UDC 004.94
DOI: https://doi.org/10.17721/AIT.2024.1.05

Vitalii OMELCHENKO, PhD Student
ORCID ID: 0000-0002-3850-6555
e-mail: vitaly.om25@gmail.com
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine

Oleksandr ROLIK, DSc (Engin.), Prof.
ORCID ID: 0000-0001-8829-4645
e-mail: arolick@gmail.com
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine

HYBRID METHOD FOR HORIZONTAL AND VERTICAL COMPUTATIONAL RESOURCE SCALING

Background. ***Increasing the efficiency of cluster computing resources while maintaining the established QoS levels is a critical task in IT infrastructure management. Dynamic management of computing resources, in particular vertical and horizontal scaling, are tools that allow automating the processes of adapting applications to dynamic loads. The aim of the work is to improve the efficiency of existing scaling methods by combining them.***

Methods. ***A hybrid scaling method using the coordination module is proposed. This module coordinates the operation of vertical and horizontal scaling components based on the given constraints, priorities, and the current state of the system. The coordination module aims to increase efficiency of the both components and prevents inconsistency in the combination of the number of instances and resource requests for each instance.***

Results. ***To achieve the given objective, a hybrid method of vertical and horizontal scaling using priority-based component coordination was developed. Priority configuration affects the order of components operation. In the case of horizontal-vertical order, the non-prior vertical component does not affect configuration of the prior horizontal component. The developed method is evaluated based on modeling the operation of an application with a load containing a constant seasonality. The experiments demonstrate a 65% reduction in the unprofitable reservation of cluster computing resources compared to static requests.***

Conclusions. ***The developed method can be used to increase the efficiency of resource utilization in clusters under dynamic loads compared to basic scaling methods. In further research, it is necessary to evaluate the method using real infrastructure. It is also necessary to investigate the work of a hybrid method using a predictive approach.***

Keywords: ***information technologies, information systems, resource management, scaling, IT-infrastructure, vertical scaling, horizontal scaling.***

Background

The dynamic allocation of computational resources for cloud applications enables the delivery of the agreed level of quality of service (QoS) to customers. An increasing number of enterprise informational systems (IS) rely on existing solutions for the management of computational resources (Rolik, Telenik, & Yasochka, 2018).

Horizontal and vertical scaling of applications in a cluster is one of the most effective tools for automating these processes in clusters (Lorido-Botran, Miguel-Alonso, & Lozano, 2014). Horizontal scaling is more versatile than vertical scaling because it provides fault tolerance, is not limited to the resources of a single physical machine, and allows flexible management of the amount of resources allocated (Al-Dhuraibi et al., 2017). In practice, vertical scaling is used when an application cannot be distributed, such as databases (Omelchenko, & Rolik, 2022). Although horizontal scaling is a flexible way of dynamically allocating resources, this process takes place within the limits of defined requirements. If the amount of computing resources allocated to the application and the actual load on each resource are not balanced, then part of this resource is not used. By adjusting the volume of requests within an application, its instances can be placed on more specialized VMs, reducing financial costs while maintaining the same level of QoS (Rochman, Levy, & Brosh, 2014).

The limitations of standalone scaling methods and the high demands on modern IS resulted in the emergence of hybrid methods (Straesser et al., 2022). In particular, vertical-horizontal and reactive-proactive methods combine the advantages of existing approaches to improve overall performance (Singh et al., 2019). However, the main disadvantage of hybrid methods is the complexity of coordinating individual components.

This work aims to increase the efficiency of computing resource utilization while maintaining the established level of QoS in dynamic resource management. To achieve this goal, a combined scaling method is proposed, which includes coordination of horizontal and vertical scaling components.

Problem statement. Horizontal scaling is the process of increasing or decreasing the number of application instances to ensure QoS levels within the agreed SLA and to increase the efficiency of the use of the cluster's compute resources (Rolik, & Omelchenko, 2024). This type of scaling can work with either a single resource type or multiple resources. In the case of scaling a single resource at a time, the scaling of the *l*-th application can be described as follows:

$$k_l(t) = \left\lceil \frac{W_l(t)}{X_l} \right\rceil, l = 1, ..., m,$$

where m is the number of applications, $W_l(t)$ is the utilization of the target compute resource, X_l is the request for the target resource type. This is the level of utilization of other types of computing resources, which can lead to both denial of service and significant unprofitable redundancy (Omelchenko, & Rolik, 2022).

In the case of scaling using a set of resources, the one with the highest utilization percentage in relation to the target value is selected:

$$k_l(t) = \max\left(\left\lceil \frac{W_{ly}(t)}{X_{ly}} \right\rceil \right), l = 1, ..., m, y = 1, ..., h,$$

where m is the number of applications, $W_{ly}(t)$ is the usage of the y-th computing resource, and X_{ly} is the request for the y-th type of resource. Accordingly, only one of the resources is taken into account and the others may be unbalanced.

The horizontal scaling model described has the disadvantage of insufficient utilization of non-priority resource types. A non-priority resource does not affect the amount of resources allocated for scheduling in general or at any particular time (Sedaghat, Hernandez-Rodriguez, & Elmroth, 2013). Fig. 1 shows an example of unbalanced resource allocation for an application where CPU time utilization is kept at a configured level of 90% but other resource utilization is relatively low. Reducing the number of instances is impossible due to the need to ensure a given level of QoS, and partial adjustment of the application's resource volume cannot be performed by the horizontal scaling component. To solve this problem, it is necessary to manually adjust the configuration of requests and horizontal scaling to current needs, which is a difficult task in large IS.

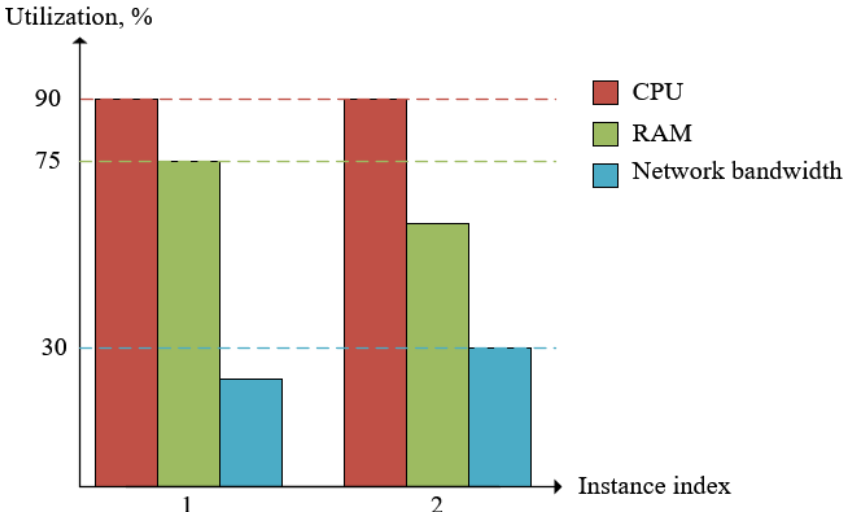

Fig. 1. Unbalanced utilization of computing resources during horizontal scaling

This paper proposes a hybrid scaling method to automate this process, which is derived from horizontal scaling but is more efficient in terms of computing resources utilization. The proposed method is designed to solve the following problems when operating horizontally scaled applications:

- increasing the level of utilization of resource types that are not a priority for horizontal scaling, in order to increase the efficiency of their use;
- resolving situations where the current instance usage of one of the resource types exceeds the set request for that resource type, in particular out-of-memory (OOM) errors leading to denial of service.

