УДК 004.7

Телелейко Інна Сергіївна
магістрант
Національного технічного університету України
«Київський політехнічний інститут імені Ігоря Сікорського»
Телелейко Инна Сергеевна
магистрант
Национального технического университета Украины
«Киевский политехнический институт имени Игоря Сикорского»
Teleleiko Inna
Graduating Student of the
National Technical University of Ukraine
«Igor Sikorsky Kyiv Polytechnic Institute»

Орлова Марія Миколаївна
кандидат технічних наук, доцент
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Орлова Мария Николаевна
кандидат технических наук, доцент
Национальный технический университет Украины
«Киевский политехнический институт имени Игоря Сикорского»
Orlova Mariia
Candidate of Technical Sciences, Associate Professor
National Technical University of Ukraine
«Igor Sikorsky Kyiv Polytechnic Institute»

СПОСІБ ДИНАМІЧНОГО БАЛАНСУВАННЯ НАВАНТАЖЕННЯ В ХМАРНОМУ СЕРЕДОВИЩІ

СПОСОБ ДИНАМИЧЕСКОЙ БАЛАНСИРОВКИ НАГРУЗКИ В ОБЛАЧНОЙ СРЕДЕ

DYNAMIC LOAD BALANCING METHOD IN CLOUD COMPUTING

Анотація. *У даній статті викладено основні аспекти сучасного стану даної проблеми, описані існуючі підходи до балансування навантаження в хмарному середовищі. Запропоновано метод динамічного балансування для оптимізації цих підходів.*

Ключові слова: *хмарні технології, веб-сервіс, балансування навантаження, алгоритми динамічного балансування.*

Аннотация. *В статье изложены основные аспекты современного состояния данной проблемы, описаны существующие подходы балансировки нагрузки в облачной среде. Предложен метод динамической балансировки для оптимизации этих подходов.*

Ключевые слова: *облачные технологии, веб-сервис, балансировка нагрузки, алгоритмы динамической балансировки.*

Summary. *The article outlined the key aspects of the current state of the problem, describes a study of existing approaches to the technology of load balancing in cloud computing. Dynamic load method was proposed as an optimization for mentioned purpose.*

Key words: *cloud technology, web service, load balancing, load balancing algorithm.*

Вступ. При побудові розподіленої системи обчислення в зв'язку з розвитком засобів передачі даних все більше використовують підходи розподіленого програмування. Виконання великої задачі розподіляється на менші підзадачі, які можуть виконуватись на різних комп'ютерах зі спільною мережею, а після всіх обчислень результати їх роботи використовуються при обчисленні початкової задачі [2]. При вирішенні деяких задач використання розподіленої системи застосовується для підвищення таких показників ефективності, як зниження вартості, збільшення надійності, досягнення певного рівня продуктивності системи, простоти масштабування тощо.

Важливим аспектом стало застосування механізмів управління ресурсами в локально та глобально розподілених середовищах. Під «ресурсами» маємо на увазі все, що так чи інакше бере участь в обробці даних: обчислювальні кластери, сховища даних, файлові системи, програмне забезпечення, мережеве устаткування, яке забезпечує з'єднання ресурсів в єдину систему.

При виконанні розподіленої програми, комп'ютери, які приймають участь в обробці даних, з'єднуються з мережею Інтернет. Обмін повідомленнями відбувається за протоколом HTTP, оскільки обмін даними мережею за іншими портами та в іншому форматі без додаткових налаштувань фільтрується більшістю брандмауерами та проксі.

До появи веб-сервісів у світі вже існували технології, що дозволяли додаткам взаємодіяти на відстані, де одна програма могла викликати будь-який інший ресурс, який при цьому міг бути запущений на комп'ютері, розташованому в іншому місті або навіть країні. Такий віддалений виклик процедур називається RPC (Remote Procedure Calling). Прикладом таких технологій є CORBA (Common Object Request Broker Architecture), RMI (Remote Method Invoking). Ідея веб-сервісу полягала в створенні такого RPC, який би взаємодіяв з HTTP пакетом.

