Distributed Data Storage Systems: Analysis, Classification and Choice

Alexander Tormasov <tor@innopolis.ru>
Anatoly Lysov <a.lysov@innopolis.ru>
Emil Mazur <e.mazur@innopolis.ru>
Innopolis University, 1, str. Universitetskaya, Innopolis
Republic of Tatarstan, Russian Federation, 420500

Abstract. There are a large number of distributed data storage systems, and the vendors have different definitions of what is their solution: cloud storage, distributed file system, or a cluster file system, etc. This imposes difficulty in the selection of the distributed storage system, because it is not clear what indicators you should pay attention in the first place.
This paper proposes an analysis of various distributed data storage systems and possible solutions to basic problems of the subject area, in particular, the issue of system scaling, data consistency, availability and partition tolerance.
In this work we have ranked distributed storage systems based on various characteristics and have chosen the top of them for a further analysis. As the result of the analysis key system development patterns and trends were identified. These trends were further studied for correlations with systems functional and non-functional attributes.
Based on the performed analysis we have classified the systems by different criteria, including presence or absence of particular functions or attributes. In the course of a comparative study we have investigated basic system functionality (archive storage, deduplication, geo-replication etc.) and system performance (system scalability limits, architecture, operating environment etc.). In addition, we analyzed safety mechanisms and system self-management tools.
Based on the analysis data and the classification of the systems we have proposed methods for distributed data storage systems selection. The results of this work may be used by researchers and practitioners to make a justified choice of a storage systems for their specific needs.

Keywords: storage systems, distributed storage systems

DOI: 10.15514/ISPRAS-2015-27(6)-15

For citation: Tormasov Alexander, Lysov Anatoly, Mazur Emil. Distributed Data Storage Systems: Analysis, Classification and Choice. Trudy ISP RAN/Proc. ISP RAS, vol. 27, issue 6, 2015, pp. 225-252 (in Russian). DOI: 10.15514/ISPRAS-2015-27(6)-15

1. Introduction

Digital data volumes keep growing at a great pace. According to IBM statistics 2014, the daily amount of new information generated worldwide is about 15 petabyte. Meanwhile, the overall number of digital data doubles approximately every two year (Fig. 1).

Considering the fact that companies do not rush to expand the budget for data storage and support, the gap between the data volume growth and associated costs of database maintenance keeps rising [1].

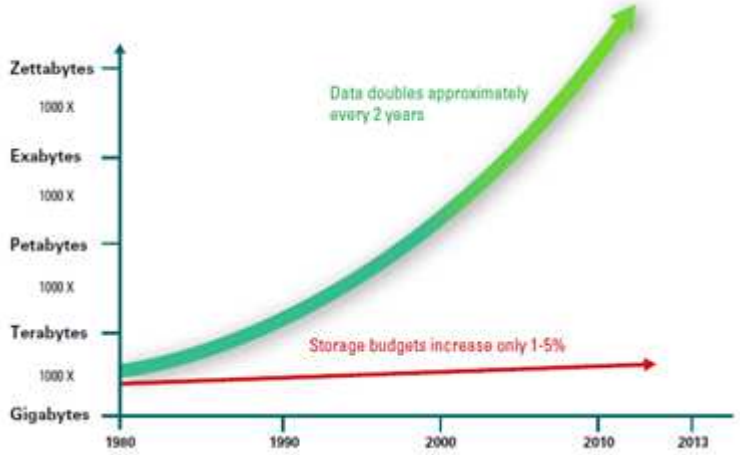

Figure 1: The growth of information

The majority of companies solve the problem of the explosive data growth via the purchase of hard drive arrays and network components thus expanding its data storage networks. This solution will eventually result in a complicated system administration (data backup, archiving etc.) and, accordingly, an increase in expenses on the system support. Thus, one can conclude that horizontal scaling is not reduce capital and operating costs associated with the database storage and maintenance.

Amid these problems, the network-wide dynamic boost of information regarding new solutions in the sphere of distributed data storage or software defined storage (SDS). SDS provide automated and policy-oriented storage services that consider the client and application-specific issues. SDS use a basic storage infrastructure and support of the software-defined environment in general. Correct deployment of SDS is a reliable solution for clients overwhelmed with large data volumes stored at complex hardware of different vendors [2]. SDS is a trend, so it's getting apparent that manufacturers have diverse opinions on the choice of the appropriate solution: cloud storage, distributed file system or cluster file system etc.

For the purpose of analysis, we combine all the solutions, including distributed file systems, cluster file systems and cloud platforms, in the group of the "Distributed Data Storage Systems". Our objective is to study a large number of mentioned solutions and develop approaches to the choice of distributed data storage systems.

In order to achieve this goal we have set the following tasks:

1. Make a list of systems to be analyzed;
2. Study each system and carry out a comparative analysis of selected systems;
3. Classify systems according to the identified common factors;
4. Design approaches to the choice of distributed data storage systems based on data analysis and classification.

The paper will be useful for those who are interested in effective data storage and is looking for options to justify a certain solution including those who are concerned about the issue of distributed data storage in general.

2. Selection of Systems to be Analyzed

Let us run a bit ahead and say that our list comprises approximately one hundred elements. Compiling the list, we used Internet articles, references found in forums, comments to discussions and a Wikipedia article that turned out to be quite useful [3].

The list turned out to be quite long, so we decided to shorten the list it. It was suggested a formula to calculate the system rating based on two indexes: the number of relevant Google links and the average number of requests in Google Trends for 2014 [4]. The formula calculates the average weighted value and the result is shown in percentage.

Scaling the list according to the rating identified the market leaders:

1. Amazon Simple Storage Service - 29.49%
2. Google File System (in particular, GFS2 - Colossus) - 27.11%
3. Microsoft Azure - 12.00%

Systems with 1 - 10% rating can be referred to the second group:

4. Global File System 2 - 8.43%
5. Ceph - 5.22%
6. Hadoop FS - 3.75%
7. Windows DFS - 2.62%
8. Quantcast File System - 2.02%
9. Gluster - 1.76%

All other systems were referred to the third group: Self-certifying File System, Server Message Block FS, General Parallel File System, VMware Virtual SAN, Openstack SWIFT, ViPR, Microsoft SharePoint Workspace, Data ONTAP,

Rackspace Cloud Files, AcroStorage, Chord File System, OdinStorage, dCache, GridFS, Elliptic Network, Cassandra File System, ExaFS, Moose File System, Coda, Coherent Remote File System, MogileFS, Apple Filing Protocol, Starfish, Lustre, CloudStore, GLORY-FS, StarFS, SmartCloud Virtual, Farsite, NetWare Core Protocol FS, Chiron FS, Parallel NFS, Oceanstore, OpenAFS, Kyoto Tycoon, Arla, InterMezzo, Panasas ActiveScale File System, HAMMER/ANVIL, Sheepdog, MapR-FS, OneFS, Cleversafe, OS4000, Gfarm, Tahoe-LAFS, OrangeFS, zFS, XtreemFS, LeoFS, IBRIX Fusion, OriFS, IFS (EMC Isilon), TerraGrid, Unilium, BeeGFS, PlasmaFS, TorFS, WebDFS, PeerFS, NimbusFS.

Considering that nine systems are not enough for a profound analysis, it was decided to take several systems from the third group that seemed interesting for analysis.

In total, the final sampling included 30 systems listed below:

Amazon S3, Google File System, MS Azure, Global File System 2, Ceph, Hadoop FS, Windows DFS, Quantcast File System, Gluster, AcroStorage, OdinStorage, TorFS, VMware Virtual SAN, OpenStack SWIFT, ViPR, Rackspace, dCache, GridFS, Elliptics Network, MooseFS, CODA, Lustre, OceanStore, OpenAFS, Kyoto Tycoon, Arla, Tahoe, zFS, Leo FS, Andrew File System.