Methods

The topic of combining vertical and horizontal scaling is increasingly appearing in scientific papers (Qu, Calheiros, & Buyya, 2018). Most approaches are based on forecasting; in the paper (Dutta et al., 2012) an approach using predictive model management is proposed. The paper proposes a model with a given number of parameters that are determined based on the cluster capacity, the predicted volume of the task queue, and the application's request for computing resources. Based on the constraints and predictions, the task of optimizing resource allocation is solved. The QoS metrics of the combined method obtained as a result of experiments show an increase compared to horizontal scaling, but the evaluation of the efficiency of the use of computing resources is not examined. In the next paper (Incerto, Tribastone, & Trubiani, 2018), another approach is proposed in which the virtual machine is scaled vertically until the limit of the physical machine is met, then horizontal scaling is applied. In this work, it is assumed that horizontal scaling can provide the required level of QoS, so considerable attention is paid to the efficiency of using the cluster's computing resources. In addition, the proposed method can use both historical data and predictive models in its work. In the paper (Quattrocchi et al., 2024), the authors propose a system model based on load and response time data using queueing theory. Using the built model of the system, the solution performs an optimization task to determine the minimum amount of resources (e.g., CPU) required to provide the target response time under the current load. Another work (Millnert, & Eker, 2020) proposes a novel approach based on control theory. The authors propose using "feedback" vertical controller and "feedforward" horizontal controllers that are coordinated by calculated time delays for applying control signals.

For a successful co-operation of horizontal and vertical scaling, it is necessary to coordinate their work (Calheiros et al., 2012). The coordination should be such that the changes made by each component are fully consistent. Unprocessed conflicting changes can lead to QoS degradation and denial of service. Fig. 2 shows the coordination module containing data about the characteristics and capabilities of each scaling method.

In the coordination module, it is proposed to use the principle of priority to coordinate the work of the components when one of the components is secondary and should in no way affect the current decisions of the higher priority control component, but only adapt to the decisions provided. Uncoordinated decision making can lead to incorrect scaling of the application and a drop in QoS. The coordination module ensures that the configurations of the scaling components are compatible as follows:

$$S(x) = \begin{cases} H(x, V(x, null)), P_V > P_H, \\ V(x, H(v, null)), P_H > P_V, \end{cases}$$

where $S(x)$ is the scaling function, $H(x,c)$ and $V(x,c)$ are the scaling functions of the horizontal and vertical components respectively, taking as input the historical data and the current configuration, P_H and P_V are the priorities of the respective components.

ISSN 2788-6603

The coordination module prefers horizontal scaling, as this type is not limited by the resources of a single physical machine and is more flexible for use in modern microservices architectures. Fig. 3 demonstrates the comparison of the vertical and horizontal scaling methods.

