Increasing reliability and fault tolerance of a secure distributed cloud storage

Nikolay Kucherov[1,†], **Mikhail Babenko**[1,††], **Andrei Tchernykh**[2,‡], **Viktor Kuchukov**[1,‡‡] **and Irina Vashchenko**[1,†‡]

[1] North-Caucasus Federal University,Stavropol,Russia
[2] CICESE Research Center,Ensenada,Mexico

E-mail: [†]nkucherov@ncfu.ru, [††]mgbabenko@ncfu.ru, [‡]chernykh@cicese.edu.mx, [‡‡]viktor-kuchukov@yandex.ru, [†‡]irishechka.26@mail.ru

Abstract. The work develops the architecture of a multi-cloud data storage system based on the principles of modular arithmetic. This modification of the data storage system allows increasing reliability of data storage and fault tolerance of the cloud system. To increase fault-tolerance, adaptive data redistribution between available servers is applied. This is possible thanks to the introduction of additional redundancy. This model allows you to restore stored data in case of failure of one or more cloud servers. It is shown how the proposed scheme will enable you to set up reliability, redundancy, and reduce overhead costs for data storage by adapting the parameters of the residual number system.

1. Introduction

Currently, cloud services, Google, Amazon, Dropbox, Microsoft OneDrive, providing cloud storage, and data processing services, are gaining high popularity. The main reason for using cloud products is the convenience and accessibility of the services offered. Thanks to the use of cloud technologies, it is possible to save financial costs for maintaining and maintaining servers for storing and securing information. All problems arising during the storage and processing of data are transferred to the cloud provider [1].

Distributed infrastructure represents the conditions in which competition for resources between high-priority computing tasks of data analysis occurs regularly [2]. Inevitably, congestion in computing resources causes a deterioration in the functioning of serving services, and sometimes long breaks in their work. For this and other reasons, a continuous flow of failures, errors, and malfunctions inevitably occurs in distributed data processing [3].

The main task of storage systems is the task of real-time data processing; however, a contradiction arises between the practical need for real-time processing of significant data bits, the limited hardware resources of modern systems, cost, reliability, and performance. Thus, it is necessary to increase the speed of arithmetic calculations, reliability, and availability by developing new mathematical models of data processing in the cloud, the algorithms of which use methods that reduce their time, and also create new algorithms for processing and storing data to reduce financial costs.

When a cloud service provider stores user data, it should be able to return this data to the user on demand. Given network downtime, user errors, and other circumstances, meeting this condition in a reliable and deterministic way can be difficult [4]. Algorithm Information Dispersal Algorithms (IDA) [5], initially developed by Michael Rabin for telecommunication systems [5], the IDA algorithm allows you to split the data in such a way that in case of loss or inaccessibility of some data, it is possible to restore the original data [6].

To increase the reliability and reliability of data processing and storage, it is advisable to use distributed data storage schemes based on the principles of modular arithmetic. One of the promising areas of modular arithmetic is the development of mathematical methods for storing and processing large-capacity data with high reliability in a cloud environment. The primary tool for improving reliability is the introduction of controlled redundancy in the system.

When building a large cloud, on which all the company's business systems work, it is necessary to ensure high fault tolerance, regularly back up data so that when the server crashes, the clouds do not turn off at the same time, but immediately switch to another server, or (within a reasonable time) were restored from backups. All this leads to an increase in the cost of creating a fault-tolerant architecture. Investments in maintaining and backing up clouds are increasing.

The IDAs proposed in [7, 8] ensure the availability and distribution of data. Redundant residue number system (RRNS) has similar properties for the Mignotte data distribution scheme [8]; its arithmetic properties allow controlling the result of data processing. RRNS presents the original numbers as residuals to the set of modules. Thus, the number is broken up into smaller numbers that are independent.

Let $p_1, p_2, \ldots, p_n$ be paired with mutually simple numbers used as a set of redundant RNS modules, and $n = k+r$. Then the range of the redundant RNS will be $P = \prod_{i=1}^{n} p_i$. The data is an integer of X, where $X is \in [0,\, P-1)$. X is defined in redundant RNS as $X \to (x_1, x_2, \ldots, x_n)$, where $x_i = |X|_{p_i}$ represents the remainder of the division of X by p_i.

Parameters $(k,\, r)$ RRNS can be selected differently depending on the need to obtain specific characteristics. Using data from any k residues from n, we can recover the data. According to the RRNS property, if the number of control modules is r, the system can detect r and fix the $r-1$ error. To localize and correct errors, we use projection methods where the number of calculated projections grows exponentially, depending on the r value.

In [9], the authors suggested using RRNS for reliable and scalable cloud storage systems. Operations can be performed in parallel, which simplifies and speeds up calculations. The redundancy of the system allows building a system with multiple error detection and correction.

Since the representation of numbers in RRNS can be considered as a data separation scheme, we can use it for reliable storage of large-dimensional data. In the case of finding a system for one PC, it is enough to detect and fix one error; most systems cope with this task. However, when we consider data of enormous capacity, it is necessary to have practical algorithms for detecting and correcting several errors. The RRNS storage scheme provides reliable and scalable storage. It has the properties of error correction codes and the possibility of distributed storage of significant data bits.

To create a reliable, fault-tolerant, and safe model for storing data in a distributed cloud structure, we will use RRNS and error correction codes. The data warehouse will have the following properties: reliability, fault tolerance, integrity, data distribution, data control, and error correction.

2. Overview of cloud storage security approaches

Big data refers to data processing methods based on non-traditional technologies because of their sizes, such as the collection, storage, retrieval, dissemination, analysis, and visualization of large volumes of data. Standard databases and tools can no longer cope with the growing flow of data. In essence, databases are no longer able to process existing volumes, ETL processes are too slow and have difficulties with a variety of data formats, so traditional BI systems are too slow and cannot handle large masses effectively unstructured data.

Processing large amounts of data often become the most problematic and challenging area in creating large aggregation services. This led to the creation of quite effective ways to solve the problem.

We will dwell on two of the most common solutions: the MapReduce distributed computing model and the Percona server, a MySQL assembly originally designed and optimized for working with big data.

