Solving the Problem of Finding Bottlenecks in Distributed Data Storage Systems using Markov Chain Theory

Ivan Mikheev[1][0000-0002-7612-1103], Olena Shapovalova[2][0000-0003-4566-6634], Rustam Burmensky[3][0000-0001-6035-7616]

[1] Kharkiv National University of Civil Engineering and Architecture, 40 Sumska str., Kharkiv, Ukraine
i.a.mikheev@gmail.com, shap_el@ukr.net, rustamburmenskii@gmail.com

Abstract. The paper is devoted to identifying bottlenecks in information and communication systems. The problems of analyzing the reliability and fault tolerance of distributed data storage system elements are considered. A system of 10 nodes is considered as an example. Input data is based on monitoring network traffic between nodes for one month. The application of the theory of Markov processes and the estimation of the probabilistic characteristics of system functioning in the stationary mode allowed to determine the most loaded nodes of the system and the direction of its modernization. The algorithm for identifying a bottleneck in a distributed data storage system was carried out on the basis of the developed data transformation mechanism and was used as the basis for creating a specialized application software in C# programming language, which allows solving a whole class of problems in various aspects of information and infocommunication technologies. The algorithm and software are ready to solve tasks of much greater dimensionality.

Keywords: Distributed System, Analysis, Bottleneck, Theory of Stationary Processes, Markov chain.

1 Introduction

Modern information technology (IT) infrastructure is a complex system that consists of a number of interrelated components, such as servers, clients, networks, etc. These components are subject to two trends: an increase in the number of elements and the complication of the interrelations between them. System's functionality determines the complexity of solutions and components. The higher the possibilities, the more complex the system. For example, the IT infrastructure of large corporate-level enterprises is, as a rule, distributed systems with complex interrelations between their elements. The large number and high complexity of the components give rise to problems of monitoring and forecasting the consequences of contingencies. These facts justify the need to work on the system in the direction of increasing its reliability.

The reliability of the IT solution is determined by its fault tolerance. Failure is understood as any violation in the system's operability, which leads to downtime in services' operations. Therefore, fault tolerance is the ability of the system to continue its normal operation even after the failure of one or more components. A number of activities, including regular and extra-regular audit of the system, accompanies providing fault tolerance in complex systems. The analysis allows determining the direction of system's further development. However, one of the most important results of the fault tolerance audit of the system is the identification of "bottlenecks".

To date, many organizations offer IT infrastructure audit services. Typically, the audit process consists of many stages and is based on empirical approaches and methodologies [1-4]. An audit helps to find answers to very important questions:

- determination of the critical IT services for business and identification of its dependence on individual elements of the data storage and processing system;
- identification of the limitations imposed by the existing data storage and processing system on IT services of the company and the formation of recommendations for the modernization of its individual components, training and certification of IT personnel, the customer, etc.;
- increasing the work efficiency of IT staff by formalizing key IT processes (data recovery, backup, recovery after failures, etc.);
- the ability to determine an effective strategy for further development of the system and justify the need for investment.

Thus, the issue of developing a universal methodology for identifying critical vulnerabilities as part of the fault tolerance audit of elements of distributed IT infrastructure subsystems remains urgent.

Building a reliable storage facility that can not only resist hardware failures, but also provide high speed data access for many clients simultaneously working with it, is a serious technical problem that does not have the only right decision. Every decision is a compromise in terms of many factors. Successful achievement of such a compromise is possible only on the basis of theoretical research, simulation modeling and experience in practical implementation of the data storage systems (DSS). The task is further complicated by the fact that there are almost no mathematical models that adequately describe DSS, built on the basis of the principles described above.

Methods of building fault-tolerant DSS based on multiple disk drives originated about 3 decades ago. These were the so-called Redundant Array of Independent Disks (RAID) — a series of disks combined into a single storage. Improving the reliability of such storage was achieved by storing redundant information (checksums) on one or more additional disks.

The first industrial system built on this principle was the Google cluster file system, intended for internal use by specialized Google applications. Similar ideas formed the basis for Azure DSS developed by Microsoft, as well as the open source CEPH system.

In a system consisting of hundreds of servers and thousands of disks, hardware and disk failures in particular are quite ordinary events. Accordingly, the software of such a system should provide automatic recovery after a hardware failure to minimize the chance of data loss. In this regard, the construction of an adequate mathematical model

of reliability is a decisive factor for creating a fault-tolerant and scalable data storage system.

2 Types of Distributed Data Storage Systems

Thus, a distributed system is a stand-alone computers that work together and combined into a single system. Their benefits include:

1) simplifying the integration of various applications into a single system;
2) scalability.

These benefits lead to more complex software modules, performance degradation and possible security problems.

In a distributed system, one computer is a server, all other machines are remote workstations. The basis of network interaction of distributed systems is OSI/ISO (open systems interconnection model). This model divides the process of interaction between the client and server into seven levels: physical, data link, network, transport, session, presentation and application.