3. System Analysis

In the course of a comparative study we have investigated basic system functionality (archive storage, deduplication, geo-replication etc.) and system performance (system scalability limits, architecture, operating environment etc.). In addition, we analyzed safety mechanisms and system self-management tools.

3.1. Analysis of the Main System Functionalities

Based on the analysis of the main system functionalities and classification provided in "A Taxonomy of Distributed Storage System" we have identified the below mentioned functions and mechanism [5]:

- Archive storage function;
- Data compression function;
- Data deduplication function;
- Controlled redundancy mechanism based on (n,k)-scheme;
- User interface support mechanism for the file system;
- Shared data access mechanism;
- Geo-replication mechanism;
- Object storage mechanism;

3.1.1. Archive Storage Function

Archive storage enables data backup storage and retrieval with the main usage scenario named “cold storage” i.e. a single data recording aimed for the long-term storage. As a rule, such data is appealed to in emergency cases [6]. The chart below shows the percentage ratio of systems under analysis in terms of the archive storage function (Fig. 2).

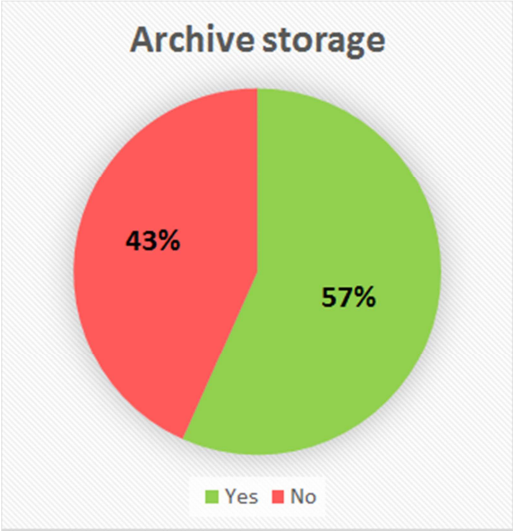

Figure 2: Archive storage

It should be noted, that the majority of systems under analysis are closed. In this and in other cases, if the functionality is not stated clearly, it is interpreted as missing.

According to the chart, the archive storage function is stated in more than a half of systems under analysis. Besides, this function is used by the rating leaders: Amazon S3, Google FS and MS Azure.

3.1.2. Data Compression Function

Data compression is algorithm-based data conversion for reducing the amount of data storage place [7]. According to suggested classification, systems can be divided into two groups: systems that use/ do not use this function (Fig. 3).

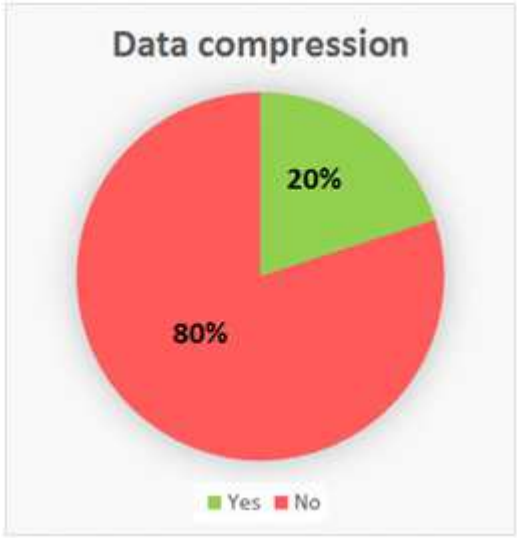

Figure 3: Data compression

Following the analysis, data compression function is used by less than a quarter of systems. It should be mentioned, that there is no “leading system” among them. Conversely, systems that identified the option of using data compression function are as follows: Ceph, HadoopFS, Gluster, RackSpace and dCashe.

3.1.3. Data Deduplication Function

Data deduplication is a technology for identification and eliminating duplicate copies of repeating data at the disk storage [8]. There are two types of deduplication: the file-level deduplication and the block-level deduplication.

In the file-level deduplication, for a deduplication unit is taken a single file serves. In this case, duplicating files are eliminated from the storage system.

In the block-level deduplication, for a deduplication unit is taken a variable length block, which is iterated in different logical objects of the data storage system [9].

Accordingly, the suggested classification includes 3 groups of systems (Fig. 4):

- Systems using the file-level deduplication function;
- Systems using the block-level deduplication function;
- Systems not using the deduplication function.

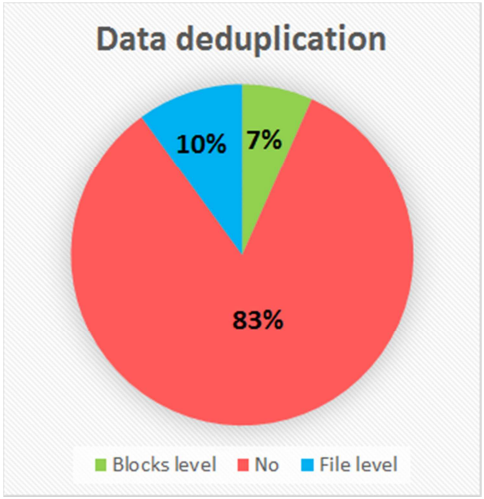

Figure 4: Data deduplication

Both data deduplication and data compression functions are supported by at least a quarter of the systems under analysis. Systems supporting the file-level deduplication are as follows: Ceph, RackSpace. Meanwhile, Amazon S3, HadoopFS and Windows DFS Systems belong to systems supporting the block-level deduplication.

Rating leaders: Google FS and MS Azure do not support deduplication.

3.1.4. Controlled Redundancy Mechanism based on (n,k)-Scheme

Here we are going to speak about a support of the so-called "erasure codes". The nature of such codes is as follows: after coding of a certain number of files we get the "n" chunks of data. Each of them is stored at a single cloud storage. In order to restore the initial file, it is necessary to collect and decode any "k" chunks of data. It should be noted that n > k. The rest (n-k) chunks can be deleted, damaged, unavailable etc (Fig. 5). Thus, any system using the "erasure codes" can solve the problem of errors emerging in (n - k) chunks [10, 11].

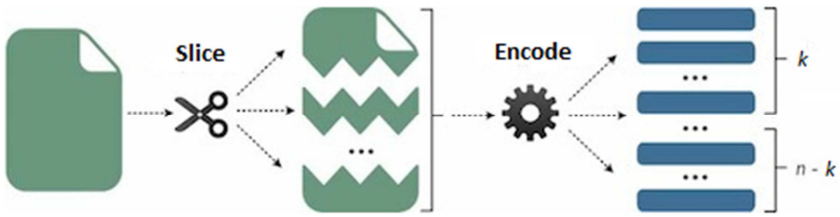

Figure 5: (n,k)-scheme

Accordingly, the suggested classification, systems can be divided into 2 groups: systems that support/ do not support such mechanism (Fig. 6).

