Journal of Edge Computing, **2023**, 2(1), pp. 37-63 https://doi.org/10.55056/jec.585

Investigating the effect of virtual machine migration accounting on reliability using a cluster model

Andrii V. Riabko[1], *Tetiana A.* Vakaliuk[2,3,4,5], *Oksana V.* Zaika[1], *Roman P.* Kukharchuk[1] and *Valerii V.* Kontsedailo[6]

[1]*Oleksandr Dovzhenko Hlukhiv National Pedagogical University, 24 Kyivska Str., Hlukhiv, 41400, Ukraine*

[2]*Zhytomyr Polytechnic State University, 103 Chudnivsyka Str., Zhytomyr, 10005, Ukraine*

[3]*Institute for Digitalisation of Education of the NAES of Ukraine, 9 M. Berlynskoho Str., Kyiv, 04060, Ukraine*

[4]*Kryvyi Rih State Pedagogical University, 54 Gagarin Ave., Kryvyi Rih, 50086, Ukraine*

[5]*Academy of Cognitive and Natural Sciences, 54 Gagarin Ave., Kryvyi Rih, 50086, Ukraine*

[6]*Inner Circle, Nieuwendijk 40, 1012 MB Amsterdam, Netherlands*

Abstract. The purpose of the article is to develop and verify with the help of mathematical modeling a software method of deploying a fault-tolerant computing cluster with a virtual machine, which consists of two physical servers (main and backup), on which a distributed data storage system with synchronous data replication from the source server to the backup server is deployed. For this purpose, the task is to conduct a computational experiment on a model of a fault-tolerant cluster, which neglects costs during recovery for the migration of virtual machines by means of the mathematical application Mathcad. Combining computing resources into clusters is a way to ensure high reliability, fault tolerance, and continuity of the computing process of computer systems. This is achieved through virtualization, which enables the movement of virtual resources, services, or applications between physical servers while maintaining the continuity of computing processes. The focus of this study is on a failover cluster, which is composed of two physical servers (primary and backup) connected through a switch, and each server has a local hard disk. A distributed storage system with synchronous data replication from the source server to the backup server is deployed on the local disks of the servers, and a virtual machine is running on the cluster. Markovian processes, flows of podias, and Kolmogorov's systems of differential equations are built into the mathematical tools of the model of a water cluster. To ensure the continuity of the computing process in case of a failure of the main server, a shadow copy of the virtual machine is launched on the backup server. The reliability of the failover cluster is measured by the coefficient of non-stationary readiness. A Markov model is proposed to assess the reliability of the failover cluster, taking into account the costs of migrating virtual machines and mechanisms that ensure the continuity of the computing process in the cluster in case of a failure of one physical server. The memory migration process maintains two copies of the virtual machine on different physical servers, enabling them to continue working on the other in the event of failure. A simplified model of the failover cluster neglects the cost of migrating virtual machines and provides an upper estimate of reliability. The study shows that the reliability of a failover cluster, as measured by the non-stationary availability factor, is significantly impacted by the virtual machine migration process. The findings of this study can be used to inform decisions about the technology chosen to ensure the failure stability and continuity of the computing process of computer systems with cluster architecture. The calculations allow us to draw a conclusion about the significant impact of virtual machine migration accounting on reliability. The calculation was performed under the following failure rates of the server, disk, and switch: $\lambda_0 = 1,115 \times 10^{-5}$ 1/h, $\lambda_1 = 3,425 \times 10^{-6}$ 1/h, $\lambda_2 = 2,3 \times 10^{-6}$ 1/h, recovery respectively: $\mu_0 = 0,33$ 1/h, $\mu_1 = 0,17$ 1/h, $\mu_2 = 0,33$ 1/h. The intensity of synchronization of the distributed storage system: $\mu_3 = 1$ 1/h, $\mu_4 = 2$ 1/h. The difference of non-stationary cluster availability coefficients is $d = K_2(t) - K_1(t) = 2.7 \times 10^{-10}$

Keywords: virtual machine, virtualization, reliability, fault tolerance, redundancy, clusters, non-stationary availability

1. Introduction

Fault tolerance is a property of a system that allows it to continue to act correctly in the event of an error or failure in some of its parts. Modern systems for processing, storing, and transmitting data for various purposes, including cyber-physical and communication systems, are subject to high requirements for reliability, security, fault tolerance, and low cost of implementation and operation [21, 23]. The requirements for computer systems largely depend on the applications they perform, their criticality to delays and continuity of service, the features of operation, and their complexity. High reliability, fault tolerance, and readiness of computer systems for critical applications are achieved by consolidating processing and storage resources based on clustering technology, dynamic distribution of requests, and virtualization. In a clustered system with virtualization, in the event of failure or disconnection of physical servers for maintenance or other work, operability is ensured by moving virtual resources, services, or applications between physical servers while maintaining the continuity of computing processes.

Modern virtualization technologies are based on the targeted migration of virtual resources between physical servers to adapt cluster systems to the accumulation of physical server failures [13]. When migrating virtual machines (VMs), a cluster can share data storage with virtual machine virtual disks, which speeds up the migration process by migrating only the main memory, virtual processor registers, and virtual device state of the virtual machines. Nevertheless, the majority of edge devices, including UAVs (Unmanned Aerial Vehicles), tablets, and cellular phones, are mobile in nature [11]. Therefore, the configuration of the cluster must be flexible enough to adapt dynamically to the evolving network topology of the edge cluster, minimizing the overall communication delay incurred by the edge devices in processing the data received from IoT devices [9, 15]. In a cluster without a shared storage implementation, the migration also moves the contents of the virtual disks of the virtual machines, which can be significant in size, which slows down the migration process. The process of moving virtual resources can be further slowed down when moving across the network. In the process of dynamic movement, it is possible to single out the stages of transferring data (registries of virtual machines, RAM, disks) to a backup server and activating the functioning of virtual machines on it.

The virtualization technology aimed at ensuring high reliability of computer systems includes High Availability Cluster and Fault Tolerance technologies, the first of which supports automatic restart of the virtual machine on healthy cluster nodes, and the second – the continuity of the computing process when it is moved to the virtual cluster servers that have retained performance. High Availability Cluster technology allows you to automatically move a virtual machine from

ryabko@meta.ua (A. V. Riabko); tetianavakaliuk@gmail.com (T. A. Vakaliuk); ksuwazaika@gmail.com (O. V. Zaika); kyxap4yk1@ukr.net (R. P. Kukharchuk); valerakontsedailo@gmail.com (V. V. Kontsedailo)
http://irbis-nbuv.gov.ua/ASUA/0051396 (A. V. Riabko); https://acnsci.org/vakaliuk/ (T. A. Vakaliuk); http://pfm.gnpu.edu.ua/index.php/struktura1/2015-04-01-14-50-26 (O. V. Zaika); http://irbis-nbuv.gov.ua/ASUA/0076404 (R. P. Kukharchuk)
0000-0001-7728-6498 (A. V. Riabko); 0000-0001-6825-4697 (T. A. Vakaliuk); 0000-0002-8479-9408 (O. V. Zaika); 0000-0002-7588-7406 (R. P. Kukharchuk); 0000-0002-6463-370X (V. V. Kontsedailo)

a failed server to a healthy server. Restoring the functionality of the virtual machine may take several minutes, depending on the configuration and loading of the physical server and the properties of user programs. With this technology, to automatically restart the virtual machine, all data must be stored on a shared data storage, which can be implemented as a device connected to all cluster nodes, or a distributed storage system. After any physical server fails, other servers can run virtual machines using virtual disks located on shared storage. In this case, the status of the virtual machine is lost, including data in RAM, registers of virtual processors, and external devices. Therefore, the system takes time to initialize the virtual machine and bring it to a pre-failure state.

For the correct operation of this virtualization mechanism, it is necessary to ensure the isolation of physical servers after a failure to exclude the simultaneous execution of the computing process by two virtual machines after a reboot to prevent data ambiguity in the shared storage.

High Availability Cluster technology assumes that after the failure of any physical server, the virtual machines running on it are automatically distributed among the surviving nodes and restarted on them. The RAM state of all virtual machines that were on the failed node is lost. Fault Tolerance technology ensures the continuity of the computing process (service) in the cluster after the failure of one physical server with the support of two copies of the virtual machine in RAM located on different physical servers so that in case of failure of one of them, continue working on the other. To organize the computing process during the operation of a virtual machine on one of the servers, the other must maintain an up-to-date copy of the RAM of the active virtual machine. In this case, virtual disk images of the virtual machine must be stored in dedicated or distributed storage with synchronous data replication. Software products that support fault tolerance technology include VMware Fault Tolerance, Kemari for Xen, and KVM.

These virtualization mechanisms affect the reliability of a cluster system, which must be taken into account when substantiating the structure of the system, and organizing computing processes and disciplines for restoring and maintaining highly reliable cluster systems. Justification of the choice of design solutions for building highly reliable cluster systems should be based on modeling in assessing the reliability, availability, fault tolerance, and performance of implementations [16, 17].

The purpose of the article is to develop and verify with the help of mathematical modeling a software method of deploying a fault-tolerant computing cluster with a virtual machine, which consists of two physical servers (main and backup), on which a distributed data storage system with synchronous data replication from the source server to the backup server is deployed.

The considered models are focused on substantiating the choice of the structure and discipline of servicing and restoring a cluster, taking into account the requirements for implemented applied tasks and virtualization mechanisms.

Developing an investigation into the effect of virtual machine migration accounting on reliability using a cluster model can offer several benefits:

1. Improved Reliability Analysis: By studying the effect of virtual machine migration accounting on reliability, you can gain insights into how different migration strategies or accounting mechanisms impact the overall reliability of the system.
2. Optimization of Migration Policies: Through the investigation, you can evaluate differ-

ent migration accounting approaches and determine which ones are more effective in maintaining system reliability.

3. Enhanced Fault-Tolerant Design: Understanding the impact of virtual machine migration accounting on reliability can inform the design of fault-tolerant systems.
4. Validation and Verification: Investigating the effect of virtual machine migration accounting on reliability provides an opportunity to validate and verify existing models or theories.
5. Decision Support: The findings from the investigation can serve as a valuable decision support tool for system administrators, architects, or operators. They can use the insights gained to make informed decisions regarding migration policies, resource allocation, system configuration, and overall management of the cluster, aiming to improve system reliability and minimize disruptions.
6. Future Research Direction: The investigation can also identify gaps or open questions in the field of virtual machine migration accounting and reliability.

Overall, the investigation provides a deeper understanding of how virtual machine migration accounting affects reliability in cluster models. This knowledge can lead to improvements in system design, resource management, and decision-making, ultimately resulting in more reliable and efficient cluster deployments.

