УДК 004.94; 004.4; 004.62

Жаріков Е. В., Коваль А. А., Терентьєв Р. А.

ДИНАМІЧНЕ РОЗМІЩЕННЯ ВІРТУАЛЬНИХ МАШИН НА ОСНОВІ НАВЧАННЯ З ПІДКРІПЛЕННЯМ В ХМАРНИХ ЦЕНТРАХ ОБРОБКИ ДАНИХ

Проаналізовано можливість застосування методу навчання з підкріпленням для управління ресурсами хмарних центрів обробки даних. Запропоновано метод динамічного розміщення віртуальних машин на основі навчання з підкріпленням, який при виборі управляючих впливів враховує витрати електроенергії та кількість порушень угоди про надання сервісів. Розроблений алгоритм агента, який враховує зміни робочого навантаження на ресурси для прийняття рішення щодо включення або переключення в сплячий режим незавантажених фізичних серверів з метою зменшення витрат електроенергії. Запропонований агент навчання з підкріпленням базується на підході Q-learning, який дозволяє визначати наближену до оптимальної політику управління режимами роботи фізичного сервера без створення моделі середовища та попередньої інформації про навантаження.

***Ключові слова**: центр обробки даних, навчання з підкріпленням, віртуалізація, енергоефективність, sla*

Вступ

Реалізація інформаційних процесів та надання доступу до сучасних інформаційних сервісів пов’язані із використанням центрів обробки даних (ЦОД), або дата-центрів – комплексних централізованих систем для створення високопродуктивної, відмовостійкої IT-інфраструктури. До головних завдань ЦОД належать консолідоване зберігання і опрацювання даних користувачів, підтримка функціонування застосувань та надання користувачам прикладних сервісів на заданому якісному рівні.

Концепція ЦОД втілена багатьма великими компаніями для забезпечення доступу великої кількості користувачів до певних ресурсів (сервісів, застосувань, обчислювальних потужностей, даних тощо). Ефективне управління ЦОД пов’язане з необхідністю розв’язання низки проблем, насамперед створення умов для функціонування інформаційно-обчислювальних потужностей ЦОД, управління віртуалізованими ресурсами, забезпечення надійності та безпеки. Вкладаючи кошти, компанії намагаються збільшити прибуток, покращити якість сервісів, зменшити витрати на експлуатацію ЦОД, знизити вартість обслуговування користувачів, що дозволить, зрештою, закласти основу для ефективної діяльності як самої компанії, так і клієнтів.

Забезпечення рівня вимог користувачів з мінімізацією витрат становить сутність проблеми управління функціонуванням ЦОД. Зазвичай цю комплексну проблему розбивають на ряд задач менших розмірів. Однією з них є задача управління ресурсами і навантаженням ЦОД.

Необхідні гнучкі рішення, які ґрунтуються на оцінюванні стану ресурсів і обсягів навантаження та полягають у правильному розподілі навантаження і ефективному управлінні ресурсами. Для систематичного прийняття правильних рішень необхідні інструментарій та комплекс методик і алгоритмів для вирішення задач управління IT-інфраструктурою. Їх створення становить важливу науково-практичну проблему, розв’язання якої вимагає розуміння процесів, які відбуваються в хостингових компаніях, функціонування IT-інфраструктури, чіткої постановки конкретних завдань дослідження, розробки математичних моделей, моделей ЦОД і відповідних методів вирішення задачі та реалізації згаданих вище методик і алгоритмів.

Технології віртуалізації надають можливість абстрагуватися від особливостей окремих груп ресурсів, об’єднати їх у апаратно-програмні комплекси потрібної конфігурації і спростити управління ними. Віртуалізація ресурсів інфраструктури полягає у створенні віртуальних машин (ВМ) – програмних абстракцій, що запускаються на платформі фізичних апаратно-програмних (хостових) систем. Віртуалізація ресурсів узагальнює підходи до створення ВМ і переносить їх на усі види ресурсів, такі як обладнання ЦОД, простори імен, мереж та ін. [1].

Впровадження віртуалізації пов’язано з комплексом додаткових заходів, оскільки процеси прийняття рішень стають складнішими і вимагають розвиненого інструментарію їх підтримки. З’являється необхідність розроблення моделей, алгоритмів здійснення міграцій ВМ та управління інформаційно-обчислювальними процесами ЦОД з урахуванням специфіки архітектурних рішень, особливостей віртуалізації та їх реалізації у системі управління IT-інфраструктурою.

В статті пропонується метод динамічного розміщення ВМ на основі навчання з підкріпленням (НП, англ. reinforcement learning, RL) [2]. НП – один із способів машинного навчання, в ході якого програмні агенти виконують дії в певному середовищі з метою максимізації сукупної винагороди як результату дій. В дискретні моменти часу агент отримує спостереження, яке зазвичай включає і винагороду, та обирає керуючий вплив з множини доступних впливів на середовище. Середовище переходить до нового стану, і визначається винагорода, пов'язана з переходом в новий стан. Метою агента НП є отримання якомога більшої винагороди. Агент може обирати будь-який вплив або згідно політики або випадково. НП є добре пристосованим для задач, які включають компроміс між довготерміновою та короткотерміновою винагородою. НП відрізняється від стандартного навчання з учителем тим, що пари правильних входів/виходів ніколи не представляються, а неоптимальні впливи (дії) явно не виправляються.

1 Постановка задачі

В статті запропонована модель ЦОД як постачальника віртуальних ресурсів, який складається з *m* фізичних машин (ФМ) з різною конфігурацією. Кожна ФМ має процесор, який може бути багатоядерним, при цьому продуктивність вимірюється в мільйонах інструкцій на секунду. Крім того, ФМ характеризується обчислювальною потужністю процесора, об'ємом оперативної пам'яті, пропускною здатністю мережі та показниками запису/читання жорсткого диску. Користувачі відсилають завдання або запити для створення *n* ВМ. Спочатку ВМ розміщуються на фізичних машинах відповідно до запитуваних характеристик. Відсоток використання ресурсів ВМ з часом буде змінюватись. Необхідно розмістити ВМ на мінімальній кількості ФМ щоб мінімізувати витрати електроенергії та водночас не порушувати умови угоди про надання сервісів (SLA).