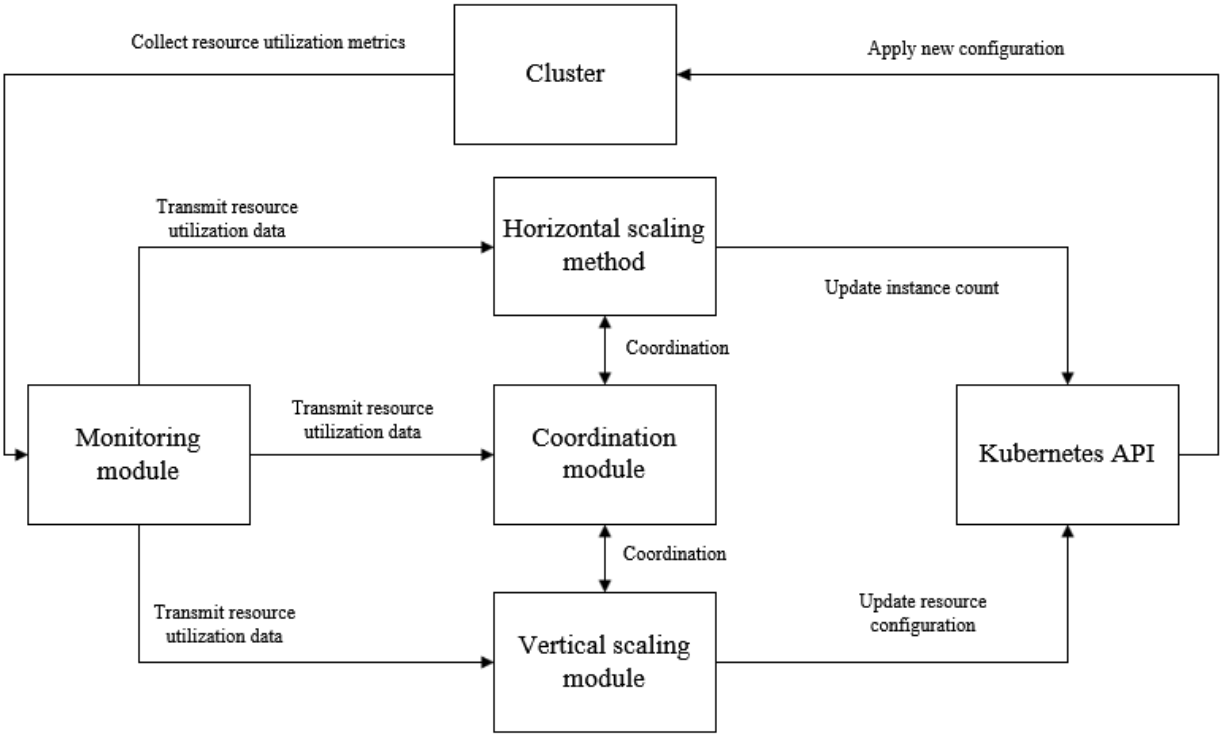

Fig. 2. Component diagram of the hybrid scaling method

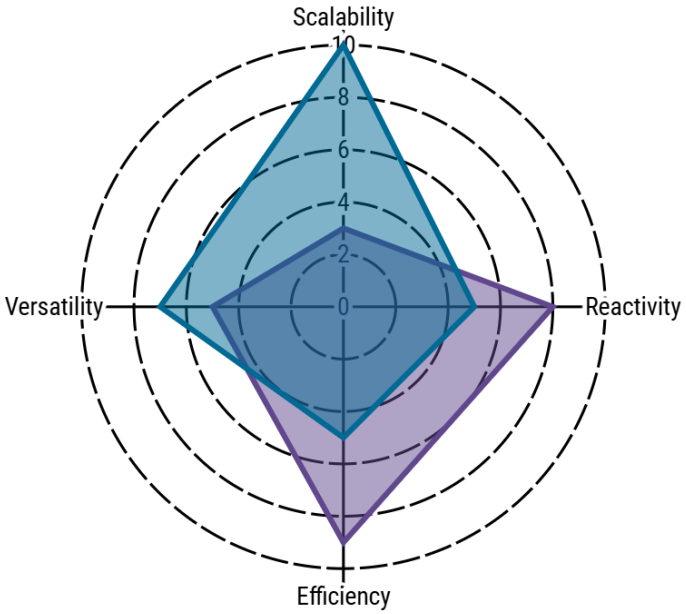

Fig. 3. Comparison of characteristics of scaling types

To implement this method, both sources of historical data on application utilization and data obtained as a result of forecasting models can be used. The data is transferred to the horizontal management module, where a decision is made on whether to increase or decrease the number of instances. This module also receives information about current resource requests $\hat{X}_{ly}$, $l = 1,\ldots,m, y = 1,\ldots,h$, where h – the number of resource types, m – the number of applications, and T_{ly}, $0 < T_{ly} \leq 1$, is the target value of utilization for each resource type. The output of the module operation is the number of instances of the l-th application k_l. An example of such a module is HPA.

In the vertical scaling module, the first step is to determine the level of utilization of each of the computing resources:

$$U_{ly} = \frac{X_{ly}}{\hat{X}_{ly}}, l = 1,...,m, y = 1,...,h,$$

where U_{ly} – utilization of the l-th application of the y-th resource type. Using the set level of target utilization T_{ly} priority resource type is found, which is used for horizontal scaling calculations:

$$y_{pivot} = \arg\max(\frac{U_{ly}}{T_{ly}}), l = 1,...,m, y = 1,...,h,$$

where y_{pivot} is the index of the resource type with the maximum relative utilization of the l-th application of the y-th type of computing resource. The next step is to estimate the required quantities of non-priority resources using the target utilization levels T_{ly}:

$$\begin{cases} \hat{X}_{ly} = f(\{X_{ly}(t_0),\ X_{ly}(t_1)...,\ X_{ly}(t_n)\}, T_{ly}),\ l = 1...m,\ y = 1...h,\ n = 1...N, \\ y \neq y_{pivot}, \end{cases}$$

where $\hat{X}_{ly}$ is the optimal values of requests for computing resources at the current iteration calculated on the basis of historical data or forecasts, for $X_{ly}(t)$ there is a transition from a scalar value to a function, since to calculate requests, it is necessary to take into account a segment of historical data or obtained forecasts, and $f(x,y)$ is a function for calculating requests. Depending on the configuration, different optimizations can be applied to this function.

When scaling using all available computing resources, it is proposed to use a generic function, the result of which guarantees the correct operation of applications and scaling components:

$$f(x,u) = \frac{\max(x)}{u},$$

where x is a vector of values for the utilization of a computing resource, and u is a target value of its utilization. If y_{pivot} is a fixed value, there is potential for optimization based on the resource type.

Each type of computing resource has its own management specifics. In this paper we propose to divide them into two groups. The first group includes resource types whose excessive usage does not lead to a complete denial of service, as the shortage is amortized in subsequent application cycles (Rodriguez, & Buyya, 2018). These types of computing resources can be artificially limited to meet defined requirements. For example, in the case of a CPU resource, the application is only given the specified amount of time to operate on the processor core, and any operations that exceed this time are performed in the next period of operation (Al-Haidari, Sqalli, & Salah, 2013). A network resource has a similar operating flow, where a queue of packets is created before transmission. For computing resources from the first group, it is proposed to use a set percentile when calculating requests:

$$f_1(x, p) = x_{sorted}\left[\left[\frac{p}{100}\cdot(n-1)\right]\right],$$

where x is a vector of values of the utilization of a certain computing resource, n is the length of the vector x, and p is target utilization percentile. This optimization can be useful for cases with short term peaks in application workload. The percentile allows you to calculate requests that cover most of the present workload scenarios and avoid resource over-provisioning.

Fig. 4 shows the integration of hybrid scaling components in a Kubernetes cluster. The monitoring components metrics-service and Prometheus provide data for horizontal scaling. The usage either predictive or reactive scaling methods is determined in the decision module. After the planned number of instances is determined, this data is passed to the vertical scaling component, which optimizes requests for computing resources based on the data received.

The apply module is responsible for passing the new deployment configuration to the Kubernetes API. After that, the next monitoring and scaling cycle is performed.

Results

To evaluate the effectiveness of the proposed method, simulation of the operation of an application that scales both horizontally and vertically is performed. Conducting experiments on a Kubernetes infrastructure is complicated by the fact that in order to update an application's requests, all of its instances need to be restarted. On a real infrastructure, this is achieved by gradually restarting each instance, which does not require significant additional computing resources and does not affect QoS levels. The simulation is done using Python and the Pandas library. The horizontal scaling model is similar to HPA and a reactive approach is used. Target utilisation levels for CPU time and memory are set and requests are reevaluated at 100 second intervals, reflecting the frequency of metric collection and aggregation in a real cluster.

The first experiment allows us to test the work when the priority resource for horizontal scaling is CPU time. Accordingly, memory is scaled vertically. In this study, the total load with a periodicity of 1000 seconds ranges from 200 millicores to 1200 millicores, and the CPU time requirements are set at 250 millicores with a target utilization of 90%. So the application scales from two to five instances. Fig. 5 shows a graph of the workload and the number of instances to process it. Also in Fig. 5, there is ascaling delay when the load appears and, accordingly, the delay before reducing the number of instances, which corresponds to the behaviour of the HPA.

Fig. 6 demonstrates scaling of the memory resource in the first experiment. The initial configuration of memory requests is not optimal and set to 100 megabytes. The model takes into account that each instance has a constant component of memory usage and another one that depends on the load. Accordingly, during the first two periods of operation, memory utilization is 50%. After the component has received a sufficient amount of data, namely 2000 seconds, it scales down memory

ISSN 2788-6603

requests to the target level of 90%. In the fourth period of operation, an atypical peak in memory usage appears and the vertical scaling component responds by increasing memory requests accordingly. After two periods of low utilization, the memory scales down again to 90% utilization during the last two periods of operation.

