

DOI: <https://doi.org/10.15276/aait.04.2021.6>

UDC 004.75:004.8:519.25

# Reducing cloud infrastructure costs through task management

Oleg N. Galchonkov<sup>1)</sup>ORCID: <https://orcid.org/0000-0001-5468-7299>; o.n.galchenkov@gmail.com. Scopus Author ID: 56081377900Mykola I. Babych<sup>2)</sup>ORCID: <https://orcid.org/0000-0002-3946-9880>; babich.tiger@gmail.comAndrey V. Plachinda<sup>2)</sup>ORCID: <https://orcid.org/0000-0001-8804-6852>; andrey.plachinda67@gmail.comAnastasia R. Majorova<sup>1)</sup>ORCID: <https://orcid.org/0000-0003-3004-6993>; anastasiamayorova2003@gmail.com<sup>1)</sup> Odessa National Polytechnic University, 1, Shevchenko Ave. Odessa, 65044, Ukraine<sup>2)</sup> Digitally Inspired LTD, 2a, Genoese St. Odessa, 65000, Ukraine

## ABSTRACT

The transition of more and more companies from their own computing infrastructure to the clouds is due to a decrease in the cost of maintaining it, the broadest scalability, and the presence of a large number of tools for automating activities. Accordingly, cloud providers provide an increasing number of different computing resources and tools for working in the clouds. In turn, this gives rise to the problem of the rational choice of the types of cloud services in accordance with the peculiarities of the tasks to be solved. One of the most popular areas of effort for cloud consumers is to reduce rental costs. The main base of this direction is the use of spot resources. The article proposes a method for reducing the cost of renting computing resources in the cloud by dynamically managing the placement of computational tasks, which takes into account the possible underutilization of planned resources, the forecast of the appearance of spot resources and their cost. For each task, a state vector is generated that takes into account the duration of the task and the required deadline. Accordingly, for a suitable set of computing resources, an availability forecast vectors are formed at a given time interval, counting from the current moment in time. The technique proposes to calculate at each discrete moment of time the most rational option for placing the task on one of the resources and the delay in starting the task on it. The placement option and launch delays are determined by minimizing the rental cost function over the time interval using a genetic algorithm. One of the features of using spot resources is the auction mechanism for their provision by a cloud provider. This means that if there are more preferable rental prices from any consumer, then the provider can warn you about the disconnection of the resource and make this disconnection after the announced time. To minimize the consequences of such a shutdown, the technique involves preliminary preparation of tasks by dividing them into substages with the ability to quickly save the current results in memory and then restart from the point of stop. In addition, to increase the likelihood that the task will not be interrupted, a price forecast for the types of resources used is used and a slightly higher price is offered for the auction of the cloud provider, compared to the forecast. Using the example of using the Elastic Cloud Computing (EC2) environment of the cloud provider AWS, the effectiveness of the proposed method is shown.

**Keywords:** Cloud computing; spot resources; resource prediction; price prediction; dynamic task management

*For citation:* Galchonkov O.N., Babych M.I., Plachynda A.V., Mayorova A.R. Reducing cloud infrastructure costs through task management. *Applied Aspects of Information Technology*. 2021; Vol. 4 No.4: 366–376. DOI: <https://doi.org/10.15276/aait.04.2021.6>

## 1. INTRODUCTION

The transition to the use of cloud services allows you to get significant savings in finance, compared to using your own computing facilities. Therefore, more and more companies are increasingly using the services of cloud service providers. This, in turn, encourages cloud providers to expand the range and variety of their services. The most famous cloud providers are AWS, Azure and Google Cloud [1], which has their own data centers around the world. Competition for customers leads to the fact that these providers provide similar services at similar prices; however, due to historical reasons and the presence of accumulated experience, there are some

difference between them. The largest share of the cloud services market is taken by AWS due to the widest variety of services and tools for developers. Microsoft Azure has a long-standing relationship with a large number of industrial companies that prefer to meet their enterprise computing needs with a familiar vendor. Google differentiates itself from other cloud service providers with its advanced machine learning technology. However, for all providers, the flip side of a wide variety of possible architectural computing solutions and tools for working with them, given that customers have an equally wide range of tasks, is that the task of minimizing the cost of renting resources by providing matching resources and tasks. At the moment, there are no ready-made solutions that allow you to do this automatically.

© Galchonkov O., Babych M., Plachynda A.,

Mayorova A., 2021

This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/3.0>)

General advice is usually given [2]:

- formulate your requirements;
- choose the right computing resources for your needs;
- check data transfer and storage restrictions;
- check if your tasks are ready for execution on spot resources (idle computational resources of the provider put up for auction [3]);
- use mixed instances that include heterogeneous resources;
- rent resources simultaneously in multiple availability zones.

Automated tools such as CAST AI and Google Anthos make it somewhat easier to follow these tips [4]. They allow you to instantly specify the most suitable cloud provider and its specific services in response to your requirements. Moreover, the developers of these tools ensured that the current prices for services are taken into account, which providers are constantly changing depending on the market situation. However, these tools allow only static optimization when starting tasks for execution. The efficiency of your tasks, the dynamic appearance and disappearance of more advantageous computing resources is not provided.

At the same time, resources “on-demand” (on-demand instances) can be rented at any time and abandoned immediately, as soon as they are no longer needed, but these are the most expensive resources per unit of time [1]. Reserved Instances and Savings Plans are about 40 % cheaper than on-demand resources. However, you rent them for an extended period, during which your requirements may change significantly and it may turn out that the rented computing facilities will not be fully used or even be idle. Spot resources (Spot Instances) allow you to save up to 90 % of the cost of rent, compared to resources “on demand”. However, they can appear and be canceled very dynamically. The task performed on the spot resource must be prepared for the fact that after the operator's warning there will be a limited time to save intermediate results and complete the work. It should be added to this that the scope of tasks and the requirements for their fulfillment for clients of cloud providers are also usually not constant and vary significantly over time. Therefore, the development of automatic tools that allow dynamically in time to manage the execution of tasks on the most profitable cloud resources is relevant.

It should be noted that the services provided by cloud providers fall into the following categories [5]:

- software as a service (SaaS);
- platform as a service (PaaS);
- infrastructure as a service (IaaS);
- other.

The highest dynamics in time is inherent in IaaS, so it is here that you can get the greatest effect from the introduction of automatic tools for dynamic management of tasks and rented resources.