The algorithm of interactions in open systems is described by standard protocols. The main protocol stack is TCP/IP. The transport layer protocol is TCP, and the network layer protocol is IP. The operating system is a link to the transport layer protocol and provides an interface for the upper layers based on sockets. Sockets provide low-level elementary operations for the direct exchange of a bit stream between two processes. There is no session or standard presentation layer in the TCP/IP protocol stack. These sometimes include secure SSL/TLS protocols.

Protocols can be divided into 2 main types: connection oriented and connectionless. When a data connection is established, the sender and recipient establish a connection, and after the transfer is completed, the connection is released. Without connection, the sender immediately sends the message to the addressee.

Socket based TCP/IP is a standard, cross-platform, but low-level service for exchanging data between components.

TCP works at the transport level, it is responsible for sending and receiving packets without data loss and in the correct order. The TCP transport protocol is called the Transmission Control Protocol and belongs to the basic stack of network protocols, since it implements all the services necessary for building network applications.

The TCP/IP protocol stack is a standard for network communications. Internet protocol stacks contain UDP (Universal Datagram Protocol, which does not require a connection), it is a universal datagram protocol and is a modification of IP.

Any interaction between remote components of a distributed system is based on TCP/IP.

3 Fault Tolerance in Distributed Systems

A characteristic feature of distributed systems is the possibility of a partial failure, a failure in one of the components of a distributed system. When creating a distributed system, it is important that it can be automatically restored after partial failures [5].

Modern information systems make high demands on software and hardware, and, in particular, on reliability and fault tolerance [6].

There are several possible reasons for failure in information systems. Statistics are shown in Fig. 1.

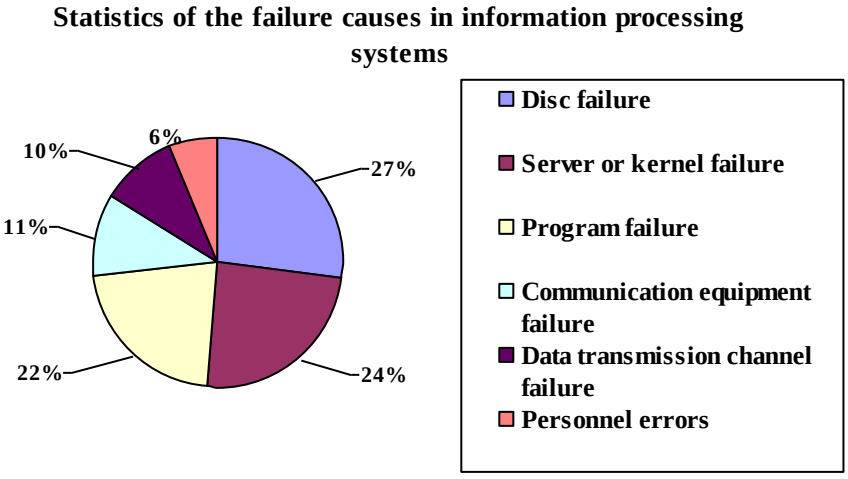


Fig. 1. Statistics of the failure causes in information processing systems

Solving the scientific task of identifying vulnerabilities in distributed systems with a further failure risk reduction is a relevant problem, which attracts the attention of researchers. Various mathematical approaches are used, including analysis and calculation of probabilistic characteristics, methods for determining the critical route, methods for solving dynamic programming problems, model-based testing methods, and algorithmic methods as well as their software implementations [7-8].

Owing to its simplicity, clarity and logical consistency, the mathematical apparatus of the theory of Markov processes found its application in the study of random processes in medicine and genetics [9-10], in the modeling of queuing systems [11] and in the financial and economic sphere [12], in modeling and choosing a strategy for managing socio-economic processes, in complex technical and information systems [13-18].

4 Mathematical Basis

The theory of Markov chains takes a worthy place in the course of studying random processes, usually depending on the parameter, which in most applications is time. The foundations of the theory were postulated by A. Markov at the beginning of the 20th century, and are valid for a random process whose evolution after any given value of the time parameter t does not depend on the evolution preceding t, if the value of the process at that moment is fixed. The apparatus of the theory of Markov chain processes is intended for a complete description of both the long-term and local dynamics of random sequences of events, including ones in steady state or limiting mode [19].

According to the ergodic theorem, under certain conditions the average over the ensemble coincides with the mean in time, that is, the same information can be obtained from long-term observation of the system and from simultaneous (and instant) observation of many independent copies of the same system [20-21].

In the application of the Markov chains to the study of the stability of distributed systems (for example, data storage systems) in order to identify their weaknesses, one of the important factors is a fairly long process, i.e. the course of the process after the termination of the initial conditions affecting it. Under certain conditions, in the end, the final stationary regime of the process is established, in which the probabilities of the states of the system $p_1 \dots p_n$ no longer depend on either time or the initial probability distribution. The vector $(p_1 \dots p_n)$ whose coordinates are the final probabilities is the final (limiting, stationary) vector and characterizes the average time of finding the system in each of its states.