2. Theoretical background

In the era of cloud computing [14], fault tolerance is a crucial technology that enables non-stop and long-lasting services to achieve high availability. This is typically accomplished through the use of virtualization technology [25]. Achieving high performance in cloud computing requires fault tolerance to be a critical requirement [8]. Virtualization is a widely used strategy, especially in the field of cloud computing, to enhance existing computing resources. Nevertheless, ensuring the stability and reliability of virtualization has become a significant subject [24]. According to Xu et al. [23], fault tolerance has a significant impact on the performance criteria of virtual machine scheduling. With the growing demand for cloud computing infrastructure, availability and reliability have become increasingly crucial due to their importance as major features in real-time computing systems [2].

Virtualization has become the foundation of cloud computing, enabling the deployment of virtual machines for data dissemination and administration. In modern applications, data is often stored using polyglot persistence, which combines SQL and NoSQL data stores. However, since these services are customized for specific storage requirements, it may be necessary to aggregate them from several heterogeneous clouds or migrate data from one cloud to another. Data migration can be performed offline when the database is independent of the application or, alternatively, the application must be taken offline during the migration process [7].

Cloud Computing is a groundbreaking model that provides internet-based access to physical and application resources. These resources are virtualized and offered to users as a service through virtualization software. Nevertheless, virtual machine (VM) migration using virtualization technology can adversely affect cloud performance, making it a major concern. The

uneven distribution of VMs during resource allocation and their frequent movement from one server to another can lead to increased energy consumption and network overhead [3, 6].

Cloud Computing is now extensively used for both personal and professional purposes [12]. Nevertheless, the widespread adoption and growth of cloud computing resources due to technological advancements have raised concerns about cloud service reliability and high energy consumption. In cloud computing, the primary challenges include ensuring data availability, backup replication, data efficiency, and reliability, as failures are frequently encountered during execution. Therefore, developing a fault tolerance technique is necessary to ensure reliability and availability while reducing energy consumption in the cloud. Currently, two primary fault-tolerant techniques exist – proactive and reactive fault tolerance [22]. To alleviate the resource burden on specific servers, the problem involves selecting one or more suitable virtual machines (VMs) for migration. Sivagami and Easwarakumar [20] introduce a new approach called Dynamic Fault Tolerant VM Migration that enforces reliability in cloud data center infrastructure through an advanced recovery mechanism for Virtual Network demand.

Placing virtual machines in highly reliable cloud applications is a challenging and crucial concern. To address this, the K-means clustering algorithm is utilized. Furthermore, the adaptive particle swarm optimization with the coyote optimization algorithm is employed to obtain the optimal cluster for virtual machine placement and reduce the challenge [19, 21]. Zhang, Chen and Jiang [26] establishes a model of initial placement for fault-tolerant virtual machines in star topological data centers of cloud systems, taking into account several factors such as the violation rate of service-level agreements, the remaining rate of resources, the rate of power consumption, the rate of failure, and the cost of fault tolerance. Fang, Chen and Xiong [4] developed a multi-factor real-time monitoring fault tolerance (MRMFT) model based on a GPU cluster to facilitate large-scale data processing.

Simultaneously, the continuously increasing demand for cloud resources results in service unavailability, which poses critical challenges such as cloud outages, violations of service-level agreements, and excessive power consumption [18]. Abdulhamid et al. [1] suggested a dynamic clustering league championship algorithm (DCLCA) scheduling technique that prioritizes fault tolerance awareness to tackle cloud task execution. This approach considers the currently available resources and minimizes the occurrence of untimely failure of autonomous tasks [1]. The growth of cloud usage has presented various challenges, including high energy consumption in Cloud Data Centers, security risks to Virtual Machines (VMs) due to co-residency with other risky VMs on the same Physical Machine, and Quality of Service (QoS) degradation caused by resource sharing. To address these issues, researchers have utilized Dynamic VM Consolidation to reduce energy consumption while minimizing QoS degradation. However, there are security concerns during data transmission when migrating VMs in a cloud environment. To solve this problem, Mangalagowri and Venkataraman [10] propose a Capability and Access Control (CAC) service scheme based on Software Defined Networks (SDN). In cloud data centers, virtual machine replication is useful for achieving fault tolerance, load balancing, and rapid response to user requests [5].

Summarizing the considered studies, it should be noted that the theory of reliability studies the patterns of failures of technical objects (which, in particular, include information, computer systems, and networks), methods, and models of reliability analysis and ensuring their stable operation under failure conditions. Reliability is understood as the property of an object to

maintain the ability to perform the necessary functions over time under given modes and conditions of use, maintenance, storage, and transportation. In other words, the reliability of an object is its ability to do what is needed in time.

Information, computing, and info-communication systems and networks have the following features. First, the need to take into account the impact of processing, storage, and transmission processes on the ability to perform the necessary functions. These processes create delays that can lead to the failure of functions in the required period and, as a result, failures in the implementation of the necessary functions.

Secondly, the need to take into account in computer systems the impact on the reliability of the operation of software, the failures of which have certain specifics in traditional technical systems. This specificity is due to the manifestations in the functioning of the system of errors of algorithms or programs that were not detected during testing or take into account some rare events that are potentially possible during the operation of information computer systems such as software and hardware systems.

Thirdly, a certain dependence on the reliability of the information system on ensuring its information security. Violation of information protection can manifest itself in deterioration of working conditions, increase in load, integrity violation, in particular, loss or distortion of information, which can lead to failure to perform the necessary system functions, erroneous performance, or an increase in the time of permissible delays. Functioning in the conditions of a security breach can, in particular, manifest itself in the initialization of some processes not provided for during normal operation, which, in addition to failure to perform the necessary functions and violation of the stationary of operating modes, can lead to an increase in load, overheating of processors and, ultimately, to an increase in the failure rate.

One of the main components of reliability is fault tolerance. Fault tolerance is the ability of a system to keep functioning in case of failures. The potential for maintaining the fault tolerance of the system depends on the types, number, combinations of failures, and location. Computing systems are characterized by the requirements to ensure the operability (reliability) not only of their structure as a set of hardware and software resources (including redundant ones) but also of the computing process, in particular, if it is necessary to maintain its continuity in the face of failures, failures and external destructive effects of random or malicious nature.

A feature of an information computer system is the need to consider it not only as a general technical object with requirements for structural and parametric reliability but also as an object that implements information and computing processes with the requirement of functional reliability.

The purpose of the article is to develop and verify with the help of mathematical modeling a software method of deploying a fault-tolerant computing cluster with a virtual machine, which consists of two physical servers (main and backup), on which a distributed data storage system with synchronous data replication from the source server to the backup server is deployed. For this purpose, the task is to conduct a computational experiment on a model of a fault-tolerant cluster, which neglects costs during recovery for the migration of virtual machines by means of the mathematical application Mathcad.

3. Research methods

In a redundant system, there are many able-bodied states, from which one initial state can be distinguished, characterized by the operability of all elements of the system and, accordingly, the best characteristics of the quality (efficiency) of functioning. For the accumulation of failures in fault-tolerant systems, the degradation of the efficiency and potential of the system to ensure reliability usually occurs.

The operational state, in which the current values of the parameters are at such a level that the failure of one element can lead to the failure of the system, is called the pre-failure state. In the sequence of states of a redundant system, between the initial state and the state before failure, there are usually one or more intermediate states. The number of failures of the elements that bring the system from the initial state to the pre-failure state characterizes the redundancy of the system and its resistance to failure. In the general case, systems have a complex combinatorial dependence of the number of failures sustained by the system during its degradation on the relative position of the failed elements.

Fault tolerance indicators should reflect the dynamics of maintaining efficiency in the event of one, two, or more element failures. Deterministic and probabilistic fault tolerance indicators are used. Deterministic indicators of stability failure: 1) d – the maximum number of element failures, under which the system's operability is guaranteed; 2) m – the maximum number of failures of elements, at which it is possible to maintain the system's operability. The maximum number of element failures, at which the system's operability is guaranteed d (this indicator is called d-reliability), corresponds to the minimum number of failures with the most unfortunate combination of element failures:

$$d = \min_i d_i \tag{1}$$

where d_i is the number of elements that failed during the transition from the initial (fully operational) state to the pre-failure state along the i-th path. Similarly

$$m = \max_i m_i \tag{2}$$

where m_i is the number of failures of elements during the transition to the state preceding the failure along the i-th path (each path can have several states preceding the failure).

A cluster is understood as a group of interconnected resources (servers, information storage devices, etc.), which is perceived by the user (query source) as a single resource. Clusters are created to achieve high availability, fault tolerance, and system performance based on the consolidation of resources. They can be created based on the same type or different types of resources (by parameters or functionality). In the first case, the cluster will be homogeneous, and in the second – heterogeneous. The joint work of cluster nodes is coordinated through a high-speed leased line or through a local network through which messages are exchanged. Clusters are distinguished between a server system without disk sharing and a server system with disk sharing.

An example of clusters when combining two servers is shown in figure 1.

When clustering a group of servers, there are options for organizing clusters with different redundancy. Options for combining servers and storage clusters are shown in figure 2.

