УДК 004.75:004.4:519.8 DOI https://doi.org/10.35546/kntu2078-4481.2025.3.2.67

О. М. ШУШУРА
доктор технічних наук, доцент,
професор кафедри цифрових технологій в енергетиці
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
ORCID: 0000-0003-3200-720X

Д. А. ІГНАТОВ
аспірант кафедри цифрових технологій в енергетиці
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
ORCID: 0009-0008-2864-8526

МОДЕЛЮВАННЯ ПРИЙНЯТТЯ РІШЕНЬ З УПРАВЛІННЯ РЕСУРСАМИ МІКРОСЕРВІСІВ ТА ВИБОРУ ВІРТУАЛЬНИХ МАШИН У СЕРЕДОВИЩІ KUBERNETES

Сучасні хмарні системи, побудовані з використанням мікросервісних архітектур, стикаються з критичною проблемою ефективного управління ресурсами в умовах динамічного навантаження. Стандартні реактивні механізми оркестратора Kubernetes, такі як Horizontal Pod Autoscaler, часто виявляються недостатніми через затримки у реакції, відсутність врахування майбутніх трендів навантаження та інерційність процесів запуску нових реплік мікросервісів і віртуальних машин. Це призводить до неефективного використання ресурсів, коливань продуктивності (flapping) та зайвих операційних витрат. У відповідь на ці виклики в статті запропоновано комплексну математичну модель для оптимального проактивного управління ресурсами на основі прогнозування навантаження, динамічне масштабування мікросервісів із врахуванням затримок та оптимальне розміщення подів на віртуальних машинах.

Метою даної роботи є розробка математичної оптимізаційної моделі для автоматичного управління ресурсами мікросервісів та динамічного вибору віртуальних машин. Формалізовано предметну область шляхом визначення ключових множин: класів віртуальних машин, активних екземплярів віртуальних машин, типів мікросервісів та їхніх реплік (подів). На основі історичних даних метрик CPU та оперативної пам'яті виконується прогнозування навантаження на горизонті планування і для кожного типу мікросервіса розраховується бажана кількість реплік, необхідна для обробки очікуваного навантаження. Сформовано та формалізовано цільову функцію і обмеження задачі оптимального управління ресурсами мікросервісів та вибору віртуальних машин, спираючись на сукупність прийнятих припущень.

Запропонований підхід забезпечує підвищення відмовостійкості та продуктивності мікросервісних додатків при одночасному зниженні операційних витрат за рахунок усунення надмірного резервування та консолідації навантаження на найбільш економічно ефективних типах віртуальних машин. Результати роботи становлять теоретичну основу для подальших досліджень та практичної реалізації системи інтелектуального оркестрування для Kubernetes.

Ключові слова: *мікросервіси, Kubernetes, управління ресурсами, прогнозування навантаження, автомасштабування, оптимізація, віртуальні машини, інформаційні технології, інформаційні системи.*

O. M. SHUSHURA
Doctor of Technical Sciences, Associate Professor,
Professor at the Department of Digital Technologies in Energy
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"
ORCID: 0000-0003-3200-720X

D. A. IHNATOV
Postgraduate Student at the Department of Digital Technologies in Energy
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"
ORCID: 0009-0008-2864-8526

MODELING DECISION-MAKING FOR MICROSERVICE RESOURCE MANAGEMENT AND VIRTUAL MACHINE SELECTION IN A KUBERNETES ENVIRONMENT

Modern cloud systems built on microservice architectures face a critical challenge of efficient resource management under dynamic workloads. Standard reactive mechanisms of the Kubernetes orchestrator, such as the Horizontal Pod Autoscaler, often prove insufficient due to delayed responses, lack of consideration for future load trends, and the inertia of launching new microservice replicas and virtual machines. This results in inefficient resource utilization, performance fluctuations (flapping), and excessive operational costs.

In response to these challenges, the paper proposes a comprehensive mathematical model for optimal proactive resource management based on load forecasting, dynamic scaling of microservices with consideration of delays, and optimal pod placement on virtual machines.

