https://doi.org/10.31649/1997-9266-2024-175-4-86-94

УДК 004.047+005.8

В. В. Коваленко[1]
М. М. Букасов[1]

МЕТОДИ ТА МОДЕЛІ СКЛАДАННЯ РОЗКЛАДІВ ДЛЯ ОРКЕСТРАТОРА KUBERNETES

[1]Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

В умовах розвитку хмарних технологій, центри обробки даних все частіше використовують оркестратор Kubernetes, що сприяє ефективному управлінню контейнеризованими застосунками. Водночас Kubernetes не є досконалим, і його використання пов'язано з певними проблемами, серед яких можна виділити проблему складання ефективних розкладів. Її актуальність пояснюється тим, що не завжди вбудований модуль kube-scheduler будує найефективніші розклади. Актуальність підсилюється окремими випадками, в яких неефективно складений розклад призводив до неможливості розгорнути застосунок. Як критерій оптимізації вибрано максимізацію коефіцієнта середньої завантаженості вузла. Це зроблено з огляду на цілі центрів обробки даних зменшувати витрати на електроенергію та припущень про неефективність простою обчислювальних ресурсів. Визначено, що математично формалізовані обмеження як частини математичних моделей зазвичай згадуються у публікаціях, у яких розглядаються евристичні методи та метаевристичні методи. Всього визначено шість основних типів математично формалізованих обмежень; серед них найрозповсюдженим і найважливішим є обмеження на розмір пам'яті. Зазначено, що метод, який вибирається для розв'язання задачі, може бути пов'язаний з вибраним критерієм оптимізації. Загалом визначено дев'ять основних типів методів, що використовуються у задачах складання розкладів для Kubernetes. Серед них як найперспективніші вибрано три: методи штучного інтелекту, евристичні методи та метаевристичні методи. Цей вибір зроблено завдяки аналізу прикладів їхнього успішного застосування у разі складання розкладів з широким спектром критеріїв оптимізації (зокрема задач з критеріями, подібними до вибраного) як у тих хмарних середовищах, що застосовують Kubernetes, так і у тих, що його не застосовують.

Ключові слова: Kubernetes, оркестрація, хмарні технології, хмарні обчислення, розклад, теорія розкладів.

Вступ

Галузь хмарних обчислень можна вважати такою, що розвивається, оскільки значна кількість організацій вибирають саме хмарні технології для досягнення власних бізнес-цілей та як основу своєї хмарної інфраструктури [1], [2]. Зі зростанням галузі хмарних обчислень, важливішим також стає питання підвищення ефективності роботи центрів обробки даних; цьому можуть сприяти стратегії оптимізації використання ресурсів: до них належать використання контейнеризованих застосунків та алгоритмів складання розкладів [3]. Цим пояснюється популярність технології Kubernetes як такої, що, по-перше, керує саме контейнеризованими застосунками, а по-друге, містить реалізації різноманітних алгоритмів складання розкладів [3], [4].

Тоді як Kubernetes стає все популярнішим, збільшується і кількість досліджень щодо подолання типових проблем, притаманних цій технології. Основні напрями відповідних досліджень такі:

1) проблеми безпеки у Kubernetes. Публікації у цьому напрямку [5]—[10] розглядають вразливості проєктування, реалізації та роботі середовищ, неправильні конфігурацій, що призводять до загроз безпеці, стратегії та системи перевірок конфігурацій та сканувань зображень контейнерів, впровадження систем примусового виконання політик безпеки, та можливості використання машинного навчання для покращення виявлення загроз. Виявлено, що значна кількість стратегій поліпшення безпеки кластерів Kubernetes не призводить до погіршення продуктивності системи;

2) проблеми балансування навантаження у Kubernetes. У публікаціях цього напрямку [11]—[14]

зазначено недоліки проєктування вбудованого функціоналу для балансування навантаження у Kubernetes. Зазначається, що вбудований функціонал підтримує тільки прості статичні стратегії та покладається на сторонній функціонал для балансування, що надається хмарним провайдером: під час роботи вбудованого функціоналу один вузол (node) як «лідер» може брати на себе занадто велике навантаження. Публікації пропонують нові алгоритми балансування навантаження, що містять розподілення «лідера» між багатьма вузлами, динамічне розподілення запитів, категоризацію робіт тощо;

3) проблеми автоматичного масштабування у Kubernetes. У публікаціях з цього напрямку [15]—[19] зазначено обмеження у вбудованому функціоналі для автоматичного масштабування, та розглядають стратегії прогнозування або відстеження навантаження, що дозволяють починати масштабування раніше, та стратегії прибирання перепон, що сповільнюють процес автоматичного масштабування. Також розглянуто застосування методів штучного інтелекту, як-от навчання з підкріпленням, для підвищення якості обслуговування (QoS). Окремо варто зазначити, що прогнозування навантаження на центр оброблення даних ускладняється нестаціонарним навантаженням на обчислювальні ресурси [20];