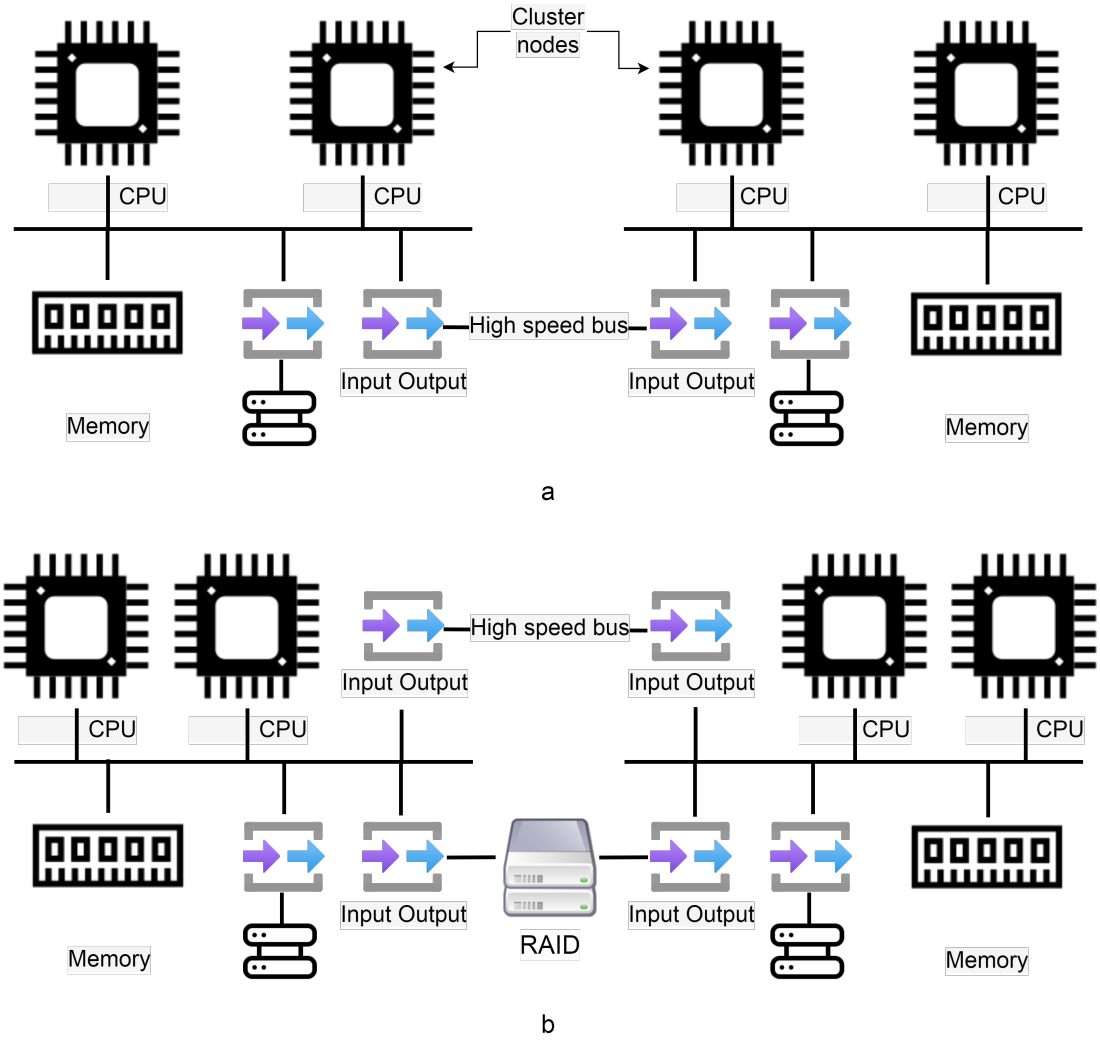


Figure 1: Clusters with merging of two servers into a system without disk sharing (a) and with disk sharing (b) [16].

In a fully permissive topology (figure 2, a), each storage device (disk array) is connected to only one cluster server. For the topology under consideration, while maintaining the fault tolerance of the configuration after node failure, failure is possible when executing functional queries due to loss of calculation results. The organization of the computational process without losing the functional requests that were executed at the time of failure is, in principle, possible for this topology, but it is associated with a significant slowdown of the computational process when organizing periodic saving of intermediate results via the local network in other nodes.

The N+1 topology (figure 2, b) means that each storage device (disk array) is connected to two cluster nodes, with one redundant server connected to all storage devices. It is used to organize high-availability clusters if one node can be allocated for redundancy. This topology reduces the load on active nodes and ensures that a load of a failed node can be restored to the standby

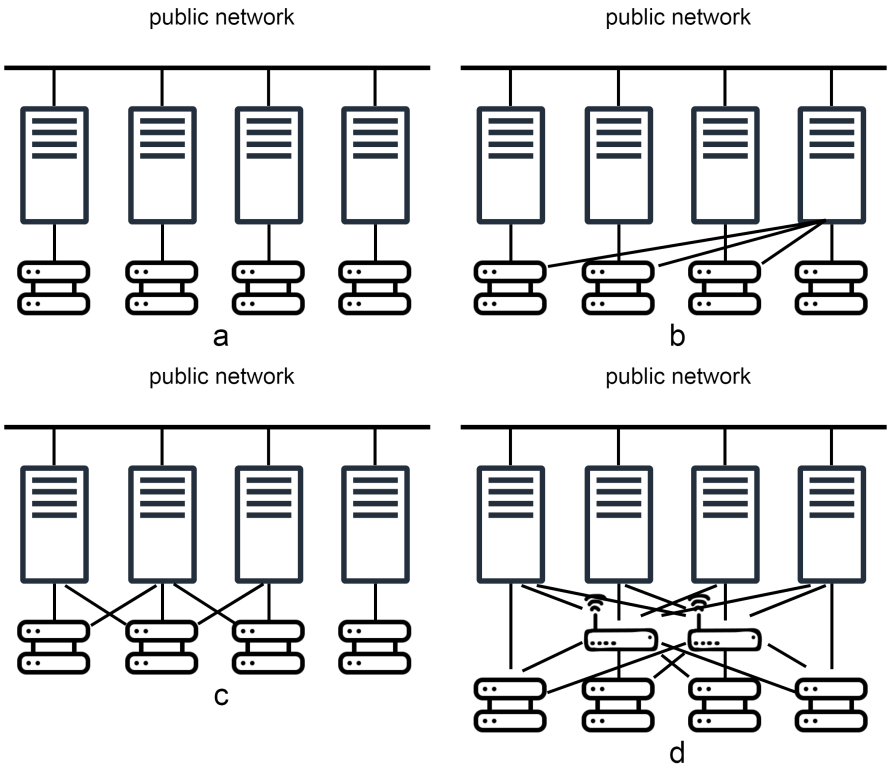


Figure 2: Cluster topologies: a – topology with completely separate access; b – N+1 topology; c – topology of cluster pairs; d – topology with the full connection of servers and storage devices through switches [16].

node without loss of quality. It maintains fault tolerance of any of the primary nodes while connecting a single redundant node. In the cluster pair topology (figure 2, c), nodes are grouped in pairs, storage devices are attached to both nodes of the pair, and each node has access to all storage devices (disk arrays) of the pair. Thus, fault tolerance is maintained within cluster pairs. In a full-access topology, servers and storage devices are connected through switches (figure 2, d), the system can be expanded by adding additional servers and storage devices to the cluster without changing existing connections. This topology provides fault tolerance for all cluster resources, which is achieved by redistributing the execution of tasks of failed nodes between healthy nodes.

The probability of failure-free operation of the structures depicted in figure 2, with the same number n of servers and storage devices, provided that at least one server and its associated storage device must work in the system, is respectively found as

$$\begin{cases} P_1(t) = 1 - (1 - p_1(t)p_2(t))^n, \\ P_2(t) = p_1(t)(1 - p_2(t))^n + (1 - p_1(t))(1 - (1 - p_1(t)p_2(t))^{n-1}), \\ P_3(t) = (1 - (1 - p_1(t))^2(1 - p_2(t))^2)^{n/2}, \\ P_4(t) = 1 - (1 - p_1(t))^n(1 - p_2(t))^n(1 - p_3(t))^g, \end{cases} \tag{3}$$

where $p_1(t)$, $p_2(t)$, and $p_3(t)$ are the probabilities of failure-free operation of servers, devices, and switches, and g is the number of switches.

Permanent operation of the infrastructure is possible only if there is an exact copy of the existing server running similar processes and services. That is, if you create a replica after a hardware failure, it will take time, which means it will lead to downtime and interruptions in the provision of services.

Fault tolerance is implemented in hardware and software. Hardware development is a "bifurcation" of the host: in other words, all the components of the system are simply duplicated, and the calculations occur at once. Synchronization is ensured by the presence of a special node. The software method is used more often but has several limitations. For example, its deployment will require the presence of a processor, communication between individual virtual machines, etc.

The programmatic way to deploy a cluster is considered in our study.

If a physical system can change from one state to another under the influence of random factors, then we can talk about a random process. The sequence of events that occur randomly is called the flow of events, and the density (intensity) of the flow is the average number of events per unit of time. Processes, where the state of the system changes at random points in time, are of great importance. Processes that are stationary, non-retarded, and ordinary are particularly important and are called the simplest or uniform Poisson processes.

Stationarity means that the probability of occurrence of a certain number of events during any group of non-intersecting time intervals depends on the length of these intervals and the number of events, but does not depend on the shift of the time intervals by the same amount. For example, the probability of occurrence of m events during the interval from t to $t + \Delta t$ does not depend on t and is a function only of the arguments m and t.

The absence of an aftereffect means that the probability of occurrence of m events during the time interval $(t, t + \Delta t)$ does not depend on how many times the events have already occurred. This means that the conditional probability of the occurrence of m events in the interval $(t, t + \Delta t)$ with any assumption about the occurrence of events up to time t coincides with the unconditional probability.

The absence of consequences means that the occurrence of any number of events at different points in time that do not overlap is independent. The ordinariness condition requires that it is practically impossible to have two or more events in a very short time interval Δt. It can be defined as the probability $P_{>1}(\Delta t)$ that more than one event will occur in this short period. Then the condition of ordinariness is as follows: $P_{>1}(\Delta t) = o(\Delta t)$.

If the probability of occurrence $P_k(t)$ is equal to k events in time t:

$$P_k(t) = \frac{(\lambda \cdot t)^k}{k!} \cdot exp(-\lambda \cdot t), k = 0, 1, 2... \tag{4}$$

$P_0(t)$ can be interpreted as the probability that the time between two consecutive events will be greater than t.

If the events form a Poisson flow, then the number m of events that fall into any time interval $(t_0, t_0 + \tau)$, is distributed according to the Poisson law. If the events form a Poisson flow, then the number m of events that fall into any time interval is distributed according to the Poisson law:

$$P_m = \frac{a^k}{m!} \cdot exp(-a) \tag{5}$$

where falling into this interval:

$$a = \int_{t_0}^{t_0+\tau} \lambda(t)dt \tag{6}$$

$\lambda(t)$ – density (intensity) of the flow.

If we call a Poisson to flow a stationary Poisson or the simplest flow, then the time interval (distance) T between two adjacent events in the simplest flow is a continuous quantity distributed according to the exponential law with density

$$\begin{cases} f(t) = 0, if t < 0 \\ f(t) = \lambda \cdot exp(-\lambda \cdot t), if t \geq 0 \end{cases} \tag{7}$$

The characteristics of the random variable T, which has an exponential distribution, include:

$$M[T] = \frac{1}{\lambda}; D[t] = \frac{1}{\lambda^2} \tag{8}$$

A random process occurs in the physical system $\sum$ when the system can change its state over time under the influence of random factors. If transitions between system states are possible only at certain moments of time $t_1, t_2, ...t_n, ...$, then this random process is called a discrete-time process. If transitions can occur at any moment in time, then the process is called a continuous-time process.

If a random process with a discrete state is Markov, then all probabilistic characteristics in the future depend only on the current state of the process and do not depend on how this process proceeded in the past. The future depends on the past only because of the current state of the process. If the process is Markov, then all the flows of events that transfer the system from state to state are Poisson.