2 Аналіз існуючих рішень

В роботі [3] представлена децентралізована архітектура енергозберігаючої системи управління ресурсами для ЦОД. Визначені проблеми мінімізації споживання енергії при виконання вимог QoS (Quality of service – якість обслуговування) та сформульовані вимоги для політики розподілу ВМ. Запропоновано три стадії безперервної оптимізації розміщення ВМ: перерозподіл відповідно до поточного споживання декількох системних ресурсів, оптимізації віртуальних мережевих топологій, встановлених між ВМ та перерозподіл ВМ з урахуванням теплового стану ресурсів. Для першої стадії застосовані евристичні алгоритми, які були оцінені за допомогою середовища моделювання CloudSim. Запропонований алгоритм мінімізації міграцій дозволив значно зменшити споживання енергії в ЦОД.

В роботі [4] розподіл віртуалізованих ресурсів в хмарному середовищі розглядається як задача автоматичного управління із застосуванням підходу навчання з підкріпленням [2]. Автори застосували підхід Q-learning [5] до управління кількістю віртуальних машин, що забезпечують роботу хмарного застосування. Запропонований алгоритм зберігає значення пар "дія-винагорода" як історичні значення і використовує їх для прийняття рішення про додавання або вимкнення віртуальних машин під час роботи хмарного застосування. Слід зазначити, що запропонований підхід орієнтований на довгострокову роботу застосування і не враховує розміщення віртуальних машин на фізичних машинах.

В статті [6] представлено метод уніфікованого навчання з підкріпленням, що відкриває новий напрям в автоконфігуруванні ВМ та апаратних засобів в умовах хмарних обчислень. В цьому методі агент НП регулює конфігурацію ВМ щоб максимізувати свою винагороду в довгостроковій перспективі. Для прискорення процесу навчання в великомасштабних системах розроблено різні моделі апроксимації винагород від дій по конфігурації ВМ. Експериментальні результати з різним робочим навантаженням продемонстрували ефективність методу.

В публікаціях [1], [7] розглянуто підхід до управління ресурсами ЦОД, призначений забезпечити ефективне їх використання у процесі функціонування ЦОД за схемою виділених серверів. Наводяться моделі і алгоритми розподілу ресурсів, структура відповідних інструментальних засобів. Описано три варіанти системи, яка надає хостингові послуги на основі моделі виділених серверів: виділені сервери на базі сервіс-орієнтованої архітектури, виділені сервери на базі традиційної трирівневої архітектури та гібридна архітектура.

Для моделі планування при надлишку ресурсів наведені чотири задачі мінімізації витрат на підтримку частини серверів, яка забезпечуватиме підтримку запитів користувачів клієнта. Для вирішення наведених вище задач оптимального розміщення екземплярів застосувань пропонуються варіанти генетичного алгоритму (ГА) і евристичного алгоритму, який враховує специфіку критеріїв і обмежень. Крім того, пропонується комбінований алгоритм, побудований на використанні методів вирішення задач лінійного програмування і неявного перебору.

В роботі [8], на основі узагальнення накопиченого досвіду розв’язання проблеми розподілу і управління навантаженням і ресурсами центрів оброблення даних, пропонуються модифікації раніше розроблених моделей. З метою розроблення універсального алгоритму для цього класу задач авторами пропонується варіант керованого ГА. Керованість означає вплив на вибір чергових операторів формування популяції у залежності від деяких важливих її параметрів. Шляхом налаштування зазначених параметрів керований ГА можна досить швидко навчити розв’язувати різноманітні проблеми зазначеного класу, особливістю яких є складний характер взаємодії критеріїв і обмежень, що раніше часто призводило до передчасного виродження популяції. Експериментальне дослідження показало, що керований ГА дозволяє отримати кращі результати відносно базового ГА.

3 Навчання з підкріпленням

Для управління віртуалізованими ресурсами ЦОД автори пропонують безмодельний варіант агента, що заснований на методі навчання з підкріпленням Q-learning [5]. Агент НП може отримати розв’язок близький до оптимального шляхом взаємодії з динамічним середовищем без будь-якої інформації про це середовище. Структура методу НП складається з:

- простору станів S – сукупність станів середовища, які агент може сприймати;
- простору дій *A* – сукупність дій, які агент може виконати.
- винагороди *r*, яка є або позитивною реакцією середовища, або штрафом (негативна реакція середовища) за виконану агентом дію. Таким чином, метою агента є зведення до мінімуму середніх довгострокових штрафів впродовж процесу навчання.

Q-learning – це один з найпопулярніших методів НП, який застосовується у багатьох галузях досліджень. На кожній ітерації алгоритму *Q*-learning агент отримує дані про поточний стан середовища $s_t \in S$ та вибирає дію $a_t \in A$, що впливає на середовище. Після виконання дії середовище переходить до наступного стану $s' \in S$

, і агент отримує винагороду r_t. На початку наступної ітерації агент оновлює *Q*-значення на основі наступного рівняння:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_t + \gamma \min_{\alpha \in A} Q(s', a) - Q(s_t, a_t)], \quad (1)$$

де $Q(s_t, a_t)$ – очікуваний довгостроковий штраф за виконання дії $a_t \in A$ в стані $s_t \in S$; α – швидкість (темп) навчання (англ. learning rate), коефіцієнт, що показує на скільки швидко нові дані про стани будуть враховуватись на наступних кроках, $\alpha \in [0,1]$; γ – коефіцієнт знецінювання (англ. discount factor), коефіцієнт, що визначає важливість майбутніх винагород, $\gamma \in [0,1]$; $\min_{\alpha \in A} Q(s', a)$ – оцінка оптимального майбутнього *Q*-значення.

Якщо $\alpha = 0$, то агент не навчається, якщо $\alpha = 1$, то агент використовує дані про найновіші управляючі впливи. Якщо $\gamma = 0$, то агент розглядає лише поточні винагороди, якщо $\gamma = 1$, то агент прагне до довгострокового мінімального штрафу.