The aim of this work is to develop a mathematical optimization model for automated microservice resource management and dynamic virtual machine selection. The subject domain is formalized through the definition of key sets: classes of virtual machines, active instances of virtual machines, types of microservices, and their replicas (pods). Based on historical CPU and memory metrics, load forecasting is performed on the planning horizon, and for each type of microservice, the desired number of replicas required to handle the expected load is calculated. The objective function and constraints of the problem of optimal resource management and virtual machine selection are constructed and formalized, relying on a set of accepted assumptions.

The proposed approach increases the resilience and performance of microservice applications while simultaneously reducing operational costs by eliminating excessive overprovisioning and consolidating workloads on the most cost-effective types of virtual machines. The results provide a theoretical foundation for further research and practical implementation of an intelligent orchestration system for Kubernetes.

Key words: *microservices, Kubernetes, resource management, load forecasting, autoscaling, optimization, virtual machines, information technology, information systems.*

Постановка проблеми

Однією з ключових задач сучасної IT-інфраструктури є забезпечення ефективного та відмовостійкого функціонування мікросервісних систем у хмарних середовищах за умов динамічно змінного навантаження. У разі різких змін навантаження, інформаційна система має швидко масштабуватись, щоб уникнути втрати продуктивності або відмови в обслуговуванні, але водночас небажано, аби це призводило до надмірних витрат на обчислювальні ресурси.

Ця проблема ускладнюється тим, що сучасні оркестратори, такі як Kubernetes, часто використовують реактивні механізми масштабування (наприклад, Horizontal Pod Autoscaler), які приймають рішення на основі минулих чи поточних даних, не враховуючи майбутніх трендів навантаження та інерційності самої системи. Запуск нових реплік мікросервісів та віртуальних машин потребує часу, що призводить до затримок у реакції на навантаження та неефективного використання ресурсів.

Крім того, вимоги до економічної ефективності та надійності сервісів обмежують можливості використання простих рішень, таких як надлишкове резервування ресурсів. Це обумовлює актуальність пошуку інтелектуальних та проактивних підходів до управління ресурсами. Вирішення цієї задачі є критично важливим для підвищення ефективності, надійності та зниження вартості експлуатації мікросервісних архітектур.

Аналіз останніх досліджень і публікацій

Стандартний підхід Kubernetes до управління ресурсами мікросервісів базується на Horizontal Pod Autoscaler (HPA), який використовує прості метрики (CPU, memory) і порогові значення для прийняття рішень про масштабування [1]. Проте цей підхід має суттєві недоліки. По-перше, HPA реагує на зміни лише після їх виникнення, що призводить до затримок у реакції [2]. По-друге, система не враховує час запуску нових реплік, що може спричиняти тимчасове зниження продуктивності. Крім того, HPA схильний до такого явища як flapping – частого додавання та видалення подів через чутливість до коротких сплесків навантаження.

Для подолання обмежень зазначеного вище реактивного підходу було запропоновано використовувати методи прогнозуючого масштабування, що використовують машинне навчання для передбачення майбутнього навантаження [3,4]. Також при порівнянні ARIMA, Prophet та LSTM-мереж виявилося, що найбільш вдалим є вибір на користь ARIMA [5]. Але незважаючи на переваги, дослідження зосереджуються лише на розрахунку бажаної кількості реплік, не розглядаючи питання їхнього розміщення на віртуальних машинах, що створює розрив між прогнозуванням потреб та реальним забезпеченням ресурсів.

Окрім цього, інше дослідження [6] визнає необхідність врахування інерційності системи автомасштабування. У роботі було розроблено алгоритм DACS (Delay-Aware Container Scheduling), який покликаний враховувати затримки обробки та мережі під час запуску подів, хоча і не розглядає час, якого потребує под при старті.