A state graph is a convenient tool for analyzing random processes with a discrete state. This geometric diagram represents possible states of the system and possible transitions between states. Each state of the system is indicated by a square or a circle, and possible transitions are indicated by arrows connecting the squares or circles. It should be noted that the arrows show only direct transitions between the states of the system (figure 3). For example, if the system can go from state S_0 to state S_3 only through states S_1 or S_2, then the arrows show only the transitions from state S_0 to state S_1 and from state S_0 to state S_2, not the transition from state S_0 to state S_3.

Reliability theory often studies situations when system transitions from one state to another occur randomly and cannot be predicted in advance. In such cases, a discrete-state, continuous-time Markov process diagram can be used to describe the process. A system is called a system with a discrete state if it has a finite set of states $S_1, S_2, ..., S_n, ...$, and the transition from one state to another is carried out by a jump. In the following, we will consider only systems with a discrete state.

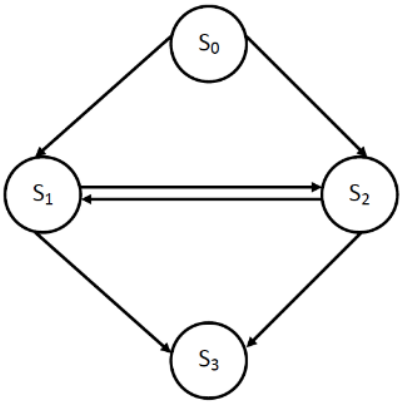


Figure 3: Graph of the state of the system [16].

If it is impossible to transition from a certain state to any other, then this state is called a "no-exit state". To describe a random process in the system, the following state probabilities are most often used: $p_1(t), p_2(t), ..., p_n(t)$, where $p_k(t)$ is the probability that the system is in the S_k state at time t.

The probabilities $p_k(t)$ must satisfy the condition:

$$\sum_{k=1}^{n} P_k(t) = 1 \tag{9}$$

Consider the probability density of the transition of the system from state S_i to state S_j.

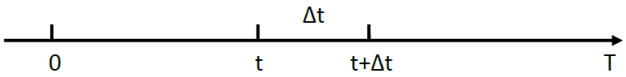


Figure 4: Representation of the system operation mode over time [16].

Let the system (figure 4) at the moment of time t be in the state S_i. Consider the elementary section Δt, which is adjacent to the moment of time t. We call the probability density (or intensity) of the transition from state S_i to state S_j the value as the limit of the ratio of the probability of transition from state S_i to state S_j during time Δt to the duration of this time interval Δt:

$$\lambda_{ij} = \lim_{\Delta t \to 0} \frac{P_{ij}(\Delta t)}{\Delta t} \tag{10}$$

where $P_{ij}(\Delta t)$ the probability that the system that was in state S_i at time t will change to state S_j during time Δt (valid only for $i \neq j$).

With a small value of the time interval Δt, the probability $P_{ij}(\Delta t)$ with an accuracy of infinitesimals of a higher order of smallness is equal to:

$$P_{ij}(\Delta t) \approx \lambda_{ij} \cdot \Delta t \tag{11}$$

If all transition intensities do not depend on time, the Markov process is called homogeneous, otherwise, the process is called heterogeneous. Let us know all λ_{ij} for all pairs (S_i, S_j). Let's build a graph of system states and put the corresponding transition probability density against each arrow (figure 5). Such a graph is called a labeled state graph.

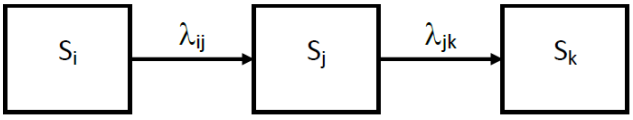


Figure 5: Labeled state graph [16].

In the presence of a marked state graph of the system, it is possible to determine the probability of the states $P_0(t), P_1(t), P_2(t)...$ as a function of time, namely, these probabilities satisfy the Kolmogorov differential equation.

As an example (figure 6), we will consider the method of obtaining a system of Kolmogorov's differential equations. Suppose that the system has five states: S_0, S_1, S_2, S_3, S_4. We need to find the probability of the system being in the state S_0 at the time t, and denote it by $P_0(t)$. Let's find the probability of the system being in the state S_0 at the time $t + \Delta t$, where Δt is a very small time interval. This is possible in two ways: a) the system will remain in its current state S_0 during the time Δt; b) the system, being in state S_3 at time t, will go to state S_0 during time Δt.

In case of a), so that the system does not change its state S_0, the probability of such an event is equal to the probability $P_0(t)$, which at the time t was in the state S_0 and did not change to the state S_1. The transition probability can be calculated (for small values of t) by the formula:

$$P_0(t) \cdot (1 - \lambda_{01} \cdot \Delta t) \tag{12}$$

where $P_0(t)$ is the probability of the system being in the S_0 state at the moment of time t, the probability of the system transitioning from the S_0 state to the S_1 state during the time interval Δt, the probability of the system not transitioning from the S_0 state to the S_1 state during the time interval Δt.

Option b) is realized if the system at time t was in state S_3 with probability $P_3(t)$ and in the time interval Δt passed into state S_0:

$$P_3(t) \cdot \lambda_{30} \cdot \Delta t \tag{13}$$

where $\lambda_{30} \cdot \Delta t$ is the possibility of transition from state S_3 to state S_0 in a small time interval Δt. Since the system at the moment $t + \Delta t$ could be in the P_0 state only in the first or second way, we get:

$$P_0(t + \Delta t) = P_0(t) \cdot (1 - \lambda_{01} \cdot \Delta t) + P_3(t) \cdot \lambda_{30} \Delta t \tag{14}$$

from here

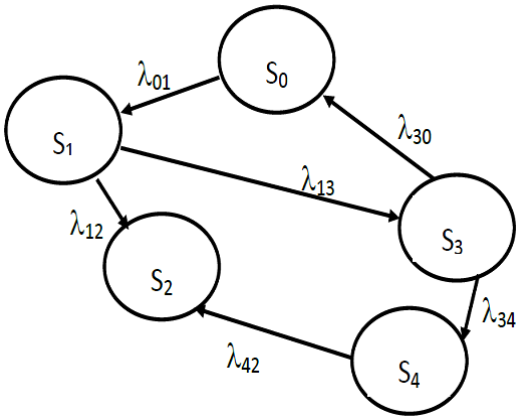


Figure 6: A fragment of the labeled graph of the system [16].

$$\frac{P_0(t+\Delta t)-P_0(t)}{\Delta t} = -P_0(t)\lambda_{01} + P_3(t)\cdot\lambda_{30} \tag{15}$$

or

$$\frac{dP_0(t)}{dt} = -P_0(t)\lambda_{01} + P_3(t)\cdot\lambda_{30} \tag{16}$$

Consider the state S_1 and derive the equation for determining the probability $P_1(t)$ that at the moment $t+\Delta t$ the system will be in the state S_1. Realization of such a state is possible if: 1) the system was at the moment t in the state S_0 and during the time Δt passed into the state S_1. The probability of such a transition is determined by the product of the corresponding probabilities:

$$P_0(t)\cdot\lambda_{01}\cdot\Delta t \tag{17}$$

2) the system at time t was in state S_1 and did not change its state during the interval, that is, it did not go either to state S_2 or to state S_3. Let's estimate the probability of implementing this option. The probability that the system, being in state S_1, will pass during time Δt to state S_2 or S_3:

$$P_1(t)\cdot(\lambda_{12}\cdot\Delta t + \lambda_{13}\cdot\Delta t) \tag{18}$$

The probability of the system not transitioning from state S_1 to any of the states:

$$P_1(t)\cdot[1-(\lambda_{12}\cdot\Delta t + \lambda_{13}\cdot\Delta t)] \tag{19}$$

We will finally get it

$$P_1(t+\Delta t) = P_1(t)\cdot[1-(\lambda_{12}\cdot\Delta t + \lambda_{13}\cdot\Delta t)] + P_0(t)\cdot\lambda_{01}\cdot\Delta t \tag{20}$$

$$\frac{P_1(t+\Delta t)-P_1(t)}{\Delta t} = -P_1(t)\cdot\lambda_{12} - P_1(t)\cdot\lambda_{13} + P_0\cdot\lambda_{01} \tag{21}$$

Or, provided that $\Delta t \to 0$

$$\frac{dP_1(t)}{dt} = -P_1(t) \cdot \lambda_{12} - P_1(t) \cdot \lambda_{13} + P_0(t) \cdot \lambda_{01} \tag{22}$$

Similarly, it is possible to obtain the dependences of the system of Kolmogorov differential equations on all other states of the analyzed system. As a result, we get a system of differential equations:

$$\begin{cases} \frac{dP_0(t)}{dt} = -P_0(t)\lambda_{01} + P_3(t) \cdot \lambda_{30} \\ \frac{dP_1(t)}{dt} = -P_1(t) \cdot \lambda_{12} - P_1(t) \cdot \lambda_{13} + P_0(t) \cdot \lambda_{01} \\ \frac{dP_2(t)}{dt} = P_4(t)\lambda_{42} + P_1(t) \cdot \lambda_{12} \\ \frac{dP_3(t)}{dt} = -P_3(t) \cdot \lambda_{30} + P_1(t) \cdot \lambda_{13} - P_3(t) \cdot \lambda_{34} \\ \frac{dP_4(t)}{dt} = P_3(t)\lambda_{34} - P_4(t) \cdot \lambda_{42} \\ P_0 + P_1 + P_2 + P_3 + P_4 = 1 \end{cases} \tag{23}$$

Integrating this system of differential equations under initial conditions, for example, $P_0(0) = 1, P_1(0) = P_2(0) = P_3(0) = P_4(0) = 0$ gives the required functions of state probabilities: $P_0(t), P_1(t), P_2(t), P_3(t), P_4(t)$.

All equations are written according to a particular rule, based on which you can write a labeled graph almost automatically: 1) in the left part of each equation there is a derivative $\frac{dp_k(t)}{dt}$; 2) in the right part there are as many members as there are arrows directly related to the given k-th state; 3) the term on the right side of the equation has a plus sign if the arrow enters the state, and a minus sign if the arrow leaves the state; 4) each member of the right-hand side of the equation is equal to the product of the density of the flow of events, which translates the system along a given arrow, by the probability of the state from which the arrow emerges.

These rules for constructing a system of Kolmogorov differential equations are valid for any continuous Markov chain. Consider the system (Figure 7).