## 2. LITERATURE REVIEW AND STATEMENT OF THE PROBLEM

Both resource consumers and providers are interested in dynamically managing cloud resources. The efforts of providers are aimed at managing resources to reduce their electricity consumption and dynamically managing pricing policies in order to increase revenues while ensuring the high quality of services provided. In this regard, resource consumers should take into account in their algorithms not only the time-varying flow of requests for resources, but also fluctuations in the amount of free resources and their cost due to the activities of providers. Moreover, the target parameters of resource consumers may also differ:

- minimization of rent, subject to restrictions on the deadline for completing each of the tasks,
- minimization of rent and time for completing tasks, combined into a single quality function with its own coefficients.

In [6], the concept of “asymmetry” was introduced to measure the uneven load of servers. Eliminating the skew in server loading and using the resource use model made it possible to build a distribution system for computing with fewer servers in use. Universal algorithms for minimizing the power consumed by servers while simultaneously controlling the cost of renting computing resources are considered in [7, 9]. In [10], a swarm algorithm is proposed for optimizing resource allocation in order to save energy, taking into account not only the energy consumption of servers directly, but also of air conditioning equipment. The construction of a resource allocation system that takes into account the efficiency of air conditioners, based on various heuristic algorithms and machine learning methods, is considered in [11]. It should be noted that these works propose optimization of resource allocation based on the current situation. In [12], it is shown that a higher efficiency of a cloud provider can be achieved by building a model for the dynamic provision of computing resources in the time domain, compared with optimization at an isolated time point. In [13], a similar model in the time domain is built on the basis of the concept of guaranteeing fairness for users of cloud resources and the maximum allowable delay. It should be noted that such a construction of the model leads to some deterioration in the services provided, since the task execution may occur with a delay. This

conflicts with the target function of users who minimize not only rental costs, but also the time it takes to complete tasks. Although for users who only minimize their rental costs, this is perfectly acceptable.

Good results can be obtained by taking into account the specifics of tasks solved in the cloud. [14] describes the Cloud Assisted Mobile Edge (CAME) computing environment designed to serve dynamic mobile requests with different quality of service requirements. For such a model, optimal resource allocation (ORP) algorithms are proposed with different instances to optimize the computational power of the edge hosts while dynamically adjusting the cloud lease strategy. The modeling carried out in [14] for the Google cluster showed that the proposed ORP algorithms are superior in efficiency to the universal algorithms for organizing cloud computing in terms of the flexibility and profitability of the system. To plan the execution of scientific tasks in the cloud, characterized by indefinite deadlines and random arrival times, the NOSF structure is proposed in [15]. It includes task preprocessing, virtual machine allocation, and a feedback process. The simulation results given in [15] show that the proposed algorithm is significantly superior to the known algorithms in terms of reducing rental costs and the likelihood of timing violations for this type of problem.

Consumers of cloud services in their strategies to minimize the rental cost of resources should take into account algorithms that use cloud providers to maximize their revenue. Providers seek to maintain a dynamic balance between the use of inadequate and increasing workload of their resources. [16] proposed an algorithm management class of virtual machines based on using the model to maximize income over a time interval from a current time to an interval forward. The model involves the use of Markov processes to predict resource utilization, the effects of maneuvering by spot resources and decision-making for dynamic pricing. Experimental results presented in [16] confirm the high efficiency of such dynamic pricing for cloud providers. An alternative game approach was proposed in [17]. Load balancing is achieved through a strategy of migrating requests between servers in a distributed, non-cooperative, and competitive environment. To achieve and maintain an equilibrium solution in time, an iterative proximal algorithm (IPA) is proposed using the calculus of variations. The same authors propose in [18] based on the game approach, a joint strategy of the provider and user of cloud services. The provider uses a server provisioning

and request distribution strategy to reduce energy costs while meeting the needs of its users. And each user tries to maximize the utility function, taking into account the profit and efficiency of the time spent. However, this approach requires further research to compare strategies when the provider has information about the plans for service requests, and when not, since the possession of additional information allows the provider to charge higher prices for resource rent. Gaming-based cloud resource management strategies have shown positive effects in the provision of GPU-accelerated media processing services [19]. The pricing method proposed in this paper has the potential to generate higher margins for both the cloud service provider and users than the original GPU cloud services pricing strategy.

Another factor used by cloud providers is the clock speed of the rented processors. The lower the frequency, the lower the power consumption. In [20], it is proposed to use a nonlinear model of power dissipated by multicore processors in the pricing algorithm. Energy savings are estimated at over 14 percent.

Thus, an analysis of the literature shows that the conditions in the cloud resources market are dynamically changing; complex pricing algorithms are used, aimed at increasing providers' profits. Moreover, users can get the greatest rental cost savings when using spot resources. However, providers also pay maximum attention to operations with spot resources. In addition to the above factors, cloud providers use the auction mechanism [21, 22]. Moreover, the provision of a spot resource can be interrupted if some other consumer offers a higher price for the resource. In this case, Azure and Google Cloud warn the user 30 seconds before the resource is disconnected, AWS – 2 minutes.

In this regard, for the consumer of cloud resources, who sets as his goal to reduce rental costs, there are three tasks:

- forecasting the appearance of spot resources with the required characteristics in a time interval;
- predicting prices for these spot resources at the same time interval in order to put up for auction prices that, on the one hand, are most beneficial to the consumer, on the other hand, they allow using the rented spot resource without interruption with a high probability;
- using an effective strategy for leasing resources and launching tasks on them, taking into account forecasts of the appearance of resources and prices for them.

A number of works are devoted to forecasting prices for spot resources, which provide very

effective ready-made algorithms. For example, [21] proposed a regression random forest (RRF) model for predicting spot prices one week and one day ahead. In [23] a regression model of k-nearest neighbors (kNN) is proposed, which is adapted to predicting the price of spot resources. In [24], for these purposes, it is proposed to use neural networks, and in [25], machine learning models.

In [26], a multipurpose genetic algorithm was implemented for dynamic forecasting of the use of cloud resources. In [27], a forecast-based resource planning method is proposed. The forecasting model is trained on a dataset created by simultaneously deploying scientific application tasks in the cloud. The resources are then scheduled using a trained forecasting model.

As a strategy for leasing resources and launching tasks on them, heuristic algorithms are usually used. So in [28], a task scheduling algorithm is proposed, which perceives the available resources as a constraint, under which it is necessary to adjust the performance of tasks, up to the crowding out of resource-intensive tasks. For most practical tasks, this approach is not suitable, since all tasks must be completed on time. In [29], a mathematical model for scheduling two-level data processing was combined with a genetic algorithm, for which special mutation and crossover operations were designed. The main focus of this algorithm is on speeding up the learning of the genetic algorithm. However, in comparison with a large volume of tasks performed, the complexity of training a genetic algorithm is many orders of magnitude less. At the same time, this algorithm does not take into account the dynamic situation, taking into account the forecast of the appearance of spot cloud resources.

