Adv Syst Sci Appl 2024; **02**:166–186
Published online at https://ijassa.ipu.ru.

Data Storage with Increased Survivability and Reliability Based on the Residue Number System

Nikolay Kucherov[1]*, Mikhail Babenko[1], Egor Shiriaev[1], Nguyen Viet Hung[2]

[1]*North-Caucasus Federal University, Stavropol, Russia*
[2]*Le Quy Don Technical University, Hanoi, Vietnam*

Abstract: The paper considers the principles of building reliable data storage and processing systems in cloud-fog environment. The factors affecting the reliability and survivability of systems are an-alyzed and the main types of failures are given. The principles of introducing information re-dundancy to improve reliability, as well as corrective capabilities and error detection algorithms of RNS codes are considered. A scheme of data processing and storage in fog with increased re-liability is proposed. Monte Carlo modeling is carried out, and the results and conclusions are given, the probability distribution of failure occurrence is constructed, showing that failure probabilities obey the gamma probability distribution.

Keywords: fog-cloud computing, residue number system, error correction, reliability, fault tolerance

1. INTRODUCTION

As the volume of stored data continues to grow, an increasing number of users are transitioning to cloud storage solutions. In today's context, the demands for data storage reliability within cloud environments are consistently escalating. Reliability in cloud storage refers to its capacity to uphold data management and retention, all while upholding predefined quality benchmarks within operational conditions over time. This concept primarily encompasses the influence of intersystem factors, particularly the occurrence of unforeseen technological breakdowns, on the performance of cloud storage systems [1].

Fog computing, often referred to as fog networks or edge computing, represents a decentralized computing model that draws data processing and storage closer to the devices and individuals responsible for generating and utilizing the data. It entails processing data at the network's periphery, in close proximity to the data's origin, rather than relying on a central server like a data center or cloud infrastructure. This approach notably di-minishes the volume of data necessitating transmission through the network, effectively curbing latency, bandwidth demands, and associated expenses. It also empowers re-al-time processing, expediting decision-making processes and enhancing overall respon-siveness [2, 3].

Fog computing is particularly useful in the Internet of Things (IoT) and other applications where devices are distributed and have limited processing power and connectivity. They allow these devices to process data locally and make decisions autonomously, without the need to constantly interact with the cloud. This can also improve security and privacy, as sensitive data can be processed and stored locally rather than being transmit-ted over a network. Fog computing also provides a bridge between the cloud and the net-work edge, allowing data

*Corresponding author: nkucherov@ncfu.ru to be processed and analyzed at different levels of granularity depending on the application requirements. They can be used in a variety of applications such as industrial control systems, smart city, transportation and healthcare [4].

One of the key advantages of edge computing lies in its capability to swiftly gather and analyze data in actual time, facilitating accelerated decision-making and heightened responsiveness. Moreover, edge computing can serve to shift computational and storage burdens away from cloud environments, subsequently mitigating the expenses and de-lays typically linked with cloud-based solutions. Nevertheless, this approach also intro-duces novel hurdles, including issues related to security, unpredictability, efficient data handling, and seamless deployment.

Fog computing infrastructure can include various devices such as routers, gateways, and embedded systems. These devices are capable of running various applications and providing services in a decentralized manner, in interaction with each other and with the cloud [5].

The survivability of a cloud-fog system refers to the robustness of its control and transceiver system against the action of causes from outside aimed at disabling the storage, as well as resistance to cascading failures.

The causes of reduced survivability are divided into two types: spontaneous, deliber-ate, and suboptimal in terms of the choice of technical solutions in the construction of the system.

The notions of reliability and survivability share commonalities while also present-ing notable distinctions. What binds them is the underlying principle of ”sustain-ability,” which encompasses a wide range of factors, encompassing diverse failures that can transpire. The sustainability index emerges as a composite metric that incorporates aspects of reliability, survivability, and fault tolerance indices.

Discrepancies between the concepts of reliability and survivability stem from distinct ways they manifest and their varying natures within the context of cloud and fog systems. Factors leading to disruptions in their regular functioning vary significantly, including the nature and scope of failures, their duration, and the strategies employed for rectification and augmenting fault tolerance. Consequently, the foundational data, calculation methodologies, precision, and fundamental nature of reliability and survivability metrics di-verge substantially. The former group benefits from substantial statistical data, encompasses essential influencing factors, and boasts robust theoretical foundations. They prove useful for reasonably precise projections, calculations, design work, and simulations. On the other hand, the cluster of survivability metrics tends to convey a qualitative perspective on the behavior of the examined cloud-fog system when subjected to external impacts or the sequential spread of failures. Now, let’s delve into the primary factors that impact system reliability:

- Mistakes associated with incorrect technical choices when designing and building a cloud storage system;
- Hard disk failures;
- Technical factors, factors that depend on the structure of the object and its operating modes, the use of redundancy, the organization of control and recovery after failure, the characteristics of component elements, the protection of elements from adverse factors, the quality of technological processes in the manufacturing process, the degree of adaptability for operation;
- Program factors depend on the quality of microprocessor software development;
- Operational factors depend on the external environment surrounding the object during operation;
- Climatic factors, temperature, humidity, solar radiation, wind loads, etc.;
- The impact of maintenance on reliability is determined by the fact that facilities are often automated rather than automatic systems. A human being is a kind of link that fits into the structure of the system.

Relationship between man and technology. With the development of technology, the complication of its structure, the expansion of its functions, the question of the relationship between man and technology is becoming more and more acute.

The paper is further organized as follows. Section 2 analyzes the factors affecting the reliability and survivability of control and communication systems. In Section 3, the types of redundancy and redundancy introduction in RNS are discussed. Corrective capacity estimation and error correction algorithms in RNS are given. Section 4 deals with structural redundancy in cloud storage systems. Section 5 considers the reliability of nonrepairable redundant objects and redundancy methods. Section 6 is devoted to the principles of fault tolerant cloud systems. Section 7 gives the structure of fault tolerant storage system in cloud fog environment. Section 8 deals with the structure of a fault tolerant data processing system in fog environment. Section 9 is devoted to the modeling of the proposed system for reliable data storage and processing in fog environment and the results obtained are described.

2. ANALYSIS OF FACTORS AFFECTING RELIABILITY AND SURVIVABILITY OF CONTROL AND COMMUNICATION SYSTEMS

The reliability of both cloud and fog storage systems becomes evident through instances of failures. The notion of failure is intricately connected to that of operability. Operability represents the condition of a system in which it can execute designated functions in alignment with specified parameters and the quality demands of the services rendered. Failure constitutes an arbitrary occurrence characterized by the disruption of the system's operability. Failures can be categorized as follows:

- Sudden – system parameters change instantly;
- Gradual – gradual change of system parameters due to wear and tear or aging;
- Work-in-progress – presence of defective elements, assembly and installation errors.

Enhancing the reliability of cloud and fog systems can be achieved through the utilization of mathematical tools, standardization practices, load distribution techniques, safeguarding against external influences, and the strategic selection of data storage schemes [6]. If the above-mentioned methods did not give the desired result, then redundancy should be used.

At the stage of development of models and schemes of data storage the main reliability characteristics of the future system are laid down. It is necessary to reasonably choose the mathematical apparatus, the accuracy of the system, redundancy and, consequently, the optimal form of data representation (data packets) during their transmission and processing. The choice of the number system essentially and determines the reliability model of cloud storage. When building cloud storage, alternative non-positional and ad hoc number systems, which are oriented to solve a wider range of problems, are often not considered. In addition to applying mathematical approaches to improve reliability and survivability, structural approaches are also used, which consist in improving reliability by introducing additional redundancy and applying redundancy. The use of information methods in the form of corrective codes can improve the reliability and survivability of data storage systems.