Вбудовані механізми розміщення подів у Kubernetes реалізуються через kube-scheduler та плагін NodeResourcesFit. Система підтримує різні стратегії розподілу ресурсів: LeastAllocated (за замовчуванням) прагне рівномірно розподілити навантаження між вузлами, тоді як MostAllocated та RequestedToCapacityRatio реалізують власну стратегію bin packing для максимізації утилізації [7].

У той же час було продемонстровано, що інтегровані підходи для розподілення подів можуть досягати до 58 % зменшення витрат порівняно зі стандартним методом [8]. Даний підхід поєднує best-fit bin packing для початкового розміщення, динамічне перепланування для консолідації робочих навантажень та автомасштабування для адаптації до змін попиту.

У контексті математичного моделювання управління ресурсами в Kubernetes можна відзначити дослідження, яке запропонувало динамічну модель, що враховує часові параметри та низькорівневі обмеження оптимізації [9]. Модель спрямована на регулювання не лише призначенням подів до віртуальних машин, але й керуванням увімкненням та вимкненням машин з метою мінімізації середньої кількості працюючих машин та максимізації коефіцієнта утилізації ресурсів.

Прогнозування навантаження також розглядалось у дослідженні [10] для динамічного вибору і міграції віртуальних машин у хмарних середовищах. Хоча при прогнозуванні використовувались агреговані метрики утилізації, що не дозволяє використовувати цей підхід при необхідності більш гнучких налаштувань, які можуть бути надзвичайно важливими у контексті мікросервісної архітектури.

Таким чином, актуальною є розробка комплексної моделі, яка б поєднувала прогнозування навантаження, динамічне масштабування мікросервісів з урахуванням затримок, оптимальне розміщення на віртуальних машинах та динамічне керування життєвим циклом цих машин для мінімізації сумарних витрат.

Формулювання мети дослідження

Метою даної роботи є розробка математичної оптимізаційної моделі для автоматичного управління ресурсами мікросервісів та динамічного вибору віртуальних машин. Розроблена модель має дозволити перейти від стандартного реактивного масштабування до проактивного, що забезпечить кращу продуктивність мікросервісної системи при динамічному навантаженні. Особливу увагу слід приділити вирішенню задачі пакування, оскільки це призведе до зниження операційних витрат завдяки більш ефективному замовленню віртуальних машин та використанню їхніх ресурсів.

Для досягнення поставленої мети необхідно вирішити задачі:

– описати основні характеристики предметної області з використанням математичного апарату теорії множин;

– сформувати та формалізувати цільову функцію та обмеження задачі оптимального управління ресурсами мікросервісів і вибору віртуальних машин, спираючись на сукупність прийнятих припущень.

Викладення основного матеріалу дослідження

Формалізація характеристик предметної області включає формування переліку основних об’єктів предметної області та представлення їх характеристик з використанням теорії множин.

Основними об’єктами предметної області в задачі управління ресурсами мікросервісів та вибору віртуальних машин визначено:

– віртуальні машини, які відносяться до певного типу, що клауд провайдер надає в оренду з певними обчислювальними ресурсами у вигляді CPU та пам’яті;

– міскросервіси (поди), які належать до певного типу мікросервісів та розміщені на деякій активній віртуальній машині.

Враховуючи, що зазначені об’єкти розглядаються в нестаціонарних умовах, під час формалізації їх характеристик будемо використовувати дискретну рівномірно розподілену з кроком Δt сітку часу $T = \{t_0, t_1, \ldots, t_N \,|\, t_i = i \cdot \Delta t\}$ де t_0 початковою точкою. Далі наведена формалізація характеристик зазначених об’єктів.

