queue. In addition, the metadata server shall correspondingly change the data main copy location to contractor DSS site at the moment of job selection by contractor. As a result, only local copies of meta- and file data relative to SC will be used at every phase of accessing the files placed in the DSS. These copies are actually located on the storage space of the SC second storage level, and it is possible to state that the access speed and the speed of working with the SC DSS will be almost identical. In this case, one more volume with project directories (/homeN in the Figure 2) appears in the file system of supercomputer's CN as DSS data representation. The work with global queue job data is organized in the similar way as with local queue job data (/home1 … /home4 in the Figure 2). The job input data placing and job output data

getting by CF dispatcher is simply file copying in this case. The necessity to copy data between different volumes disappears if storage systems have enough disk space.

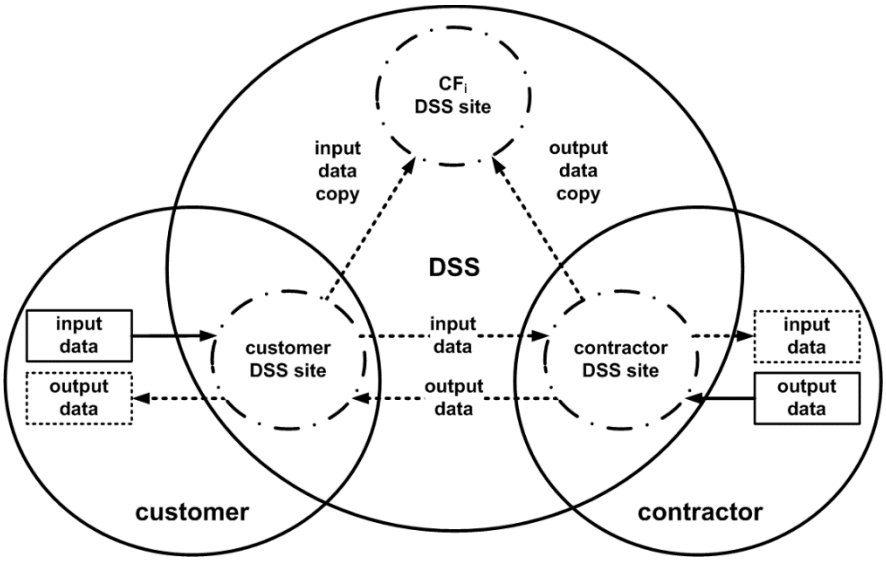

Fig. 3. Job Data Life Cycle in DSS

It is worth mentioning that job data at every life cycle stage are used on one CF only. This leads to the following conclusions:

- since data is processed at each phase exclusively on one CF, there is no need to solve the problem of distributed locks in the DSS;
- the issue of multi-user parallel data access at the CF level is resolved by the NFS protocol mechanisms.

Thus, it is possible to distinguish the following peculiarities of the developed DSS architecture:

1) every SC has a fully functional DSS site including a cloud storage gateway, metadata server, data server, backup server and other necessary components;
2) the cloud storage gateway interacts with local data and metadata servers only;
3) the backup manager either implements API to manage backup creation, or its backup allocation algorithms are modified accordingly;
4) the metadata server either implements API to change the main data copy location, or its algorithms that manage location of the main data copy are modified accordingly;
5) the cloud storage gateway provides the NFS file protocol access interface of versions 3 and 4 to the storage space of DSS in every SC.

Adding API for external control of DSS components or modifying their working algorithms requires changing their program code. Free program code modification is impossible in proprietary DSSs, therefore its components should be distributed under anyone of the free software licenses (GNU, Apache, BSD, etc.). This is another requirement to the developed DSS.

Data Security in DSS

When a management scheme is decentralized, all SCs are independent and have equal rights and capabilities in the distributed network. Every SC has its own user account database and user authorization system. Administrators of different SCs cannot and should not give administrative rights to each other. A job fetched from the global queue can migrate from one SC, where customer is located, to the other, where contractor is located. When migrating a job, it is necessary to match user accounts in different SCs. In the paper [2] proposes to use the global catalog of user identifiers with a table of correspondences to their local identifiers in each SC. At the same time files and directories moved between different SCs accordingly change owner and group identifiers.

Let us consider possible variants of user authentication in SC in detail.