In the studies [20-21] it was shown that for a homogeneous Markov chain, i.e. system S with discrete states $S_1, S_2, ..., S_n$ and discrete time, if the final probabilities exist, the final vector $(p_1 \dots p_n)$ can be found from the equation (1) subject to the normalization condition $p_1 + p_2 + \dots + p_n = 1.$

$$(p_1 \dots p_n) = (p_1 \dots p_n) * P; \tag{1}$$

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1r} \\ p_{21} & p_{22} & \cdots & p_{2r} \\ \cdots & \cdots & \cdots & \cdots \\ p_{r1} & p_{r2} & \cdots & p_{rr} \end{bmatrix},$$

where P is the matrix of transition probabilities.

Sufficient conditions for the existence of limit probabilities are formulated in the theorem that if a homogeneous Markov chain with a finite number (n) of states is regular, then there are final probabilities $p_1 \dots p_n$, and

$$\lim_{m\to\infty} p^m = \left.\begin{pmatrix} p_1 \dots p_n \\ \cdots\cdots\cdots \\ p_1 \dots p_n \end{pmatrix}\right\} \text{n rows.}$$

A regularity condition for a Markov chain is the existence of a natural number m such that any element of the matrix P^m is positive except for elements in columns whose numbers coincide with the unstable states of the system S.

Thus, to find the final probabilities, it is necessary to check the Markov chain for regularity, and if it turns out to be so, then it is possible find the final stationary probability vector from equation (1).

The mathematical apparatus of the Markov chain theory is a good variant of a combination of simplicity and quality for solving such problems. It makes it possible to identify critical nodes in terms of fault tolerance based on the analysis of statistics. The simplicity of the algorithmic implementation of the proposed approach should be noted, which is an important factor in automating the process of identifying critical vulnerabilities in distributed systems.

5 Application Example

As an example of the application of the theory of stationary processes in the basis of the methodology for identifying critical vulnerabilities in distributed systems, network traffic was collected between 10 stationary nodes in a distributed data warehouse system [22]. It is worth noting that the proposed algorithm can be applied to solve such problems for real distributed systems, which can have a much larger number of nodes.

Quantitative characteristics of the amount of information transmitted from node to node were recorded during the month. The obtained data were partially processed and tested for relevance. The averaged statistical data were summarized in a table characterizing the amount of information aij transmitted from node i to node j (Table 1). For the demonstration example, several generalizations are accepted:

- the size of the files is taken into account, and not their number;
- data warehouses physical implementation is not taken into account;
- network infrastructure and remoteness of nodes are not considered.

Table 1. Data transfer matrix

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**
A	65,8	87,8	329,1	373,0	87,8	263,3	394,9	109,7	109,7	373,0
B	38,7	88,5	38,7	5,5	38,7	110,6	83,0	16,6	77,4	55,3
C	15,7	0	156,8	78,4	0	78,4	86,2	164,6	39,2	164,6
D	82,8	20,7	207,0	41,4	124,2	113,9	155,3	62,1	51,8	176,0
E	19,5	39,0	185,1	19,5	39,0	77,9	19,5	243,5	39,0	292,2
F	367,4	183,7	102,1	61,2	102,1	224,5	40,8	326,6	285,7	347,0
G	95,6	54,6	75,1	88,8	0,0	116,1	34,2	109,3	34,2	75,1
H	5,2	15,5	11,3	7,2	18,5	4,1	15,5	3,1	4,1	18,5

I	12,2	44,6	40,5	48,6	12,2	0	89,1	4,1	60,8	93,2
J	243,6	299,8	281,1	243,6	112,4	262,4	37,5	18,7	281,1	93,7

The normalized data in the form of probabilities of transmission of outgoing traffic between nodes in the system are given in Table 2.

Table 2. Probability matrix

		Receiver									
		A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**
Source	**A**	0,03	0,04	0,15	0,17	0,04	0,12	0,18	0,05	0,05	0,17
	B	0,07	0,16	0,07	0,01	0,07	0,2	0,15	0,03	0,14	0,1
	C	0,02	0	0,2	0,1	0	0,1	0,11	0,21	0,05	0,21
	D	0,08	0,02	0,2	0,04	0,12	0,11	0,15	0,06	0,05	0,17
	E	0,02	0,04	0,19	0,02	0,04	0,08	0,02	0,25	0,04	0,3
	F	0,18	0,09	0,05	0,03	0,05	0,11	0,02	0,16	0,14	0,17
	G	0,14	0,08	0,11	0,13	0	0,17	0,05	0,16	0,05	0,11
	H	0,05	0,15	0,11	0,07	0,18	0,04	0,15	0,03	0,04	0,18
	I	0,03	0,11	0,1	0,12	0,03	0	0,22	0,01	0,15	0,23
	J	0,13	0,16	0,15	0,13	0,06	0,14	0,02	0,01	0,15	0,05

The time interval between the transmissions of outgoing traffic between nodes cannot be arbitrarily small and therefore the time instants $t_1, t_2 \dots t_n$ can be chosen so close to each other that the system does not change its state between them. Consequently, the process taking place in the system can be considered a discrete-time process, and Table 2 in terms of Markov chains can be interpreted as a matrix of transition probabilities.