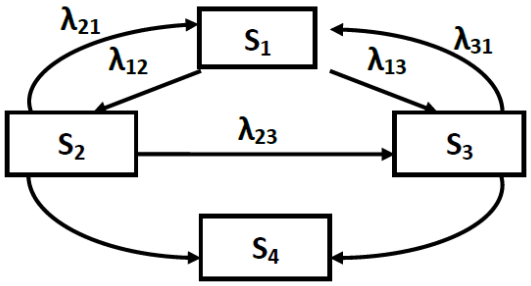


Figure 7: A labeled graph of a system with a discrete state and continuous time [16].

The system of differential equations of such a system has the form:

$$\begin{cases} \frac{dp_1(t)}{dt} = p_2(t) \cdot \lambda_{21} + p_3(t) \cdot \lambda_{31} - p_1(t) \cdot (\lambda_{12} + \lambda_{13} \\ \frac{dp_2(t)}{dt} = p_1(t)\lambda_{12} - p_2(t) \cdot \lambda_{21} - p_2(t) \cdot \lambda_{23} - p_2(t) \cdot \lambda_{24} \\ \frac{dp_3(t)}{dt} = p_1(t) \cdot \lambda_{13} + p_2(t) \cdot \lambda_{23} - p_3(t) \cdot \lambda_{31} - p_3(t) \cdot \lambda_{34} \\ \frac{dp_4(t)}{dt} = p_2(t) \cdot \lambda_{24} + p_3(t) \cdot \lambda_{34} \end{cases} \quad (24)$$

The initial conditions of integration of such a system reflect the state of the system at the initial time. So, if at the moment $t = 0$ the system was in the state S_k, then it is considered that: $p_k(0) = 1, p_i(0) = 0$ if $i \neq k$. The number of equations in the system can be reduced by one if we take into account the condition that for any t (for this system): $p_1(t) + p_2(t) + p_3(t) + p_4(t) = 1$.

Consider a technical system with discrete states in which random Markov processes with continuous time flow. Let's assume that these are the intensities of event flows that transfer the system from one state to another state, that is, all event flows are the simplest (stationary Poisson). Let's formulate the following task: what will happen to the system at $t \to \infty$? If the functions $P_i(t)$ tend to any limits, then we will call them the limiting probabilities of the states. The following general proposition can be proved. If the number of states of the system is finite and for a finite number of steps it is possible to go to any other state (closed system, figure 8, a), then the marginal probabilities of the states exist and they do not depend on time or on the initial state of the system. At the same time, the condition remains:

$$\sum_i p_i = 1 \quad (25)$$

For open systems (figure 8, b), there are no boundary states.

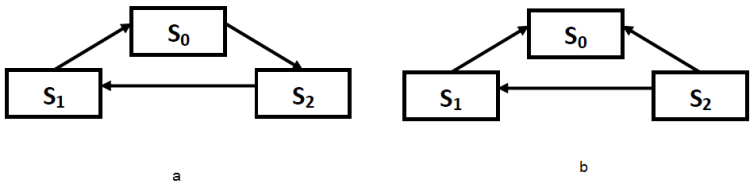


Figure 8: a – closed system graph; b - is the graph of the open system [16].

Thus, at $t \to \infty$, the system establishes some limit stationary mode, which consists in the fact that the system randomly changes its states, but the probability of each of them does not depend on time: each of the states is realized with some constant probability P_i.

At the same time, the marginal probability P_i is the average relative time of the system being in the given i-th state, i.e. after the system transitions to the stationary mode of operation, it will be in the state S_i for a time that is proportional to P_i.

For example, if the system has states S_0, S_1, S_2, and the marginal probabilities are equal to 0.4; 0.1; 0.5, then after switching to the stationary mode, 40% of the time the system will be in the S_0 state, 10% - in the S_1 state, and 50% - in the S_2 state.

4. Results and discussions

To calculate the marginal probabilities in the system of Kolmogorov's differential equations, it is necessary to set the left parts of the equations equal to zero (as derivatives of constants, since now the probability of states does not depend on time). Then the original system of differential equations is transformed into a system of linear algebraic equations, the solutions of which together with (12) make it possible to determine the marginal probabilities of P_i. The labeled graph of the closed system is shown in figure 9.

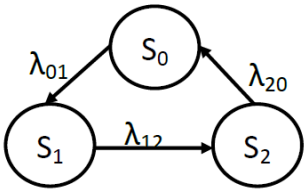


Figure 9: Labeled graph of a closed system [16].

Kolmogorov's system of differential equations:

$$\begin{cases} \frac{dP_0(t)}{dt} = -P_0(t)\lambda_{01} + P_2(t) \cdot \lambda_{20} \\ \frac{dP_1(t)}{dt} = -P_1(t) \cdot \lambda_{12} + P_0(t) \cdot \lambda_{01} \\ \frac{dP_2(t)}{dt} = -P_2(t)\lambda_{20} + P_1(t) \cdot \lambda_{12} \\ P_0 + P_1 + P_2 = 1 \end{cases} \tag{26}$$

Matlab can be used to solve this system. To exclude lower characters, we introduce the notation:

$$a = \lambda_{01}; b = \lambda_{20}; c = \lambda_{12}. \tag{27}$$

The code will look like this:

```
>> syms p0 p1 p2
>> [p0 p1 p2]=solve('a*p0-b*p2=0','c*p1-a*p0=0', 'b*p2-c*p1=0',
'p0+p1+p2=1',p0,p1,p2)
 p0 = c*b/(c*b+a*b+a*c)
 p1 = a*b/(c*b+a*b+a*c)
 p2 = c*a/(c*b+a*b+a*c)
```

Thus, the calculated dependences of the marginal probabilities of the analyzed system take the form:

$$\begin{aligned} p_0 &= \frac{\lambda_{20} \cdot \lambda_{12}}{\lambda_{01} \cdot \lambda_{20} + \lambda_{01} \cdot \lambda_{12} + \lambda_{20} \cdot \lambda_{12}} \\ p_1 &= \frac{\lambda_{01} \cdot \lambda_{20}}{\lambda_{01} \cdot \lambda_{20} + \lambda_{01} \cdot \lambda_{12} + \lambda_{20} \cdot \lambda_{12}} \\ p_2 &= \frac{\lambda_{01} \cdot \lambda_{12}}{\lambda_{01} \cdot \lambda_{20} + \lambda_{01} \cdot \lambda_{12} + \lambda_{20} \cdot \lambda_{12}} \end{aligned} \tag{28}$$

The numerical value of the marginal probability corresponds to the relative time of the system in this state. For example, if the values of marginal probabilities are obtained for the analyzed system: $P_0 = 0.4; P_1 = 0.2; P_2 = 0.5$, then this means that the system will be in state S_0 40% of the time, state S_1 20% of the time, and state S_2 50% of the time.

Consider a highly reliable cluster implemented with virtualization technology focused on maintaining the continuity of the service (computing process). A failover cluster in the simplest case consists of two physical servers (primary and backup) with high-speed network interfaces (figure 10). Each server has one local hard disk drive (HDD) connected via SATA or SAS interface. Both servers have a hypervisor, clustering software, and virtualization management installed on the HDD. A distributed storage system with synchronous data replication from the source server to the backup server is deployed on the local disks of the servers. The cluster is running a virtual machine in failover mode.

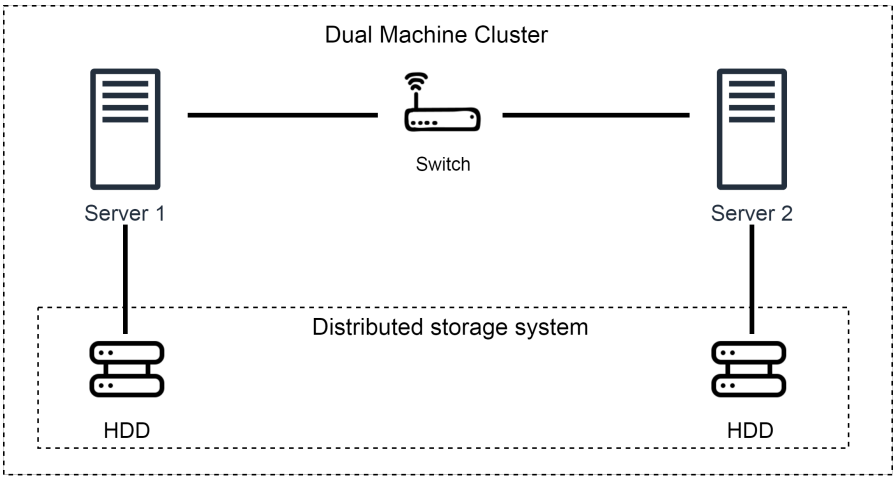


Figure 10: Fault-tolerant cluster structure [16].

The system assumes the launch of a shadow copy of the virtual machine on the backup server, which allows, after the failure of the main server, to continue the computing process on the virtual machine of the backup server without interruption. Support for the continuity of the computing process during automatic recovery after a failure (reconfiguration) requires constant synchronization of RAM and disk data, for which it is possible to use high-speed network adapters and second-level switches, for example, 10G Ethernet or InfiniBand; organizations on servers of a distributed storage system that supports synchronous replication of disk data from the primary to a backup server or a separate server for organizing an external storage system.

Let us consider the restoration of system resources that are lost as a result of failures, which is carried out immediately after a failure (provides for instantaneous detection of the occurrence of a failure using control, devices, and personnel ready to carry out repair work).

For fault-tolerant cluster systems, we take the non-stationary availability factor K and the non-stationary availability function $K(t)$ as the reliability indicator. Non-stationary availability factor – the possibility that the system at a certain point in time is ready to perform the necessary functions (is working). It characterizes the readiness of the object to perform the necessary

function at an arbitrary time t, which is close enough to the moment of a fixed change in the state of the system (before the operation, after prevention, testing, reconfiguration, or recovery). A non-stationary availability factor is applied when the stationary mode, in which the probability of states depends on time, has not yet been established. In general, these indicators depend on the failure and recovery rates of the system elements, the time of its continuous operation, and the type and frequency of redundancy.

$$K(t) \cong \frac{\mu}{\lambda+\mu} + \frac{\lambda}{\lambda+\mu} \cdot \exp[-t \cdot (\lambda+\mu)] \tag{29}$$

where $K(t) = P_0(t)$.

$$K = \lim_{t\to\infty} K(t) = \frac{\mu}{\lambda+\mu} = \frac{1}{1+\sum\limits_{i=1}^{n} \frac{\lambda_i}{\mu_i}} \tag{30}$$

where n is the number of elements of the non-redundant system, λ_i, μ_i are the corresponding failure and restoration rates of the element of the i-th type and $i = 1, 2, ..., n$; $\lambda = \sum\limits_{i=1}^{n} \lambda_i$ – system failure rate. System update rate is

$$\mu = \frac{\lambda}{\sum\limits_{i=1}^{n} \frac{\lambda_i}{\mu_i}} = \frac{\sum\limits_{i=1}^{n} \lambda_i}{\sum\limits_{i=1}^{n} \frac{\lambda_i}{\mu_i}} \tag{31}$$

The above dependencies indicate that the higher the coefficient and the readiness function, the lower the ratio $\frac{\lambda_i}{\mu_i}$.

