DOI 10.34185/1562-9945-3-158-2025-16
УДК 004.75

І.П. Малініч, Я.В. Іванчук

ОСОБЛИВОСТІ РОЗМІЩЕННЯ МІКРОСЕРВІСІВ СИСТЕМ УПРАВЛІННЯ НАВЧАННЯМ У ГІБРИДНИХ ХМАРАХ

Анотація: У статті приділяється увага різним підходам та методам розміщення мікросервісів у хмарах, а також особливості їх застосування при розміщенні у гібридній хмарі мікросервісів систем управління навчанням. Актуальність дослідження полягає у тому що перенесення у хмару високонавантажених систем управління навчанням з великої кількістю інтеграційних зв'язків з іншими сервісами є досить складним завданням, яке не завжди можна вирішити за допомогою існуючих методів. Метою даного дослідження є розробка рішення для впорядкування потоків даних процесу розміщення мікросервісів з використанням різних методів та врахуванням особливостей освітніх інформаційних систем. Завдяки використанню методу структурного аналізу створено схему потоків даних автоматизованого розміщення мікросервісів систем управління навчанням у гібридній хмарі. Також розглянуто різні методи розміщення мікросервісів у хмарах: практичні та наукові, а також порівняно їх недоліки та переваги. Визначено основні показники оцінки мережевої доступності, які застосовуються для розміщення мікросервісів. Завдяки запропонованій схемі потоків даних автоматизованого розміщення мікросервісів у гібридній хмарі визначено способи вдосконалення методів розподілу мікросервісів таким чином щоб вони давали користувачеві достатній контроль над процесом, а також враховували особливості систем управління навчанням.
Ключові слова: мікросервіс, хмара, сервер, контейнеризація, система управління навчанням, хмарні обчислення.

Постановка проблеми. Для організації навчального процесу в закладах освіти часто використовуються електронні системи управління навчанням (англ. Learning Management Systems) типу Moodle Canvas LMS, Forma LMS та інші [1]. Однією із найбільш популярних систем управління навчанням (СУН) в Україні є Moodle, яка використовується багатьма закладами вищої освіти (ЗВО) та іншими навчальними закладами. Зазвичай якщо подібна система використовується у невеликих закладах освіти, де загалом не більше тисячі учнів, то для її розміщення може бути достатньо використання серверних ресурсів хостингу чи віртуального серверу у хмарі [2]. При подібних масштабах зазвичай не потрібно виконувати ніяких складних завдань з системного адміністрування чи підходів DevOps (акронім від англ. development і operations) для розміщення подібних систем та підтримки їх роботи. Проте, для навчальних закладів

більшого масштабу, таких як університети із кількістю студентів понад 7000 осіб зазвичай складно розміщувати СУН на звичайному хостингу чи серверах початкового рівня [3]. Для високонавантажених СУН може бути недостатньо ресурсів одного серверу і потрібно шукати способи масштабування баз даних, файлових сховищ та обчислювальних потужностей між різними серверами або в межах хмари. Більш сучасні СУН на початку, при встановленні, мають поділ на мікросервіси, які розгортаються в окремих контейнерах таких середовищах, як Docker чи Kubernetes [4] і можуть бути підключеними до хмарних сховищ типу AWS S3 та SharePoint, а також до хмарних баз даних (DynamoDB, Firebase). Також багато СУН мають широку інтеграцію з іншими ресурсами навчального закладу, наприклад з бібліотечними ресурсами, інструментами розробки програмного забезпечення, або навіть із системою прокторингу [1]. Таким чином загальна серверна інформаційна система навчальних закладів може бути розподілена між різними серверами або хмарами.

З початком повномасштабної війни в Україні багато ЗВО перемістили певну частину своїх інформаційних ресурсів у публічні хмари (наприклад у США та Європі), залишаючи при цьому інтеграцію із своїми ресурсами на фізичних серверах, які у них залишились в Україні. Це зумовлено насамперед тим, що розміщення усіх онлайн-ресурсів у хмарному середовищі може бути недоцільно, через те що це може бути занадто дороговартісно та спричиняти сповільнення доступу. Так як на фізичних серверах зазвичай встановлюються менш критичні системи, або інформаційні системи які не можна було винести за периметр локальної мережі організації, тому дослідження методів розміщення мікросервісів систем управління навчанням у гібридних хмарах є актуальним.

Мета дослідження. Метою цього дослідження є створення рішення для оптимізації потоків даних у процесі розміщення мікросервісів, застосовуючи різні методи та враховуючи специфіку освітніх інформаційних систем.

Аналіз останніх досліджень і публікацій. Заклади освіти які під час повномасштабної війни не розглядали можливість розміщення своїх онлайн-ресурсів у публічних хмарах або звичайних серверах поза межами країни стикаються із ризиками зниження доступності під час знеструмлень, пошкодження каналів зв’язку та втрати даних внаслідок ракетних та дронових атак на цивільну інфраструктуру. На момент написання даної статті це залишається для українських компаній та організацій найбільш вагомим аргументом для створення конфігурацій з розподілом обчислень між публічною хмарою та місцевих серверів (англ. on-premises) на технічному майданчику організацій. Подібні конфігурації хмарних інформаційних систем називаються гібридними хмарами. Гібридна хмара складається з приватної частини на серверах організації та гібридної хмари, яка надається постачальником хмарних послуг [2].