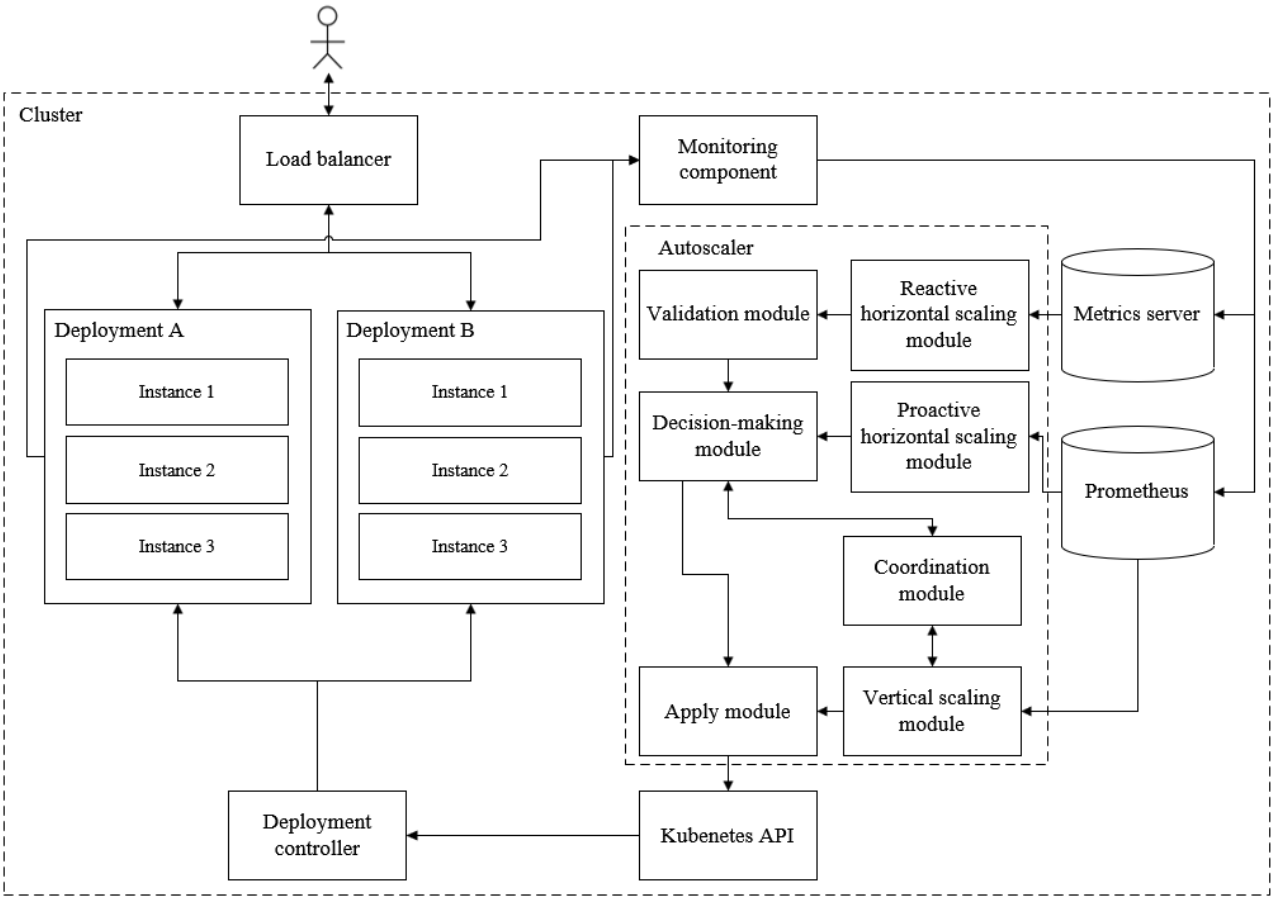

Fig. 4. Structural diagram of a cluster containing hybrid scaling

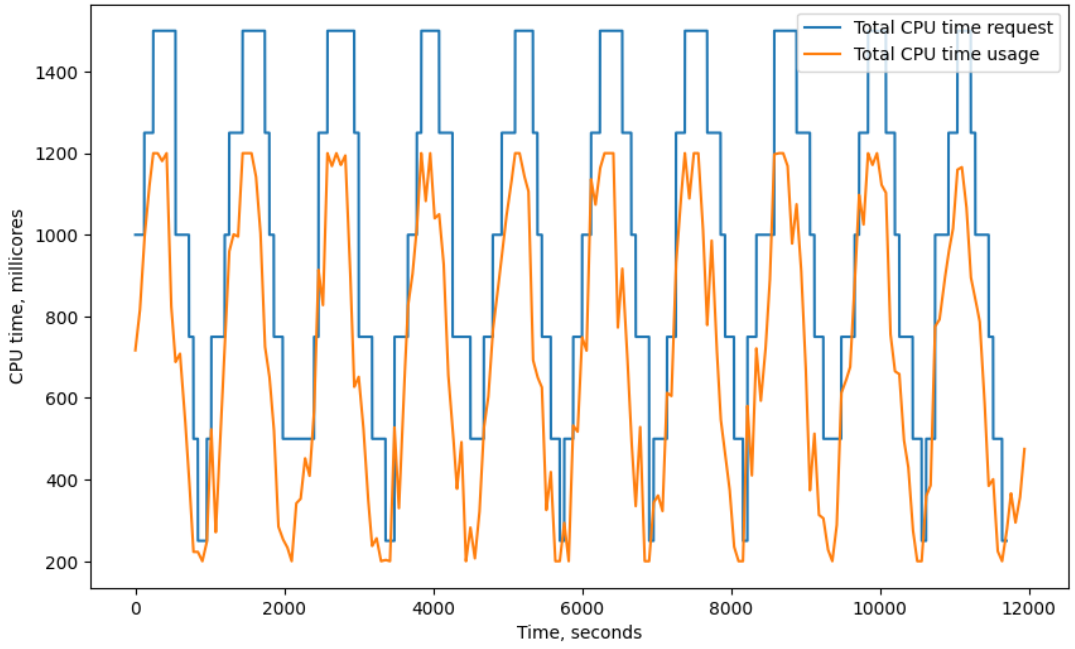

Fig. 5. Horizontal scaling of the application using CPU time

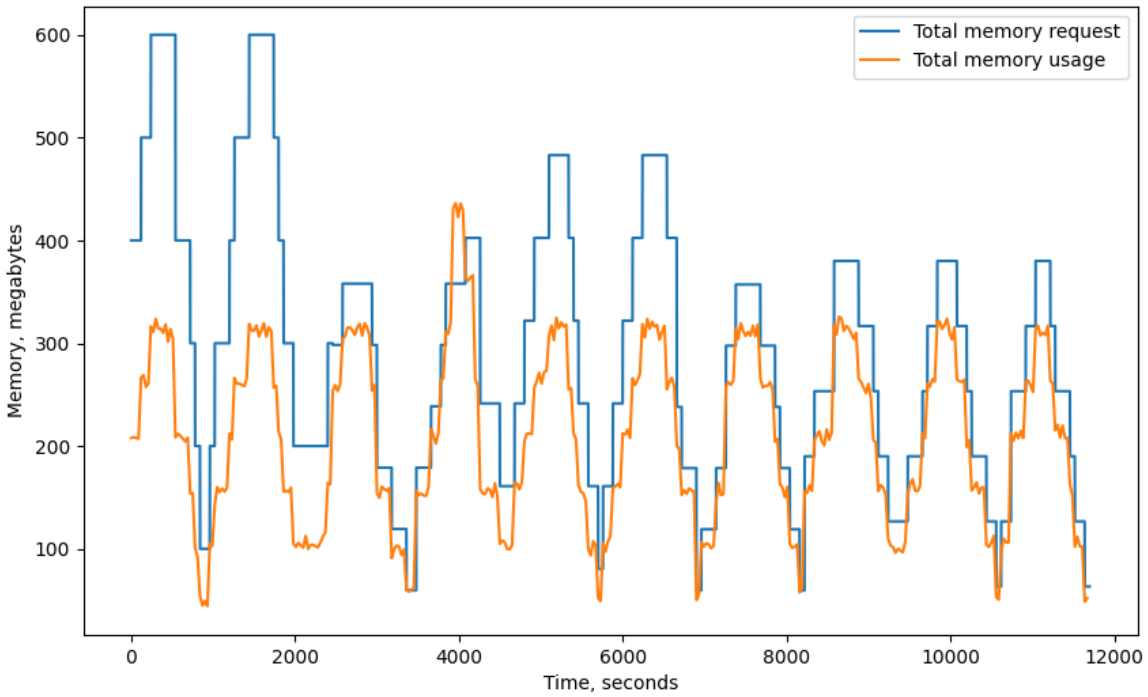

Fig. 6. Vertical scaling of memory requests

The next experiment is intended to evaluate the method's performance when horizontal scaling is performed using memory and CPU time is vertically scaled. Fig. 7 shows the application load and the number of application instances.

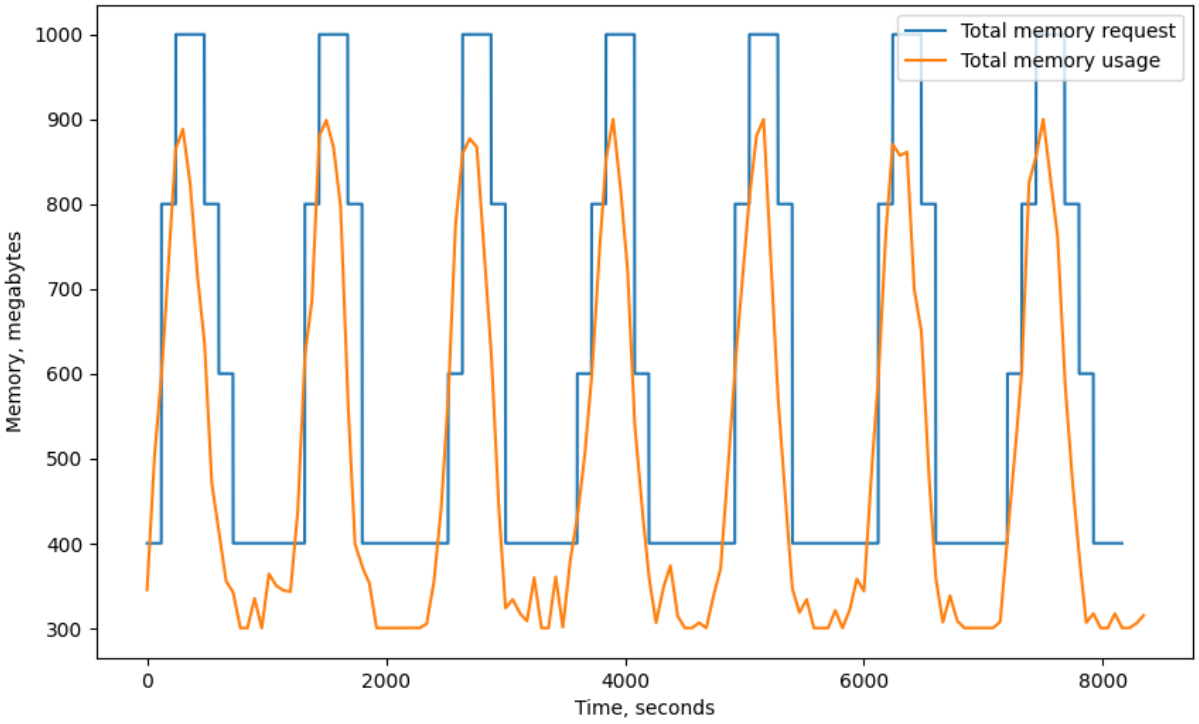

Fig. 7. Horizontal scaling of the application using memory

Fig. 8 shows the results of the vertical scaling. The 95th percentile is used to calculate the CPU time requirements. In Fig. 8, there is a similar pattern of operation for downscaling and upscaling. The difference with the first experiment is the higher level of resource utilization for the vertical scaling.

The last experiment is focused on evaluating the efficiency depending on the length of historical data taken into account during vertical scaling. Horizontal scaling is performed from memory at 1000-second intervals, as shown in Fig. 9.