Під час запуску віртуальної машини розробник чи адміністратор повинен обрати певний тип віртуальної машини з поміж списку, який пропонується клауд провайдером. Різні типи розрізняються між собою як ресурсними характеристиками, так і вартістю використання. Відповідно, формується множина класів віртуальних машин $c_j \in C$ де j номер класу. Кожен тип віртуальної машини c_j характеризується кортежем (cpu_j, mem_j, $cost_j$) де cpu_j – кількість віртуальних CPU (vCPU); mem_j – об’єм оперативної пам’яті; $cost_j$ – вартість використання однієї віртуальної машини типу c_j за один крок Δt. Активні протягом часу спостереження віртуальні машини формують множину V, кожен елемент якої описується кортежем $(c_j, t_k^{start}, t_k^{stop})$. Кожен екземпляр v_k наслідує всі ресурсні характеристики і вартість свого класу $c_j \in C$, а також має моменти запуску і зупинки відповідно – t_k^{start}, t_k^{stop}. Зазначені множини дозволяють обчислювальні ресурси, які клієнт отримує від клауд-провайдера для запуску мікросервісів.

Мікросервісна архітектура дозволяє краще справлятися з динамічним навантаженням при використанні додатків: якщо навантаження зростає лише на певний мікросервіс, то можна збільшити кількість реплік (подів) лише конкретного мікросервіса, а не горизонтально масштабувати великий моноліт. Завдяки такому більш точному масштабуванні можна скоротити операційні витрати. Нехай S – множина всіх типів мікросервісів, кожен елемент якої характеризується кортежем $(cpu_l, mem_l, t_k^{start}, t_k^{stop}, n_l^{\min}, n_l^{\max})$, де cpu_l, mem_l – ресурси, необхідні для однієї репліки даного типу; t_k^{start}, t_k^{stop} – час запуску та зупинки репліки; $n_l^{\min}, n_l^{\max}$ – мінімальна та максимальна

кількість реплік, що мають бути активними протягом усього часу. Позначимо $n_{l,i}$ як кількість реплік мікросервісу s_l – момент часу t_i, яка у будь-який момент часу має задовольняти умові $n_l^{\min} \le n_{l,i} \le n_l^{\max}$. Запущені в роботу за весь час спостереження репліки (поди) мікросервісів формують множину P, кожен елемент якої характеризується кортежем $(v_k, s_l, t_p^{start}, t_p^{stop})$, де v_k – конкретна віртуальна машина, на якій розміщений под; s_l – клас мікросервіса, до якого належить под; t_p^{start}, t_p^{stop} – моменти запуску та зупинки пода. Кожен под p_p наслідує всі характеристики свого типу s_l. Зазначимо, що розміщення пода на активній віртуальній машині означає використання відповідних ресурсів даної віртуальної машини. Окрім цього, слід зазначити, що у процесі роботи репліки мікросервісів можуть переміщуватися між активними віртуальними машинами: под зупиняється на одній машині та запускається на іншій. Тому для спостереженням даного явища міграції введемо бінарну змінну $x_{p,k,i} \in \{0, 1\}$, яка характеризує, чи розміщений под p_p на віртуальній машині v_k у момент часу t_i. Дані множини дозволяють поєднати наявні ресурси, придбані у вигляді віртуальних машин та їхніх обчислювальних ресурсів, та власне використання цих ресурсів компонентами інфраструктури додатків.

Розглянемо використання ресурсів детальніше: кожен под p_p кожен момент часу використовує певний обсяг ресурсів CPU та оперативної пам'яті. Відповідно, введемо множину метрик подів $PM = \{(cpu_{p,i}, mem_{p,i}) \mid p \in P, t_i \in T\}$, до якої входять спостереження кількості спожитих ресурсів кожною з реплік мікросервісів. Оскільки для прийняття рішень про масштабування використовуються агреговані по мікросервісам метрики, введемо також і множину метрик мікросервісів $SM = \{(cpu_{l,i}, mem_{l,i}) \mid s_l \in S, t_i \in T\}$, де $cpu_{l,i} = \sum_{p \in P \mid s_p = s_l} cpu_{p,i}$ $mem_{l,i} = \sum_{p \in P \mid s_p = s_l} mem_{p,i}$. Дані множини дозволяють спостерігати за поточним навантаженням у вигляді обсягу спожитих ресурсів.

Для вирішення задачі оптимального управління ресурсами мікросервісів та вибору віртуальних машин, необхідно на основі формалізованих множин об'єктів предметної області та їх характеристик розробити її математичну постановку.