Let's examine the elements influencing the survivability of cloud systems. A notable distinction between evaluating survivability and other comparable concerns, like assessing reliability, is the general inability to apply probability concepts for the occurrence of specific scenarios. To enhance survivability, it becomes feasible to adopt a numerical system where the failure of a specific component does not result in the complete cessation of the entire system's operation.

It should be borne in mind that the number system significantly affects the algorithms of operation, exchange protocols, technical characteristics, reliability and survivability of the data storage system. For example, the decimal number system is irrational for use in computer technology, but is optimal for human perception of data.

Hence, the significance of selecting the operational numbering system for cloud storage systems is heightened. This choice primarily shapes the system's reliability model, redundancy methodology, counteracting cascading failures, and the mitigation of accumulating errors. An intrinsic qualitative measure of survivability is the system's resilience, denoted by E survived, subsequent to a predefined series of influences. For cloud and fog systems, the main reliability metrics are:

1. Probability of uptime operation in the time interval from 0 to t_0
$$p(t_0) = p(0; t_0) = p\{\varepsilon \geq t_o] = 1 - F_1(t_0),$$
where $F_1(t) = p[\varepsilon_1 \leq t]$ – distribution of failures in time to first failure;
2. Probability of uptime operation of the object in the time interval from t to $t + t_0$
$$p(t, t + t_0) = p\{\varepsilon_1 \geq t + t_0 | \varepsilon_1 > t] = \frac{p(0, t + t_0)}{p(0, t)} = \frac{p(t + t_0)}{p(t)};$$
3. Failure distribution density
$$f(t) = \frac{d}{dt}F(t) = -\frac{d}{dt}p(t);$$
4. Intensity of object failures at the moment of time
$$\mu(t) = \frac{1}{1 - F(t)} \cdot \frac{d}{dt}F(t);$$
5. Average time to failure
$$T_i = M(\varepsilon_1) = \int_0^\infty x dQ(x) = \int_0^\infty p(x) dx.$$

Presently, the primary pragmatic approach to enhance the dependability and survivability of storage systems involves the implementation of redundancy and error correction codes.

Redundancy entails augmenting the reliability of a cloud system by incorporating supplementary clouds beyond the minimum necessary for the system's regular functioning.

The main types of reservations:

- Time redundancy – utilizing excess time to accomplish tasks;
- Functional redundancy – the use of additional elements with which functions can be performed autonomously;
- Structural redundancy – a method of increasing reliability by introducing additional redundant elements included in the overall system structure;
- Information redundancy – improving reliability by storing multiple copies of data.

Optimal stability, reliability, and survivability emerge when a system seamlessly integrates and interweaves these methods. This state is exemplified by the Residue Number System (RNS), which encompasses these attributes. When constructing cloud storage systems utilizing RNS, a blend of redundant and operational clouds will be present (for each RNS base), facilitating the harmonious incorporation of the various redundancy strategies outlined earlier.

3. INFORMATION REDUNDANCY IN CLOUD STORAGE SYSTEMS

Information redundancy is one of the main types of redundancy that is used to ensure the reliability and availability of information. It is the process of backing up data in order to restore it in case of data loss or corruption [7, 8].

The main task of information redundancy is to guarantee the safety of data in case of equipment failure, operator errors, natural disasters or other unforeseen circumstances.

There are various methods of information redundancy, including.

Full Backup. With this method, a complete copy of all data is made, which requires a large amount of disk space and time to create a backup. However, data recovery after a failure is fast and easy.

Incremental Backup. This method creates a copy of only the changed data since the last full or incremental backup. This method reduces the size of backups and backup creation time, but restoring data may take longer and require more complex procedures.

Differential Backup. This method copies only the changed data since the last full backup. The difference from the incremental method is that the differential backup contains only changes since the last full backup, not since the last incremental backup. This method takes up more space than the incremental method, but the data recovery procedure is usually simpler and faster.

Snapshots. This method of data backup, which stores the state of the file system at a particular point in time. Snapshots do not require additional copies of data, but they do require operating system support and are stored within the file system.

Replication is a method of creating copies of data on other servers or disk arrays in order to ensure availability and reliability of information in case of failure of the primary server. Replication allows you to reduce the time it takes to recover data because if the primary server fails, you can switch to a backup that is already up and running.

Every information backup approach comes with its own strengths and weaknesses, and the selection of a method hinges on the distinct needs and constraints of the system. For instance, a complete backup might prove optimal for smaller systems handling modest data volumes, whereas in larger systems managing substantial data loads, incremental or differential backups could offer a more fitting solution.

Also an important aspect of information backup is the choice of storage location for backups, as they must be protected from potential threats such as hacking or natural disasters. In this case, it can be effective to use cloud data storage, which provides a high level of security and availability.

Diverse strategies are employed to enhance the dependability of data storage within cloud environments. The use of approaches that allow to perform error control contributes to greater utilization of cloud storage systems. To the code, which allows to control errors that occur in the process of information processing, we present the requirement of arithmetic.

By arithmeticity of the code allowing to perform error control we will understand such a property of the code when we will have "correct" information in the result when performing any actions with "correct" information, i.e. we require transparency of performing actions with information.

It is also necessary for the code to be able to fix error packets. This property is necessary to ensure system survivability.

3.1. Principles of Introducing Information Redundancy in the RNS

The residue number system is a numerical framework in which numbers are depicted as a collection of non-negative subtractions $x_1, x_2, \dots, x_n$ by mutually prime moduli $p_1, p_2, \dots, p_n$ [9, 10].

Let the number X be represented by residues $x_1, x_2, \dots, x_n$ on the bases $p_1, p_2, \dots, p_n$. Let us assume that k residues are sufficient to uniquely represent the number X, $k < n$, $p_1 < p_2 < \dots < p_n$, with working bases $p_1, p_2, \dots, p_k$.

The range of single-valued representation by selected modules is equal to the product of these modules $P_n = \prod_{i=1}^{n} p_i$.

Corrective properties of the modular code appear when redundancy is introduced. Let the number X be represented by residues $x_1, x_2, \dots, x_n$ on the bases $p_1, p_2, \dots, p_n$. The number X in general case can vary in the range of $|\cdot|_{P_n}^{+}$. Let us impose a restriction on the range of possible variation, for this purpose we will assume that k residues are sufficient for a one-to-one representation of a number X, and $k < n$. Let's assume that $p_1 < p_2 < \cdots < p_k < \cdots < p_n$, whereby the working bases will be considered as $p_1 < p_2 < \cdots < p_k$.

The range of single-digit representation of numbers on these bases is equal to $P_n = \prod_{i=1}^{k} p_i$.

Consequently, in the depiction of a number X using residues $x_1, x_2, ..., x_n$, any subset of r residues can be omitted without compromising the clarity of the representation of the number X. This attribute empowers RNS to manage errors effectively and establishes it as a non-linear code, recognized as an R-code.

