УДК 004.94; 004.4; 004.62

Е.В. Жаріков

КЕРУВАННЯ РЕСУРСАМИ ХМАРНИХ ЦЕНТРІВ ОБРОБКИ ДАНИХ НА ОСНОВІ ЕВРИСТИЧНОГО ПОШУКУ

Проаналізовано особливості хмарного центру обробки даних (ЦОД) з точки зору керування ресурсами. Для вирішення задачі керування ресурсами хмарного центру обробки даних запропоновано і досліджено двостадійний метод консолідації віртуальних машин на базі використання локального променевого пошуку. В статті проаналізовано роботу евристики першої та другої стадій запропонованого методу, розроблений алгоритм променевого пошуку для вирішення задачі керування ресурсами. Для аналізу роботи методу використані дані про надходження завдань в кластер Google. Запропонований метод дозволяє переключити в режим зниженого енергоспоживання в середньому 56 відсотків фізичних серверів, що потенційно визначені для переключення в режим сну за допомогою верхньої оцінки необхідної ємності ресурсів. Перерозподіл віртуальних машин виконується з урахуванням обмеження допустимої кількості міграцій на один фізичний сервер.

Ключові слова: віртуалізація, керування ресурсами, хмарні обчислення, евристичний пошук.

Вступ

Керування ресурсами хмарного ЦОД – важлива задача, вирішення якої в сучасних умовах забезпечує роботу таких сервісних моделей хмарних обчислень, як інфраструктура як сервіс (англ. IaaS), платформа як сервіс (англ. PaaS) та застосунки як сервіс (англ. SaaS). Для досягнення бажаних показників енергоефективності та продуктивності роботи IT інфраструктури в центрах обробки даних застосовуються технології віртуалізації апаратного забезпечення, програмних засобів, мереж, сховищ даних, робочих місць та інші.

Сервісна модель IaaS дозволяє більш ефективно використовувати апаратне забезпечення фізичного сервера (ФС) за рахунок віртуалізації його локальних ресурсів. ФС надає такі ресурси, як процесорний час, пам'ять, локальну підсистему зберігання даних та підсистему доступу до мережі. При цьому, клієнтові надається частина ресурсів ФС у вигляді віртуальної машини (ВМ) або контейнеру. Для реалізації сучасних інформаційних послуг клієнт розгортає одну або декілька ВМ необхідної конфігурації, яка визначена провайдером хмарних послуг. Кожній ВМ гіпервізор надає частку ресурсів ФС. З точки зору керування ресурсами хмарного ЦОД, ресурси всіх ФС об’єднуються в *пул* і надаються віртуальним машинам для використання. Таким чином, виникає комплекс задач, пов’язаних з керуванням ресурсами хмарного ЦОД. Один із варіантів перерозподілу ресурсів пулу між віртуальними машинами полягає в реалізації процесу *консолідації віртуальних машин* (англ. virtual machine consolidation). Консолідація віртуальних машин – це розміщення віртуальних машин на фізичних серверах на базі технологій віртуалізації з метою досягнення певних показників ефективності використання ресурсів ЦОД. Фактично, керуючі впливи виробляються для віртуальних машин, фізичних серверів, мережевих пристроїв, сховищ, застосувань та інших підсистем.

Для роботи кожного екземпляру застосування в хмарному ЦОД зазвичай створюється окрема ВМ. На відміну від *монолітних* застосувань, які потребують нарощування ресурсів *вертикально* (англ. scale-up) при зростанні навантаження, застосування для хмарних ЦОД (англ. cloud-native applications) потребують нарощування ресурсів горизонтально (англ. scale-out). Горизонтальне нарощування ресурсів полягає у створенні додаткових екземплярів застосування у вигляді ВМ для обслуговування зростаючого навантаження (запитів клієнтів). Останнім часом, для забезпечення роботи хмарних застосувань широко використовуються так звані *контейнери* (англ. containers). Контейнер – це середовище для виконання екземпляра застосування. Контейнери ізо-

льовані один від одного і виконуються в спільній віртуальній машині.

В статті розглядається робота застосувань на базі ВМ. Виходячи з певних бізнес-потреб клієнт може динамічно змінювати конфігурацію ВМ або налаштовує механізми балансування навантаження, відмовостійкості та резервного копіювання. В процесі роботи хмарного сервісу клієнти створюють множину ВМ. В результаті зміни навантаження, зміни кількості запитів клієнтів та кількості пакетних задач, кількість віртуальних машин змінюється. Відповідно, змінюється кількість ВМ, що виконуються на окремому ФС, тому деякі ФС виявляються незавантаженими до максимально можливого порогу і витрачають зайву енергію.

З метою більш ефективного використання ресурсів хмарного ЦОД засоби віртуалізації надають можливість "живої" міграції (англ. live migration) ВМ з одного ФС на інший. При цьому, вплив на роботу ВМ є мінімальним, і для клієнта міграція ВМ не впливає на виконання задач. Але навантаження на фізичні сервери, які обмінюються цією ВМ, зростає.