The MapReduce distributed computing model presented by Google is used by the company in computer clusters for parallel computing over very large, even a few petabytes, data sets [10].

The advantage of MapReduce is its ability to distribute preprocessing and convolution operations in a distributed manner. Preprocessing operations can be performed in parallel as they work independently of each other. However, the process may be less efficient compared to more sequential algorithms, since the purpose of the MapReduce algorithm is to apply it to large amounts of data that can be processed by a large number of servers. However, MapReduce can be used to sort a massive amount of data, and requires only a few hours, even for volumes of the order of petabyte of data. Concurrency also provides the ability to recover from partial failures: if a failure occurs in a work node, then its work can be transferred to another working node. Thus, although the semantics differ from the prototype, the framework is based on the functions mar and reduce, which are widely used in functional programming.

Percona Server is a MySQL build. This build includes the XtraDB storage engine by default, which is different from the MySQL+InnoDB plugin. Key indicators are better performance/scalability, especially on modern multi-core servers. XtraDB repository is based on InnoDB-plugin and is fully compatible with it. However, it is characterized by higher performance due to the integration of patches from Google and Percona.

In the area of data integration, the main problem is the speed and controllability of structured data. For file storage and subsequent processing of big data, special file systems are available, such as HDFS from Hadoop, but also the so-called NoSQL databases. These methods must be consistent with classical analytical databases that continue to use. Only in this way can data consistency be maintained, and typical relational operations can be performed without problems.

Fast Big Data processing focuses on Google's MapReduce approach. The following algorithm is behind this: the task is divided into the smallest possible parts, then distributed for parallel processing on as many computers, and then combined again. Thus, high parallel processing of poly structure data is possible. Another tool that can process big data in seconds is in-memory computing, such as SAP HANA, offered by SAP. Here, computer memory is used as data storage. Unlike data stored on the hard drive, this can significantly increase the speed of access to data. Some solutions rely on analytic databases. These are mainly column-oriented databases that break down with the general concept of traditional row-oriented databases. They filter out unnecessary areas and thus provide flexibility and, above all, quick access.

Traditional systems store data in structured relational database management systems, file systems, and replication. Intensive and extensive research explores various aspects of cloud storage. However, mitigating the risks of integrity, availability, and reliability has not been appropriately addressed in the scientific literature.

Performance, resiliency, reliability, and scalability are important factors in the big data processing. The storage infrastructure must provide reliable storage space with a robust access interface for querying and analysis. Distributed storage can be based on multiple clouds. Typically, data is divided into several parts that must be stored on different clouds to ensure availability in the event of a failure. However, distributed storage failures can cause inconsistency between different copies of the same data [11]. You can use large databases. In this case, for high performance, data processing and analysis should be performed in parallel.

Table 1. Overview of Distributed Cloud Storage Methods

Method	Properties						
	Availability	Confidentiality	Homomorphic	Integrity	Privacy	Reliability	Scalability
Abu-Libdeh [13]	+					+	
Adya [14]	+	+		+	+	+	+
Ateniese [15]		+		+	+	+	
Bessani [16]	+	+		+	+	+	
Bowers [17]	+			+	+	+	+
Celesti [9]	+			+	+	+	
Dimakis [18]		+		+	+	+	+
Erkin [19]			+		+		+
Gentry [20]			+		+		
Ghemawat [11]	+	+				+	+
Gomathisankaran [21]		+	+	+	+	+	+
Kong [22]		+		+	+	+	+
Li [23]	+	+		+	+	+	+
Lin [24]	+	+		+	+	+	+
Pang [25]	+	+		+	+	+	+
Parakh 2011 [26]	+	+		+	+	+	+
Parakh 2009 [27]		+		+	+	+	+
Ruj [28]	+	+		+	+		
Samanthula [29]			+		+		
Sathiamoorthy [30]		+				+	+
Shah [31]		+				+	+
Wang [32]	+	+		+	+	+	+
Wylie [33]	+	+		+			+
Yang [34]	+	+		+			+

When storing data in the cloud, you should consider reliability, scalability, security, privacy. These features are also crucial for mobile devices where specifications and power consumption are limited. Tchernykh et al., In [12], showed that distributed data storage under uncertainty in cloud computing could use data replication, redundant RNS, erase codes (EC), and regeneration codes (RC).

Table 1 presents the main known methods for organizing distributed data storage in the clouds. A comparison is made on the following properties: reliability, scalability, availability, confidentiality, integrity. The most effective for complexity is the method presented in [11]. However, its main drawback is that the data is stored in the explicit using replication, which leads to limited applicability.

An alternative approach to creating a reliable storage system is to use error correction codes based on redundant RNS, EC codes [18], and RC codes [35]. Table 1 deserves special attention to the distributed data storage scheme [21, 36], which provides data security, integrity, reliability, and scalability. The authors proposed two approaches to building systems based on data distribution schemes in redundant RNS. The user stores redundant RNS modules.

Data processing leads to an exponential increase in network and memory load and makes it inapplicable in practice [37].

RRNS represents the original numbers as residuals for the set of modules. Thus, the number is broken up into smaller numbers that are independent.

In paper [9], the authors suggested using RRNS for reliable and scalable cloud storage systems. Operations can be performed in parallel, which simplifies and speeds up calculations. The redundancy of the system allows us to build a network with multiple error detection and correction.

A common problem for most systems is the detection and correction of one error. When reliability is provided for one computer, a single error can be detected and corrected. However, when we consider big data, it is necessary to have efficient algorithms for detecting and correcting several errors [38]. The RRNS scheme for data storage provides reliable and scalable storage. It has properties of error correction codes and the possibility of distributed data storage

3. Methods to improve the reliability and fault tolerance of data storage and processing

The cloud system is a hardware and software system designed to store, organize access, manage, and restore data. Accordingly, potential threats of loss, distortion, or inaccessibility of information may have a physical or software cause.

Usually, during the operation of the cloud system, it is possible to have emergencies that must be taken into account at the level of technology of the cloud system, and these solutions should provide the system with the required level of security and reliability. Consider the most common situations.