Перш ніж викликати віддалену процедуру, необхідно описати виклик в XML файлі формату SOAP (Simple Object Access Protocol, це XML розмітка, яка використовується в веб-сервісах). Все, відправляється через HTTP, спочатку перетворюється в XML опис SOAP, потім застосовується в HTTP пакеті та надсилається на інший комп'ютер в мережі по TCP/IP. WSDL (Web Services Description Language) представляє собою формат XML файлу для опису сервісів мережі як набору кінцевих операцій, які працюють за допомогою повідомлень, які містять документо-орієнтовану інформацію. Документ WSDL повністю описує зовнішній інтерфейс веб-сервісу. Він надає інформацію про послуги, які можна отримати, скориставшись методами сервісу, і способи звернення до цих методів [7].

Далі розглянемо загальний підхід використання веб-сервісу. У веб-сервісах завжди є клієнт і сервер. Сервер — це веб-сервіс (endpoint: кінцева точка, куди доходять SOAP повідомлення від клієнта). Основні кроки реалізації: (i) опис інтерфейсу веб-сервісу; (ii) реалізація інтерфейс; (iii) запуск веб-сервісу; (iv) створення клієнта і віддалений виклик потрібного методу веб-сервісу.

Веб-сервіси складають єдину концепцію створення таких додатків, функції яких використовують за допомогою стандартних протоколів Інтернет. Реалізація виконується за допомогою технологій, які стандартизовані World Wide Web Consortium (W3C) [7]:

Але все частіше з'являється потреба у користувача в будь-який момент часу мати можливість в нехтуванні ресурсами, яка робоча станція не в змозі забезпечити. Підвищення продуктивності обчислень до прийнятного рівня часто можна досягнути шляхом перерозподілу обчислювальних ресурсів між задачами. Основними характеристиками хмарних обчислень є масштабованість, еластичність, мобільність, необмежений обсяг даних, що обробляються, та можливість нарощувати ресурси [1].

Механізм балансування навантаження

Метою даного дослідження є оцінка можливостей запропонованого методу динамічного балансування навантаження в хмарному середовищі та визначення такого розподілу завдань, який забезпечує приблизно однакове обчислювальне навантаження та мінімальні витрати на передачу даних між ними. Динамічне балансування передбачає перерозподіл обчислювального навантаження на вузли під час виконання програми. Технологія управління розподіленими ресурсами є одною з найважливіших задач, яка направлена на забезпечення керування інформаційної інфраструктури в умовах значного зростання навантаження та збільшення кількості компонентів мережі.

Балансування навантаження в хмарному середовищі відрізняється від класичної моделі архітектури балансування навантаження та її впровадження за допомогою серверів для виконання балансування навантаження, оскільки важко передбачити кількість запитів, які будуть передаватися на сервер. Це забезпечує нові можливості, а також представляє свій унікальний комплекс завдань. Балансування навантаження є однією з центральних проблем в області хмарних обчислень [8]. Це механізм, який рівномірно розподіляє динамічне локальне навантаження на всіх вузлах по всій хмарі, щоб уникнути ситуації, коли деякі вузли сильно завантажуються, тоді як інші не працюють. Це допомагає досягти підвищення кількості користувачів та використання ресурсів, а отже, покращення загальної ефективності та економічної вигоди ресурсів системи. Він також гарантує, що кожен обчислювальний ресурс розподіляється ефективно та справедливо [9]. Проблема балансування обчислювального навантаження розподіленого додатка виникає з тих причин, що [5]: (i) неоднорідна структура розподіленого додатка: різні логічні процеси вимагають різні обчис-

лювальні потужності; (ii) неоднорідна структура обчислювального комплексу: різні обчислювальні вузли характеризуються різною продуктивністю; (iii) неоднорідна структура міжвузлової взаємодії: лінії зв'язку, що з'єднують вузли, можуть мати різні характеристики пропускної спроможності.

