UDC 004.047, 005.8, 519.85 https://doi.org/10.20535.2786-8729.5.2024/316545

DYNAMIC MATHEMATICAL MODEL FOR RESOURCE MANAGEMENT AND SCHEDULING IN CLOUD COMPUTING ENVIRONMENTS

Vladyslav Kovalenko *
National Technical University of Ukraine
“Igor Sikorsky Kyiv Polytechnic Institute”, Kyiv, Ukraine
https://orcid.org/0009-0001-8723-914X

Olena Zhdanova
National Technical University of Ukraine
“Igor Sikorsky Kyiv Polytechnic Institute”, Kyiv, Ukraine
https://orcid.org/0000-0002-8787-846X

*Corresponding author: vlad.kov@ukr.net

The object of the research is resource management and scheduling in Kubernetes clusters, in particular, data centers. It was determined that in many publications dedicated to optimization models of scheduling for Kubernetes, mathematical models either do not include constraints at all, or only have the constraints determined on the high level only. The purpose of the research is the creation of a dynamic low-level mathematical optimization model for resource management and scheduling in cloud computing environments that utilize Kubernetes. Examples of such environments include the data centers where the customers can rent both dedicated servers and resources of shared hosting servers that are allocated on demand. The suggested model was created using the principles of creation of mathematical models of discrete (combinatorial) optimization, and was given the name “dynamic” because it takes the time parameter into account.

The model receives data about individual servers in the cluster and individual pods that should be launched as an input. The model aims to regulate not only individual assignments of pods to nodes, but also turning on and off the servers. The model has objectives of: minimization of the average number of shared hosting servers running; maximization of the average resource utilization coefficient on such servers; minimization of the number of occasions when the servers are turned on and off; minimization of resource utilization by the pods that are running on shared hosting servers but created by the customers renting the dedicated servers. The model considers resource constraints, among other limitations.

Key words: cloud computing, orchestration, Kubernetes, optimization, mathematical modeling

1. Introduction

Cloud computing technologies are getting chosen by a large number of organizations as a backbone of their IT infrastructure for achieving their business needs [1 – 2]. Therefore, problems related to efficient utilization of cloud providers’ servers arise [1 – 2]. These problems can be solved or mitigated by containerization of applications and usage of optimization strategies, such as usage of efficient scheduling algorithms [1, 3]. An increasingly popular Kubernetes orchestrator is capable of managing containerized applications and includes implementation of several scheduling strategies [1, 3 – 4]. Nevertheless, problems of efficient resource management and scheduling remain among of the most relevant challenges for Kubernetes users, as schedules built by *kube-scheduler* (Kubernetes’ in-built scheduler) are often suboptimal, and occasionally lead to inability to deploy the services [1, 5].

Publications dedicated to the development of new scheduling algorithms for Kubernetes mention the optimization objectives, but not the constraints the system has to operate within. Formulating a specific mathematical model with clear objectives and constraints may be helpful in further development of efficient scheduling algorithms. This article is dedicated to formulating a specific mathematical model for the scheduling problem that was described on a high level in [1]. It includes the objectives of minimizing the total number of nodes (servers) in the cluster, and maximizing the average workload of a single node. The relevance of this publication can be explained by the relevance of a more general problem of resource management in the cloud environments, and the common objective of cloud service providers to minimize energy consumption [1, 6].

2. Literature review and problem statement

The publication [1] contains a detailed literature review dedicated to scheduling problems in Kubernetes. In [1], thirty-two publications, all of them published abroad, proposing a specific solution for Kubernetes were reviewed and classified by the type of methods implemented in the solutions. It was specified which objectives and constraints were considered in the solutions using each of the specific types of methods. The key findings of [1] were as follows:

– the Kubernetes nodes are roughly equivalent to physical or virtual machines, meanwhile Kubernetes pods are abstract entities consisting of containers with applications;

– during the process of scheduling, pods are getting assigned to nodes;

– the commonly used methods can be classified into nine types, of which methods powered by artificial intelligence, heuristic methods, and metaheuristic methods can be considered the most promising;

– maximization of resource utilization coefficient is a common objective;

– few publications (such as [7 – 9]) include mathematical optimization models with explicitly declared constraints, and those constraints that are declared are usually described on a high level.