Let us show by example the ability of R-code in RNS to control errors. Let some number $0 \leq x \leq 100$. For representation in residues, we choose bases $p_1 = 3$, $p_2 = 5$, $p_3 = 7$. The residues x_1, x_2, x_3 on the corresponding bases can be used to represent a number in the range from 0 to 104. Thus we have an almost lossless representation $X = (x_1, x_2, x_3)$. Let $X = 49 = (1, 4, 0)$. Let's add one more base $p_4 = 11$. to the three existing bases. Then our number $X = 49$ be represented by four residues $X = (1, 4, 0, 5)$. Suppose a fault occurred during data transmission and as a result an error occurred on the basis of $p_2 = 5$. In the received data, we will leave only three parts of the data corresponding to the grounds $p_1 = 2$, $p_2 = 7$, $p_3 = 11$, $(1, 0, 5)$. Let's restore the data from the three parts and get that they correspond to the transmitted number 49. This erase operation can be performed with any of the bases and the data will be restored correctly as a result.

A linear corrective L-code can also be constructed in RNS. To construct the L-code it is enough to remove the restrictions on pairwise mutual simplicity of bases.

Removing the conditions of pairwise mutual simplicity leads to the fact that redundancy in the representation of a number X will appear not only due to the reduction of the number of working bases, but also due to the fact that the range of representation of a number X by k residues will be less than the power of the range of number representations.

3.2. Estimation of Corrective Capacity of Codes in RNS

In order to evaluate the ability of RNS to detect, localize and correct errors we introduce the notion of weight $W_{P_n}(|X|^+_{P_n})$ of a number X.

We will define the weight $W_{P_n}(|X|^+_{P_n})$ of a number X as the count of non-zero residues. This delineation of a number's weight aligns with the description of code weight in the Hemming metric.

In $W_{P_n}(|X|^+_{P_n})$, the lower index of P_n at W indicates that the number X is depicted using a quantity of residues such that $X \in |\cdot|^+_{P_n}$.

Evidently, the count of residues through which the number X, is represented is equal to ω in this case. Writing the argument $|X|^+_{P_n}$ means that $X \in |\cdot|^+_{P_n}$.

$$W_{P_n}(|X|) = \sum_{i=1}^{n} \sigma_i(x_i), \text{ where } \sigma_i = \begin{cases} 0, & \text{if} \quad x_i = 0 \\ 1, & \text{if} \quad x_i \neq 0 \end{cases}$$

Utilizing this delineation of a number's weight, $W_{P_n}(|X|^+_{P_n})$ it is possible to compute it both directly on X, and on the set of residuals $x_1, x_2, \ldots, x_n$.

The concept of weight attributed to a number X depicted using residues can be employed to establish the concept of distance between two points in a space, where each point corresponds to the numbers X_1 and X_2 [11].

We denote the distance d_{X_1,X_2} between points in space X_1 and X_2 as the weight of the disparity between X_1 and X_2.

$$d_{X_1,X_2} = W_{P_n}(|X_1 - X_2|^+_{P_n})$$

To ascertain the error-correcting capabilities of the code, we calculate the mean weight $\overline{W}_{P_n}(|\cdot|^+_{P_n})$ of non-zero complexes. If the value of number X is modified within the range $|\cdot|^+_{P_n}$, the division period of zeros becomes p_i, and the count of zeros in base p_i will be equivalent to

$$\omega_i = \frac{P_k}{p_i} - 1, (i = \overline{1,k})$$

Subsequently, the quantity of non-zero elements within the base becomes identical to

$$\overline{\omega}_i = P_k - 1 - \omega_i = P_k - \frac{P_k}{p_i}$$

The sum of weights S of residues $x_i (i = 1, 2, .., k)$ when X changes from zero to $P_k - 1$ is determined from the expression

$$S = \sum_{i=1}^{k} \left(P_k - \frac{P_k}{p_i} \right) = P_k \left(k - \sum_{i=1}^{k} \frac{1}{p_i} \right)$$

Average weight $\overline{W}_{P_n}(|\cdot|^+_{P_n})$ can be defined as

$$\overline{W}_{P_n}(|\cdot|^+_{P_n}) = \frac{S}{P_k - 1} = \frac{P_k}{P_k - 1} \cdot \left(k - \sum_{i=1}^{k} \frac{1}{p_i} \right)$$

The more adept the code is in error correction, the greater the distinctions among numbers within the code representation, resulting in an increased code distance. Consequently, the distance will vary between distinct numbers.

When we compute the mean weight of the numbers composing the null space, the minimum code distance will always remain below the upper limit of the minimum code distance. The minimal weight of elements within the null code space is equivalent to

$$d_{\min} = \min\{W_{P_n}(|\cdot|^+_{P_n})] = \omega - k + 1 \approx r + 1$$

When $d_{\min} \approx r + 1$ we can ensure the detection of any error. Simultaneously, the utilization of RNS empowers us to identify a greater quantity of errors. As an illustration, when utilizing RNS with a single supplementary base exceeding the values of any worker, we can detect 100% of single errors and 95% of double errors.

Additionally, it's feasible to identify all errors of a multiplicity t if $t < d_{\min}$. This signifies that RNS facilitates the detection of errors with a specific multiplicity, and the ascertainable error multiplicities are determined by the minimum code distance $d_{\min}$.

3.3. Algorithms for Error Detection in RNS

Suppose a number $X = (x_1, x_2, \ldots, x_n)$ has undergone a distortion and instead of a number X we have $\tilde{X} = (\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n)$, such that $\tilde{x} \in |\cdot|^+_{P_n}$. Let us compute the difference e beetwin $\tilde{X}$ and X, which will determine the magnitude of the error

$$e = |\tilde{x} - x|^+_{P_n}$$

Since errors in distinct digits are mutually exclusive, it becomes viable to illustrate e in a manner akin to representing a number X, i.e., by residues $e = (e_1, e_2, \ldots, e_n)$ which will be defined $e_i = |\tilde{x}_i - x_i|^+_{P_n}$.

From the accepted residuals $\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n$ we calculate the number $\tilde{X}$ by solving systems of comparisons

$$\tilde{X}_i \equiv \tilde{x}_i (\mathrm{mod} p_i), (i = 1, 2, \ldots, k)$$

To address this system of comparisons, we will employ the technique of orthogonalized vectors in the following format

$$B_i = (0, 0, \ldots, 1, \ldots, 0) = \frac{m_i P_k}{p_i}$$

where $m_i \in |\cdot|^+_{p_i}$ such that $B_i \equiv 1(\mathrm{mod} p_i)$.

By utilizing orthogonalized vectors, the number X can be expressed in the configuration

$$X = \sum_{i=1}^{k} x_i B_i - r_{P_k} x(|X|^+_{P_k}) P_k$$

where $r_{P_k}(|X|^+_{P_k})$ – core function [11–13], such that if the argument $|X|^+_{P_k}$ takes such a value that X is computed using orthonormalized vectors $X \in |\cdot|^+_{P_k}$.

After computing the value of $\tilde{X}$ determine the values of the residues $\tilde{X} = (\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n)$ of this number by the $p_{k+1}, \ldots, p_n$.

Syndrome components are defined as

$$C_j = |x_{k+j} - \tilde{x}_{k+j}|^+_{p_{k+j}}, \ j = 1, 2, \ldots, n-k$$

We will show that it is possible to determine the fact of error occurrence from syndrome components. Let us represent the number X as

$$\tilde{X} = |X + e|^+_{P_k}$$

Suppose that only residuals on bases $p_1, p_2, \ldots, p_k$, which we consider informational, are affected by errors. With this in mind.

$$C_j = \left| \left| \sum_{\phi \in J_k} e_\phi B_\phi \right|^+_{p_k} \right|^+_{p_{k+j}}$$

where e_ϕ – error value in ϕ-digit, J_k – set formed from the numbers of information bases, the residuals of which are distorted.