ISSN 2788-6603

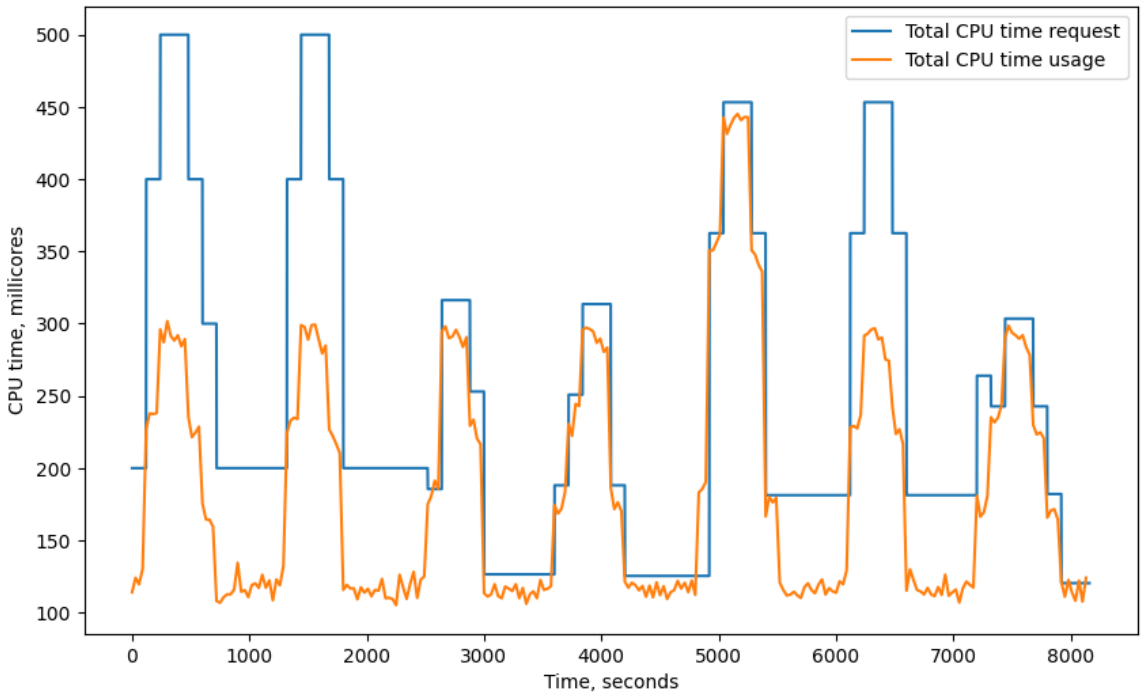

Fig. 8. Vertical scaling of requests using CPU time

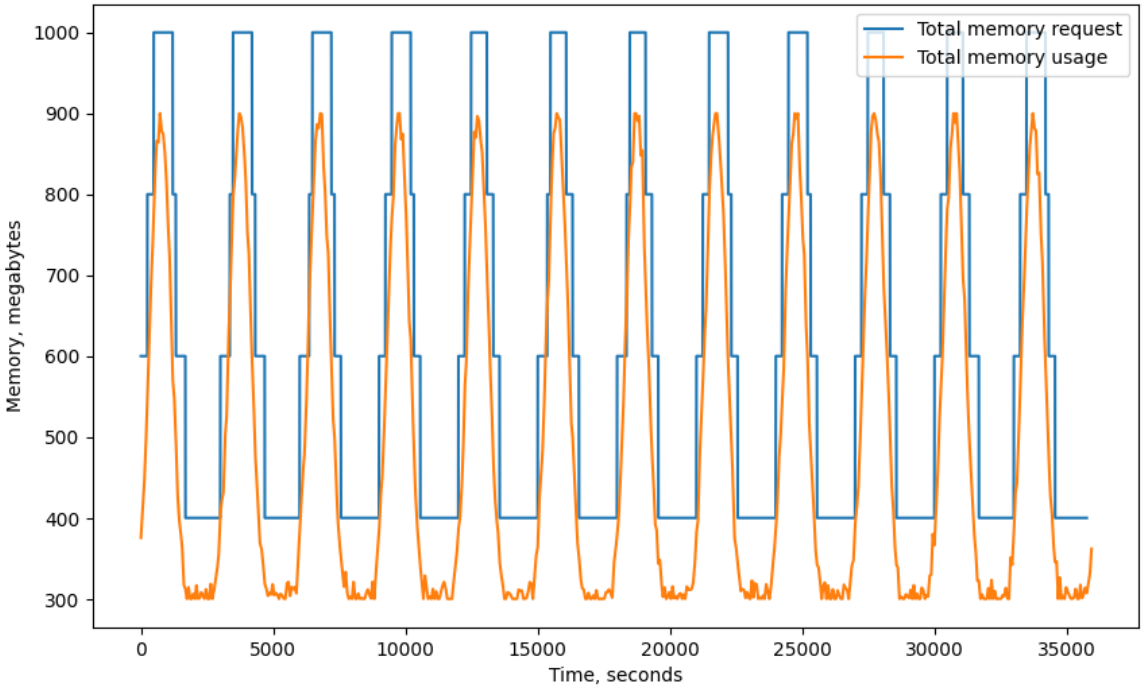

Fig. 9. Horizontal scaling of the application using memory

Fig. 10 shows the results of the vertical scaling component operation. The size of the historical data taken into account is 1000 seconds. The target CPU time utilization is 95%.

A similar experiment is shown in Fig. 11, but the size of the historical data is increased to 4000 seconds:

The results show that scaling occurs with a much shorter period, which, on the one hand, leads to lower resource efficiency and, on the other hand, to better QoS performance at peak loads. This experiment demonstrates the possibility of adapting the developed method to the set SLA requirements.

Further, Fig. 12 compares CPU time utilization for static and dynamic requests. The ratio of resource utilization to requests for dynamic requests ranges from 80% to 120%. In this case, the level of QoS depends on many other factors, including the level of the set limits. At the same time, the use of static requests, at the 95th percentile level, leads to a low utilization rate of 30% to 90% in the given conditions.

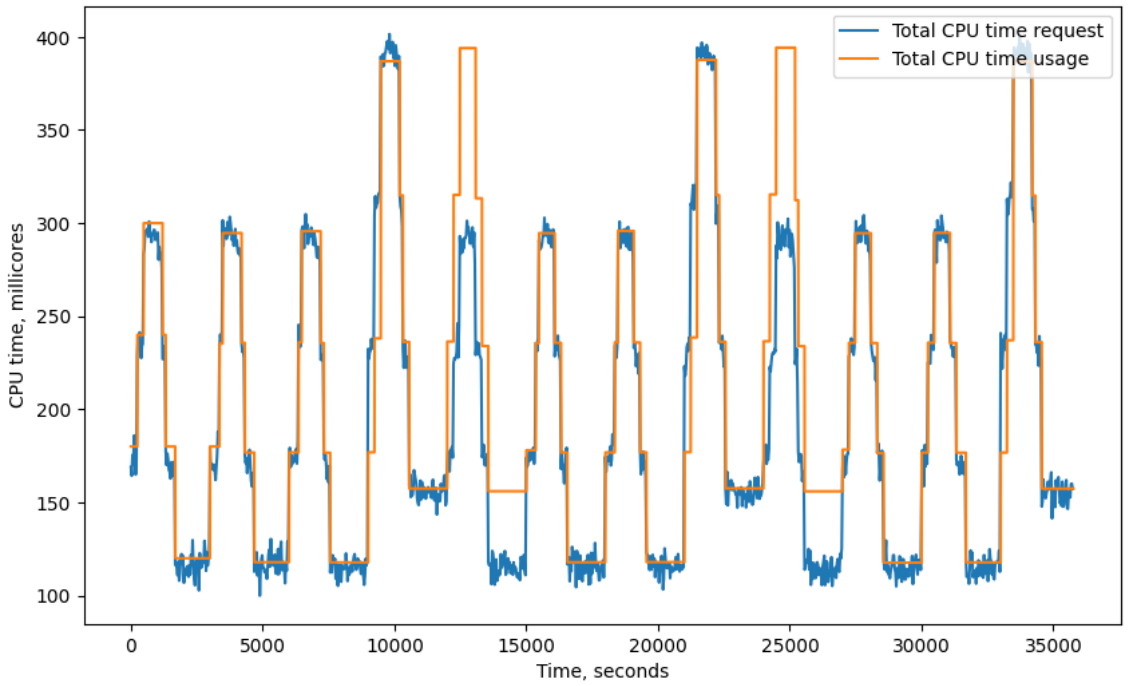

Fig. 10. Vertical scaling of the application for CPU time with a historical data length of 1000 seconds