It is worth pointing out that Kubernetes pods can be created either manually or automatically (via Kubernetes jobs for short-term, one-time tasks, or via Kubernetes deployments for long-running applications) [10, pp. 103, 113–114, 139].

Scheduling in Kubernetes' in-built scheduler (*kube-scheduler*) consists of two stages:

– filtering (determining the set of nodes where the pod can be run);

– scoring (determining the best node where the pod would eventually be scheduled) [11].

In order to reduce energy consumption, currently existing solutions turn off the idle servers and redistribute the resources [12 – 13].

The problem which was identified and which is covered by this article is the absence of a more complete, dynamic (time-flexible), low-level mathematical model. Such a model would take both Kubernetes-specific constraints and business rules of cloud service providers into account. The model would mathematically formalize the process of turning the servers on and off, alongside improving the scoring stage of the scheduling process.

3. The aim and objectives of the study

The aim of the study is to formulate a detailed low-level mathematical optimization model for resource management and scheduling for Kubernetes environments and with the usage of Kubernetes terms which would also be dynamic, and thus would quickly react to the changes in the environment. The model must take the cloud service providers' rules of business operations (such as existence of several customers, and the need to allocate specific pods to dedicated servers or shared hosting servers specifically) into account. The model's newly formulated objectives and constraints could later become a base for the future scheduling solutions for Kubernetes alongside previously existing models' findings.

Formulation of such a new model would enable the development of a solution which would determine exactly when individual nodes should be turned on and off. Despite the commonness of resource utilization optimization objective, turning on and off the individual servers was not considered in previous scheduling models, as per [1]. This can improve Kubernetes' performance specifically for cloud service providers' needs, in comparison to *kube-scheduler*'s performance with default settings.

To achieve the goal of formulating such a detailed model, the task of formulating the mathematical model is set. Based on the typical structure of mathematical optimization models, the following subtasks of this task were identified:

– identifying the exogenous parameters (input data parameters);

– identifying the decision variables (parameters to manage during the course of resource management);

– identifying the objectives;

– identifying the constraints.

4. The study materials related to the environment that is being modeled

Data centers have been known to rent their computing power to the customers via the following paradigms:

– dedicated servers (when the customer rents the server with all its computing resources);

– shared hosting servers (when the data center provides resources according to the customer's needs, but a single physical server's resources may be shared between several customers; such approach to providing resources may be called multi-tenancy) [14 – 15].

Usually, it is the shared hosting paradigm which is associated with cloud computing environments. Organizations' migration to the cloud (to the approach of renting shared hosting servers' resources) often results in decrease in operations costs and improved flexibility in terms of scalability [14]. Nevertheless, cloud providers like Amazon may offer options like Dedicated Instances, which, despite being hosted in the cloud environment, include a physical server fully dedicated to a single customer [16]. Usage of dedicated servers is usually billed per time periods depending on the amount of resources [17]. Usage of shared hosting servers in cloud environment is billed on pay-as-you-go or pay-per-use model [18 – 19].

The mathematical model would overview the operations model of a cloud service provider that can provide both dedicated servers and resources of shared hosting servers to its customers. In this model, the customers using dedicated servers may also temporarily rent the resources of shared hosting servers. More specifically, they may authorize running specific pods on the shared hosting services when they need to upscale their services and do not have the capacity on their dedicated servers.

The mathematical model described in this article is created based on common approaches to creating the mathematical models for discrete (combinatorial) optimization problems in operations research. Some of these most common model-building approaches are described in [20].

The mathematical model will include multiple criteria that may conflict. The solutions implemented on top of the model may therefore use multi-criteria decision-making methods in order to achieve balance between optimizing different criteria.

In real-life problems, some limitations may be soft rather than rigid. In order to decrease the restrictiveness of the model, and to avoid the possibility of the model not producing a feasible solution, the following approach is taken:

– rigid limitations that strictly cannot be violated (such as limitations related to hardware or Kubernetes architecture) are formulated as constraints;

– softer limitations that would not be violated ideally, but can be violated in exceptional circumstances, are formulated as additional objectives of minimization of a numeric representation of undesirable events.

5. Results of investigating and formulating mathematical model components

5.1. Exogenous parameters

Let the cloud service provider's cluster consist of S servers (equivalent to Kubernetes nodes), and let the provider's U customers (users) have P pods to run on the servers within time planning horizon T.