Коли агент знову отримає стан $s_t \in S$, він вибере дію з мінімальним *Q*-значенням. Політика π вибору найкращої дії у стані $s_t \in S$ вираховується наступним рівнянням:

$$\pi(s_t) = \min_{\alpha \in A} Q(s_t, a), \quad (2)$$

Таким чином, метою агента НП є знаходження оптимальної політики відображення $S \to A$.

Алгоритм *Q*-learning має наступні етапи:

Крок 1. Для кожної пари $s \in S$ та $a \in A$, ініціалізувати *Q*-значення, що дорівнює нескінченності;

Крок 2. Отримати поточний стан $s_t \in S$;

Крок 3. Обрати дію $a_t \in A$ випадковим чином або за допомогою (2);

Крок 4. Виконати дію $a_t \in A$;

Крок 5. Отримати винагороду r_t;

Крок 6. Отримати новий стан $s' \in S$ та оновити $Q(s_t, a_t)$ за допомогою (1);

Крок 7. $s \leftarrow s'$;

Крок 8. Повернутися до кроку 2.

Існує два способи вибору дії з простору можливих дій у кожному стані:

– вибір випадкової дії може відбуватися на початку навчання, коли оптимальні дії ще не відомі;
– дія вибирається відповідно до вивченої політики π.

4 Модель ЦОД

Модель ЦОД показана на Рис. 1. Структура складається з набору ФМ, кожна з яких характеризується апаратною та операційною платформами і має фіксовану кількість ресурсів. Кожна ФМ може містити певну кількість ВМ.

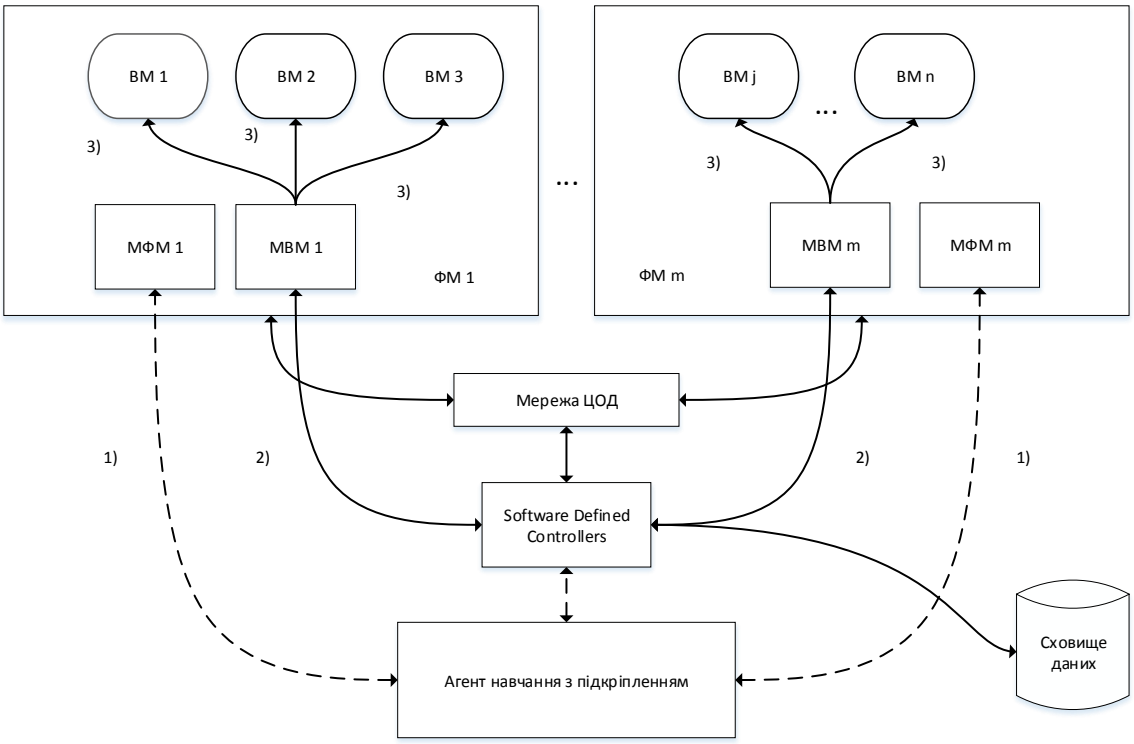

Рис. 1 – Модель ЦОД

ЦОД орієнтований на обробку пакетних та транзакційних завдань. Вони вирішуються в рамках трьох основних хмарних моделей IaaS, PaaS та SaaS. Модель IaaS є основною для обслуговування сучасних ЦОД. Зберігання даних забезпечується централізованим сховищем даних.

Забезпечення ресурсами здійснюється на основі їх наявності для уникнення перерозподілу ресурсів ЦОД. При цьому зміни клієнтських вимог до ресурсів можуть збільшуватися до певного рівня не впливаючи на інших клієнтів та у відповідності до SLA. Сучасна практика перерозподілу ресурсів з метою попередження небажаних наслідків, спричинених браком ресурсів із збільшенням навантаження та потреб клієнтів, призводить до збільшення споживання енергії та експлуатаційних витрат.

Послідовність операцій алгоритму динамічного розміщення ВМ:

1) спостереження агентом навчання за поточним станом фізичних машин. Агент НП отримує інформацію про поточні стани ФМ від менеджерів фізичних машин (МФМ), після надходження завдань або запитів на розміщення ВМ від користувачів. Ця інформація являє собою поточне використання ресурсів ФМ. Після цього агент обирає режим роботи для кожної ФМ на основі алгоритму *Q*-learning.

2) надсилання керуючих впливів до моніторів віртуальних машин (МВМ). Оптимізація розміщення ВМ відбувається в залежності від рішення агента щодо режиму роботи фізичних машин. Якщо агент обрав сплячий режим, то всі ВМ на ФМ повинні мігрувати до інших ФМ. Вибір віртуальної машини для міграції з перевантаженої ФМ відбувається відповідно до політики вибору. Політика полягає у виборі ВМ, яка вимагає мінімального часу міграції, ніж інші віртуальні машини на ФМ. Час міграції розраховується шляхом ділення пам'яті, призначеної для ВМ, на доступну пропускну здатність мережі між поточною і цільовою ФМ. Після вибору віртуальної машини відбувається пошук цільової ФМ.

