DOI: 10.35681/1560-9189.2025.27.1.335755

УДК 004.75

Д. С. Ардашов
Інститут проблем реєстрації інформації НАН України
вул. М. Шпака, 2, 03113 Київ, Україна

Виявлення уражень та ізоляція уражених елементів комп'ютерної системи, яку розгорнуто в хмарному середовищі

Визначено базові архітектурні функціональні особливості хмарних платформ, що забезпечують механізми ізоляції ресурсів як елементів комп'ютерної системи, яку розгорнуто в хмарному середовищі. Розглянуто вбудовані засоби моніторингу платформи AWS з метою виявлення уражень та ізоляції уражених елементів розгорнутих у платформі додатків. Ідентифіковано ризики деградації доступності хмарних додатків, спричинених спільним використанням фізичної інфраструктури провайдера, та методи запобігання їм. Наведено приклад функціонування механізмів системи розпізнавання потенційних загрозливих впливів AWS GuardDuty.

***Ключові слова**: живучість хмарних систем, моніторинг у хмарних платформах, відмовостійкі хмарні архітектури.*

Вступ

Хмарні обчислення — це інформаційний сервіс, який надається через мережу Інтернет, що включає в себе розміщення, обробку та зберігання даних. За визначенням US NIST (Національного інституту стандартів і технологій США) [1], хмарні обчислення — це модель забезпечення зручного глобального мережевого доступу до спільного пулу обчислювальних ресурсів (наприклад, мереж, серверів, сховищ, додатків і сервісів), які можна конфігурувати й швидко надавати та звільняти з мінімальними зусиллями з управління або взаємодії з постачальниками послуг. Відповідні сервіси надаються на умовах використання, а не володіння технічними ресурсами, а організації (провайдери), які здатні їх забезпечувати у світовому масштабі, ще називають *гіперскейлерами* [2].

Основою надання хмарних послуг гіперскейлерами є глобальна програмно-технічна інфраструктура, яка належить провайдеру.

Як можна побачити на зазначеній схемі (рис. 1), спільні обчислювальні ресурси хмарної платформи об'єднані в зони доступності, які є незалежними датацентрами, що поєднані відповідною мережевою інфраструктурою провайдера на

рівні регіонів. Регіони, в свою чергу, утворюють глобальну хмарну інфраструктуру гіперскейлера. Важливо відмітити, що провайдери використовують власну мережеву інфраструктуру високої пропускної здатності для зв'язку своїх регіонів доступності, що додатково забезпечує високу надійність і доступність додатків, які розгортаються в хмарі. Зазначені додатки зазвичай розбудовуються споживачами хмарних послуг у вигляді комп'ютерної системи, що складаються із взаємопов'язаних сервісів і сховищ даних. При цьому інфраструктура провайдера, здебільшого, задіяна в режимі її спільного використання завдяки технологіям віртуалізації, що є ключовою концепцією хмарних обчислень.