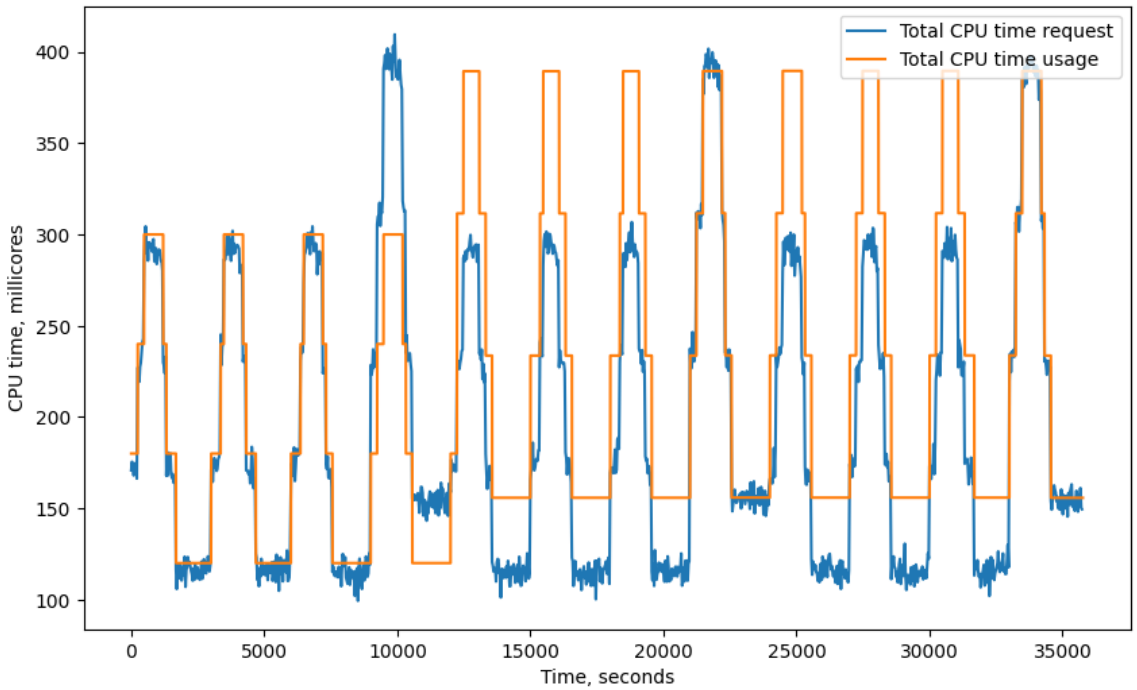

Fig. 11. Vertical scaling of the application by CPU time with 4000 seconds of historical data

The amount of allocated CPU time for dynamic requests is 65% less than for static ones. The conducted experiments are simulations and the results may differ significantly on real infrastructure, but they allow us to evaluate the theoretical possibility of applying the developed method.

Discussion and conclusion

The paper proposes a hybrid scaling method that includes both a horizontal and a vertical component. An optimization method is proposed for resources such as CPU time and network bandwidth. The simulation results show that vertical scaling can increase the efficiency of using cluster computing resources and prove the feasibility of developing a full-featured solution for use in Kubernetes and other orchestration solutions. The developed method proposes reactive management based on current state of the system, but it is possible to use forecasts to manage resource allocation in advance.

ISSN 2788-6603

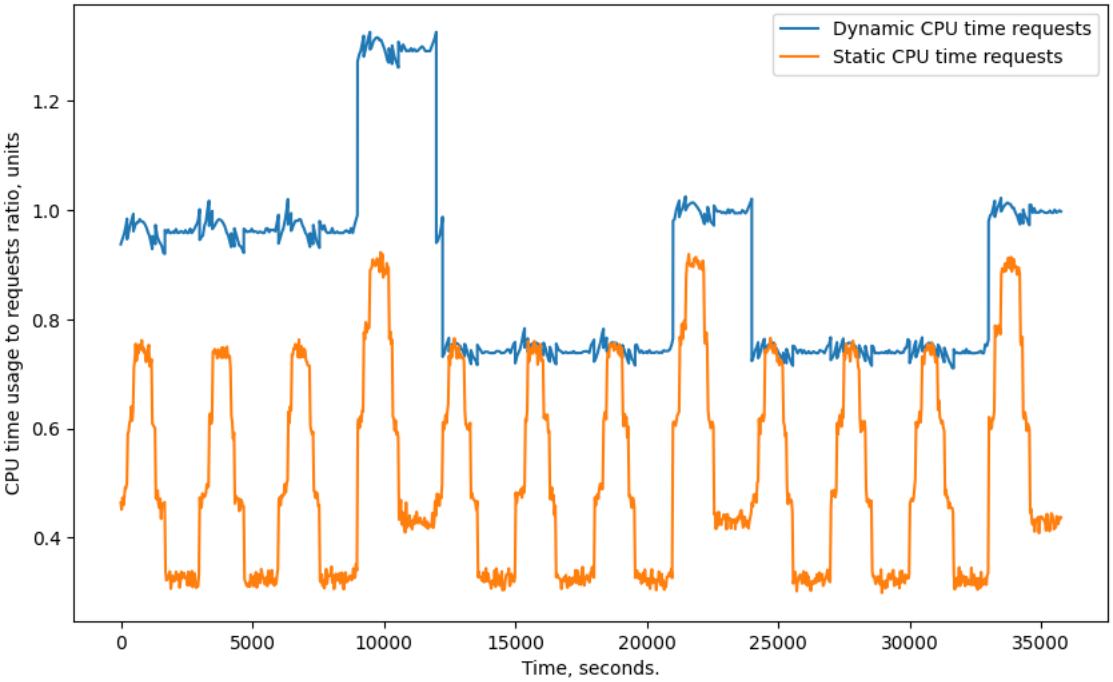

Fig. 12. Comparison of resource utilization with static and dynamic control

Authors' contribution: Vitalii Omelchenko – method development, solution implementation, performing of experiments. Oleksandr Rolik – development of the method and methodology, evaluation of results.

References

Al-Dhuraibi, Y., Paraiso, F., Djarallah, N., & Merle, P. (2017). Elasticity in cloud computing: State of the art and research challenges. *IEEE Transactions on Services Computing*, 11(2), 430–447. Institute of Electrical and Electronics Engineers. https://doi.org/10.1109/TSC.2017.2711009

Al-Haidari, F., Sqalli, M. H., & Salah, K. (2013). Impact of CPU utilization thresholds and scaling size on autoscaling cloud resources. *IEEE 5th International*. In I. Foster, E. Feig, & S. Yau (Eds.) *Conference on Cloud Computing Technology and Science*, 2 (pp. 256–261). Institute of Electrical and Electronics Engineers.