4) проблеми ефективного складання розкладів у Kubernetes. Ці проблеми тісно пов’язані з проблемами балансування навантаження та автоматичного масштабування, оскільки у них також постає питання про кількість необхідних обчислювальних ресурсів, їхнє ефективне використання, та ефективне призначення робіт на ресурси. Ефективне використання ресурсів, водночас, сприятиме зменшенню витрат центру оброблення даних. До технології Kubernetes відноситься вбудований компонент kube-scheduler, що відповідає за побудову розкладів, і який має на меті визначати, які щойно створені зграї (pods) будуть призначені на які вузли; зграї можуть бути створені на основі об’єктів типу «Робота» (Job), натомість вузли приблизно відповідають фізичним чи віртуальним машинам [1], [21]—[23]. Алгоритм, який використовує kube-scheduler, базується на алгоритмах розв’язання задачі про розміщення в ємності, також відомої як задачі пакування (bin packing problem) [1], [24]. Відомо, що Kubernetes дає можливість використовувати власні системи складання розкладів замість вбудованого компонента для складання розкладів kube-scheduler [25]. Водночас відомо, що розклади, складені kube-scheduler, часто є неоптимальними, а в окремих випадках неефективне розміщення зграй на вузлах в результаті роботи kube-scheduler унеможливлювали розгортання застосунку [26].

У статті загалом розглянуто методи та математичні моделі, що використовуються у різноманітних задачах складання розкладу для хмарних середовищ із застосуванням Kubernetes. Попередні публікації [3], [27], в яких розглянуто задачі складання розкладів у Kubernetes, фокусувалися на класифікації типів критеріїв, типів навантажень та типів розглянутих середовищ. Поточний огляд фокусується насамперед на методах, що використовуються у різноманітних задачах складання розкладу; це має на меті спростити вибір метода для розв’язання будь-якої задачі у майбутніх дослідженнях. Також у поточному огляді привернуто увагу на обмеження, які існують у математичних моделях задач складання розкладів.

Метою дослідження є проведення аналізу методів та математичних моделей, що використовуються у задачах складання розкладів, і класифікація відповідних методів в залежності від вибраних критеріїв оптимізації.

Постановка задачі для дослідження

Ефективне використання обчислювальних ресурсів сприяє зменшенню витрат центрів обробки даних. Беручи до уваги саме витрати на енергоресурси від роботи серверів як головний критерій ефективності роботи центру обробки даних, для подальшого дослідження вибрано конкретну задачу складання розкладу. Зроблено такі припущення:

– простій обчислювального ресурсу є невигідним;

– чим менше вузлів є активними у кластері, тим більша середня завантаженість кожного активного вузла (відповідно, вибраний критерій близький до критерію мінімізації кількості вузлів);

– чим менше вузлів є активними у кластері, тим менше фізичних машин може бути необхідно для їхнього розміщення, і тим меншими будуть витрати центру оброблення даних на електроенергію, необхідну для роботи серверів;

Як критерій подальшої оптимізації та метрики для відслідковування в подальших експериментах вибрана максимізація коефіцієнта середньої завантаженості вузла.

У [28] зазначається, що у публікаціях, пов'язаних зі складанням розкладів для хмарних середовищ, критерії оптимізації витрат на інфраструктуру хмарного провайдера розглядаються не так часто, як критерії виконання вимог користувачів. Водночас це дослідження є не першим, що розглядає критерій, пов'язаний з мінімізацією витрат електроенергії; до прикладу, у [29] розглянуто моделювання роботи апаратного забезпечення для зниження споживання електроенергії, а у [30] розглянуто політику вбудованого компонента для складання розкладів kube-scheduler з критерієм мінімізації викидів вуглецю до атмосфери.

Огляд методів розв'язання задач та критеріїв оптимізації у задачах

Для вирішення зазначених проблем зазвичай використовуються такі типи методів:

1) евристичні методи (такі, що будуються на основі розумних, правдоподібних міркувань [31]);

2) метаевристичні методи (полягають у комбінуванні декількох наявних процедур, серед яких можна виділити одну провідну і декілька підлеглих; до розповсюджених метаевристичних методів можна віднести методи природних обчислень [31]);

3) методи штучного інтелекту;

4) методи багатокритеріальної оптимізації (полягають у формулюванні множини критеріїв, єдиної для всіх альтернатив, та оцінки альтернатив за кожним із критеріїв [32]);

5) методи, що базуються на пріоритетах;

6) методи, що базуються на графах;

7) методи ухвалення рішення на основі даних про мережу;

8) методи ухвалення рішення на основі даних, отриманих від апаратного забезпечення;

9) методи динамічного складання розкладів (полягають у зміні порядку виконання інструкцій апаратним забезпеченням з метою зменшення кількості простоїв у роботі програмного забезпечення [33]).

Окремо варто зазначити, деякі публікації розглядають декілька методів одночасно.

У табл. 1 наведено детальну класифікацію публікацій за типами методів, що використовуються, у зіставленні з критеріями оптимізації, які автори намагаються покращити за допомогою відповідних методів.