Формальна постановка задачі містить обмеження та цільову функцію задачі оптимального управління ресурсами мікросервісів і вибору віртуальних машин, сформованих на основі набору припущень.

Зазначена задача оптимального управління формується на основі прогнозування навантаження по використанню мікросервісів на деякому горизонті прогнозування $\tau_i = \{t_{i+1}, t_{i+2}, \ldots, t_{i+H}\} \subset T$, де H – глибина горизонту. На основі історичних даних метрик SM формується така функція f, що $\forall t_i \in T, \forall s_l \in S$:

$$\widehat{SM}_l(\tau) = (\widehat{SM}_{l,i+1}, \widehat{SM}_{l,i+2}, \ldots, \widehat{SM}_{l,i+H}) = f(SM_{l,i}, SM_{l,i-1}, \ldots, SM_{l,0}), \tag{1}$$

де $SM_{l,i}$ – значення метрик для мікросервісу s_l момент t_i; $\widehat{SM}_{l,i+1}$ – прогноз майбутнього значення для мікросервісу s_l момент t_{i+1}.

Далі визначається кількість реплік мікросервісів, яка необхідна для обробки прогнозованого навантаження на горизонті τ, використовуючи значення прогнозу $\widehat{SM}_l(\tau)$:

$$\widehat{SM}_l(\tau) n_l^*(\tau) = \left\{ \max\left(\frac{\widehat{cpu}_{l,i+1}}{\alpha_l \cdot cpu_l}, \frac{\widehat{mem}_{l,i+1}}{\alpha_l \cdot mem_l} \right) \cdot \sigma_l, \ldots, \max\left(\frac{\widehat{cpu}_{l,i+H}}{\alpha_l \cdot cpu_l}, \frac{\widehat{mem}_{l,i+H}}{\alpha_l \cdot mem_l} \right) \cdot \sigma_l \right\}, \tag{2}$$

де $\alpha_l \in [0, 1]$ – коефіцієнт утилізації, який визначає цільовий рівень використання ресурсів однієї репліки мікросервіса s_l; $\sigma_l \ge 1$ – коефіцієнт запасу на помилку прогнозування.

Даний розрахунок забезпечує, що кількість реплік буде достатньою для обробки прогнозованого навантаження по обом метрикам, коефіцієнт α дозволяє управління ефективністю використання ресурсів, а коефіцієнт σ дозволяє справлятися з неточністю прогнозу.

Фактична кількість реплік $n_l(\tau)$ змінюється з затримкою відносно бажаної кількості $n_l^*(\tau)$ через затримки запуску подів, тому для урахування цього явища використаємо наступне рівняння:

$$\forall t_i \in \tau : n_{l,i} = n_{l,i-1} + a^+_{l,i-d_l^{start}} - a^-_{l,i-d_l^{start}}, \tag{3}$$

де $n_{l,i}$ – фактична кількість реплік мікросервісу s_l у момент t_i; $d_l^{start} = \frac{t_l^{start}}{\Delta t}, d_l^{stop} = \frac{t_l^{stop}}{\Delta t}$ – затримки запуску/зупинки в кроках, викликані часом запуску/зупинки однієї репліки; $a^+_{l,i-d_l^{start}}$ та $a^-_{l,i-d_l^{stop}}$ – кількість реплік мікросервіса типу s_i, рішення про запуск/зупинку яких було прийнято d_l^{start} / d_l^{stop} кроків назад.

Аналогічно враховуються затримки під час запуску чи зупинки віртуальних машин:

$$\forall t_i \in \tau : y_{j,i} = y_{j,i-1} + u^+_{j,i-d^{start}} - u^-_{j,i-d^{stop}}, \tag{4}$$

де $y_{j,i}$ – фактична кількість активних віртуальних машин c_j у момент t_i; $d^{start} = \frac{t^{start}}{\Delta t}, d^{stop} = \frac{t^{stop}}{\Delta t}$ – затримки запуску/зупинки в кроках, викликані часом запуску/зупинки віртуальної машини; $u^{+}_{l,i-d^{start}}$ та $u^{-}_{l,i-d^{stop}}$ – кількість віртуальних машин типу c_j, рішення про запуск/зупинку яких було прийнято d^{start}/d^{stop} кроків назад.