Серед існуючих напрямків досліджень, присвячених використання мікросервісів у СУН особливо можна виділити дослідження методів побудови ефективної архітектури СУН [3] за допомогою застосування різних підходів організації компонентів онлайн-сервісів. Розглянуто підходи розподілення мікросервісів на основі системи досвіду ре-

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

алізації різних версій API, розподілу за доменами, зв'язками з базами даних та міжсервісної інтеграції. Приділено увагу аналізу протоколів обміну даними між мікросервісами та їх моделями, здійснено порівняння принципів оркестрації та хореографії (організації транзакцій між мікросервісами). У досліджені розглянуто принципи організації модульної структури СУН і наведено споживання ресурсів реалізацією архітектури, побудованої на основі запропонованих у дослідженні методів.

Дослідженню передували вивчення міжсервісної інтеграції СУН із системами репозиторіїв навчальних матеріалів та академічних текстів [5]. У вивченні способів міжсервісної взаємодії розглянуто протокол OAI-PMH [5] та його реалізації у програмних бібліотеках різних мов програмування. Розглянуто міжнародні стандарти структурованих даних на основі Dublin Core та його подальший вплив на інтеграцію СУН із репозиторіями матеріалів. У науковій праці [1] розглянуто можливість пілотного запуску СУН у межах одного факультету на прикладі системи JetIQ [1]. Хоча у самій публікації безпосередньо цього не згадується, але на основі її даних та даних наведених у статті [3] можна побачити певні етапи еволюції системи управління електронним навчанням та її поступовий розподіл на окремі мікросервіси.

Іншим випадком міжсервісної інтеграції, який впливає на подальший мікросервісний розподіл є інтеграція СУН процесом із системами прокторингу [6]. У праці розглянуто методи інтеграції цих систем між собою, а також описано якими саме даними вони обмінюються між собою. У контексті даного дослідження дозволяє визначити, як інтеграція з прокторингом впливає на хмарну архітектуру СУН. У науковій праці [7] описується метод оптимізації процесу розміщення мікросервісів у хмарі, де пропонується розподіленню великого сервісу на ряд мікросервісів із забезпеченням найменшого рівня трафіку між ними, а також упаковуванню їх у віртуальні машини із врахуванням обмежень у ресурсах. При низькому рівні інтеграції між веб-сайтами або онлайн-сервісами організації мікросервіси досить легко розподілити між серверами на технічному майданчику організації та публічній хмарі постачальника. Адміністратор або менеджер визначає хмарний бюджет на пріоритети роботи кожного сайту чи сервісу, відповідно до чого і здійснюється ручний розподіл. Більш складною є ситуація, у якій між сервісами (або мікросервісами) існує досить високий рівень інтеграції.

Існує досить багато методів та підходів для прийняття рішень щодо розміщення мікросервісів у гібридних хмарах. Існує дві категорії подібних методів та підходів: практичні підходи, які щоденно використовують DevOps-інженери для проєктування хмар, а також наукові методи, запропоновані дослідниками. Серед розроблених спільнотою DevOps-інженерів для проєктування хмарних технологій є набір найкращих практик, що опираються на рішення, які гарно себе зарекомендували у відомих реалізованих проєктах. Наукові методи мають практичне застосування лише якщо у інженерів-практиків достатньо кваліфікації щоб їх використати. Незважаючи на наукову цінність, зазвичай їх нелегко інтегрувати у програмні інструменти, що використовуються DevOps-інженерами, особливо якщо у дослідників мало можливостей для співпраці із розробниками цих інструментів.

У сфері DevOps існують наступні практичні підходи територіального розміщення мікросервісів у мультирегіональних хмарах: розміщення мікросервісів за регіонами з найбільшим споживанням їх трафіку [8], використання глобального інструменту балансування навантаження, реплікація мікросервісів між регіонами, розподілення баз даних за регіонами, використання технології Service Mesh [4], а також використання AI-інструментів прийняття рішень щодо розміщення мікросервісів у мультирегіональних хмарних середовищах [2]. Майже всі описані підходи працюють у мультирегіональних хмарах, але разом з тим не пропонують дієвих рекомендацій з прийняття рішень щодо розміщення окремих мікросервісів у певній частині гібридної хмари – приватній чи публічній. AI-інструменти [9] для прийняття подібних рішень зазвичай є сервісами певного постачальника хмарних послуг, наприклад AWS Trusted Advisor, Azure Analysis Services та Google Active Assist [9]. Вони використовують зокрема наукові методи, запропоновані різними дослідниками, але віддають перевагу тим методам, які найбільше затребувані клієнтами [2]. Їх застосування є більш практичним для мультирегіональних хмар, що надаються тим самим постачальником, що й сам AI-інструмент. Дані інструменти в цілому досить позитивно себе зарекомендували окрім створення гібридних хмар, де є інфраструктура, яка повністю не контролюється тим самим постачальником хмарних послуг [2].

Поряд з практичними підходами прийняття рішень щодо територіального розміщення мікросервісів є також наукові підходи та методи для вирішення цієї проблеми. Саме для гібридних хмар існує підхід для розміщення програмних компонентів у хмарі на основі визначення інтенсивності трафіку між ними [10], але у ньому не приділено уваги самим мікросервісам. Також існує метод оптимізації розміщення мікросервісів зі зменшенням міжмашинного трафіку [7], але не передбачено його застосування у гібридних хмарах. У іншому методі розміщення мікросервісів запропоновано використання нечіткої логіки [11]. У деяких наукових методах та підходах акцент зроблений на мережевих параметрах. Серед них можна виділити підхід у вигляді політики оптимізації децентралізованого розміщення сервісів у туманних обчисленнях [12], в якому приділено увагу кількості мережевих переходів у хмарах та способам їх зменшення. Ще одним підходом, де враховуються мережеві метрики є розміщення сервісів для суспільних мережевих мікро-хмар [13]. У інших існуючих наукових методах та підходах більше приділяється увага розміщенню обчислень у мультирегіональних хмарах, без поділу на публічні та приватні хмари та/або без зосередження уваги саме на мікросервісній архітектурі: метод розміщення віртуальної машини з врахуванням пропускної здатності датацентрів [14], підхід розміщення віртуальних машин з урахуванням розширення мережі датацентру [15], метод розміщення обчислювальних ресурсів у розподілених хмарах [16], модель розгортання мікросервісних додатків основі графіків вартості [17], а також генетичний штрафний метод для розміщення SaaS-сервісів [18]. Здебільшого дослідники та інженери, які запропонували для цього описані методи та підходи, не проводять великої різниці між віртуальними машинами, контейнерами та розгорнутих у них мікросервісах.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