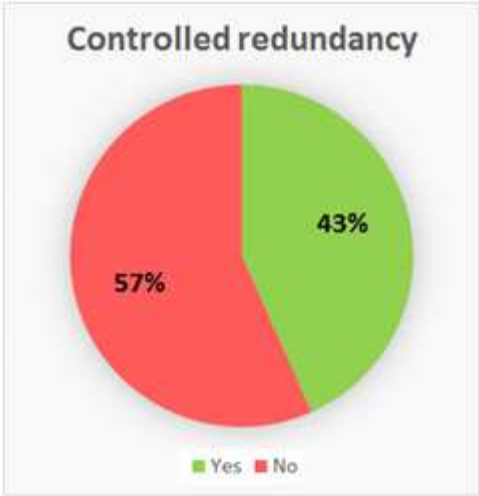

Figure 6: Controlled redundancy

Following the chart, the controlled redundancy mechanism based on (n,k)-scheme is supported by at least a half of systems under study. As for the leading systems, MS Azure and Google FS are the only to support the function. Amazon S3 is likely to support the function in one form or another. However, the functionality was not clearly stated.

3.1.5. Shared Data Access Mechanism

Based on the analysis, the shared data access mechanism is not supported by every system. Besides, among systems supporting this mechanism there are systems requiring / not requiring user authorization (not to be confused with user authentication) [5].

Accordingly, the suggested classification includes 3 groups of systems (Fig. 7):

- Systems providing shared data access with required authorization (Sharing-systems);
- Systems providing shared data access without required authorization (Anonymity-systems);
- Systems not providing shared data access.

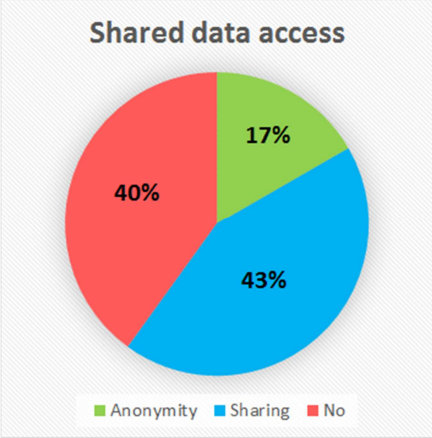

Figure 7: Shared data access

According to the chart, more than a half of systems use the shared access mechanism in one form or another. All the rating leaders support authorization-free access. HadoopFS and Rackspace refer to this group as well.

Systems that support the shared access to files and require authorization are as follows: Global File System 2, Ceph, QFS, Gluster, CODA, dCache, Elliptics Network, Leo FS, Lustre, SWIFT, ViPR, zFS, AFS.

3.1.6. Geo-Replication Mechanism

Geo-replication is a mechanism of synchronizing several object copies between geographically separated data centers [12]. Accordingly, systems can be divided into 2 groups: systems that support / do not support such mechanism (Fig. 8).

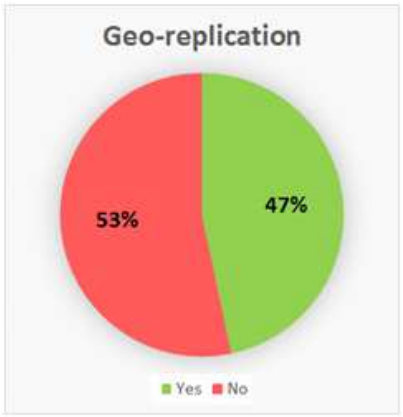

Figure 8: Geo-replication

Following the chart, over a half of systems under study have the geo-replication mechanism, including, in particular, the rating leaders: Amazon S3, Google FS and MS Azure.

3.1.7. Object Storage Mechanism

Object storage is a data storage architecture that unlike file system performs data management on the object level instead of blocks and sectors. [13].

Abstraction from multiple low-level storage tasks is one of the basic object storage principles. System administrators shall not be responsible for the logic volumes management, the disk space usage control and the RAID arrays configuration, which makes it easier to manage the storage system and reduces maintenance costs [13].

As opposed to classic arrays, object storage isolates the technological part of data storage, thus it enables easy system scaling, eliminates the hardware-dependence [14].

Thus, object storage provides relatively inexpensive scalable tool, which perfectly suits for effective storage of large non-structured data volumes. However, these advantages are achieved through lower requirements to data consistency. See the Section 2.6.6. for detailed information.

Coming back to the analysis, object storage is supported by almost a half of systems under study, including the rating leaders (Fig. 9).

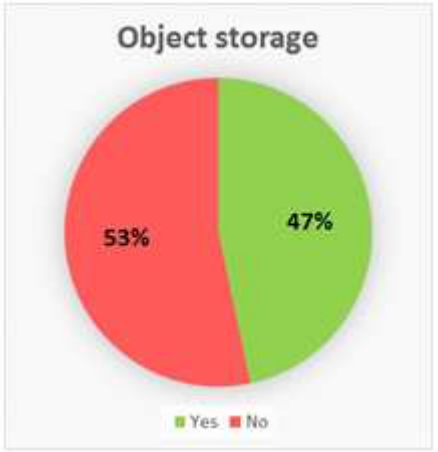

Figure 9: Object storage

3.2. Analysis of main system features

Besides the key system functionalities, comparative analysis included the investigation of non-functional system features. The following features were identified in the course of the analysis:

- System architecture;
- Scalability (based on data volume, number of servers, number of users);
- Cross-platformity (of server and client part);
- Access interface support NFS, SMB, https, WebDAV, S3 or proprietary protocol;
- Runtime environment;
- CAP theorem solution.

3.2.1. Architecture

Based on the analysis of system architectures, the latter were divided into two large classes: systems with client-server architecture and systems with peer-to-peer architecture. In the first case, a node acts as either a client or a server. Conversely, in the peer-to-peer architecture each node can be a client and a server at the same time.

In the client-server architecture, all control functions are concentrated in a single spot which provides the optimal safety and security level along with good performance. However, such concentration is a bottleneck, in particular, it affects system scalability and fault-tolerance because in case of a server failure it damages the whole system.

There are two types of the client-server infrastructure: globally-centralized and locally-centralized. In the first case, only one server is responsible for service provision and client servicing. Besides, all the problems related to the client-server architecture are apparent. In the second case, enhanced scalability and fault-tolerance can be achieved through the distribution of the control functions between groups of servers communicating with each other via data replication. Nevertheless, systems with the client-server architecture still have scalability limits.

As opposed to the client-server architecture, the peer-to-peer architecture is more suitable for untrusted environments where control and storage functions are distributed among all the nodes. Advantages of the peer-to-peer architecture refer to high scalability, self-management and fault-tolerance. However, such systems have their own shortcomings, including, low safety level, complex control if compared to the client-server architecture.

There are three types of the peer-to-peer architecture: globally-centralized, locally-centralized and pure peer-to-peer. In globally-centralized architecture, one of the system nodes serves as a central server which stores the information regarding other system nodes. Here the weak points are related with system scalability and fault-tolerance just like in the client-server architecture. In case of locally-centralized architecture, functions of the central server are distributed among several nodes with high performance. These nodes are called the super-nodes or the super-servers and their key function is to provide clients with information related to data location and accessibility.

Finally, the pure peer-to-peer architecture contains no specific nodes that identify the location of other nodes. Each node can be a server or a client which makes the peer-to-peer system well-adjustable to dynamic environment where the nodes can connect or disconnect from the network any time. As for the shortcomings, this type of infrastructure suffers from the low scalability and load asymmetry, requiring special mechanisms to remedy such defects. Besides, the system needs safety mechanisms to reduce the number of common threats [15].