Dynamic models are used to calculate the fault tolerance characteristics of complex systems. If the behavior of the system can be described by a Markov action, the mathematical model of the reliability of such a system is a system of differential equations. When studying the functioning of recoverable systems under the Poisson law of distribution of failure and restoration flows (the intensity of the failure flow $\lambda(t)$ and the restoration intensity $\mu(t)$ are constants), the mathematical model of such a system is a system of ordinary differential equations. The system of ordinary differential equations can be solved analytically or numerically.

Consider the case when the failure rate (t) is a function of time. Figure 11 shows the Markov graph of the restored element, the mathematical model of which is a system of nonlinear differential equations.

If you build Markov models of system fault tolerance, consisting of several renewable elements, then the state space of the model will increase. The system of differential equations with respect to $P_i(t)$ $(i = 1, 2, ..., n)$ will have the general form

$$\begin{cases} P_1'(t) = -P_1(t) \sum \lambda_{1i}(t) + \sum P_i(t)\lambda_{i1}(t), \\ \ldots \\ P_k'(t) = -P_k(t) \sum \lambda_{ki}(t) + \sum P_i(t)\lambda_{ik}(t), \\ \ldots \\ P_n'(t) = -P_n(t) \sum \lambda_{ni}(t) + \sum P_i(t)\lambda_{in}(t), \end{cases} \tag{32}$$

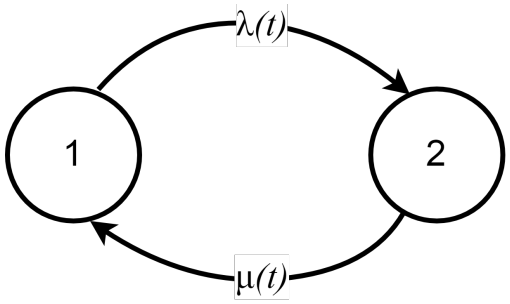


Figure 11: Markov graph of a fault-tolerant element [16].

where the first sum on the right side of the equation contains the intensity of transitions from the current state k, and the second sum is the intensity of transitions to state k; transitions corresponding to failures have time-dependent coefficients; transitions corresponding to the restoration of working capacity are constants.

In the general case, it is difficult to obtain an analytical solution to a system of nonlinear differential equations; therefore, it is advisable to use numerous methods for solving. For example, Mathcad has a built-in function $rkfixed$, which is considered basic and implements the fourth-order Runge-Kutta method with a fixed step. This function is designed to solve systems of first-order differential equations

$$\begin{aligned} y_1' &= f_1(x, y_1, y_2, ..., y_n), \\ y_2' &= f_2(x, y_1, y_2, ..., y_n), \\ &... \\ y_n' &= f_n(x, y_1, y_2, ..., y_n), \end{aligned} \tag{33}$$

The function $rkfixed(y, x_1, x_2, npoints, D)$ returns a matrix of $1 + npoints$ rows, in which the first column contains the solution, and the other columns contain the solution and its first $n - 1$ derivatives.

Function arguments are:

y is the vector of initial values (n elements);

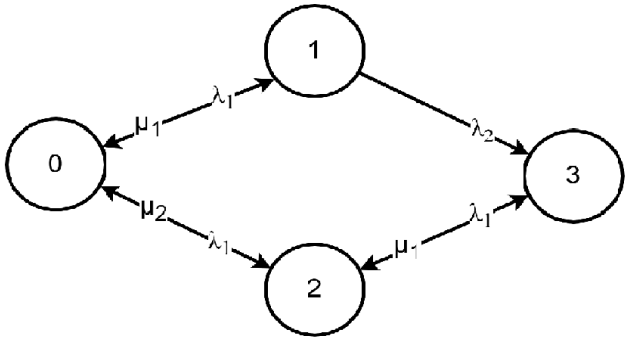
x_1 and x_2 are the limits of the interval on which we are looking for a solution;

$npoints$ – the number of points inside the interval (x_1, x_2) in which we are looking for a solution. They are chosen from the condition of obtaining the desired accuracy of numerical integration;

D is a vector of n elements – the first derivatives of the desired function.

As an example, consider the solution of a system of differential equations for finding the non-stationary availability factor of a duplicated system. On it, column 0 corresponds to time $(t = Z_n, 0)$, and the subsequent columns are the probabilities of states depending on time (figure 12). Also shown is a plot of the non-stationary availability factor (availability function) versus time.

Let us build a Markov model of the reliability of a failover cluster with online recovery, taking into account the implementation of mechanisms for moving a virtual machine. The state



$\lambda 1 := 10^{-3}$ $\mu 1 := 1$

$\lambda 2 := 0.5 \times 10^{-3}$ $\mu 2 := 1.5$

$P := \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$ $D(t, P) := \begin{pmatrix} -\lambda 1 \cdot P_0 - \lambda 2 \cdot P_0 + \mu 1 \cdot P_1 + \mu 2 \cdot P_2 \\ -\mu 1 \cdot P_1 + \lambda 1 \cdot P_0 - \lambda 2 \cdot P_1 \\ \lambda 2 \cdot P_0 + \mu 1 \cdot P_3 - \mu 2 \cdot P_2 - \lambda 1 \cdot P_2 \\ \lambda 2 \cdot P_1 + \lambda 1 \cdot P_2 - \mu 1 \cdot P_3 \end{pmatrix}$

$Z := rkfixed(P, 0, 100, 1000, D)$ $n := 0 .. 1000$

$Z =$

	0	1	2	3	4
0	0	1	0	0	0
1	0.1	0.99986	$9.51532 \cdot 10^{-5}$	$4.64249 \cdot 10^{-5}$	$4.64081 \cdot 10^{-9}$
2	0.2	0.99973	$1.81234 \cdot 10^{-4}$	$8.63739 \cdot 10^{-5}$	$1.72413 \cdot 10^{-8}$
3	0.3	0.99962	$2.59109 \cdot 10^{-4}$	$1.20751 \cdot 10^{-4}$	$3.60622 \cdot 10^{-8}$
4	0.4	0.99952	$3.29559 \cdot 10^{-4}$	$1.50333 \cdot 10^{-4}$	$5.96487 \cdot 10^{-8}$
5	0.5	0.99943	$3.93292 \cdot 10^{-4}$	$1.75791 \cdot 10^{-4}$	$8.67882 \cdot 10^{-8}$
6	0.6	0.99935	$4.5095 \cdot 10^{-4}$	$1.97699 \cdot 10^{-4}$	$1.16475 \cdot 10^{-7}$
7	0.7	0.99928	$5.03111 \cdot 10^{-4}$	$2.16553 \cdot 10^{-4}$	...

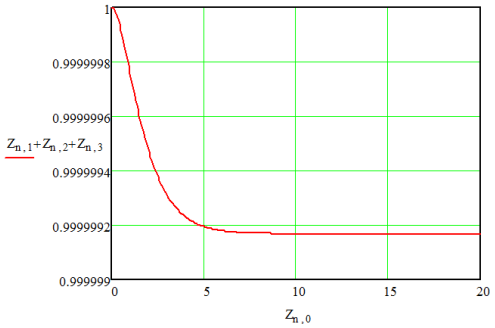


Figure 12: Mathcad worksheet with the results of solving a system of differential equations, which is a mathematical model of the reliability of a duplicated system consisting of elements 0, 1, 2, and 3 [16].

and transition diagram of a failover cluster with online recovery when implementing virtual machine movement is shown in figure 13. In the figure, the healthy states of the cluster (healthy states without failed nodes) are indicated by vertices circled with a solid line; repairman – thick solid line. The "VM" mark at the top of the graphs indicates the server on which the virtual machine with the virtual service is currently running. The top crossed out with two lines means the failure of the node, with one line – the state of the node in which it is currently not functioning and, accordingly, does not fail.

The diagram shows the failure rates $(\lambda_0, \lambda_1, \lambda_2)$ and updates (μ_0, μ_1, μ_2) of the server, disk, and switch, respectively. The intensity of updating (synchronization of the distributed storage system), including the introduction of an up-to-date replica of data on the restored disk – μ_3. The intensity of restoring a virtual machine after an automatic restart, which includes starting a virtual machine on a standby server and loading a user program on it – μ_4.

To find the state probabilities from the given state and transition diagrams, systems of algebraic equations are compiled when estimating the stationary availability factor or differential equations when estimating the non-stationary availability factor. We write the system of differential equations according to the state and transition diagram (figure 13) as follows:

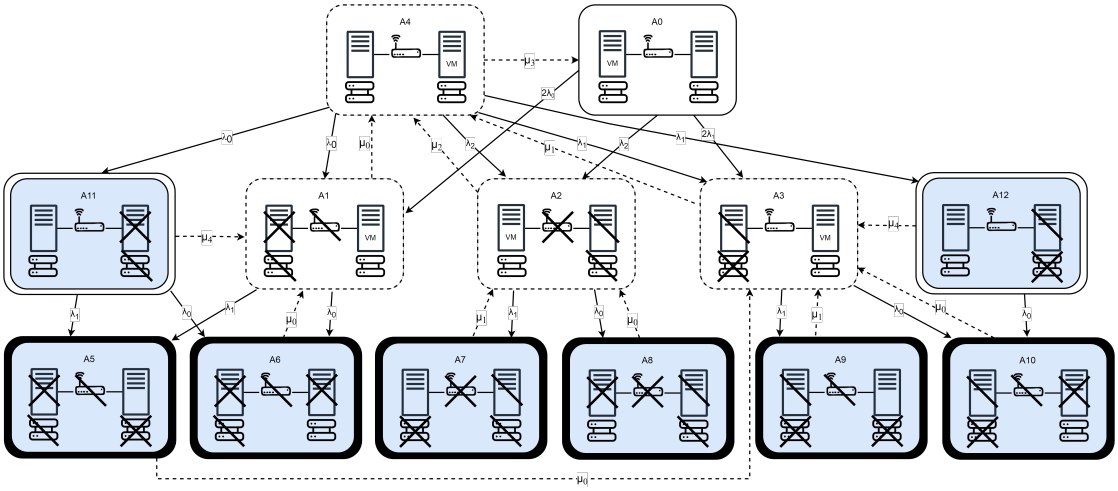


Figure 13: A fault-tolerant cluster state and transition diagram with online recovery that reflects the mechanisms of virtual machine virtualization and migration [16].