Calheiros, R. N., Toosi, A. N., Vecchiola, C., & Buyya, R. (2012). A coordinator for scaling elastic applications across multiple clouds. *Future Generation Computer Systems*, 28(8), 1350–1362. https://doi.org/10.1016/j.future.2012.03.010

Dutta, S., Gera, S., Verma, A., & Viswanathan, B. (2012). SmartScale: Automatic Application Scaling in Enterprise Clouds. In I. Foster, E. Feig, & S. Yau (Eds.). *IEEE 5th International Conference on Cloud Computing* (pp. 221–228). IEEE. https://doi.org/10.1109/cloud.2012.12

Incerto, E., Tribastone, M., & Trubiani, C. (2018). *Combined Vertical and Horizontal Autoscaling Through Model Predictive Control.* In M. Aldinucci, L. Padovani, & M. Torquati (Eds.). *Lecture Notes in Computer Science. Vol. 11014. Parallel Processing* (pp. 147–159). Springer International Publishing. https://doi.org/10.1007/978-3-319-96983-1_11

Lorido-Botran, T., Miguel-Alonso, J., & Lozano, J. A. (2014). A Review of Auto-scaling Techniques for Elastic Applications in Cloud Environments. *Journal of Grid Computing*, 12(4), 559–592. https://doi.org/10.1007/s10723-014-9314-7

Millnert, V., & Eker, J. (2020). HoloScale: Horizontal and vertical scaling of cloud resources. In N. Antonopoulos, O. Rana, & Ch. Jiang (Eds.) *2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*, 55 (pp. 196–205). IEEE. https://doi.org/10.1109/ucc48980.2020.00038

Omelchenko, V., & Rolik, O. (2022). Automation of resource management in information systems based on reactive vertical scaling. *Adaptive systems of automatic control,* 2(41), 65–78. National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute". https://doi.org/10.20535/1560-8956.41.2022.271344

Qu, C., Calheiros, R. N., & Buyya, R. (2018). Auto-scaling web applications in clouds. *ACM Computing Surveys*, 51(4), 1–33. https://doi.org/10.1145/3148149

Quattrocchi, G., Incerto, E., Pinciroli, R., Trubiani, C., & Baresi, L. (2024). Autoscaling Solutions for Cloud Applications Under Dynamic Workloads. In *IEEE Transactions on Services Computing*, 17(3), 804–820. IEEE. https://doi.org/10.1109/tsc.2024.3354062

Rodriguez, M. A., & Buyya, R. (2018). Container-based cluster orchestration systems: A taxonomy and future directions. *Software: Practice and Experiencee,* 49(5), 698–719. https://doi.org/10.1002/spe.2660

Rochman, Y., Levy, H., & Brosh, E. (2014). Efficient resource placement in cloud computing and network applications. *SIGMETRICS Performance Evaluation Review*, 42(2), 49–51. https://doi.org/10.1145/2667522.2667538

Rolik, O., Telenik, S., & Yasochka, M. (2018). *Enterprise IT-infrastructure management.* Naukova dumka [in Ukrainian]. [Ролік О., Теленик С., & Ясочка, М. (2018). *Управління корпоративною IT-інфраструктурою*. Наукова думка].

Rolik, O., & Omelchenko, V. (2024). Proactive horizontal scaling method for Kubernetes. *Radio Electronics, Computer Science, Control*, 1, 221–227. National University "Zaporizhzhia Polytechnic". https://doi.org/10.15588/1607-3274-2024-1-20

Sedaghat, M., Hernandez-Rodriguez, F., & Elmroth, E. (2013). A virtual machine re-packing approach to the horizontal vs. vertical elasticity trade-off for cloud autoscaling. In S. Hariri, & A. Sill (Eds.). *2013 ACM Cloud and Autonomic Computing Conference* (pp. 1–10). ACM. https://doi.org/10.1145/2494621.2494628

Singh, P., Gupta, P., Jyoti, K., & Nayyar, A. (2019). Research on auto-scaling of web applications in cloud: Survey, trends, and future directions. *Scalable Computing: Practice and Experience*, 20(2), 399–432. https://doi.org/10.12694/scpe.v20i2.1537

Straesser, M., Grohmann, J., von Kistowski, J., Eismann, S., Bauer, A., & Kounev, S. (2022). Why is it not solved yet? In D. Feng, & S. Becker (Eds.). *2022 ACM/SPEC International Conference on Performance Engineering* (pp. 105–115). ACM. https://doi.org/10.1145/3489525.3511680

Отримано редакцією журналу / Received: 12.09.24
Прорецензовано / Revised: 23.10.24
Схвалено до друку / Accepted: 07.11.24

Віталій ОМЕЛЬЧЕНКО, асп.
ORCID ID: 0000-0002-3850-6555
e-mail: vitaly.om25@gmail.com
Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського", Київ, Україна

Олександр РОЛІК, д-р техн. наук, проф.
ORCID ID: 0000-0001-8829-4645
e-mail: arolick@gmail.com
Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського", Київ, Україна

ГІБРИДНИЙ МЕТОД ГОРИЗОНТАЛЬНОГО І ВЕРТИКАЛЬНОГО МАСШТАБУВАННЯ ОБЧИСЛЮВАЛЬНИХ РЕСУРСІВ

В с т у п . ***Підвищення ефективності використання обчислювальних ресурсів кластера при дотриманні встановлених рівнів QoS є критично важливим завданням у керуванні IT-інфраструктурою. Динамічне керування обчислювальними ресурсами, зокрема вертикальне та горизонтальне масштабування, є інструментом, що дозволяє автоматизувати процеси адаптації застосунків до динамічних навантажень. Метою цієї роботи є підвищення ефективності існуючих методів масштабування за допомогою їхнього комбінування.***

М е т о д и . ***Запропоновано гібридний метод масштабування з використанням модуля координації. Цей модуль узгоджує роботу компонентів вертикального та горизонтального масштабування на основі заданих обмежень, пріоритетів і поточного стану системи. Модуль координації відповідає за підвищення ефективності обох компонентів масштабування і запобігає неузгодженості у процесі конфігурації кількості екземплярів застосунку та запитів на обчислювальні ресурси.***

Р е з у л ь т а т и . ***Для досягнення поставленої мети розроблено гібридний метод вертикального та горизонтального масштабування з використанням координації компонентів на основі пріоритетів. Пріоритетна конфігурація визначає порядок роботи компонентів під час узгодження їхньої роботи. У випадку горизонтально-вертикального порядку непріоритетний вертикальний компонент не впливає на конфігурацію пріоритетного горизонтального компонента. Проведено оцінювання розробленого методу на основі моделювання роботи застосунку з навантаженням, що містить постійну сезонність. Експерименти демонструють зменшення збиткового резервування обчислювальних ресурсів кластера на 65 % порівняно зі статичними запитами.***

В и с н о в к и . ***Розроблений метод можна застосувати для підвищення ефективності використання ресурсів у кластерах під час динамічних навантажень порівняно з базовими методами масштабування. У подальших дослідженнях необхідно оцінити метод на реальній інфраструктурі. Також варто дослідити роботу гібридного методу з використанням предиктивного підходу.***

К л ю ч о в і с л о в а : ***інформаційні технології, інформаційні системи, керування ресурсами, масштабування, IT-інфраструктура, вертикальне масштабування, горизонтальне масштабування.***

Автори заявляють про відсутність конфлікту інтересів. Спонсори не брали участі в розробленні дослідження; у зборі, аналізі чи інтерпретації даних; у написанні рукопису; в рішенні про публікацію результатів.

The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses or interpretation of data; in the writing of the manuscript; in the decision to publish the results.

ISSN 2788-6603