3) команди міграції ВМ. МВМ надсилають команди міграції до ВМ, які повинні мігрувати на інші ФМ.

Математична модель. Наступні змінні визначені для динамічної моделі ЦОД між моментами часу t та $t+1$.

Режим роботи *i*-ої ФМ позначається як $z_i(t) \in \{0,1\}$, $i = \overline{1,m}$, якщо $z_i(t) = 0$, то *i*-а ФМ знаходиться в сплячому режимі, якщо $z_i(t) = 1$, то *i*-а ФМ знаходиться в активному режимі роботи.

Використання *k*-го ресурсу *j*-ою ВМ на *i*-ій ФМ позначається як $res_{ij}^k(t) \in [0,1]$ $i = \overline{1,m}$, $j = \overline{1,n}$, $k \in K$, де *K* це кількість типів ресурсів, таких як обчислювальна потужність процесора, об'єм оперативної пам'яті, пропускна здатність мережі та показник запису/читання жорсткого диску. Значення $res_{ij}^k(t)$ нормалізується відносно найбільшого об'єму *k*-го ресурсу ФМ. Використання *k*-го ресурсу на *i*-ій ФМ позначається як $Res_i^k(t) \in [0,1]$, $i = \overline{1,m}$, $k \in K$.

Змінна $w_{ij}^h(t) \in \{0,1\}$, $h = \overline{1,H}$ показує, чи працює *j*-а ВМ типу *h* на *i*-ій ФМ ($w_{ij}^h(t) = 1$) чи не працює ($w_{ij}^h(t) = 0$). *H* це кількість типів ВМ. Тип *j*-ї ВМ, що працює на *i*-ій ФМ позначається як $h_{ij}(t)$.

Об'єм *k*-го ресурсу, необхідна для *j*-ої ВМ, позначається як $c_j^k(t) \in [0,1]$ та нормалізується відносно найбільшого об'єму *k*-го ресурсу ФМ. Використання *k*-го ресурсу *j*-ою ВМ не повинно перевищувати необхідну ємність $res_j^k(t) \le c_j^k(t)$ окрім випадків, коли змінено тип ВМ.

Об'єм *k*-го ресурсу *i*-ої ФМ позначається як $C_i^k(t) \in [0,1]$ та нормалізується відносно найбільшого об'єму *k*-го ресурсу ФМ. Значення нормалізується таким чином, що найбільший об'єм *k*-го ресурсу ФМ рівний 1.

Позначимо через $u_{ij}(t) \in \{0,1\}$ міграцію *j*-ої ВМ з і-ої ФМ. Міграція відбулася тоді, коли $u_{ij}(t) = 1$.

Позначимо через $x_{ij}(t) \in \{0,\ldots,m\}$ індекс ФМ до якої мігрує *j*-а ВМ з *i*-ої ФМ Якщо *j*-а ВМ не мігрує, тоді $x_{ij}(t) = i$.

Позначимо через $y_i(t) \in \{0,\ldots,n\}$ індекс ВМ, яка мігрує з *i*-ої ФМ до ФМ з індексом $x_{ij}(t)$. Якщо *j*-а ВМ не мігрує, тоді $y_i(t) = j$.

Позначимо через $v_j(t) \in \{0,1\}$ зміну типу *j*-ої ВМ. Якщо тип j-ої ВМ потрібно змінити, то $v_j(t) = 1$.

Позначимо через $a_i(t) \in \{-1,0,1\}$ зміну режиму роботи *i*-ої ФМ. Якщо $a_i(t) = -1$, тоді *i*-у ФМ потрібно перевести до сплячого режиму, якщо $a_i(t) = 1$, тоді *i*-у ФМ потрібно перевести до активного режиму, якщо $a_i(t) = 0$, тоді режим роботи *i*-ої ФМ не потрібно змінювати.

Динаміка *i*-ої ФМ може бути виражена наступним чином:

$$z_i(t+1) = z_i(t) + a_i(t) \tag{3}$$

Динаміка *j*-ої ВМ в *i*-ій ФМ може бути виражена наступним чином:

$$\begin{aligned} &w_{ij}(t+1) = w_{ij}(t)(1-u_{ij}(t)) + w_{x_{ij}(t)y_{ij}(t)}(t)u_{ij}(t) \\ &u_{ij}(t) = 0 \mid w_{ij}(t) = 0 \end{aligned} \tag{4}$$

Динаміка використання *k*-го ресурсу *j*-ої ВМ в *i*-ій ФМ може бути виражена наступним чином:

$$res_{ij}^{k}(t+1) = w_{ij}(t)(res_{new}^{k}(t) + res_{ij}^{k}(t))(1-u_{ij}(t)) + res_{x_{ij}(t)y_{ij}(t)}^{k}(t)u_{ij}(t) \quad (5)$$

Зміна типу *j*-ої ВМ може бути виражена наступним чином:

$$h_j(t+1) = (w_{ij}(t)(h_{ij}(t)(1-v_j(t)) + h_{new}v_j(t)))(1-u_{ij}(t)) + h_{x_{ij}(t)y_{ij}(t)}(t)u_{ij}(t) \quad (6)$$

Використання ресурсів *i*-ої ФМ для кожного ресурсу *k* повинно бути обмеженим. Таким чином, повинна виконуватись умова (7):

$$\sum_{j=1}^{N} res_{ij}^{k}(t) \le C_i^k \quad (7)$$

Кількість ФМ, що працюють в поточний момент часу *t* позначимо як $M_{PM}(t) = \sum_{i=1}^{m} z_i(t)$, $M_{PM}(t) < m$. Мінімальне значення $M_{PM}(t)$ свідчить про кращу якість розташування ВМ.

Кількість ВМ, що розгорнуто в поточний момент часу *t* позначимо як $N_{VM}(t) = \sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij}(t)z_i(t)$, $N_{VM}(t) < n$.