Таблиця 1

Класифікація публікацій за методами та критеріями

№ з/п	Тип методів	Джерела	Конкретні методи, алгоритми та стратегії, що використовуються	Критерії оптимізації, які використовуються в задачах за вибраними методами
1	Евристичні методи	[34]—[37]	Агресивне пакування (щільне заповнення не повністю заповнених обчислювальних ресурсів); модифікація правила справедливості домінуючих ресурсів, що враховує потреби та поточне споживання ресурсів; алгоритм, який враховує баланс; врахування споживання ресурсів роботами; жадібний алгоритм	Мінімізація: витрат центру обробки даних; перевантаження серверів. Максимізація: метрик справедливості; частки успішно виконаних робіт; якості обслуговування (QoS)
2	Метаевристичні методи	[38]—[40]	Алгоритм мурашиних колоній; алгоритм рою часток	Мінімізація: накладних витрат під час передачі даних (у мережі між мікросервісами); максимального коефіцієнта використання ресурсів фізичної машини; середньої кількості невдалих викликів мікросервісів; затримки; ризику поодиноких збоїв; моменту завершення останньої роботи; витрат на використання ресурсів

Продовження табл. 1

3	Методи штучного інтелекту (ШІ)	[28], [37], [41]—[51]	Методи машинного навчання (МН): нейронні мережі (зокрема глибинне навчання, навчання з підкріпленням); інтелектуальний аналіз даних; марківський процес прийняття рішень; методи прогнозування вузла, призначення роботи на який буде найефективнішим; прогнозування майбутньої завантаженості вузлів	**Мінімізація**: перевантаження серверів; інтерференції (у випадку робіт з тренуванням алгоритмів МН); дисбалансу навантаження; загального часу виконання робіт; моменту завершення останньої роботи; кількості вчасно виконаних робіт; витрат на використання ресурсів. **Максимізація**: якості досвіду (QoE); продуктивності; коефіцієнта використання ресурсів
4	Методи багатокритеріальної оптимізації	[38], [51], [52]	TOPSIS; використання метаевристичних методів та методів штучного інтелекту для оптимізації більше ніж одного критерію (див. вище)	**Мінімізація**: накладних витрат під час передачі даних у мережі між мікросервісами; максимального коефіцієнта використання ресурсів фізичної машини; середньої кількості невдалих викликів мікросервісів; витрат на використання ресурсів; моменту завершення останньої роботи
5	Методи, які базуються на пріоритетах	[24], [53]	Призначення робіт з найвищим пріоритетом першими з урахуванням обмежень спорідненості (affinity); визначення пріоритетів на основі кількості необхідних ресурсів; збільшення та зменшення частот викликів функцій збільшення та зменшення відповідно кількості вузлів; використання стратегії групового призначення робіт (gang scheduling)	**Максимізація**: якості обслуговування (QoS); коефіцієнта використання ресурсів
6	Методи, що базуються на графах	[47], [54]—[55]	Використання нейронних мереж, заснованих на графах; нечітка кластеризація графа, що сприятиме запуску декількох образів того самого мікросервіса на різних вузлах; використання стратегії пакетного планування робіт (batch job scheduling), що полягає у групуванні робіт зі схожими характеристиками	**Мінімізація:** часу відгуку; витрат на використання ресурсів; вихідного трафіку
7	Методи ухвалення рішення на основі даних про мережу	[56], [57]	Акцентування на розміщенні пов'язаних між собою мікросервісів, гарантуючи резервацію пропускної здатності; використання динамічних метрик від мережі	**Мінімізація:** затримок у мережі; часу відповіді застосунків
8	Методи ухвалення рішення на основі даних, отриманих від апаратного забезпечення	[29], [30]	Отримання даних за метриками від різних вузлів і перепризначення робіт на вузли з найкращими показниками	**Мінімізація:** споживання електроенергії; викидів вуглецю у атмосферу
9	Методи динамічного складання розкладів	[58]—[61]	Врахування часу, необхідного для введення/виведення інформації на/з диску вузла; врахування залежностей між роботами; врахування шкали часу зграй; врахування історичних даних про виконання робіт; врахування поточної завантаженості; вимкнення недовикористаних ресурсів та перепризначення робіт на інші ресурси	**Мінімізація:** накладних витрат під час передачі даних; витрат на використання ресурсів. **Максимізація:** використання прискорювачів апаратного забезпечення

У статтях, вказаних у табл. 1, зазначається, що авторам вдалося поліпшити значення вибраних ними критеріїв оптимізації, в порівнянні зі значеннями критеріїв під час роботи компонента за замовчуванням (kube-scheduler). Варто зазначити, що переліки критеріїв оптимізації відрізняються для різних типів методів, що свідчить про те, що критерій оптимізації може впливати на вибір типу метода.

Як видно з табл. 1, найбільша частка досліджень (12 публікацій) основана саме на використанні методів ШІ у складанні розкладів, тож використання методів ШІ можна вважати перспективним напрямом у дослідженні поставленої задачі.

Також перспективним можна вважати напрям використання евристичних та метаевристичних

методів як простих у реалізації і водночас ефективних. Відомо, що для хмарних середовищ, що не використовують Kubernetes, використання евристичних та метаевристичних алгоритмів, як-от алгоритму променевого пошуку та адаптивних генетичних алгоритмів, також підвищує ефективність розкладів [62], [63].

У контексті подальшого проведення експериментів та необхідності порівнювати ефективність розробленого алгоритму в порівнянні з поведінкою за замовчуванням, слід зазначити публікацію [64], що описує розроблення симулятора для перевірки ефективності складання розкладів, що також містить реалізації алгоритмів складання розкладів. Цей чи аналогічний програмний продукт може бути використано під час проведення експериментів над готовою моделлю складання розкладів.