Обмеження рішень про зупинку реплік мікросервісів формулюється як:

$$\forall t_i \in T, \forall s_l \in S : a^{-}_{l,i} \le n_{l,i}. \tag{5}$$

Схожим чином формулюється обмеження рішень про зупинку віртуальних машин:

$$\forall t_i \in T, \forall c_j \in C : u^{-}_{j,i} \le y_{j,i}. \tag{6}$$

аланс розміщення подів має гарантувати, що їхнє розміщення збігається з загальною кількістю реплік мікросервісів:

$$\forall t_i \in T, \forall s_l \in S : \sum_j x_{p,k,i} = n_{l,i}. \tag{7}$$

Ресурсні обмеження покликані гарантувати, що система не використовує більше обчислювальних ресурсів, ніж насправді існує:

$$\forall t_i \in T, \forall s_l \in S, \forall c_j \in C : \sum_l n_{l,j,i} \cdot cpu_l \le y_{j,i} \cdot cpu_j . \tag{8}$$

$$\forall t_i \in T, \forall s_l \in S, \forall c_j \in C : \sum_j n_{l,j,i} \cdot mem_l \le y_{j,i} \cdot mem_j. \tag{9}$$

Обмеження покриття попиту для задачі пакування часто формулюється через м’яке обмеження для того, аби залишити задачу розв’язною, тож обмеження можна сформулювати наступним чином:

$$\forall t_i \in T, \forall s_l \in S : n_{l,i} = n^{*}_{l,i} - e^{-}_{l,i} + e^{+}_{l,i}, \tag{10}$$

де $e^{-}_{l,i}$ кількість невистачаючих реплік мікросервіса типу s_l у момент t_i; $e^{-}_{l,i}$ кількість надлишкових реплік мікросервіса типу s_l у момент t_i.

Основні характеристики об’єктів мають бути невід’ємними:

$$\forall t_i \in T, \forall s_l \in S, \forall c_j \in C, \forall p_p \in P, \forall v_k \in V : n_{l,i}, y_{j,i}, x_{p,k,i}, a^{\pm}_{l,i}, u^{\pm}_{j,i} \in Z_0. \tag{11}$$

Тепер, коли обмеження моделі було формалізовано, можемо перейти до опису цільової функції, яка має зменшити витрати на оренду віртуальних машин та врахувати негативні наслідки, описані в концептуальній моделі, у вигляді штрафів. Отже, цільова функція, що складається з сумарних витрат на оренду віртуальних машин протягом усього часу спостереження, штрафів за запуск/зупинку нових віртуальних машин задачі, штрафів за запуск/зупинку нових реплік мікросервісів та штрафів за переміщення подів між віртуальними машинами, має наступний вид:

$$\min\left(\sum_{i,j} \cos t_j \cdot y_{j,i} + \sum_{i,j} \left(\lambda^{+} u^{+}_{j,i} + \lambda^{-} u^{-}_{j,i} \right) + \sum_{i,l} \left(\omega^{+} e^{+}_{l,i} + \omega^{-} e^{-}_{l,i} \right) + + \varphi \sum_{p,k,i} \left| x_{p,k,i} - x_{p,k,i-1} \right| \right). \tag{12}$$

Обмеження (3)–(11) та цільова функція (12) сформовані на основі припущень:

– всі часові параметри системи, такі як час запуску чи зупинки репліки мікросервіса або віртуальної машини, є цілими та кратними Δt;

– для всіх типів віртуальних машин час запуску і зупинки є однаковим;

– вартість використання віртуальної машини певного типу є сталою для всіх класів протягом усього періоду спостереження;

– під час розрахунку кількості поточних реплік мікросервіса, здатних обробляти навантаження, враховуються лише ті репліки, рішення про зупинку яких ще не було прийнято;

– під час розрахунку кількості активних віртуальних машин, на яких можуть бути розміщені поди, враховуються лише ті віртуальні машини, рішення про зупинку яких ще не було прийнято.