Таким чином, однією з головних задач при керуванні ресурсами хмарного ЦОД є розміщення і перерозміщення віртуальних машин таким чином, щоб задіяти меншу кількість фізичних серверів та зменшити кількість міграцій віртуальних машин. Процес перерозподілу віртуальних машин серед фізичних серверів, консолідацію ВМ, можливо виконувати як неперервно, так і дискретно. Вирішення проблеми консолідації віртуальних машин представлено в багатьох публікаціях [1]. Але дослідження виконувалися для наявних на той час технологій віртуалізації та хмарних технологій. В сучасних ЦОД впроваджуються все нові і нові апаратні та системні засоби, які співіснують з технологіями попередніх поколінь. Таким чином, актуальною є розробка нових алгоритмів і методів для керування обчислювальними ресурсами ЦОД в цілому, та розміщенням віртуальних машин, зокрема.

Для вирішення задачі керування ресурсами хмарного ЦОД пропонується двостадійний метод на базі використання алгоритму променевого пошуку. Запропонований метод призначений для вирішення однієї з підзадач керування ресурсами шляхом консолідації віртуальних машин. В статті проаналізовано роботу евристики першої та другої стадій запропонованого методу, розроблений алгоритм променевого пошуку, уточнені функція оцінювання та умови закінчення роботи алгоритму. Для аналізу роботи методу використані дані про надходження завдань у кластер Google.

1. Аналіз публікацій

Останнім часом запропоновано багато методів та алгоритмів для вирішення проблеми керування ресурсами ЦОД [1–3]. Зокрема, задача консолідації віртуальних машин розглядається як оптимізаційна задача з різними цільовими функціями. Також, задача консолідації віртуальних машин розглядається як багатокритеріальна оптимізаційна задача [4, 5]. Складність цієї задачі полягає у наявності великої кількості станів середовища та обмежень. Крім того, складність виявляється при формулюванні цільової функції, до якої входять декілька показників, які треба оптимізувати. В результаті аналізу існуючих рішень з’ясувалося, що досягти ефективності деяких показників одночасно виявляється неможливим. Наприклад, неможливо досягнути одночасно високої швидкості розгортання ВМ та енергозбереження, або одночасно високої продуктивності та енергозбереження.

Для промислових ЦОД з хмарними інфраструктурами використовуються прості алгоритми керування, такі як first-fit, best-fit та їх модифікації (Eucalyptus [6], Microsoft [7], Google [8]). Це обумовлено вимогами робастності застосувань клієнтів та участю адміністраторів в автоматизованому процесі керування ресурсами ЦОД при постійному моніторингу якісних показників.

В дослідженнях використовуються такі цільові функції, як мінімізація споживання електроенергії, мінімізація порушень угод про якість сервісу (SLA), мінімізація мережевого трафіку,

максимізація продуктивності та використання ресурсів. Однією з основних умов для сучасних кластерів ФС є можливість роботи методів керування ресурсами в режимі онлайн [3]. Ці методи використовують потоки керування, що працюють *паралельно*. Разом з цим допускається застосування методів керування ресурсами, які спрацьовують при появі певних умов роботи кластеру, або через певний проміжок часу. Такі методи використовують потоки керування, що працюють *послідовно*. Для такого випадку нові завдання зазвичай розміщуються на ФС, які не задіяні в процесі консолідації ВМ.

Крім традиційних евристик та детермінованих алгоритмів при керуванні ресурсами ЦОД використовуються і алгоритми локального пошуку, такі як еволюційні алгоритми [9], алгоритми оптимізації мурашиних колоній [10], табу пошук [11], пошук з емуляцією відпалу [12]. Більш ефективним на нашу думку є використання локального пошуку на другій стадії оптимізації, після підготовки відповідного набору станів детермінованими алгоритмами з метою покращити рішення, знайдене на першій стадії.

Таким чином, в цій статті проаналізовано застосування алгоритму променевого пошуку в складі двостадійного методу для керування ресурсами хмарного ЦОД з метою зменшити кількість міграцій ВМ та збільшити кількість ФС, переключених у режим сну.

2. Модель системи

На теперішній час більшість послуг клієнти отримують на базі хмарних центрів обробки даних. Хмарні ЦОД – це складні системи, що складаються з серверних підсистем, підсистем зберігання даних, мережевих підсистем та підсистем інженерного забезпечення.

В статті розглянуто задачу керування ресурсами окремої підсистеми ЦОД у вигляді кластера з використанням віртуалізації. Для побудови кластеру в сучасних ЦОД використовують два підходи компоновки: з використанням гетерогенної або гомогенної конфігурації. Обидва підходи мають свої недоліки та переваги, однак уникнути розміщення гетерогенних конфігурацій в масштабі ЦОД в більшості випадків не вдається. Конфігурації кожного кластеру можуть відрізнятися з причин еволюції елементної бази серверів, сховищ та мережевих пристроїв, а також появи нових вимог користувачів до IT-інфраструктури. Однак має місце і найгірший випадок, коли конфігурації фізичних серверів у кластері відрізняються.

Дослідження роботи запропонованого двостадійного методу, що базується на алгоритмі променевого пошуку, виконані для кластеру, який складається з фізичних серверів різної конфігурації. Кожен ФС надає для локальних ВМ такі ресурси як: процесорний час (CPU), обсяг пам’яті (RAM), доступ до підсистеми зберігання даних (IOPS), доступ до мережевої підсистеми (NET) та інші. В залежності від наявної ємності ресурсів ФС, вимог до ресурсів з боку ВМ, часу виконання завдань всередині її та інтенсивності надходження завдань кількість ВМ, що виконуються на ФС, постійно змінюється. Зміна кількості локальних ВМ відбувається в таких станах: розгортання нової ВМ, завершення роботи ВМ, міграція ВМ.

