УДК 004.77

S.F. Telenyk, O.V. Savchuk, E.O. Pocrovskyi, O.M. Morgal, O.A. Pokhylenko
NTUU "Igor Sikorskyi KPI", Ukraine
41, Polytechnic St., Kyiv, 03056

ON RELIABILITY MODELING AND EVALUATING IN CLOUD SERVICES SYSTEM

С.Ф.Теленик, О.В. Савчук, Є.О. Покровський, О.М. Моргаль, О.А. Похиленко
НТУУ "КПІ ім. Ігоря Сікорського", Україна
вул. Політехнічна, 41, м. Київ, 03056

ПРО МОДЕЛЮВАННЯ НАДІЙНОСТІ ТА ОЦІНЮВАННЯ В СИСТЕМІ ХМАРНИХ СЕРВІСІВ

A typical system of reliability of cloud services with a cloud management system is explored. The proposed model is service-oriented and hierarchical. It includes two phases of query maintenance: 1) time-out and overflow; 2) execution. The model is described by the system of Chapman-Kolmogorov equations. The final stage of the process is protection of user requests and results of tasks execution. An approach to cloud data security is proposed which, unlike traditional approaches, guarantees confidentiality and security of information even at compromising the provider of cloud service or its infrastructure.

Key words: cloud services system, data security, reliability modeling

Досліджується типова система надійності хмарних сервісів з системою керування хмарами. Пропонована модель сервісно-орієнтована та ієрархічна. Вона включає дві фази обслуговування запитів: 1) попередня (тайм-аут і переповнення); 2) виконання. Модель описується системою рівнянь Чепмена-Колмогорова. Останній етап процесу - захист запитів користувачів та виконання завдань. Запропоновано підхід до захисту даних хмарних даних, який, на відміну від традиційних підходів, гарантує конфіденційність та безпеку інформації, навіть при компрометації постачальника хмарних сервісів або його інфраструктури.

Ключові слова: система хмарних сервісів, безпека даних, моделювання надійності

Introduction

Recently at the diagnosing of the technical infrastructure and its components there is more demand determine the physical condition of the facility, and very important information about the properties of the physical environment depending on the type of hidden or overt defects and performance degradation or various destructive processes. All these processes cause the possibility of changing the technical condition. And this requires the development of appropriate methods of predicttive diagnosis of equipment based on the detection of relevant signs of change in their environment. The transition from production and implementation to the service sector sets new requirements and provides new opportunities for software.

The international standard ISO 9001: 2015 contains characteristics that allow you to evaluate software from the perspective of a user, developer and project manager. We recommend 6 main characteristics of software quality, each of which is detailed by several (only 21) subcharacteristics.

Functionality is detailed with suitability for use, precision, security, ability to interact and consistent with standards and design rules. Software reliability is characterized by a level of completeness (no errors), error tolerance and restart. Applicability is characterized by an intelligibility, training and ease of use. Efficiency is characterized by resourceful and temporary economic efficiency. Accompaniment is characterized by convenience for analysis, variability, stability and test. Tolerability is characterized by adaptability, structuring, substitution and implementation. Characteristics and subcharacteristics in the standard are defined very briefly, without comments and recommendations for their application to specific systems and projects.

Cloud computing enables the massive-scale service sharing, which allows for users to access technology - enabled services without a knowledge of, an expertise with, or a control over the technology of infrastructure that supports them. Cloud computing is different from, but is associated with distributed (grid), utility computing and transparent computing. Transparent computing [1,2] means that complex reverse services are transparent to users who see only a simple interface that is convenient to use.

The main advantages of cloud computing services are: a self-service, wide network access, resource pooling, fast scaling. Despite these benefits, the widespread use of this new technology faces a number of obstacles, including security and privacy. In addition to traditional security risks, as with any cloud computing system connected to the Internet, cloud systems have particular security and privacy issues due to virtualization and their multilevel nature [3].

The reliability of cloud computing is difficult to analyze due to the characteristics of the massive-scale service sharing, wide-area network, heterogeneous of software and hardware components and complicated interactions among them. Therefore, reliability models for pure software/hardware or conventional networks can not be simply applied to study the reliability of the cloud [1].

Problems

The problems to be solved:

- Explore the cloud computing reliability model to provide convenient, on-demand access to a common pool of computing resources with minimal management effort from the service provider.
- Ensure a protection of user information before access to the generic pool of computing resources.