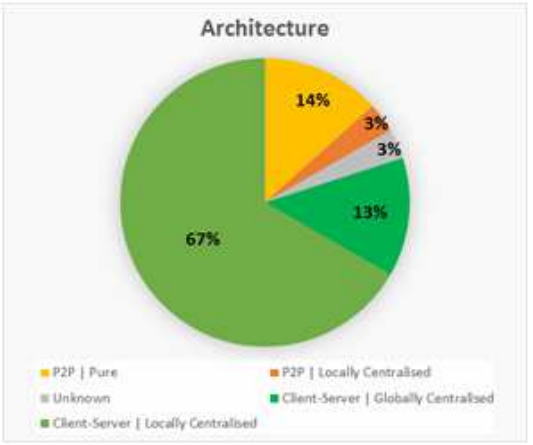

Figure 10: Architecture

Following the chart (Fig. 10), more than ¾ of systems under analysis have the client-server architecture. Meanwhile, the architecture of the majority of such systems refers to the locally-centralized one. The market leaders are among them.

Less than 20% of systems have the peer-to-peer architecture. We haven't found examples of globally-centralized architecture among them.

3.2.2. Scalability

We have analyzed suggested thresholds of system scalability according to the maximum volume of stored data, the number of servers and users who can operate with the system simultaneously.

When investigating the systems, we tried to avoid quality estimation and identified digital scalability indexes instead. It worked with just a part of systems. Some of them showed the abstract index "a lot". Another part showed unlimited capabilities. In addition, we could not find any information about the scalability thresholds of certain systems. In total, we have agreed that the "unlimited" is the maximum score, the "a lot" is 1 order lower than the "unlimited", while the "unknown" is a minimum score.

As a result, we have identified the following scalability range:

- Data Volume:
  Unknown < Petabytes (10^{15}) < Exabytes (10^{18}) < Brontobytes (10^{27}) < Not limited
- Number of servers:
  Unknown < x10 < $x10^2$ < $x10^3$< A lot < Not limited
- Number of users:
  Unknown < $x10^2$ < $x10^3$< $x10^4$ < $x10^6$< Not limited

The chart below integrates all three features and shows the whole picture (Fig. 11):

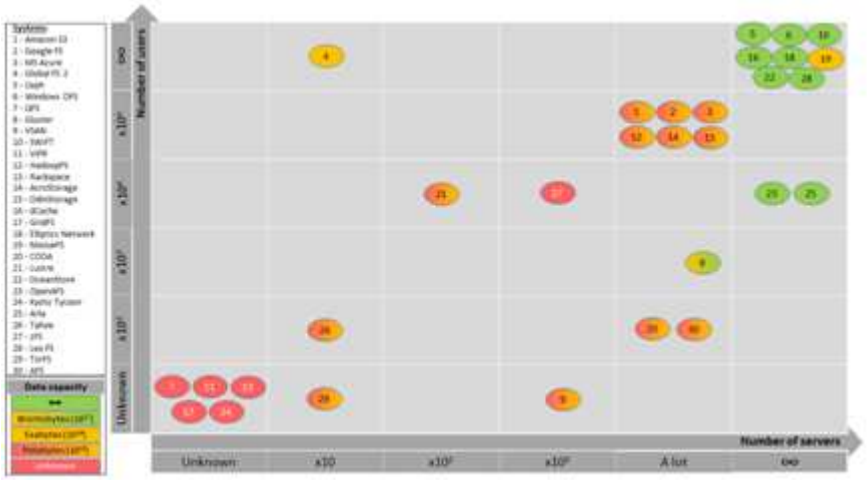

Figure 11: Scalability

According to the chart, one can make the following conclusions:

- On average, 7 systems claim to have unlimited capabilities or, in particular, their scalability threshold is limited solely with the company budget. Also, it should be noted that none of the rating leaders is among such systems;
- More than the third part of systems measure their data volume threshold in petabytes with only a few of them measuring the data volume threshold in exabytes and brontobytes;
- Among the sixth part of systems under analysis, none of them showed any information about scalability thresholds;
- In conclusion, the following trend is observed: a number of systems determine their limits as follows: “multiple servers”, “millions of users” and “petabytes of data”.

3.2.3. Cross-Platformity

Analysis of the platforms’ ability to support the server and the client part showed the following results (Fig. 12):

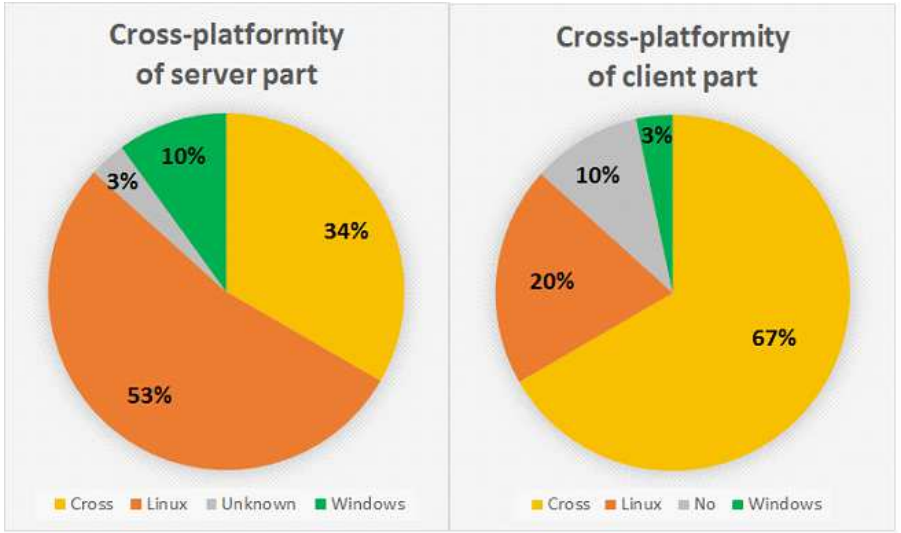

Figure 12: Cross-platformity

If we are talking about the server part, more than a half of systems prefers Unix-platforms. The rating leaders, in particular, Amazon S3 and Google FS, are among them. The leader, MS Azure refers to 10%, with its server part based on Windows.

The two third part of system under analysis have the cross-platform client part, including all three rating leaders.

3.2.4. Support of Access Interfaces

Support of access interfaces was one of the criteria of system analysis, in particular, the possibility to connect (mount) the NFS and SMB file systems as well as REST API (S3) and WebDAV access protocols.

See the chart below to find the information on distribution of systems in terms of the number of supported access interfaces mentioned above (Fig. 13).

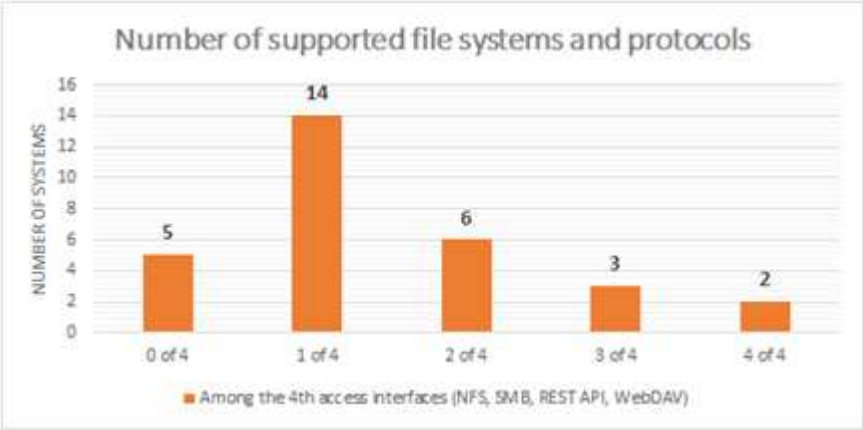

Figure 13: Number of supported file systems and protocols among the 4th kind of access interfaces (NFS, SMB, REST API, WebDAV)

However, the chart above does not take into account the proprietary access interfaces available in the majority of systems. If we add this information, the chart will look as follows (Fig. 14):