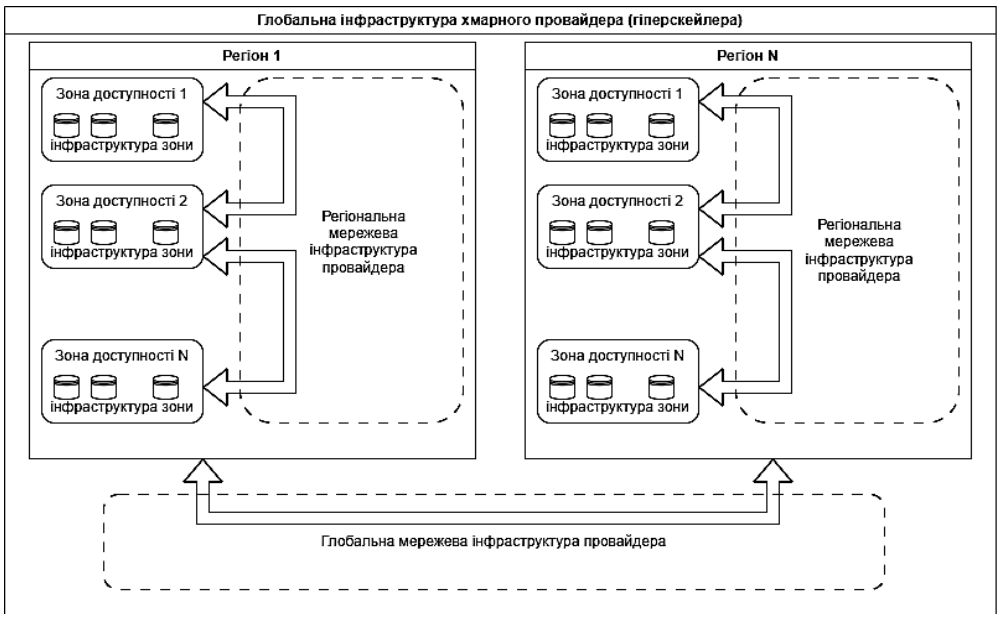

Рис. 1. Схематичне зображення глобальної інфраструктури провайдера хмарних послуг

Вирізняють дві основні технології віртуалізації: віртуалізацію на рівні фізичного обладнання (Virtual Machines) (яка за замовченням мається на увазі при використанні терміну «віртуалізація») та віртуалізація на рівні операційної системи, яка отримала назву «контейнеризація» (Containers). Схематично, згідно рівнів абстрагування фізичного обладнання, вони виглядають, як показано на рис. 2.

У випадку *віртуалізації* (Virtual Machines), фізичні ресурси обладнання за допомогою гіпервізора (hypervisor) одночасно використовуються декількома віртуальними машинами, які працюють кожна під управлінням своєї гостьової операційної системи (Guest OS), забезпечуючи ізольовані середовища виконання (Bins/Lib) для окремих додатків (App) сервісу або користувача.

Тобто, віртуалізація дозволяє створювати віртуальні версії фізичних ресурсів, таких як сервери, сховища даних і мережі. Це — абстракція на рівні апаратного забезпечення.

Контейнеризація (Containers) — це легша та більш гнучка альтернатива віртуалізації, яка віртуалізує безпосередньо операційну систему (Operation System), дозволяючи запускати декілька ізольованих просторів (контейнерів, що містять

відповідно конфігуровані середовища виконання (Bins/Lib) та користувацькі додатки (App) на одному сервері (фізичному чи віртуальному).

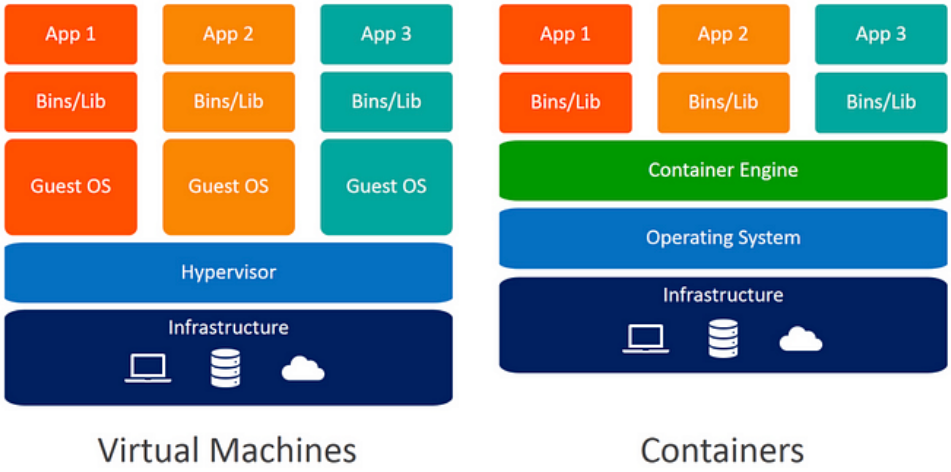

Рис.2. Віртуалізація та контейнеризація як рівні абстракції над фізичною обчислювальною інфраструктурою

Забезпечення відмовостійкості та доступності системи, розгорнутої в хмарному середовищі, вимагає своєчасного виявлення уражень та ізоляції її елементів, що має свої специфічні особливості, завдяки властивостям хмарної архітектури: масштабуванню та віртуалізації. Для такого середовища важливо розглядати підходи, які сприяють швидкому виявленню потенційних загроз та ефективній ізоляції уражених елементів системи з мінімальним впливом на інші компоненти. Дана стаття пропонує огляд відповідних технік на прикладі додатків, які розгортаються в хмарному середовищі AWS (Amazon Web Services).