Description of the cloud computing system

The architecture of the typical cloud service system is depicted in Fig. 1, which is also a typical representation of most modern or future cloud service systems [1,4]. In the middle of there is a cloud management system (CMS), which consists of a set of servers (either centralized or distributed). The CMS mainly fulfils four different functions, namely: 1) to manage a request queue that receives job requests from various users for cloud services; 2) to manage computing resources (such as PCs, clusters, supercomputers, etc.) above the Internet; 3) to manage data resources (for example, databases, published information, content of the URL, etc.) all over the Internet; 4) to schedule a request and divide it into different subtasks; assigning subtasks to different computing resources that may have access different data resources over the Internet.

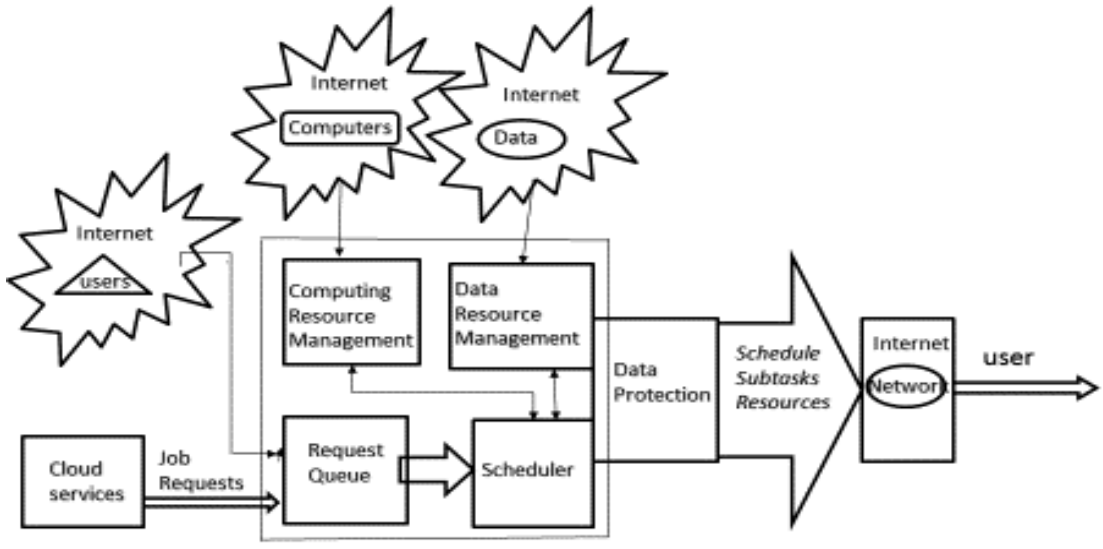

Fig. 1. Cloud Service System

When a user requests a certain given cloud service, a workflow template is provided for describing and managing the cloud service [5]. The service workflow template includes various subtasks *Si* (*i=1,…,m*) and their interrelationship (data dependency). It also shows the required data resources to which subtasks need to be accessed, for example, at startup. With the cloud service workflow, the scheduler in the CMS can assign these subtasks to different computing resources *Kj* (*j = 1,…,n*) when allocating data resources. While the computing resources *Kj* and data resources *Di* receive commands or subtasks from the CMS, they form the network according to the connectivity or accessibility, for example, K_2 is directly related to K_3, but can not directly communicate with K_4 due to the connectivity (for example, computers K_2 and K_3 may be both behind routers that translate their original IP addresses so that they cannot directly build the TCI/IP connection, or they do not have access to each other).

The network of clouds can be very large, and each link is a virtual link, which may go through many components (routers / cables / optical fibers / machines) over a long distance. Thus, computing resources will work together via the network to run the subtasks while accessing necessary data from their data resources. When the job is finished, the results will return to the user who request this service, as shown in Fig. 1.

There are a variety of types of failures that may affect to the reliability of a cloud service, including overflow, time-out, data resource missing, computing resource missing, software and hardware failures, database failure and network failure. It is impossible to provide nothing of failures in complex software designing, due to the reliability of them functioning always has a finite, limited value. It is assumed that there is a correct, reference state of the object in relation to which the presence of a failure may be determined.