For simplification of the model, the timescale would be considered discrete. In the model, each period t ($t = 1, \dots, T$) would correspond to a brief time window where:

– no servers are getting turned on and off;
– no servers are gaining or losing their status of a dedicated server;
– no new pods are getting launched or shut down.

Since such periods may not be equal in length, each period t would be characterized by its length L_t. The model was named dynamic due to the presence of the time parameter.

Each server s ($s = 1, \dots, S$) would be characterized by:
– c'_s – its CPU size;
– m'_s – its memory size;
– d_{ust} – whether the server is rented as a dedicated one for the specific customer during a certain period:

$$d_{ust} = \begin{cases} 1, \text{if the server } s \text{ is running as a dedicated server} \\ \quad \text{for the customer } u \text{ during the period } t, \\ 0, \text{otherwise,} \end{cases} \qquad u = 1, \dots, U, t = 1, \dots, T \tag{1}$$

Each pod p ($p = 1, \dots, P$) would be characterized by:
– c_p – its CPU requirements;
– m_p – its memory requirements;
– a_{up} – whether it was created by the specific customer:

$$a_{up} = \begin{cases} 1, \text{if the pod } p \text{ was created by the customer } u, \\ 0, \text{otherwise,} \end{cases} \qquad u = 1, \dots, U \tag{2}$$

– b_p – whether the customer has authorized running it on a shared server:

$$b_p = \begin{cases} 1, \text{if the pod } p \text{ was authorized to run on a shared server,} \\ 0, \text{otherwise.} \end{cases} \tag{3}$$

The limitations on the values of exogenous parameters that characterize the problem are as follows:
– a single pod can only be created by a single customer:

$$\sum_{u=1}^{U} a_{up} = 1, \quad p = 1, \dots, P; \tag{4}$$

– at any given period, a single server can only be rented as a dedicated one for either a single customer or for no customer at all:

$$\sum_{u=1}^{U} d_{ust} \le 1, \quad s = 1, \dots, S, t = 1, \dots T. \tag{5}$$

The detailed description of all exogenous parameters would be needed for further formulation of the objectives and the constraints. The scientific novelty of this subtask includes introduction of parameters of relations between pods and servers on one side, and customers on the other side. Such relations were not present in the previous models, and they would help telling dedicated servers and shared hosting servers apart.

5.2. Decision variables

As per the scheduling problem reviewed in the article, the system must determine:
– which servers should be turned on (up and running) in the cluster;
– the details of assignment of individual pods to individual nodes (servers).

These values would be stored in the following variables:
– x_{st} – whether individual servers are turned on in the certain period:

$$x_{st} = \begin{cases} 1, \text{if the server } s \text{ is turned on in the period } t, \\ 0, \text{otherwise,} \end{cases} \qquad s = 1, \dots, S, t = 1, \dots, T; \tag{6}$$

– y_{pst} – whether the pod is running on a given server in the certain period:

$$y_{pst} = \begin{cases} 1, \text{if the pod } p \text{ is running on the} \\ \quad \text{server } s \text{ in the period } t, \\ 0, \text{otherwise,} \end{cases} \qquad p = 1, \dots, P, s = 1, \dots, S, t = 1, \dots, T. \qquad (7)$$

The scientific novelty of this subtask includes introduction of parameters dedicated to turning the servers on and off into the scheduling model specifically.

5.3. Objectives

The first objective considered in the system is minimization of the number of servers turned on in the cluster. Assuming some of the servers in the clusters are rented as specific customers' dedicated servers that the cloud service provider cannot turn off – this objective should be specified as number of shared hosting servers turned on, specifically.

The value $\sum_{u=1}^{U} d_{ust}$ equals 1 if the server s is rented as a dedicated one for any customer, and 0 if it does not. Conversely, the value $(1 - \sum_{u=1}^{U} d_{ust})$ equals 1 if the server is operating as a shared hosting server, and 0 if it does not. Therefore, the total number of shared hosting servers turned on in the cluster in the moment t equals $\sum_{s=1}^{S} x_{st}(1 - \sum_{u=1}^{U} d_{ust})$. Assuming the system needs to minimize the average value of such number across the timescale (rather than in every single period, otherwise the launch of some pods may be infinitely delayed) – the first objective function can be formulated as the time-averaged version of the statement above:

$$\min z^1 = \frac{\sum_{t=1}^{T} L_t [\sum_{s=1}^{S} x_{st}(1 - \sum_{u=1}^{U} d_{ust})]}{\sum_{t=1}^{T} L_t}. \qquad (8)$$

The second and third objectives considered in the system are maximization of the average resource utilization coefficient of the servers turned on in the cluster. The second one would be responsible for optimizing CPU utilization, the third one would be responsible for optimizing memory utilization. For reasons similar to those in the first objective, only shared hosting servers would be counted in these objectives.