One or more of the hard drives in the data warehouse has failed; total or partial data loss. It is necessary to be able to recover data stored on this hard drive.

The motherboard on the server crashes, the server in which it is installed will become unavailable, and all recent changes or the results of the current server operation will be lost. It is necessary to be able to service the incoming load with a given quality, which was initially intended for this server. In this case, you need to restore data that was changed as a result of transactions completed on this server.

Lost communication between the server and the storage loss of all current changes processed on this server. A virus on the server can lead to failures in access to the server, which will lead to the loss of recent changes and stop work. The integrity of the data in the repository, archive logs, the failure of which will exclude the possibility of restoring the lost part of the data, are also at risk. Since virus behavior is unpredictable, the consequences can also be random.

Tools to ensure the resiliency of the cloud system and protect data from loss should have the following properties [39]:

- reliability – all information necessary for data recovery should be stored in a safe place on reliable media and be backed up;
- flexibility – backup should be made so that if necessary it was possible to restore all data in general and specific data files;
- manageability – backup files should be quickly and conveniently managed so that recovery can be performed as soon as possible;
- availability – reservation work should not interfere with the work under any circumstances. Also, the restoration work should not be visible to the user.

To ensure fault tolerance, the following neutral approaches are used.

Offline reservation. The easiest way to backup data: all incoming load is served by the central instance, from which recovery points are removed with a specific frequency by completely copying the data to the medium.

If a failure occurs, the last (or later) recovery point is uploaded to the data server (this can be the main or backup copy of the data), and all incoming load is switched to it.

In this case, three schemes are possible:

- offline backup is performed on a separate medium of the same server;
- offline backup is performed on the media of a separate server designed to backup company information over the network (most likely, it cannot be used as a server for storing and processing data due to limited resources)
- offline backup is performed on a separate server, which is allocated as a backup/standby server.

Removing recovery points during cold backups can only be done periodically since this implies a shutdown of the primary storage server. Periodic removal of recovery points is performed using database tools or external utilities that copy data files at the operating system level. In the case of external services, access to databases is temporarily blocked, and the control file, log files, and archive log files are copied to the storage medium (hard disk). Thus, upon completion of the described operation, an exact copy of the available data is obtained at the time of stopping work with them.

The disadvantage of using this approach is that the frequency of copying data is limited and, as a rule, coincides with the hours of least load on the cloud server. Backup work has to be done either at night or on weekends, which, in turn, allows rollbacks only to these points in time with a possible loss of information for the entire last day or week.

Offline redundancy finds its application with small capacity data. The data recovery process takes a long time and is a reverse copy of the available information from the backup medium to the main one. And if we take into account that the customer strictly normalizes the duration of the recovery process both operability and normal operation using basic resources, then situations may arise when such an approach is unacceptable.

Continuous removal of recovery points. The approach of continuously removing recovery points is to track the changes that occur with the data in real-time and save the history of changes to a dedicated drive. It is assumed that in this way, any data can be restored to any date specified by the administrator. However, this approach gives rise to many problems that manufacturers have to solve.

For example, with frequent updating of data, the amount of knowledge about changes in the data may exceed the amount of the source data by several times. The good news is that in the event of a failure, only the data of incomplete transactions that existed at the time of the crash are lost. As a compromise, you can use the option of periodically removing data images on a schedule and maintaining a continuous history of changes for any limited period.

Online reservation. This is a whole class of data backup methods, the distinguishing feature of which is that there is no need to stop the working part of the data from performing the backup. In normal mode, all requests are served by the main part, and changes in it are synchronized with the backup. The backup database is most often inaccessible for users or read-only [40, 41].

If a failure occurs in the main part of the data, the entire load switches to the backup instance. You can set any value for the period of removing recovery points, however, since removing a recovery point requires copying the changelog, archiving, transferring it to the backup database and other actions, the performance of the main server decreases noticeably during this. Therefore, it is necessary to select a period for taking points in such a way as to achieve the optimum between a drop in performance and a decrease in data loss in the event of a failure. Often the period of log removal is regulated through the size of the log file, for example, when it reaches 100 MB, it switches to the next log, and the completed one is sent to the backup database. The physical diversity of copies of data copies makes it possible to avoid hardware failure and increases the system's resistance to accidents. This approach can significantly improve the reliability of data storage in the event of failures in one of the data instances. In the simplest way to organize this approach, 100% redundancy of data storage is required, which means that the cost of this solution increases significantly.

In modern systems, a load-balancing approach is often used: several data instances are created, each of which processes part of the requests. When one case fails, the load is distributed among the others. Load balancing reduces storage redundancy.

In practice, a method is used, the essence of which is to store complete copies of the stored information on several system servers. This method guarantees reliability since data can be restored even if one of the servers is operational. However, this method is costly because z replicas lead to an expansion of z time. One way to reduce the expansion speed is to use erase codes to encode messages. The message is encoded as a codeword, which is a character vector, and each server stores a codeword symbol. A storage server failure is modeled as an error with deleting stored codeword characters. Random linear codes support distributed encodings, that is, each character of a codeword is calculated independently. To save a message of sizes of k blocks, each storage server linearly combines blocks with randomly selected coefficients and stores the symbol of the codeword and coefficients.

To receive a message for the stored codeword characters and coefficients, the user requests the storage servers k and solves the linear systems.

To increase reliability in the storage and processing of information in the cloud, an IDA-based approach is used. IDA allows you to share data between several servers (participants) involved in the processing or storage of information. Any set of servers when storing information contains a sufficient amount of information necessary for its recovery. One of the ways to increase the speed, reliability, and fault tolerance of computing tools has been the creation of computing systems based on modular arithmetic, that is, codes in which numbers are represented in RNS.