For systematic, coordinated struggle with failures it is necessary to investigate factors that affect the reliability of software from random, existing potential possible defects in specific programs, and in extraordinary circumstances diagnostic tests of the system may be performed. If under these situations fairly rapid recovery will happen, such that no refusal is fixed, then such events do not affect on the main indicators of reliability – the amount of failure and the readiness coefficient.

Hence, the reliability of the operation of programs is a dynamic concept, which manifests itself in time and differs significantly from the notion of the correctness of programs. The basic principle of classification of accidents, failures and errors in programs in the absence of their physical destruction is the time division by the duration of recovery after any distortion of programs, data or computational process recorded as disabled. Consider the possible failures [1]: *Overflow* (failure): queue queries due to restrictions on the maximum number of queries. *Time-out:* - a waiting time (or completion) failure that is set by the user or service monitor [5]. *Data resource missing* is due to the fact that some previously registered data have been deleted, but the data resource manager has not been updated. *Computing resource missing:* is due to the fact that the computer turns off without notifying the CMS. *Software failure*: is because subtasks work on various computing resources and contain software malfunctions [6]. *Database failure:* is that stores the necessary data resources takes place, and subtasks at work can not access the required data. *Hardware failure:* is due to the fact that computing resources and data resources generally have hardware (such as computers or servers) that may also fail.

In addition, these different types of failures are not independent in the cloud service. The proposed cloud reliability model is service-oriented and hierarchical. It comprehensively examines different types of errors that have a significant impact on the success or failure of cloud services, including overflow, time-outs, data resource missing, computing resource missing, software failure, database failure, hardware and network failures. And the final stage of management is the protection of data and computing, which is

also carried out by the object-oriented approach.

Cloud services reliability modeling

1. Determination of the probability of failures such as time out and overflow

In [6] there is a simple classification of the above errors into two groups for the life cycle: 1) Request phase failures: overflow and timeout. 2) Execution phase failures: data resource missing, computing resource missing, software failure, database failure, hardware failure and network failurers. These two groups of bounces may be considered independent. However, the failure in each group strongly correlates. Thus, the simulation of the reliability of cloud services may be divided into two parts: simulation of reliability at the request phase and modeling the reliability of the implementation phase.

In the first step, if the job request is not executed by the scheduler before the set time, it will be discarded. The rejection rate is denoted by μr. Assume that the queue of the request is N. It is assumed that the arrival of applications for work is subject to the Poisson process with the arrival rate λa.

Usually, there are multiple schedule servers to serve the requests. Let the total number of S homogeneous schedule servers run simultaneously to execute queries. The service time for completing one request for each such server is considered to be exponentially distributed with the μs parameter. Thus, such a process can be modeled using the Markov process with expectation, in which the state n (n = 0,1, ..., N) is the number of requests (Fig. 2) [1,7,8]:

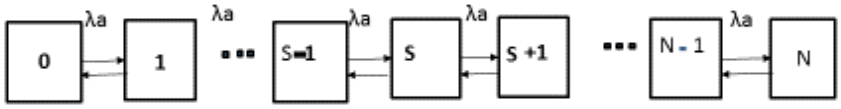

Fig. 2. Markov model for request queue [1].

Failures are detected with intensity λ, and corrected with the intensity μ, The service rate from the state n to the state $n + 1$ is λa. At the state N, the arrival of a new request will make the request queue overflow, so the request is dropped, and the queue will still stays at state N. The service rate of a request by scheduler server is μr. If $n \leq S$, then n requests can be immediately serviced by S scheduler servers, so the departure rate of any one request is nμr. If $n > S$, only S request are being simultaneously serve by shedule servers, so the departure rate $S\mu r$. The dropping rate for any one request in the queue to reach its due time is $n\mu d$ (n = 1,2, ..., N). Denote by q_n the stable probability that the system will remain in the state n (n = 0,1, ..., N), where qn is deduced by solving the Chapman-Kolmogorov equations [1,8]:

$$R_{\text{overflow}} = \sum_{n=0}^{N-1} q_n, \qquad (1)$$

$$\sum_{n=0}^{N} q_n = 1, \qquad (2)$$

where $R_{overflow}$ is the probability of that an overflow failure will not occur, q_n ($n = 0,1, ..., N$).