Given that the value of e_k is non-zero and the supplementary bases adhere to the criterion of mutual coprimality, any instance of at least one syndrome component being non-zero signifies the existence of an error. Similarly, it can be demonstrated that error management is attainable when errors arise within the control bases, either individually or collectively.

3.4. Error Detection Algorithm Using Positional Features

Error control for any code is reduced to finding out whether the message belongs to the set of numbers that make up the zero space of the code. When performing an error finding operation for a number X, it is desirable that this range be close to the range P. The error detection process in RNS is reduced to comparing the number $\tilde{X}$ with the value of the range P_k [13]. The computed positional characteristic $\Pi_{P_n}(|X|^+_{P_k})$ determines the position of $\tilde{X}$ inside the range $|\cdot|^+_{P_k}$.

$$\Pi_{P_n}(|X|^+_{P_k}) = \begin{cases} 0, & \text{if} \quad X < P_k \\ 1, & \text{if} \quad X \geq P_k \end{cases}$$

By the character of change, positional characteristics are divided into linear (monotonic) and nonlinear ones. An instance of a non-linear positional attribute is represented by the number rank function $r_{P_n}(|X|^+_{P_k})$. The function $r_{P_n}(|X|^+_{P_k})$ is periodic with period equal to P_k.

$$r_{P_n}(|X|^+_{P_k}) = \left|\sum_{i=1}^{k+1} x_i\beta_i\right|^+_{p_{k+1}}, \; \beta_i = \left|\frac{B_i}{p_k}\right|^+_{p_{k+1}}$$

The function $r_{P_n}(|X|^+_{P_k})$ depends on $k+1$ variables, and the possible domain of definition of the argument X is in the range $|\cdot|^+_{P_n}$, $P_n = p_{k+1}P_k$.

4. STRUCTURAL REDUNDANCY IN CLOUD STORAGE SYSTEMS

Structural redundancy in cloud storage systems is one method of ensuring the reliability and availability of data in the cloud. Structural redundancy involves using multiple physical devices to store data and replicating data on these devices.

Cloud storage systems typically utilize distributed storage, which is a collection of physical storage devices working together. These devices can be located in different locations and have different characteristics such as capacity, speed, reliability and bandwidth [14].

To ensure the reliability and availability of data in cloud systems, various backup technologies such as replication, sharding, etc. are used. Structural redundancy involves creating multiple copies of data on different physical devices using special algorithms and protocols that ensure data consistency and integrity.

If one of the devices fails, data can be restored from backups stored on other devices. At the same time, the cloud continues to function and provide access to data without downtime or data loss.

The basic idea behind structural redundancy in cloud storage systems is to create a distributed network of servers that work together to ensure data availability and reliability. Each server in such a network can store only part of the data, which reduces the load on a single server and increases the speed of data access.

Classification of various methods of structural redundancy is performed according to the following features:

- According to the redundant element inclusion scheme;
- By homogeneity of redundancy;
- By method of reserve activation;
- On functional recovery of failed elements;
- By reserve loading.

The fundamental attribute of structural redundancy is its multiplicity, which signifies the proportion of redundant elements to the total number of elements (both redundant and primary). This proportion is expressed as a non-simplified fraction.

The redundancy multiple m is determined by the ratio:

$$m = \frac{p}{o}$$

where p – number of reserve elements; o – number of main elements.

If $m = 1$, redundancy is executed with integral multiplicity. In practical terms, this signifies the presence of a single primary element and a corresponding redundant element, resulting in a total count of two elements for the object.

When $m = 3$, redundancy is employed with integral multiplicity. This indicates the existence of one primary element along with three supplementary elements, resulting in a total of four elements.

When $m = \frac{4}{2}$, it signifies a fractional multiplicity redundancy scenario, wherein there are 4 reserve elements, 2 primary elements, and a total of 6 elements. It's not feasible to simplify the fraction, as reducing $m = \frac{4}{2}$ to $m = 2$ would lead to redundancy with integral multiplicity, entailing 2 reserve elements and a total of 3 elements.

As already mentioned, structural redundancy in cloud storage systems involves the use of distributed systems in which data is stored on multiple devices (servers) in different geographical areas. Each server has its own copy of the data, and if one of the servers fails, the data can be restored from another server.

An important aspect of structural redundancy is the organization of the network infrastructure that provides connectivity between servers and allows data to be transferred between them. Typically, high-speed links such as optical networks are used as network connections to reduce latency and increase data transfer speeds.

In addition, an important aspect of structural redundancy is the organization of data integrity control system, which allows detecting and correcting data errors. Various methods can be used for this purpose, such as checksums, Hamming codes and others.

An illustration of structural redundancy within cloud storage is the Redundant Array of Independent Disks (RAID), which amalgamates numerous physical disks into a unified logical storage unit to ensure elevated data reliability and accessibility. Various RAID levels offer distinct degrees of data safeguarding and recuperation should disk failure or other system anomalies arise.

5. RELIABILITY OF NON-RECOVERABLE REDUNDANT FACILITIES

Reliability quantification of redundant facilities is determined by the type of redundancy.

5.1. Total Hot Spare Capacity with Integer Multiplicity

In general, hot spare capacity, the main and each redundant object have the same number of elements connected in series in the reliability structural diagram. The redundant objects are included in parallel with the main object and themselves [15].

In each object with series connection of elements the probability of uptime operation is determined on the basis of expressions:

$$P_0(t) = \prod_{i=1}^{n} P_{0i}(t), P_1(t) = \prod_{i=1}^{n} P_{1i}(t), \ldots, P_m(t) = \prod_{i=1}^{n} P_{mi}(t)$$

The main and redundant facilities, consisting of elements connected in series, constitute the structural reliability scheme of the facility with parallel connection.

Then the probability of uptime operation $P_C(t)$ of the object:

$$P_C(t) = 1 - \left[1 - \prod_{i=1}^{n} P_i(t)\right]^{m+1}$$

where $P_i(t)$ – probability of uptime operation of i-th element.

If the failure rate of an element is known and it is a constant value, then:

$$P_C(t) = 1 - (1 - e^{-\lambda_0 t})^{m+1}$$

where $\lambda_0 = \sum_{i=1}^{n} \lambda_i$ – failure rate of the main facility or any of the backup facilities.

Mean time between failures (MTBF) of the redundant object:

$$T = \frac{1}{\lambda_0} \sum_{j=0}^{m} \frac{1}{j+1}$$

Failure intensity of the redundant object:

$$\lambda_C = \frac{(m+1)\lambda_0 P_0(t) Q_0^m(t)}{1 - Q_0^{m+1}(t)}$$

General hot standby for non-recoverable objects gives a good effect at small values of the product $\lambda_0 t$. This type of redundancy is most appropriate for redundancy of sufficiently reliable single-use facilities with short continuous operation time.

The redundancy multiplicity m required to achieve a given reliability of the redundant object at a certain point in time at a given reliability of the main object can be found from the expression:

$$m = \frac{\ln[1 - P_3(t)]}{\ln[1 - P_0(t)]} - 1 = \frac{\ln Q_3(t)}{\ln Q_0(t)} - 1$$

where $P_3(t)$ – specified reliability of the redundant object at a certain point in time; $P_0(t)$ – reliability of the main facility without redundancy.