Thus, the analysis of literature data shows that for forecasting the prices of computing resources there are a large number of different algorithms that can be taken off-the-shelf. To predict the appearance of spot resources on time intervals, modifications of the same algorithms can be used. To create full-featured algorithms for minimizing rent by consumers, there are not enough effective algorithms for managing tasks that dynamically take into account forecasts for the appearance of various spot resources and forecasts of prices for renting these resources.

### 3. THE AIM OF THE STUDY

The purpose of this work is to develop a methodology for reducing the cost of cloud infrastructure through dynamic scheduling of tasks, taking into account, in addition to the characteristics of the tasks themselves, forecasts for the emergence

of spot resources and the rental price of these resources.

Minimization of rent is subject to deadline restrictions for each task.

### 4. DEVELOPMENT OF COST REDUCTION METHOD

Despite the fact that the proposed methodology is universal in nature and can be used when working with any cloud provider, for the specific presentation and verification of efficiency, further presentation will be carried out for AWS. Moreover, the minimum warning time for a spot resource disconnection for AWS is 2 minutes, as opposed to 30 seconds for Google and Azure. This is a significant AWS advantage when users prepare tasks for execution on spot resources.

AWS currently offers over 400 different computing resources on EC2 that differ in processor performance, RAM size, and more. To reduce the amount of calculations to reduce costs, it is advisable to choose a small number of L types of resources that will be leased. These types of resources should be selected based on the types of tasks that need to be addressed. We believe that the smaller the number of the resource type, the less its computational capabilities and the lower its rental cost, and also, if the problem can be efficiently solved on the  $i$ -th resource, then it cannot be done badly on resources with more high numbers.

As a rule, any company working with EC2 AWS has:

- Reserved Instances - for persistent loads;
- Scheduled Reserved Instances – for loads constantly occurring at well-defined times.

The company pays for these resources on a permanent basis, so they cannot be diminished dynamically. However, if, for some reason, the planned tasks do not fully load these resources, then the implementation of additional tasks on them is obtained, as it were, for free. Therefore, from the point of view of reducing the cost of renting resources, the underloading of these two resources is considered equivalent to the resource with the maximum priorities (01, 02, ..., 0L – the second digit denotes the type of resource).

The next in priority are spot resources with the minimum resource interruption warning time (on AWS it is 2 minutes), which are subdivided into sub-resources with types 1, ... , L. Respectively their priorities are 11, 12, ... , 1L. The cost of renting these resources is an order of magnitude lower than when the same types of resources are rented “on demand”.

We assign even lower priorities for spot resources with long interruption warning times: 211, 221, ..., 2L1, 212, ..., 2L2, ..., 2L6. Here the second digit denotes the type of resource, the third denotes the time of the warning about the interruption of the resource provision in hours.

We assign the lowest priority to resources “on demand”, since they have the highest rental cost: 31, ..., 3L, where the second digit denotes the type of resource.

In order for the execution of tasks to be dynamically controlled, for each task must be specified:

- the duration of execution in time intervals corresponding to the time interval of the warning about the interruption of the lease, for simplicity we will assume this time equal to 2 minutes, for the warning times 1 hour, 2 hours, etc. everything will be similar;

- the minimum type of computational resource that suits this task;

- the deadline for completing this task.

In accordance with these parameters, each task at startup is associated with a state vector shown in Fig. 1. Here  $n$  denotes the current moment in time,  $(k+1)$  – duration of the task,  $(n+m+1)$  – task completion deadline. The task must be prepared in such a way that it can be interrupted with the preservation of intermediate results after receiving a warning from AWS about the termination of the provision of this computational resource. In other words, the task must be divided into  $k + 1$  sequentially executed stages, each of which can be reloaded for execution on another resource [30, 31].

The leftmost unit in the state vector corresponds to the task stage that is the first to be executed. If in a discrete time  $n$  this stage was performed, then this unit is removed and the vector is shortened by one element. If this stage is not performed in a discrete time  $n$ , then all units are shifted to the right and the vector is shortened by one element from the left.

If at time  $n$  the prediction unit has issued a forecast for the appearance of a corresponding resource for this task, providing reduced rental costs, then all units are shifted to the right by time intervals when the execution of the corresponding stages on profitable resources is predicted. The cells on the left are filled with zeros, which means that the task is put on hold for the corresponding number of time intervals. When the forecast is canceled, the units return to their original places.

The forecast for the task execution on any computing resource is formed by analogy with the task state vector. An example of a forecast is shown in Fig. 2. An example of a forecast vector shows that for the problem under consideration there is a forecast of the appearance of a profitable resource and it is desirable to put the problem on hold for  $t$  time intervals.

If there are only units left in the task state vector and there are currently no free profitable resources, then the corresponding resource “on demand” is leased for the task.

When renting spot resources, an essential point is the level of the price offered by the lessee for this type of computing resource. To reduce the likelihood of interrupting the task, it is recommended to use the price forecasting module [21, 23], [26, 27] for renting a resource and offer a slightly increased price compared to the forecast. This will avoid interruptions and save on the cost of traffic for reloading the task to another computing resource.

As an objective function to be minimized, we will use the total cost of renting resources from the current moment of time to  $V$  forward discrete.  $V$  is chosen large enough so that all problem vectors end before this point in time,

$$F[n] = F_0[n] + F_1[n] + F_2[n] + F_3[n], \quad (1)$$

where:  $F[n]$  – total possible rental cost from the current time  $n$  to the time  $n + V$ ;  $F_0[n]$  – total possible

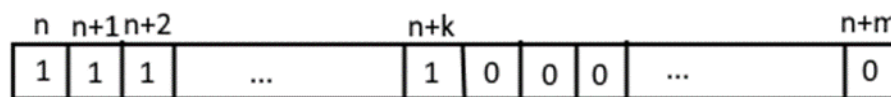


Fig. 1. Task state vector

Source: compiled by the authors

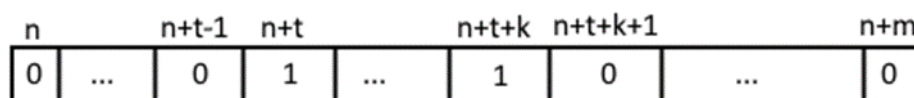


Fig. 2. Example of a forecast vector

Source: compiled by the authors

cost of additional loading of planned resources in the time interval from the current moment of time  $n$  to the moment of time  $n + V$ ;  $F_1[n]$  – total possible cost of renting spot resources with a possible interruption time of 2 minutes in the time interval from the current time  $n$  to the time  $n + V$ ;  $F_2[n]$  – total possible rental cost of spot resources with possible interruption time 1 hour, ..., 6 hours in the time interval from the current time  $n$  to the time  $n + V$ ;  $F_3[n]$  – the total possible cost of renting resources “on demand” in the time interval from the current time  $n$  to the time  $n + V$ .