Також залишається не вирішеним питання використання більш дешевих спотових хмарних та локальних обчислювальних ресурсів для менш критичних завдань. У даній роботі планується зосередити фокус саме на розміщенні контейнеризованих мікросервісів у певних частинах гібридної хмари. При проектуванні інформаційних систем в обчислювальних хмарах, зокрема на основі мікросервісів, перед архітекторами та DevOps-інженерами ставиться ціль, щоб такі системи могли найбільш ефективно працювати, була швидко та постійно доступними з мережі Інтернет для персоналу компанії-користувача та її клієнтів, а також не створювали надлишкових витрат на їх утримання. При проектуванні інформаційних систем у гібридних хмарах завжди виникає вузьке місце (англ. *bottleneck*) між обома частинами такої хмари: приватної та публічної [2]. Виникнення проблем на межі обох хмар часто призводить до деградації якості роботи сервісів, що працюють у гібридних хмарах. Для проведення оцінювання якості роботи різних розгорнутих у хмарі сервісів зазвичай потрібно проводити моніторинг різних метрик, які для кожного хмарного сервісу часто є індивідуальними. Різні дослідники [15,16] та хмарні архітектори [2,4] виділяють показники мережевої доступності розміщених у хмарі сервісів як найбільш універсальні метрики для оцінки якості їх роботи зі сторони кінцевого користувача.

Серед цих показників можна виділити наступні:

– аптайм (англ. *uptime*) – час безперервної роботи інформаційної системи;

– час відгуку сервера (або TTFB) – час у мілісекундах до першого байту, надісланого сервером з моменту запиту;

– затримка пакетів (або затримка пінгу, англ. *ping*) – час у мілісекундах скільки пакет проходить від клієнта до сервера і назад;

– кількість мережевих переходів (англ. *traceroute hops*) – кількість проміжних мережевих вузлів між клієнтом та сервером.

Ці чотири основних показника можна використати для того, щоб оцінити мережеву доступність інформаційних систем у гібридних хмарах. Аптайм – це основний параметр, за допомогою якого можна оцінити наскільки система стабільно працює та зберігає зовнішню доступність з мережі Інтернет. Аптайм вимірюється у відсотках і відображає скільки часу в рік система була доступною. Протилежним поняттю аптайму є даунтайм (англ. *downtime*), який відображає скільки часу в рік система була недоступною. Рівень аптайму забезпечується постачальником хмарних послуг згідно «Угоди про рівень послуг» – SLA (англ. *Service Level Agreement*). Згідно загальноприйнятим стандартам хмарної та телеком-індустрій прийнятне значення SLA коливається між 99,9% та 99,999% До прикладу при SLA 99,9% для постачальника допускається наступний час недоступності послуг: щоденно – 1,44 хвилини; щотижнево – 10,08 хвилин; щомісячно – 43,8 хвилин; щорічно – 8.76 годин [19]. В умовах використання гібридної хмари забезпечення високого рівня аптайму можна легко досягнути лише у публічній частині хмари завдяки SLA, у той час як у своїй власній приватній хмарі компанія-клієнт має сама дбати про рівень аптайму, що не завжди досяжно і фінансово виправдано. Іншим показником який пропонується використовувати для оцінки інформаційних систем у гібридних хмарах є час відгуку сервера (TTFB). Він найбільш чітко відображає на-

явність внутрішніх вузьких місць у розгорнутій у хмарі інформаційній системі. Зокрема, зростання затримки TTFB у гібридній хмарі може бути обумовлене розривом критичних залежностей між спорідненими мікросервісами, які опинились у різних частинах хмари. У такому випадку на збільшення затримки TTFB може впливати велика дистанція між приватною та публічною частинами хмари, що характеризується ростом затримки пакетів та збільшенням кількості мережевих переходів. Проте, ці показники далеко не завжди корелюються з часом затримки TTFB, оскільки на цей параметр можуть впливати також помилки конфігурації сервісів у хмарі, а також помилки допущені при проектуванні та розробці мікросервісів. Тому ці два показники використовуються при оцінці мережевої доступності інформаційних систем окремо.