Кількість ВМ, що знаходяться в стані міграції в поточний момент часу *t* позначимо як $N_{mig}(t) = \sum_{i=1}^{m}\sum_{j=1}^{n} u_{ij}(t)z_i(t)$, $N_{mig}(t) < n$.

Кількість ВМ, що розгорнуто в поточний момент часу *t* в *i*-ій ФМ позначимо як $N_i(t) = \sum_{j=1}^{n} w_{ij}(t)$.

Метою управління віртуалізованими ресурсами ЦОД є керування режимами роботи фізичних машин через змінну $a_i(t)$, визначення значень змінних розташування ВМ $x_{ij}(t)$, $y_{ij}(t)$ та значення змінної міграцій $u_{ij}(t)$ для мінімізації загального штрафу з точки зору споживання електроенергії та кількості порушень SLA.

5 Метод динамічного розміщення ВМ на основі навчання з підкріпленням

В роботі авторами запропоновано метод динамічного розміщення ВМ на основі НП з метою зменшення витрат електроенергії в ЦОД та зменшення кількості порушень SLA. SLA включає в себе вимоги та обмеження щодо якості послуг, що надаються провайдером: часу відгуку, доступності, пропускної здатності, безпеки та ін. SLA виконується для кожного клієнта, коли вся продуктивність, яку потребують застосунки всередині ВМ, забезпечується в будь-який час.

ВМ розміщуються на мінімальній кількості ФМ відповідно до поточних затребуваних ресурсів. Коли використання ресурсів на ФМ є низьким, всі ВМ перерозподіляються на інші ФМ, а недостатньо завантажена ФМ переходить до сплячого режиму. Крім того, коли ФМ перевантажена, деякі ВМ повинні бути переміщені, щоб зменшити кількість порушень SLA. Якість алгоритму розміщення ВМ може поліпшуватися в процесі роботи агента НП та алгоритму *Q*-learning для визначення режиму роботи кожної ФМ (сплячий або активний).

Алгоритм може динамічно адаптувати ФМ до зміни робочого навантаження. Важливою частиною алгоритму є вирішення питання про те, чи (1) додаткова ФМ повинна забезпечити ефективне використання ресурсів зі збільшенням робочого навантаження, або (2) резервні ФМ можуть бути переведені в сплячий режим або (3) чи достатня поточна кількість ФМ. Агент НП вважається важливою частиною алгоритму для прийняття такого рішення.

ФМ, які задіяні в процесі оптимізації, позначаються наступними мітками:

- HIBERNATE – позначаються ФМ, які потрібно перевести до сплячого режиму;
- PLACEMENT – позначаються ФМ, які задіяні в процесі розміщення ВМ;
- AVAILABLE – позначаються ФМ, які доступні для процесу розміщення ВМ.

Алгоритм динамічного розміщення ВМ на основі НП наведено на Рис. 2. Результатом роботи алгоритму є список віртуальних машин L^{VM}, які потрібно розмістити в ЦОД.

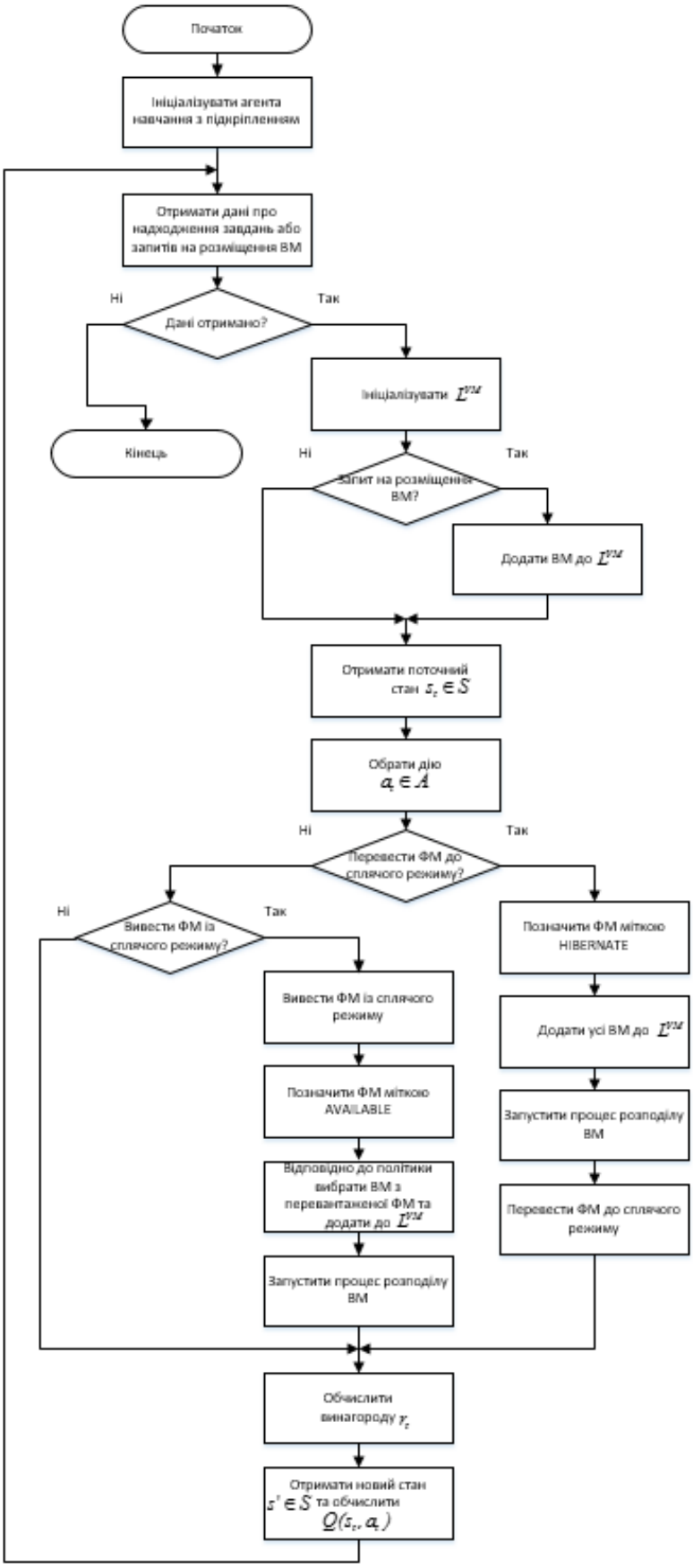

Рис. 2 – Алгоритм динамічного розміщення ВМ на основі НП

Алгоритм розподілу та міграцій ВМ наведено в Алгоритмі 1. Вибирається список L^{PM}, який містить ФМ з міткою AVAILABLE. До кожної ФМ із списку L^{PM} надсилаються вимоги до обчислювальної потужності процесора, об’єму оперативної пам'яті, пропускної здатності мережі та показника запису/читання жорсткого