В сучасних умовах, при використанні промислових гіпервізорів та швидкісних локальних мереж, середній час міграції віртуальної машини в середині ЦОД складає приблизно півхвилини, в залежності від обсягу пам’яті, що використовує ВМ.

Крім того, на процеси керування ресурсами також впливає час включення фізичного сервера, який складає 3–4 хвилини в залежності від конфігурації. Таким чином, краще перемикати фізичний сервер в режим сну. Переключення фізичного сервера з режиму сну в режим роботи відбувається значно швидше, чим холодний старт сервера.

Аналіз роботи запропонованого методу виконано з урахуванням двох ресурсів, що надаються для ВМ: CPU та RAM. Однак метод може бути доповнений іншими ресурсами, які потребує ВМ.

Вибір саме двох ресурсів обумовлено використанням вхідних даних з набору Google cluster-usage traces (GCT) [13]. Для аналізу роботи алгоритму променевого пошуку використано дві таблиці з набору GCT, а саме "Machine events" та "Task events table". Випадковим чином з першої таблиці обрано 6000 ФС, з другої таблиці обрано 70000 завдань. З таблиці "Machine events" для кожного ФС використано такі атрибути: machine ID, capacity: CPU, capacity: memory. З таблиці "Task events table" для кожного ФС використано такі атрибути: "task index within the job", "machine ID", "resource request for CPU cores", "resource request for RAM". Дані в таблицях нормовані відносно ФС з найбільшим значенням ємності відповідного ресурсу серед його кластеру.

Процеси циклу керування можуть повторюватись через певні проміжки часу при наявності двох вимог: міграції ВМ, визначені на попередньому кроці завершені та є передумови для переключення ФС в режим сну. Таким чином, запропонований двостадійний метод керування ресурсами використовує потоки керування, що працюють *послідовно.*

3. Постановка задачі

Кластер керування складається з множини P з M фізичних серверів та множини V з N віртуальних машин, $N, M \in \mathbb{N}$. В процесі розробки та аналізу роботи алгоритму променевого пошуку та в процесі циклу керування міграціями кількість ФС та ВМ не змінюється.

Для кожного завдання з таблиці "Task events table" розгортається окрема ВМ. В загальному випадку, між запусками алгоритму променевого пошуку кількість ФС та ВМ може змінюватись.

Задана ємність j-ї ВМ для ресурсу k, що позначена як $c_j^k \in (0,1]$, $k \in \{CPU, RAM\}$, визначається вимогами завдання і нормовано відносно ФС з найбільшою ємністю ресурсу k. Ємність фізичного сервера i для ресурсу k, що позначена $C_i^k \in (0,1]$, визначається типом ФС і нормовано відносно ФС з найбільшою ємністю ресурсу k.

Множина P складається з множини A фізичних серверів, які визначені для вимикання, та множини B фізичних серверів, які надають ресурси для ВМ, що будуть мігрувати з ФС, які належать до множини A, $A \cup B = P$, $A \cap B = \varnothing$.

Міграцію віртуальної машини j на фізичний сервер і позначимо як $U_{ij} \in \{0,1\}$. Міграція відбувається якщо $U_{ij} = 1$. Кожна ВМ з множини V має свій ID, який в процесі роботи алгоритму пов'язаний з номером j. Кожний ФС теж має свій ID, який в процесі роботи алгоритму пов'язаний з номером i.

Основною метою розробки і застосування двостадійного методу керування ресурсами хмарного ЦОД є зменшення кількості міграцій ВМ під час циклу керування та збільшення кількості ФС, що переключені в режим сну.

4. Двостадійний метод керування ресурсами на основі променевого пошуку

В основу метода покладено алгоритм променевого пошуку та евристики для оцінки стану кластера ЦОД. Робота методу складається з двох стадій. На першій стадії відбувається підготовка даних, на другій стадії визначається план міграцій ВМ.

Вхідними даними алгоритму променевого пошуку є: n – ширина променя, A – список ФС, які є претендентами для перемикання в режим сну, B – список ФС для обміну ВМ, в якому є вільні ресурси і є можливість розмістити додаткові завдання у вигляді ВМ.

Ідея алгоритму: перегляд i-го ФС зі списку A та пошук таких або такого ФС зі списку B, куди можливо мігрувати j-ту ВМ з i-го ФС. Якщо вдалося звільнити всі або частку ФС зі списку A, видаємо результат у вигляді матриці U_{ij}, яка є планом міграцій.

Опис роботи алгоритму променевого пошуку.

Перша стадія: підготовка вхідних даних для другої стадії. Формування списку A фізичних серверів, які треба звільни-

ти від ВМ, та списку B фізичних серверів для визначення плану міграцій.

Друга стадія виконується для кожного ФС із списку A.

1. На кожному кроці обираємо одну ВМ, назначену на i-й ФС і розглядаємо наступні варіанти обміну (міграцій):

a) мігрувати ВМ на інший ФС, в якого залишилось достатньо вільних ресурсів CPU та RAM;

b) перевірити можливість обміну з іншим ФС віртуальною машиною, яка вимагає менше ресурсів (так ми розглядаємо лише стани з кращою оцінкою та уникаємо можливих зациклювань) і, в результаті, ФС не буде перенавантаженим після обміну.

2. З усіх можливих обмінів обирається n обмінів з найвищою оцінкою.

3. Завершуємо пошук, якщо i-й ФС вдалося звільнити від віртуальних машин або, якщо не можна побудувати нові стани (тобто не залишилось жодного варіанта для реалізації допустимого обміну a) або b)).