5.2. Separate Hot Spare Capacity with Integer Multiplicity

In case of separate hot standby, the main and each redundant object can have the same or different number of elements. The redundant objects are switched on in parallel to themselves and in parallel to the elements of the main object.

Each i-th element of the main object is redundant by a redundant element. In case of separate redundancy, the object consists of n elements connected in series, each of which has m elements connected in parallel, in the sense of reliability.

The probability of uptime of the object $P_C(t)$ under separate hot standby is equal to:

$$P_C(t) = \prod_{i=1}^{n} \{1 - [1 - P_i(t)]^{m_i+1}\} = \prod_{i=1}^{n} [1 - Q_i^{m_i+1}(t)]$$

where m_i is the redundancy multiplicity of each i-th element of the main object.

If the elements have the same reliability and the same redundancy multiplicity, the probability of uptime operation of the object:

$$P_C(t) = \left[1 - (1 - e^{-\lambda t})^{m+1}\right]^n = [1 - Q_i^{m+1}(t)]^n$$

MTBF of the redundant object:

$$T = \frac{(n-1)!}{\lambda(m+1)} \sum_{j=0}^{m} \frac{1}{\frac{j+1}{m+1}\left(\frac{j+1}{m+1} + 1\right) \cdots \left(\frac{j+1}{m+1} + n - 1\right)}$$

Failure intensity of the redundant object

$$\lambda_C = \frac{n \cdot (m+1) \cdot \lambda \cdot e^{-\lambda t} \cdot (1 - e^{-\lambda t})^m}{1 - (1 - e^{-\lambda t})^{m+1}} = \frac{n(m+1) \cdot \lambda \cdot P(t) Q^m(t)}{1 - Q^{m+1}(t)}$$

where λ is the failure rate of one element.

Other things being equal, separate redundancy gives a significant reliability improvement compared to general redundancy. Separate redundancy is used to increase the reliability of facilities with a large number of elements and longtime of use.

5.3. Hot Spare Capacity with Fractional Multiplicity

A majoritarian object can be considered as a variant of an object with parallel connection of elements, the failure of which will occur if k elements out of n elements connected in parallel are operable.

The presented scheme is operable when any two, three, four or all five of its five elements are operable. The combinatorial method can be used to calculate the reliability of majoritarian objects. Then the probability of uptime of the object:

$$P_k = C_n^k p^k (1-p)^{n-k}$$

where $C_n^k = \frac{n!}{k!(n-k)!}$

MTBF of the redundant object with equal reliability elements is equal:

$$T_C = \frac{1}{\lambda} \sum_{j=0}^{k} \frac{1}{n+j}$$

The analysis of the last expression allows us to conclude that the average uptime of a redundant object is less than that of an unreserved object. Therefore, redundancy with fractional multiplicity is inexpedient to use for objects with long continuous operation time.

5.4. Cold Backup with Integer Multiplicity

In the case of general cold backup, the main facility operates, the backup facilities are disconnected by means of special switching devices. If the main object fails, it is disconnected and one of the redundant objects is connected instead. Thus, the redundant object will fail when the main and all reserve objects fail. It is assumed that the switching devices are absolutely reliable. Then the probability of uptime is equal to the probability of failure.

$$P_C(t) = \sum_{j=0}^{m} P(A_j) = e^{-\lambda_0 t} \sum_{j=0}^{m} \frac{(\lambda_0 t)^j}{j!}$$

where $\lambda_0 = \sum_{i=1}^{n} \lambda_i$.

Failure rate:

$$\lambda_C(t) = \frac{\lambda_0 (\lambda_0 t)^m}{m! \sum_j^m \frac{(\lambda_0 t)^j}{j!}}$$

MTBF:

$$T_C = \frac{m+1}{\lambda_0} = (m+1)T_0$$

5.5. Separate Cold Backup with Integer Multiplicity

Probability of uptime of the main object when the elements are connected in series

$$P_0(t) = \prod_{i=0}^{n} P_i(t)$$

where $P_i(t)$ – probability of uptime of the object elements redundant by the method of replacement.

The probability of uptime of an object if all elements are equally reliable:

$$P_C(t) = e^{\lambda_0 t} \left[\sum_{j=0}^{m} \frac{(\lambda t)^j}{j!} \right]^n$$

Failure rate:

$$\lambda_C(t) = \frac{\lambda_0(\lambda t)^m}{m! \sum_{j=0}^{m} \frac{(\lambda t)^{m'}}{j!}}$$

where λ – failure rate of one element, $\lambda_0 = n\lambda$.

MTBF:

$$T_C = \frac{m+1}{\lambda_0} = (m+1)T_0$$

Separate redundancy by replacement, all other things being equal, gives the greatest reliability gain compared to other types of redundancy. And this gain is greater the more elements the redundant system has.

5.6. Sliding Reservation

Sliding redundancy is used to increase the reliability of several identical (or interchangeable) elements of the facility by one or more reserve elements. In this case, the failure of the object will occur if the number of failed main elements exceeds the number of redundant elements.

This type of redundancy is used if all elements of the object perform the same functions. The main object has n elements, and m elements are in cold standby. In this case $m < n$. If any main element fails, any of the reserve elements is connected instead of it. In this case, the redundancy multiplicity is m/n.

The probability of uptime of such an object is equal to:

$$P_C(t) = \sum_{j=0}^{m} P(A_j) = e^{-\lambda_0 t} \sum_{j=0}^{m} \frac{(\lambda_0 t)^j}{j!}$$

where $\lambda_0 = \sum_{i=1}^{n} \lambda_i$.

Failure rate:

$$\lambda_C(t) = \frac{\lambda_0(\lambda_0 t)^m}{m! \sum_{j=0}^{m} \frac{(\lambda_0 t)^j}{j!}}$$

MTBF:

$$T_C = \frac{m+1}{\lambda_0} = (m+1)T_0$$

This means that the reliability of the facility is equal to that of a facility with total redundancy with replacement, but, at the same time, it has n times fewer redundant elements.

6. PRINCIPLES OF BUILDING A FAULT TOLERANT CLOUD SYSTEM

When building cloud-based high-reliability systems, there are two approaches to improve reliability: preventing failure and creating fault-tolerant and survivable systems [16].

In the first approach, one tries to eliminate all possible sources of failure by constructing a system from highly reliable elements. However, in this case, as the intensity of element failures approaches threshold values due to physical parameters or technologies, the cost of the system increases dramatically.

In the second approach, the probability of failure of cloud system components is not reduced, but the system is subjected to additional requirements: it must be error free when individual parts or storages fail. These additional requirements and hence fault tolerant operation of the system is ensured by applying different approaches:

- Distributing load and data across multiple nodes so that if one node fails, the others can continue to operate without downtime;

- Backup data and distribute it across multiple nodes so that if one node fails, the data can be quickly restored to other nodes;
- Use of geographically distributed data centers, to reduce the risks associated with natural or man-made disasters in one region;
- Use of modern virtualization and containerization technologies, which allow to increase flexibility and scalability of the system, as well as to quickly transfer work from one node to another;
- Ensuring the security of stored or processed data and the system as a whole by means of multi-level protection, encryption, authentication and user authorization;
- Continuous testing and analysis of system operation to identify bottlenecks, optimize performance and improve reliability;
- Fault tolerance systems can be categorized into four types;
- Systems with multiple redundancy, having N identical modules and determining the solution result on the basis of majorization;
- Systems with structural redundancy, in which part of the modules are used for the solution, the remaining modules are in reserve and are activated as active modules fail;
- Hybrid redundant systems consisting of multiple redundant modules for failure replacement;
- Systems with gradual degradation (survivable), in which all serviceable modules are used to perform the assigned task, i.e., all modules are active and in case of failure the system is reconfigured and the task is continued

. The performance of the first three types of systems does not change as individual modules fail. In the fourth type of system the performance decreases as degradation (failure of modules) occurs [17, 18].