An analysis of the data in Table 1 and the corresponding labeled graph allows to conclude that the system does not have unstable states [23]. Therefore, in order for the Markov chain to be regular and the system to have final probabilities it is necessary and sufficient that there be a natural number m such that all the elements of P^m are positive. This condition is met when m = 2, which allows to conclude that the Markov chain is regular.

When m = 2 (Table 3), all elements of the matrix are strictly greater than zero, $p_{ij} > 0$, i.e. the necessary and sufficient condition of regularity is satisfied, which allows us to conclude that the Markov chain under consideration is regular and the final probabilities exist and can be determined.

Table 3. Matrix P^2

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**
A	0,09	0,08	0,14	0,09	0,05	0,12	0,09	0,11	0,09	0,15
B	0,09	0,10	0,11	0,08	0,04	0,12	0,10	0,10	0,11	0,16
C	0,09	0,09	0,14	0,09	0,07	0,10	0,09	0,09	0,08	0,15
D	0,08	0,08	0,14	0,09	0,04	0,11	0,08	0,12	0,08	0,17
E	0,08	0,11	0,14	0,09	0,08	0,10	0,09	0,08	0,09	0,15
F	0,07	0,10	0,12	0,09	0,07	0,10	0,12	0,07	0,10	0,16
G	0,08	0,09	0,13	0,08	0,07	0,11	0,11	0,08	0,09	0,16
H	0,08	0,09	0,14	0,08	0,05	0,13	0,08	0,11	0,09	0,16

I	0,09	0,09	0,14	0,10	0,04	0,12	0,10	0,08	0,10	0,14
J	0,07	0,08	0,13	0,08	0,05	0,11	0,12	0,09	0,10	0,17

In order to identify the potential vulnerabilities of fault tolerance of distributed system nodes, we use (1) to form the equations system (2).

$$\begin{cases} p_1 = 0{,}03p_1 + 0{,}07p_2 + 0{,}02p_3 + 0{,}08p_4 + 0{,}02p_5 + 0{,}18p_6 + 0{,}14p_7 + 0{,}05p_8 + 0{,}03p_9 + 0{,}13p_{10} \\ p_2 = 0{,}04p_1 + 0{,}16p_2 + 0{,}02p_4 + 0{,}04p_5 + 0{,}09p_6 + 0{,}08p_7 + 0{,}15p_8 + 0{,}11p_9 + 0{,}16p_{10} \\ p_3 = 0{,}15p_1 + 0{,}07p_2 + 0{,}2p_3 + 0{,}2p_4 + 0{,}19p_5 + 0{,}05p_6 + 0{,}11p_7 + 0{,}11p_8 + 0{,}1p_9 + 0{,}15p_{10} \\ p_4 = 0{,}17p_1 + 0{,}01p_2 + 0{,}1p_3 + 0{,}04p_4 + 0{,}02p_5 + 0{,}03p_6 + 0{,}13p_7 + 0{,}07p_8 + 0{,}12p_9 + 0{,}13p_{10} \\ p_5 = 0{,}04p_1 + 0{,}07p_2 + 0{,}12p_4 + 0{,}04p_5 + 0{,}05p_6 + 0{,}18p_8 + 0{,}03p_9 + 0{,}06p_{10} \\ p_6 = 0{,}12p_1 + 0{,}2p_2 + 0{,}1p_3 + 0{,}11p_4 + 0{,}08p_5 + 0{,}11p_6 + 0{,}17p_7 + 0{,}04p_8 + 0{,}14p_{10} \\ p_7 = 0{,}18p_1 + 0{,}15p_2 + 0{,}11p_3 + 0{,}15p_4 + 0{,}02p_5 + 0{,}02p_6 + 0{,}05p_7 + 0{,}15p_8 + 0{,}22p_9 + 0{,}02p_{10} \\ p_8 = 0{,}05p_1 + 0{,}03p_2 + 0{,}21p_3 + 0{,}06p_4 + 0{,}25p_5 + 0{,}16p_6 + 0{,}16p_7 + 0{,}03p_8 + 0{,}01p_9 + 0{,}01p_{10} \\ p_9 = 0{,}05p_1 + 0{,}14p_2 + 0{,}05p_3 + 0{,}05p_4 + 0{,}04p_5 + 0{,}14p_6 + 0{,}05p_7 + 0{,}04p_8 + 0{,}15p_9 + 0{,}15p_{10} \\ p_{10} = 0{,}17p_1 + 0{,}1p_2 + 0{,}21p_3 + 0{,}17p_4 + 0{,}3p_5 + 0{,}17p_6 + 0{,}11p_7 + 0{,}18p_8 + 0{,}23p_9 + 0{,}05p_{10} \\ p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_7 + p_8 + p_9 + p_{10} = 1 \end{cases} \quad (2)$$

The result (2) is a system of linear algebraic equations. The solution of the system can be obtained by one of the known methods: Gaussian elimination, Cramer's rule, etc.