Each of the components of the total rental cost is determined as follows:

$$F_0[n] = \sum_{i=1}^E \sum_{j=1}^L s_j^0[n] a_{ij}^0[n] t_{ij}^0[n], \quad (2)$$

where:  $s_j^0[n] = 0$  – cost of additional loading of the  $j$ -th type of planned computing resource;  $a_{ij}^0[n]$  – the coefficient of placement of the  $i$ -th task on the  $j$ -th planned computing resource;  $t_{ij}^0[n]$  – the planned delay in the launch of the  $i$ -th task at the  $j$ -th scheduled computing resource.

$$F_1[n] = \sum_{i=1}^E \sum_{j=1}^L s_j^1[n] a_{ij}^1[n] t_{ij}^1[n], \quad (3)$$

where:  $s_j^1[n]$  – the cost of renting the  $j$ -th type of spot computing resource with an interruption warning time of 2 minutes;  $a_{ij}^1[n]$  – the coefficient of placing the  $i$ -th task on the  $j$ -th spot computing resource with an interruption warning time of 2 minutes;  $t_{ij}^1[n]$  – the planned delay in starting the  $i$ -th task on the  $j$ -th spot computing resource with an interruption warning time of 2 minutes.

$$F_2[n] = \sum_{p=1}^6 \sum_{i=1}^E \sum_{j=1}^L s_{jp}^2[n] a_{ijp}^2[n] t_{ijp}^2[n], \quad (4)$$

where:  $s_{jp}^2[n]$  – the cost of renting the  $j$ -th type of spot computing resource with an interruption warning time  $p$  hours;  $a_{ijp}^2[n]$  – the coefficient of placing the  $i$ -th task on the  $j$ -th spot computing resource with an interruption warning time  $p$  hours;  $t_{ijp}^2[n]$  – planned delay in starting the  $i$ -th task on the  $j$ -th spot computing resource with an interruption warning time  $p$  hours.

$$F_3[n] = \sum_{i=1}^E \sum_{j=1}^L s_j^3[n] a_{ij}^3[n] t_{ij}^3[n], \quad (5)$$

where:  $s_j^3[n]$  – the cost of renting the  $j$ -th type of computing resource “on demand”;  $a_{ij}^3[n]$  – allocation coefficient of the  $i$ -th task on the  $j$ -th type of computing resource “on demand”;  $t_{ij}^3[n]$  – planned

delay in starting the  $i$ -th task on the  $j$ -th type of computing resource “on demand”.

Minimization of the total possible rental cost  $F[n]$  each current moment  $n$  is produced by choosing the corresponding values of the allocation coefficients  $a_{ij}^0[n]$ ,  $a_{ij}^1[n]$ ,  $a_{ijp}^2[n]$ ,  $a_{ij}^3[n]$  and planned delays in launching tasks  $t_{ij}^0[n]$ ,  $t_{ij}^1[n]$ ,  $t_{ijp}^2[n]$ ,  $t_{ij}^3[n]$  by genetic algorithm [32, 33] with the following restrictions:

- each  $i$ -th task can be launched or scheduled to run on only one computing resource, that is, for each  $i$  from the entire set of allocation factors  $a_{ij}^0[n]$ ,  $a_{ij}^1[n]$ ,  $a_{ijp}^2[n]$ ,  $a_{ij}^3[n]$  only one can be equal to 1, the rest must be equal to zero;

- every  $i$ -th task can be scheduled to run on some resource only if there are delays that lead to the coincidence of all units in the vector of the  $i$ -th task with units in the forecast vector of this resource, otherwise the corresponding allocation factor is zero;

- planned delays in launching tasks  $t_{ij}^0[n]$ ,  $t_{ij}^1[n]$ ,  $t_{ijp}^2[n]$ ,  $t_{ij}^3[n]$  should not cause the units in the task vectors to shift beyond the deadline for completing the corresponding tasks;

- if any task has already been launched on some resource, then it is not interrupted by the program, even if a more profitable option for its placement is predicted, interruption can only be carried out by a cloud provider within the framework of the auction mechanism.

The chromosome length for each calculation is determined by the number of tasks and the number of suitable resource types. Chromosome contains the placement coefficient  $a_{ij}^0[n]$ ,  $a_{ij}^1[n]$ ,  $a_{ijp}^2[n]$ ,  $a_{ij}^3[n]$  and planned delays in starting tasks  $t_{ij}^0[n]$ ,  $t_{ij}^1[n]$ ,  $t_{ijp}^2[n]$ ,  $t_{ij}^3[n]$ . During the experiments, the number of chromosomes that underwent mutations was 20. The coefficient of mutations was 0.35. The number of iterations of the genetic algorithm to minimize the total rental cost each time was constant and was equal to 20.

At each current moment of time  $n$ , the decisive block for renting resources issues allocation coefficients for execution  $a_{ij}^0[n]$ ,  $a_{ij}^1[n]$ ,  $a_{ijp}^2[n]$ ,  $a_{ij}^3[n]$  and planned delays in launching tasks  $t_{ij}^0[n]$ ,  $t_{ij}^1[n]$ ,  $t_{ijp}^2[n]$ ,  $t_{ij}^3[n]$ .

A generalized block diagram of a program that implements the technique is shown in Fig. 3.