4. Using a residue number system to increase fault tolerance

The considered cloud system is shown in the figure 1, based on the principles of RNS, has p_i cloud servers that are designed to store and process big data on k working and r control bases (parts). One of the approaches to solving the problem of increasing the reliability of the cloud system is based on the redistribution of parts of the data in case of failure of part of the working or control channels [41, 42, 43]. At the same time, by reliability, we mean the ability of the cloud system to remain operational in case of failure of one or several data processing and storage servers, while reducing, within acceptable limits, some performance indicators. This feature of constructing a model for storing and processing big data allows you to build a cloud system with constant failures of working or control servers operating in the RNS. With continuous failures of the working and control bases, the cloud system goes into the S_V state, which entails the need to redistribute data between servers and find the set $N = \{Q_V\}$ to distribute big data. If for each state of the cloud system, a solution is chosen to redistribute the Q_V data between the cloud servers, the problem is to find the best option for redistributing the data between the cloud servers for each $S_V\ inS$ server. It can be solved as a problem of optimization according to one of the indicators accepted as an objective function with given restrictions on other indicators. Consider a statement on the distribution of data between cloud servers operating in an RNS.

Suppose that during the operation of the cloud system there are failures of cloud servers, and the servers are independent. Let $S(t)$ be the state of the cloud system at time t:

$$S(t) = d_1, d_2, \dots, d_n, \tag{1}$$

where n – number of channels of the cloud system.

$$d_i = \begin{cases} 0, & if \text{ i } server\ is\ operational \\ 1, & if \text{ i } the\ server\ is\ down \end{cases}$$

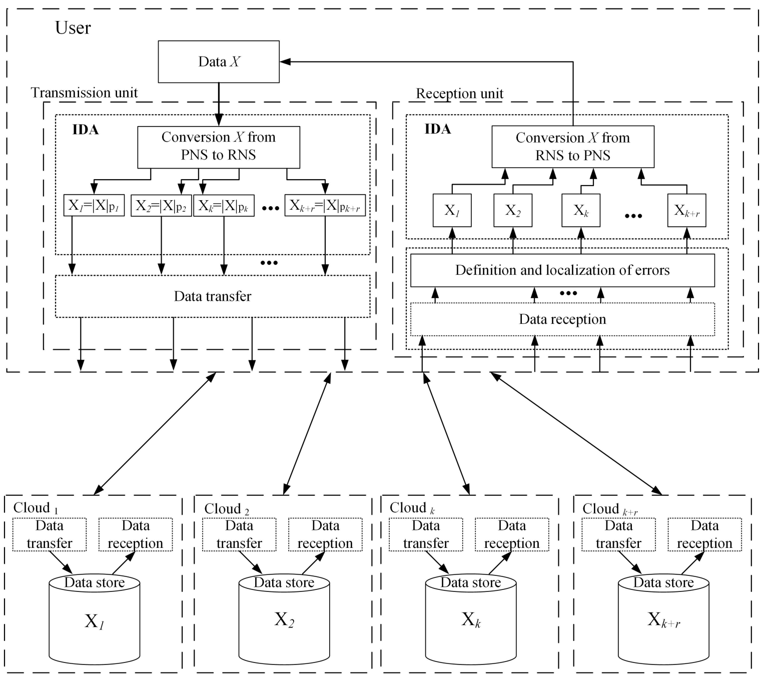

Figure 1. Distributed Storage Scheme

Consider the time interval $[t_0, t_z]$. Let $S(t_0) = 0, 0, \ldots, 0$ – initial state of the cloud system, the initial distribution of the system is equal to

$$A_{0_i} = \{|\alpha_1|_{p_i}, |\alpha_2|_{p_i}, \ldots, |\alpha_L V|_{p_n}\},$$

where $|\alpha_l|_{p_i}$ – data corresponding to i-cloud server; $i = \overline{1, n}$; $l = \overline{1, L}$; L – number of tasks. Multiple $P = \{p_1, p_2, \ldots, p_n\}$ is a set of all working and control servers in the cloud system.

During the operation of the cloud system at a time t_k ($t_0 < t_k \leq t_z$) some servers may fail. Then the P set can be split into two subsets:

- $P^O_{t_k}$ – a subset of all failed servers in the cloud system.
- $P^P_{t_k}$ – a subset of all working servers in the cloud system.

Let S_V be the state at t_k; then

$$A_V = \{|\alpha_1|^+_{p_i}, |\alpha_2|^+_{p_i}, \ldots, |\alpha_L V|^+_{p_i}\},$$

where $i = \overline{1, n_p}$ – a lot of all data that provides acceptable limits on the quality of stored data that can be stored in the cloud system in the state S_V, in other words, for processing which the cloud system has the necessary number of servers, $n_p = n - n_O$ – number of cloud servers in the state S_V, n_O – number of failed cloud servers.

$$A_i = \{\left||\alpha_j|^+_{p_1}\right|^+_{p_i}, \left||\alpha_j|^+_{p_2}\right|^+_{p_i}, \ldots, \left||\alpha_j|^+_{p_n}\right|^+_{p_i}\},$$

where $j = \overline{1, L_V}$, $\left|\alpha_j|^+_{p_1}\right|^+_{p_i} \in A_V$ – a subset of data that a functioning cloud server can store and process if the cloud system is in the S_V state.

In a cloud system with persistent failures, efficient cloud servers are used for data storage and processing. If a server failure occurs, the cloud system is reconfigured to exclude all failed cloud servers. In this case, a new redistribution between working cloud servers of data that have not been sent to the outgoing servers. When a cloud system switches from $S(t_0)$ to $S(t_k) = S_V$, various decisions can be made regarding specific failed and operational cloud servers, which are determined by the importance of data and operational servers.

Let's enter the following designations: A_V^O – a set of own failed cloud server data for the state S_V; A_V^P – a set of all own failed cloud server data for the state S_V.

Let's enter the following designations: A_V^O – a set of own failed cloud server data for the state S_V; A_V^P – a set of all own failed cloud server data for the state S_V.

- U_O^C – all data in the A_O^P set are stored in the cloud if $n_O \leq l$, where $l = \frac{d_{min}}{2}$ – number of corrected errors (parts), d_{min} – minimum code distance of redundant RNS code;
- U_O^O all data in the A_O^P set is saved, and the part is discarded if $1 < n_O < n_{O,extr}$, where $n_{O,extr}$ is the allowed number of failed cloud servers;
- U_O^O part of data set A_O^P is saved, and part is discarded if $n_O = n_{O,extr}$.