1. Local files on the server. Information necessary for user authentication is stored in special files on the server or computing node of CF. This method is used if it is impossible to apply other authentication methods.

2. CF authentication center. Service and information necessary for user authentication are transferred to a specialized server (a group of servers), for example, LDAP or Active Directory, to create a dedicated center for user authentication and management.

3. SC authentication center. All CF of SC use authentication center from the second variant. This is the main variant of user authentication used in SC today.

4. Federated identity-based authentication [27]. User accounts are stored in the organization in which they are employed. When an attempt is made to access the SC network resources, authorization request is sent to the organization employing the user and if the answer is positive access is granted.

It is suggested to use federated identity-based authentication for SC network and to provide a unified access system based on identity federation [3]. However, creation of SC network identity federation requires raising and solving a number of organizational and legal issues alongside with addressing scientific and technical challenges. According to [2] it is suggested to use SC authentication centers, before the mechanisms of federated authentication are implemented and as a fallback variant in case of failures.

Paper [3] shows that in a distributed SC network it is necessary to provide for mapping some global user identifier to local identifiers in each of the SCs. It is reasonable to assume that the consequence of mapping a global user to a local user in SC will be the need to implement for each SC its own view of owner and group data, access privileges and/or access control lists (ACLs) for files and directories placed in the DSS. This corresponds to discretionary access control feature of Linux [28], which provides for the ability to store multiple ACLs. As the scope of a file or directory placed in the DSS generally extends to several SCs, there are three options for managing data access restrictions.

1) In the SC, there is a local user account to access data. A specially reserved service identifier for the group and a local account identifier for the owner are used for the files.

2) In the SC, there is a local user account to access data. A specially reserved service identifiers for the group and the owner are used for the files. Extended access rights (additional ACLs) are used for a local account.

3) In the SC, there is the depersonalization mechanism to access data. It includes artificial group and user identifiers from the previously reserved pool of identifiers that are mapped a global user identifier during the job duration.

With relation to implementation of data access rights control mechanisms in DSS, this will mean that DSS will require either API to access the global user identifier storage in DDBMS [2], or a mechanism for storing global user identifiers and mapping local user identifiers in various SCs to its global identifier. DSS will also require its own Role Based Access Control system (RBAC) [29] that allows creating global security groups for user directories placed in DSS storage space, mapping user security groups from authentication centers of various SCs and assigning corresponding ACLs to objects stored. DSS must support integration with existing user accounts authentication and storage mechanisms used in SC (for example, LDAP and Kerberos). Federated authentication requires DSS to be able to implement its own authentication proxy server that supports SAML and OpenID standards (https://digital.gov.ru/uploaded/presentations/esiametodicheskierekomendatsii258.pdf).

From the security point of view, every user should have access only to his own data and his workgroup (several workgroups, if he participates in several projects) data, without receiving any direct or indirect information about the data of other users and project groups. SC user in storage systems of the second storage level is limited in access to other people's data by access control mechanisms available in Linux OS and NFS protocols. However, he is not limited in access to the list of project directories (as well as to account data of their owners). If we assume that DSS implements the single file namespace, as it is suggested in [1], then the scope of foreign project directories for a third-party user and the degree of security breach increase in proportion to the number of SCs in the network. In the data access architecture at the fourth storage level as opposed to the second (see Fig. 2), a cloud storage gateway appears. Its means make it possible and necessary to implement visibility restrictions in SC project directories. Those objects should be visible, the ACLs of which contain local user accounts of this SC, i.e. a feature should be implemented that is similar to Access-Based Enumeration in MS Windows Server systems [30].

Storing several copies of an object (file) in DSS storage nodes located in different SCs raises the problem of potential unauthorized access to users' data by administrators of DSS storage nodes. Currently DSS mainly uses two mechanisms to solve this problem: stored data encryption or information dispersal algorithms [31] between DSS nodes. As shown above, a peculiarity of the job data life cycle is that at each cycle phase the customer and the contractor work with a local copy of the storage object (file) placed in the local storage of the DSS object node. Therefore, using information dispersal mechanisms instead of the backup mechanism will lead to a noticeable drop in DSS performance.