The solution of system (2) is the vector of final probabilities (3).

$$\begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \\ p_8 \\ p_9 \\ p_{10} \end{pmatrix} = \begin{pmatrix} 0{,}081 \\ 0{,}089 \\ 0{,}133 \\ 0{,}087 \\ 0{,}057 \\ 0{,}11 \\ 0{,}101 \\ 0{,}093 \\ 0{,}091 \\ 0{,}158 \end{pmatrix} \quad (3)$$

The probability values determine the average time of receiving information by the corresponding node [24]. With the same technical characteristics of the nodes, the most loaded elements of the system would be nodes J and C. However, it should be noted that the technical characteristics of the nodes are generally not the same. Therefore, in order to correctly compare the risk of their failure, it is advisable to take into account both the final probabilities of the nodes, which characterize the average time they are in the active state, and the total or average load values on them. The product of these two indicators (Table 3) allows us to estimate the risk of failure or unstable operation of the element, and the correlation of this value with the maximum

permissible level of load (technical characteristic of the nodes) suggests a safety margin of each of the nodes.

Table 4. The product of the final probabilities by the average load values

	P lim	P lim * Average
A	0,081	17,7714
B	0,089	4,9217
C	0,133	10,4272
D	0,087	9,0045
E	0,057	5,5518
F	0,11	22,451
G	0,101	6,8983
H	0,093	0,9579
I	0,091	3,6855
J	0,158	29,6092

Thus, the nodes J, C and F of the system are in the state of obtaining information on average 15.8%, 13.3% and 11% of the time, respectively, the maximum amounts of received information fall on J (29.6092), F (22.451) and A (17.7714), and node C (10.4272) is not so critically loaded.

As a rule, distributed systems have a significant number of nodes and the calculation of probability indicators characterizing the vulnerability of the system is not an easy task, requiring cumbersome calculations. To eliminate this difficulty, a design was carried out and a software application was implemented for solving the problems of identifying bottlenecks in information and communication systems.

In the course of designing a software application for identifying the most loaded node of a distributed system, the task of developing a single-user application that implements a search algorithm based on an inter-node data transfer matrix was considered and implemented. The following requirements were imposed on the developed application:

- it should be possible to enter the input data both manually and from an external file;
- there should be a feature of creating a load chart for the system nodes;
- there should be a feature of saving the chart as an image.

A functional model of the application was built by the analysis of the subject area. 10 use cases are shown in the UML use case diagram (Fig. 2).

The program user can directly create and open input data files. After opening the file, it is possible to edit it. Two save options are available while editing:

- save to the current file;
- save to another file.

- After making the necessary changes, the user can determine the load on the nodes of the distributed system and view two charts:
- chart of the load on the system nodes;
- final probabilities chart.

The chart can be saved to a file.

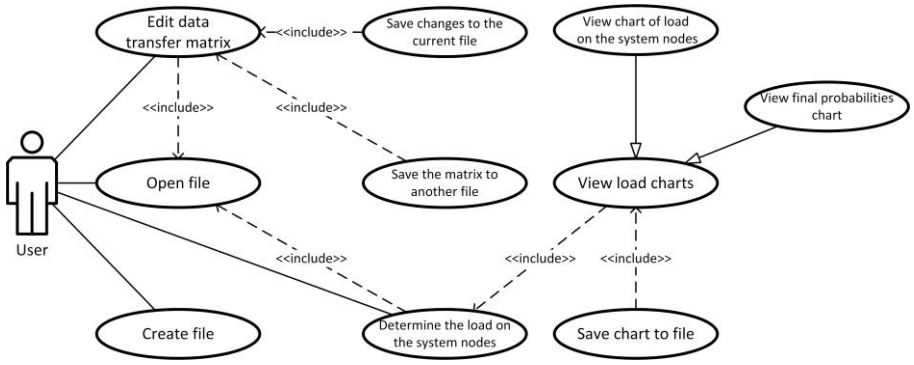


Fig. 2. UML use case diagram

When designing a software system, a sequence of user actions was determined to estimate the load on the system nodes based on the input data file. When opening or creating an input data file, the user gets access to the form of editing the data transfer matrix with the corresponding menu, which allows saving changes to the current or new file. After the validation of the entered data, the application proceeds to the stage of finding a solution to the problem with an indication of the results on the tab with the final calculation table. The user interface of the application allows switching between the following tabs:

- result table;
- input data;
- chart of load on the system nodes;
- final probabilities chart.

The chart can be saved to a file at runtime. The file can be closed upon work completion and choosing the option of saving information.

As a result of subject area object-oriented analysis, a UML class diagram was made, which contains 6 classes (Fig. 3).