диску вибраної ВМ зі списку L^{VM}. Якщо ФМ має ресурси для ВМ, то така ФМ позначається міткою PLACEMENT та повертає прогнозоване навантаження. Якщо жодна ФМ не має ресурсів для ВМ, то ВМ повертається в кінець списку L^{VM}. В іншому випадку ВМ мігрує до ФМ з максимальним навантаженням. Всі інші ФМ позначаються міткою AVAILABLE. Якщо ВМ вдруге обирається зі списку L^{VM} і жодна ФМ знову не має ресурсів для ВМ, то така ВМ видаляється зі списку L^{VM} та залишається на поточній ФМ.

Алгоритм 1

Вхід: L^{VM}

Вихід: рішення про розміщення ВМ на ФМ

1. **while** L^{VM} is not empty **do**
2. Вибрати VM з L^{VM};
3. Ініціалізувати L^{PM};
4. Вибрати p ФМ з міткою AVAILABLE та додати їх до L^{PM};
5. $LT^{PM} \leftarrow NULL$;
6. $Res_{VM} \leftarrow \{CPU_{VM}, RAM_{VM}, NET_{VM}, STIO_{VM}\}$;
7. **for** $i = 1$ **to** p **do**
8. Надіслати вимоги Res_{VM} до L_i^{PM};
9. Отримати відповідь від L_i^{PM};
10. **if** L_i^{PM} має ресурси для VM **then**
11. Позначити L_i^{PM} міткою PLACEMENT;
12. Отримати прогнозовані значення Res_{PM} від L_i^{PM};
13. $LT^{PM} \leftarrow L_i^{PM}$;
14. **end if**
15. **end for**
16. **if** LT^{PM} is empty **then**
17. **if** VM вдруге вибрана зі списку L^{VM} **then**
18. Видалити VM зі списку L^{VM};
19. **else**
20. Повернути VM в кінець списку L^{VM};
21. **else**
22. Вибрати ФМ з LT^{PM}, що підходить для розміщення VM;
23. Розмістити VM на ФМ;
24. Позначити всі інші ФМ в LT^{PM} міткою AVAILABLE;
25. **end if**
26. **end while**

Виконується декілька перевірок перед тим, як ФМ підтвердить можливість розміщення ВМ. Повинна виконуватись вимога (8) для кожного ресурсу:

$$Res_{PM}^{AVAIL} > Res_{VM}, \tag{8}$$

де Res_{PM}^{AVAIL} – доступна кількість ресурсів ФМ, Res_{VM} – вимоги до кількості ресурсів з боку ВМ.

ФМ починає процес розміщення ВМ, якщо виконується вимога (9) для кожного ресурсу:

$$Res_{PM}^{MAX} > Res_{PM}^{P}, \tag{9}$$

де Res_{PM}^{MAX} – максимальна кількість ресурсів ФМ, Res_{PM}^{P} – прогнозоване навантаження.

Рівняння (10) визначає поточну ємність ЦОД, яка використовується для визначення динаміки використання ресурсів:

$$CAP_{Res}^{DC} = \frac{\sum_{j=1}^{n} Res_{VM}^{j}}{\sum_{i=1}^{m} Res_{PM}^{i}}, \tag{10}$$

де n – кількість ВМ в певний проміжок часу, m – кількість ФМ, Res_{VM}^{j} – вимоги до кількості ресурсів j-ї ВМ, Res_{PM}^{i} – поточне використання ресурсів i-ї ФМ.

6 Модель агента навчання з підкріпленням

Ефективний метод розміщення повинен зменшити кількість активних ФМ відповідно до поточного навантаження на ФМ ЦОД. Потрібно приймати рішення про те, коли перевести ФМ в сплячий або активний режим. З цієї причини пропонується агент НП як важлива частина алгоритму динамічного розміщення ВМ. Агент обирає режим ФМ на основі попередніх даних і отримує винагороду при зміні стану середовища. З цією метою агент вивчає політику виявлення режиму ФМ на вхідних запитах та налаштовує відповідну політику за допомогою *Q*-learning.

В момент часу *t* агент сприймає поточний стан усіх ФМ, виконує пошук дії за допомогою алгоритму *Q*-learning та відправляє знайдену дію на виконання до МФМ (Рис. 1). Кожен елемент з простору станів *S* представляє поточну обчислювальну потужність процесорів, об'єм оперативної пам'яті, пропускну здатність мережі та показник запису/читання жорсткого диску усіх ВМ на кожній ФМ:

$$s_t = \{\{CPU_{PM1}, RAM_{PM1}, NET_{PM1}, STIO_{PM1}\}, \{CPU_{PM2}, RAM_{PM2}, NET_{PM2}, STIO_{PM2}\}, ..., \{CPU_{PMi}, RAM_{PMi}, NET_{PMi}, STIO_{PMi}\}, \{CPU_{PMm}, RAM_{PMm}, NET_{PMm}, STIO_{PMm}\}\},$$

де $i = \overline{1,m}$, m – кількість ФМ, $CPU \in [0,1]$ – обчислювальна потужність процесора, $RAM \in [0,1]$ – використаний об'єм оперативної пам'яті, $NET \in [0,1]$ – показник завантаженості мережі, $STIO \in [0,1]$ – показник запису/читання жорсткого диску. Кожен показник використання ресурсу нормалізується відносно максимального об'єму відповідного ресурсу ФМ.