Ще одним мережевим параметром який впливає на якість роботи онлайн-сервісів є затримка передачі пакетів. Її можна виміряти відносно певного хоста (наприклад серверу) за допомогою використання мережевої утиліти ping [2]. Подібна затримка зазвичай є критичною при використанні онлайн-сервісів, що працюють в режимі реального часу, наприклад VoIP-телефонія, конференц-зв'язок, онлайн-радіо, відеостріми та онлайн-ігри. При збільшенні затримок пакетів до неприйнятного рівня якість цих сервісів починає падати і ці затримки відображаються у вигляді відчутних кінцевому користувачеві затримок зв'язку. Зазвичай для веб-сервісів, де не використовуються комунікація в режимі реального часу це лише призводить до збільшення TTFB та зазвичай не дуже сильно погіршує роботу веб-додатків чи веб-сайтів. Однак цей параметр може грати важливу роль при виявленні проблем зі швидкодією міжхмарних зв'язків у гібридній хмарі. Зростання цього параметру може відобразитись на швидкодії певних сервісів, доступ до яких відбувається з іншої частини гібридної хмари, наприклад реляційних баз даних [2]. Зовнішня затримка пакетів поза межами приватної та публічної частинами гібридної хмари, які йдуть через Інтернет, не залежить від внутрішньої конфігурації хмари, а натомість залежить від топології взаємоз'єднання магістральних Інтернет-провайдерів, точок обміну трафіку та їх пропускних здатностей. Подібна кон'юнктура їх взаємоз'єднань не є постійною і може постійно змінюватись в залежності від аварійних ситуацій, DDoS-атак [2]. В Інтернеті затримка пакетів виміряна за допомогою ICMP Ping не завжди є показником якості зв'язку, оскільки у деяких мережах протокол ICMP може мати нижчий пріоритет QoS ніж інші протоколи. Натомість для аналізу оптимальності розподілу мікросервісів всередині хмари даний параметр дозволяє визначати наявність вузьких місць у мережевій конфігурації та уникати їх виникнення у подальшому.

Разом із першими трьома показниками оцінки мережевої доступності є також четвертий – кількість мережевих переходів. Їх можна виміряти відносно певного хоста (наприклад серверу) за допомогою використання мережевої утиліти traceroute [2]. Цей показник є також корисним для аналізу оптимальності розміщення мікросервісів усередині хмари, проте, як і затримка пакетів, вона може бути корисним лише для оцінки мережі всередині хмари. Хоча існує думка, що є чітка кореляція між затримкою передачі пакетів та кількістю хопів [2], але вона спостерігається по більшій мірі лише в іде-

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

альних умовах, наприклад в навчальних мережевих стендах, в яких немає справжнього трафіку [14].

Якщо у хмарній інформаційній системі існує жорстка зв'язність між мікросервісами (рис. 1), то це створює ряд проблем у стабільності та продуктивності такої системи, оскільки жорстка зв'язність є антипатерном. Як і в об'єктно-орієнтованому програмуванні, коли при жорсткій зв'язності мікросервісів один із них аварійно завершує роботу – тоді всі інші сервіси також не зможуть нормально працювати. Впорядкування архітектури зв'язків називають декуплінгом, яке покликане зробити кожен мікросервіс більш незалежним та самостійним. Правильний підхід до декуплінгу мікросервісів передбачає дотримання принципу єдиної відповідальності (англ. *Single Responsibility Principle*), у якій кожен мікросервіс або його окремий модуль виконує лише одне конкретне завдання. Для визначення ступеню зв'язності у спільності розробників програмного забезпечення використовується метрика зв'язності Coupling(C) [20]:

$$Coupling\,(C) = 1 - \frac{1}{d_i + 2 \times c_i + d_o + 2 \times c_o + g_d + 2 \times g_c + w + r} \quad (1)$$

де d_i – кількість вхідних параметрів даних, c_i – кількість вхідних контрольних параметрів, d_o – кількість вихідних параметрів даних, c_o – кількість вихідних контрольних параметрів, g_d – кількість глобальних змінних, які використовуються як дані, g_c – кількість глобальних змінних, які використовуються для контролю, w – кількість викликаних компонентом (fan-out), r – кількість компонентів, що викликають компонент, який розглядається (fan-in).

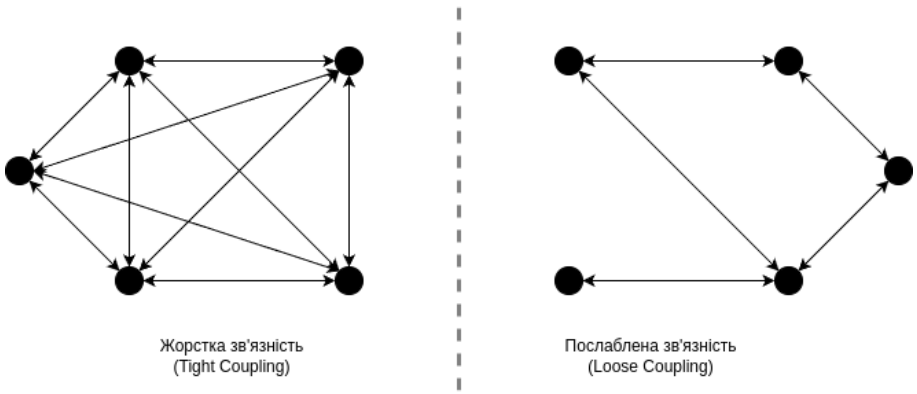

Рисунок 1 - Різні ступені зв'язності між мікросервісами

Чим більший рівень зв'язності має модуль або мікросервіс, тим більше значення значення має параметр Coupling (C), значення якого варіюється від 0,67 (слабка зв'язність) до 1,0 (сильна зв'язність). Крім зв'язності в архітектурі мікросервісів, як і у об'єктно-орієнтованому програмуванні грає важливу роль також таке поняття як когезія. Когезія – це ступінь зчеплення елементів всередині модуля чи мікросервісу. Когезія і зв'язність майже завжди обговорюються разом, оскільки вони тісно взаємопов'язані, але у той же час когезія не залежить від кількості зв'язків між елементами [21]. Це можна помітити зокрема із огляду на формулу визначення ступеня когезії:

$$coh(c,k) = \frac{N_c^k}{N^k + N_c + N_c^k} \quad (2)$$

де N – кількість елементів, c – компонент, що розглядається, k – ключ компоненту.