Концептуально схеми балансування навантаження можна розділити на два типи: статичні та динамічні. Статичне балансування виконується на етапі проектування розподіленого додатка. Дуже часто при розподілі логічних процесів на процесори використовується досвід попередніх запусків, застосовуються генетичні алгоритми. Однак попереднє розміщення логічних процесів між процесорами неефективне. Первинною метою оптимізації балансування навантаження є перерозподіл збалансованого навантаження за допомогою завдань та мінімізація потреб між процесами зв'язку з оптимальним використанням ресурсів та часом роботи. Отже, покращення продуктивності обчислювальних вузлів шляхом вирівнювання робочих навантажень елементів обробки є метою балансування навантаження. Балансування навантаження може здійснюватися за допомогою як апаратних, так і програмних інструментів. Застосування динамічного балансування з урахуванням поточного завантаження серверів дозволяє побудувати хмарне середовище, яке оптимально використовує всі наявні ресурси.

Балансування навантаження досягається в середовищі хмари у два етапи: по-перше, це розподіл завдання серед вузлів, другий полягає в тому, щоб відстежувати віртуальну машину та виконувати операцію балансування навантаження за допомогою міграції завдань або підходу міграції віртуальної машини. Метою планування завдань є створення графіка і присвоєння кожного завдання вузлу (віртуальній машині) за певний період часу, так що всі завдання виконуються за мінімальний проміжок часу.

Три основні етапи потрібні для планування завдання в хмарному середовищі. На етапі користувача подаються робочі місця через графічний інтерфейс користувача або веб-інтерфейс з вимогою на обслуговування в частині якості обслуговування (QoS, Quality of service), апаратного забезпечення, програмного забезпечення тощо. На етапі планувальника фазових завдань виконуються всі завдання планування завдання та операції балансування навантаження. Обробник запиту на роботу переадресовує автентичний запит до планувальника завдань для подальшої обробки, де формується відповідь всім завданням відповідної віртуальної машини та призначеного завдання. Планувальник завдань містить інформацію про стан всіх вузлів (незайняті або зайняті). На останньому етапі в хмарному середовищі формується базова архітектура планування виконання завдань. Такий етап називається фазою на рівні хмари. Центр обробки даних містить хости, і кожен хост містить віртуальну машину.

Статичне балансування навантаження

Статичний алгоритм балансування навантаження потребує додаткової інформації про кількість завдань та інформацію про наявний ресурс. Коли статичний алгоритм працює, немає необхідності постійно стежити за ресурсом. В роботі [10] запропоновано вдосконалений алгоритм заповнення (IBA) за допомогою методу збалансованої спіралі (BS) для зменшення часу обробки задач, алгоритм IBA забезпечує гарантію якості обслуговування в хмарному середовищі, але цей алгоритм неефективний, коли завдання потрапляє у випадковому порядку в систему. Під часом обробки задачі мається на увазі загальний час міграції завдання. Досягнення кращої якості обслуговування з високим використанням ресурсів запропоновано в роботі [11] завдяки алгоритму IBA з EASY для планування завдання в середовищі хмари. Коли користувач надсилає запит на послугу, він також додає якість параметрів послуг, таких як кінцевий термін, пріоритет, вартість тощо. Інший алгоритм [12] визначає час виконання завдань, враховуючи пріоритет завдання, тобто першорядне завдання буде виконано, по-перше, після виконання непріоритетного завдання.

Динамічне балансування навантаження

Алгоритм динамічного навантаження не передбачає жодних попередніх відомостей про дії в глобальному стані системи, базується виключно на існуючому або поточному стані системи, тобто існує можливість балансування навантаження. В розподіленій системи алгоритм динамічного навантаження виконується всіма вузлами, які присутні в системі, і завдання балансування навантаження розподіляється між ними. Взаємодія між вузлами для здійснення балансування навантаження може мати дві форми: кооперативну та некооперативну. Алгоритми динамічного розподілу навантаження, що мають розподілений характер, часто виробляють більше повідомлень, ніж нерозподілені, оскільки кожний з вузлів у системі повинен взаємодіяти з кожним іншим вузлом. Перевага такого підходу полягає в тому, що навіть якщо один або декілька вузлів не активуються, це не призведе до зупинки всього процесу балансування навантаження [5].