Висновки

Таким чином, в роботі запропоновано підхід до моделювання прийняття рішень по управлінню ресурсами мікросервісів у середовищі Kubernetes та динамічному вибору віртуальних машин, який базується на прогнозуванні навантаження, динамічному моделюванні та bin packing. Виконано формалізацію основних характеристик предметної області з використанням математичного апарату теорії множин, та розроблено формальну постановку задачі оптимального управління ресурсами мікросервісів і вибору віртуальних машин.

Запропонована модель дозволяє прогнозувати майбутнє навантаження на систему, визначати необхідні кроки масштабування, аналізувати їхній вплив з точки зору операційних витрат та приймати найбільш ефективні

рішення по вибору віртуальних машин, необхідних для розгортки реплік мікросервісів. Результати роботи можуть бути використані у подальших дослідженнях та реальних імплементаціях розробленої моделі у програмне забезпечення, що дозволить покращити рівень обслуговування клієнтів додатків, побудованих на мікросервісній архітектурі та скоротити операційні витрати завдяки проактивному масштабуванню і ефективнішому використанню ресурсів клауд-провайдерів.

Список використаної літератури

1. Kubernetes. Horizontal Pod Autoscaling. URL: https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/ (дата звернення: 03.07.2025).

2. Федоришин Б., Красько О. (2024) Міграція сервісів в кластері Kubernetes на основі прогнозування навантаження. *Інфокомунікаційні технології та електронна інженерія*, випуск 4, номер 2, сторінки 82–92. URL: https://doi.org/10.23939/ictee2024.02.082 (дата звернення: 09.07.2025).

3. Сімакін С., Божуха Л. (2024) Прогнозування навантаження на сервер з використанням ШІ для оптимізації веб–сервісів. *Актуальні проблеми автоматизації та інформаційних технологій*, номер 28, сторінки. 234–243. URL: http://doi.org/10.15421/432422 (дата звернення: 12.07.2025).

4. Snehal Chaflekar, Rajendra Rewatkar. (2025) Novel load prediction in microservice architecture using attention mechanism-based deep LSTM networks. *International Journal of Innovative Research and Scientific Studies*, vol. 8, no. 3, pp. 1046–1058. URL: https://doi.org/10.53894/ijirss.v8i3.6751 (дата звернення: 15.07.2025).

5. Гутман Д., Сирота О. (2023) Проактивне автоматичне масштабування вверх для Kuberneters. *Адаптивні системи автоматичного управління*, номер 1, сторінки 32-38. URL: https://doi.org/10.20535/1560-8956.42.2023.278925 (дата звернення: 23.07.2025).

6. Wei-Kuang Lai, You-Chiun Wang, Syu-Chen Wei. (2023) Delay-Aware Container Scheduling in Kubernetes. *IEEE Internet of Things Journal*, vol. 10, no. 13, pp. 11813–11824. URL: https://doi.org/10.1109/JIOT.2023.3244545 (дата звернення: 01.08.2025).

7. Kubernetes. Resource Bin Packing. URL: https://kubernetes.io/docs/concepts/scheduling-eviction/resource-bin-packing/ (дата звернення: 07.08.2025).

8. Rodriguez, M. A., & Buyya, R. (2018) Containers Orchestration with Cost-Efficient Autoscaling in Cloud Computing Environments. *ArXiv*, abs/1812.00300. URL: https://doi.org/10.48550/arXiv.1812.00300 (дата звернення: 15.08.2025).

9. Guruge PB and Priyadarshana YHPP. (2025) Time series forecasting-based Kubernetes autoscaling using Facebook Prophet and Long Short-Term Memory. Frontiers in Computer Science, vol. 7. URL: https://doi.org/10.3389/fcomp.2025.1509165 (дата звернення: 19.08.2025).