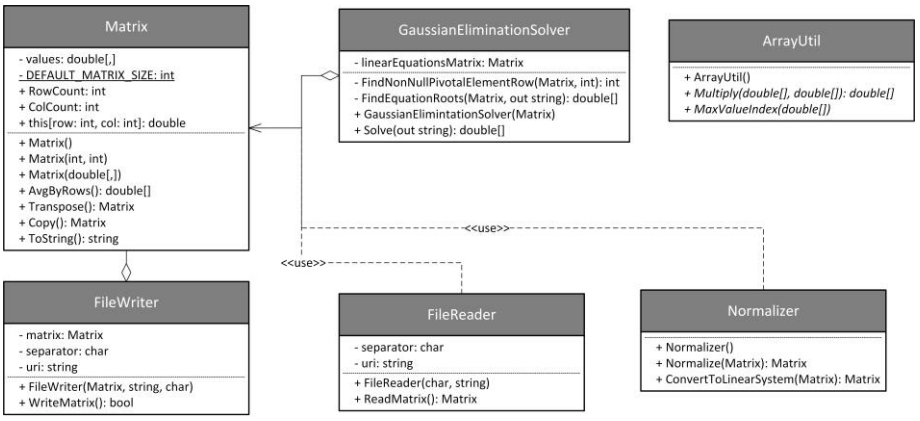


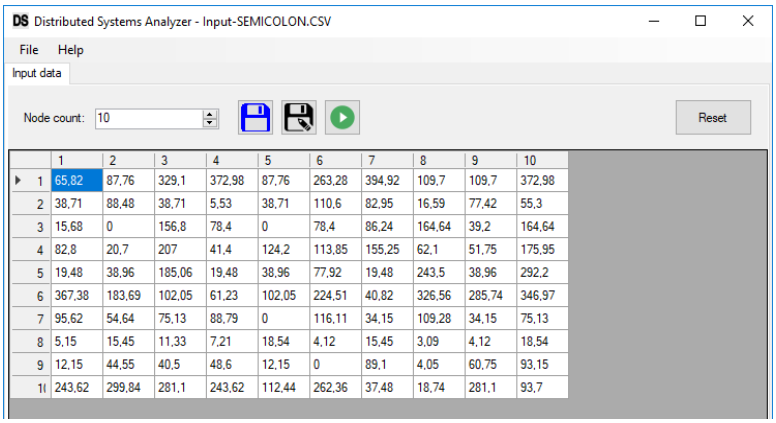
Fig. 3. UML class diagram

The software system is written in the C# programming language using IDE Visual Studio 2017 and .NET Framework 4.5. The developed system supports three input data file formats:

- comma-separated CSV files;
- semicolon-separated CSV files;
- tab-delimited text files.

When saving a file, the decimal separator set in the regional standards of the operating system for the current user is used (in comma-separated CSV files the decimal separator is always a dot).

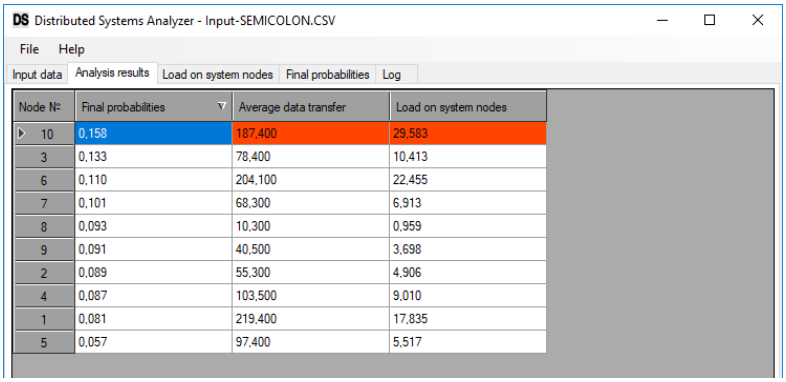
After opening the input data file, the “Input data” tab will be shown, on which you can edit the data transfer matrix (Fig. 4). Changes can be saved. The number of nodes can be changed in the “Node count” field (the value must be in the range from 2 to 100). Click the “Calculate” button to solve the problem. The “Reset” button is used to reset the changes.



	1	2	3	4	5	6	7	8	9	10
1	65,82	87,76	329,1	372,98	87,76	263,28	394,92	109,7	109,7	372,98
2	38,71	88,48	38,71	5,53	38,71	110,6	82,95	16,59	77,42	55,3
3	15,68	0	156,8	78,4	0	78,4	86,24	164,64	39,2	164,64
4	82,8	20,7	207	41,4	124,2	113,85	155,25	62,1	51,75	175,95
5	19,48	38,96	185,06	19,48	38,96	77,92	19,48	243,5	38,96	292,2
6	367,38	183,69	102,05	61,23	102,05	224,51	40,82	326,56	285,74	346,97
7	95,62	54,64	75,13	88,79	0	116,11	34,15	109,28	34,15	75,13
8	5,15	15,45	11,33	7,21	18,54	4,12	15,45	3,09	4,12	18,54
9	12,15	44,55	40,5	48,6	12,15	0	89,1	4,05	60,75	93,15
10	243,62	299,84	281,1	243,62	112,44	262,36	37,48	18,74	281,1	93,7

Fig. 4. Input data tab

After clicking on the “Calculate” button, the load on the nodes of the distributed system will be calculated. The “Analysis results” tab displays the results of solving the problem. The row in the table highlighted in red contains a record of the most loaded node in the system (Fig. 5).



Node №	Final probabilities	Average data transfer	Load on system nodes
10	0,158	187,400	29,583
3	0,133	78,400	10,413
6	0,110	204,100	22,455
7	0,101	68,300	6,913
8	0,093	10,300	0,959
9	0,091	40,500	3,698
2	0,089	55,300	4,906
4	0,087	103,500	9,010
1	0,081	219,400	17,835
5	0,057	97,400	5,517

Fig. 5. Analysis results tab

The “Load on system nodes” tab displays the load chart for the nodes of the distributed system (Fig. 6). The load is defined as the product of the final probabilities of receiving traffic by system nodes by the average amount of data transferred.