$$
\begin{cases}
P_0'(t) = -(2\lambda_0 + \lambda_2 + 2\lambda_1)P_0(t) + \mu_3 P_4(t), \\
P_1'(t) = -(\lambda_1 + \lambda_0 + \mu_0)P_1(t) + \lambda_0 P_4(t) + 2\lambda_0 P_0(t) + \mu_4 P_{11}(t) + \mu_0 P_6(t), \\
P_2'(t) = -(\lambda_1 + \lambda_0 + \mu_2)P_2(t) + \mu_1 P_7(t) + \mu_0 P_8(t) + \lambda_2 P_4(t) + \lambda_2 P_0(t), \\
P_3'(t) = -(\lambda_1 + \lambda_0 + \mu_1)P_3(t) + \mu_1 P_9(t) + \mu_0 P_{10}(t) + \mu_4 P_{12}(t) + \mu_0 P_5(t) + \\
+\lambda_1 P_4(t) + 2\lambda_1 P_0(t), \\
P_4'(t) = -(\lambda_1 + \lambda_0 + \lambda_2 + \mu_3 + \lambda_1 + \lambda_0)P_4(t) + \mu_0 P_1(t) + \mu_2 P_2(t) + \mu_1 P_3(t), \\
P_5'(t) = -\mu_0 P_5(t) + \lambda_1 P_1(t) + \lambda_1 P_{11}(t), \\
P_6'(t) = -\mu_0 P_6(t) + \lambda_0 P_1(t) + \lambda_1 P_0(t), \\
P_7'(t) = -\mu_1 P_7(t) + \lambda_1 P_2(t), \\
P_8'(t) = -\mu_0 P_8(t) + \lambda_0 P_2(t), \\
P_9'(t) = -\mu_0 P_9(t) + \lambda_1 P_3(t) + \lambda_1 P_{12}(t), \\
P_{10}'(t) = -\mu_0 P_{10}(t) + \lambda_0 P_3(t) + \lambda_0 P_{12}(t), \\
P_{11}'(t) = -\mu_4 P_1 1(t) + \lambda_0 P_4(t), \\
P_{12}'(t) = -\mu_4 P_1 2(t) + \lambda_1 P_4(t),
\end{cases}
\tag{34}
$$

As a result, the simplified Markov model of cluster reliability, without taking into account the impact of reducing the availability of the cluster, and the cost of migrating virtual machines, respectively, leads to an upper estimate of the system reliability, presented in figure 14.

The system of differential equations corresponding to the state and transition diagram, shown

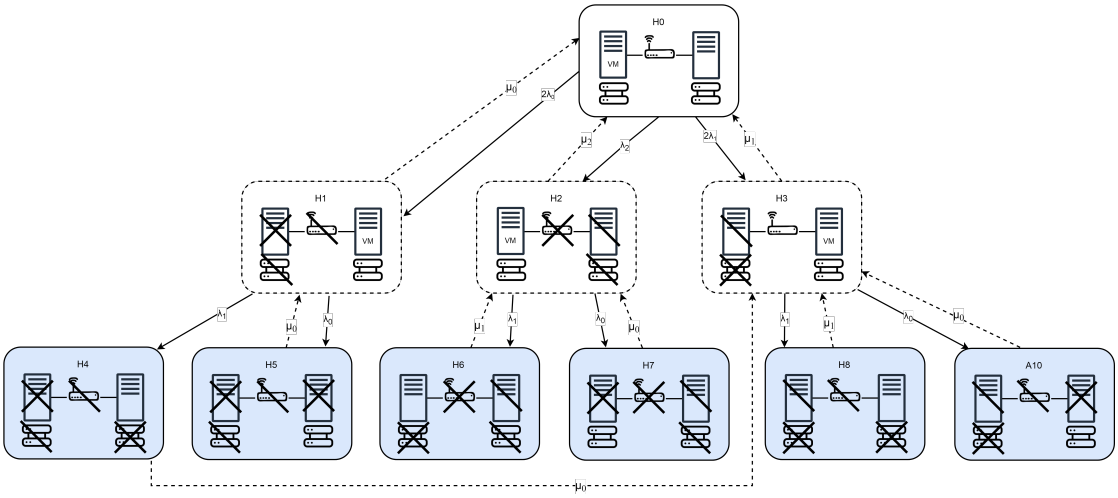


Figure 14: State and transition graph of the operational recovery cluster without taking into account the costs of virtual machine migration [16].

in figure 14, has the form:

$$\begin{cases} P_0'(t) = -(2\lambda_0 + \lambda_2 + 2\lambda_1)P_0(t) + \mu_0 P_1(t) + \mu_2 P_2(t) + \mu_1 P_3(t), \\ P_1'(t) = -(\lambda_1 + \lambda_0 + \mu_0)P_1(t) + 2\lambda_0 P_0(t) + \mu_0 P_5(t), \\ P_2'(t) = -(\lambda_1 + \lambda_0 + \mu_2)P_2(t) + \lambda_2 P_0(t) + \mu_1 P_6(t) + \mu_0 P_7(t), \\ P_3'(t) = -(\lambda_1 + \lambda_0 + \mu_1)P_3(t) + \mu_0 P_4(t) + \mu_1 P_8(t) + \mu_0 P_9(t) + 2\lambda_1 P_0(t), \\ P_4'(t) = -\mu_0 P_4(t) + \lambda_1 P_1(t), \\ P_5'(t) = -\mu_0 P_5(t) + \lambda_0 P_1(t), \\ P_6'(t) = -\mu_0 P_6(t) + \lambda_1 P_2(t), \\ P_7'(t) = -\mu_0 P_7(t) + \lambda_0 P_2(t), \\ P_8'(t) = -\mu_1 P_8(t) + \lambda_1 P_3(t), \\ P_9'(t) = -\mu_0 P_9(t) + \lambda_0 P_3(t), \end{cases} \tag{35}$$

The results of calculating the coefficients of non-stationary availability of the cluster for the models corresponding to the diagrams in figures 2 and 10 are shown in figure 15.

In figure 15, curves 1 and 2 correspond to the evaluation of the function of non-stationary availability factors $K_1(t)$ and $K_2(t)$ based on the diagrams in figure 13 and figure 14. Curve 3 in figure 11 corresponds to the difference $d = K_2(t) \dot{-} K_1(t)$ (the d value axis is on the right). The calculation was performed under the following failure rates of the server, disk, and switch: $\lambda_0 = 1.115 \times 10^{-5}$ 1/h, $\lambda_1 = 3.425 \times 10^{-6}$ 1/h, $\lambda_2 = 2.3 \times 10^{-6}$ 1/h recovery respectively: $\mu_0 = 0.33$ 1/h, $\mu_1 = 0.171$/h, $\mu_2 = 0.33$ 1/h. The difference of non-stationary cluster availability coefficients is $d = K_2(t) \dot{-} K_1(t) = 2.7 \times 10^{-10}$.

There are several limitations associated with the mathematical modeling of the method of deployment for a fault-tolerant computing cluster with the given configuration. Here are a few key limitations: 1) Complexity of Real-world Factors; 2) Incomplete Representation of

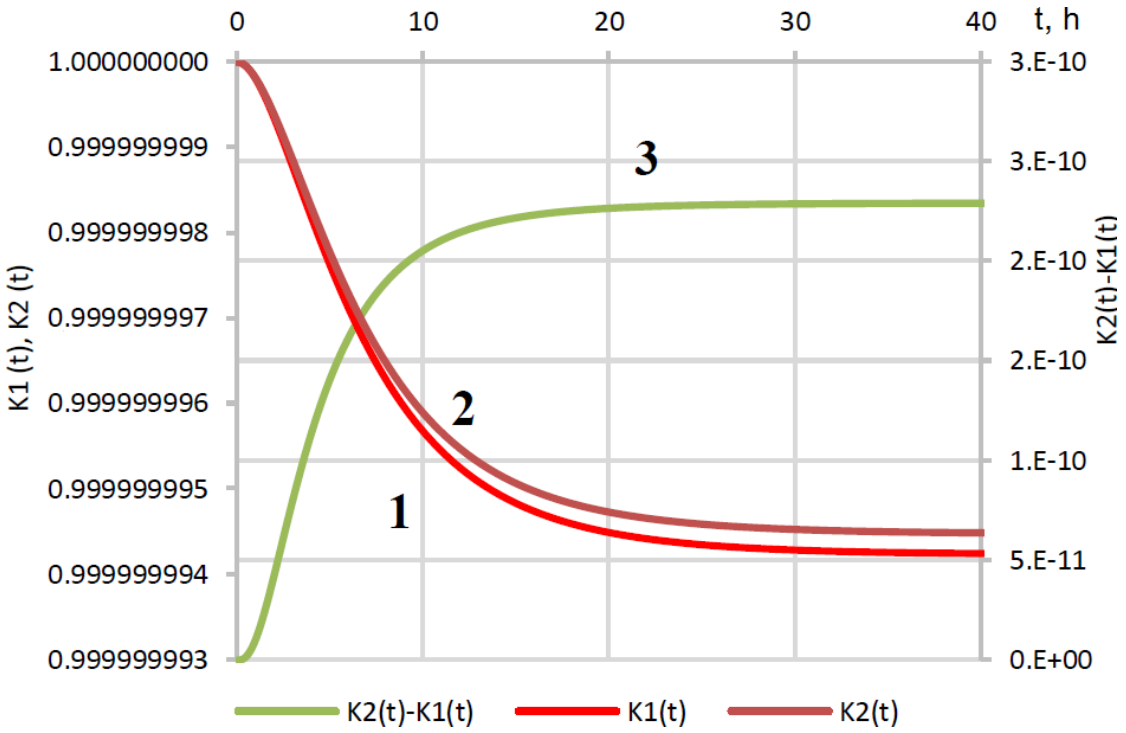


Figure 15: Non-stationary availability factors of a fault-tolerant cluster with and without taking into account virtual machine migration costs [16].

System Dynamics; 3) Uncertainty and Variability; 4) Lack of Incorporation of Human Factors; 5) Validation and Verification Challenges; 6) Lack of Adaptability to Changing Environments. Given these limitations, it is crucial to complement mathematical modeling with real-world testing, simulations, and practical experience to ensure the effectiveness and reliability of the fault-tolerant computing cluster deployment method.

5. Conclusions

The article describes a software-based method of deploying a fault-tolerant computing cluster consisting of two physical servers (primary and backup) with a local solid-state disk. The servers are connected via a switch. A distributed data storage system with synchronous data replication from the primary to the backup server is deployed on the server disks, and a virtual machine is running on the cluster. A model of a fault-tolerant cluster that neglects the cost of virtual machine migration during recovery is built. The calculations were performed in the computer mathematics system Mathcad. The calculations allow us to conclude that the accounting of virtual machine migration has a significant impact on reliability.