Огляд обмежень у задачах

Серед вищерозглянутих статей, багато містять математичні моделі (ММ) задачі складання розкладу, в яких виражаються критерій оптимізації (цільова функція) та обмеження. У цьому розділі будуть наведені основні обмеження, наявні у математичних моделях. Варто зазначити, що повноцінні ММ (такі, що містять не лише критерій оптимізації, але і математично формалізовані обмеження) наводяться переважно у статтях, що розглядають евристичні методи, метаевристичні методи та методи ухвалення рішення на основі даних про мережу.

Наведені у публікаціях математично формалізовані обмеження можна класифікувати так:

1) обмеження на розмір пам’яті (кількість ядер та розмір пам’яті, що вимагається запитом, має бути меншим або дорівнювати кількості ядер та розміру пам’яті, що можуть надаватися на вузлі відповідно);

2) обмеження на час виконання запиту (фактичний час виконання запиту має бути меншим або дорівнювати максимально допустимій затримці);

3) обмеження на кількість екземплярів контейнерів (кількість виділених мікросервісів має дорівнювати кількості відповідних екземплярів контейнерів);

4) обмеження на кількість контейнерів, розгорнутих на вузлі (на будь-якому вузлі може бути не більше одного контейнера з одного мікросервісу; це необхідно для запобігання конфліктів ресурсів одного вузла поміж екземплярів контейнерів одного мікросервісу; подібні обмеження називаються правилами антиспорідненості (anti-affinity rules) і їхня реалізація присутня зокрема і у kube-scheduler [1], [22], [65]);

5) визначення розгорнутого застосунку (застосунок вважається розгорнутим тоді і тільки тоді, якщо всі його репліки розгорнуті);

6) визначення призначеної зграї (зграя вважається призначеною тоді і тільки тоді, якщо всі її екземпляри-репліки створені).

У табл. 2 подана відповідність між типами обмежень, типами методів та джерелами, у яких описані будь-які обмеження та методи.

Таблиця 2

Відповідність між типами обмежень та типами методів

№ з/п	Тип обмеження	Типи методів		
		Евристичні методи	Метаевристичні методи	Методи ухвалення рішення на основі даних про мережу
1	Обмеження на розмір пам’яті	[37]	[38]	[58]
2	Обмеження на час виконання запиту	[37]	—	—
3	Обмеження на кількість екземплярів контейнерів	—	[38]	—
4	Обмеження на кількість контейнерів, розгорнутих на вузлі	—	[38]	[58]
5	Визначення розгорнутого застосування	—	—	[58]
6	Визначення призначеної зграї	—	—	[58]

Статті, що розглядають методи типів 3—6, 8—9 (нумерація згідно з табл. 1), не містять згадок математично формалізованих обмежень. Водночас статті, що розглядають методи ШІ, без математичної формалізації згадують обмеження на час виконання запиту та обмеження пропускної здат-

ності мережі [49], а статті, що розглядають методи динамічного складання розкладів, без формалізації зазначають те, що у разі пакетного планування робіт у робіт існують директивні строки [61].

Окремо варто зазначити, що у статті [37] реалізацію обмежень у Kubernetes здійснено за допомогою політик. Водночас у поточній версії Kubernetes, політики у разі складання розкладів більше не підтримуються; замість них документація пропонує використання конфігурацій модуля kube-scheduler [66], [67].

Висновки

Зазначено зростання інтересу до хмарних технологій та до інформаційної технології-оркестратора Kubernetes серед організацій, що використовують їх для досягнення бізнес-цілей, та серед дослідників. Для збільшення ефективності роботи хмарних центрів обробки даних варто розв'язати наукові проблеми і задачі, до яких належить зокрема задача складання розкладів.

Описано загальні відомості про вбудований до Kubernetes модуль для складання розкладів kube-scheduler, описані проблеми, які можуть виникати у разі неефективно складеного розкладу. Як задачі для подальшого розгляду вибрано задачі з пов'язаними між собою критеріями оптимізації: мінімізація витрат на енергоресурси від роботи серверів та максимізація коефіцієнта середньої завантаженості машини.

Розглянуті публікації класифіковано за використовуваними методами; визначено 9 типів методів, кожний з яких застосовується у задачах з різноманітними критеріями оптимізації. Серед яких як найперспективніші виділено методи штучного інтелекту та метаевристичні методи як такі, що розглядаються у значній кількості публікацій та використовуються у задачах складання розкладу з широким масивом критеріїв оптимізації. Додатково виділено 6 типів математично формалізованих обмежень, які часто зустрічаються у математичних моделях.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

[1] В. В. Коваленко, і Д. О. Вітковський, «Контейнерна технологія (оркестратор) управління виділеними хмарними ресурсами,» *Інженерія програмного забезпечення і передові інформаційні технології (Soft Tech-2023)*, Київ, Україна, 19-21 груд. 2023. Київ: НТУУ «КПІ ім. Ігоря Сікорського», 2023, с. 173-177.

[2] L. Golightly, V. Chang, Q. A. Xu, X. Gao, and B. S. C. Liu, "Adoption of cloud computing as innovation in the organization," *International Journal of Engineering Business Management*, vol. 14, Jan. 2022. [Electronic resource]. Available: 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, Jun. 2023. [Electronic resource]. Available: https://doi.org/10.1186/s13677-023-00471-1 .

[4] "Overview," *Kubernetes*. [Electronic resource]. Available: https://kubernetes.io/docs/concepts/overview/ .