Базова концепція ізоляції обчислювальних ресурсів у хмарному середовищі.

Елементами системи, яку розгорнуто на хмарній платформі, є розподілені сервіси, що виконуються у віртуалізованому середовищі платформи: на віртуальній машині у випадку віртуалізації з гіпервізором, або в контейнері, якщо застосовується контейнерна віртуалізація. Архітектурну діаграму такої умовної системи наведено на рис. 3. Концепція хмарних еластичних обчислень [1] передбачає функціонування сервісів у вигляді одночасного виконання його екземплярів у рамках автоматичної масштабованої групи (Auto Scaling Group), що, зазвичай, одночасно охоплює декілька зон доступності хмарної платформи (Availability Zones) [3]. Запит користувачів (Users) до системи скеровується до конкретного екземпляра обчислювального сервісу системою еластичного балансування навантаження (Elastic Load Balancing), відповідно до конфігураційних налаштувань останнього та враховуючи завантаженість екземплярів у групі.

Реагуючи на кількість запитів в одиницю часу, згідно заданих політик, автоматична масштабована група запускає або зупиняє екземпляри сервісу, маючи за

мету забезпечити баланс доступності системи і ефективності використання нею ресурсів хмарної платформи.

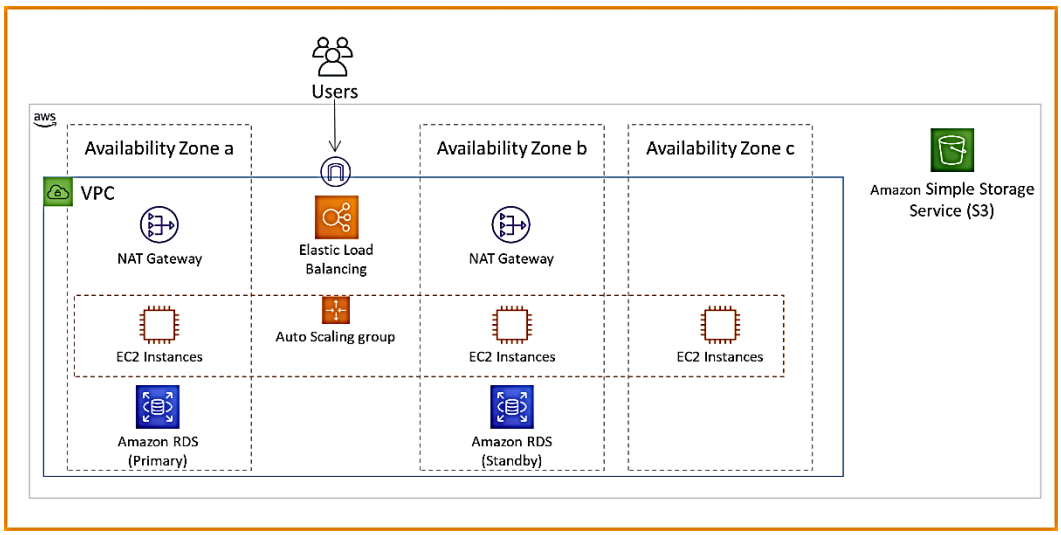

Рис. 3 Архітектурна діаграма умовної комп'ютерної системи, розгорнутої у хмарному середовищі AWS: *Auto Scaling Group* — автоматична масштабована група; *Availability Zones* — зони доступності; *Elastic Load Balancing* — система еластичного балансування навантаження; *Users* — користувачі

Завдяки концепції еластичних обчислень, яка побудована на принципі віртуалізації, екземпляри сервісів виконуються в ізоляції як на рівні віртуальної машини, контейнера, так і на рівні фізичної інфраструктури провайдера (завдяки розгортанню в різних зонах доступності).