If $n < S$, then the new request that has arrived can be immediately served without any waiting time. Therefore, the probability for the time-outs and overflows are not to occur (i.e., the request phase is reliable):

$$R_{request} = \sum_{n=0}^{S-1} q_n + \sum_{n=S}^{N-1} q_n \int_0^{T_3} f_n(t)dt \qquad (3)$$

where $f(t)$ is the probability density of time-outs. The sum in (3) between [0, $N-1$] contains the condition that a failure due to overflow does not occur, as analyzed in (1).

1)

If some software module contains a failure, identical "backup" modules will also contain the same error. Therefore, the next step is to correct failures by the system itself.

2. The first phase of cloud computing reliability modeling

The preparatory phase of modeling will be carried out in the form of a mass maintenance system (MMS). Define the characteristics of all its main elements: the characteristics of the flow of requests and the source of requests; rule of queuing; characteristics of the service system (characteristics of the service law, number of channels, number of service phases, service rule). For systems without loss (with unlimited expectations), the fairly important indicator of service quality is: the average number of requests in the queue, the average number of system requirements, the average waiting time for the queue requirement, the average time required to stay in the system, the idle factor or the load factor of the service system and other.

Usually the operation of the MMS is described by "birth and death" process. The heterogeneity implies a difference both in the form of the received information and in the characteristics of the flow and the characteristics of service requirements from different streams. Taking into account the physical nature of the formation of flows, one can determine the law of distributing the intervals between the requirements of the flow.

Calculated expressions for the probabilities of states of the system of MMS [9]:

$$p_k = \frac{\alpha^k/k!}{\sum_{k=0}^{n}(\alpha^k/k! + \alpha^n/n!)\sum_{s=1}^{\infty}[\alpha^s/\prod_{m=1}^{s}(n+m\beta)]}, \quad (0 \le k \le n) \qquad (4)$$

$$p_{n+s} = \frac{[\alpha^n/n!][\alpha^s/\prod_{m=1}^{s}(n+m\beta)]}{\sum_{k=0}^{n}[\alpha^k/k! + \alpha^n/n!]\sum_{s=1}^{\infty}[\alpha^s/\prod_{m=1}^{s}(n+m\beta)]}, \quad (s \ge 0) \qquad (5)$$

When we know the probabilities of all states of the system, it is easy to determine the probability of P_H that the request will leave the system unattended: this is the ratio of the average number of requests from the queue per unit time to the average number of requests received per unit time. Find the average number of requests coming from the queue per unit time. To do this, we first calculate the mathematical expectation m_s, the number of requests that are in the queue:

$$m_s = M[s] = \sum_{s=1}^{\infty} s p_{n+s} = \frac{\alpha^n/n![\sum_{s=1}^{\infty} s\alpha^s/\prod_{m=1}^{s}(n+m\beta)]}{\sum_{k=0}^{n}(\alpha^k/k!+\alpha^n/n!)\sum_{s=0}^{\infty}[\alpha^s/\prod_{m=1}^{s}(n+m\beta)]} \qquad (6)$$

To obtain P_H, someone must multiply the m_s request by the average density of the "waste" of one

$$P_H = \frac{\beta}{\alpha}\frac{\alpha^n/n!\sum_{s=1}^{\infty}[s\alpha^s/\prod_{m=1}^{s}(n+m\beta)]}{\sum_{k=0}^{n}(\alpha^k/k!+\alpha^n/n!)\sum_{s=1}^{\infty}[\alpha^s/\prod_{m=1}^{s}(n+m\beta)]}. \qquad (7)$$

The throughput of the system is characterized by the probability that the request which got into the system will be served:

$$q = 1 - P_H. \qquad (8)$$

It is obvious that the throughput of the system with expectations, with the same λ and μ, will always be higher than the capacity of the system with failures: in the event of waiting, unplanned, not all requests that caught n channels occupied, but only some.

Bandwidth increases with an increase in average waiting time $m_{te}=1/v$. The direct use of the formula (4-7) is somewhat complicated by the fact that it includes infinite amounts. We introduce instead of densities λ and v "given" densities:

$$\lambda / \mu = \lambda m_{pr} = \alpha,$$
$$v/ \mu = v/m_{pr} = \beta. \qquad (9)$$

The parameters α and β mean respectively the average number of requests and the average number of care queues that are queued for the average service time of one request.