Для зменшення кількості станів авторами пропонується розбити показник використання кожного ресурсу на чотири інтервали:

$$CPU \in \{[0,0.25),[0.25,0.5),[0.5,0.75),[0.75,1]\};$$
$$RAM \in \{[0,0.25),[0.25,0.5),[0.5,0.75),[0.75,1]\};$$
$$NET \in \{[0,0.25),[0.25,0.5),[0.5,0.75),[0.75,1]\};$$
$$STIO \in \{[0,0.25),[0.25,0.5),[0.5,0.75),[0.75,1]\}.$$

Тобто, зміна показників в межах інтервалів не призведе до переходу в інший стан.

Агент виконує дію на основі спостережуваного стану середовища. Простір дій визначається як набір $A = \{a_1(t), a_2(t), ..., a_i(t), a_m(t)\}$, де $a_i(t) \in \{-1,0,1\}$, $i = \overline{1,m}$. Кожна дія з A переводить ФМ у сплячий або активний режим роботи до наступного інтервалу часу $t+1$. Алгоритм динамічного розміщення ВМ переводить кожну ФМ у режим роботи на основі рішення агента. Потім агент отримує винагороду та вираховує *Q*-значення після зміни всіх режимів роботи до початку інтервалу часу $t+1$, але після завершення виконання обраної дії. Основною метою алгоритму динамічного розміщення ВМ є мінімізація споживання електроенергії та кількості порушень SLA, обчислюючи значення штрафу *P*, яке складається з двох значень: штраф за порушення SLA та споживання електроенергії.

$$P_t = \alpha \left| P_t^{SLA} \right|^2 + \beta \left| P_t^{power} \right|^2, \tag{11}$$

де α та β – ваги, що визначають відносну важливість P_t^{SLA} та P_t^{power} відповідно.

Штраф за порушення SLA. Досягнення бажаних вимог QoS є надзвичайно важливим для середовища хмарних обчислень. Вимоги QoS зазвичай визначаються в термінах SLA, які описують такі характеристики, як пропускна спроможність або час відгуку. Оскільки ці характеристики можуть змінюватися для різних застосунків, необхідно визначити метрику, яка не залежить від робочого навантаження і може бути використана для оцінки SLA будь-якої ВМ, що розгортається в ЦОД.

Якщо фактичне використання *k*-го ресурсу *j*-ою ВМ на *i*-ій ФМ $res_{ija}^k(t) \in [0,1]$ більше ніж очікуване використання *k*-го ресурсу $res_{ije}^k(t) \in [0,1]$ за інтервал часу, то це вважається порушенням SLA:

$$res_{ija}^k(t) > res_{ije}^k(t)$$

Штраф за порушення SLA розраховується шляхом ділення сумарної кількості порушень SLA по усім ресурсам після виконання дії $a_t \in A$ на сумарну кількість порушень на попередньому кроці перед виконанням дії:

$$P_t^{SLA} = \sum_{k=1}^{K} \delta^k \left(\frac{count_{SLA}^k(t)}{count_{SLA}^k(t-1)} \right), \tag{12}$$

де δ^k – вага, що визначає відносну важливість *k*-го ресурсу, $\sum_{k=1}^{K} \delta^k = 1$; $count_{SLA}^k(t)$ – кількість порушень SLA по *k*-му ресурсу після виконання дії $a_t \in A$; $count_{SLA}^k(t-1)$ – кількість порушень SLA по *k*-му ресурсу на попередньому кроці перед виконанням дії $a_t \in A$.

Якщо $P_t^{SLA} < 1$ то це означає, що агент вибрав правильну дію, щоб мінімізувати кількість порушень SLA.

Штраф за збільшення споживання електроенергії. Штраф розраховується шляхом ділення значення споживання електроенергії в поточному інтервалі часу на значення споживання електроенергії попереднього інтервалу часу. Отже, P_t^{power} являє собою загальну суму споживання електроенергії усіх ФМ:

$$P_t^{power} = \sum_{j=1}^{n} \left(\frac{power(t)}{power(t-1)} \right), \tag{13}$$

де $power(t)$ – значення споживання електроенергії в поточному інтервалі часу; $power(t-1)$ – значення споживання електроенергії попереднього інтервалу часу.

Усі розміщення та розподіли ВМ, а також зміна режиму роботи фізичної машини, повинні завершитися до початку наступного інтервалу управління *t+1*. Тоді *Q*-значення для кожної пари стан-дія часового інтервалу оновлюється через загальну суму штрафів P_t.

Q-значення пари стан-дія $Q(s_t, a_t)$ являє собою очікувані сумарні витрати електроенергії та кількості порушень SLA, спричинені дією, прийнятою у стані $s_t \in S$. Коли агент наступного разу спостерігає за станом, він вибирає режим роботи ФМ, який забезпечує мінімальне *Q*-значення. Агент НП вибере дію, що має найменше *Q*-значення (кількість порушень SLA і витрати електроенергії). Алгоритм роботи агента НП на інтервалі управління від *t* до *t*+1 має наступні етапи:

Крок 1. Для кожної пари $s \in S$ та $a \in A$, ініціалізувати *Q*-значення, що дорівнює нескінченності (виконується один раз на початку роботи алгоритму);

Крок 2. Отримати поточний стан $s_t \in S$;

Крок 3. Обрати дію $a_t \in A$ на основі статичного порогу або за допомогою рівняння (2);

Крок 4. Виконати дію $a_t \in A$;

Крок 5. Обчислити штраф за порушення SLA P_t^{SLA} за допомогою рівняння (12);

Крок 6. Обчислити штраф за збільшення споживання електроенергії P_t^{power} за допомогою рівняння (13);

Крок 7. Обчислити сумарний штраф P_t після зміни режиму роботи, використовуючи рівняння (11);

Крок 8. Отримати новий стан $s' \in S$ та оновити значення $Q(s_t, a_t)$ за допомогою рівняння (1).