The generalized algorithm of functioning of any fault-tolerant system is reduced to the following standard procedures: detecting a failure in the system, diagnosing the failed module, eliminating the influence of the failed device, reconfiguring the system.

Systems with constant performance are described by the following characteristics: time to system failure t_c, system reliability $R(t) = P_c[t_c \geq t]$ and MTBF defined as the mathematical expectation of t_c.

In degraded systems, performance degrades over time, so characteristics linking performance to reliability characteristics are introduced.

- Reliability of operation $[R^*(t, T)]$ – the probability that the system will correctly perform the assigned task in time T;
- Average number of calculations between failures N_{av};
- t_τ, i.e., the duration of the interval at which the reliability of operation for a task with operation time T reaches a given value;
- Computation availability factor $[d_c(t)]$ i.e., the average value of system performance at time t;
- Performance threshold (t_c), i.e., the time interval at which the value $d_c(t)$ reaches a given value.

7. STRUCTURE OF FAULT TOLERANT DATA STORAGE AND PROCESSING SYSTEM IN CLOUD FOG ENVIRONMENT

Typically, a fault-tolerant storage system in a cloud environment is constructed with multiple tiers, each assigned distinct functions aimed at ensuring data reliability and accessibility [18–20].

The physical infrastructure layer is the first layer in the structure of a resilient cloud storage system. It includes physical servers, storage, networking equipment, and other components that provide data storage and availability.

To provide fault tolerance at this level, various technologies such as server clustering, data replication, load balancing mechanisms, etc. are used. For example, server clustering allows you to combine several physical servers into one cluster, which increases system availability and performance, and data replication allows you to create copies of data on several servers, which allows you to ensure data availability in case of failure of one server.

At the physical infrastructure level, virtualization technologies can also be used to make more efficient use of server resources and provide greater resiliency, for example, by enabling applications and data to be quickly migrated between physical servers in the event of a hardware failure.

The virtualization layer in a cloud environment is responsible for managing and virtualizing the physical infrastructure. This layer abstracts away physical servers and data storage, allowing for more flexible resource management and efficiency.

Virtualization provides the ability to create virtual machines and containers that can run on a single physical server using the resources of that server. Virtual machines can be quickly moved between servers in the event of a hardware failure or for load balancing.

Virtualization also enables you to create snapshots of virtual machines and datastores, making it easier to back up data and restore it in the event of failures. In addition, virtualization provides the ability to create virtual networks and network devices, allowing you to manage your network infrastructure more flexibly.

Thus, the virtualization layer in a cloud environment plays an important role in providing system resiliency by enabling rapid migration of applications and data between physical servers, data backup, and resource access control.

The application layer is the layer in the cloud infrastructure that hosts the applications that utilize the data stored in the cloud. This layer includes software that is used to process, store, and transfer data, including databases, data processing applications, and web services.

At the application layer, cloud infrastructure utilizes data backup mechanisms, monitoring and management mechanisms, and automatic disaster recovery mechanisms. In addition, the application layer typically utilizes mechanisms that provide scalability to support the growing number of users and data being processed.

The data access layer is an abstraction layer that provides access to data from applications that utilize the cloud environment. This layer typically implements a mechanism to access data through APIs or protocols such as HTTP or HTTPS.

To provide security and fault tolerance at this layer, data encryption, access control, and user authentication mechanisms are used. For example, authorization mechanisms such as OAuth or OpenID Connect can be used for access control, and encryption protocols such as SSL or TLS can be used for data encryption.

In addition, at the data access layer, a data caching mechanism can be used to increase the speed of data access. Data caching allows a copy of the data to be stored in server memory or on other devices, which speeds up data access and reduces the load on the data warehouse. However, when using data caching, it is necessary to ensure that the cache is consistent with the data store to avoid possible data integrity issues.

The RNS allows to significantly improve the parameters of cloud storage compared to the storage based on traditional arithmetic, as well as to obtain new, more progressive design and structural solutions. Application of the RNS provides independent and parallel processing of each part of data, which determines the specifics of cloud storage organization. Redundant coding in RNS provides survivability of the system even in catastrophic situations, when the flow of faults and failures is very large, then the system will produce functioning.

The main feature of RNS based cloud structure is the potential ability to function in case of failures, losses, inaccessibility of data parts. This allows to preserve the operability at the expense of performance and performance degradation. The specified property provides high survivability of the system and is one of the main factors on which the development of ways to build fault-tolerant data storage systems that can degrade gradually is one of the main ways to build complexes with high viability, fault tolerance and reliability is based.

Let us consider issues related to fault tolerance of cloud storage system based on one of the most important properties of a residual class system, namely, the possibility of exchange operations between accuracy, speed and reliability.

The cloud storage structure shown in Fig. 7.1 consists of independent channels and cloud storage. In order to realize the adaptation methods, it is necessary to strive for the ultimate homogeneity and independence of all equipment.

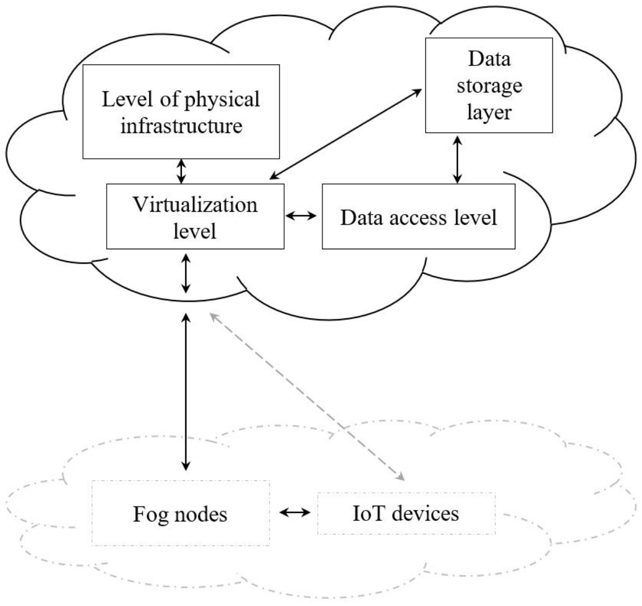

Fig. 7.1. Structure of the cloud fog system.

8. STRUCTURE OF A FAULT TOLERANT DATA PROCESSING SYSTEM IN A FOG ENVIRONMENT

Fog computing is an architectural concept that extends cloud computing capabilities by moving computing resources closer to the edge of the network, to Internet of Things (IoT) devices and other end-user devices.

The basic idea of fog computing is to process data and perform computations at a closer distance to the data source. This reduces latency, improves responsiveness, and enables more efficient utilization of network resources.

There are certain reliability issues of data processing on fog nodes. Here are some of them:

Lack of network connectivity: Fog nodes may be connected to a network with limited bandwidth or may temporarily lose connectivity with cloud servers. This may lead to problems with data transfer or synchronization with cloud services.