Let's consider failed cloud servers dummy and equate their data to zero. Simultaneously with the data of A_P^V set, one of the following decisions is made:

- U_P^C – all data of A_P^V set are saved, and the task continues;
- U_P^{C-I} – all data of A_P^V set is stored in the cloud system, part of data is stored and processed on assigned servers, and part is redistributed.

When the inequality $n_O > n_{O,expr}$ is executed, a complete failure of the cloud system occurs. The set of U of possible solutions that can be made when redistributing data in the state S_V is determined by the set of all possible combinations of solutions for these subset A_P^V, A_O^V considered above:

$$U = \left\{(U_O^C U_P^O), (U_C^O U_P^{C-I}), (U_O^O U_P^C), (U_O^O U_P^{C-I}), (U_O^{C-O} U_C^P), (U_O^{C-O} U_P^{C-I})\right\} \tag{2}$$

Let's consider refusals on working and control channels which are possible at the transition of cloud system constructed based on RNS, in a condition S_V and change of indicators of quality of functioning of cloud system at data redistribution in it according to one of the decisions U, table 2.

In case the decision $U_1 = U_O^C U_P^C$ is made, the working channels process all data of the set, and the obtained result of the operation with some distortion is restored due to the correcting properties of the code, i.e., in an algorithmic way.

Solution $U_2 = U_O^C U_P^{C-I}$ does not make sense (failure of the whole system) because saving data of A_V^O set does not allow to redistribute data of A_V^P set. Other solutions are used to organize data redistribution in the cloud system.

The implementation of any of these solutions involves excluding failed servers from the cloud system by blocking their inputs and outputs and redistributing data between work servers. In this case, the cloud system assumes the availability of additional software, hardware, and time resources [42, 44].

Let's consider indicators of functional capacity (E) and a time indicator of a separate server (T) as indicators of quality of work of cloud system to which certain requirements are made at the realization of data redistribution.

Nominal E^h, T^h and the maximum allowable E^{extr}, T^{extr} values of accepted performance quality indicators determine the area of cloud system performance as a subset of $M_p = \{y_\mu\}$ of such y_μ states for which

$$E^h \geq E_\mu \geq E^{extr} \, and \, T^h \geq T_\mu \geq T^{extr}. \tag{3}$$

Table 2. Dependence of the indicators of the cloud system operating in the RNS on the decision made when the working and control channels are degraded

Decision	Server failures		*Indicators*[1]	
	Works	Control	Reliability	Accuracy
$U_1 = U_O^C U_P^C$	$-$	$+$	0	*
	$+$	$-$	0	*
	$+$	$+$	0	*
$U_2 = U_O^C U_P^{C-I}$	$-$	$+$	System crash	
	$+$	$-$		
	$+$	$+$		
$U_3 = U_O^O U_P^C$	$-$	$+$	0	*
	$+$	$-$	0	*
	$+$	$+$	*	*
$U_4 = U_O^O U_P^{C-I}$	$-$	$+$	0	0
	$+$	$-$	0	1
	$+$	$+$	0	0
$U_5 = U_O^{C-O} U_P^C$	$-$	$+$	0	0
	$+$	$-$	0	1
	$+$	$+$	0	0
$U_6 = U_O^{C-I} U_P^{C-I}$	$-$	$+$	0	0
	$+$	$-$	0	1
	$+$	$+$	0	*

1) 0 – index reduction, 1 – index increase, * – retention of this indicator

Let us formulate the task of ensuring the health of the cloud system in case of failure of some cloud servers. Let the known structure of the cloud system working based on the RNS with the specified cloud servers perform the algorithm of data storage and processing, presented as a set of A_{0_i} and the initial state for all servers of the system S_0. Then, in case of failure of some part of servers, the task is to find the strategy of data redistribution between working servers with ensuring the fulfillment of requirements to reliability indicators of the cloud system. There are two main types of data redistribution organization: statistical and dynamic redistribution.

The statistical method assumes that before the beginning of the cloud system for a given subset $S = S_V$, its state in memory is optimal plans for the distribution of cloud servers.

When a cloud system switches to the state $S_V \in S$ in its cloud servers, processing of the data corresponding to the distribution G_V begins. If a subset of S cannot be specified or its size requires an unacceptably large amount of memory to store all valid programs, dynamic channel redistribution is applied. The rebuilding time, in this case, may significantly exceed the rebuilding time in static mode.

A combination of static and dynamic data redistribution methods is possible. Its essence is that at the first stage static data redistribution is performed for some predetermined set of system states $S_1(t)$, and at the second stage the most probable transitions $S_1(t) \to S_2(t)$ are defined, a new plan of data redistribution for all possible states is formed. I take into account the requirements for cloud systems in terms of performance. We will use statistical redistribution of data from failed servers of the cloud system between working servers.

5. Comparative assessment of cloud reliability of operating in the residue number system

At construction of a mathematical model of reliability of a cloud, the following properties of RNS are considered:

- independence of data parts, which allows you to process data parts and consider them independent elements independently;
- equality of data parts, which makes it possible to consider redundant channels as reserve elements for the rest;
- low discharge, which causes insignificant variation in the reliability characteristics of the clouds.

For the analysis of the reliability of cloud functioning based on RNS, it is expedient to use mathematical ratios of reliability theory and different ways of introducing structural redundancy. When calculating reliability characteristics, it is assumed that

- at the beginning of time there are $n = k + r$ clouds, where k is the number of workers, r is the number of control clouds;
- the minimum number of clouds (parts) required for proper data recovery is $n - (r + 1)$; failure of more than $r - 1$ clouds is considered a failure of the entire cloud system;
- cloud failures are statistically independent events.

The task of building a mathematical model of reliability can be formulated as follows.

The data submitted to the RNS and transmitted to the cloud for processing and storage are converted using k workers and r controls. If working parts are lost or unavailable, they can be replaced by the control base.