Потоки даних автоматизованого розміщення мікросервісів у гібридній хмарі. У мікросервісному підході до проектування хмарних інформаційних систем когезія залежить від кількості модулів у конкретному мікросервісі та їх взаємозв'язки з модулями в іншому мікросервісі, оскільки кожен мікросервіс зазвичай складається з різних взаємопов'язаних модулів. Високозв'язні мікросервіси більш чутливі до впливу різних факторів мережевої конфігурації, такі як швидкість каналу, стабільність зв'язку, фрагментація пакетів, стиснення та шифрування [22]. Незважаючи на те що розглянуті наукові методи показали хорошу ефективність, але під час розгляду особливостей їх застосування виявлено, що вони не дають повного контролю над процесом прийняття рішень. Тому, на рисунку 2 виділені окремі процеси в існуючих методах (фігури 1-4), завдяки розділенню яких користувач може отримати більший контроль над прийняттям рішень. Даний підхід дозволяє вдосконалити автоматизацію процесу прийняття рішень щодо розміщення мікросервісів у гібридній хмарі виділивши три окремі процеси які передуватимуть верифікації початкової конфігурації, а саме: аналіз мережевої зв'язності між мікросервісами, прогнозування потреби мікросервісів у хмарних ресурсах, аналіз пріоритетів кіберзахищеності кожного мікросервісу, а також сам процес прийняття рішень, який буде відбуватись після верифікації початкової конфігурації. При розробці методу аналізу мережевої зв'язності між мікросервісами необхідно дослідити, які є різновиди зв'язків між різними мікросервісами та іншими хмарними компонентами, та визначити які з них можуть бути розділені міжхмарним зв'язком і що на це впливає. Для цього необхідно дослідити як мережевий трафік передається між різними частинами хмари. При побудові хмарної архітектури СУН на основі мікросервісів у хмарі можуть спостерігатись наступні види трафіку між мікросервісами: безпосередні API-виклики, обмін повідомленнями, трафік доступу до баз даних, а також запити від мережевих клієнтів [3].

Практично реалізувати функціонування подібної схеми потоків даних як на рисунку 2 можливо у вигляді інформаційної системи у на основі мікросервісів. Такий підхід дозволить поєднати різні технології, які будуть використані при реалізації тих чи інших методів і моделей. Подібна інформаційна система зможе забезпечити користувачеві можливість отримувати інформацію про нинішній стан гібридної хмари, про можливі подальші сценарії щодо розміщення мікросервісів у цільовій гібридній хмарі, а також виконувати моделювання перерозподілу мікросервісів. При виконанні практичної реалізації подібної інформаційної системи необхідно передбачити різні етапи взаємодії з користувачем. На початковому етапі користувач повинен мати можливість отримувати інформацію про нинішній стан гібридної хмари та мікросервісів, які у ній розміщені.

Наступним чином користувач повинен мати можливість виконувати аналіз зв'язності між мікросервісами, а також проводити аналіз їх пріоритетів кіберзахищеності. Для цього пропонується виконувати моделювання зовнішніх вторгнень для кож-

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

ного мікросервісу щоб спросити визначення пріоритету кіберзахисту. Іншим обов'язковим етапом у розміщення мікросервісів у гібридній хмарні має бути прогнозування забезпечення мікросервісів хмарними ресурсами. Хоча існують практичні підходи для конфігурування прогнозів у хмарах для універсальних хмарних обчислень [2], але є необхідність у створенні більш специфічних методів або моделей для гібридних хмар та СУН.