Водночас, ураження екземпляра сервісу або відмови в його роботі можуть призвести до нестабільності роботи системи в цілому, зниження показників її живучості в результаті розповсюдження негативного впливу. Таким чином, актуальності набуває питання ідентифікації ресурсів, що піддалися негативному впливу, та їхній своєчасній примусовій ізоляції на рівні системи, зокрема шляхом задіяння зазначених механізмів масштабування.

Ідентифікація та ізоляція ресурсів, що повністю або частково втратили функціональність унаслідок їхньої безпосередньої відмови

Хмарні платформи забезпечують розробників набором різних інструментів і технік ідентифікації відхилень у роботі комп'ютерної системи, яку розгорнуто в хмарному середовищі, що забезпечує спостережність системи [4]. Своєчасне виявлення та ізоляція елементів системи, що піддаються негативному впливу, є запорукою її відмовостійкості.

Прямі методи контролю ресурсу полягають у періодичному контролі статусу екземпляра сервісу на рівні автоматичної масштабованої групи та перевірки цього статусу системою балансування навантаження при отриманні запиту до сервісу.

Система спостереження (Cloud Watch у випадку AWS)[5] безперервно аналізує логи та метрики кожного екземпляра сервісу автоматичної масштабованої гру-

пи. У разі виходу значень метрик за визначені заздалегідь застосованою політикою порогові значення, отримання негативного результату запиту щодо працездатності, система генерує сповіщення (про зміну статусу відповідного Cloud Watch Alarm у випадку AWS) та скеровує його в шину сповіщень (SNS у випадку AWS).

Автоматична масштабована група (АМГ) (Auto Scaling Group у випадку AWS), що керує розгортанням екземплярів сервісу, підписується на відповідні сповіщення в шині сповіщень і виконує необхідні дії з масштабування у випадку отримання відповідного повідомлення. Зокрема, якщо повідомлення містить відомості щодо сповіщення про відмову певного екземпляра сервісу заданої автоматичної масштабованої групи, на рівні останньої виконуються такі дії:

— АМГ відкликає реєстрацію пошкодженого екземпляра сервісу на рівні системи балансування навантаження;

— АМГ запускає новий екземпляр сервісу та реєструє його в системі балансування навантаження;

— АМГ видаляє екземпляр сервісу, що відмовив;

— система балансування навантаження скеровує вхідні запити на новий екземпляр сервісу.

Діаграма на рис. 4 ілюструє зазначений механізм ідентифікації та ізоляції екземпляра сервісу, що відмовив.