[5] S. I. Shamim, "Mitigating security attacks in kubernetes manifests for security best practices violation," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. New York, NY, USA: Assoc. Comput. Machinery, 2021, pp. 1689-1690. [Electronic resource]. Available: https://doi.org/10.1145/3468264.3473495 .

[6] A. Rahman, S. I. Shamim, D. B. Bose, and R. Pandita, "Security Misconfigurations in Open Source Kubernetes Manifests: An Empirical Study," *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 4, May 2023. [Electronic resource]. Available: https://doi.org/10.1145/3579639 .

[7] A. Aly, M. Fayez, M. Al-Qutt, and A. M. Hamad, "Multi-Class Threat Detection Using Neural Network and Machine Learning Approaches in Kubernetes Environments," in *2024 6th International Conference on Computing and Informatics (ICCI)*. 2024, pp. 103-108. [Electronic resource]. Available: https://doi.org/10.1109/ICCI61671.2024.10485133 .

[8] R. Shevchuk, M. Karpinski, M. Kasianchuk, I. Yakymenko, A. Melnyk, and R. Tykhyi, "Software for Improve the Security of Kubernetes-based CI/CD Pipeline," in *2023 13th International Conference on Advanced Computer Information Technologies (ACIT)*. 2023, pp. 420-425. [Electronic resource]. Available: https://doi.org/10.1109/ACIT58437.2023.10275654 .

[9] Á. Revuelta Martinez, "Study of Security Issues in Kubernetes (K8s) Architectures; Tradeoffs and Opportunities," Uppsala Univ., Dept. Inf. Technol., 2023. [Electronic resource]. Available: https://www.diva-portal.org/smash/get/diva2:1781490/FULLTEXT01.pdf .

[10] S. Gwak, T.-P. Doan, and S. Jung, "Container Instrumentation and Enforcement System for Runtime Security of Kubernetes Platform with eBPF," *Intelligent Automation & Soft Computing*, vol. 37, no. 2, pp. 1773-1786, 2023. [Electronic resource]. Available: https://doi.org/10.32604/iasc.2023.039565 .

[11] N. Nguyen, and T. Kim, "Toward Highly Scalable Load Balancing in Kubernetes Clusters," *IEEE Communications Magazine*, vol. 58, no. 7, pp. 78-83, 2020. [Electronic resource]. Available: https://doi.org/10.1109/MCOM.001.1900660 .

[12] K. Takahashi, K. Aida, T. Tanjo, and J. Sun, "A Portable Load Balancer for Kubernetes Cluster," in *Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region*. New York, NY, USA: Assoc. Comput. Machinery, 2018, pp. 222-231. [Electronic resource]. Available: https://doi.org/10.1145/3149457.3149473 .

[13] Q. Liu, E. Haihong, and M. Song, "The Design of Multi-Metric Load Balancer for Kubernetes," in *2020 International Conference on Inventive Computation Technologies (ICICT)*, 2020, pp. 1114-1117. [Electronic resource]. Available: https://doi.org/10.1109/ICICT48043.2020.9112373 .

[14] A. Dua, S. Randive, A. Agarwal, and N. Kumar, "Efficient Load balancing to serve Heterogeneous Requests in Clustered Systems using Kubernetes," in *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*,

2020, pp. 1-2. [Electronic resource]. Available: https://doi.org/10.1109/CCNC46108.2020.9045136 .

[15] Д. Гутман та О. Сирота, «Проактивне автоматичне масштабування вверх для Kuberneters,» *Адаптивні системи автоматичного управління*, т. 1, № 42, с. 32-38, трав. 2023. [Електронний ресурс]. Режим доступу: https://doi.org/10.20535/1560-8956.42.2023.278925 .

[16] Я. Маєвський, та Н. Праворська, «Підвищеня ефективності автоматизації масштабування мікросервісів у системі керування контейнеризованими застосунками Kubernetes,» *Вісник Хмельницького національного університету. Технічні науки*, т. 313, № 5, с. 260-264, жовт. 2022. [Електронний ресурс]. Режим доступу: https://doi.org/10.31891/2307-5732-2022-313-5-260-264 .

[17] T.-T. Nguyen, Y.-J. Yeom, T. Kim, D.-H. Park, and S. Kim, "Horizontal Pod Autoscaling in Kubernetes for Elastic Container Orchestration," *Sensors*, vol. 20, no. 16, p. 4621, Aug. 2020. [Electronic resource]. Available: https://doi.org/10.3390/s20164621 .

[18] M.-N. Tran, D.-D. Vu, and Y. Kim, "A Survey of Autoscaling in Kubernetes," in *2022 Thirteenth International Conference on Ubiquitous and Future Networks (ICUFN)*, 2022, pp. 263-265. [Electronic resource]. Available: https://doi.org/10.1109/ICUFN55119.2022.9829572 .

[19] A. A. Khaleq, and I. Ra, "Intelligent Autoscaling of Microservices in the Cloud for Real-Time Applications," *IEEE Access*, vol. 9, pp. 35464-35476, 2021. [Electronic resource]. Available: https://doi.org/10.1109/ACCESS.2021.3061890 .

[20] E. Zharikov, S. Telenyk, and P. Bidyuk, "Adaptive Workload Forecasting in Cloud Data Centers," *Journal of Grid Computing*, vol. 18, no. 1, pp. 149-168, Nov. 2019. [Electronic resource]. Available: https://doi.org/10.1007/s10723-019-09501-2

[21] "Viewing Pods and Nodes," *Kubernetes*. [Electronic resource]. Available: https://kubernetes.io/docs/tutorials/kubernetes-basics/explore/explore-intro/ .