Вищезгадані алгоритми не підходять для середовища в реальному часі, де навантаження на вузол дуже часто змінюються, тобто не можна передбачити майбутнє навантаження, тому будемо використовувати динамічний алгоритм. Немає потреби в попередній інформації про ресурс (віртуальну машину) і завдання в динамічному алгоритмі, оскільки такий тип алгоритму постійно контролює ресурс. А. Лакра та Д. Ядав [13] запропонували алгоритм зменшення оберненого часу, вартості та оптимізації параметру пропускної спроможності. Існують інші типи динамічних алгоритмів, які використовують евристичний підхід, як алгоритм max-min [14] та

метауристичний підхід для вирішення задачі планування задач у середовищі хмари. Н. Цірітас та ін. [15] запропонував алгоритм зменшення часу виконання та вартості зв'язку за допомогою методу міграції задач. В роботі [16] запропоновано алгоритм динамічного навантаження з урахуванням часу проходження та визначенням середнього коефіцієнта.

В даній роботі запропоновано спосіб динамічного балансування навантаження в хмарному середовищі на основі методу міграції задач та модернізованого алгоритму Weighted Least Connections, а саме алгоритм динамічного навантаження з урахуванням часу обробки завдання та середнього коефіцієнта використання ресурсів як параметра.

Формулювання проблеми

Планування виконано n завдань в m вузлах (віртуальних машинах) має бути виконано таким чином, щоб користувач хмарного середовища міг виконувати своє завдання за мінімальний час роботи з максимальним використанням ресурсів. Планувальник задач отримує N запитів на завдання (задачі) $T_1, T_2 T_3 T_4 \ldots\ldots\ldots T_N$. У даній роботі не розглядається якість параметрів сервісу: пріоритет, вартість тощо. Всі завдання не є пріоритетними та незалежними, кожне завдання має довжину TL_1, p швидкість обробки, кількість процесорів q, обсяг основної пам'яті r та обов'язково смугу пропускання B. Планувальник задач в хмарному середовищі містить інформацію про віртуальну машину M, а саме швидкість обробки процесора, кількість процесорів, пам'ять, пропускну спроможність $VM_1, VM_2 VM_3 VM_4 \ldots\ldots\ldots VM_N$. Для обчислення потенціал окремої віртуальної машини та ємність використовуємо формулу:

$$C_{VM} = p * q \tag{1}$$

де p — швидкість обробки процесора в мільйонах інструкцій за секунду; q — кількість процесорів зайнятих для виконання завдання.

Потужність всієї віртуальної машини

$$C = \sum_{j=1}^{M} {}_{VM} \tag{2}$$

Завантаження інформації на віртуальну машину: планувальник завдань із хмарного середовища розподіляє завдання на віртуальну машину, кожна віртуальна машина має чергу для зберігання навантаження. Загальна довжина черги у віртуальній машині визначається як навантаження на цю віртуальну машину. Завантаження віртуальної машини можна розрахувати як

$$LVM_{i,\,t} = \frac{K * TL_{i(t)}}{S(VM_{i,t})} \tag{3}$$

де $K = 1, 2, 3 \ldots N$ завдання. $S(VM_{i,t})$ визначається як швидкість обслуговування віртуальної машини в момент часу t, який можна виразити у формі потужності p та кількості процесорів q як $p * x(t)$, де $x = 1, 2, 3 \ldots q$.

Навантаження віртуальної машини за часом t обчислюємо як кількість задач на конкретній віртуальній машині, поділену на швидкість обслуговування віртуальної машини. Отже, загальне навантаження на всю віртуальну машину дорівнює

$$L = \sum_{j=1}^{M} LVM_j \tag{4}$$

Якщо заплановане робоче навантаження всієї системи є більшим, ніж її потенціал, то центр обробки даних не зможе впоратися з усім майбутнім запитом, тому або відмовиться від майбутнього запиту на завдання, або збільшить віртуальну машину за допомогою концепції еластичності. Якщо майбутнє робоче навантаження менше, ніж потенціал всієї віртуальної системи, то знаходиться навантаження на кожну окрему віртуальну машину та визначається, чи є віртуальні машини, які перевантажені або завантажені. Якщо всі віртуальні машини перевантажені, то завдання переноситься на завантажену віртуальну машину так, щоб всі завдання могли бути виконані за мінімальний час. Час передачі завдання може бути розрахований TT = довжина завдання (TL) / пропускна спроможність (B).