Порівняння станів виконується за допомогою критерію:

$$J = \sum_{i=1}^{m} u_i^2 + \sum_{i=1}^{n} f_i^2 , \qquad (1)$$

де u_i – кількість використовуваних ресурсів на i-му ФС, f_i – кількість вільних ресурсів на i-му ФС, m – кількість ФС у списку B, n – кількість ФС у списку A.

Таким чином, алгоритм поступово зменшує використовувані ресурси на ФС, які належать до списку A, та завантажує ФС зі списку B.

Умови завершення алгоритму

Для завершення роботи алгоритму пропонуються умови.

1. Вичерпання списку A.

2. Неможливість мігрувати всі ВМ з певної визначеної кількості ФС Th_A поспіль.

Якщо другу умову не застосовувати, цикли пошуку виконуються занадто довго, порівняно з часом на створення нової ВМ та часом на міграцію для її з середніми вимогами до ресурсів пам'яті. Для визначення Th_A пропонується застосувати евристику, яка полягає у розгляді певного відсотку фізичних серверів із списку A, але не менше, ніж α фізичних серверів. У реалізації досліджуваного алгоритму прийнято $Th_A = 0.05$, $\alpha = 10$. З меншими значеннями α алгоритм пропускає значну кількість ФС, що могли бути переключені в режим сну, але в результаті завершення алгоритму були не звільнені від локальних ВМ. Даний критерій завершення вводиться для зменшення часу виконання алгоритму.

Опис першої стадії роботи методу.

Отримання списку фізичних серверів, з яких треба мігрувати всі ВМ з метою подальшого переключення ФС в режим сну пропонується здійснювати за допомогою двох методик: *нижньої границі* та *порогу вільних ресурсів*. Розглянемо кожну з методик більш детально.

Методика нижньої границі полягає у визначенні такої кількості фізичних серверів, які вимкнути в результаті міграції усіх ВМ, що на них працює, не уявляється можливим. Таке уявлення виникає з гіпотез, що використовуються в роботі алгоритму визначення нижньої границі.

Пошук нижньої границі виконується таким чином:

1) знаходимо середній обсяг наявних ресурсів за всіма ФС, на яких працюють ВМ,

$$T_{av}^k = \frac{1}{Q} \sum_{i=1}^{Q} C_i^k , \; 0 < Q \le M .$$

Значення для кожного ресурсу k розраховуються окремо;

2) по кожному ресурсу окремо рахуємо суму необхідних ресурсів для виконання всіх наявних ВМ,

$$D^k = \sum_{j=1}^{R} c_j^k , \; 0 < R \le N ;$$

3) окремо по кожному ресурсу рахуємо відношення необхідних ресурсів до середнього обсягу ресурсів та округляємо значення до більшого цілого, $E^k = \lceil D^k / T_{av}^k \rceil$;

4) нижньою границею буде найбільше число E^k з отриманих на кроці 3;

5) сортуємо фізичні сервери одним з наступних способів:

a) за обсягом ресурсів ФС, потім за відношенням використаних ресурсів до кількості працюючих ВМ;

b) за обсягом ресурсів ФС, потім за кількістю назначених завдань, потім відношенням використаних ресурсів до кількості локальних ВМ.

В результаті досліджень роботи алгоритму з'ясувалося, що варіант b) виявився значно ефективнішим. При його використанні на одному й тому самому наборі даних вдалося вимкнути в середньому 82 ФС, тоді як при використанні варіанту a) вдалося вимкнути в середньому лише 33 ФС.

В результаті, знаходимо різницю між кількістю ФС, що використовуються, та отриманою нижньою границею. Знайдене число – це кількість ФС списку A на вимкнення, які є першими у відсортованому списку. Решта ФС потрапляє у список B. Таким чином, методика нижньої границі дозволяє отримати оцінку наявної вільної ємності фізичних ресурсів кластеру.

Ідея методики порогу вільних ресурсів полягає у такому формуванні списку A, при якому до цього списку потрапляють ФС, загальна кількість невикористаних ресурсів яких перевищує обсяг ресурсів одного з ФС кластеру.

Позначимо поріг вільних ресурсів як β. Побудова списку A з використанням порогу вільних ресурсів виконується наступним чином:

1) встановлюємо значення β;

2) з множини P відбираємо ФС, у яких кожного вільного ресурсу більше за значення β;

3) по кожному ресурсу окремо рахуємо суму вільних ресурсів,

$$F^k = \sum_{i=1}^{Q} C_i^{k,\,free},\ 0 < Q \le M;$$

4) сортуємо обрані ФС аналогічно до варіанту b) з методики нижньої границі;

5) проходимо за відсортованим списком. Поки сума ресурсів більша за обсяг поточного ФС віднімаємо від суми цей обсяг, додаємо поточний ФС в список A та переходимо до наступного ФС, інакше завершуємо проходження списку. Таким чином, отримуємо список A з ФС, які треба переключити в режим сну, та список B з ФС для обміну віртуальними машинами.

5. Оцінка результатів моделювання

Дослідження роботи двостадійного методу виконане на фрагментах набору даних GCT [13]. Дані для тестування і дослідження методу обрані з GCT і являють собою частину за певний діапазон часу. Це необхідно, щоб врахувати всі дані за один і той самий період часу в журналах GCT. З набору випадковим чином відібрано 70000 завдань з певними вимогами до ресурсів k. Для кожного завдання буде розгорнута окрема ВМ в процесі моделювання. Обираємо 6000 фізичних серверів, з яких на 5832 серверах розміщено ВМ, та на 168 серверах завдань немає, але вони доступні для розміщення ВМ. Моделювання виконане на комп'ютері з процесором i5-3570 та обсягом системної пам'яті 4024 Мб.