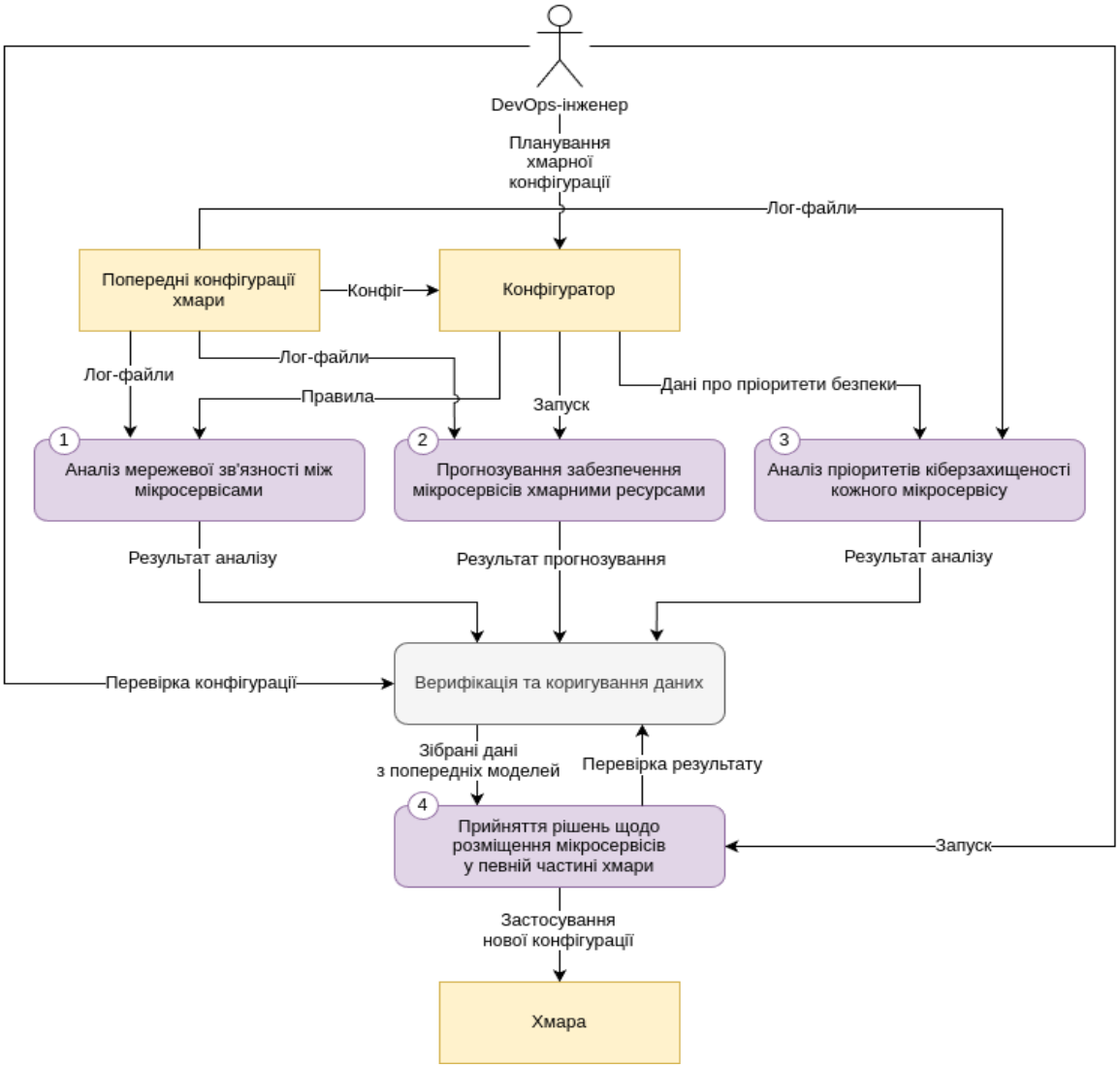

Рисунок 2 - Схема потоків даних автоматизованого розміщення мікросервісів у гібридній хмарі

При використанні приватної хмари зростання ресурсів обмежене навяними серверами, тому для подальшого масштабування приватної хмари необхідно закуповувати чи орендувати нові фізичні сервери, що не завжди можливо зробити. Для публічної хмари масштабування системи не потребує закупівлі нових серверів, але є зазвичай дорожчим рішенням [2] ніж закупівля серверів. Постачальники хмарних послуг пропонують авторозширення послуг якщо початково замовленого об'єму недостатньо. Однак цей процес необхідно контролювати та здійснювати прогнозування росту масштабів інформаційних систем.

Висновки. Проведено аналіз особливостей існуючих методи прийняття рішень щодо розміщення мікросервісів у хмарах, завдяки якому з'ясовано що існуючі методи розміщення є достатньо ефективними, але дають мало контролю над процесом архітекторам хмар, що є більш важливим при динамічному рості споживання хмарних ресурсів СУН та збільшенню їх функціоналу з часом. Для формування бачення подальшого розвитку методів розміщення мікросервісів СУН у гібридній хмарі визначено основні показники доступності та зв'язності мікросервісів.

Також запропоновано схему потоків даних процесу автоматизованого розміщення мікросервісів у гібридній хмарі, яка відповідає реаліям функціонування та розвитку СУН. Існуючі на даний момент інструменти розпізнавання та аналізу програмного коду дозволяють вдосконалити раніше існуючі методи досліджень гібридних хмар та особливостей розгортань мікросервісів у них. Крім того потребують подальшого дослідження вплив індивідуальної архітектури мікросервісів на їх розміщення у гібридній хмарі, а також як ризики кібербезпеки хмар впливають на даний процес прийняття рішень. Саме тому у подальшому потрібно дослідити процес розподілу мікросервісних обчислень у гібридних хмарах з врахуванням особливостей функціонування та розвитку СУН, а також розвинути запропоновану ідею автоматизації даного процесу, запропонувати нові чи вдосконалені методи або моделі, що реалізують окремі їх процеси.

ЛІТЕРАТУРА

1. Bisikalo, O. V., Palamarchuk, Y. A., Kovalenko, O. O. Results of the implementation of the pilot project of the management system for learning and coordination of educational, methodological, and scientific activities "JetIQ" // Proceedings of the 9th Scientific and Practical Conference, Lviv, November 21-23, 2017. – Львів: Publishing House of the Shevchenko Scientific Society, 2017. – С. 73–77.

2. Comer D. The Cloud Computing Book: The Future of Computing Explained. – Chapman and Hall/CRC, 2021. – 400 с.

3. Паламарчук Є. А. Methods of building microservice architecture of e-learning systems // Information Technologies and Computer Engineering. – 2022. – № 1. – С. 43–54.

4. Chawla H., Kathuria H. Building microservices applications on Microsoft Azure: designing, developing, deploying, and monitoring. – Apress, 2019. – 280 с.

5. Малініч І. П, Паламарчук Є. А. Способи отримання доступу до метаданих статей через API-інтерфейси пакету програмного забезпечення DSpace // Матеріали XLVI науково-технічної конференції підрозділів ВНТУ, Вінниця, 22–24 березня 2017 р. – Вінниця, 2017.

6. Денесяк О. І., Паламарчук Є. А. Комплексна система прокторингу у інформаційних технологіях аналізу контексту в системах оцінювання знань // Вісник ВПІ. – 2021. – № 6. – С. 93–99.