The requirement to provide guaranteed protection from the issuance of an unreliable result determines the need to keep in working order at least one control and k working parts.

In this way, robust cloud storage and processing structure will be matched by a sliding redundancy method in which the backup elements are loaded. When assuming the simplest element failure flow and their equal reliability, given that the failed elements are not recovered, we come to the expression to calculate the probability of failure of the next type:

$$R_1 = \sum_{i=0}^{r-1} P'^{k+r-i} \left(1 - P'\right)^i , \tag{4}$$

where R_1 is the probability of failure of a cloud system based on a single-level RNS, for a given time, P is the probability of failure of one cloud.

$$R_2 = \sum_{i=0}^{r-1} P'^{k+r-i} \left(1 - P'\right)^i , \tag{5}$$

$$R_{1,2} = R_1 \cdot R_2, \tag{6}$$

where $R_{1,2}$ is the probability of failure of a cloud system based on a two-level RNS within a given time, P is the probability of failure of one cloud.

When allocating the cloud operation time before failure according to the exponential law, the probability of no-failure operation within a given time will be determined as:

$$P'(t) = e^{-\lambda_{TP} \cdot t}. \tag{7}$$

Then

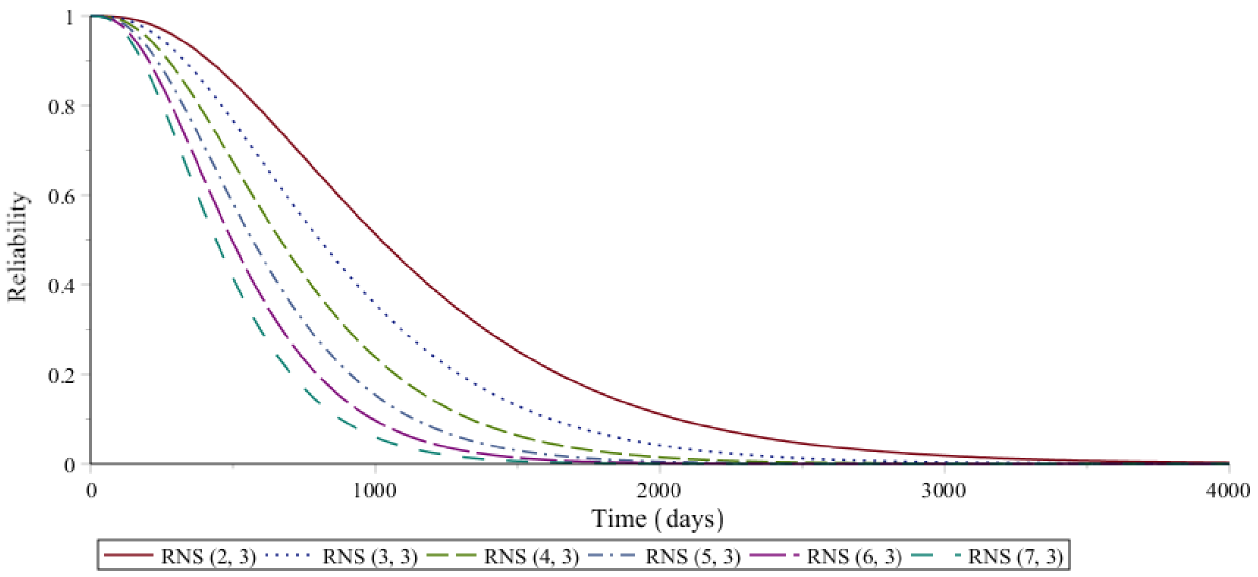

Figure 2. Dependence of the probability of failure-free operation of the system on the ratio of the number of workers (first digit in brackets) and control (second digit) clouds at failure rate $\lambda = 8 \cdot 10^{-5}$

- the failure rate can be considered a constant value;
- there are no other input data for the calculation than the failure rate;
- The results obtained are suitable for engineering assessments of fault tolerance.

Let's denote the failure rate of one cloud λ_P, and select as an average number of bits in a particular system

$$\gamma_{cp} = \left[\sum_{i=1}^{n} \frac{\gamma_i}{n} \right], \tag{8}$$

where n – number of clouds (RRNS bases) in the system, γ_i – number of bits in the n channel, [ ] – whole number, define

$$\lambda_{TP} = \gamma_{cp} \cdot \lambda_{P'}, \tag{9}$$

where λ_{TP}– single cloud failure rate.

Calculation data for the system with different ratio of the number of working and control bases figure 2, indicate a significant gain in reliability when using the reserve.

Of greatest interest is the comparison on the selected indicator of systems functioning on the basis of RNS, PNS and paper [11], and providing equal ranges of representation of numbers. Then at intensity of refusals of one digit equal to λ_P, intensity of refusals of PNS

$$\lambda_{nc} = \gamma_{nc} \lambda_P, \tag{10}$$

where λ_{nc} =32 is the number of bits in the position processor.

Definition and operative correction of erroneous results using positional systems are possible only under the condition of simultaneous work of several computing devices on a principle of voting; for comparison, the majority computing structure functioning on principle "2" from "3" and providing masking of single failures and failures is chosen. The probability of non-failure operation of such a system without taking into account the reliability of the voting element is set by the expression [42].