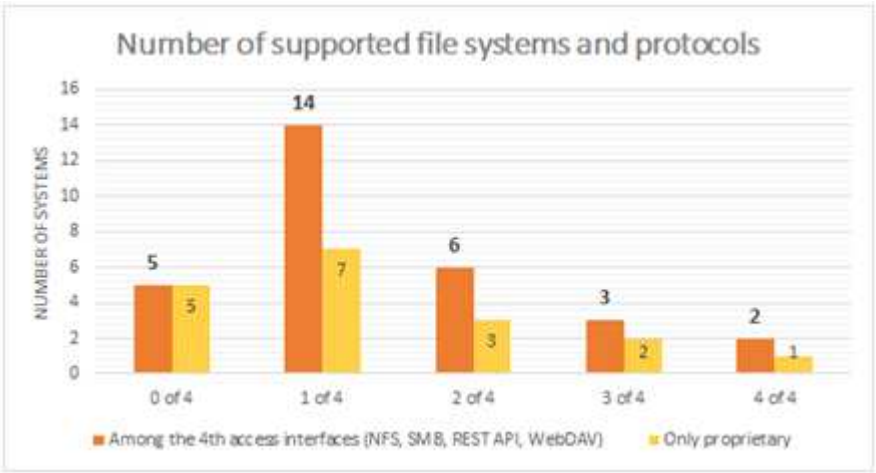

Figure 14: Number of supported file systems and protocols among the 4th kind of access interfaces (NFS, SMB, REST API, WebDAV) and proprietary interfaces

According to the second chart, most of the systems have their own access interfaces. It should be noted that only two systems support all 4 interfaces. These systems are Ceph and HadoopFS. Besides, Ceph has its own proprietary interface. Also, there

are 5 systems, supporting only proprietary interfaces. The leader Google FS along with zFS, TorFS, QFS, OceanStore are among them.

The rating leaders Amazon S3 and MS Azure refer to the group "1 of 4" supporting REST API (S3) protocols.

The chart below shows that the most "popular" interfaces are NFS and REST API. Also, more than a half of systems have their own proprietary access interfaces (Fig. 15).

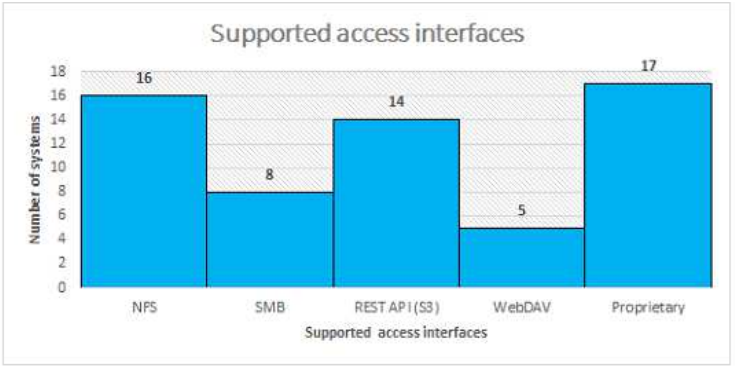

Figure 15: Supported access interfaces

3.2.5. Operating Environment

The operating environment is an important functional aspect of any distributed system. Trusted infrastructure is usually isolated from alien networks, which makes it predictable and easy-to-administrate. Controlled environment provides a high-quality servicing and trust. However, it adds to a substantial scalability limitation. Conversely, the untrusted environment implies a close interaction with the open access networks. In the open environment, it is hard or practically impossible to keep any records of users or control them. Systems located in the untrusted environment are subject to multiple attacks. Thus, such systems shall be equipped with additional safety mechanisms.

Based on our analysis, we have singled out 4 main types of the operating environments: Trusted, Partially trusted, Untrusted and Alien Infrastructure.

If a system is deployed on its own infrastructure, i.e. in the local corporate network on its own hardware, including the cases when any other software is not installed on the hardware platform, such environment shall be considered ***trusted***.

Likewise, if the system is deployed on its own infrastructure, i.e. in the local corporate network and installed on its own hardware, given that at least one different-type software is installed on the hardware platform, such environment shall be considered ***partially-trusted.***

If such system is deployed on its own hardware but is connected through the alien network, in particular, the Internet, such environment shall be considered ***untrusted.***

If the system is deployed on the alien hardware and connected through the alien network, in particular, the Internet, such environment shall be considered the ***alien infrastructure***.

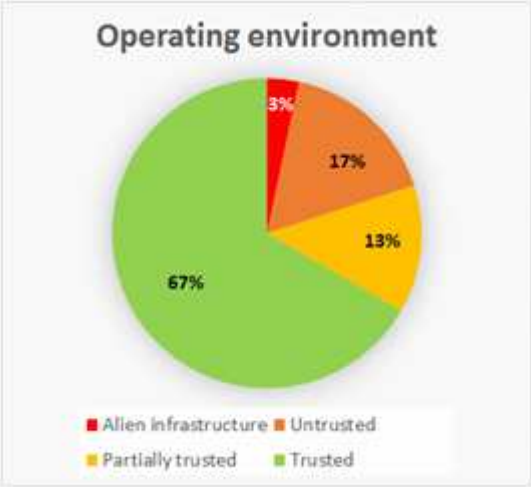

Figure 16: Operating environment

According to the chart (Fig. 16), only the third part of systems is able to operate in untrusted environments. Tahoe, however, can operate in the environment called the "alien infrastructure". In Tahoe, when the access to resources is opened, it is followed by data encryption, encoding and caching based on the (n-k) scheme or the principle of "minimal privileges" [16].

Amazon S3 and MS Azure, the rating leaders, refer to 67% of systems, that can operate only in the trusted environment.

Google FS, the rating leader, belongs to 13% systems, that can operate in partially-trusted environments.

3.2.6. Solutions to the CAP-theorem

According to Wikipedia, the CAP theorem (also known as Brewer's theorem) is a heuristic statement of the fact that in any distributed computing can provide only two out of three features as follows:

- Data consistency i.e. all computing nodes see the same data;
- Availability i.e. every request to the distributed system receives a correct response;

- Partition tolerance i.e. partition of the distributed system into several separate sections does not result into incorrect response received from each section [17].

In the second half of 2000-s, Brewer introduced the acronym BASE (Basically Available, Soft-State, Eventually Consistent) that implied that the requirements of integrity and availability are only partially fulfilled.

Consistency models are numerous. However, the systems under analysis can be referred to the following three models:

- Strong consistency;
- Eventual consistency;
- Weak consistency.

Strong consistency model guarantees that after the update any further data access will restore the updated values [18].

Eventual consistency model guarantees that in case of no data changes all the inquires will be eventually returned to the latest updated value.

Weak consistency model guarantees that further data inquires will restore the updated value. Before the updated value is restored it is necessary to fulfill a certain requirements. The inconsistency window is the time between the update and the moment when each user is sure to see the updated value.