[22] "Kubernetes Components," *Kubernetes*. [Electronic resource]. Available: https://kubernetes.io/docs/concepts/overview/components/ .

[23] "Controllers," *Kubernetes*. [Electronic resource]. Available: https://kubernetes.io/docs/concepts/architecture/controller/ .

[24] S. Prasad Lahu, "Dynamic Resources allocation using Priority Aware scheduling in Kubernetes," MSc Research Project, Nat. College Ireland, 2019. [Electronic resource]. Available: https://norma.ncirl.ie/4137/1/prasadlahushelar.pdf .

[25] "Configure Multiple Schedulers," *Kubernetes*. [Electronic resource]. Available: https://kubernetes.io/docs/tasks/extend-kubernetes/configure-multiple-schedulers/ .

[26] 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. [Electronic resource]. Available: https://doi.org/10.1007/978-3-030-91431-8_14 .

[27] Z. Rejiba, and J. Chamanara, "Custom Scheduling in Kubernetes: A Survey on Common Problems and Solution Approaches," *ACM Comput. Surv.*, vol. 55, no. 7, Dec. 2022. [Electronic resource]. Available: https://doi.org/10.1145/3544788 .

[28] A. Mampage, S. Karunasekera, and R. Buyya, "Deep reinforcement learning for application scheduling in resource-constrained, multi-tenant serverless computing environments," *Future Generation Computer Systems*, vol. 143, pp. 277-292, 2023. [Electronic resource]. Available: https://doi.org/10.1016/j.future.2023.02.006 .

[29] P. Townend *et al.*, "Invited Paper: Improving Data Center Efficiency Through Holistic Scheduling In Kubernetes," in *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, 2019, pp. 156-15610. [Electronic resource]. Available: https://doi.org/10.1109/SOSE.2019.00030 .

[30] A. James, and D. Schien, "A Low Carbon Kubernetes Scheduler," 2019. [Electronic resource]. Available: https://ceur-ws.org/Vol-2382/ICT4S2019_paper_28.pdf .

[31] Л. Ф. Гуляницький, і О. Ю. Мулеса, *Прикладні методи комбінаторної оптимізації*. Київ, Україна: Видавничо-полігр. центр «Київ. ун-т», 2016.

[32] О. С. Жураковська, *Теорія прийняття рішень*. Київ: НТУУ «КПІ ім. Ігоря Сікорського», 2020. [Електронний ресурс]. Режим доступу: https://ela.kpi.ua/server/api/core/bitstreams/deaf214b-73d1-447f-8e2c-7ce1417e6d83/content .

[33] R. Parthasarathi, *Computer Architecture*. INFLIBNET Cent., 2018. [Electronic resource]. Available: https://www.cs.umd.edu/~meesh/411/CA-online/ .

[34] A. Chung, J. Park, and G. Ganger, "Stratus: cost-aware container scheduling in the public cloud," 2018, pp. 121-134. [Electronic resource]. Available: https://doi.org/10.1145/3267809.3267819 .

[35] A. Beltre, P. Saha, and M. Govindaraju, "KubeSphere: An Approach to Multi-Tenant Fair Scheduling for Kubernetes Clusters," in *2019 IEEE Cloud Summit*. 2019, pp. 14-20. [Electronic resource]. Available: https://doi.org/10.1109/CloudSummit47114.2019.00009 .

[36] Z. Liu, C. Chen, J. Li, Y. Cheng, Y. Kou, and D. Zhang, "KubFBS: A fine-grained and balance-aware scheduling system for deep learning tasks based on kubernetes," *Concurrency and Computation: Practice and Experience*, vol. 34, no. 11, Jan. 2022. [Electronic resource]. Available: https://doi.org/10.1002/cpe.6836 .

[37] 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, 2024. [Electronic resource]. Available: https://doi.org/10.1016/j.future.2024.03.007 .

[38] 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. [Electronic resource]. Available: https://doi.org/10.1109/ACCESS.2019.2924414 .

[39] A. Alelyani, A. Datta, and G. M. Hassan, "Optimizing Cloud Performance: A Microservice Scheduling Strategy for Enhanced Fault-Tolerance, Reduced Network Traffic, and Lower Latency," *IEEE Access*, vol. 12, pp. 35135-35153, 2024. [Electronic resource]. Available: https://doi.org/10.1109/ACCESS.2024.3373316 .

[40] Z. Wei-guo, M. Xi-lin, and Z. Jin-zhong, "Research on Kubernetes' Resource Scheduling Scheme," in *Proceedings of the 8th International Conference on Communication and Network Security*. New York, NY, USA: Assoc. Comput. Machinery, 2018, pp. 144-148. [Electronic resource]. Available: https://doi.org/10.1145/3290480.3290507 .

[41] M. Carvalho, and D. F. Macedo, "Container Scheduling in Co-Located Environments Using QoE Awareness," *IEEE Transactions on Network and Service Management*, vol. 20, no. 3, pp. 3247-3260, 2023. [Electronic resource]. Available: https://doi.org/10.1109/TNSM.2023.3244090 .