7. Hu Y., de Laat C., Zhao Z. Optimizing service placement for microservice architecture in clouds // Applied Sciences. – 2019. – Vol. 9, no. 21. – P. 4663. – DOI: 10.3390/app9214663.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

8. Magda D. Run Java microservices across multiple cloud regions with Spring Cloud // DZone. – 2022. – URL: https://dzone.com/articles/run-java-microservices-across-multiple-cloud-regiof (дата звернення: 16.10.2024).
9. Garapati S. E., Giral E., Antonijević S. Rethinking Monitoring for Cloud Environments: BMC Software AIOps Case Study // European Conference on Service-Oriented and Cloud Computing. – Cham: Springer Nature Switzerland, 2022. – P. 109–115.
10. Chen X., Tang S., Lu Z., Wu J., Duan Y., Huang S. C., Tang Q. iDiSC: A new approach to IoT-data-intensive service components deployment in edge-cloud-hybrid systems // IEEE Access. – 2019. – Vol. 7. – P. 59172–59184. –
URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8706879.
11. Petrakis E. G., Skevakis V., Eliades P., Aznavouridis A., Tsakos K. ModSoft-HP: Fuzzy Microservices Placement in Kubernetes // Electronics. – 2023. – Vol. 13, no. 1. – P. 65.
12. Guerrero C., Lera I., Juiz C. A lightweight decentralized service placement policy for performance optimization in fog computing // Journal of Ambient Intelligence and Humanized Computing. – 2019. – Vol. 10. – P. 2435–2452. – DOI: 10.1007/s12652-018-10068-5.
13. Selimi M., Cerdà-Alabern L., Freitag F., Veiga L., Sathiaseelan A., Crowcroft J. A lightweight service placement approach for community network micro-clouds // Journal of Grid Computing. – 2019. – Vol. 17. – P. 169–189. – DOI: 10.1007/s10723-018-09427-2.
14. Wang M., Meng X., Zhang L. Consolidating virtual machines with dynamic bandwidth demand in data centers // INFOCOM 2011. – 2011. – P. 71–75.
15. Meng X., Pappas V., Zhang L. Improving the scalability of data center networks with traffic-aware virtual machine placement // Proceedings of the 2010 IEEE INFOCOM. – San Diego, CA, USA, 2010. – P. 1–9.
16. Alicherry M., Lakshman T. Network-aware resource allocation in distributed clouds // Proceedings of the 2012 IEEE INFOCOM. – Orlando, FL, USA, 2012. – P. 963–971.
17. Leitner P., Cito J., Stöckli E. Modelling and managing deployment costs of microservice-based cloud applications // Proceedings of the 9th International Conference on Utility and Cloud Computing. – Shanghai, China, 2016. – P. 165–174. – ACM.
18. Yusoh Z. I. M., Tang M. A penalty-based genetic algorithm for the composite SaaS placement problem in the cloud // Proceedings of the IEEE Congress on Evolutionary Computation. – Barcelona, Spain, 2010. – P. 1–8.
19. Rogers O. Cloud SLAs punish, not compensate // Uptime Institute. – 2022. – URL: https://journal.uptimeinstitute.com/cloud-slas-punish-not-compensate/ (дата звернення: 16.10.2024).
20. Pressman R. S. Software Engineering – A Practitioner's Approach. – 4th ed. – McGraw-Hill, 1982. – ISBN 0-07-052182-4.
21. Tulka T. How cohesion and coupling correlate // URL: https://blog.ttulka.com/how-cohesion-and-coupling-correlate/ (дата звернення: 16.10.2024).
22. Ivanchuk Y. V., Horobets Y. V., Koval K. O. Method for increasing the degree of protection in message encryption based on a time-constant algorithm // System Technologies: Regional Interuniversity Collection of Scientific Papers. – 2022. – Vol. 2, no. 139. – P. 3–13. – DOI: 10.34185/1562-9945-2-139-2022-01.

REFERENCES

1. Bisikalo, O. V., Palamarchuk, Y. A., & Kovalenko, O. O. (2017). Results of the implementation of the pilot project of the management system for learning and coordination of educational, methodological, and scientific activities "JetIQ". In *Proceedings of the 9th Scientific and Practical Conference, Lviv, November 21-23, 2017* (pp. 73-77). Lviv: Publishing House of the Shevchenko Scientific Society.

2. Comer, D. (2021). *The Cloud Computing Book: The Future of Computing Explained*. Chapman and Hall/CRC.

3. Palamarchuk, Y.A. (2022). Methods of building microservice architecture of e-learning systems. *Information Technologies and Computer Engineering, (1),* 43-54.

4. Chawla, H., & Kathuria, H. (2019). *Building microservices applications on Microsoft azure: designing, Developing, Deploying, and Monitoring*. Apress.

5. Малініч І., & Паламарчук, Є. (2017). Способи отримання доступу до метаданих статей через API-інтерфейси пакету програмного забезпечення DSpace. In *Proceedings of the XLVI Scientific and Technical Conference of VNTU Departments, Vinnytsia, March 22-24*, 2017.

6. Денесяк, О. І., & Паламарчук, Є. А. (2021). Комплексна система прокторингу у інформаційних технологіях аналізу контексту в системах оцінювання знань. *Вісник ВПІ, (6),* 93-99.