The CPU utilization coefficient of a single server s in the moment t equals $\sum_{p=1}^{P} c_p y_{pst}$. The average CPU utilization coefficient across all shared hosting servers equals the sum of such coefficients, divided by the total number of shared hosting servers: $\frac{\sum_{s=1}^{S}[(1-\sum_{u=1}^{U} d_{ust}) \sum_{p=1}^{P} c_p y_{pst}]}{\sum_{s=1}^{S} x_{st}(1-\sum_{u=1}^{U} d_{ust})}$. For reasons similar to those in the first objective, the second objective can be formulated as the time-averaged version of the statement above:

$$\max z^2 = \frac{\sum_{t=1}^{T} L_t [\frac{\sum_{s=1}^{S}[(1 - \sum_{u=1}^{U} d_{ust}) \sum_{p=1}^{P} c_p y_{pst}]}{\sum_{s=1}^{S} x_{st}(1 - \sum_{u=1}^{U} d_{ust})}]}{\sum_{t=1}^{T} L_t}. \qquad (9)$$

The third objective is similar to the second objective, with the exception of references to memory utilization coefficients rather than CPU utilization coefficients:

$$\max z^3 = \frac{\sum_{t=1}^{T} L_t [\frac{\sum_{s=1}^{S}[(1 - \sum_{u=1}^{U} d_{ust}) \sum_{p=1}^{P} m_p y_{pst}]}{\sum_{s=1}^{S} x_{st}(1 - \sum_{u=1}^{U} d_{ust})}]}{\sum_{t=1}^{T} L_t}. \qquad (10)$$

Turning the servers on and off too often may increase the speed of disks' wear and tear [13]. Therefore, it may be desirable to plan the workload in a way that minimizes the number of occasions on which individual servers should be turned on and off. For example, the pods with low-priority jobs could be scheduled when the total workload is low. This can be achieved by introducing **the fourth objective**, which would be described as minimization of the total number of occasions when the server's status ("turned on or off") is changed. $|x_{s(t+1)} - x_{st}|$ equals 1 when the server s had a status change between the periods t and $(t+1)$. The fourth objective can be formulated as the total number of status changes across all the servers and across the timescale:

$$\min z^4 = \sum_{t=1}^{T-1} \sum_{s=1}^{S} |x_{s(t+1)} - x_{st}|. \tag{11}$$

With the second and third objectives being designed with maximization of average workload per server, pods which were authorized for running on shared hosting servers may be allocated on such servers. This may lead to overcharging complaints from the customers, especially in case of existence of extra capacity on the dedicated servers. This can be prevented by introducing **the fifth and sixth objectives**. They would be described as minimization of the total resource consumption (CPU and memory) by the pods which are run on the shared hosting servers, but were created by the customers who rent dedicated servers, across the timescale. A decision was made to include such resource consumptions into the objectives with the weights equal to number of dedicated servers the customer rents. This was done in order to:

– not introduce an extra Boolean parameter containing whether the customer rents at least one dedicated server or not;

– to emphasize the increasing importance of assigning pods to dedicated servers with the increase of number of dedicated servers rented by the customer.