Знаходимо час виконання завдання T_i на віртуальній машині VM_j

$$T_{exej} = \frac{\sum_i^N E_{ij} * TL_i}{p * q}, \tag{5}$$

де $E_{ij} = 1$, якщо завдання T_i призначене віртуальній машині VM_j, інакше значення = 0.

Необхідно визначити час, який складається з часу виконання та передачі завдання (за умови, що будь-яке завдання може переміщуватися з перевантаженої віртуальної машини до завантаженої).

$$MST = \max\left\{\sum_{j=1}^{M} T_{exej}\right\}. \tag{6}$$

Опис алгоритму Weighted Least Connections. Запропонований алгоритм є динамічним, адже використовується у кожній перевантаженій станції та контролюється з головної станції. Алгоритм зважених найменших зв'язків використовує «ваговий» компонент на основі відповідних можливостей кожного сервера. Але необхідно заздалегідь вказати або визначити «вагу» кожного сервера.

Балансувальник навантаження, який реалізує алгоритм зважених найменших зв'язків враховує два параметри: вагу (ємність) кожного сервера та поточну кількість клієнтів, які в даний час підключені до кожного сервера. Вибір вузла системи даним способом балансування навантаження виконується на основі кількості активних з'єднань, тобто потужності сервера. Алгоритм Weighted Least Connections

динамічний та здатний назначати «вагу» серверам «на льоту».

Аналіз алгоритму Weighted Least Connections. Недоліки алгоритму: немає ніякого сенсу застосовувати цей алгоритм для задач з короткими сесіями, характерними, наприклад, для HTTP протоколу. Для такого виду задач кращим застосуванням буде Round Robin.

Переваги алгоритму: даний алгоритм підійде для задач, пов'язаних з тривалими з'єднаннями. Наприклад, для розподілу навантаження між серверами бази даних. Якщо на деяких вузлах буде занадто багато активних підключень, нових запитів вони вже не отримають, і навпаки.

Експериментальні результати

Розрахунок часу обробки: виділяється завдання всій віртуальній машині за алгоритмом планування, після чого розпочинається моніторинг системи віртуальних машин. Завдяки цьому визначаємо стан кожної віртуальної машини, де з'ясовуємо необхідність балансування навантаження — перенесення завдань з перевантаженої віртуальної машини до менш завантаженої. Для досягнення максимальної оптимізації та виявлення достовірних результатів збільшуємо кількість повідомлень та їх довжину. Граничним значенням стає значення перевантаження всієї системи.

Середній коефіцієнт використання ресурсів (Average Resource Utilization Ratio, ARUR): основна мета розподілу навантаження полягає в тому, щоб максимально використовувати ресурси. Середній коефіцієнт використання ресурсів розраховуємо за формулою:

$$ARUR = \left(\frac{середній\ \ час}{час\ \ обробки} \right) * 100 \qquad (7)$$

де середній час = Σ часу, витраченого ресурсом (VM_j), щоб закінчити всю роботу ресурсу, де $j = \{1, 2, 3 \ldots\ldots\ldots M\}$. Діапазон середнього коефіцієнта використання ресурсів становить від 0 до 1, максимальне значення для ARUR становить 1 (використання ресурсу становить 100%), найгірше — 0 (ресурс знаходиться в ідеальному стані).

Для оцінки запропонованого методу була розроблена модель динамічного балансування навантаження в хмарному середовищі на основі методу міграції задач та модернізованого алгоритму Weighted Least Connections. У сценарії моделювання ми порівнюємо продуктивність запропонованої системи з запропонованим алгоритмом балансування навантаження і способом розподілення навантаження та без нього. Під «звичайною» мається на увазі система без використання запропонованого способу. У ході експериментальних досліджень було виявлено зменшення часу обробки завдання та збільшення середнього коефіцієнта використання ресурсів. Час обробки завдань на 5-ти віртуальних машин при 10, 20 та 30 завдань зменшився від 30% до 10% (у порівнянні з алгоритмом FCFS та з алгоритмом max-min відповідно).