During DSS deployment or operation, the disk space allocated of the SC for its elements may be depleted. If it is impossible to promptly allocate additional disk space, the DSS shall be capable of using the disk space, provided by commercial clouds, encrypting data allocated there.

Having considered the listed security aspects of processing data in DSS, we can formulate the following requirements to DSS:

1) implementation of own user identifiers storage and role based access control mechanism (RBAC);
2) implementation of the own mechanism of data access restrictions representation for every SC;
3) LDAP and Kerberos support;

4) implementation of own authentication proxy servers with SAML and OpenID standard support;
5) ensure that in every SC only those data are visible that the users of this SC have access rights to;
6) support for integration with existing commercial cloud storages;
7) support for encryption of storage object copies, placed on DSS sites and third-party cloud storages.

Metadata in DSS

The load and availability of WAN channels change dynamically depending on the time of day, communication provider failures and a number of other events on the Web. The load and availability of CFs are also influenced by a variety of factors. To implement data prefetching under these conditions correctly, we need to collect statistic data on a real time basis regarding data transfer channels load and distribution of jobs from the global queue sending the statistics to the forecast model input to correct its work.

Alongside with data for a forecast model, to further research and develop the placing data algorithms in DSS it is necessary to store service information related to the job life cycle: time stamps of start and finish of copying job files to the DSS, start and finish of replicating data between DSS sites, information about modifications to the index of the main data copy location, etc. In order the DSS could operate properly, it is also necessary to store information on access restrictions and the scope of files and directories for every SC. The engine of federated authentication may also require storing the service metadata for files and directories of DSS.

In 2002, support for extended file attributes (xattr) first introduced in the POSIX.1e standard was added to the Linux kernel [32]. Extended file attributes allow users to associate computer files with metadata not interpreted by the file system. Extended attributes are “name:value” pairs that are permanently associated with files and directories, just as environment variables are associated with a process. Extended attributes can be used to store a time stamp, hash, digital certificate, the type of data stored in the file, a XML document schema, additional information of discretionary access control and any other arbitrary data.

The metadata storage principle, similar to the one used in extended file attributes (“name:value” pairs), is used in object-oriented storage systems. The main approach to design such systems is to store data in the form of objects and associated metadata, while the metadata of objects is used to identify, represent and manage them.

Let us remind that the developed DSS needs no implementation of distributed locks (i.e. files on data servers can be considered as binary objects), but at the same time it is necessary to implement specific representations of access restrictions and visibility scope of the data placed in the DSS for every SC. These requirements to DSS correspond to the concept of object-oriented storage, i.e. the developed DSS should have the properties not only of a cloud, but also of an object-oriented storage system.

If we assume that a certain directory will correspond to a user job in a separate DSS section, then unique identifiers of the job and its owner, requirements for resources of CF and the names of the job launch scripts can be written as a set of attributes of this directory. The prologue and epilogue scripts and the job body will be written as objects within the job directory. Placing the job input and output files as objects in the nested subdirectories, we will get a job passport fully placed in the DSS.

As discussed previously, the own mechanism for storing global user identifiers and the role-based access control system replace the user account storage of the SC network, i.e. two of the three entities stored in the DDBMS (job passport and user account information) can be placed in the DSS. If the SC configuration record in the DDBMS corresponds to the DSS object and the record fields correspond to the extended attributes of the DSS object, then the information regarding SC network configuration can also be placed in a separate DSS section (similarly with placing the job passport in DSS). If further researches prove that metadata replication mechanisms are not inferior in speed and reliability of operation to similar DDBMS mechanisms, then the functions of DDBMS in the SC network can be transferred to DSS.

Conclusion

A distributed storage system of supercomputer centers network must support a user job life cycle and take into account several aspects of secure access to job data in a distributed infrastructure. Therefore, the present paper highlights architectural peculiarities of DSS implementation and formulates requirements to it. It is shown that the requirements are met by a cloud storage system that implements the functions of an object-oriented system. The immediate prospects for work are to study whether modern cloud storage systems are compliant with the formulated requirements and to select the best variant for practical DSS implementation with a further study of the selected storage system for speed and reliability of metadata replication mechanisms.

Acknowledgements. The paper was prepared according to the state assignment to conduct fundamental scientific research, topic #0065-2019-0016.