Якісними показниками роботи алгоритму є кількість вимкнених ФС PM_{sleep} та кількість міграцій ВМ VM_{mig}, за допомогою яких ФС вивільнюються від ВМ. Крім того, враховано час роботи методу, t.

В таблиці 1 представлені результати роботи запропонованого методу з консолідації ВМ для різних значень β. В таблиці 2 представлені результати роботи методу при консолідації ВМ з використанням методики нижньої границі.

Таблиця 1

Результати моделювання консолідації ВМ з використанням методики порогу вільних ресурсів

β	B	A	PM_{sleep}	VM_{mig}	t, s
0.005	199	3	0	0	0.53
0.004	299	3	0	0	0.69
0.003	1098	9	0	0	4.05
0.002	1393	10	0	0	6.55
0.001	1868	28	13	219	38.08
0.0009	1943	31	15	263	42.29
0.0008	2002	39	22	349	67.93
0.0007	2081	43	24	399	81.02
0.0006	2204	48	28	432	94.14
0.0005	2302	55	32	504	105.48
0.0004	2407	63	36	554	141.43
0.0003	2565	73	43	683	143.16
0.0002	2764	82	48	782	166.54
0.0001	3706	95	59	992	238.06
0.00005	4323	106	68	1199	330.18
0.00001	5095	116	75	1336	463.36
0	5328	125	84	1459	518.34

Таблиця 2

Результати моделювання консолідації ВМ з використанням методики нижньої границі

B	A	PM_{sleep}	VM_{mig}	t, s
5707	125	82	1460	508.09

Вплив значення порогу β на якісні показники роботи алгоритму є суттєвим (рис. 1 та 2), залежність виявилася майже лінійною. Чим менший поріг β тим більше ФС потрапляють на вхід алгоритму.

При $\beta = 0$ деякі ФС (а саме ті, у яких хоча б один ресурс повністю зайнятий) не потрапляють на вхід алгоритму, тому що при перевірці наявності ресурсів використовується строга нерівність. При $\beta < 0.0002$ алгоритм з методикою порогу вільних ресурсів починає працювати більш ефективно і на граничних значеннях працює краще чим з методикою нижньої границі.

Після формування списку A ефективність пошуку кінцевого стану кластера можливо оцінити за допомогою відношення кількості ФС, що заплановані для переключення у режим сну, до кількості ФС, що фактично визначені для перемикання в режим сну після роботи другої стадії методу. Ефективність перемикання ФС в режим сну зростає з ростом значення порогу β і знаходиться в діапазоні від 45 % до 65 %.

Для перевірки роботи алгоритму з різними методиками та їх порівняння створено декілька наборів даних з однаковою кількістю ФС та завдань за інші періоди часу в наборі даних GCT. В результаті моделювання отримані показники ефективності, які розрізняються не більше ніж на 7 %.

Перевагою використання методики з порогом вільних ресурсів є можливість адаптуватися до стану кластера шляхом