Similarly to the part of the first objective, the value $(1 - \sum_{v=1}^{U} d_{vst})$ represents whether the server is operating as a shared hosting server (e.g. not dedicated to a single customer, not necessarily the current one). The value $\sum_{q=1}^{S} d_{uqt}$ equals the total number of dedicated servers rented by the same customer. The fifth objective can be formulated as:

$$\min z^5 = \sum_{t=1}^{T} L_t \sum_{u=1}^{U} \sum_{s=1}^{S} \sum_{p=1}^{P} \left[\left(1 - \sum_{v=1}^{U} d_{vst}\right) \left(\sum_{q=1}^{S} d_{uqt}\right) c_p y_{pst} \right]. \tag{12}$$

The sixth objective is similar to the fifth objective, with the exception of references to memory utilization coefficients rather than CPU utilization coefficients:

$$\min z^6 = \sum_{t=1}^{T} L_t \sum_{u=1}^{U} \sum_{s=1}^{S} \sum_{p=1}^{P} [\left(1 - \sum_{v=1}^{U} d_{vst}\right) \left(\sum_{q=1}^{S} d_{uqt}\right) m_p y_{pst}]. \tag{13}$$

It can be said about the objectives (12) – (13) that they represent the soft limitations, as stated in chapter 4. Ideally, the pod created by a customer renting a dedicated server would always be assigned to a dedicated server, otherwise, the objective value would increase. However, in exceptional circumstances (when no other option is feasible), the system shall allow assigning such a pod to a shared server.

The scientific novelty of this subtask includes formulating new types of objectives alongside the types previously discovered in [1]. This stems from the need to manage the pods created by both types of customers: those who rent a dedicated server, and those who use the resources of shared hosting servers.

5.4. Constraints

The only type of constraints identified in all models in [7 – 9], as per [1], are resource constraints (in terms of both CPU and memory).

The first set of constraints proposed for this model would be resource constraints per server. At any given time, the sums of CPU or memory requirements of all pods running on a specific server should not exceed the total CPU or memory resources available on the server. The first set of constraints is stated in (14) for CPU resources and (15) for memory resources.

$$\sum_{p=1}^{P} c_p y_{pst} \le c'_s, \quad s = 1,\dots,S, t = 1,\dots,T. \tag{14}$$

$$\sum_{p=1}^{P} m_p y_{pst} \le m'_s, \quad s = 1,\dots,S, t = 1,\dots,T. \tag{15}$$

The second set of constraints proposed for this model would be resource constraints per customer. At any given time, the sums of CPU or memory requirements of all pods created by a specific customer:

– should not exceed the total CPU or memory resources of this customer's dedicated servers, if this customer rents dedicated servers and did not authorize running any of their pods on a shared hosting server;

– is not limited by anything, if the customer authorized running any of their pods on a shared hosting server (this includes the situation where the customer does not rent any dedicated servers).

The value $\sum_{s=1}^{S} y_{pst}$ equals 1 if the pod p is running on some node (server) in the period t, and 0 otherwise. The value $\sum_{p=1}^{P} a_{up} b_p$ equals 0 if the customer u did not authorize running any of their pods on a shared hosting server, and is larger than 0 otherwise. The value M would represent the number which is several orders of magnitudes larger than any other number in the model, to be a replacement of "infinity." The second set of constraints is stated in (16) for CPU resources and (17) for memory resources.

$$\sum_{p=1}^{P} a_{up} c_p (\sum_{s=1}^{S} y_{pst}) \le \sum_{s=1}^{S} d_{ust} c'_s + M \sum_{p=1}^{P} a_{up} b_p, \quad u = 1,\dots,U, t = 1,\dots,T. \tag{16}$$

$$\sum_{p=1}^{P} a_{up} m_p (\sum_{s=1}^{S} y_{pst}) \le \sum_{s=1}^{S} d_{ust} m'_s + M \sum_{p=1}^{P} a_{up} b_p, \quad u = 1,\dots,U, t = 1,\dots,T. \tag{17}$$

A decision was made to omit other types of constraints identified in [1] from the current model due to its low level. For example, this model is working with individual pods rather than applications. Additionally, this model does not take relationship between individual pods (such as affinity rules and anti-affinity rules) into account, because it is assumed that this model works in addition to (not as a replacement of) existing scheduling solutions' models. Instead, this model's aim is to include constraints related to the business model of cloud service providers.