References

1. Shabanov B., Ovsiannikov A., Baranov A., Leshchev S., Dolgov B., Derbyshev D. The distributed network of the supercomputer centers for collaborative research. Program Systems: Theory and Applications, 2017, vol. 35, pp. 4, pp. 245–262 (in Russ.). DOI: 10.25209/2079-3316-2017-8-4-245-262.

2. Shabanov B.M., Telegin P.N., Ovsyannikov A.P., Baranov A.V., Lyakhovets D.S., Tikhomirov A.I. The Jobs Management System for the Distributed Network of the Supercomputer Centers. Proc. NIISI RAS. 2018, vol. 8, no. 6, pp. 65-73 (in Russ.). DOI: 10.25682/NIISI.2018.6.0009.

3. Baranov A.V., Ovsyannikov A.P., Shabanov B.M. Federative Identity for the Distributed Infrastructure of the Supercomputer Centers. Proc. NIISI RAS. 2018, vol. 8., no. 6, pp. 79-83 (in Russ.). DOI: 10.25682/NIISI.2018.6.0011.

4. Levin V.K. Supercomputing Trends. Computational Nanotechnology, 2014, no. 1, pp. 35-38 (in Russ.).

5. Tikhomirov A.I., Baranov A.V. Methods and tools for organizing the global job queue in the geographically distributed computing system. Bul. SUSU, Series Computational Mathematics and Software Engineering, 2017. Т. 6, № 4. С. 28–42 (in Russ.). DOI: 10.14529/cmse170403.

6. Krishan Kumar Singh, Mohit Kumar, Mayank Singhal, and Aashrita Dubey. Elasticsearch. IJMTER, vol. 5, no. 5, pp. 23-28. DOI: 10.21884/ijmter.2018.5137.2jz19.

7. Severance C., Roy T. Fielding: Understanding the REST Style. Computer, 2015, vol. 48, no. 6, pp. 7-9. DOI: 10.1109/MC.2015.170.

8. Hady F.T., Foong A., Veal B. and Williams D. Platform Storage Performance With 3D XPoint Technology. Proc. IEEE, 2017, vol. 105, no. 9, pp. 1822-1833. DOI: 10.1109/JPROC.2017.2731776.

9. Sivashankar and Ramasamy S. Design and implementation of non-volatile memory express. Proc. IEEE ICRTIT, 2014, pp. 1-6. DOI: 10.1109/ICRTIT.2014.6996190.

10. Leshchev S.A., Aladyshev O.S. Features of a Network of Data Storages for Supercomputer Center. Proc. NIISI RAS, 2017, vol. 7, no. 4, pp. 151-156 (in Russ.).

11. IO-500. Virtual Institute for I/O. URL: https://www.vi4io.org/io500/about/start (дата обращения: 27.06.2019).

12. Callaghan B., Pawlowski B., Staubach P. NFS Version 3 Protocol Specification. 1995. DOI: 10.17487/rfc1813.

13. Shepler S., Callaghan B., Robinson D., Thurlow R., Beame C., Eisler M., Noveck D. NFS version 4 Protocol. 2000. DOI: 10.17487/rfc3010.

14. Grunbacher A. POSIX Access Control Lists on Linux. Proc. USENIX Tech. Conf., 2003, pp. 259-272.

15. Shepler S., Eisler M., Noveck D. NFS Version 4 Minor Version 1 Protocol. 2010. DOI: 10.17487/rfc5661.

16. Baranov A.V., Kiselev T.F., Kormilitsin E.S., Ogaryshev V.F., Telegin P.N. Modification of the statistic subsystem of the Joint Supercomputer Center of the Russian Academy of Sciences. Proc. NIISI RAS, 2018, vol. 8, no. 4, pp. 136-144 (in Russ.). DOI: 10.25682/NIISI.2018.4.0016.

17. Curewitz K.M., Krishnan P. and Vitter J.S. Practical prefetching via data compression. Proc. ACM SIGMOD, NY, USA, 1993, pp. 257-266. DOI: 10.1145/170035.170077.

18. Jianxi Chen and Dan Feng. An intelligent prefetching strategy for a data grid prototype system. Proc. IEEE WCNM, 2005, pp. 1320-1323. DOI: 10.1109/WCNM.2005.1544298.