Obviously, when $\beta \to \infty$, the system with expectation should turn into Erlang's system with failures (the request instantly leaves the queue). Let us consider another extreme case: a pure system with expectation ($\beta \to 0$, $t \to \infty$, $\alpha < n$). In this system, requests do not go out of the queue at all, and therefore $P_н = 0$: each requests sooner or later waits for service [8]

$$p_0 = \left[\sum_{k=0}^{n} \frac{\alpha^k}{k!} + \frac{\alpha^{n+1}}{n!\,(n-\alpha)}\right]^{-1} \qquad (10)$$

Hence, using formulas (4) and (5), we find

$$p_k = \frac{\alpha^k/k!}{\sum_{k=0}^{n}[\alpha^k/k! + \alpha^{n+1}/n!\,(n-\alpha)]}, \quad (0 \le k \le n) \qquad (11)$$

and similar for $k = n + s$ $(s \ge 0)$

$$p_{n+s} = \frac{a^{n+s}/n!\,n^3}{\sum_{k=0}^{n} a^k/k! + a^{n+1}/n!\,(n-a)} \qquad (12)$$

The average number of queuing requests is determined from the formula (6) at $\beta \to 0$:

$$m_s = \frac{\dfrac{a^{n+1}}{nn!\,(1-a/n)^2}}{\sum_{k=0}^{n} \dfrac{a^k}{k!} + \dfrac{a^{n+1}}{n!\,(n-a)}} \qquad (13)$$

3. Cloud service systems modeling whith using of typical mathematical schemes

It is known that there are two basic principles for constructing simulating algorithms: "Δt" and "δz" [8]. In modeling algorithm s built on the principle of "δz", that is, in random-step algorithms, Q-scheme elements are re-evaluated during modeling only at moments of special states (appearance of requests from the *RD* source or changes in the status of the channel *C*). In this case, the duration of the step depends on both the characteristics of the system itself and the effects of the environment *E*. When the asynchronous method of constructing a modeling algorithm, the leading (synchronizing) element is not used, and the next step of modeling (viewing elements of the Q-scheme) can correspond to any state the whole set of *RD* elements, the *SD* or *C* drive. In this case, the elements of the Q-scheme are arranged sporadically - only those elements that can change their state (view with forecasting) may be checked.

Consider the use of Q-circuits for a formal description of the process of functioning of a system. A typical situation in the job of such systems is the appearance of requests (requirements) for service and completion of service at random moments of time, that is, the stochastic nature of the process of their functioning. In the general case, the moments of receipt of requests in the system from the external environment form an incoming stream, and the moments of service termination form the output stream.

Formalizing some real system with a Q-scheme, you need to build the structure of such a system. As elements of the structure of Q-schemes we will consider elements of three types: *RD* - sources; *SD* - drives; *C* - service channels.

When using the principle of "δz" constructing an asynchronous modeling algorithm it is expedient to consider the process of

changing the states of elements of the Q-scheme in the direction opposite to the direction of request motion in the system. This can be done by cyclically reviewing all the elements of the Q-scheme at each step of the modeling, and determining which requests transitions from one element to another can occur at this time of system time. This asynchronous cyclic modeling algorithm in terms of viewing states of elements of the Q-scheme is identical to the deterministic modeling algorithm. The only difference is that the countdown of the system time is carried out as follows:

$$t_n = \min\ (\min_{k,j} t_{k,j}; \min_{m} t_m), \qquad (14)$$

that is, the time of the next step is determined at least from the minimum completion time of the service started by all the channels of all phases of the Q-scheme and the minimum time of regular requests received from the source. System time (14) is the minimum time of release of a channel $K_{k,j}$ or the time until a new request with a *RD* is received.