The third set of constraints proposed for this model would prohibit running the pods on shared hosting servers if they were not authorized for this:

$$\sum_{s=1}^{S} [(1 - \sum_{u=1}^{U} d_{ust}) y_{pst}] \le b_p, \quad p = 1,\dots,P, t = 1,\dots,T. \tag{18}$$

The fourth set of constraints proposed for this model would prohibit running the pods on dedicated servers if they were created by any customer except the one who rents the server. When the server s is running as a dedicated one, but not for the customer u in the period t specifically, then $d_{ust} = 0$, and $\sum_{q=1}^{S} d_{uqt} = 1$. Therefore, the value $1 + d_{ust} - \sum_{q=1}^{S} d_{uqt}$ equals 1 in such situations, and 0 otherwise. The fourth set of constraints can be formulated as:

$$\sum_{p=1}^{P} a_{up} y_{pst} \le M \left(1 + d_{ust} - \sum_{q=1}^{S} d_{uqt} \right), \quad u = 1,\dots,U, s = 1,\dots,S, t = 1,\dots,T. \tag{19}$$

The fifth set of constraints would regulate that the pod can have its status changed ("running or not running") no more than two times. Throughout the timescale, a pod can be launched, stopped, or both, but it cannot be launched or stopped more than once, and it cannot be moved to a different node directly [21]. $|\sum_{s=1}^{S} y_{ps(t+1)} - \sum_{s=1}^{S} y_{pst}|$ equals 1 when the pod p had a status change between the periods t and $(t+1)$. The fifth set of constraints can be formulated as:

$$\sum_{t=1}^{T-1} \left| \sum_{s=1}^{S} y_{ps(t+1)} - \sum_{s=1}^{S} y_{pst} \right| \le 2, \quad p = 1, \dots, P. \tag{20}$$

The sixth set of constraints would regulate that the same pod cannot be running on two servers simultaneously (assuming different pod replicas as treated as different pods):

$$\sum_{s=1}^{S} y_{pst} \le 1, \quad p = 1, \dots, P, t = 1, \dots, T. \tag{21}$$

The seventh set of constraints would regulate that no pods can be run on a server which has been turned off. This can be described as "the number of pods running on the server is limited by 0 for a server which is turned off, and not limited by anything for a server which is turned on:"

$$\sum_{p=1}^{P} y_{pst} \le M x_{st}, \quad s = 1, \dots, S, t = 1, \dots, T. \tag{22}$$

The scientific novelty of this subtask includes formulating new types of constraints that take the features of the environment (such as two types of servers, dedicated and shared hosting) into account. Such constraints also describe the limitations to the possible combinations of assignments in a more detailed way than high-level constraints identified in [1].

6. Analysis of the obtained results of mathematical model creation

The dynamic low-level scheduling model for Kubernetes that takes data center's rules of business operation into account is provided in the formulae (1) – (22).

The proposed model successfully covers both the cloud service provider's objectives of utilizing the servers' resources efficiently, and the customers' objectives of not overpaying for the provided services. The proposed model focuses more on determining the exact number of nodes in the cluster that should be running.

However, the model may be less focused on the assignment of each individual pod to a specific node. To overcome this challenge, this model may be accompanied by the second model, that can be called "static model." Such a static model could be responsible for determining individual assignments during the time intervals when the number of nodes that are running in the cluster is not changing.

The dynamic model and the static model can comprise a two-tier decision-making system. In such a system, the dynamic model could be responsible for making global decisions, and the static model could be responsible for making quicker operational decisions.

Further scheduling solution development and related experiments are required to determine the effectiveness of the mathematical model for real-world use.

Conclusion

The dynamic low-level mathematical model was formulated to solve the problem of resource management for cloud computing environments, such as data centers. The model has accounted for new potential objectives and constraints which are unique to the environments containing both the dedicated servers (rented by specific customers) and shared (multi-tenant) hosting servers.

The model's exogenous parameters include the attributes of individual servers and pods, both from technical and business perspective.

The model's decision variables include whether each individual server is turned on or off, and whether each individual pod is running on a particular node (server).

The model's objectives include: minimization of the average number of shared hosting servers turned on; maximization of average resource usage per such server; minimization of the number of occasions when servers get turned on and off; minimization of the resource usage by the pods running on shared hosting servers but created by the customers renting the dedicated servers. Such objectives are meant to increase the environment's economic efficiency by decreasing the amount of idle computing resources and therefore reducing overall power consumption, while also keeping customer satisfaction high.

The model's constraints include: resource constraints; limitations on individual pods' ability to run on individual servers; technical constraints such as inability of a pod to be launched or stopped more than once, to be moved to a different node, or to run on more than one server simultaneously.