19. Liu G., Wei H., Wang X. and Peng W. Research on Data Interoperability Based on Clustering Analysis in Data Grid. Proc. IEEE I-ESA, 2009, pp. 97-103. DOI: 10.1109/I-ESA.2009.19.

20. Beigrezaei M., Haghighat A.T., Meybodi M.R. and Runiassy M. PPRA: A new pre-fetching and prediction based replication algorithm Meybodi in data grid. Proc. 6th ICCKE, 2016, pp. 257-262. DOI: 10.1109/ICCKE.2016.7802149.

21. Chen P., Chang J., Su Y. and Shieh C. On-demand data co-allocation with user-level cache for grids. Concurrency Computat.: Pract. Exper., 2010, vol. 22, iss. 18, pp. 2488-2513. DOI:10.1002/cpe.1587.

22. Po-Cheng Chen, Jyh-Biau Chang, Yi-Sheng Lin, Ce-Kuen Shieh, Yi-Chang Zhuang. Transparent on-demand co-allocation data access for grids. IJAHUC, 2010, vol. 5, no. 4, pp. 227-234. DOI: 10.1504/IJAHUC.2010.032997.

23. Saad W., Abbes H., Cérin C. and Jemni M. A Data Prefetching Model for Desktop Grids and the Condor Use Case. Proc. 12th IEEE TrustCom, 2013, pp. 1065-1072. DOI: 10.1109/TrustCom.2013.130.

24. Saadat N., Rahmani A.M. PDDRA: A new pre-fetching based dynamic data replication algorithm in data grids. Future Generation Computer Systems, 2012, vol. 28, iss. 4, pp. 666-681. DOI: 10.1016/j.future.2011.10.011.

25. Tian T., Luo J., Wu Z. and Song A. A Prefetching-based Replication Algorithm in Data Grid. Proc. 3rd ICPCA, 2008, pp. 526-531. DOI: 10.1109/ICPCA.2008.4783644.

26. Saharov I.E. Method of proactive caching for supplements realising at network environment of grid computing. VNIIZHT Bull., 2009, no. 2, pp. 29-33 (in Russ.).

27. Ovsyannikov A.P., Savin G.I., Shabanov B.M. Identity federation of the research and educational networks. Programmnye Produkty i Sistemy [Software and Systems], 2012, no. 4, pp. 3-7 (in Russ.).

28. Howells D. Credentials in Linux – The Linux Kernel documentation. 2017. Available at: https://www.kernel.org/doc/html/v4.14/security/credentials.html#credentials-in-linux (accessed: June 27, 2019).

29. Role-based access control. In: Access Control Systems. Springer, 2006, pp. 190-251. DOI: 10.1007/0-387-27716-1_8.

30. Access-based Enumeration. Microsoft Docs. Available at: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/dd772681(v=ws.10) (accessed: August 27, 2019).

31. Jones M. Anatomy of a cloud storage infrastructure. 2012. URL: https://developer.ibm.com/articles/cl-cloudstorage/ (дата обращения 27.06.2019).

32. Jonson L., Needham B. IEEE Std 1003.1e PSSG Draft 17. 1997. Available at: http://wt.tuxomania.net/topics/1999_06_Posix_1e/download/Posix_1003.1e-990310.pdf (accessed: 27.06.2019).

УДК 004.457+004.031.2+004.382.2
DOI: 10.15827/2311-6749.19.3.3

МЕТОДЫ И СРЕДСТВА ОРГАНИЗАЦИИ РАСПРЕДЕЛЕННЫХ СИСТЕМ ХРАНЕНИЯ ДАННЫХ

А.В. Баранов [1]*, к.т.н., доцент, antbar@mail.ru*
С.А. Лещев [1]*, bmserg@jscc.ru*

[1] *Межведомственный суперкомпьютерный центр Российской академии наук – филиал ФГУ «Федеральный научный центр Научно-исследовательский институт системных исследований Российской академии наук», Ленинский просп., 32а, г. Москва, 119334, Россия*

Одним из ключевых методов повышения эффективности использования вычислительных ресурсов суперкомпьютерных центров коллективного пользования (СКЦ) является их объединение в единую распределенную сеть. Для возможности миграции пользовательских заданий и данных в сети СКЦ должна быть предусмотрена распределенная система хранения данных (РСХД), обеспечивающая единое файловое пространство для всех центров. Работа авторов посвящена исследованию существующих методов и средств организации распределенных систем хранения данных и анализу их применимости для сети СКЦ. В статье сформулированы требования к РСХД для разных аспектов ее разработки и применения: организации хранения данных и их метаданных, поддержки жизненного цикла пользовательских заданий, обеспечения безопасности данных. Кроме этого, определены возможные варианты и средства реализации в создаваемой РСХД сети СКЦ сформулированных требований.