7. Hu, Y., de Laat, C., & Zhao, Z. (2019). Optimizing service placement for microservice architecture in clouds. *Applied Sciences, 9(21),* 4663. https://doi.org/10.3390/app9214663

8. Magda, D. (2022). Run Java microservices across multiple cloud regions with Spring Cloud. DZone. Retrieved from https://dzone.com/articles/run-java-microservices-across-multiple-cloud-regio

9. Garapati, S. E., Giral, E., & Antonijević, S. (2022). Rethinking Monitoring for Cloud Environments: BMC Software AIOps Case Study. In European Conference on Service-Oriented and Cloud Computing (pp. 109-115). Cham: Springer Nature Switzerland.

10. Chen, X., Tang, S., Lu, Z., Wu, J., Duan, Y., Huang, S. C., & Tang, Q. (2019). iDiSC: A new approach to IoT-data-intensive service components deployment in edge-cloud-hybrid systems. *IEEE Access, 7,* 59172–59184.
https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8706879

11. Petrakis, E. G., Skevakis, V., Eliades, P., Aznavouridis, A., & Tsakos, K. (2023). ModSoft-HP: Fuzzy Microservices Placement in Kubernetes. *Electronics, 13(1),* 65.

12. Guerrero, C., Lera, I., & Juiz, C. (2019). A lightweight decentralized service placement policy for performance optimization in fog computing. *Journal of Ambient Intelligence and Humanized Computing, 10,* 2435–2452. https://doi.org/10.1007/s12652-018-10068-5

13. Selimi, M., Cerdà-Alabern, L., Freitag, F., Veiga, L., Sathiaseelan, A., & Crowcroft, J. (2019). A lightweight service placement approach for community network micro-clouds. *Journal of Grid Computing, 17,* 169–189. https://doi.org/10.1007/s10723-018-09427-2

14. Wang, M., Meng, X., & Zhang, L. (2011). Consolidating virtual machines with dynamic bandwidth demand in data centers. *INFOCOM 2011,* 71–75.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

15. Meng, X., Pappas, V., & Zhang, L. (2010). Improving the scalability of data center networks with traffic-aware virtual machine placement. In *Proceedings of the 2010 IEEE INFOCOM* (pp. 1–9). San Diego, CA, USA.
16. Alicherry, M., & Lakshman, T. (2012). Network-aware resource allocation in distributed clouds. In *Proceedings of the 2012 IEEE INFOCOM* (pp. 963–971). Orlando, FL, USA.
17. Leitner, P., Cito, J., & Stöckli, E. (2016). Modelling and managing deployment costs of microservice-based cloud applications. In *Proceedings of the 9th International Conference on Utility and Cloud Computing* (pp. 165–174). ACM. Shanghai, China.
18. Yusoh, Z. I. M., & Tang, M. (2010). A penalty-based genetic algorithm for the composite SaaS placement problem in the cloud. In *Proceedings of the IEEE Congress on Evolutionary Computation* (pp. 1–8). Barcelona, Spain.
19. Rogers, O. (2022). Cloud SLAs punish, not compensate. Uptime Institute. Retrieved from https://journal.uptimeinstitute.com/cloud-slas-punish-not-compensate/
20. Pressman, R. S. (1982). *Software Engineering – A Practitioner's Approach* (4 ed.). McGraw-Hill. ISBN 0-07-052182-4.
21. Tulka, T. (2020). How cohesion and coupling correlate. Retrieved from https://blog.ttulka.com/how-cohesion-and-coupling-correlate/
22. Ivanchuk, Y. V., Horobets, Y. V., & Koval, K. O. (2022). Method for increasing the degree of protection in message encryption based on a time-constant algorithm. *System Technologies: Regional Interuniversity Collection of Scientific Papers, 2(139)*, 3–13. https://doi.org/10.34185/1562-9945-2-139-2022-01

Received 14.04.2025.
Accepted 16.04.2025.

Features of microservices placement of learning management systems in hybrid clouds

The article focuses on various approaches and methods for deploying microservices in the cloud, as well as the specifics of their application for hosting microservices in a hybrid cloud environment for learning management systems. The relevance of the study consists in the problem that migrating high-load learning management systems (LMS) with numerous integration links to other services to the cloud is a complex task that is difficult to be solved using existing methods. LMS'es such as Moodle, Canvas LMS, Forma LMS, and others are often used to organize the educational process in Ukrainian educational institutions. One of the most popular LMS in Ukraine is Moodle, which is used by many higher education institutions and other educational establishments. Many LMS have extensive integration with other resources of the educational institution, such as library resources, software development tools, or even proctoring systems. The overall server information system of educational institutions can be distributed across different servers or clouds and proper solutions for this process should be found. The aim of study is developing a solution for organizing data flows in the process of deploying microservices using various methods including the features of educational information systems. Using the structural analysis method, a data flow solution for the automated deployment of microservices in learning management systems within a hybrid cloud environment created. Also various methods for deploying microservices in the cloud were examined, both practical and scientific, comparing their positive and negative effects.

Key metrics for assessing network availability for microservice deployment have been identified. The proposed data flow solution for the automated deployment of microservices in a hybrid cloud provides ways to improve microservice distribution methods giving users sufficient control over the process including features of learning management systems.

Keywords: microservice, cloud, server, containerization, learning management system, distributed cloud computing.

Малініч Ілля Павлович– асистент кафедри Комп'ютерних наук, Вінницький національний технічний університет.
Іванчук Ярослав Володимирович – доктор технічних наук, професор кафедри Комп'ютерних наук, Вінницький національний технічний університет.

Malinich Illia – assistant lecturer of Computer Sciences, Vinnytsia National Technical University.
Ivanchuk Yaroslav – ScD (Eng), professor of Computer Sciences department, Vinnytsia National Technical University.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)