4. Reliability's modeling of the cloud computing execution phase

4.1. Minimal subtasks spanning graph (MSSG)

During the execution phase, possible failures due to data resource missing, computing resource missing, software failure, database failure, hardware failures, and network failures. We apply the logic-probabilistic method of determining reliability of calculations. The simplest case of a sequential subtask structure can be represented as a minimal subtasks spanning graph (MSSG) of the available its elements (nodes and links), which ensures the success of the execution of the entire service. We calculate the lower bound of the probability of failure-free operation of all subtasks:

$$Pre(t) = \bigcap_{i=1}^{M} pi(t) \qquad (15)$$

Each MSSG contains exactly one set of data resources without duplication, for which data resources and pre-subtasks are provided that provide a certain input (input data). For each *i*-th element, the probability of a fail-free operation of the $p_i(t)$ is calculated according to the normal law:

$$pi(t) = \prod_{j=1}^{K} exp\{-\lambda i \cdot T_{pi}\},$$
$$\lambda = const. \qquad (16)$$

Consequently, searching for all MSSGs and defining the working hours of their elements is a preparatory step in identifying the reliability of the cloud service. To solve the problem of constructing and passing the graph, the algorithm is proposed that implements the search in depth and width in accordance with the structure [9,10].

4.2. Minimum overlap execution graph (MOEG)

If any set of subtasks *M* is successful, then the execution is reliable for the cloud service to perform the required set of subtasks, so the MBSG may be displayed as overlapping of the above MSSG sets [1,12]. *M* of MSSG graphs are obtained and combined for the creation of MOEGs. For each common element, when the graphs intersect together, record more working time as the final working time of this element in the MOEG.

With a list of *N* graphs of the MOEG and the corresponding completion time, the reliability of the cloud service may be determined at execution phase failures, as follows

$$Pe(t) = P\left(\bigcup_{i=1}^{N} P_{MOEGi}\right), \qquad (17)$$

this means that any MOEG from the total number of MOEG routes that will be achieved will make the performance of the cloud service successful at execution time. Let us mark the event E_j as the successful work of the *j*-th MOEG, and mark the event $\overline{E_j}$ – unexecuting of it. Using the Bayes theorem [11] under conditional probability, we can deduce (16) as the sum of conditional probabilities:

$$R_{execution} = P(\cup_{i=1}^{N} MOEG_i) = \sum_{j=1}^{N_i} P(E_j) \cdot P(\overline{E_1}, \overline{E_2}, \ldots, \overline{E_{j-1}} | E_j) \qquad (18)$$

The next step generates all possible combinations of identified critical elements that lead to the event $\overline{E_1}, \overline{E_2}, \ldots, \overline{E_{j-1}}|E_j$ by means of a binary search and calculates the probability of these combinations. Their summation is $P\left(\overline{E_1}, \overline{E_2}, \ldots, \overline{E_{j-1}}|E_j\right)$.

When calculating the probability of failure of the MOEG elements, the maximum time of the corresponding entries in the list for this MOEG should be used. Finally, if the cloud service needs to be successfully completed, the request phase and the execution phase must be reliable at the same time.

$$P_{service}(t) = P_{\text{re}}(t)P_{\text{e}}(t) \qquad (19)$$

where $P_{re}(t)$ can be obtained from (15), and $R_e(t)$ can be deduced from (17).

Protecting authorized user information

Most authorized user information solutions are still based on a traditional security concept and are mainly focused on system-oriented or VM (virtual machine) - oriented approaches.

In [12], the proposed solution uses the Chinese Remnant Theorem and the public key cryptosystem for the secure exchange of secure user data stored in a cloud computing environment among authorized users. The solution is based on modules, each of which provides a set of services, which are mainly located in the management of the data owner.

The first module is intended to encrypt data before it is outsourced to a cloud infrastructure. The second module creates opportunities for a more effective way of managing access control policies through a secret key that is intended for sharing and access to data check. The third module is used to provide secure search for encrypted data through the generation of encrypted keywords. The results of all the previous modules are used to create the resulting container.

Using the container is the result of a solution finding that increases security and confidentiality and meets the cloud computing environment. Using the container, as part of the proposed solution, is able to preserve the privacy, integrity and authenticity of cloud-based data, even from the cloud-based provider with minimal impact on the functionality of encrypted data that matches the capabilities of search and distribution for authorized users. Eventually, it is important Remember that the security features offered in the container depend on the cryptographic algorithms used in this solution and are under the control of the data owner.

Conclusion

Some problems of modeling and estimating reliability in the cloud services system are considered in the article, namely, the cloud computing model for providing convenient network access upon request to the common pool of computing resources that are configured. For a typical structure, cloud service failures are analyzed in the main stages of its life cycle. The Markov model for waiting queue queues and calculation of bounce parameters are obtained by solving the Chapman-Kolmogorov equations on the example of a five-channel system with unlimited time.