Ключевые слова*: высокопроизводительные вычисления, суперкомпьютерный центр коллективного пользования, распределенные вычисления, распределенные системы хранения данных, управление суперкомпьютерными заданиями.*

Благодарности. *Работа выполнена в рамках государственного задания по проведению фундаментальных научных исследований, тема № 0065-2019-0016.*

Литература

1. Шабанов Б. М., Овсянников А. П., Баранов А.В., Лещев С. А., Долгов Б.В., Дербышев Д.Ю. Проект распределенной сети суперкомпьютерных центров коллективного пользования // Программные системы: теория и приложения. 2017. Т. 35. №4, С. 245–262. DOI: 10.25209/2079-3316-2017-8-4-245-262.

2. Шабанов Б.М., Телегин П.Н., Овсянников А.П., Баранов А.В., Тихомиров А.И., Ляховец Д.С. Система управления заданиями распределенной сети суперкомпьютерных центров коллективного пользования // Труды НИИСИ РАН. 2018. Т. 8. № 6. С. 65-73. DOI: 10.25682/NIISI.2018.6.0009.

3. Баранов А.В., Овсянников А.П., Шабанов Б.М. Федеративная аутентификация в распределенной инфраструктуре суперкомпьютерных центров // Труды НИИСИ РАН. 2018. Т. 8. № 6. С. 79-83. DOI: 10.25682/NIISI.2018.6.0011.

4. Левин В.К. Тенденции развития суперкомпьютеров. Computational Nanotechnology. 2014. № 1. С. 35-38 (рус.).

5. Баранов А.В., Тихомиров А.И. Методы и средства организации глобальной очереди заданий в территориально распределенной вычислительной системе // Вестн. ЮУрГУ: Вычислительная математика и информатика. 2017. Т. 6. № 4. С. 28–42. DOI: 10.14529/cmse170403.

6. Krishan Kumar Singh, Mohit Kumar, Mayank Singhal, and Aashrita Dubey. Elasticsearch. IJMTER, vol. 5, no. 5, pp. 23-28. DOI: 10.21884/ijmter.2018.5137.2jz19.

7. Severance C., Roy T. Fielding: Understanding the REST Style. Computer, 2015, vol. 48, no. 6, pp. 7-9. DOI: 10.1109/MC.2015.170.

8. Hady F.T., Foong A., Veal B. and Williams D. Platform Storage Performance With 3D XPoint Technology. Proc. IEEE JPROC, 2017, vol. 105, no. 9, pp. 1822-1833. DOI: 10.1109/JPROC.2017.2731776.

9. Sivashankar and Ramasamy S. Design and implementation of non-volatile memory express. Proc. IEEE ICRTIT, 2014, pp. 1-6. DOI: 10.1109/ICRTIT.2014.6996190.

10. Лещев С.А., Аладышев О.С. Особенности современных сетей компьютерной памяти в суперкомпьютерных центрах // Труды НИИСИ РАН. 2017. Т. 7. № 4. С. 151-156.

11. IO-500. Virtual Institute for I/O. URL: https://www.vi4io.org/io500/about/start (дата обращения: 27.06.2019).

12. Callaghan B., Pawlowski B., Staubach P. NFS Version 3 Protocol Specification. 1995. DOI: 10.17487/rfc1813.

13. Shepler S., Callaghan B., Robinson D., Thurlow R., Beame C., Eisler M., Noveck D. NFS version 4 Protocol. 2000. DOI: 10.17487/rfc3010.

14. Grunbacher A. POSIX Access Control Lists on Linux. Proc. USENIX Tech. Conf., 2003, pp. 259-272.

15. Shepler S., Eisler M., Noveck D. NFS Version 4 Minor Version 1 Protocol. 2010. DOI: 10.17487/rfc5661.

16. Баранов А.В., Киселёв Е.А., Кормилицин Е.С., Огарышев В.Ф., Телегин П.Н. Модернизация подсистемы сбора и обработки статистики центра коллективного пользования вычислительными ресурсами МСЦ РАН // Труды НИИСИ РАН. 2018. Т. 8. № 4. С. 136-144. DOI: 10.25682/NIISI.2018.4.0016.

17. Curewitz K.M., Krishnan P. and Vitter J.S. Practical prefetching via data compression. Proc. ACM SIGMOD, NY, USA, 1993, pp. 257-266. DOI: 10.1145/170035.170077.

18. Jianxi Chen and Dan Feng. An intelligent prefetching strategy for a data grid prototype system. Proc. IEEE WCNM, 2005, pp. 1320-1323. DOI: 10.1109/WCNM.2005.1544298.

19. Liu G., Wei H., Wang X. and Peng W. Research on Data Interoperability Based on Clustering Analysis in Data Grid, Proc. IEEE I-ESA, 2009, pp. 97-103. DOI: 10.1109/I-ESA.2009.19.

20. Beigrezaei M., Haghighat A.T., Meybodi M.R. and Runiassy M. PPRA: A new pre-fetching and prediction based replication algorithm Meybodi in data grid. Proc. 6th ICCKE, 2016, pp. 257-262. DOI: 10.1109/ICCKE.2016.7802149.

21. Chen P., Chang J., Su Y. and Shieh C. On-demand data co-allocation with user-level cache for grids. Concurrency Computat.: Pract. Exper., 2010, vol. 22, iss. 18, pp. 2488-2513. DOI:10.1002/cpe.1587.

22. Po-Cheng Chen, Jyh-Biau Chang, Yi-Sheng Lin, Ce-Kuen Shieh, Yi-Chang Zhuang. Transparent on-demand co-allocation data access for grids. IJAHUC, 2010, vol. 5, no. 4, pp. 227-234. DOI: 10.1504/IJAHUC.2010.032997.

23. Saad W., Abbes H., Cérin C. and Jemni M. A Data Prefetching Model for Desktop Grids and the Condor Use Case. Proc. 12th IEEE TrustCom, 2013, pp. 1065-1072. DOI: 10.1109/TrustCom.2013.130.

24. Saadat N., Rahmani A.M. PDDRA: A new pre-fetching based dynamic data replication algorithm in data grids. Future Generation Computer Systems, 2012, vol. 28, iss. 4, pp. 666-681. DOI: 10.1016/j.future.2011.10.011.

25. Tian T., Luo J., Wu Z. and Song A. A Prefetching-based Replication Algorithm in Data Grid. Proc. 3rd ICPCA, 2008, pp. 526-531. DOI: 10.1109/ICPCA.2008.4783644.

26. Сахаров И.Е. Метод упреждающего кэширования для приложений, исполняющихся в сетевой среде распределенных вычислений // Вестн. ВНИИЖТ. 2009. № 2. С. 29-33.

27. Овсянников А.П., Савин Г.И., Шабанов Б.М. Удостоверяющие федерации научно-образовательных сетей // Программные продукты и системы. 2012. № 4. С. 3-7.

28. Howells D. Credentials in Linux – The Linux Kernel documentation. 2017. URL: https://www.kernel.org/doc/html/v4.14/security/credentials.html#credentials-in-linux (дата обращения: 27.06.2019).

29. Role-Based Access Control. In: Access Control Systems, pp. 190-251. Springer, 2006. DOI: 10.1007/0-387-27716-1_8

30. Access-based Enumeration. Microsoft Docs. URL: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/dd772681(v=ws.10) (дата обращения: 27.08.2019).

31. Джонс М. Анатомия облачной инфраструктуры хранения данных. Модели, функции и внутренние детали. 2012. URL: https://www.ibm.com/developerworks/ru/library/cl-cloudstorage/cl-cloudstorage-pdf.pdf (дата обращения: 27.06.2019).

32. Jonson L., Needham B. IEEE Std 1003.1e PSSG Draft 17. October 1997. URL: http://wt.tuxomania.net/topics/1999_06_Posix_1e/download/Posix_1003.1e-990310.pdf (дата обращения: 27.06.2019).