Висновки. У даній роботі запропоновано метод динамічного балансування навантаження в хмарному середовищі на основі методу міграції задач та модернізованого алгоритму Weighted Least Connections, а саме алгоритм динамічного навантаження з урахуванням часу проходження та середнього коефіцієнта використання ресурсів як параметра. Метою даного методу є ефективне використання ресурсів у хмарному середовищі. Експериментальні результати показали зменшення часу обробки та збільшення середнього коефіцієнта використання ресурсів.

Література

1. Mell, P. The NIST Definition of Cloud Computing [Electronic resource] / Peter Mell, Timothy Grance. — Recommendations of the National Institute of Standards and Technology NIST. — 2011. — SP 800–145. — Електронні дані. — Режим доступу: https://csrc.nist.gov/publications/detail/sp/800–145/final

2. Bass L. Software Architecture In Practice, Second Edition / L. Bass, P. Clements, R. Kazman. — Boston: Addison-Wesley. — 2003. — ISBN0–321–15495–9. — pp. 21–24.

3. Donaldson V. Program Speedup in a Heterogeneous Computing Network / V. Donaldson, F. Berman, R. Paturi // Journal of Parallel and Distributed Computing. — September 1994. — Vol. 21. — No 3. — pp. 316–322.

4. Leopold C. Parallel and Distributed Computing. A survey of Models, Paradigms, and Approaches Wiley Series on Parallel and Distributed Computing / C. Leopold // Albert Zomaya Series Editor. — 2001.

5. Телелейко І. Аналіз методів динамічного балансування навантаження в хмарному середовищ / І. С. Телелейко // X Наукова конференція магістрантів та аспірантів «Прикладна математика та комп'ютинг — ПМК'2018». — 2018.

6. Cloud computing: distributed internet computing for IT and scientific research / [M. D. Dikaiakos, D. Katsaros, P. Mehra and others.]. — Internet Computing, IEEE. — 2009. — № 13. — pp. 10–13.

7. Телелейко І. Інтеграція сервіс-орієнтованої архітектури з grid-системами / І. С. Телелейко // VIII Наукова конференція магістрантів та аспірантів «Прикладна математика та комп'ютинг — ПМК'2016. — 2016.

8. Rima B. A Taxonomy and Survey of Cloud Computing Systems / B. P. Rima, E. Choi, and I. Lumb // Proceedings of 5th IEEE International Joint Conference on INC, IMS and IDC, Seoul. — Korea, August 2009. — pp. 44–51.

9. Alakeel A. A Guide to dynamic Load balancing in Distributed Computer Systems» / A. M. Alakeel // International Journal of Computer Science and Network Security (IJCSNS). — Vol. 10. — No. 6. — June 2010. — pp. 153–160.

10. Suresh A. Improving scheduling of backfill algorithms using balanced spiral method for cloud metascheduler / A. Suresh, P. Vijayakarthick // International Conference on Recent Trends in Information Technology. — Chennai. — 2011.

11. Dubey K. A Priority Based Job Scheduling Algorithm Using IBA and EASY Algorithm for Cloud Metaschedular / K. Dubey // International Conference on Advances in Computer Engineering and Applications. Ghaziabad. — India. — 2015.

12. Kumar M. Priority Aware Longest Job First (PA-LJF) algorithm for utilization of the resource in cloud environment / M. Kumar, S. C. Sharma // Computing for Sustainable Global Development (INDIACom). — 3rd International Conference on. IEEE. — 2016.

13. Lakra A. Multi-Objective Tasks Scheduling Algorithm for Cloud Computing Throughput Optimization / A. Lakra, D. Yadav // Procedia Computer Science. — 2015.

14. Ren X. A Dynamic Load Balancing Strategy for Cloud Computing platform based on Exponential Smoothing Forecast / X. Ren // International Conference on Cloud Computing and Intelligence Systems. — China. — 2011.

15. Tziritas N. On minimizing the resource consumption of cloud applications using process migrations / N. Tziritas // Journal of Parallel and Distributed Computing. — 2013.

16. Kumar M. Dynamic load balancing algorithm for balancing the workload among virtual machine in cloud computing / M. Kumar, S. C. Sharmab — 2017.