To see the full picture and understand which of the "two out of three" parameters of CAP-theorem are fulfilled by the systems under analysis, please refer to the chart below that integrates all three parameters (Fig. 17):

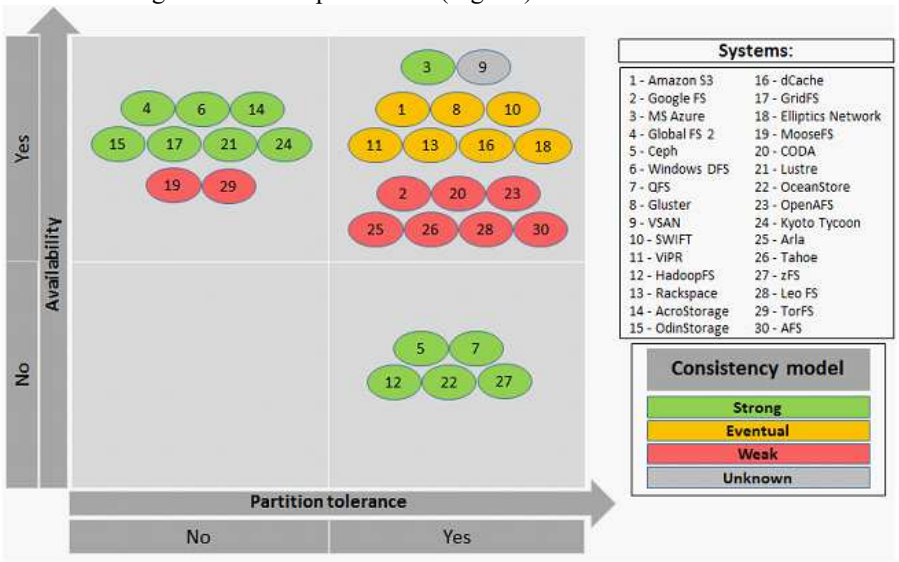

Figure 17: CAP-theorem

According to the chart, more than a half of systems are “PA” i.e. cannot guarantee consistency, but are CAP-available and partition tolerant. Besides, one half of such systems supports the eventual consistency while the other half supports the weak consistency.

The rating leaders also belong to such systems: Amazon S3 supports the eventual consistency, Google FS – the weak consistency. It is worth mentioning MS Azure. Azure developers claim to have found solutions for all three problems, providing strong consistency, availability and resilience. Meanwhile, this solution is valid for only certain types of system failures, in particular, “node failures” and “top-of-rack” failures. To provide this solution, strict consistency and availability are divided into two levels: Stream Layer and Partition Layer accordingly. In case of the node / workstation failure, the stream layer switches to the properly functioning node/workstation and keeps operating (reading/recording). Meanwhile, the partition layer identifies lost data and duplicates them to the properly functioning nodes/ workstations [19].

The second rating position belongs to the “CA”-systems, that cannot guarantee the partition tolerance but support strict consistency and availability. Here, 2 systems are worth mentioning. Technically, they cannot be referred to the “CA”-systems, because they support weak consistency.

The third rating position is taken by the “PC”-systems, that cannot guarantee availability but support strict consistency and partition tolerance.

3.3. Analysis of Safety Functions and System Self-Management Tools

3.3.1. System Safety Functions

The main system safety functions refer to the following:

- User authentication;
- Data access management;
- Data privacy;

User authentication can rely either on the local base, located in the system, or on the network authorization protocols (Kerberos, Radius, LDAP etc.).

Access control list (ACL) is a tool that is often used to solve the problem of the data access management.

Data privacy is provided through the encryption mechanism.

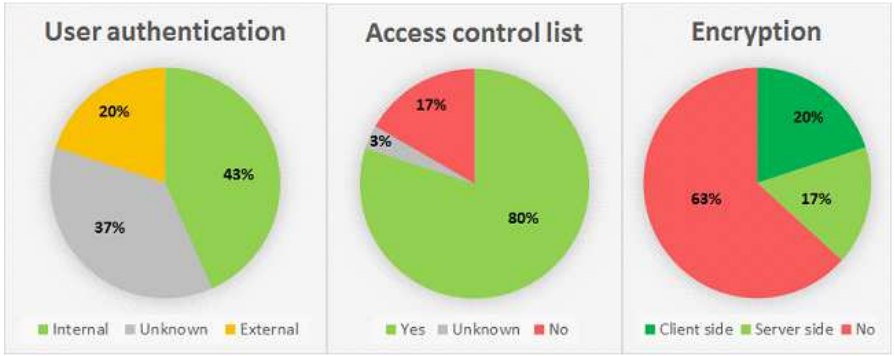

Figure 18: System safety functions

According to the charts (Fig. 18):

- Most of systems, including the rating leaders, use the local user base for authentication. However, it should be taken into account that no information regarding authentication mechanisms was found in respect of the majority of systems;
- More than a three quarter of systems under study, including the rating leaders, use access control lists;
- Almost two-thirds of systems do not use encryption. Among all the rating leaders, encryption is used only by Amazon S3. Encryption is based on the client part and employs symmetric encryption algorithm AES in GCM mode with 256-bit encryption keys [20].

3.3.2. Analysis of system self-management mechanisms

System self-management is a process when computer systems manage their own operation without human interaction. Modern distributed computer systems are heterogeneous and represent a combination of various information technologies combining network, mobile and wireless technologies. Manual control of such systems is complicated and labor-consuming which is the main aspect decelerating the development of such systems [21].

IBM is the major contributor to the development of self-managing systems. In 2001, the company created a special initiative group. IBM singles out 4 main features of self-managing systems [22]:

- **Self-Configuration** is the ability of a system to coordinate the values of low-level parameters (for instance, components set-up) with the high-level rules determined by business goals, and apply such parameters;
- **Self-Optimization / Self-Adaptation** is the ability of a system to provide continuous control and management of its resources in order to achieve the most efficient operation depending on certain requirements;

- **Self-Healing** is the ability of a system to identify and fix the occurred errors and heal in case of the single components failure using all possible means;
- **Self-Protection** is the ability of a system to protect itself from any harmful effects or successive failures. Systems operating on the Internet are exposed to a wide range of attacks. Thus, self-protection is of special importance.

The chart below (Fig. 19): shows that self-healing and self-optimization functions are dominating among the systems under study. Self-configuration and self-protection functions are less frequent.

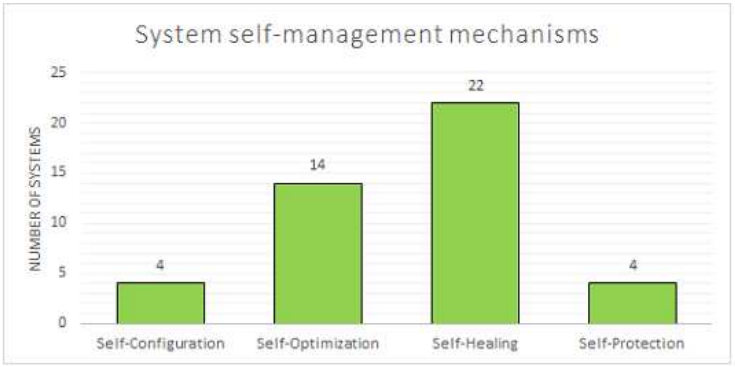

Figure 19: System self-management mechanisms

Self-healing function can be as follows:

- Healing after the disk failure;
- Healing after the node failure;
- Healing after the datacenter failure.

The chart below shows (Fig. 20) the results of the analysis in terms of available self-healing functions. More than two-thirds of systems support healing functions after the failure of disks and servers. Less than the third part of systems support the healing function after the collapse of data-centers.