References

[1] V. V. Kovalenko, and M. M. Bukasov, "Scheduling Methods and Models for Kubernetes Orchestrator," *Visnyk of Vinnytsia Politechnical Institute*, vol. 175, no. 4, pp. 86–94, 2024, https://doi.org/10.31649/1997-9266-2024-175-4-86-94.

[2] L. Golightly, V. Chang, Q. A. Xu, X. Gao, and B. S. Liu, "Adoption of cloud computing as innovation in the organization," *International Journal of Engineering Business Management*, vol. 14, Jan. 2022, https://doi.org/10.1177/18479790221093992.

[3] K. Senjab, S. Abbas, N. Ahmed, and A. u. R. Khan, "A survey of Kubernetes scheduling algorithms," *Journal of Cloud Computing*, vol. 12, no. 1, p. 87, Jun. 2023, https://doi.org/10.1186/s13677-023-00471-1.

[4] "Overview." Kubernetes. [Online]. Available: https://kubernetes.io/docs/concepts/overview/

[5] T. Lebesbye, J. Mauro, G. Turin, and I. C. Yu, "Boreas – A Service Scheduler for Optimal Kubernetes Deployment," in *Service-Oriented Computing*. Cham: Springer Int. Publishing, 2021, pp. 221–237, https://doi.org/10.1007/978-3-030-91431-8_14.

[6] P. Townend, S. Clement, D. Burdett, R. Yang, J. Shaw, B. Slater, and J. Xu, "Invited Paper: Improving Data Center Efficiency Through Holistic Scheduling In Kubernetes," in *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, San Francisco East Bay, CA, USA, Apr. 4–9, 2019. IEEE, 2019, pp. 156–15610, https://doi.org/10.1109/sose.2019.00030.

[7] Y. Qiao, S. Shen, C. Zhang, W. Wang, T. Qiu, and X. Wang, "EdgeOptimizer: A programmable containerized scheduler of time-critical tasks in Kubernetes-based edge-cloud clusters", *Future Generation Computer Systems*, vol. 156, pp. 221–230, Jul. 2024, https://doi.org/10.1016/j.future.2024.03.007.

[8] M. Lin, J. Xi, W. Bai, and J. Wu, "Ant Colony Algorithm for Multi-Objective Optimization of Container-Based Microservice Scheduling in Cloud," *IEEE Access*, vol. 7, pp. 83088–83100, 2019, https://doi.org/10.1109/access.2019.2924414.

[9] J. Santos, C. Wang, T. Wauters, and F. D. Turck, "Diktyo: Network-Aware Scheduling in Container-based Clouds," *IEEE Transactions on Network and Service Management*, p. 1, 2023, https://doi.org/10.1109/tnsm.2023.3271415.

[10] B. Burns, J. Beda, and K. Hightower, *Kubernetes: Up and Running. Dive into the Future of Infrastructure*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, Inc., 2019.

[11] S. Huaxin, X. Gu, K. Ping, and H. Hongyu, "An Improved Kubernetes Scheduling Algorithm for Deep Learning Platform," in *2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, Chengdu, China, Dec. 18–20, 2020. IEEE, 2020, pp. 113–116, https://doi.org/10.1109/iccwamtip51612.2020.9317317.

[12] S. Telenyk, O. Rolik, E. Zharikov, and Y. Serdiuk, "Energy efficient data center resources management using beam search algorithm," *Czasopismo Techniczne*, vol. 4, pp. 127–138, 2018, https://doi.org/10.4467/2353737xct.18.060.8372.

[13] M. Callau-Zori, L. Arantes, J. Sopena, and P. Sens, "MERCi-MIsS: Should I Turn off My Servers?" Springer Int. Publishing, 2015, pp. 16–29, https://doi.org/10.1007/978-3-319-19129-4_2.

[14] F. Abidi and V. Singh, “Cloud servers vs. dedicated servers — A survey,” in *2013 IEEE International Conference in MOOC, Innovation and Technology in Education (MITE)*, Jaipur, India, Dec. 20–22, 2013. IEEE, 2013, pp. 1–5, https://doi.org/10.1109/mite.2013.6756294.

[15] P.-J. Maenhaut, H. Moens, V. Ongenae, and F. De Turck, “Migrating legacy software to the cloud: approach and verification by means of two medical software use cases,” *Software: Practice and Experience*, vol. 46, no. 1, pp. 31–54, Jan. 2016, https://doi.org/10.1002/spe.2320.