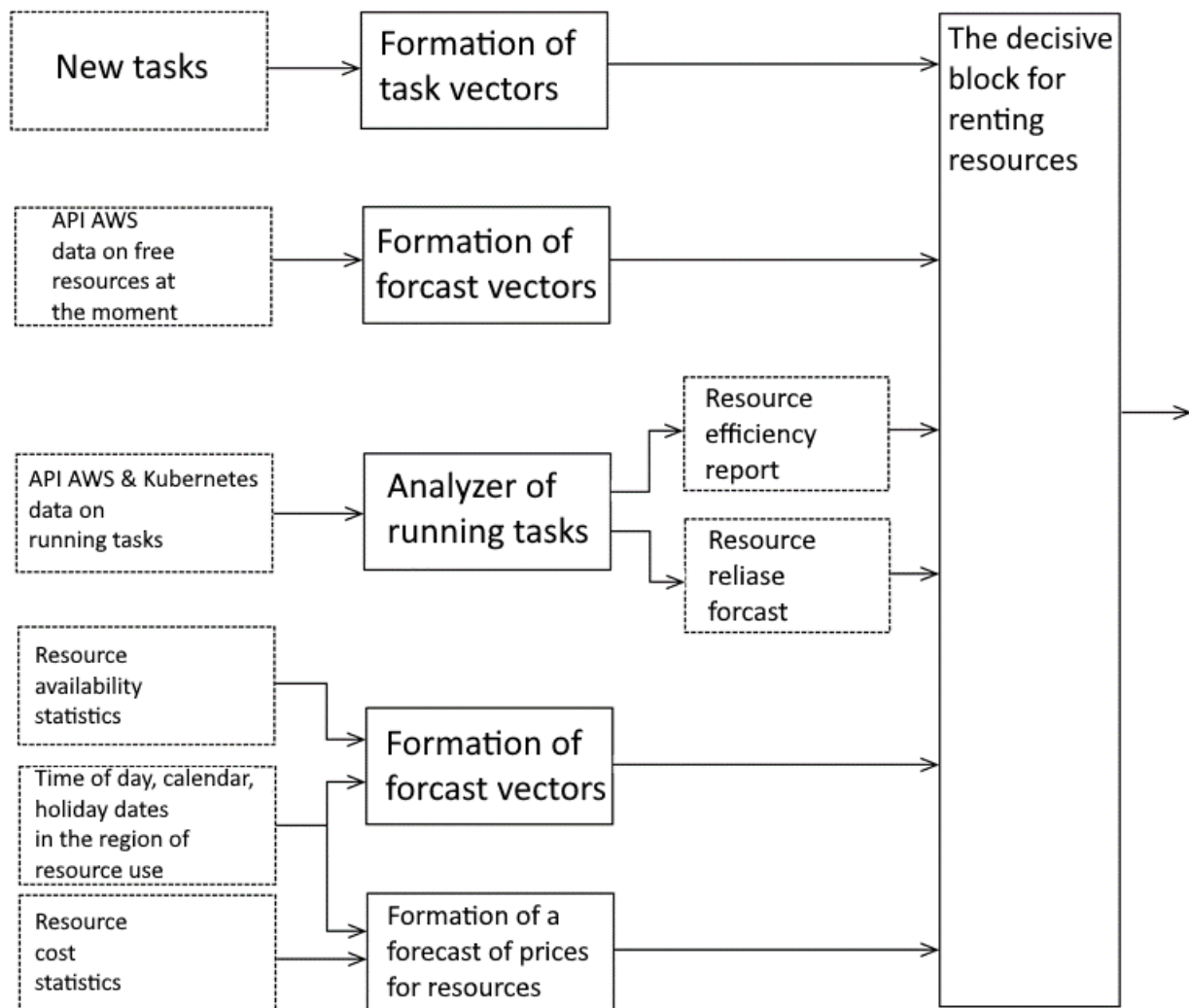


Fig. 3. Generalized block diagram of a program that implements the technique

Source: compiled by the authors

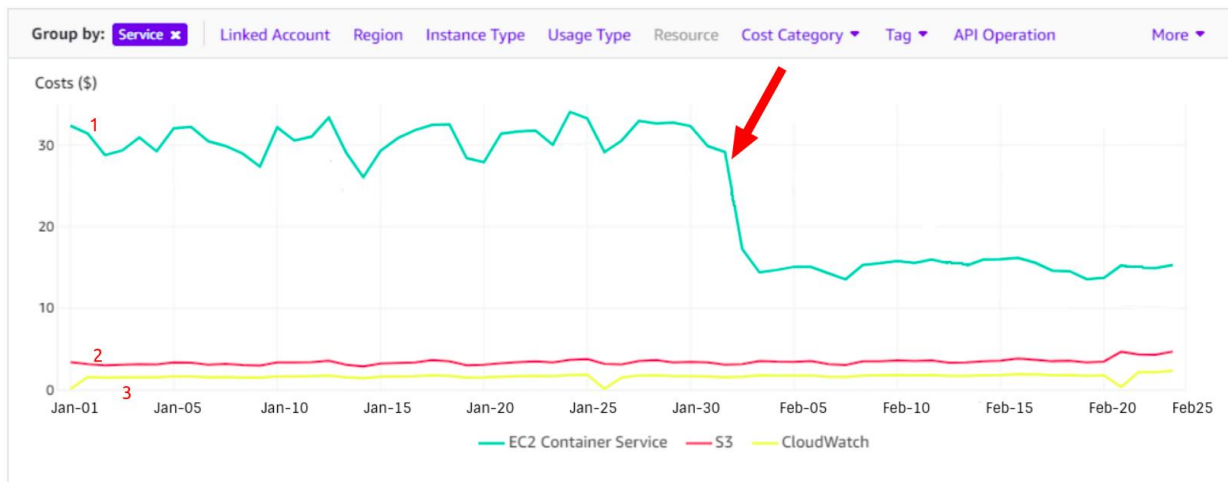
### 5. EXPERIMENTAL RESULTS OF USING THE PROPOSED METHOD

Fig. 4 is a screenshot of the lease payment schedules from January 1 to February 25, 2021, as captured using the AWS Cost Explorer service. The beginning of the use of the technique on January 31st is shown by an arrow. The figure shows that the use of the methodology allowed reducing the costs of EC2 by about half, while the work of the methodology with EC2 resources practically did not affect the cost of renting S3 storage and the CloudWatch monitoring service.

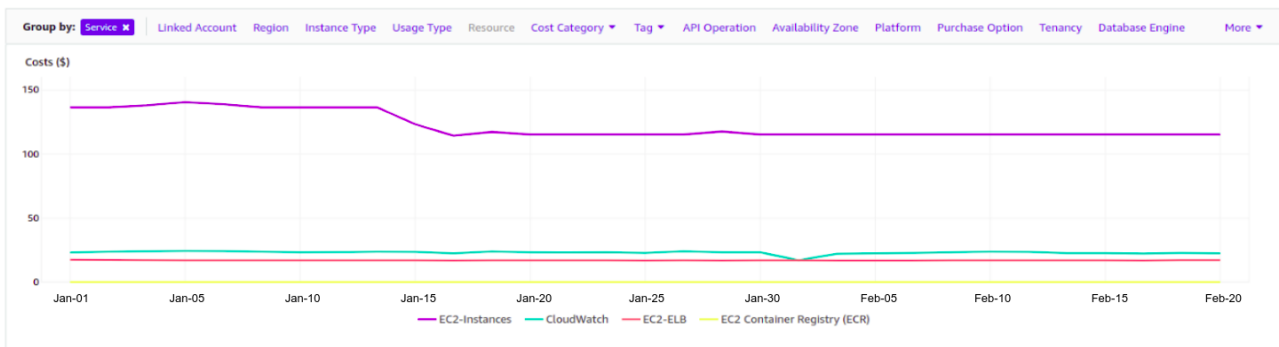
The graphs in Fig. 4 were obtained under the following conditions. During the period under review, about 140 different tasks were launched. Of these, 70 % were computational tasks with a runtime of more than 1 hour, the possibility of a startup delay of up to 1 hour and the ability to interrupt with a discreteness of 2 minutes; 20 % – computational tasks with a duration of up to 1 hour, the possibility of a start delay of up to 1 hour and the possibility of in-