$$P_{PNS} = 3P'^2_{PNS} - 2P'^3_{PNS}, \tag{11}$$

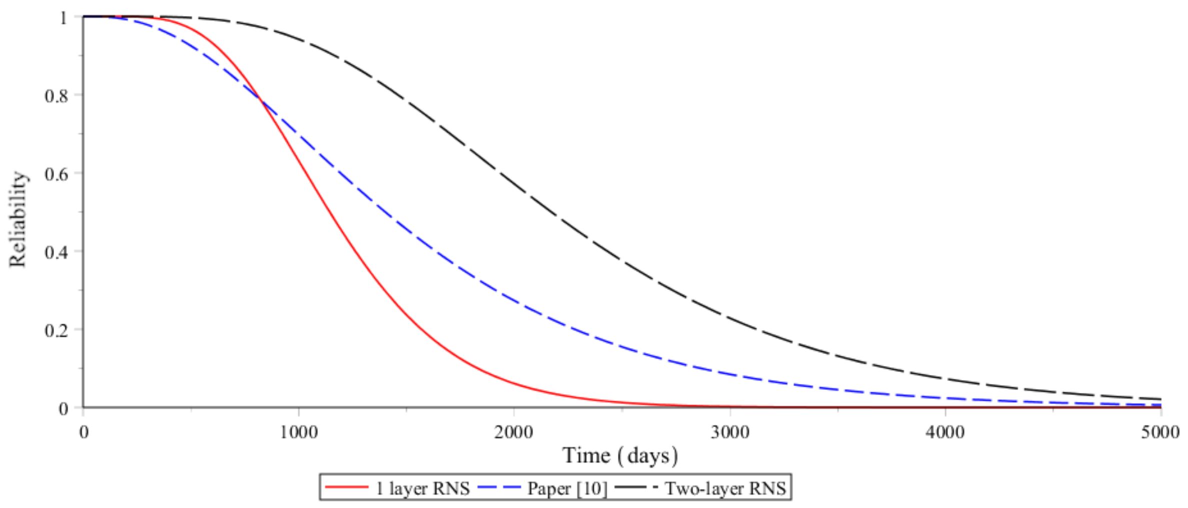

Figure 3. The graph of comparative estimation of probability of non-failure operation of the system built using various algorithms

where P_{PNS} is the probability that the position processor will fail.

Figure 3 shows a graph of the different processing and storage systems in the cloud. The best characteristics of the probability of trouble-free operation is a system based on a two-level RRNS, which allows you to maintain high reliability, in 1000 days after starting work, it is 94%.

Now let's define the M average time to failure of a cloud system operating in RRNS (time for which the γ clouds will fail), as well as the expected time of M_1 of the first failure in this cloud:

$$M = \sum_{i=n-r+1}^{n} \frac{1}{\lambda_{TP} \cdot i}; \quad M_1 = \frac{1}{\lambda_{pm}}. \tag{12}$$

In this case, you can use the normalized value of M:

$$m = \frac{M}{M_1} = \sum_{i=n-r+1}^{n} \frac{1}{i} \tag{13}$$

The analysis of the received dependencies testifies to advantage in the reliability of algorithms of storage and processing of data based on RRNS over existing systems at an essential gain in redundancy. Thus, at the use of algorithms of processing and data storage based on RRNS, it is possible to receive a considerable increase in reliability of system without additional hardware expenses at the decrease in redundancy.

Reliability assessment of data storage and processing algorithms in the cloud environment built based on RRNS, and their comparison with similar on qualitative functional characteristics of existing positioning systems indicates their significant advantage, which is explained by the presence of the effect of elementary sliding redundancy. The ratio of redundant equipment of positional and modular schemes of data processing and storage for various parameters of the parried failures.

When running a cloud system for a long time, it is important to know the number of failures expected in a particular period. This data is necessary to select the best configuration for a multi-cloud system. To determine the number of random events over a certain period, we apply the Poisson distribution. The complete list of probabilities is as follows: probability of zero bounces – $e^{-\lambda \cdot t}$, probability of one failure – $\lambda \cdot t^{-\lambda \cdot t}$, two bounces – $\frac{(\lambda \cdot t)^2}{2!} \cdot e^{-\lambda \cdot t}$, three bounce

– $\frac{(\lambda\cdot t)^3}{3!}\cdot e^{-\lambda\cdot t}$, n bounce – $\frac{(\lambda\cdot t)^n}{n!}\cdot e^{-\lambda\cdot t}$. To get the number of months with 1, 2,... ...you need to multiply the specified probabilities by the number of periods of T on the considered period t. When calculating the Poisson distribution, there are the same limitations as for the exponential distribution.

The analysis of the table data shows that within 35.3 months, the RRNS-based multi-cloud will operate smoothly, almost for the entire period of operation.

Let's assume that a single failure leads to a failure of one cloud server; then, the number of failures will be determined by the number of erroneous bits of the final calculation. The obtained relations allow choosing the redundant RRNS code with two control bases detecting two and correcting one error. The error detection must be done after each final result and, therefore, must not affect performance. Error correction and restoration of the correct result with the obtained reliability characteristics will occur quite rarely. The carried out estimation of reliability of cloud system functioning based on RRNS with reconfigurable structure and comparison with systems similar to qualitative aspects testify to their advantage.

Conclusions

In this paper, the architecture of a multi-cloud data storage system based on the principles of modular arithmetic is developed. The offered modification of the cloud system of data storage and processing allows increasing reliability and fault tolerance of data storage. To increase the fault-tolerance of developed methods in case of cloud server failure, redistribution of processed data between available servers is applied. The introduction of small redundancy allows making processing or restoration of the stored data in case of failure of control servers. This model allows you to restore stored data in case of failure of one or more cloud servers. An adaptive data storage system based on the residual number system, error correction codes, and data distribution schemes has been developed. It provides a theoretical basis for calculating the probability of information loss, data redundancy, encoding/decoding speed, and configuration parameters to meet different object preferences, workloads, and cloud properties.

Acknowledgements The reported study was funded by Russian Federation President Grant MK-24.2020.9, MK-341.2019.9, SP-2236.2018.5 and SP-3149.2019.5

References

[1] Bowers K D, Juels A and Oprea A 2009 *Proceedings of the 16th ACM conference on Computer and communications security* pp 187–198

[2] Babenko M, Kucherov N, Tchernykh A, Chervyakov N, Nepretimova E and Vashchenko I 2017 *Proceedings of the Third International Conference BOINC: FAST 2017* pp 77–84

[3] Grossman R L, Gu Y, Sabala M and Zhang W 2009 *Future Generation Computer Systems* **25** 179–183

[4] Grossman R and Gu Y 2008 *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* pp 920–927

[5] Rabin M O 1989 *Journal of the ACM (JACM)* **36** 335–348

[6] Tchernykh A, Miranda-López V, Babenko M, Armenta-Cano F, Radchenko G, Drozdov A Y and Avetisyan A 2019 *Cluster Computing* **22** 1173–1185