[16] “Amazon EC2 Dedicated Instances.” Amazon Elastic Compute Cloud. [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/dedicated-instance.html.

[17] G. Cornetta, J. Mateos, A. Touhafi, and G.-M. Muntean, “Design, simulation and testing of a cloud platform for sharing digital fabrication resources for education,” *Journal of Cloud Computing*, vol. 8, no. 1, p. 12, Aug. 2019, https://doi.org/10.1186/s13677-019-0135-x.

[18] N. K. Sehgal, P. C. P. Bhatt, and J. M. Acken, “Cost and Billing Practices in Cloud,” in *Cloud Computing with Security and Scalability.* Cham: Springer Int. Publishing, 2022, pp. 177–195, https://doi.org/10.1007/978-3-031-07242-0_10.

[19] D. Lowe, and B. Galhotra, “An Overview of Pricing Models for Using Cloud Services with analysis on Pay-Per-Use Model,” *International Journal of Engineering & Technology*, vol. 7, no. 3.12, pp. 248–254, Jul. 2018, https://doi.org/10.14419/ijet.v7i3.12.16035.

[20] O. H. Zhdanova, V. D. Popenko, and M. O. Sperkach, *Doslidzhennia operatsii. Vstup do dyskretnoho prohramuvannia. Praktykum,* (in Ukrainian). Kyiv: NTUU Igor Sikorsky Kyiv Polytechnic Institute, 2019. [Online]. Available: https://ela.kpi.ua/handle/123456789/32225.

[21] C. Centofanti, W. Tiberti, A. Marotta, F. Graziosi, and D. Cassioli, “Taming latency at the edge: A user-aware service placement approach,” *Computer Networks*, p. 110444, Apr. 2024, https://doi.org/10.1016/j.comnet.2024.110444.

УДК 004.047, 005.8, 519.85

ДИНАМІЧНА МАТЕМАТИЧНА МОДЕЛЬ ДЛЯ УПРАВЛІННЯ РЕСУРСАМИ ТА СКЛАДАННЯ РОЗКЛАДІВ У СЕРЕДОВИЩАХ ХМАРНИХ ОБЧИСЛЕНЬ

Владислав Коваленко
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна
https://orcid.org/0009-0001-8723-914X

Олена Жданова
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна
https://orcid.org/0000-0002-8787-846X

Об’єктом дослідження є управління ресурсами та складання розкладів у кластерах Kubernetes, зокрема у центрах обробки даних. Було визначено, що у багатьох публікаціях, присвячених оптимізаційним моделям складання розкладів для Kubernetes, математичні моделі або не містять обмежень взагалі, або мають обмеження, визначені лише на високому рівні. Метою дослідження є побудова динамічної низькорівневої математичної оптимізаційної моделі для управління ресурсами та складання розкладів у середовищах хмарних обчислень, що використовують Kubernetes. До прикладів таких середовищ відносяться центри обробки даних, де клієнти мають змогу орендувати як виділені сервери, так і ресурси серверів спільного хостингу, що виділяються на вимогу. Запропонована модель була побудована із використанням принципів побудови математичних моделей дискретної (комбінаторної) оптимізації та отримала назву динамічної, оскільки враховує параметр часу.

Модель приймає на вхід дані про окремі сервери кластера та про окремі Pod’и, що мають бути запущеними. Модель має на меті регулювати не лише окремими призначеннями Pod’ів на вузли, але і вмиканням і вимиканням серверів. Модель має цільові функції: мінімізації середньої кількості запущених серверів спільного хостингу; максимізації середнього коефіцієнту використання ресурсів на таких серверах; мінімізації кількості увімкнень та вимикань серверів; мінімізації використання ресурсів Pod’ами, що запущені на серверах спільного хостингу, але створені замовниками, що орендують виділені сервери. Модель приймає до уваги: обмеження на обсяги ресурсів; обмеження на здатність окремих Pod’ів бути запущеними на окремих вузлах; технічні обмеження, як-от неможливість запускати або зупиняти Pod більш ніж один раз, переміщувати Pod з вузла на вузол, чи запускати Pod більш ніж на одному вузлі.

Key words: хмарні обчислення, оркестрація, Kubernetes, оптимізація, математичне моделювання.