The object-oriented approach and data security solutions in the cloud are proposed, which, unlike traditional approaches, guarantees the confidentiality and security of information, even when compromising the provider of cloud services or its infrastructure.

References

1. Dai, Y.S., Yang, B., Dongarra, J., Zhang G. (2018). *Cloud Service Reliability: Modeling and Analysis.* Available from: http://www.netlib.org/utk/people/JackDongarra/PAPERS/Cloud-Shaun-Jack.pdf.
2. Zhang, Y., Zhou Y. (2006). *Transparent computing: A new paradigm for pervasive computing.* Proceedings of the 3rd International Conference on Ubiquitous Intelligence and Computing (UIC-06), LNCS 4145, 1–11, 2006.
3. Iankoulova, M. Daneva (2012). *Cloud Computing Security Requirements: a Systematic Review.* Sixth International Conference on Research Challenges in Information Science.
4. Xie, M., Dai, Y.S., Poh, K.L. (2004). *Computing Systems Reliability: Models and Analysis.* Springer: New York, 2004. 330 p.
5. Xing, L., Dai, Y.S. (2008). *A new decision diagram model for efficient analysis on multi-state systems.* IEEE Transactions on Dependable and Secure Computing, Accepted for Publication, 2008, Publishers: IEEE Press.

6. Abramson, D., Buyya, R., Giddy, J. (2002) *A computational economy for grid computing and its implementation in the Nimrod-G resource broker*. Future Generation Computer Systems, 18(8), pp. 1061-1074.
7. Dai, Y.S., Pan, Y., Zou, X.K. (2007). *A hierarchical modeling and analysis for grid service reliability.* IEEE Transactions on Computers, 56(5), pp. 681-691.
8. Venttsel, E.S., Ovcharov, L.A. (2000) *Teoriya sluchaynyih protsessov i ee inzhenernyie prilozheniya*. M.: Vyisshaya shkola, 383 s.
9. Sovetov, B.Ya., Yakovlev, S.A. (2001) *Modelirovanie sistem*. M.: Vyisshaya shkola, 343 s.
10. Volodarskyi, Ye.T., Kukharchuk, V.V., Podzharenko, V.O., Serdiuk, H.B. (2001) *Metrolohichne zabezpechennia vymiriuvan i kontroliu.* Vinnytsia, Veles, 211 s.
11. Pysarenko, A., Tyshenko, D. (2017) *Model pidsystemy diahnostyky transportnoho zakhodu na osnovi Beiiesovskoi merezhi.* Summer Infocom'2017. Materialy IV MNPK z informatsiinykh system ta tekhnolohii, m. Kyiv, 1-2 chervnia 2017 r. K.: Inzhynirynh, s. 62-65.
12. Pyrozhkov, O.Iu., Savchuk, O.V. (2018) *Informatsiino-oriientovana kontseptsiia zabezpechennia bezpeky khmarnykh obchyslen.* Infokomunikatsiini systemy ta tekhnolohii, vyp. №2(2), s. 32-36.