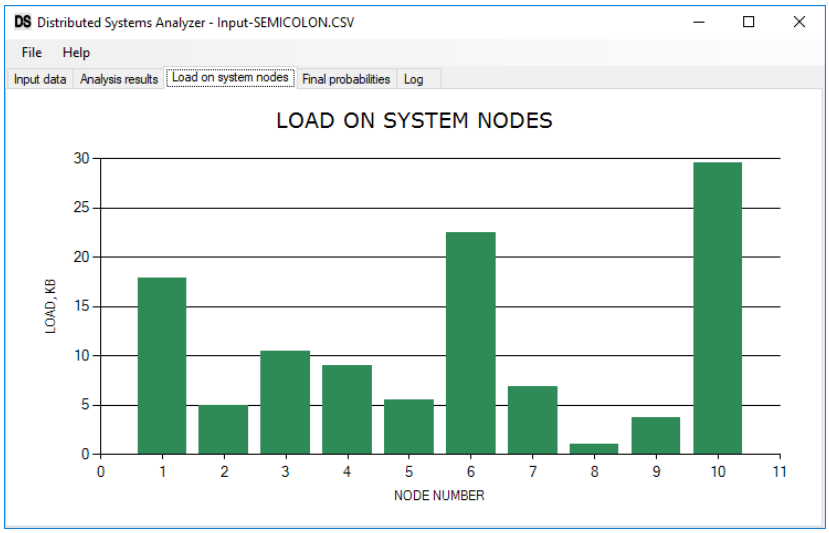


Fig. 6. Load on system nodes tab

The “Final probabilities” tab displays the final probabilities chart (Fig. 7). The final probability represents the probability of failure of a distributed system node.

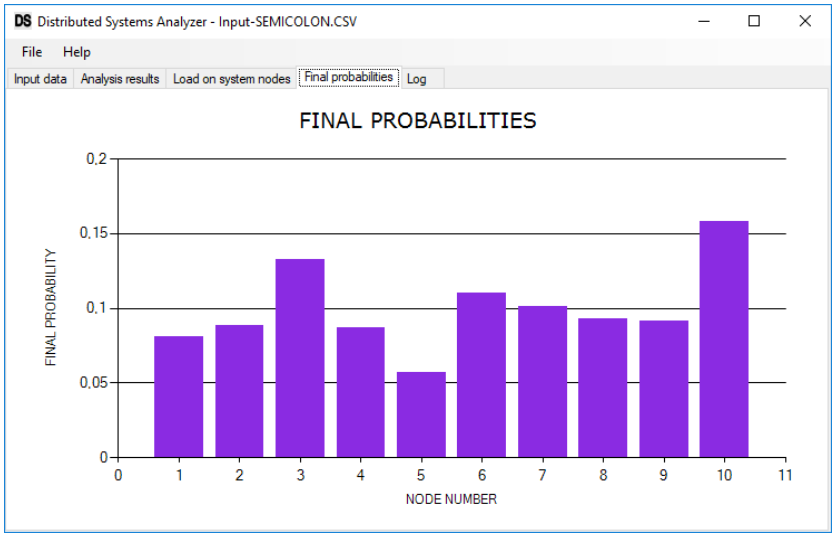


Fig. 7. Final probabilities tab

The “Log” tab displays the history of solving the system of linear equations using the method of Gaussian elimination

Software performance testing was carried out on matrices of different sizes. The graph of the average system operation time versus the number of nodes is shown in Fig. 8.

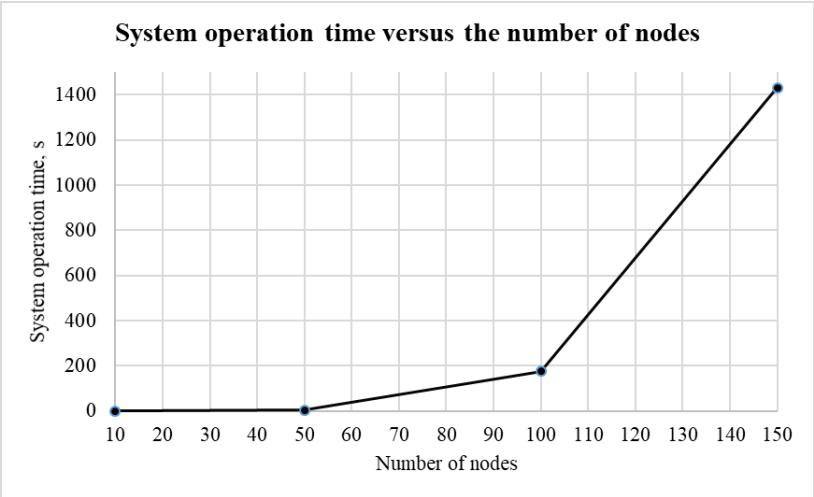


Fig. 8. The graph of system operation time versus the number of nodes

6 Conclusion

The application of the apparatus of the theory of stationary processes makes it possible to determine critical vulnerabilities in distributed systems based on statistical data. The methodology provides a clear and mathematically substantiated answer to the question posed by the definition of a potentially vulnerable component of the system. The application of the methodology is possible to solve a whole class of problems in various aspects of information and infocommunication technologies.

