Original Research Article: full paper

(2016), «EUREKA: Physics and Engineering»
Number 3

INFORMATION MODEL OF CLOUD APP SCALING WITH VARIABLE LOAD PEAKS

Tamara Savchuk
Department of computer science
Vinnytsia national technical university
95 Khmelnytske road, Vinnytsia, Ukraine, 21050
savchtam@gmail.com

Andrii Kozachuk
Department of computer science
Vinnytsia national technical university
95 Khmelnytske road, Vinnytsia, Ukraine, 21050
kozachuk35@rambler.ru

Abstract

The information model of cloud app was done. It is a formal description of cloud app infrastructure and possible transitions between them, and cloud app current working state classification criterion. Cloud app current state classification criterion on the basis of Page-Hinckley method and calendar of events related to the cloud app working state considers the current state to one of three classes in order to improve the accuracy of prediction of cloud app workload.

Proposed criterion was compared with standard offline criterion that analyzes information about the entire time series of cloud app through a considerable time after the events that lead to the load peak, and therefore can't be used when grading in real time. It is shown that the classification of cloud app state is consistent in 92 % of cases.

The resulting information model of cloud app scaling with variable load peaks can be used as a component of information technology for cloud app scaling with variable load peaks.

Keywords: cloud computin, cloud app information model, cloud app state classification.

DOI: 10.21303/2461-4262.2016.00079

© Tamara Savchuk, Andrii Kozachuk

1. Introduction

Developing information technology related to cloud app scaling it should be able to formally submit information about cloud app, its states and possible transitions between them. This can be achieved by introducing information model for cloud app scaling that combine classification criterion of cloud app working state and formal description of its condition.

Objectives of the study:

1. Form a classification criterion for cloud app working state based on historical information about its workload.

2. Develop a cloud app information model that allows predicting its condition.

2. Overview of the problem

Modern cloud app information models largely represent as vector of the quantities characterizing the current state of cloud app infrastructure and its workload [1]. However, little attention is paid to scaling operations, which in cloud app with variable load peaks are used with high intensity. Ability of Platform-as-a-service (PaaS) to carry out both horizontal and vertical scaling also doesn't take into account [2].

An important part of the information model for cloud app scaling is a description of classification of its working state that is taken into account when choosing the appropriate models for prediction of a cloud app state. The problem of classification of cloud app working state can be divided into two sub-tasks: determining the presence of load peak and direction of the trend.

The task of determining the presence of load peak can be solved by using the following methods:

– search of peaks of time series;

– detecting events.

Methods of detecting events in time series can be classified by data structure to which these methods are used [3]:

– for time series of scalar data;

– for time series of vector data;

– for time series of spatially distributed data.

Data on the number of network requests to the cloud app can be submitted in the form of a series of scalar values, and as a series of vectors, where each vector includes three values – the number of network requests, the time after the last load peak in the calendar of events and the time until the next events in the calendar of events.

Among the methods used to vector data are the following:

– classification using Bayesian network-based trust;

– classification using neural networks [4];

– classification using decision trees [5, 6];

– clustering methods.

The peculiarity of the application of these methods is the need for the training sample, which is a manually marked event of time series, and "black box effect" when not always possible to clearly describe the correlation between peculiarities of internal structure of data mining and test time series. The need to generate the training sample of network requests by other methods of classification complicates the use of vector methods for solving the problem of cloud app working state classification. Such system as Stucco [7] and WSARE [8] also can be used for determining events in the time series of the vectors.

Stucco is an intelligent system for event data search among heterogeneous sources. This system has the advantage of subject areas with poorly structured base data. Base data on cloud app have a clear structure, and therefore use of Stucco to analyze data on the cloud app is impractical. WSARE is a search system for anomalies in multivariate data, the system can detect the set of attributes that in the short term are custom value, creating thus an anomaly. Given that the number of data attributes that describe the cloud app work is limited and what to monitor anomalies only one attribute – the number of network requests, use of WSARE is not appropriate to classify the cloud app working condition.

Let's consider methods of regression analysis, reporting the occurrence of load peak when the time series exceeds a certain threshold [9]. In the simplest case, the threshold is defined as the average of the time series. It is also possible to determine the threshold as moving average or using exponential smoothing [3].

Page-Hinckley method is nonparametric method of detecting events [10] and can solve the problem of early disorder detection of time series [11]. Given the short duration of cloud app load peaks, detection rate of events is a key that determines the feasibility of its use in detection of variable loads in cloud app.

Wavelet blow up method is the method for searching peaks in time series. It is based on shifted wavelet trees [12]. Also, for searching peaks in time series, it is possible to use techniques that are considered the point peaks of smoothed time series, exceeding several times its average. They can also be used for search of local maxima of time series, but does not provide the definition of the beginning and end of the event, which does not solve the problem of identifying the load peaks in the time series of network requests to the cloud app only through methods for searching peaks.

3. Materials and Methods

Information model of cloud app can be represented as a set of information about system resources used by a cloud app, requests, cloud app working state classification criterion, which results are used by information technology, which makes cloud app scaling and transfers them from one possible condition to another (**Fig. 1**).

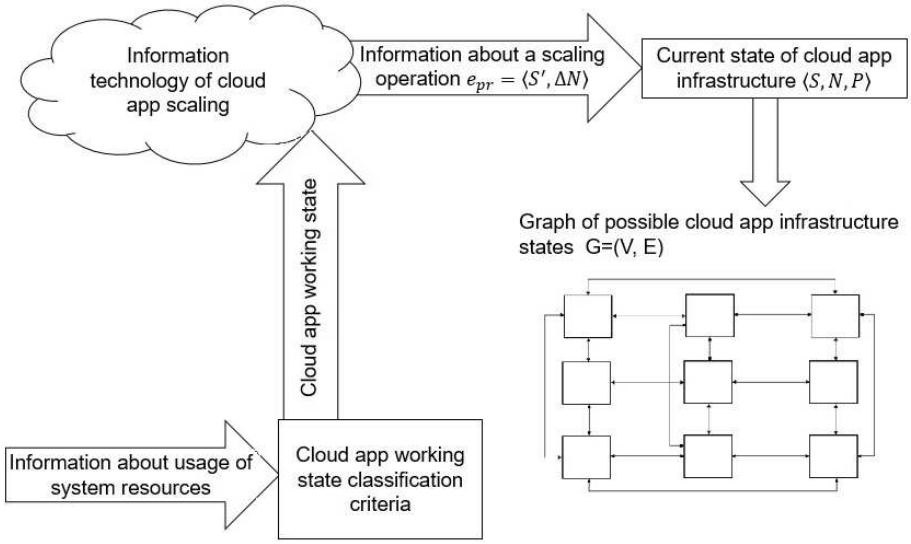

Fig. 1. Scheme of information model of cloud app scaling

3. 1. Possible states of cloud app infrastructure

Let's consider the possible ways of formal description of cloud app infrastructure state. Given the fact that cloud app operates in a «platform as a service» mode [13], the current state of infrastructure can be described by the following characteristics:

– number of the allocated CPU cores;

– amount of available RAM;

– capacity of input and output data channels;

– cost of infrastructure maintenance.

Processor cores and memory at the level of the platform are allocated in values that are divisible by the size of a virtual machine [14], which reduces the number of possible states of cloud app infrastructure. Given this limitation, a description of cloud app state includes the following features:

– number of allocated virtual machines;

– size of the virtual machine;

– cost of infrastructure maintenance.

Transition between the cloud app infrastructure states is possible via the scaling. There are two types of scaling: vertical (changing the type of virtual machine) and horizontal (changing the number of virtual machines) [2]. There are also mixed options for scaling, providing simultaneous execution of vertical and horizontal scaling. Scaling operations are divided into: increase the number of dedicated computing resources (up scale) and reduce the number of allocated computing resources (down scale).

Let's mark each possible state of cloud app infrastructure as $\langle S, P, N \rangle$, where S – the size of the virtual machine, N – number of allocated virtual machines, P – cost of maintenance of virtual machines for 1 hour, and build a graph of transitions using scaling between possible states of cloud app infrastructure. Graph nodes are $\langle S, P, N \rangle \in V$ describing computing resources, where V – the set of graph nodes, edges of graph are transitions $\left(\langle S_i, P_i, N_i \rangle, \langle S_j, P_j, N_j \rangle\right) \in E$ between possible states of infrastructure, where E – the set of edges of the graph. In general, the graph of cloud app infrastructure state is direct, but in practice the possibility of transition from state i to state j is accompanied by the possibility of transition from state j to state i, it is therefore appropriate to consider the graph as undirected.

The set of graph nodes and edges between them is determined by the capabilities of hosting platform [15]. In general, the number of virtual machines is unlimited and transition between any two states of the graph is possible. So, graph of cloud app infrastructure state is infinite and complete. However, in practice there is certain limitation, so graph is finite and not complete, so it is

possible to use all classical approaches to working with finite graphs. Infrastructure states graph of cloud app that in Microsoft Azure hosting [16] in the base tariff plan is shown in **Fig. 2**, prices are per month of usage, billing is hourly.

As shown in **Fig. 2**, features of some nodes such B1x2 and B2x1 are coincide. App work in the B2x1 state is more efficient due to lack of need for load balancing mechanism between virtual machines. Using the B1x2 state may be appropriate in the presence of the potential need to switch in B1x3 or B1x1 states because this operation will be carried out faster than B2x1 state.

When performing cloud app scaling, which is characterized by load peaks, it is important to pay attention to duration of the scaling. This information can be stored in the graph curves of cloud app infrastructure state. This graph is converted from an undirected to direct in order to display time difference for forward and reverse scaling with better semantics. Given the scaling duration in the graph curves, the transition from the state i in the state j at time t_{ij} will be indicated as follows: $\left(\langle S_i, P_i, N_i \rangle, \langle S_j, P_j, N_j \rangle, t_{ij}\right)$.

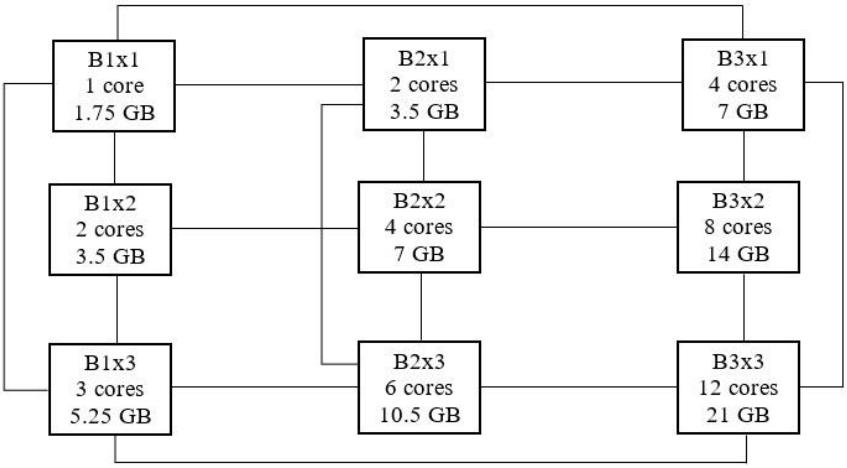

Fig. 2. Graph of the infrastructure states using Microsoft Azure

3. 2. Cloud app working state classification criterion

When performing cloud app scaling, its current state should be taken into account, which determines the need to develop criterion for the classification of its working state, it will include it to one of three classes:

1. Lack of load peaks and trend.
2. The presence of load peak, growing trend.
3. The presence of load peak, descending trend.

Page-Hinckley method is used for determine the presence of load peak. It doesn't need the training sample set and focused on the rapid detection of events that is especially important due to the rapid increase in the number of network requests during load peaks associated with cloud apps. This method involves the calculation of aggregated values m_T:

$$m_T = \sum_{t=t_0}^{T} \left(x_t - \bar{x}_T - \partial\right), \quad (1)$$

where $\bar{x}_T$ – the average value of time series at time T, (t0=1) – first element index of time series, δ – magnitude [17].

Using the method involves the calculation of the minimum value of all values of $M_T = \min\left(m_{t_0}, ..., m_T\right)$ among all m_T values for the entire analyzed time interval. The method assumes that the load peak occurs under the condition (2).

$$(m_T - M_T > \pi), \quad (2)$$

where π – a threshold that is usually is set proportionally to mean square deviation (δ) of time series:

$$\pi = 4\sigma/\delta. \quad (3)$$

Re-initialization of the calculation benchmark is taken place in the case of load peak: $t_0 = T$.

Load peaks in the time series of network requests to the cloud app are characterized by a finite length, so that the number of network requests per unit of time after the event returns to its original value. [18] This makes it possible to determine the duration of the load peak. The end of the load peak occurs at the next (4).

$$\overline{x}_t \le \overline{x}_{(t_0;t_s)}, \quad (4)$$

where t_s – step time corresponding to the start of the load peak, $\overline{x}_t \le \overline{x}_{(t_0;t_s)}$ – the average value of time series on the interval $[t_0; t_s]$. Due to the described features in load peak of time series of network requests to the cloud app, re-initialization of the calculation benchmark should be performed at the end of load peak: $t_0=t_e$, where t_e – end time of the previous load peak.

For cloud app working state classification it is necessary, in addition to identifying load peaks, to distinguish two states of working conditions during load peaks – with growing and descending number of network requests. Determining the direction of the trend time series is possible using the method of comparing local maxima [19], which is the comparison of the values of the last LocMax (X, 0) and the second to last LocMax (X, 1) local maximum of time series X. If the last local maximum is greater than second to last, the value of the time series of network requests are growing, otherwise – descending.

Comparison to the third and subsequent local maxima, which increases the accuracy of trend identification for time series, can be used, but it increases the period of inertia accounting the required number of items after turning the trend to identify this turn. Given that the duration of the load peak in cloud apps is short, detection efficiency of trend reversal is a priority and therefore we used only comparing the last two local maxima.

To determine the local maximum recursive algorithm LocMax (X, K) is used, where X – the time series of network requests, K – number of local maximum.

Application of Page-Hinckley method doesn't allow using advantages of the availability of calendar of events that can lead to load peas. Calendar of events can be represented by a plurality of pairs (5).

$$C = \{c_1, c_2, ...c_n\}, \quad (5)$$

where $c_i=(t_i, d_i)$ – event represented by the start time t_i and duration d_i. Occurrence of events via calendar can be identified with the condition ε_i

$$\varepsilon_i = (T \ge t_i) \wedge (T < t_i + d_i), \quad (6)$$

$$(t_i, d_i) = c, \exists c \in C. \quad (7)$$

Classification on the basis only calendar data can lead to underestimation of errors and missing load peaks in the calendar. Therefore, it is appropriate to take into account in the classification criterion as the value of time series and calendar events. Thus, cloud app working state classification criterion can be defined according to the ratios (8)–(13).

$$m_{T,c0} = \sum_{t=t_0}^{T} (x_t - \overline{x}_T - \partial_{c0}), \quad (8)$$

$$m_{T,c1} = \sum_{t=t_0}^{T} \left(x_t - \overline{x}_T - \partial_{c1} \right), \tag{9}$$

$$m_{T,c0} - M_{T,c0} > \pi_{c0} \mid \neg\varepsilon, \tag{10}$$

$$m_{T,c1} - M_{T,c1} > \pi_{c1} \mid \varepsilon, \tag{11}$$

$$\partial_{c0} > \partial_{c1}, \tag{12}$$

$$\varepsilon = \varepsilon_1 \vee \varepsilon_2 \vee \varepsilon_3 ..., \tag{13}$$

where $m_{T,c0}$, $M_{T,c0}$, π_{c0} – the current aggregate value of the time series, the minimum aggregate value and the threshold for any point T in the time series, which doesn't in the calendar of events; $m_{T,c1}$, $M_{T,c1}$, π_{c1}, – the current aggregate value of the time series, the minimum aggregate value and the threshold for any point T in the time series, which is in the calendar of events; ∂_{c0} and ∂_{c1} – the magnitude of the above two states.

Thus, criterion for identification of cloud app current working state was formed. It determines current cloud app state as one of three classes, based on the advanced Page-Hinckley method that takes into account the calendar of events related to cloud app load peaks.

4. Experimental procedures

For classification it was used time series of network requests to the site of the Football World Cup [18]. It was obtained by processing log of the site traffic, which is a set of records of network requests that come to the site of the championship. Data set of network requests to the site are aggregated in time series of network requests from sampling in 1 minute (**Fig. 3**). The resulting time series, comprising 28,000 members, was exported to a separate table of Microsoft SQL Server relational database.

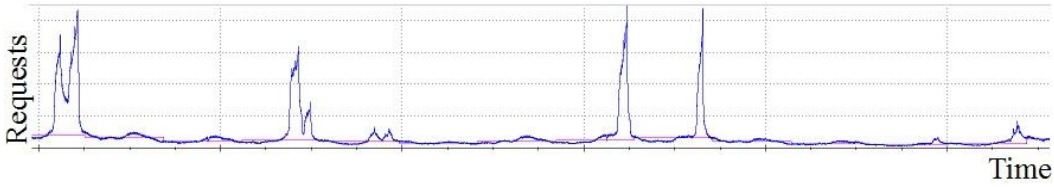

Fig. 3. Aggregate time series of network requests

Using SQL-script, which takes into account the nature of load peaks and the time of the events leading to the load peaks (football matches) and have access to information about the entire time series of network requests, cloud app working state classification was done with high reliability. The script uses an algorithm to search local maximum of time series to determine the transition point between the growing and descending parts during the load peak. Cloud app working state classification criterion was applied to each item of time series.

5. Results

Comparison of results of cloud app working state classification was done using the proposed criterion and standard offline criterion, using information about the entire time series and such features of load peaks as length of football matches.

As a result, it was highlighted 10 areas where there are load peaks with duration of 850 minutes. It was shown that the classification using the proposed criterion and offline criterion match in 92 % of cases. The presence of load peak is correctly determined in 96 % of cases. An example of the results of cloud app working state classification is shown in **Fig. 4**.

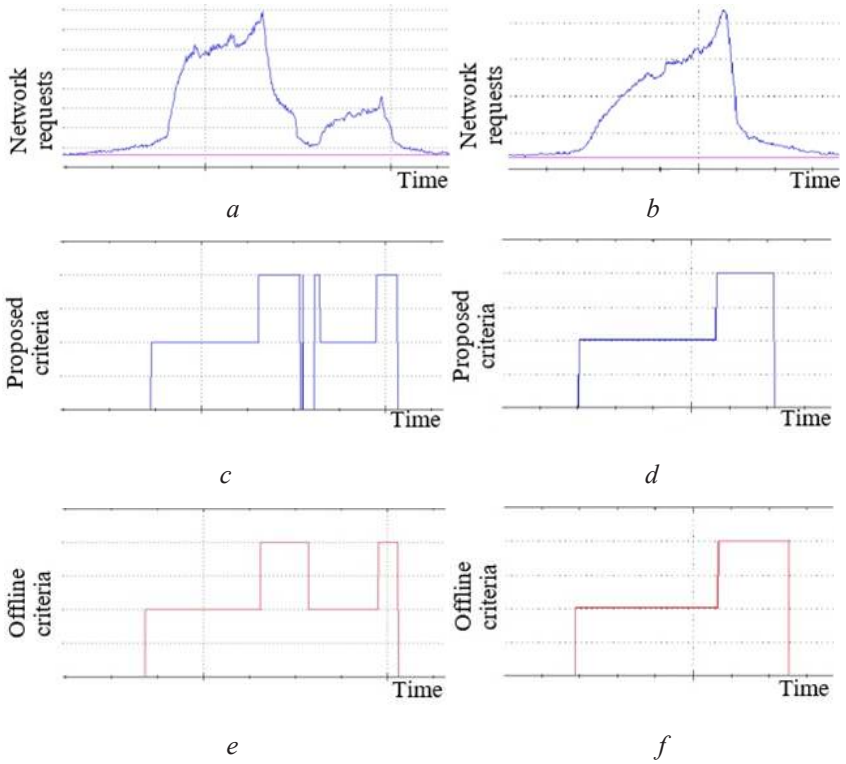

Fig. 4. The results of cloud app working state classification: *a, b* – fragments of time series of network requests with cloud app load peaks; *c, d* – cloud app working state classification using the proposed criterion; *e, f* – cloud app working state classification using the standard offline criterion

6. Discussion

The obtained information model of cloud app scaling with variable load peaks was used as a component of information technology [20], which is able to perform cloud app scaling with variable load peaks.

Use of the cloud app working state classification criterion as a part of the information model of cloud app scaling allows to reasonably select a specific prediction models [21] and scaling strategies for each cloud app working state.

7. Conclusions

The information model of cloud app was made. It is a formal description of cloud app infrastructure states and possible transitions between them and the cloud app working state classification criterion.

Cloud app current working state classification criterion with variable load peaks was formed. It is carried current state to one of three classes, based on the Page-Hinckley method advanced into account the calendar of events related to the work of the cloud app, in order to improve the accuracy of prediction of cloud app workload.

References

[1] Marco, A. S. Netto, Carlos, Cardonha, Renato, L. F. Cunha, Marcos, D. Assuncao (2014). Evaluating Auto-scaling Strategies for Cloud Computing Environments. IEEE 22nd International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems, 187–196. doi: 10.1109/mascots.2014.32

[2] Garcia-Gomez, S., Escriche-Vicente, M., Arozarena-Llopis, P., Lelli, F., Taher, Y., Momm, C. et. al. (2012). 4CaaSt: Comprehensive Management of Cloud Services through a PaaS. 2012 IEEE 10th International Symposium on Parallel and Distributed Processing with Applications. doi: 10.1109/ispa.2012.72

[3] Neill, D. B., Wong, W.-K. (2009). Tutorial on Event Detection. Available at: http://www.cs.cmu.edu/~neill/papers/eventdetection.pdf

[4] Specht, D. F. (1991). A general regression neural network. IEEE Transactions on Neural Networks, 2 (6), 568–576. doi: 10.1109/72.97934

[5] MacLennan, J., Tang, Z., Crivat, B. (2011). Data mining with Microsoft SQL server 2008. John Wiley & Sons.

[6] Ananij, V. (2006). Levitin Glava 10. Ogranicheniya moshhi algoritmov: Derevya prinyatiya resheniya. Algoritmy: vvedenie v razrabotku i analiz=Introduction to The Design and Analysis of Aigorithms. Moscow:«Vilyams, 409–417.

[7] Sistema poshuku podij u riznoridnix danix Stucco. Available at: http://stucco.github.io

[8] Sistema viyavlennya anomalij u chasovix ryadax WSARE. Available at: http://www.autonlab.org/autonweb/16620.html

[9] Patil, G. P., Taillie, C. (2004). Upper level set scan statistic for detecting arbitrarily shaped hotspots. Environmental and Ecological statistics, 11 (2), 183–197. doi: 10.1023/b:eest.0000027208.48919.7e

[10] Lucenko, O. P., Bajbuz, O. G. (2012). Oglyad metodiv poshuku rozladnan i perspektivi їxnogo zastosuvannya u texnichnomu analizi birzhovix kotiruvan. Aktualni problemi avtomatizaciї ta informacijnix texnologij, 16, 84–96.

[11] Nikiforov, I. V. (1983). Posledovatelnoe obnaruzhenie izmeneniya svojstv vremennyx ryadov. Moscow: Nauka, 199.

[12] Palshikar, G. (2009). Simple algorithms for peak detection in time-series. In Proc. 1st Int. Conf. Advanced Data Analysis, Business Analytics and Intelligence.

[13] Buxmann, P., Hess, T., Lehmann, S. (2008). Software as a Service. Wirtschaftsinformatik, 50 (6), 500–503. doi: 10.1007/s11576-008-0095-0

[14] Virtual Machine and Cloud Service Sizes for Azure. Available at: https://msdn.microsoft.com/en-us/library/azure/dn197896.aspx

[15] Golyachuk, N. V., Golyachuk, S. E., Rixlyuk, V. S. (2014). Xmarni obchislennya: zavtrashnij den biznesu. Ekonomichni nauki. Ceriya: Oblik i finansi, 11 (1), 37–43.

[16] Squillace, R. (2015). How to Use the Autoscaling Application Block. Available at: https://azure.microsoft.com/en-us/documentation/articles/cloud-services-dotnet-autoscaling-application-block/

[17] Andrienko, G., Andrienko, N., Mladenov, M., Mock, M., Poelitz, C. (2010). Extracting events from spatial time series. In Information Visualisation (IV), 2010 14th International Conference, 48–53. doi: 10.1109/iv.2010.17

[18] 1998 World Cup Web Site Access Logs. Available at: http://ita.ee.lbl.gov/html/contrib/WorldCup.html.

[19] Trenkenshu, A. I. (2014). Programmnaya identifikaciya klyuchevyx figur i predskazanie tendencij grafikov birzhevyx kotirovok po ekstremalnym priznakam na osnove algoritmov sortirovki. Taganrog.

[20] Savchuk, T. O. (2015). Information technology of scaling cloud app with variable load peaks. Technology audit and production reserves, 5 (2 (25)), 4–11. doi: 10.15587/2312-8372.2015.51716

[21] Kozachuk, A. V., Savchuk, T. O. (2015). Prognozuvannya kilkosti merezhevix zapitiv do xmarnogo zastosunku. Visnyk Nacional'nogo universytetu "L'vivs'ka politehnika".