[7] Asmuth C and Bloom J 1983 *IEEE transactions on information theory* **29** 208–210

[8] Mignotte M 1982 *Workshop on Cryptography* (Springer) pp 371–375

[9] Celesti A, Fazio M, Villari M and Puliafito A 2016 *Journal of Network and Computer Applications* **59** 208–218

[10] Chervenak A, Deelman E, Foster I, Guy L, Hoschek W, Iamnitchi A, Kesselman C, Kunszt P, Ripeanu M, Schwartzkopf B *et al.* 2002 *SC'02: Proceedings of the 2002 ACM/IEEE Conference on Supercomputing* (IEEE) pp 58–58

[11] Ghemawat S, Gobioff H and Leung S T 2003 *Proceedings of the nineteenth ACM symposium on Operating systems principles* pp 29–43

[12] Tchernykh A, Schwiegelsohn U, Talbi E g and Babenko M 2019 *Journal of Computational Science* **36** 100581

[13] Abu-Libdeh H, Princehouse L and Weatherspoon H 2010 *Proceedings of the 1st ACM symposium on Cloud computing* pp 229–240

[14] Adya A, Bolosky W J, Castro M, Cermak G, Chaiken R, Douceur J R, Howell J, Lorch J R, Theimer M and Wattenhofer R P 2002 *ACM SIGOPS Operating Systems Review* **36** 1–14

[15] Ateniese G, Fu K, Green M and Hohenberger S 2006 *ACM Transactions on Information and System Security (TISSEC)* **9** 1–30

[16] Bessani A, Correia M, Quaresma B, André F and Sousa P 2013 *Acm transactions on storage (tos)* **9** 1–33

[17] Bowers K D, Juels A and Oprea A 2009 *Proceedings of the 16th ACM conference on Computer and communications security* pp 187–198

[18] Dimakis A G, Ramchandran K, Wu Y and Suh C 2011 *Proceedings of the IEEE* **99** 476–489

[19] Erkin Z, Veugen T, Toft T and Lagendijk R L 2012 *IEEE transactions on information forensics and security* **7** 1053–1066

[20] Gentry C 2010 *Communications of the ACM* **53** 97–105

[21] Gomathisankaran M, Tyagi A and Namuduri K 2011 *2011 45th Annual Conference on Information Sciences and Systems* (IEEE) pp 1–5

[22] Kong Z, Aly S A and Soljanin E 2010 *IEEE Journal on Selected Areas in Communications* **28** 261–267

[23] Li M, Lou W and Ren K 2010 *IEEE Wireless communications* **17** 51–58

[24] Lin H Y and Tzeng W G 2011 *IEEE transactions on parallel and distributed systems* **23** 995–1003

[25] Pang L J and Wang Y M 2005 *Applied Mathematics and Computation* **167** 840–848

[26] Parakh A and Kak S 2011 *Information Sciences* **181** 335–341

[27] Parakh A and Kak S 2009 *Information Sciences* **179** 3323–3331

[28] Ruj S, Nayak A and Stojmenovic I 2011 *2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications* (IEEE) pp 91–98

[29] Samanthula B K, Elmehdwi Y, Howser G and Madria S 2015 *Information Systems* **48** 196–212

[30] Sathiamoorthy M, Asteris M, Papailiopoulos D, Dimakis A G, Vadali R, Chen S and Borthakur D 2013 *arXiv preprint arXiv:1301.3791*

[31] Shah N B, Rashmi K, Kumar P V and Ramchandran K 2011 *IEEE Transactions on Information Theory* **58** 2134–2158

[32] Wang C, Wang Q, Ren K, Cao N and Lou W 2011 *IEEE transactions on Services Computing* **5** 220–232

[33] Wylie J J, Bigrigg M W, Strunk J D, Ganger G R, Kiliccote H and Khosla P K 2000 *Computer* **33** 61–68

[34] Yang C C, Chang T Y and Hwang M S 2004 *Applied Mathematics and Computation* **151** 483–490

[35] Lin S J, Chung W H and Han Y S 2014 *2014 ieee 55th annual symposium on foundations of computer science* (IEEE) pp 316–325

[36] Babenko M, Tchernykh A, Chervyakov N, Kuchukov V, Miranda-López V, Rivera-Rodriguez R, Du Z and Talbi E G 2019 *Programming and Computer Software* **45** 532–543

[37] Tchernykh A, Babenko M, Chervyakov N, Miranda-López V, Kuchukov V, Cortés-Mendoza J M, Deryabin M, Kucherov N, Radchenko G and Avetisyan A 2018 *International Journal of Approximate Reasoning* **102** 60–73

[38] Liu D T and Franklin M J 2004 *Proceedings of the International Conference on Very Large Data Bases* pp 600–611

[39] Tchernykh A, Babenko M, Chervyakov N, Cortés-Mendoza J M, Kucherov N, Miranda-López V, Deryabin M, Dvoryaninova I and Radchenko G 2017 *2017 28th International Workshop on Database and Expert Systems Applications (DEXA)* (IEEE) pp 137–141

[40] Amrhein D and Quint S 2009 *DeveloperWorks, IBM* **8**

[41] Tchernykh A, Babenko M, Chervyakov N, Miranda-López V, Avetisyan A, Drozdov A Y, Rivera-Rodriguez R, Radchenko G and Du Z 2020 *IEEE Internet of Things Journal*

[42] Chervyakov N, Babenko M, Tchernykh A, Kucherov N, Miranda-López V and Cortés-Mendoza J M 2019 *Future Generation Computer Systems* **92** 1080–1092

[43] Babenko M, Chervyakov N, Tchernykh A, Kucherov N, Shabalina M, Vashchenko I, Radchenko G and Murga D 2017 *2017 28th International Workshop on Database and Expert Systems Applications (DEXA)* (IEEE) pp 147–151

[44] Kesselman C and Foster I 2004 *Grid: Blueprint for a New Computing Infrastructure (Elsevier Series in Grid Computing)* (Morgan Kaufmann Publishers)