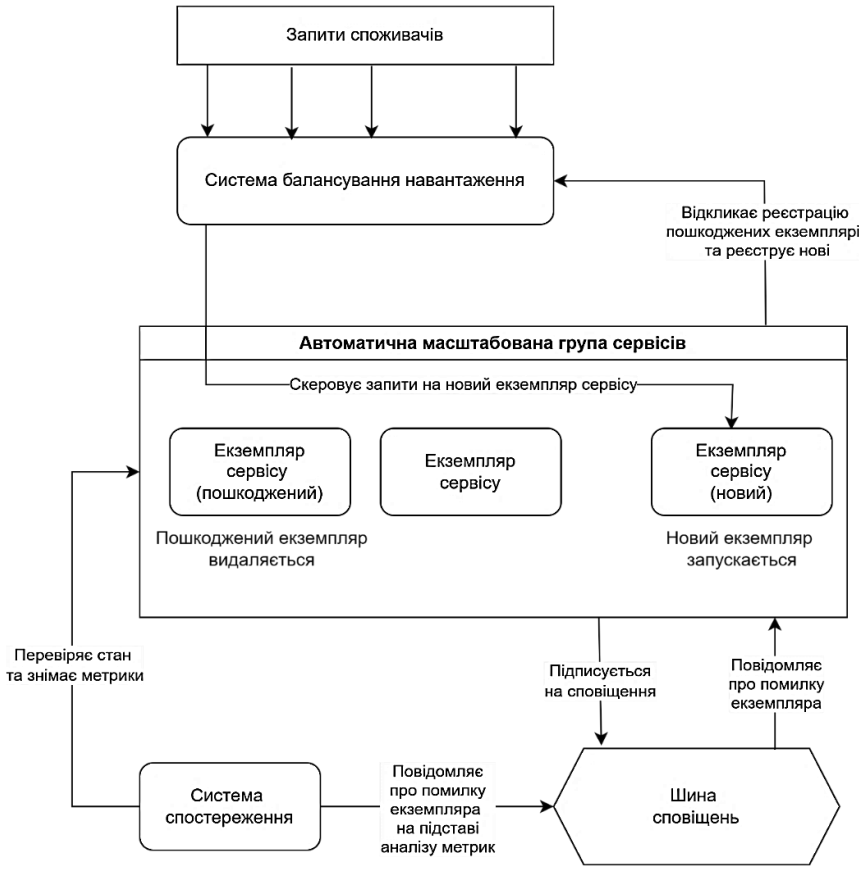

Рис. 4. Діаграма процесів контролю та керування доступністю сервісу на рівні автоматичної масштабованої групи

Ідентифікація та ізоляція ресурсів, що піддалися негативному впливу внаслідок відмови залежностей

У попередньому розділі було розглянуто механізм ідентифікації елементів системи, що представлені екземпляром сервісів, які втратили функціональність унаслідок безпосередньої відмови. Зазначений приклад демонстрував ізоляцію пошкодженого екземпляра та відновлення системи шляхом запуску нового.

Водночас, інколи мають місце ситуації, коли сервіс, який відмовив з тих чи інших причин, стає недоступним на тривалий час, що викликає каскадні помилки функціонування на рівні залежних елементів системи.

Для виявлення структури таких залежностей, побудови відповідної топології і виявлення вузьких місць функціонування системи, використовується такий важливий інструмент спостережуваності як аналіз розподілених трейсів. Стандартом хмарних платформ є використання відкритого формату OpenTelemetry [6] для трейсів, що дає гнучкість у виборі засобу їхнього аналізу з-поміж пропрієтарних рішень (AWS X-Ray для випадку AWS) [7] і розробок сторонніх постачальників (DataDog, Prometeus та інші).

Засоби аналізу трасувань безпосередньо не виконують автоматизованих дій, але допомагають у налаштуванні метрик, які, в свою чергу, підлягають моніторингу системою спостереження. В разі, якщо для таких метрик задані порогові значення, їхні перевищення можуть слугувати тригером відповідних сповіщень і запуску дій масштабування на рівні автоматичної масштабованої групи.

Гарною практикою вважається впровадження механізму «розриву ланцюга» (Circuit Breaker), щоби запобігти каскадному розповсюдженню затримки обслуговування по ланцюгу залежностей.

Даний механізм, зазвичай, є елементом систем оркестрації мікросервісів, що надаються постачальником хмарних послуг як додаткові сервіси.

Механізм «розриву ланцюга» стає проміжною ланкою поміж парами залежних сервісів і забезпечує наступні стани свого функціонування.

Замкнутий (Closed): у цьому стані Circuit Breaker дозволяє запитам проходити до захищеного сервісу. Він відстежує кількість помилкових запитів. Якщо кількість помилок не перевищує встановлений поріг, Circuit Breaker залишається в замкнутому стані.

Розімкнутий (Open): якщо кількість помилкових запитів перевищує поріг, Circuit Breaker переходить у розімкнутий стан. У цьому стані він блокує всі подальші запити до захищеного сервісу, запобігаючи його перевантаженню та поширенню помилок.

Напіввідкритий (Half-Open): через певний час Circuit Breaker переходить у напіввідкритий стан. У цьому стані він дозволяє невеликій кількості тестових запитів пройти до захищеного сервісу. Якщо ці запити успішні, Circuit Breaker повертається до замкнутого стану. Якщо ж запити знову зазнають помилки, він знову переходить у розімкнутий стан.