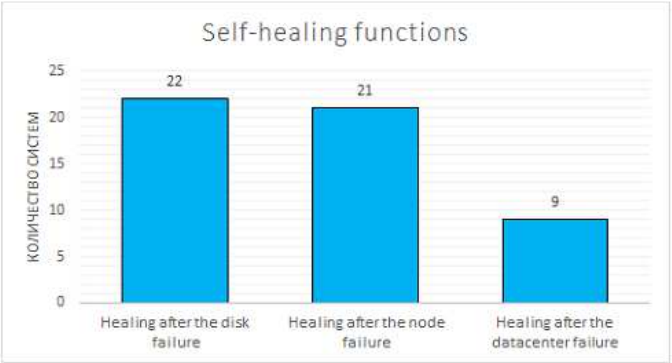

Figure 20: System self-healing functions

The **self-optimization** functions can be referred to the following:

- Optimizing consistency level is the selection of efficient consistency level of transferred data based on the analysis of users activity;
- Caching mechanism. The name speaks for itself. Self-adaptation is understood as the analysis of the application load and self-configuration of caching parameters;
- Adaptation of operation with SSD i.e. identification of speed-critical elements (caches, journals etc.), based on the analysis of the data inquiry frequency and their reading/ recording time including storage migration to SSD drives;
- Predicting disk failure i.e. use of the hard drive condition technologies like the “SMART”, and prediction of the disk failure period;
- Load balancing i.e. keeping record of the network functioning and acceptance of distributed data transmission to other nodes;
- Control of electricity i.e. power-off in order to save on disks that store redundant information;
- Control of content popularity i.e. calculation of data inquires and approval of the popular content replica increase or, conversely, deleting data which was not requested for a long time.

Results of the analysis in terms of self-optimization functions are illustrated on the chart below (Fig. 21). Apparently, caching is the most popular function supported by almost half of systems. Over one third of systems support the self-adaptation function with SSD. None of the systems demonstrated control of electricity.

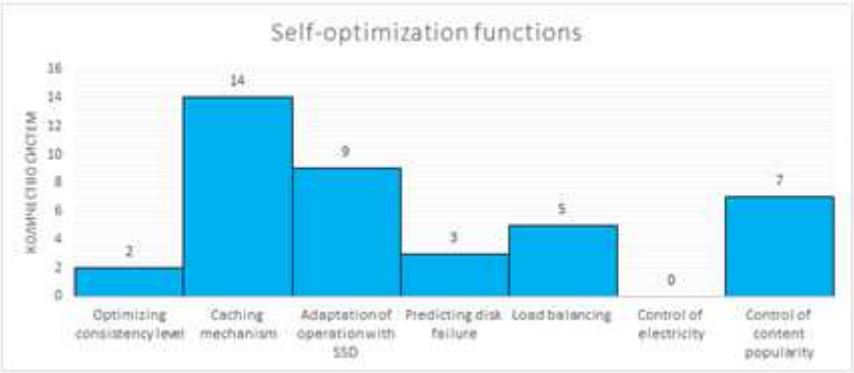

Figure 21: System self-optimization functions

4. Findings

According to the analysis, the broad market of distributed data storage systems lacks a single approach to design of such systems. In most cases, manufacturers rely on business tasks and try to reach the balance between high performance, scalability, safety and easy management, which makes quite a challenging task.

Below is the list of our findings that illustrate why certain functions and mechanisms are necessary for the distributed systems in terms of business tasks.

1. If your company stores data that does not require the every-day access but need to be stored for a long time, for instance, in accordance with the legal requirements, it is reasonable to use the archive storage mechanism. In most cases, the so-called "cold" data storage is more cost-efficient as opposed to the "hot" data storage.
2. Distributed storage is de facto more partition tolerant, so according to the CAP-theorem, the question of choice remains between the data consistency and data availability. If you do not store data that is always in-demand, it would be more rational to sacrifice the data consistency and use the "PA-model" with the weak data consistency.
3. The mechanisms like (n,k)-encoding, data compression and deduplication will sufficiently reduce the amount of stored information. Meanwhile, such system will be more expensive.
4. No one is secured from natural and man-induced disasters [23]. So, if you want your information to be protected from such factors, it would be wise to store data in the remote places geographically separated from each other using the geo-replication mechanism.
5. If the storage access is supported via the untrusted data channel, it's apparent that encryption mechanisms shall be used to provide data privacy. However, if data processing and retrieval occurs in the controlled area, encryption can be ignored in order to enhance the performance.
6. It makes no sense trying to affect the architecture, cross-platformity, operating environment and object storage due to the conceptual and system-forming nature of these features that determine all the rest functions and mechanisms. As for the scalability, it depends largely on the company's budget.

A small number of systems have the full range of self-awareness and self-management functions. Nevertheless, we believe that the development of such technologies is quite promising. Self-awareness tools make the system more flexible, which is important in terms of current business realities.

5. Conclusion

In this research paper we have analyzed over 30 distributed data storage systems. In the course of work it became apparent that the majority of systems under study are closed. For this reason, it was decided that if the functionality was not clearly specified, it is missing. Thus, it is most likely that statistic data described in this paper do not show 100% real picture.

Despite this fact, in this paper we have managed to classify systems in accordance with various parameters, including basic functional capabilities, features, safety mechanisms and self-management which served as the basis for approaches to distributed data storage systems. Thus, one can say that the objectives of these research work were completed in full and the goal was achieved.

Acknowledgements. This work has been supported by the Russian Ministry of education and science with the project "Development of new generation of cloudy technologies of storage and data control with the integrated security system and the guaranteed level of access and fault tolerance". (agreement: 14.612.21.0001, ID: RFMEFI61214X0001).

References

[1]. IBM Platform Computing Edition, Software Defined Storage For Dummies,New Jersey, 2014, pp.4-5 .

[2]. Software-Defined Storage (SDS) - perspektivy rosta [growth prospects] - ChannelForIT. Available at: http://channel4it.com/blogs/Programmno-opredelyaemye-hranilishcha-vsyo-bolee-vostrebovany-6787.html (accessed 23 July 2015). (In Russian)

[3]. List of file systems - Wikipedia, the free encyclopedia. Available at: https://en.wikipedia.org/wiki/List_of_file_systems (accessed 23 July 2015).

[4]. Google Trends. Available at: https://www.google.com/trends (accessed 23 July 2015).

[5]. M.Placek, R. Buyya, A taxonomy of distributed storage systems, p. 53.

[6]. Kak Facebook sekonomil 75% energii, kotoraya trebuetsya dlya khraneniya dannykh pol'zovatelei [Fasebook saved 75% of the energy required to store users' data] / King Servers company's blog / Habrahabr. Available at: http://habrahabr.ru/company/kingservers/blog/257699/ (accessed 23 July 2015). (In Russian)

[7]. Data compression - Wikipedia, the free encyclopedia. Available at: https://en.wikipedia.org/wiki/Data_compression (accessed 23 July 2015).

[8]. Deduplikatsiya dannykh — podkhod NetApp [Deduplication - NetApp's approach] / NetApp Company's Blog / Habrahabr. Available at: http://habrahabr.ru/company/netapp/blog/110482/ (accessed 23 July 2015). (In Russian)