[42] Q. Si, X. Lu, W. Li, and P. Pu, "DeepLRA: An Efficient Long Running Application Scheduling Framework with Deep

Reinforcement Learning in the Cloud," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 14325 LNAI, pp. 157-163, 2024. [Electronic resource]. Available: https://doi.org/10.1007/978-981-99-7019-3_16 .

[43] Y. Bao, Y. Peng, and C. Wu, "Deep Learning-based Job Placement in Distributed Machine Learning Clusters," in *Proceedings – IEEE INFOCOM*. 2019, pp. 505-513. [Electronic resource]. Available: https://doi.org/10.1109/INFOCOM.2019.8737460 .

[44] X. Xie *et al.*, "DRS: A deep reinforcement learning enhanced Kubernetes scheduler for microservice-based system," *Software: Practice and Experience*, Oct. 2023. [Electronic resource]. Available: https://doi.org/10.1002/spe.3284 .

[45] X. Wang, K. Zhao, and B. Qin, "Optimization of Task-Scheduling Strategy in Edge Kubernetes Clusters Based on Deep Reinforcement Learning," *Mathematics*, vol. 11, pp. 42-69, Oct. 2023. [Electronic resource]. Available: https://doi.org/10.3390/math11204269 .

[46] Y. Fu *et al.*, "Progress-based Container Scheduling for Short-lived Applications in a Kubernetes Cluster," in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 278-287. [Electronic resource]. Available: https://doi.org/10.1109/BigData47090.2019.9006427 .

[47] Y. Han, S. Shen, X. Wang, S. Wang, and V. C. Leung, "Tailored Learning-Based Scheduling for Kubernetes-Oriented Edge-Cloud System," in *IEEE INFOCOM 2021 – IEEE Conference on Computer Communications*, 2021, pp. 1-10. [Electronic resource]. Available: https://doi.org/10.1109/INFOCOM42981.2021.9488701 .

[48] I. Harichane, S. A. Makhlouf, and G. Belalem, "A Proposal of Kubernetes Scheduler Using Machine-Learning on CPU/GPU Cluster," in *Intelligent Algorithms in Software Engineering*, R. Silhavy, Ed. Cham: Springer Int. Publishing, 2020, pp. 567-580. [Electronic resource]. Available: https://doi.org/10.1007/978-3-030-51965-0_50 .

[49] J. Dogani, and F. Khunjush, "Proactive auto-scaling technique for web applications in container-based edge computing using federated learning model," *Journal of Parallel and Distributed Computing*, vol. 187, 2024. [Electronic resource]. Available: https://doi.org/10.1016/j.jpdc.2024.104837 .

[50] Z. Xu, Y. Gong, Y. Zhou, Q. Bao, and W. Qian, *Enhancing Kubernetes Automated Scheduling with Deep Learning and Reinforcement Techniques for Large-Scale Cloud Computing Optimization*, 2024. [Electronic resource]. Available: https://doi.org/10.48550/arXiv.2403.07905 .

[51] D. Jorge-Martinez *et al.*, "Artificial intelligence-based Kubernetes container for scheduling nodes of energy composition," *International Journal of System Assurance Engineering and Management*, Jul. 2021. [Electronic resource]. Available: https://doi.org/10.1007/s13198-021-01195-8 .

[52] T. Menouer, "KCSS: Kubernetes container scheduling strategy," *The Journal of Supercomputing*, vol. 77, no. 5, pp. 4267-4293, May 2021. [Electronic resource]. Available: https://doi.org/10.1007/s11227-020-03427-3 .

[53] M. F. Bestari, A. I. Kistijantoro, and A. B. Sasmita, "Dynamic Resource Scheduler for Distributed Deep Learning Training in Kubernetes," in *2020 7th International Conference on Advance Informatics: Concepts, Theory and Applications (ICAICTA)*, 2020, pp. 1-6. [Electronic resource]. Available: https://doi.org/10.1109/ICAICTA49861.2020.9429033 .

[54] E. Petrakis, V. Skevakis, P. Eliades, A. Aznavouridis, and K. Tsakos, "ModSoft-HP: Fuzzy Microservices Placement in Kubernetes," *Electronics*, vol. 13, p. 65, Dec. 2023. [Electronic resource]. Available: https://doi.org/10.3390/electronics13010065 .

[55] C. Misale *et al.*, "Towards Standard Kubernetes Scheduling Interfaces for Converged Computing," in *Driving Scientific and Engineering Discoveries Through the Integration of Experiment, Big Data, and Modeling and Simulation*, J. Nichols *et al.* Eds. Cham: Springer Int. Publishing, 2022, pp. 310-326. [Electronic resource]. Available: https://doi.org/10.1007/978-3-030-96498-6_18 .

[56] J. Santos, C. Wang, T. Wauters, and F. De Turck, "Diktyo: Network-Aware Scheduling in Container-Based Clouds," *IEEE Transactions on Network and Service Management*, vol. 20, no. 4, pp. 4461-4477, 2023. [Electronic resource]. Available: https://doi.org/10.1109/TNSM.2023.3271415 .

[57] Ł. Wojciechowski *et al.*, "NetMARKS: Network Metrics-AwaRe Kubernetes Scheduler Powered by Service Mesh," in *IEEE INFOCOM 2021 – IEEE Conference on Computer Communications*, 2021, pp. 1-9. [Electronic resource]. Available: https://doi.org/10.1109/INFOCOM42981.2021.9488670 .