Проблема «галасливого сусіда» в багатокористувацькому хмарному середовищі та методи її вирішення

Як було зазначено вище, хмарні обчислення реалізують багатокористувацьку архітектуру (під користувачем тут ми маємо на увазі організацію, яка розгортає свою обчислювальну систему в хмарі), де декілька користувацьких обчислювальних систем (додатків) спільно використовують загальну фізичну інфраструктуру, залишаючись логічно ізольованими у віртуальних машинах або контейнерах.

Незважаючи на абстрагування фізичних обчислювальних, мережевих та інших ресурсів за допомогою гіпервізора (або середовища керування контейнерами), користувачі на рівні додатків фактично конкурують за спільні обчислювальні ресурси фізичного сервера, такі як процесорний час, пропускна здатність мережі та дисковий ввід/вивід (рис. 5).

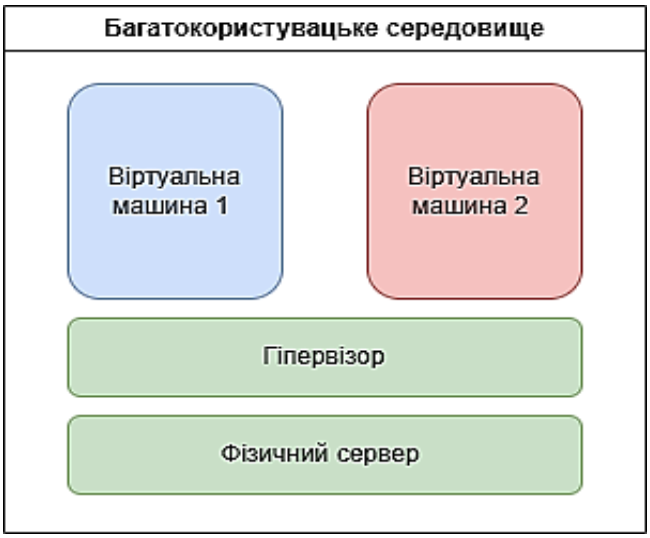

Рис. 5. Виконання додатків різних користувачів віртуальними машинами, що запущені на одному сервері

Надмірне використання ресурсів одним або кількома додатками може спричиняти деградацію продуктивності інших додатків, що відоме як *проблема «галасливого сусіда»* [8]. Нестабільність розподілу ресурсів у хмарному середовищі призводить до непередбачуваності продуктивності та негативно впливає на якість обслуговування, ускладнюючи гарантування стабільної роботи додатків.

Приклад прояву зазначеної проблеми проілюстровано на рис. 6. У наведеному прикладі елемент «Віртуальна машина 1» певного хмарного додатку починає інтенсивне використання обчислювального ресурсу фізичного сервісу. В певний проміжок часу споживання сягає величини, що спричиняє нехватку спільного ресурсу для задоволення обчислювальних потреб елемента «Віртуальна машина користувача 2» іншого додатку та призводить до загальної деградації продуктивності або недоступності останнього.

Таким чином, елементи різних, умовно незалежних обчислювальних систем (застосунків), що розгорнуті в спільному хмарному середовищі, можуть негативно

впливати один на одного. При цьому, зазначені системи можуть належати та керуватися різними організаціями. Отже, безпосередньо хмарна інфраструктура стає провідником такого негативного впливу.