A Markov model of fault-tolerant cluster reliability is proposed, which takes into account the costs of virtual machine migration. A simplified model of a fault-tolerant cluster that neglects the cost of virtual machine migration during recovery is constructed. A significant impact on the

reliability of a fault-tolerant cluster (estimated by the non-stationary availability factor) is shown by taking into account virtualization mechanisms, in particular, virtual machine migration.

Synchronization intensity of the distributed storage system: $\mu_3 = 1$ 1/h, $\mu_4 = 2$ 1/h. The calculations were performed in the Mathcad computer mathematics system. The difference of non-stationary cluster availability coefficients is $d = K_2(t) - K_1(t) = 2.7 \times 10^{-10}$. The results allow us to conclude that accounting for virtual machine migration has a significant impact on reliability.

Given limitations, it is crucial to complement mathematical modeling with real-world testing, simulations, and practical experience to ensure the effectiveness and reliability of the fault-tolerant computing cluster deployment method.

Overall, the investigation provides a deeper understanding of how virtual machine migration accounting affects reliability in cluster models. This knowledge can lead to improvements in system design, resource management, and decision-making, ultimately resulting in more reliable and efficient cluster deployments.

References

[1] Abdulhamid, S.M., Latiff, M.S.A., Madni, S.H.H. and Abdullahi, M., 2016. Fault tolerance aware scheduling technique for cloud computing environment using dynamic clustering algorithm. *Neural Computing and Applications*, 29, pp.279–293. Available from: https://doi.org/10.1007/s00521-016-2448-8.

[2] Attallah, S.M., Fayek, M.B., Nassar, S.M. and Hemayed, E.E., 2021. Proactive load balancing fault tolerance algorithm in cloud computing. *Concurrency and Computation: Practice and Experience*, 33(10), p.e6172. Available from: https://doi.org/10.1002/cpe.6172.

[3] Belgacem, A., Saïd, M. and Ferrag, M.A., 2023. A machine learning model for improving virtual machine migration in cloud computing. *The Journal of Supercomputing*, pp.1–23. Available from: https://doi.org/10.1007/s11227-022-05031-z.

[4] Fang, Y., Chen, Q. and Xiong, N., 2019. A multi-factor monitoring fault tolerance model based on a GPU cluster for big data processing. *Information Sciences*, 496, pp.300–316. Available from: https://doi.org/10.1016/j.ins.2018.04.053.

[5] Gonzalez, C. and Tang, B., 2020. FT-VMP: Fault-Tolerant Virtual Machine Placement in Cloud Data Centers. *2020 29th International Conference on Computer Communications and Networks (ICCCN)*. pp.1–9. Available from: https://doi.org/10.1109/ICCCN49398.2020.9209676.

[6] Jin, H., Deng, L., Wu, S., Shi, X., Chen, H. and Pan, X., 2014. MECOM: Live migration of virtual machines by adaptively compressing memory pages. *Future Generation Computer Systems*, 38, pp.23–35. Available from: https://doi.org/10.1016/j.future.2013.09.031.

[7] Kaur, K., Bharany, S., Badotra, S., Aggarwal, K., Nayyar, A. and Sharma, S., 2022. Energy-efficient polyglot persistence database live migration among heterogeneous clouds. *The Journal of Supercomputing*, 79, pp.1–30. Available from: https://doi.org/10.1007/s11227-022-04662-6.

[8] Kumari, P. and Kaur, P., 2021. A survey of fault tolerance in cloud computing. *Journal of*

King Saud University - Computer and Information Sciences, 33(10), pp.1159–1176. Available from: https://doi.org/10.1016/j.jksuci.2018.09.021.

[9] Lobanchykova, N.M., Pilkevych, I.A. and Korchenko, O., 2022. Analysis and protection of IoT systems: Edge computing and decentralized decision-making. *Journal of Edge Computing*, 1(1), p.55–67. Available from: https://doi.org/10.55056/jec.573.

[10] Mangalagowri, R. and Venkataraman, R., 2023. Ensure secured data transmission during virtual machine migration over cloud computing environment. *International Journal of System Assurance Engineering and Management*. Available from: https://doi.org/10.1007/s13198-022-01834-8.

[11] Modlo, Y.O., Semerikov, S.O., Bondarevskyi, S.L., Tolmachev, S.T., Markova, O.M. and Nechypurenko, P.P., 2019. Methods of using mobile Internet devices in the formation of the general scientific component of bachelor in electromechanics competency in modeling of technical objects. In: A.E. Kiv and M.P. Shyshkina, eds. *Proceedings of the 2nd International Workshop on Augmented Reality in Education, Kryvyi Rih, Ukraine, March 22, 2019*. CEUR-WS.org, *CEUR Workshop Proceedings*, vol. 2547, pp.217–240. Available from: https://ceur-ws.org/Vol-2547/paper16.pdf.

[12] Nechypurenko, P., Selivanova, T. and Chernova, M., 2019. Using the Cloud-Oriented Virtual Chemical Laboratory VLab in Teaching the Solution of Experimental Problems in Chemistry of 9th Grade Students. In: V. Ermolayev, F. Mallet, V. Yakovyna, V.S. Kharchenko, V. Kobets, A. Kornilowicz, H. Kravtsov, M.S. Nikitchenko, S. Semerikov and A. Spivakovsky, eds. *Proceedings of the 15th International Conference on ICT in Education, Research and Industrial Applications. Integration, Harmonization and Knowledge Transfer. Volume II: Workshops, Kherson, Ukraine, June 12-15, 2019*. CEUR-WS.org, *CEUR Workshop Proceedings*, vol. 2393, pp.968–983. Available from: https://ceur-ws.org/Vol-2393/paper_329.pdf.

[13] Oleksiuk, V. and Oleksiuk, O., 2021. The practice of developing the academic cloud using the Proxmox VE platform. *Educational Technology Quarterly*, 2021(4), p.605–616. Available from: https://doi.org/10.55056/etq.36.

[14] Popel, M., Shokalyuk, S.V. and Shyshkina, M., 2017. The Learning Technique of the SageMathCloud Use for Students Collaboration Support. In: V. Ermolayev, N. Bassiliades, H. Fill, V. Yakovyna, H.C. Mayr, V.S. Kharchenko, V.S. Peschanenko, M. Shyshkina, M.S. Nikitchenko and A. Spivakovsky, eds. *Proceedings of the 13th International Conference on ICT in Education, Research and Industrial Applications. Integration, Harmonization and Knowledge Transfer, ICTERI 2017, Kyiv, Ukraine, May 15-18, 2017*. CEUR-WS.org, *CEUR Workshop Proceedings*, vol. 1844, pp.327–339. Available from: https://ceur-ws.org/Vol-1844/10000327.pdf.

[15] Rajashekar, K., Karmakar, S., Paul, S. and Sidhanta, S., 2023. Topology-Aware Cluster Configuration for Real-Time Multi-Access Edge Computing. *Proceedings of the 24th International Conference on Distributed Computing and Networking*. New York, NY, USA: Association for Computing Machinery, ICDCN '23, p.286–287. Available from: https://doi.org/10.1145/3571306.3571417.

[16] Riabko, A.V., Vakaliuk, T.A., Zaika, O.V., Kukharchuk, R.P. and Kontsedailo, V.V., 2023. Cluster fault tolerance model with migration of virtual machines. *Proceedings of the 3rd Edge Computing Workshop, doors 2023, Zhytomyr, Ukraine, April 7, 2023*. CEUR-WS.org, *CEUR Workshop Proceedings*, vol. 3374, pp.23–40. Available from: https://ceur-ws.org/

Vol-3374/paper02.pdf.

[17] Ryabko, A.V., Zaika, O.V., Kukharchuk, R.P. and Vakaliuk, T.A., 2022. Graph theory methods for fog computing: A pseudo-random task graph model for evaluating mobile cloud, fog and edge computing systems. *Journal of Edge Computing*, 1(1), p.1–16. Available from: https://doi.org/10.55056/jec.569.

[18] Saxena, D. and Singh, A.K., 2022. OFP-TM: An Online VM Failure Prediction and Tolerance Model towards High Availability of Cloud Computing Environments. *The Journal of Supercomputing*, 78(6), p.8003–8024. Available from: https://doi.org/10.1007/s11227-021-04235-z.

[19] Sheeba, A. and 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(7), p.e7610. Available from: https://doi.org/10.1002/cpe.7610.

[20] Sivagami, V.M. and Easwarakumar, K.S., 2019. An Improved Dynamic Fault Tolerant Management Algorithm during VM migration in Cloud Data Center. *Future Generation Computer Systems*, 98, pp.35–43. Available from: https://doi.org/10.1016/j.future.2018.11.002.

[21] Souza, A., Vittorio Papadopoulos, A., Tomas, L., Gilbert, D. and Tordsson, J., 2018. Hybrid Adaptive Checkpointing for Virtual Machine Fault Tolerance. *2018 IEEE International Conference on Cloud Engineering (IC2E)*. pp.12–22. Available from: https://doi.org/10.1109/IC2E.2018.00023.

[22] Talwar, B., Arora, A. and Bharany, S., 2021. An Energy Efficient Agent Aware Proactive Fault Tolerance for Preventing Deterioration of Virtual Machines Within Cloud Environment. *2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*. pp.1–7. Available from: https://doi.org/10.1109/ICRITO51393.2021.9596453.

[23] Xu, H., Xu, S., Wei, W. and Guo, N., 2022. Fault tolerance and quality of service aware virtual machine scheduling algorithm in cloud data centers. *The Journal of Supercomputing*. Available from: https://doi.org/10.1007/s11227-022-04760-5.

[24] Yang, C.T., Chou, W.L., Hsu, C.H. and Cuzzocrea, A., 2014. On improvement of cloud virtual machine availability with virtualization fault tolerance mechanism. *The Journal of Supercomputing*, 69(3), pp.1103–1122. Available from: https://doi.org/10.1007/s11227-013-1045-1.

[25] Yu, C.Y., Lee, C.R., Tsao, P.J., Lin, Y.S. and Chiueh, T.C., 2020. Efficient Group Fault Tolerance for Multi-tier Services in Cloud Environments. *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*. pp.1–7. Available from: https://doi.org/10.1109/ICC40277.2020.9149253.

[26] Zhang, W., Chen, X. and Jiang, J., 2021. A multi-objective optimization method of initial virtual machine fault-tolerant placement for star topological data centers of cloud systems. *Tsinghua Science and Technology*, 26(1), pp.95–111. Available from: https://doi.org/10.26599/TST.2019.9010044.