terruption with a discreteness of 2 minutes; 10 % – tasks of all other types. The resource planning horizon and rental prices were 8 hours. The proposed price for spot resources was set at 5 % higher than the forecast price. The tasks were performed in the AWS data-centers in the us-east-1 region (N. Virginia). The listed tasks were formed by the development team, which included 1 frontend programmer, 2 backend programmers, 2 QA testers and 1 DevOps.

Fig. 5 shows a screenshot of the rental payment schedules from January 1 to February 20, 2021 for a different development group. The beginning of the use of the technique is January 13th. The reduction in resource rental costs in EC2 was over 18 percent. The composition of the group was the same. The differences consisted in the fact that during the period under review, more than 500 tasks were launched. The ratio between the types of tasks was similar, but the duration of the tasks from the first group was much more than 1 hour and several times longer than the tasks of the first group of developers. The resource planning horizon and rental prices



**Fig. 4. Schedule of changes in lease payments:**  
**1 – use of EC2 resources; 2 – use of S3 storage; 3 – use of CloudWatch monitoring service**  
 Source: compiled by the authors



**Fig. 5. Schedule of changes in lease payments:**  
**1 – use of EC2 resources; 2 – use of S3 storage; 3 – use of CloudWatch monitoring service;**  
**4 – EC2 Container Registry (ECR)**  
 Source: compiled by the authors

were 16 hours. The proposed price for spot resources was set at 5 % higher than the target price. The tasks were performed in AWS data centers in the us-east-1 (N. Virginia) region.

The obtained results indicate the efficiency of using the proposed methodology to save on the cost of renting computing resources in EC2. Moreover, efficiency increases when performing shorter tasks. This is explained by the fact that the decision block was looking for solutions focused on the continuous execution of tasks.

### 6. THE DISCUSSION OF THE RESULTS

Experimental verification of the proposed method has shown its effectiveness. At the same time, well-known algorithms for predicting the appearance of free spot resources and their prices were used, as well as a typical construction of a genetic algorithm that minimized the rental cost. Resource rentals were carried out only in one region us-east-1 (N. Virginia).

It seems promising to develop and study an algorithm that will analyze the feasibility of launching

tasks in different data centers located in different time zones and with different cloud providers. It should be noted that the efficiency of using the technique can be increased due to preliminary processing of the tasks being performed – dividing them into sub-tasks with a shorter execution time. However, everything is not so simple here, because when dividing into subtasks, you will have to increase the amount of stored intermediate data, which can lead to an increase in the cost of renting S3 storage.

When using the proposed methodology, the price for spot resources was set 5% higher than the forecast price. This made it possible to have practically no interruptions of running tasks. In the future, it is planned to conduct an additional study of the possibility of using a lower constant level of excess, as well as a dynamic change in this level depending on the parameters of the launched tasks and the forecast of the volumes of available spot resources.

It is also of interest to modify the methodology for the case when it is necessary to minimize not only the rental cost, but also the task execution time.

## 7. CONCLUSIONS

The paper proposes a method for reducing the cost of cloud infrastructure, which assumes, with a minimum time step, to recalculate the dynamics of launched tasks and leased resources based on current forecasts for the release or the possibility of addi-

tional loading of planned leased resources, predicting the appearance of spot resources and their prices. Minimization of rent is made subject to restrictions on the deadline for completing each of the tasks. Experimental verification of the proposed method has confirmed its effectiveness.