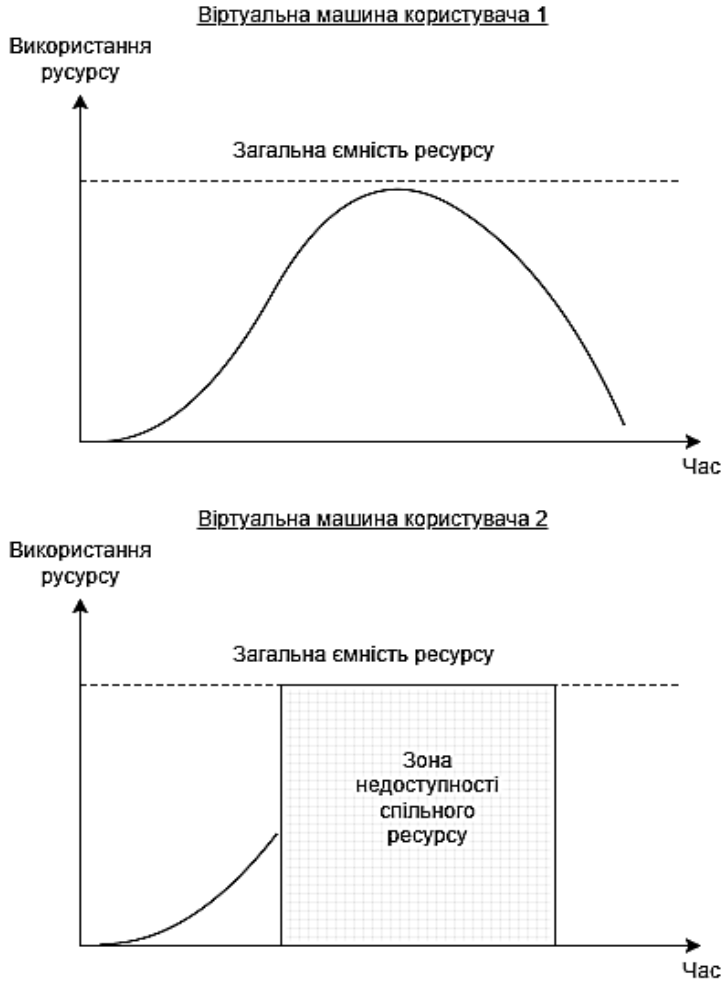

Рис. 6. Деградація продуктивності елемента хмарного додатку «Віртуальна машина користувача 2» внаслідок надмірного споживання елементом «Віртуальна машина користувача 1» спільних ресурсів сервера, на якому вони виконуються

Провайдери хмарних послуг надають свої рекомендації [9] щодо уникнення проблеми «галасливого сусіда», які, здебільшого, зводяться до:

— резервування клієнтом додаткових ресурсів;

— виокремлення окремого фізичного сервера лише для обслуговування навантажень одного клієнта;

— налаштування додаткових політик використання ресурсів сервера віртуальними сервісами;

— додаткові рекомендації щодо імплементації сервісів у хмарному середовищі.

Окремої уваги заслуговує використання технології eBPF запуску користувацького коду на рівні ядра Linux [10], зокрема ті можливості, що вона надає для

створення ефективних інструментів забезпечення спостережності хмарної обчислювальної системи.

Дана технологія була використана компанією Netflix задля надшвидкого та надійного виявлення ознак деградації сервісів, спричиненої конкурентним споживанням апаратних ресурсів загального використання [11].

Системи розпізнавання потенційних загрозливих впливів

Сучасні хмарні платформи пропонують додаткові сервіси, що послуговують підвищенню рівня експлуатаційної безпеки комп'ютерних систем, які розгорнуто у хмарному середовищі.

Прикладом такої системи є система розпізнавання потенційних загроз AWS GuardDuty [12].

AWS GuardDuty — це керований сервіс виявлення потенційно загрозливих впливів, який постійно відстежує облікові записи AWS та робочі навантаження на предмет зловмисної та/або несанкціонованої активності. Система використовує алгоритми машинного навчання та доступні дані про відомі загрози, аналізує події і журнали з різних джерел даних AWS, включаючи журнал змін конфігурації (CloudTrail), журнали потоків віртуальної приватної хмари (VPC) та журнали DNS для ідентифікації потенційних загроз безпеці, таких як скомпрометовані екземпляри сервісів і віртуальних машин, спроби несанкціонованого доступу та незвичні шаблони мережевого трафіка.