Vulnerability to Failure: Fog nodes may be prone to failures or malfunctions due to hardware issues, software bugs, or external influences. This may result in data loss or temporary unavailability for data processing.

Data Security: Data processing on fog nodes may pose data security risks. Such nodes may be more vulnerable to cyberattacks and may not have the same levels of protection as centralized cloud servers. Appropriate security measures are required to protect data transmitted and processed on fog nodes.

Scaling: Managing and scaling a large number of fog nodes in a distributed environment can be challenging. As a fog computing system grows and new nodes are added, efficient management mechanisms are needed to ensure reliability and consistency across the entire system.

Resource Optimization: Allocating and optimizing computational resources on fog nodes can be a complex task. Application requirements, node load, and available resources must be considered to achieve optimal reliability and performance.

Data synchronization: In a fog computing environment, there may be problems with data synchronization between different nodes. While transferring data between nodes and synchronizing computation, it is necessary to ensure data consistency and avoid possible conflicts.

Limited resources: Fog nodes may have limited computing resources, memory, power consumption and network bandwidth. This can impact system reliability and performance, especially when processing large amounts of data or performing complex computations.

Change Management: In a fog computing environment, changes to the network infrastructure, node configuration, or available resources may occur. Change and update management can be complex and require effective mechanisms to minimize potential system disruption and ensure reliability.

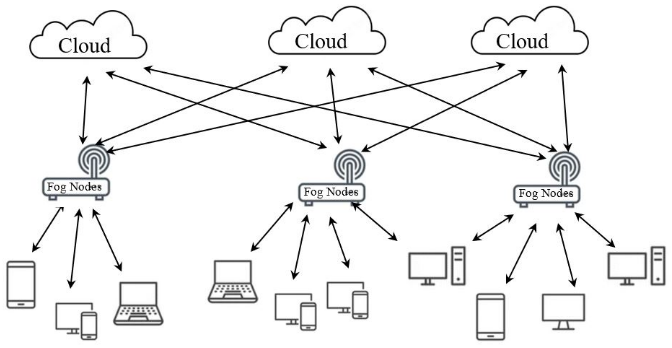

Fig. 8.2. Data storage and processing system in a fog environment.

However, modern technologies and development techniques can cope with these challenges. For example, resource reservation, disaster recovery mechanisms, data replication, and cryptographic techniques can improve the reliability of data processing on fog nodes.

9. MODELING OF THE PROPOSED SYSTEM FOR RELIABLE DATA STORAGE AND PROCESSING IN FOG ENVIRONMENT

To validate our theoretical conclusions about the reliability of the proposed system, we have performed Monte Carlo simulation of the developed system. In a foggy system, any large number of devices can be disconnected from the access point because it has physically moved and no signal reaches it. Also, it may become inaccessible due to battery drain and device disconnection. The proposed system allows to build a system with a small redundancy, designed for unforeseen failures. For this purpose, let's build a mathematical model.

The developed model of data storage and processing in a fog environment consists of k working devices and r control devices, to increase reliability with a slight increase in redundancy. In case of failure or unavailability of working devices, they can be replaced by control devices and vice versa. The requirement for the computational scheme to provide guaranteed protection against the issuance of unreliable results determines the need to maintain in an operable state $k_{\min}$ working and at least one control device. Thus, the reliable structure of the computational model will correspond to the known method of sliding redundancy with the loaded state of redundant elements.

When carrying out calculations on fog devices each of the above-mentioned possible system failures can occur both separately and jointly.

We have introduced a single-tier architecture for data processing and storage on fog devices, offering simplified frameworks for crafting data processing strategies. The assembly of data storage and processing systems can be accomplished through a range of module counts. In this system, there are k operational bases and r control bases, summing up to $n = k + r$. Establishing the data processing system is achievable through complete replication of the operational range $(k = r)$ or through partial replication $(k > r)$. We will determine the overall reliability of the information processing and storage design by employing RNS for distributed data storage and processing. Additionally, we will assess the available space across diverse system configurations.

We will calculate the reliability of the system by the following construction parameters:

1. The system parameters were randomly selected as follows: $k = 3 \dots 26$, where r varies from 1 to 19.
2. The probability of complete failure of the data processing and storage system was calculated by the formula

$$P_0 = \sum_{i=n-k+1}^{n} C_n^i P_d^i q^{n-1}$$

   where $q = 1 - P_d$, P_d – a device of size 3015 was randomly selected from the set of failures.

The Monte Carlo method was used to calculate for 10,000 repetitions for each of the random failures from the set. The simulation results are shown in Fig. 9.3, 9.4, 9.5.

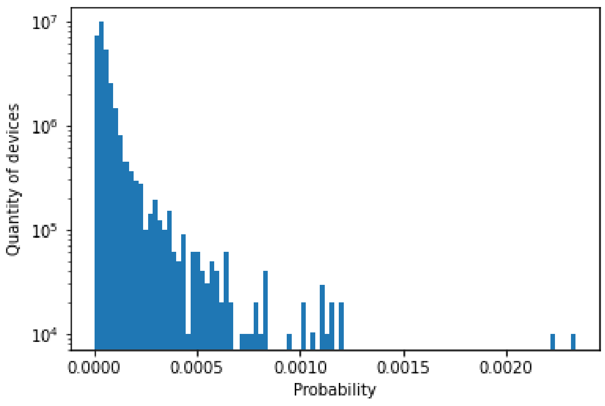

Fig. 9.3. Failure distribution of the system under parameters $r = 3$, $k = 3$

The failure probability distribution was then plotted and compared with the Gamma probability distribution, and the results are shown in figure 4.

10. CONCLUSION

The principles of building reliable data storage and processing systems in cloud-fog environment were considered in the article. The factors affecting the reliability and survivability of systems have been analyzed and the main types of failures have been

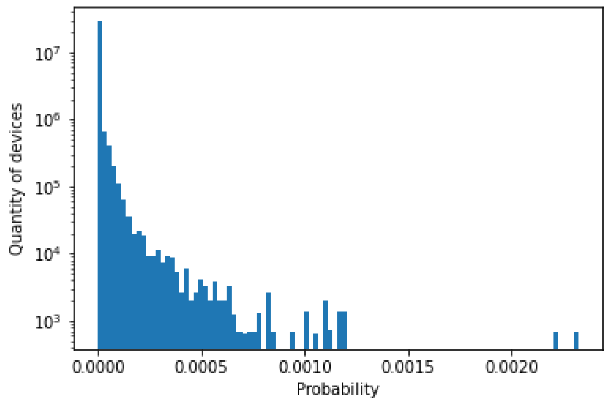

Fig. 9.4. Failure distribution of the system under parameters $r = 5 \ldots 7$, $k = 1 \ldots 5$

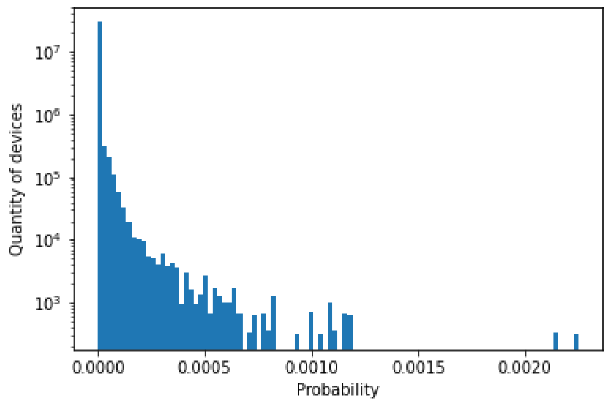

Fig. 9.5. Failure distribution of the system under parameters$r = 19 \ldots 26$, $k = 15 \ldots 19$