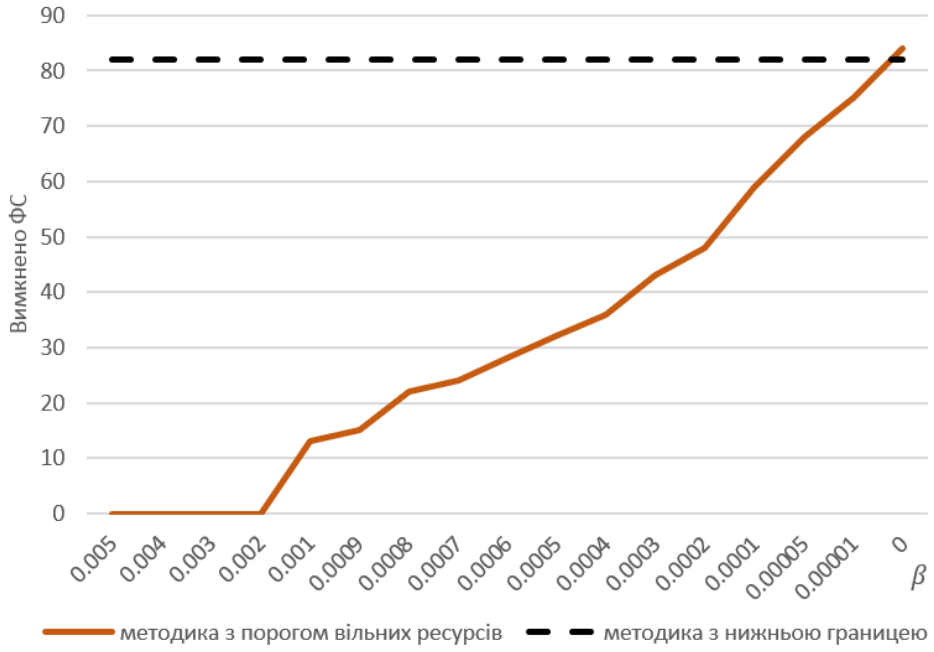

Рис. 1. Вплив порогу на кількість ФС, переключених в режим сну

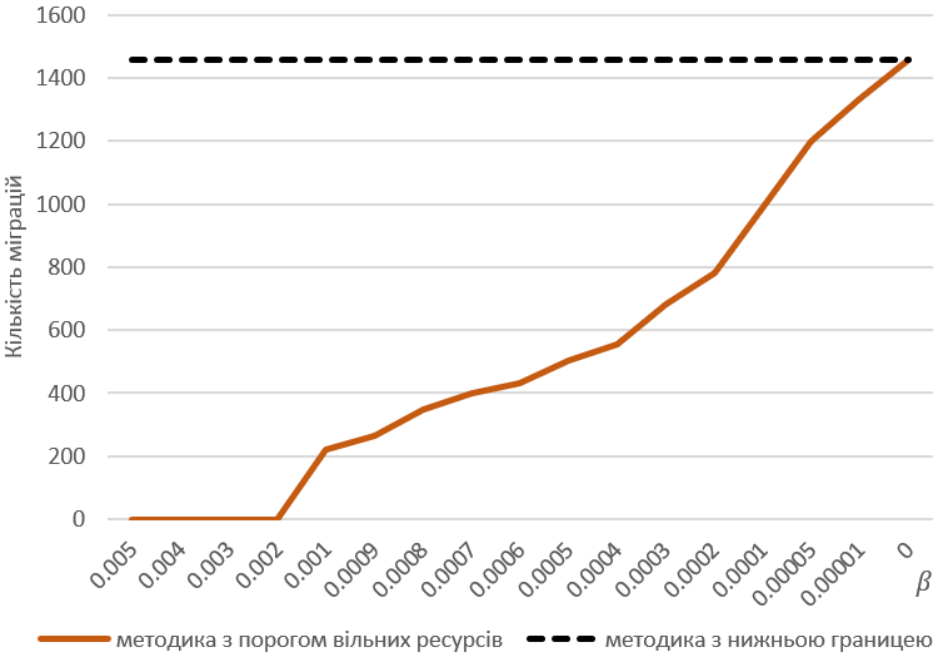

Рис. 2. Вплив порогу на кількість міграцій

зміни порогу перед циклом керування, враховуючи інші ресурси, час міграції ВМ та інтенсивність надходження заявок на створення нових ВМ.

В процесі дослідження роботи методу, з метою встановлення впливу ширини променя на якісні показники роботи алгоритму, ширина променю для алгоритму променевого пошуку змінювалась в діапазоні від 1 до 15 (табл. 3 та 4).

При використанні методики нижньої границі збільшення ширини променя алгоритму призводить до суттєвого збільшення часу пошуку рішення. Так, при ширині променя n=10 час виконання алгоритму значно зростає і стає неприпустимим для використання в кластері з високою динамікою процесів створення ВМ (табл. 3).

Таблиця 3

Результати моделювання консолідації ВМ з використанням методики нижньої границі

n	PM_{sleep}	VM_{mig}	*t*, с
1	23	423	66.72
2	23	435	101.33
3	23	433	137.75
4	23	450	173.57
5	82	1460	508.09
6	82	1457	812.05
7	83	1471	832.64
8	81	1418	894.11
9	83	1479	1169.78
10	80	1417	-
11	82	1428	-
12	82	1449	-
13	85	1520	-
14	83	1501	-
15	84	1487	-

Таблиця 4

Результати моделювання консолідації ВМ з використанням методики з порогом вільних ресурсів

n	PM_{sleep}	VM_{mig}	*t*, с
1	27	422	53.51
2	35	548	116.5
3	57	986	186.26
4	57	959	204.49
5	57	940	247.78
6	57	918	331.03
7	55	922	383.36
8	59	969	466.55
9	57	943	492.38
10	56	957	471.13
11	56	944	549.1
12	58	972	584.52
13	60	1063	718.3
14	58	1021	791.87
15	61	1042	844.12

Крім того, зміна ширини променя при використанні кожної методики може не призводити до суттєвого покращення якісних показників, при суттєвому збільшенні часу пошуку. Так, для методики з нижньою границею зміна ширини променя з 5 до 15 не призводить до покращення якісних показників, а для методики з порогом вільних ресурсів покращення якісних показників не відбувається при ширині променя від 3 до 15 (рис. 3 та 4).