[9]. Vvedenie v deduplikatsiyu dannykh [Introducition to data duplication] / Veeam Software company's blog. Available at: http://habrahabr.ru/company/veeam/blog/203614/ (accessed 23 July 2015). (In Russian)
[10]. Erasure code - Wikipedia, the free encyclopedia. Available at: https://en.wikipedia.org/wiki/Erasure_code (accessed 23 July 2015).
[11]. O chem stoit zadumat'sya, sokhranyaya svoi dannye v oblake. Chast' 2 [What you should thinking about, when you saving data in cloud. Part 2] / Habrahabr. Available at: http://habrahabr.ru/post/141487/ (accessed 23 July 2015). (In Russian)
[12]. Vysokaya dostupnost' web-saita: georeplikatsia failov saita s "lsyncd" [Web site's high availability: file geo-replication of site with "lsyncd"] / Habrahabr. Available at: http://habrahabr.ru/company/infobox/blog/252751/ (accessed 23 July 2015). (In Russian)
[13]. Object storage - Wikipedia, the free encyclopedia. Available at: https://en.wikipedia.org/wiki/Object_storage (accessed 23 July 2015).
[14]. Ob''ektnaya sistema khraneniya dannykh - konkurent zheleznykh SKHD [Object system of data storage - the competitor of hardware SAN]. Available at: http://www.jetinfo.ru/stati/konkurenty-zheleznykh-skhd (accessed 23 July 2015). (In Russian)
[15]. Y. R'kaina, Reliable and persistent storage for CoDeS a distributed collaborative system // 2013, p. 40
[16]. An Overview of Tahoe-LAFS. Secure and fault tolerant distributed storage system. Available at: https://code.google.com/p/nilestore/wiki/TahoeLAFSBasics (accessed 23 July 2015).
[17]. CAP theorem - Wikipedia, the free encyclopedia. Available at: https://ru.wikipedia.org/wiki/Теорема_CAP (accessed 23 July 2015). (In Russian)
[18]. Soglassovannye v konechnom schete [Eventually Consistent]. Available at: http://habrahabr.ru/post/100891 (accessed 23 July 2015). (In Russian)
[19]. B. Calder, J. Wang, A. Ogus, N. Nilakantan, A. Skjolsvold, S. McKelvie, Y. Xu, S. Srivastav, J. Wu, H. Simitci, J. Haridas, C. Uddaraju, H. Khatri, A. Edwards, V. Bedekar, S. Mainali, R. Abbasi, A. Agarwal, M. Fahim ul Haq, M. Ikram ul Haq, D. Bhardwaj, S. Dayanand, A. Adusumilli, M. McNett, S. Sankaran, K. Manivannan, L. Rigas Windows Azure Storage: a highly available cloud storage service with strong consistency // SOSP '11 Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, 2011, 143-157.
[20]. M. Campagna, AWS Key Management Service Cryptographic Details, 2015, p. 28
[21]. Autonomic computing - Wikipedia, the free encyclopedia. Available at: https://en.wikipedia.org/wiki/Autonomic_computing (accessed 23 July 2015).
[22]. Self-management (computer science) - Wikipedia, the free encyclopedia. Available at: https://en.wikipedia.org/wiki/Self-management_(computer_science) (accessed 23 July 2015).
[23]. Google poteryal chast' dannykh pol'zovatelei iz-za udara molnii - BBC Russkaya sluzhba [Google lost a part of users' data because of a lightning strike - BBC Russian Service]. Available at: http://www.bbc.com/russian/international/2015/08/150819_google_lightning_data (accessed 25 August 2015). (In Russian)

Распределенные системы хранения данных: анализ, классификация и варианты выбора

Александр Тормасов <tor@innopolis.ru>
Анатолий Лысов <a.lysov@innopolis.ru>
Эмиль Мазур <e.mazur@innopolis.ru>
Университет Иннополис, 4200500, Россия, Республика Татарстан, г. Иннополис, ул. Университетская, д.1

Аннотация. В статье содержится анализ различных распределенных систем хранения данных и возможных решений основных проблем этой области, в частности, проблем масштабирования систем, согласованности данных, доступности и устойчивости к разделению. Проведенный анализ позволил выявить ряд закономерностей и попытаться классифицировать системы на основе разных параметров, в частности, при наличии или отсутствии специфических функций и механизмов. Варианты выбора распределенных систем хранения данных основываются на анализе и классификации.

Ключевые слова: системы хранения, распределенные системы хранения

DOI: 10.15514/ISPRAS-2015-27(6)-15

Для цитирования: Тормасов Александр, Лысов Анатолий, Мазур Эмиль. Распределенные системы хранения данных: анализ, классификация и варианты выбора. Труды ИСП РАН, том 27, вып. 6, 2015 г., стр. 225-252. DOI: 10.15514/ISPRAS-2015-27(6)-15.

Литература

[1]. IBM Platform Computing Edition, Software Defined Storage For Dummies,New Jersey, 2014, pp.4-5 .
[2]. Программно-определяемые хранилища всё более востребованы – ChannelForIT. http://channel4it.com/blogs/Programmno-opredelyaemye-hranilishcha-vsyo-bolee-vostrebovany-6787.html (дата обращения 23 July 2015).
[3]. List of file systems - Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/List_of_file_systems (дата обращения 23 July 2015).
[4]. Google Trends. https://www.google.com/trends (обращение 23 July 2015).
[5]. M.Placek, R. Buyya, A taxonomy of distributed storage systems, p. 53.
[6]. Как Facebook сэкономил 75% энергии, которая требуется для хранения фоточек котиков и селфи пользователей / Блог компании King Servers / Habrahabr. http://habrahabr.ru/company/kingservers/blog/257699/ (дата обращения 23 July 2015).
[7]. Data compression – Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Data_compression (дата обращения 23 July 2015).
[8]. Дедупликация данных – подход NetApp / Блог компании NetAppg / Habrahabr. http://habrahabr.ru/company/netapp/blog/110482/ (дата обращения 23 July 2015).

[9]. Введение в дедупликацию данных / Блог компании Veeam Software. http://habrahabr.ru/company/veeam/blog/203614/ (дата обращения 23 July 2015).
[10]. Erasure code - Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Erasure_code (обращение 23 July 2015).
[11]. О Чем Стоит Задуматься, Сохраняя Свои Данные в Облаке. Часть 2 / Habrahabr. http://habrahabr.ru/post/141487/ (дата обращения 23 July 2015).
[12]. Высокая доступность веб-сайта: георепликация файлов сайта с lsyncd / Habrahabr. http://habrahabr.ru/company/infobox/blog/252751/ (дата обращения 23 July 2015).
[13]. Object storage - Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Object_storage (дата обращения 23 July 2015).
[14]. Объектная система хранения данных - конкурент железных СХД. http://www.jetinfo.ru/stati/konkurenty-zheleznykh-skhd (дата обращения 23 July 2015).
[15]. Y. R'kaina, Reliable and persistent storage for CoDeS a distributed collaborative system // 2013, p. 40
[16]. An Overview of Tahoe-LAFS. Secure and fault tolerant distributed storage system. https://code.google.com/p/nilestore/wiki/TahoeLAFSBasics (дата обращения 23 July 2015).
[17]. Теорема CAP – Wikipedia, the free encyclopedia. https://ru.wikipedia.org/wiki/Теорема_CAP (дата обращения 23 July 2015).
[18]. Согласованные в конечном счете (Eventually Consistent) . http://habrahabr.ru/post/100891 (дата обращения 23 July 2015).
[19]. B. Calder, J. Wang, A. Ogus, N. Nilakantan, A. Skjolsvold, S. McKelvie, Y. Xu, S. Srivastav, J. Wu, H. Simitci, J. Haridas, C. Uddaraju, H. Khatri, A. Edwards, V. Bedekar, S. Mainali, R. Abbasi, A. Agarwal, M. Fahim ul Haq, M. Ikram ul Haq, D. Bhardwaj, S. Dayanand, A. Adusumilli, M. McNett, S. Sankaran, K. Manivannan, L. Rigas Windows Azure Storage: a highly available cloud storage service with strong consistency // SOSP '11 Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, 2011, 143-157.
[20]. M. Campagna, AWS Key Management Service Cryptographic Details, 2015, p. 28
[21]. Autonomic computing - Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Autonomic_computing (дата обращения 23 July 2015).
[22]. Self-management (computer science) - Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Self-management_(computer_science) (дата обращения 23 July 2015).
[23]. Google потерял часть данных пользователей из-за удара молнии – Русская служба BBC. http://www.bbc.com/russian/international/2015/08/150819_google_lightning_data (дата обращения 23 July 2015).