Ключові функції AWS GuardDuty включають:

— виявлення потенційних загроз у реальному часі: GuardDuty аналізує журнали подій AWS CloudTrail, журнали потоків VPC та журнали DNS у реальному часі для ідентифікації підозрілої активності та потенційних загроз безпеці;

— інтелектуальне виявлення підозрілої взаємодії із системою: GuardDuty використовує алгоритми машинного навчання та бази даних відомих зловмисних активностей для виявлення поширених шаблонів атак та індикаторів компрометації (IOC);

— централізовану інформаційну панель загроз: GuardDuty надає централізовану інформаційну панель, яка дозволяє користувачам переглядати та досліджувати виявлення потенційних загроз безпеці, визначати пріоритети заходів з відновлення та отримувати уявлення про нові ймовірні загрози;

— автоматизоване відновлення: GuardDuty інтегрується з безсерверними функціями AWS Lambda та іншими сервісами AWS для автоматизації дій у відповідь, таких як ізоляція скомпрометованих екземплярів сервісів, оновлення груп безпеки та запуск сповіщень для груп безпеки;

— виявлення ворожого та інфікованого вірусами вмісту файлових сховищ на рівні сервісів і служби зберігання об'єктів S3, ізоляція його шляхом переміщення до карантину або присвоєнням мітки, що обмежує доступ.

Приклад сценарію

1. GuardDuty виявляє, що екземпляр сервісу спілкується з відомою зловмисною IP-адресою.
2. GuardDuty генерує повідомлення про виявлення небезпеки.
3. Повідомлення надсилається до Security Hub.

4. Правило шини повідомлень (EventBridge) налаштовано на запуск безсерверної функції (Lambda), коли генерується повідомлення про виявлення небезпеки GuardDuty з певним рівнем небезпеки (наприклад, високим).

5. Функція запускається та використовує API сервісу еластичних обчислень (сервіс, що розгортає екземпляри сервісів) для зміни групи безпеки екземпляра, стосовно якого було згенеровано повідомлення про виявлення небезпеки, блокуючи весь вхідний і вихідний трафік. Скомпрометований екземпляр сервісу тепер ізольовано.

Висновки

Сучасні хмарні платформи забезпечують широкий спектр механізмів своєчасного виявлення негативних впливів на елементи комп'ютерної системи, яку розгорнуто в хмарному середовищі, та ізоляції таких елементів. Основою таких механізмів є властивість спостережності системи, що забезпечена інструментами моніторингу та, безпосередньо, концепція еластичних обчислень у хмарі.

Стандартизація телеметричних даних хмарних платформ дозволяє інтегрувати сторонні сервіси моніторингу та реагування на потенційні загрози в процесі розбудови систем забезпечення живучості хмарних рішень.

Сучасним трендом виявлення потенційних загрозливих впливів є використання інструментів машинного навчання та штучного інтелекту.

Водночас, цікавим постає питання вивчення характеру розповсюдження негативних впливів на предмет його відповідності характеру протікання хвильових процесів. Подальші дослідження в цьому напрямку можуть мати потенціал розробки нових підходів у вдосконаленні методів забезпечення живучості комп'ютерних систем, які розгорнуто у хмарному середовищі.

1. NIST SP 800-145. The NIST Definition of Cloud Computing.

2. What is hyperscale? URL: https://www.ibm.com/think/topics/hyperscale

3. Amazon EC2 Auto Scaling User Guide. URL: https://docs.aws.amazon.com/autoscaling/ec2/userguide/

4. What is observability? IBM. URL: https://www.ibm.com/think/topics/observability

5. Amazon CloudWatch User Guide. URL: https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/cloudwatch_architecture.html

6. OpenTelemetry documentation. URL: https://opentelemetry.io/docs/

7. AWS X-Ray Developer Guide. URL: https://docs.aws.amazon.com/xray/latest/devguide/aws-xray.html

8. AWS Well-Architected Framework. URL: https://docs.aws.amazon.com/wellarchitected/latest/saas-lens/noisy-neighbor.html

9. MS Azure Architecture center. URL: https://learn.microsoft.com/en-us/azure/architecture/antipatterns/noisy-neighbor/noisy-neighbor

10. eBPF Documentation. URL: https://ebpf.io/what-is-ebpf/#what-is-ebpfio

11. Noisy Neighbor Detection with eBPF. URL: https://netflixtechblog.com/noisy-neighbor-detection-with-ebpf-64b1f4b3bbdd

12. Amazon GuardDuty User Guide. URL: https://docs.aws.amazon.com/guardduty/latest/ug/what-is-guardduty.html

Надійшла до редакції 30.04.2025