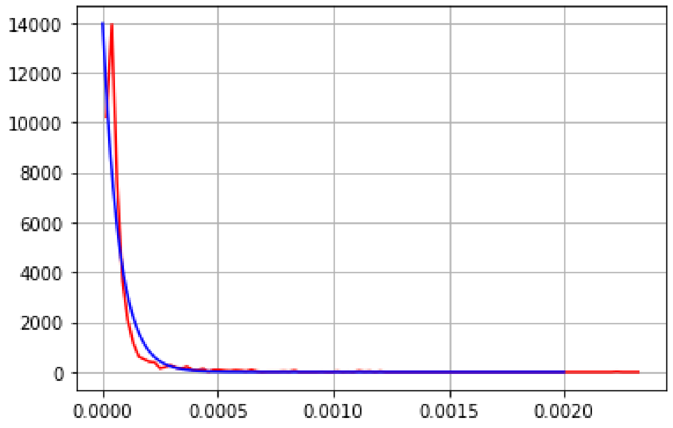

Fig. 9.6. Probability distribution, for system configurations $r = 3$, $k = 3$

given: sudden, gradual, operational. The principles of introducing information redundancy to increase reliability are considered, as well as the corrective abilities and error detection algorithms of RNS codes were considered. The scheme of data processing and storage in fog

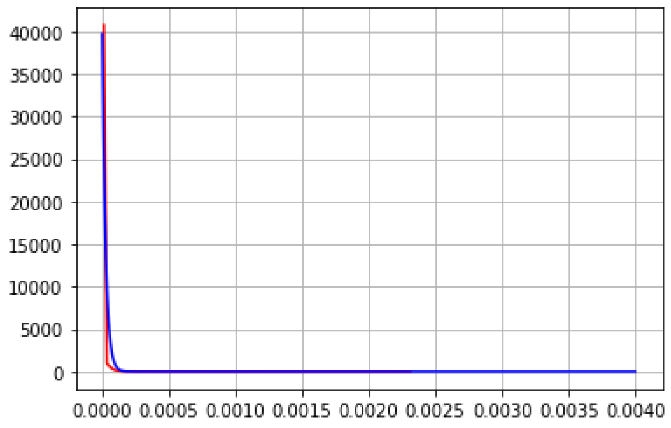

Fig. 9.7. Failure distribution of the system under parameters$r = 5 \ldots 7$, $k = 1 \ldots 5$

with increased reliability is proposed. Monte Carlo modeling was carried out, which showed that the proposed system allows you to process and store data in a foggy environment at a high level of reliability. A probability distribution of failure occurrence was constructed, which showed that failure probabilities obey the gamma probability distribution.

Funding: The research was supported by the Russian Science Foundation Grant No. 22-71-10046, https://rscf.ru/en/project/22-71-10046/.

REFERENCES

1. Al Kez, D., Foley, A. M., Laverty, D., Del Rio, D. F., & Sovacool, B. (2022) Exploring the sustainability challenges facing digitalization and internet data centers. *Journal of Cleaner Production*, **371**, 133633.
2. Yi, S., Li, C., & Li, Q. (2015) A survey of fog computing: concepts, applications and issues. In *Proceedings of the 2015 workshop on mobile big data*, 37–42.
3. Priyadarshini, R., Barik, R. K., & Dubey, H. (2018) Deepfog: fog computing-based deep neural architecture for prediction of stress types, diabetes and hypertension attacks. *Computation*, **6**, 62.
4. Babenko, M., Tchernykh, A., Pulido-Gaytan, B., Avetisyan, A., Nesmachnow, S., Wang, X., & Granelli, F. (2022) Towards the sign function best approximation for secure outsourced computations and control. *Mathematics*, **10**, 2006.
5. Apat, H. K., Nayak, R., & Sahoo, B. (2023) A comprehensive review on internet of things application placement in fog computing environment. *Internet of Things*, p. 100866.
6. Dai, Y.-S., Yang, B., Dongarra, J., & Zhang, G. (2009) Cloud service reliability: Modeling and analysis. In *15th IEEE Pacific Rim International Symposium on Dependable Computing*, 1–17, Citeseer.
7. Spillner, J., Bombach, G., Matthischke, S., Muller, J., Tzschichholz, R., & Schill, A. (2011) Information dispersion over redundant arrays of optimal cloud storage for desktop users. In *2011 Fourth IEEE International Conference on Utility and Cloud Computing*, 1–8, IEEE.
8. Ji-Yi, W., Jian-Lin, Z., Tong, W., & Qian-li, S. (2011) Study on redundant strategies in peer to peer cloud storage systems. *Applied mathematics & information sciences*, **5**, 235S–242S.
9. Garner, H. L. (1959) The residue number system. In *Papers presented at the the March 3-5, 1959, western joint computer conference*, 146–153.

10. Pei, D., Salomaa, A., & Ding, C. (1996) *Chinese remainder theorem: applications in computing, coding, cryptography*. World Scientific.
11. Krasnobayev, V., Kuznetsov, A., Popenko, V., Kononchenko, A., & Kuznetsova, T. (2020) Determination of positional characteristics of numbers in the residual class system. In *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, 151–156, IEEE.
12. Akushsky, I., Burtsev, V., & Pak, N. (1977) Calculation of the positional characteristic (core) of the non-positional code. *Theory of Coding and Optimization of Complex Systems; Nauka: Alma-Ata, Kazakhstan*, 17–25.
13. Popov, D. & Gapochkin, A. (2018) Development of algorithm for control and correction of errors of digital signals, represented in system of residual classes. In *2018 International Russian Automation Conference (RusAutoCon)*, 1–3, IEEE.
14. Cheraghlou, M. N., Khadem-Zadeh, A., & Haghparast, M. (2016) A survey of fault tolerance architecture in cloud computing. *Journal of Network and Computer Applications*, **61**, 81–92.
15. Araujo, J., Oliveira, D., Matos, R., Alves, G., & Maciel, P. (2023) Availability and reliability modeling of mobile cloud architectures. *IEEE Transactions on Industrial Informatics*.
16. Al-Dulaimy, A., Ashjaei, M., Behnam, M., Nolte, T., & Papadopoulos, A. V. (2022) Fault tolerance in cloud manufacturing: An overview. In *International Conference on Mobile Computing, Applications, and Services*, 89–101, Springer.
17. Sheeba, A. & Uma Maheswari, B. (2023) An efficient fault tolerance scheme based enhanced firefly optimization for virtual machine placement in cloud computing. *Concurrency and Computation: Practice and Experience*, **35**, e7610.
18. Mohammadi, V., Rahmani, A. M., Darwesh, A., & Sahafi, A. (2023) Fault tolerance in fog-based social internet of things. *Knowledge-Based Systems*, **265**, 110376.
19. Semmoud, A., Hakem, M., Benmammar, B., & Charr, J.-C. (2022) A new fault-tolerant algorithm based on replication and preemptive migration in cloud computing. *International Journal of Cloud Applications and Computing (IJCAC)*, **12**, 1–14.
20. Naim, M. H., Zain, J. M., & Abd Jalil, K. (2022) Fault tolerance mechanism for software application through fog computing as middleware. *International Journal of Computing and Digital Systems*, **11**, 45–54.