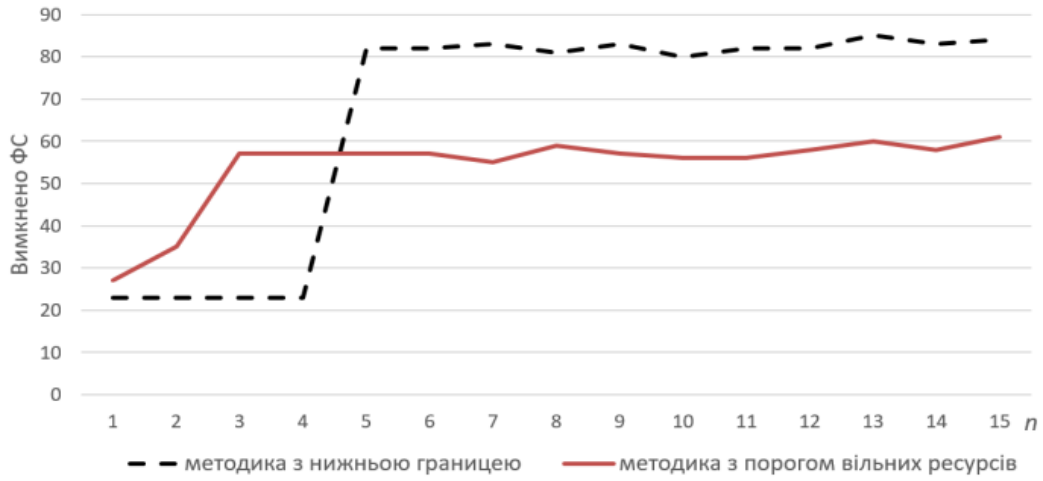

Рис. 3. Вплив ширини променя на кількість ФС, переключених в режим сну

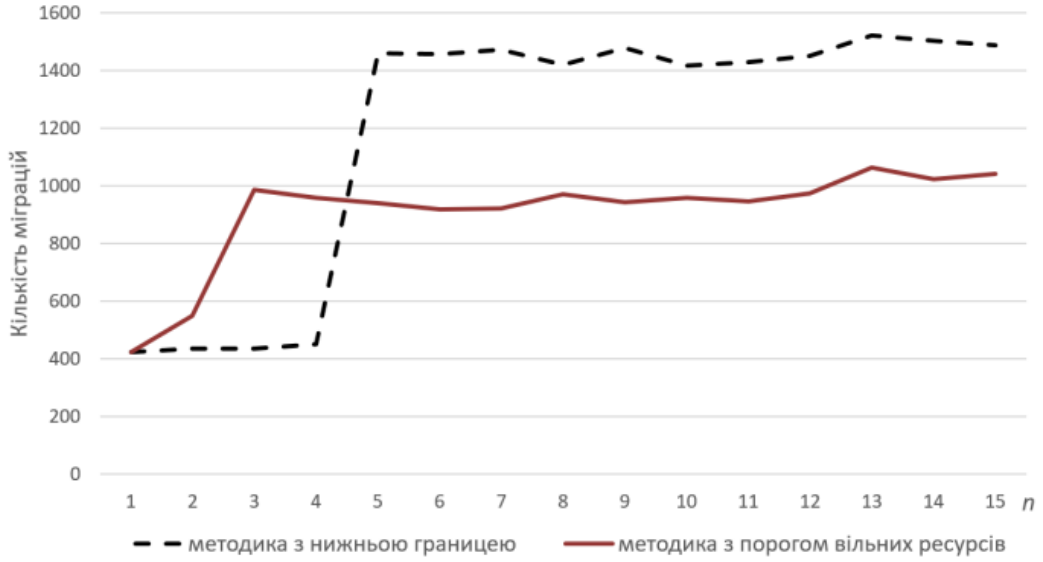

Рис. 4. Вплив ширини променя на кількість міграцій

Таким чином, треба звертати увагу на кількість міграцій, забезпечуючи їх допустиму кількість. З урахуванням висновків, рекомендується обирати ширину променя від 5 до 8, в залежності від умов роботи методу.

Висновки

Для керування ресурсами хмарного ЦОД на рівні кластера в статті запропоновано і досліджено двостадійний метод на базі алгоритму променевого пошуку. В статті проаналізовано роботу евристики першої та другої стадій запропонованого методу, розроблений алгоритм променевого пошуку для вирішення задачі керування ресурсами. Для аналізу роботи методу використані дані про надходження завдань в кластер Google. Запропонований метод дозволяє переключити в режим сну в середньому 56 відсотків фізичних серверів, що потенційно визначені для переключення в режим сну за допомогою верхньої оцінки необхідної ємності ресурсів.

Перерозподіл віртуальних машин виконується з урахуванням обмеження допустимої кількості міграцій. Завдяки урахуванню обмеження кількості міграцій на один фізичний сервер запропонований метод може бути застосований в реальних умовах ЦОД.

В результаті дослідження пропонується використовувати методику з порогом вільних ресурсів, що показала більш якісні

результати консолідації ВМ. Перевагою використання методики з порогом вільних ресурсів є можливість адаптуватися до стану кластера шляхом зміни порогу перед циклом керування, враховуючи інші ресурси, час міграції ВМ та інтенсивність надходження заявок на створення нових ВМ.

Також встановлено, що рекомендована ширина променя в алгоритмі променевого пошуку складає від 5 до 8, в залежності від умов роботи методу, обмежень на кількість міграцій ВМ та обмежень на час виконання алгоритму.

Для перевірки роботи алгоритму з різними методиками та їх порівняння створено декілька наборів даних з однаковою кількістю ФС та завдань за інші періоди часу в наборі даних GCT. Отримані показники ефективності розрізняються не більше ніж на 7 %.