Література

1. Dai, Y.S., Yang, B., Dongarra, J., Zhang, G. (2018) *Cloud Service Reliability: Modeling and Analysis*. Отримано з: http://www.netlib.org/utk/people/JackDongarra/PAPERS/Cloud-Shaun-Jack.pdf.
2. Zhang, Y., Zhou, Y. (2006). *Transparent computing: A new paradigma for pervasive computing.* Proceedings of the 3rd International Conference on Ubiquitous Intelligence and Computing (UIC-06), LNCS 4145, 1–11, 2006.
3. Iankoulova, M. Daneva (2012): *Cloud Computing Security Requirements: a Systematic Review*. Sixth International Conference on Research Challenges in Information Science.
4. Xie, M., Dai, Y.S., Poh K.L. (2004) *Computing Systems Reliability: Models and Analysis.* Springer: New York, 2004. 330 p.
5. Xing, L., Dai, Y.S. (2008). *A new decision diagram model for efficient analysis on multi-state systems.* IEEE Transactions on Dependable and Secure Computing, Accepted for Publication, 2008, Publishers: IEEE Press.
6. Abramson, D., Buyya, R., Giddy, J. (2002) *A computational economy for grid computing and its implementation in the Nimrod-G resource broker*. Future Generation Computer Systems, 18(8), pp. 1061-1074.
7. Dai, Y.S., Pan, Y., Zou, X.K. (2007) *A hierarchical modeling and analysis for grid service reliability*. IEEE Transactions on Computers, 56(5), pp. 681-691.
8. Вентцель, Е.С., Овчаров, Л.А. (2000). *Теория случайных процессов и ее инженерные приложения*. М.: Высшая школа, 383 с.
9. Советов, Б.Я., Яковлев, С.А. (2001) *Моделирование систем*. М.: Высшая школа, 343 с.
10. Володарський, Є.Т., Кухарчук, В.В., Поджаренко, В.О., Сердюк, Г.Б. (2001) *Метрологічне забезпечення вимірювань і контролю*. Вінниця, Велес, 211 с.
11. Писаренко, А., Тишенко, Д. (2017) *Модель підсистеми діагностики транспортного заходу на основі Бейєсовської мережі*. Summer Infocom'2017. Матеріали IV МНПК з інформаційних систем та технологій, м. Київ, 1-2 червня 2017 р. К.: Інжиніринг, с. 62-65.
12. Пирожков, О.Ю., Савчук, О.В. (2018) *Інформаційно-орієнтована концепція забезпечення безпеки хмарних обчислень*. Інфокомунікаційні системи та технології, вип. №2(2), с. 32-36.

РЕЗЮМЕ

С.Ф. Теленик, О.В. Савчук,
Є.О. Покровський,
О.М. Моргаль, О.А. Похиленко

Про моделювання надійності та оцінювання в системі хмарних сервісів

Досліджена модель надійності хмарних обчислень з метою забезпечення зручного мережевого доступу по запиту до загального пулу обчислювальних ресурсів з мінімальними зусиллями управління з боку постачальника послуг.

Розглянута архітектура типової системи хмарних сервісів із системою керування хмарами (СКХ), що складається з набору серверів. Мережа хмар може складатися з багатьох віртуальних ланок. Всі обчислювальні ресурси працюють разом через мережу, щоб забезпечити доступ до необхідних даних з ресурсів даних для розв'язання підзавдання. Результати завдання, що завершено та захищено, повертаються користувачеві.

Пропонована модель надійності хмар є сервіс-орієнтованою та ієрархічною, включає дві фази обслуговування запитів: 1) попередню (тайм-аут та переповнення); 2) виконання. Завершальним етапом роботи є захист запитів користувача та результатів виконання завдань. Визначені вірогідності помилок тайм-аута та переповнення. Показник відкидання запитів через переповнення виводиться шляхом вирішення ресурсами рівнянь Чепмена-Колмогорова.

Функціонування системи масового обслуговування описується процесом типу «загибель та розмноження». Знаючи ймовірності відмов всіх елементів системи, можна легко визначити ймовірність P_H того, що заявка покине систему без обслуговування. Пропускна здатність системи характеризується ймовірністю того, що заявка, що потрапила в систему, буде обслугована. Розглянуті умови чистої системи очікування, коли заявки взагалі не йдуть з черги.

Для формального опису процесу стохастичного функціонування хмарної системи використана Q-схема. Застосований асинхронний циклічний алгоритм з випадковим кроком та спорадичним переглядом елементів кожного підзавдання. Довжина чергового кроку визначається з мінімальних часів закінчення початого обслуговування всіма каналами всіх фаз Q-схеми та мінімального часу надходження чергових заявок з джерела.

На етапі виконання застосований логіко-ймовірнісний метод визначення безпомилковості обчислень підзавдань. Найпростіший випадок послідовної структури підзавдання представлений у вигляді мінімального зв’язуючого графа доступних і-их елементів (вузлів і зв’язків), що гарантує успіх виконання всього підзавдання. Наведено формулу розрахунку нижньої границі ймовірності безпомилкового обчислювання всього підзавдання.

На фазі виконання обчислюється мінімальний граф перекриття. Для успішного завершення сервісного обслуговування обидві фази запиту й виконання повинні бути надійними одночасно. Для завершального етапу пропонується підхід до забезпечення безпеки даних в хмарі, що на відміну від традиційних підходів гарантує конфіденційність та безпеку інформації навіть при компрометації провайдера хмарних послуг або його інфраструктури.

Надійшла до редакції 12.10.2018