[58] D. Li, Y. Wei, and B. Zeng, "A Dynamic I/O Sensing Scheduling Scheme in Kubernetes," in *Proceedings of the 2020 4th International Conference on High Performance Compilation, Computing and Communications*. New York, NY, USA: Assoc. Comput. Machinery, 2020, pp. 14-19. [Electronic resource]. Available: https://doi.org/10.1145/3407947.3407950 .

[59] C.-Y. Lin., T.-A. Yeh., and J. Chou., "DRAGON: A Dynamic Scheduling and Scaling Controller for Managing Distributed Deep Learning Jobs in Kubernetes Cluster," in *Proceedings of the 9th International Conference on Cloud Computing and Services Science – CLOSER*. SciTePress, 2019, pp. 569-577. [Electronic resource]. Available: https://doi.org/10.5220/0007707605690577 .

[60] G. El Haj Ahmed, F. Gil-Castiñeira, and E. Costa-Montenegro, "KubCG: A dynamic Kubernetes scheduler for heterogeneous clusters," *Software: Practice and Experience*, vol. 51, no. 2, pp. 213-234, Sep. 2020. [Electronic resource]. Available: https://doi.org/10.1002/spe.2898 .

[61] Z. Zhong, and R. Buyya, "A Cost-Efficient Container Orchestration Strategy in Kubernetes-Based Cloud Computing Infrastructures with Heterogeneous Resources," *ACM Trans. Internet Technol.*, vol. 20, no. 2, Apr. 2020. [Electronic resource]. Available: https://doi.org/10.1145/3378447 .

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

[63] O. Rolik, S. Telenyk, E. Zharikov, and V. Samotyy, "Dynamic Virtual Machine Allocation Based on Adaptive Genetic Algorithm," in *Cloud Computing 2017: The Eighth International Conference on Cloud Computing, GRIDs, and Virtualization*, Athens, Greece. IARIA, 2017, pp. 108-114. [Electronic resource]. Available: https://personales.upv.es/thinkmind/dl/conferences/cloudcomputing/cloud_computing_2017/cloud_computing_2017_6_20_20053.pdf .

[64] S. Wen *et al.*, "K8sSim: A Simulation Tool for Kubernetes Schedulers and Its Applications in Scheduling Algorithm Optimization," *Micromachines*, vol. 14, no. 3, 2023. [Electronic resource]. Available: https://doi.org/10.3390/mi14030651 .

[65] "Kubernetes Scheduler," *Kubernetes*. [Electronic resource]. Available: https://kubernetes.io/docs/concepts/scheduling-eviction/kube-scheduler/ .

[66] "Scheduling Policies," *Kubernetes*. [Electronic resource]. Available: https://kubernetes.io/docs/reference/scheduling/policies/.

[67] “Scheduler Configuration,” *Kubernetes*. [Electronic resource]. Available: https://kubernetes.io/docs/reference/scheduling/config/ .

Рекомендована кафедрою автоматизації та інтелектуальних інформаційних технологій ВНТУ

Стаття надійшла до редакції 25.06.2024

Коваленко Владислав Вадимович — аспірант кафедри інформаційних систем та технологій, e-mail: vlad.kov@ukr.net ;

Букасов Максим Михайлович — канд. техн. наук, доцент кафедри інформаційних систем та технологій.

Національний технічний університет України «Київський політехнічний інститут ім. Ігоря Сікорського», Київ

V. V. Kovalenko[1]
M. M. Bukasov[1]

Scheduling Methods and Models for Kubernetes Orchestrator

[1]National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”

In the conditions of evolution of cloud technologies, data centers use Kubernetes orchestrator more and more often, it enables the efficient management of containerized applications. At the same time, Kubernetes is not perfect, and its usage is related to certain problems, among which the problem of efficient scheduling can be highlighted. Its relevance can be explained by the fact that in-built kube-scheduler module does not always build the most efficient schedules. The relevance is intensified by the known cases when inefficiently formed schedule resulted in impossibility to deploy the application. Maximization of coefficient of average workload of the node was chosen as an optimization criterion. It was done based on the objectives of data centers to decrease the energy costs, and the assumptions about inefficiency of computing resources idling. It was determined that mathematically formalized constraints as parts of mathematical models are usually mentioned in the publications dedicated to heuristic methods and metaheuristic methods. In total, six main types of mathematically formalized constraints were determined; the most common and important among them is the memory size constraint. It is highlighted that the method that is getting chosen for solving a problem can be linked with the chosen optimization criterion. Generally, nine main types of methods that are used in problems of effective scheduling for Kubernetes were determined. Among them, three were chosen as the most promising ones: artificial intelligence methods, heuristic methods and metaheuristic methods. The reasons behind their selection include the examples of their successful usage in the formation of schedules with wide spectrum of optimization criteria (including problems with criteria that are similar to the chosen one) both in the cloud environments that use Kubernetes and the cloud environments that don’t use it.

Keywords: Kubernetes, orchestration, cloud technologies, cloud computing, schedule, scheduling theory.

Kovalenko Vladyslav V. — Post-Graduate Student of the Chair of Information Systems and Technologies, e-mail: vlad.kov@ukr.net ;

Bukasov Maksym M. — Cand. Sc. (Eng.), Associate Professor of the Chair of Information Systems and Technologies