10. Maiyza, A. I., Hassan, H. A., Sheta, W. M. (2025) VTGAN based proactive VM consolidation in cloud data centers using value and trend approaches. *Scientific Reports*, vol. 15, no. 20133. URL: https://doi.org/10.1038/s41598-025-04757-z (дата звернення: 27.08.2025).

References

1. Kubernetes (n.d.). Horizontal Pod Autoscaling. Kubernetes. Retrieved from: https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/ (accessed 03 July 2025).

2. Fedoryshyn B., Krasko O. (2024) Mighracija servisiv v klasteri Kubernetes na osnovi proghnozuvannja navantazhennja [Migration of services in a kubernetes cluster based on workload forecasting]. *Infocommunication technologies and electronic engineering journal*, vol. 4, no. 2, pp. 82-92. Retrieved from: https://doi.org/10.23939/ictee2024.02.082 (accessed 09 July 2025).

3. Simakin S., Bozhukha L. (2024) Proghnozuvannja navantazhennja na server z vykorystannjam ShI dlja optymizaciji veb–servisiv. [Server load forecasting using AI for web services optimization]. *Aktualjni problemy avtomatyzaciji ta informacijnykh tekhnologhij* [Current problems of automation and information technologies] № 28, pp. 234–243. Retrieved from: http://doi.org/10.15421/432422 (accessed 12 July 2025).

4. Snehal Chaflekar, Rajendra Rewatkar. (2025) Novel load prediction in microservice architecture using attention mechanism-based deep LSTM networks. *International Journal of Innovative Research and Scientific Studies*, vol. 8, no. 3, pp. 1046–1058. Retrieved from: https://doi.org/10.53894/ijirss.v8i3.6751 (accessed 15 July 2025).

5. Gutman D., Sirota O. (2023) Proaktyvne avtomatychne masshtabuvannja vverkh dlja Kuberneters [Proactive Upstream Autoscaling for Kubernetes]. *Adaptyvni systemy avtomatychnogho upravlinnja* [Adaptive Automatic Control Systems] no. 1, pp. 32–38. Retrieved from: https://doi.org/10.20535/1560-8956.42.2023.278925 (accessed 23 July 2025).

6. Wei-Kuang Lai, You-Chiun Wang, Syu-Chen Wei. (2023) Delay-Aware Container Scheduling in Kubernetes. *IEEE Internet of Things Journal*, vol. 10, no. 13, pp. 11813 – 11824. Retrieved from: https://doi.org/10.1109/JIOT.2023.3244545 (accessed 01 August 2025).

7. Kubernetes (n.d.). Resource Bin Packing. Kubernetes. Retrieved from: https://kubernetes.io/docs/concepts/scheduling-eviction/resource-bin-packing/ (accessed 07 August 2025).

8. Rodriguez, M. A., & Buyya, R. (2018) Containers Orchestration with Cost-Efficient Autoscaling in Cloud Computing Environments. ArXiv, abs/1812.00300. Retrieved from: https://doi.org/10.48550/arXiv.1812.00300 (accessed 15 August 2025).

9. Guruge PB and Priyadarshana YHPP. (2025) Time series forecasting-based Kubernetes autoscaling using Facebook Prophet and Long Short-Term Memory. *Frontiers in Computer Science*, vol. 7. Retrieved from: https://doi.org/10.3389/fcomp.2025.1509165 (accessed 19 August 2025).

10. Maiyza, A. I., Hassan, H. A., Sheta, W. M. (2025) VTGAN based proactive VM consolidation in cloud data centers using value and trend approaches. *Scientific Reports* (electronic journal), vol. 15, no. 20133. Retrieved from: https://doi.org/10.1038/s41598-025-04757-z (accessed 27 August 2025).

Дата першого надходження рукопису до видання: 29.09.2025
Дата прийнятого до друку рукопису після рецензування: 23.10.2025
Дата публікації: 28.11.2025