It is planned to adapt the vulnerability search algorithms for large-dimension matrices, to apply numerical solution methods and multithreading in further software development.

References

1. Tuncer, O., Ates, E., Zhang, Y., A.Turk, et. al.: Online Diagnosis of Performance Variation in HPC Systems Using Machine Learning. IEEE Transactions on Parallel and Distributed Systems 30(4), 883 – 896 (2019).

2. Ramabhadran, S., Pasquale, J.: Analysis of long running replicated systems. In: 25th IEEE International Conference on Computer Communications (INFOCOM-2006), pp. 1-9. IEEE (2006).
3. Ganesh, L., Weatherspoon, H., Marian, T., Birman, K.: Integrated Approach to Data Center Power Management. IEEE Transactions on Computers. 62(6), 1086-1096 (2013).
4. Kavun, S., Zamula, A., Mikheev, I.: Calculation of expense for local computer networks. 2017 4th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T). 146-151 (2017).
5. Apache Hadoop 3.2.1 – HDFS Architecture, https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html.
6. Kalashnikov, V., Dempe, S., Mordukhovich, B., Kavun, S.: Bilevel Optimal Control, Equilibrium, and Combinatorial Problems with Applications to Engineering. Mathematical Problems in Engineering. 2017, 1-3 (2017).
7. Zakharchenko, S.: Approach for quantitative estimation of the results of vulnerabilities search in parallel software systems using the model test method. Proceedings of Petersburg Transport University. 1, 64-69 (2014).
8. Pikalov, A., Galimov, R.: Analysis of methods for increasing resiliency of a wireless network distributed systems of control and management of technological objects. Privolzhsky Scientific Herald. 6(58), 23-27 (2016).
9. Škulj, D.: Discrete time Markov chains with interval probabilities. International Journal of Approximate Reasoning. 50, 1314-1329 (2009).
10. Lee, S., Ko, J., Tan, X., Patel, I., Balkrishnan, R., Chang, J.: Markov Chain Modelling Analysis of HIV/AIDS Progression: A Race-based Forecast in the United States. Indian J Pharm Sci. 76(2), 107-115 (2014).
11. Stefanyuk, V.: Deterministic Markov chains. Information processes. 11(4), 423–427 (2011).
12. Lin, S., Li, Y., Li, Y., Ai, B., Zhong, Z.: Finite-state Markov channel modeling for vehicle-to-infrastructure communications. 2014 IEEE 6th International Symposium on Wireless Vehicular Communications (WiVeC 2014). (2014).
13. Jones, E., Epstein, D., García-Mochón, L.: A Procedure for Deriving Formulas to Convert Transition Rates to Probabilities for Multistate Markov Models. Medical Decision Making. 37, 779-789 (2017).
14. Cheng, C., Chiu, S., Cheng, C., Wu, J.: Customer lifetime value prediction by a Markov chain based data mining model: Application to an auto repair and maintenance company in Taiwan. Scientia Iranica. 19, 849-855 (2012).
15. Petrov, E., Kharina, N., Kharyushin, V.: Mathematical models and algorithms of filtering digital halftone images on the basis of complex Markov chains. Digital signal processing. 3, 52-57 (2012).
16. Serzhantova, M., Ushakov, A.: Finite Markov chains in the model representation of the human operator activity in quasi-functional environment. Scientific and Technical Journal of Information Technologies, Mechanics and Optics. 16(3), 524-532 (2016).
17. Kolesnikova, K.: Applied aspects of the use of Markov chains to model weakly structured project management systems. Information technologies in education, science and production. 4(5), 77-82 (2013).
18. Kavun, S.: Conceptual fundamentals of a theory of mathematical interpretation. International Journal of Computing Science and Mathematics. 6, 107 (2015).
19. Markov, A.: Selected Works. The Theory of Numbers. Probability Theory. Publishing house of the Academy of Sciences, Moscow (1951).
20. Kemeny, J., Snell, J.: Finite Markov chains. Springer-Verlag, New York (1976).
21. Labsker, L.: Probabilistic modeling in the financial and economic area. Alpina Publisher, Moscow (2002).
22. Zhang, G., Zheng, W., Shu, J.: ALV: A New Data Redistribution Approach to RAID-5 Scaling. IEEE Transactions on Computers. 59, 345-357 (2010).

23. Kavun, S., Mykhalchuk, I., Kalashnykova, N., Zyma, O.: A Method of Internet-Analysis by the Tools of Graph Theory. Intelligent Decision Technologies. 15, 35-44 (2012).
24. Mikheev, I., Shapovalova, O., Burmensky, R.: Methodology for Identifying Critical Vulnerabilities in Distributed Systems. 2018 5th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T). 654-658 (2018).