1. Pires F.L., & Barán, B. (2015, May). A virtual machine placement taxonomy. In Proc. of the 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid). P. 159–168.
2. Ahmad R.W., Gani A., Hamid S.H.A., Shiraz M., Yousafzai A., & Xia F. A survey on virtual machine migration and server consolidation frameworks for cloud data centers. Journal of Network and Computer Applications. 2015. 52. P. 11–25.
3. Telenyk S., Zharikov E., & Rolik O. An approach to virtual machine placement in cloud data centers. In Radio Electronics & Info Communications (UkrMiCo), 2016 International Conference. IEEE. September, 2016. P. 1–6.
4. Pires F.L., & Barán B. Multi-objective virtual machine placement with service level agreement: A memetic algorithm approach. In Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing. IEEE Computer Society. 2013, December. P. 203–210.
5. Saber T., Ventresque A., Brandic I., Thorburn J., & Murphy L. Towards a multi-objective vm reassignment for large decentralised data centres. 8th International Conference on Utility and Cloud Computing (UCC), 2015 IEEE/ACM. IEEE. 2015, December. P. 65–74.
6. Eucalyptus community [Online] – Available from: http://open.eucalyptus.com/
7. Lee S., Panigrahy R., Prabhakaran V., Ramasubramanian V., Talwar K., Uyeda L., & Wieder U. Validating heuristics for virtual machines consolidation. Microsoft Research, MSR-TR-2011-9. 2011. P. 1–14.
8. Sharma B., Chudnovsky V., Hellerstein J.L., Rifaat R., & Das C.R. Modeling and synthesizing task placement constraints in Google compute clusters. In Proceedings of the 2nd ACM Symposium on Cloud Computing. ACM. (2011, October). p. 3.
9. Mark, C. C. T., Niyato, D., & Chen-Khong, T. Evolutionary optimal virtual machine placement and demand forecaster for cloud computing. IEEE International Conference on Advanced Information Networking and Applications (AINA), IEEE. 2011, March. P. 348–355.
10. Gao Y., Guan H., Qi Z., Hou Y., & Liu L. A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. Journal of Computer and System Sciences. 2013. 79(8). P. 1230–1242.
11. Ferreto, T., De Rose, C. A., & Heiss, H. U. Maximum migration time guarantees in dynamic server consolidation for virtualized data centers. In European Conference on Parallel Processing Springer, Berlin, Heidelberg. 2011, August. P. 443–454.
12. Wu Y., Tang M., & Fraser W. A simulated annealing algorithm for energy efficient virtual machine placement. In IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE. 2012, October. P. 1245–1250.
13. Reiss C., Wilkes J., & Hellerstein J.L. Google cluster-usage traces: format+ schema. Google Inc., White Paper. 2011. P. 1–14.

References

1. Pires, F. L., & Barán, B. (2015, May). A virtual machine placement taxonomy. In Proc. of the 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid). pp. 159-168.
2. Ahmad, R. W., Gani, A., Hamid, S. H. A., Shiraz, M., Yousafzai, A., & Xia, F. (2015). A survey on virtual machine migration and

server consolidation frameworks for cloud data centers. Journal of Network and Computer Applications, 52, pp. 11-25.
3. Telenyk, S., Zharikov, E., & Rolik, O. (2016, September). An approach to virtual machine placement in cloud data centers. In Radio Electronics & Info Communications (UkrMiCo), 2016 International Conference (pp. 1-6). IEEE.
4. Pires, F. L., & Barán, B. (2013, December). Multi-objective virtual machine placement with service level agreement: A memetic algorithm approach. In Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing (pp. 203-210). IEEE Computer Society.
5. Saber, T., Ventresque, A., Brandic, I., Thorburn, J., & Murphy, L. (2015, December). Towards a multi-objective vm reassignment for large decentralised data centres. 8th International Conference on Utility and Cloud Computing (UCC), 2015 IEEE/ACM (pp. 65-74). IEEE.
6. Eucalyptus community [Online] – Available from: http://open.eucalyptus.com/
7. Lee, S., Panigrahy, R., Prabhakaran, V., Ramasubramanian, V., Talwar, K., Uyeda, L., & Wieder, U. (2011). Validating heuristics for virtual machines consolidation. Microsoft Research, MSR-TR-2011-9, pp. 1-14.
8. Sharma, B., Chudnovsky, V., Hellerstein, J. L., Rifaat, R., & Das, C. R. (2011, October). Modeling and synthesizing task placement constraints in Google compute clusters. In Proceedings of the 2nd ACM Symposium on Cloud Computing (p. 3). ACM.
9. Mark, C. C. T., Niyato, D., & Chen-Khong, T. (2011, March). Evolutionary optimal virtual machine placement and demand forecaster for cloud computing. IEEE International Conference on Advanced Information Networking and Applications (AINA), (pp. 348-355). IEEE.
10. Gao, Y., Guan, H., Qi, Z., Hou, Y., & Liu, L. (2013). A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. Journal of Computer and System Sciences, 79(8), pp. 1230-1242.
11. Ferreto, T., De Rose, C. A., & Heiss, H. U. (2011, August). Maximum migration time guarantees in dynamic server consolidation for virtualized data centers. In European Conference on Parallel Processing (pp. 443-454). Springer, Berlin, Heidelberg.
12. Wu, Y., Tang, M., & Fraser, W. (2012, October). A simulated annealing algorithm for energy efficient virtual machine placement. In IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2012 (pp. 1245-1250). IEEE.
13. Reiss, C., Wilkes, J., & Hellerstein, J. L. (2011). Google cluster-usage traces: format+ schema. Google Inc., White Paper, 1-14.

Одержано 01.10.2017

Про авторів:

Жаріков Едуард В'ячеславович,
кандидат технічних наук,
доцент, докторант Національного технічного університету України "КПІ імені Ігоря Сікорського".
Кількість наукових публікацій в українських виданнях – 86.
Кількість наукових публікацій в зарубіжних виданнях – 17.
Індекс Хірша – 1.
http://orcid.org/0000-0003-1811-9336.

Місце роботи автора:

Національний технічний університет України "КПІ імені Ігоря Сікорського".
Тел.: 38044 204 86 10.
E-mail: zharikov.eduard@acts.kpi.ua