## REFERENCES

1. “AWS vs Azure vs google cloud-what is the best cloud platform for your business?”. 2021; January. – Available from: <https://embee.co.in/blog/aws-vs-azure-vs-google-cloud/>. – [Accessed: Feb 2021].
2. “Cloud cost optimization: 10 best practices for FinOps”. – Available from: <https://www.densify.com/resources/cloud-cost-management-best-practices>. – [Accessed: Feb 2021].
3. “Spot instances”. – Available from: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-instances.html>. – [Accessed: Feb 2021].
4. “Google anthos vs CAST AI comparison: Who wins?”. 2021; January. – Available from: <https://cast.ai/blog/cast-ai-vs-cloudability-which-one-to-pick-for-cloud-cost-optimization/n>. – [Accessed: Feb 2021]
5. “Cloud computing: principles and paradigms”. Edited by: Buyya, R., Broberg, J., Goscinski, A.-A. John, Wiley & Sons. Inc. Publication. 2011. 674 p.
6. Xiao, Z., Song, W. & Chen, Q. “Dynamic resource allocation using virtual machines for cloud computing environment”. *IEEE Trans Parallel Distrib Syst* . 2012; Vol. 24 No. 6: 1107–1117. DOI: <https://doi.org/10.1109/TPDS.2012.283>.
7. Wang, X., Wang, X., Che, H., Li, K., Huang, M. & Gao, C. “An intelligent economic approach for dynamic resource allocation in cloud services”. *IEEE Trans Cloud Comput* . 2015; Vol. 3 No. 3: 275–289. DOI: <https://doi.org/10.1109/TCC.2015.2415776>.
8. Mashayekhy, L., Nejad, M. M., Grosu, D. & Vasilakos, A. V. “An online mechanism for resource allocation and pricing in clouds”. *IEEE Trans Comput*. 2015; Vol. 65 No. 4: 1172–1184. DOI: <https://doi.org/10.1109/TC.2015.2444843>.
9. Zaman, S. & Grosu, D. “A combinatorial auction-based mechanism for dynamic VM provisioning and allocation in clouds”. *IEEE Trans Cloud Comput* . 2013; Vol. 1 No. 2: 129–141. DOI: <https://doi.org/10.1109/TCC.2013.9>.
10. Chou, L. D., Chen, H. F., Tseng, F. H., Chang, H. C. & Chang, Y. J. “DPRA: dynamic power-saving resource allocation for cloud data center using particle swarm optimization”. *IEEE Syst J*. 2016; Vol. 12 No. 2: 1554–1565. DOI: <https://doi.org/10.1109/JSYST.2016.2596299>
11. Pahlevan, A., Qu, X., Zapater, M. & Atienza, D. “Integrating heuristic and machine-learning methods for efficient virtual machine allocation in data centers”. *IEEE Trans Comput Aided Design of Integrated Circuits and Systems*. 2017; Vol. 37 No. 8: 1667–1680. DOI: <https://doi.org/10.1109/TCAD.2017.2760517>.
12. Shi, W., Zhang, L., Wu, C., Li, Z. & Lau, F.C. “An online auction framework for dynamic resource provisioning in cloud computing”. *IEEE/ACM Trans Networking*. 2015; Vol. 24 No. 4: 2060–2073. DOI: <https://doi.org/10.1109/TNET.2015.2444657>.
13. Du, J., Zhao, L., Feng, J. & Chu, X. “Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee”. *IEEE Trans Commun*. 2018; Vol. 66 No. 4: 1594–1608. DOI: <https://doi.org/10.1109/TCOMM.2017.2787700>.
14. Xiao, Ma, Shangguang, Wang, Shan, Zhang, Peng, Yang, Chuang, Lin & Xuemin, Shen. “Cost-efficient workload scheduling in cloud assisted mobile edge computing”. *IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*. 2017. DOI: <https://doi.org/10.1109/IWQoS.2017.7969148>.
15. Rodriguez, M. A. & Buyya, R. “Budget-driven scheduling of scientific workflows in IaaS clouds with fine-grained billing periods”. *ACM Transactions on Autonomous and Adaptive Systems*. 2017; Vol.12 No. 2: 5:1–5:22. DOI: <https://doi.org/10.1145/3041036>.
16. Alzhouri, F., Agarwal A. & Liu, Y. “Maximizing cloud revenue using dynamic pricing of multiple class virtual machines”. *IEEE Transactions on Cloud Computing*. 2021; Vol. 9 No. 2: 682–695. DOI: <https://doi.org/10.1109/TCC.2018.2878023>.
17. Liu, C., Li, K. & Li, K. “A Game Approach to multi-servers load balancing with load-dependent server availability consideration”. *IEEE Transactions on Cloud Computing*. 2021; Vol. 9 No. 1: 1–13. DOI: <https://doi.org/10.1109/TCC.2018.2790404>.

18. Liu, C., Li, K., Li, K. & Buyya, R. “A new service mechanism for profit optimizations of a cloud provider and its users”. *IEEE Transactions on Cloud Computing*. 2021; Vol. 9 No. 1: 14–26. DOI: <https://doi.org/10.1109/TCC.2017.2701793>.
19. Li, H., Ota, K., Dong, M., Vasilakos, A. V. & Nagano, K. “Multimedia processing pricing strategy in GPU-accelerated cloud computing”. *IEEE Transactions on Cloud Computing*. 2020; Vol. 8 No. 4: 1264–1273. DOI: <https://doi.org/10.1109/TCC.2017.2672554>.
20. Lučanin, D., Pietri, I., Holmbacka, S., Brandic, I., Lilius, J. & Sakellariou, R. “Performance-based pricing in multi-core geo-distributed cloud computing”. *IEEE Transactions on Cloud Computing*. 2020; Vol. 8 No. 4: 1079–1092. DOI: <https://doi.org/10.1109/TCC.2016.2628368>.
21. Khandelwal, V., Chaturvedi, A. K. & Gupta, C. P. “Amazon EC2 spot price prediction using regression random forests”. *IEEE Transactions on Cloud Computing*. 2020; Vol. 8 No. 1: 59–72. DOI: <https://doi.org/10.1109/TCC.2017.2780159>.
22. Ben-Yehuda, O. A., Ben-Yehuda, M., Schuster, A. & Tsafir, D. “Deconstructing amazon EC2 spot instance pricing”. *ACM Transactions on Economics and Computation*. 2013; Vol. 1 No. 3: 1–20. DOI: <https://doi.org/10.1145/2509413.2509416>.
23. Liu, W., Wang, P., Meng, Y., Zhao, C. & Liu, Z. Z. “Cloud spot instance price prediction using kNN regression”. *Hum. Cent. Comput. Inf. Sci.* 2020. p.10–34. DOI: <https://doi.org/10.1186/s13673-020-00239-5>.
24. Turchenko, V., Shults, V., Turchenko, I. & Wallace, R. M. “Spot price prediction for cloud computing using neural networks”. *International Journal of Computing*. 2013; Vol. 12 No. 4: 348–359.
25. Malik, M. & Bagmar, N. “Forecasting price of amazon spot instances using machine learning”. *International Journal of Artificial Intelligence and Machine Learning*. 2021; Vol. 11 No. 2: 71–82. DOI: <https://doi.org/10.4018/IJAIML.20210701.0a5>.
26. Tseng, F. H., Wang, X., Chou, L. D., Chao, H. C. & Leung, V. C. “Dynamic resource prediction and allocation for cloud data center using the multi objective genetic algorithm”. *IEEE Syst J.* 2017; Vol. 12 No. 2: 1688–1699. DOI: <https://doi.org/10.1109/JSYST.2017.2722476>.
27. Kaur, G. & Bala, A. “An efficient resource prediction-based scheduling technique for scientific applications in cloud environment”. *Concurrent Eng Res Appl.* 2019; Vol. 27 No. 2:112–125. DOI: <https://doi.org/10.1177/1063293X19832946>.
28. Gawali M. B. & Shinde S. K. “Task scheduling and resource allocation in cloud computing using a heuristic approach”. *Journal of Cloud Computing: Advances, Systems and Applications*. 2018; Vol. 7 No. 4: 1–16. DOI: <https://doi.org/10.1186/s13677-018-0105-8>.
29. Xiong, Y., Huang, S., Wu, M., She, J. & Jiang, K. “A Johnson’s-rule-based genetic algorithm for two-stage-task scheduling problem in data-centers of cloud computing”. *IEEE Transactions on Cloud Computing*. 2019; Vol. 7 No. 3: 597–610. DOI: <https://doi.org/10.1109/TCC.2017.2693187>.
30. Pavlenko, V. D. & Pavlenko, S. V. “Organization of computations in clusters using transparent parallelizing principles”. *Herald of Advanced Information Technology. Publ. Science i Technical*. Odessa: Ukraine. 2019; Vol.2 No. 1: 57–70. DOI: <https://doi.org/10.15276/hait.02.2019.6>.
31. “Spot instance interruptions”. – Available from: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/spot-interruptions.html#interruption-behavior>. – [Accessed: Feb 2021].
32. Coley, D. A. “An introduction to genetic algorithms for scientists and engineers”. *World Scientific Publishing Co. Pte. Ltd.* 1999. 244 p.
33. Buontempo, F. “Genetic algorithms and machine learning for programmers”. *The Pragmatic Programmers, LLC*. 2019. 234 p.

**Conflicts of Interest:** the authors declare no conflict of interest

Received 22.12.2020

Received after revision 10.03.2021

Accepted 16.03.2021

DOI: <https://doi.org/10.15276/aait.04.2021.6>

УДК 004.75:004.8:519.25

## **Зменшення витрат на хмарну інфраструктуру за рахунок управління завданнями**

**Олег Миколайович Галчонков<sup>1)</sup>**

ORCID: <https://orcid.org/0000-0001-5468-7299>; o.n.galchenkov@gmail.com. Scopus ID: 6507967225

**Микола Іванович Бабіч<sup>2)</sup>**

ORCID: <https://orcid.org/0000-0002-3946-9880>; babich.tiger@gmail.com

**Андрій Володимирович Плачинда<sup>2)</sup>**

ORCID: <https://orcid.org/0000-0001-8804-6852>; andrey.plachinda67@gmail.com

**Анастасія Романівна Майорова<sup>1)</sup>**

ORCID: <https://orcid.org/0000-0003-3004-6993>; anastasiamayorova2003@gmail.com

<sup>1)</sup> Одеський національний політехнічний університет, пр. Шевченка, 1. Одеса, 65044, Україна

<sup>2)</sup> Digitally Inspired LTD, вул. Генуезька, 2а. Одеса, 65000, Україна

## АНОТАЦІЯ

Перехід все більшої кількості підприємств від своєї обчислювальної інфраструктури в хмари обумовлений зменшенням витрат на її підтримку, найширшими можливостями по масштабуванню, наявністю великої кількості засобів автоматизації діяльності. Відповідно хмарні провайдери надають все більшу кількість різноманітних засобів та інструментів для роботи у хмарах. У свою чергу, це породжує завдання раціонального вибору типів хмарних послуг відповідно до особливостей розв'язуваних завдань. Одним із найпопулярніших напрямів зусиль споживачів хмарних сервісів є зменшення витрат на оренду. Основною базою цього напрямку є використання спотових ресурсів. У статті запропоновано методику зменшення витрат на оренду обчислювальних ресурсів у хмарі за рахунок динамічного управління розміщенням обчислювальних завдань, яке враховує можливе недозавантаження планових ресурсів, прогноз появи спотових ресурсів та вартості на них. Для кожної задачі формується вектор стану, що враховує тривалість виконання завдання та необхідний граничний термін виконання. Для відповідних наборів обчислювальних ресурсів формуються вектора прогнозу доступності на заданому часовому інтервалі, рахуючи від поточного моменту часу. Методика пропонує прораховувати в кожен дискретний момент часу найбільш раціональний варіант розміщення задачі на одному з ресурсів та затримку запуску задачі на ньому. Варіант розміщення та затримки запуску визначаються шляхом мінімізації функції вартості оренди на тимчасовому інтервалі за допомогою генетичного алгоритму. Однією з особливостей використання спотових ресурсів є аукціонний механізм їх надання хмарним провайдером. Це означає, що якщо є кращі пропозиції ціни оренди від будь-якого споживача, то провайдер може попередити вас про відключення ресурсу і зробити це відключення через оголошений час. Для мінімізації наслідків від такого відключення методика передбачає попередню підготовку завдань шляхом розбиття їх на підетапи з можливістю швидкого збереження поточних результатів у пам'яті та подальшого перезапуску з місця зупинки. Крім цього, для збільшення ймовірності того, що завдання не буде перервано, використовується прогноз ціни на типи ресурсів, що використовуються, і на аукціон хмарного провайдера пропонується дещо завищена ціна, порівняно з прогнозом. На прикладі використання середовища Elastic Cloud Computing (EC2) хмарного провайдера AWS показана ефективність запропонованої методики.

**Ключові слова:** хмарні обчислення; спотові ресурси; передбачення ресурсів; прогнози ціни; динамічне управління завданнями

## ABOUT THE AUTHORS



**Oleg N. Galchonkov** – Candidate of Engineering Sciences, Associate Professor of Information Systems Department. Odessa Polytechnic National University, 1, Shevchenko Ave. Odessa, 65044, Ukraine

ORCID: <http://orcid.org/0000-0001-5468-7299>; o.n.galchonkov@gmail.com. Scopus Author ID: 56081377900

**Research field:** Artificial intelligence; evolution systems; machine learning

**Олег Миколайович Галчонков** – канд. техн. наук, доцент кафедри Інформаційних систем. Одеський національний політехнічний університет, пр. Шевченка, 1. Одеса, 65044, Україна



**Mykola I. Babych** – Candidate of Engineering Sciences, BI Engineer (FE Developer). Digitally Inspired LTD, 2a, Genoese St. Odessa, 65000, Ukraine

ORCID: <https://orcid.org/0000-0002-3946-9880>; babich.tiger@gmail.com

**Research field:** Business intelligence; data mining; data visualization; data science; data structures in the construction of DWH; in-depth study of ETL processes

**Микола Іванович Бабіч** – канд. техн. наук, ВІ інженер (FE розробник). Digitally Inspired LTD, вул. Генуезька, 2а. Одеса, 65000, Україна



**Andrey V. Plachinda** – DevOps engineer, Digitally Inspired LTD, 2a, Genoese St. Odessa, 65000, Ukraine

ORCID: <https://orcid.org/0000-0001-8804-6852>; andrey.plachinda67@gmail.com

**Research field:** Automation; DevOps; administration; cloud technologies

**Андрій Володимирович Плачинда** – DevOps інженер, Digitally Inspired LTD, вул. Генуезька, 2а. Одеса, 65000, Україна



**Anastasia R. Majorova** – Student of Information Systems Department.

Odessa Polytechnic National University, 1, Shevchenko Ave. Odessa, 65044, Ukraine

ORCID: <https://orcid.org/0000-0003-3004-6993>; anastasiamayorova2003@gmail.com

**Research field:** Artificial intelligence; evolution systems; machine learning

**Анастасія Романівна Майорова** – студентка кафедри Інформаційних систем. Одеський національний політехнічний університет, пр. Шевченка, 1. Одеса, 65044, Україна