Крок 9. $s \leftarrow s'$;

Крок 10. Повернутися до кроку 2.

Під час початку процесу навчання та кожного разу, коли агент не відвідував поточний стан раніше, пропонується застосовувати дії, що враховують нижнє або верхнє порогові значення використання ресурсів, замість вибору *Q*-значенньь, що дорівнюють нескінченності. Поріг є більш ефективним ніж випадковий вибір у звичайному Q-learning. Якщо використання ресурсів ФМ не перевищує, наприклад, 40% від загальної кількості доступних ресурсів, то агент переводить ФМ в сплячий режим роботи. Також, на ранніх стадіях навчання агента, перспективним є використання підходу НП одночасно з іншими політиками управління, наприклад [3].

9 Висновок

Проаналізовано можливість застосування методу навчання з підкріпленням для управління ресурсами хмарних центрів обробки даних. Запропоновано метод динамічного розміщення віртуальних машин на основі навчання з підкріпленням, який при виборі управляючих впливів враховує витрати електроенергії та кількість порушень SLA. Перевагою методу динамічного розміщення ВМ є здатність виконувати в режимі онлайн розміщення нових віртуальних машин одночасно з перерозподілом вже працюючих віртуальних машин.

Розроблений алгоритм агента, який, через сприймання стану фізичних машин та віртуальних машин, враховує зміну робочого навантаження на ресурси для прийняття рішення щодо включення або переключення в сплячий режим незавантажених фізичних машин з метою зменшення витрат електроенергії. Запропонований агент навчання з підкріпленням базується на підході *Q*-learning, який дозволяє визначати наближену до оптимальної політику управління режимами роботи фізичного сервера без створення моделі середовища та попередньої інформації про навантаження. Запропоновані моделі використовуються для моделювання процесів управління ресурсами хмарного ЦОД.

Література

1. Теленик С.Ф. Управління навантаженням і ресурсами центрів оброблення даних при виділених серверах [Текст]: /С.Ф. Теленик, О.І. Ролік, М.М. Букасов, Р.В. Римар, К. О. Ролік.– Автоматика. Автоматизація. Електротехнічні комплексі та системі. – 2009. – №2 (24). – С.122 – 136.
2. Sutton R.S. Reinforcement Learning: An Introduction [Текст]: /R.S.Sutton, A.G.Barto //MIT Press.– 1998. – c.360
3. Beloglazov A. Energy Efficient Resource Management in Virtualized Cloud Data Centers [Текст]: матеріали of 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, The University of Melbourne, Australia, 2010: тези / Anton Beloglazov, Rajkumar Buyya. – С.826 – 831.
4. Dutreilh X. From data center resource allocation to control theory and back [Текст]: /X. Dutreilh, A. Moreau, J. Malenfant, N. Rivierre, and I. Truck //Cloud Computing.– 2010. – С.410 – 417.

5. Watkins C. J. C. H., Dayan P. Technical note: Q-learning [Текст]: /C. J. C. H. Watkins and P. Dayan //Machine Learning.– 1992. – №3(8). – С.279–292.
6. Cheng-Zhong Xu. URL: A Unified Reinforcement Learning Approach for Autonomic Cloud Management [Текст]: Department of Electrical & Computer Engineering Wayne State University, Detroit, Michigan: тези / Cheng-Zhong Xu, Jia Rao, Xiangping Bu. – С.1 – 15.
7. Теленик С.Ф. Управління ресурсами центрів оброблення даних [Текст]: /С.Ф. Теленик, О.І. Ролік, М.М. Букасов, К. Крижова //Вісник Львів. УН-ТУ. – Серія прикл. матем. інформ., 2009. – Вип. 15. – С.325 – 340.
8. Теленик С.Ф. Генетичні алгоритми вирішення задач управління ресурсами і навантаженням центрів оброблення даних [Текст]: /С.Ф. Теленик, О.І. Ролік, М.М. Букасов, С.А. Андросов.– Автоматика. Автоматизація. Електротехнічні комплекси та системи. – 2010. – №1 (25). – С.106 – 120.
9. Telenyk S. Architecture and Conceptual Bases of Cloud IT Infrastructure Management [Текст]: /S. Telenyk, E. Zharikov, O. Rolik //Advances in Intelligent Systems and Computing.– 2017. – Vol. 512. – С. 41–62.

Проанализирована возможность применения метода обучения с подкреплением для управления ресурсами облачных центров обработки данных. Предложен метод динамического размещения виртуальных машин на основе обучения с подкреплением, который при выборе управляющих воздействий учитывает затраты электроэнергии и количество нарушений соглашения о предоставлении сервисов. Разработан алгоритм агента, который учитывает изменения рабочей нагрузки на ресурсы для принятия решения о включении или переключение в спящий режим незагруженных физических серверов с целью уменьшения затрат электроэнергии. Предложенный агент обучения с подкреплением базируется на подходе Q-learning, который позволяет определять приближенную к оптимальной политику управления режимами работы физического сервера без создания модели среды и предварительной информации о нагрузке.
Ключевые слова: центр обработки данных, обучение с подкреплением, виртуализация, энергоэффективность, SLA

The possibility of application of the reinforcement learning method to cloud data center resources management is analyzed. The method of dynamic virtual machine placement based on reinforcement learning is proposed. The proposed method takes into account the power consumption and the number of SLA violations while producing control impacts. The proposed agent's algorithm takes into account the changes in the resource utilizations to make a decision on whether or not to switch underloaded physical servers to the sleep mode in order to reduce the power consumption. The proposed reinforcement learning agent is based on the Q-learning approach which allows to determine the optimal policy for managing the physical servers without creating an environment model and preliminary information about the workload.
Keywords: data center, reinforcement learning, virtualization, energy efficiency, SLA

Інформація про авторів.

Жаріков Едуард В'ячеславович, кандидат технічних наук, доцент, докторант Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського".

http://orcid.org/0000-0003-1811-9336

Коваль Андрій Анатолійович, студент факультету інформатики та обчислюваної техніки Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського".

Терентьєв Роман Анатолійович, студент факультету інформатики та обчислюваної техніки Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського".