ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: **«ПІДВИЩЕННЯ ПРОДУКТИВНОСТІ БАЛАНСУВАЛЬНИКІВ НАВАНТАЖЕННЯ У СЕРЕДОВИЩАХ ХМАРНИХ ОБЧИСЛЕНЬ»**

на здобуття освітнього ступеня магістр
за спеціальності 123 Комп'ютерна інженерія
(код,найменування спеціальності)
освітньо-професійної програми Комп'ютерні системи та мережі
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Христина ГОЛОСУН
(підпис) *(ім'я, ПРІЗВИЩЕ здобувача)*

Виконав: здобувач вищої освіти гр.КСДМ-61
Христина ГОЛОСУН
(ім'я, ПРІЗВИЩЕ)

Керівник: Анастасія ВЄЧЕРКОВСЬКА
к.т.н, доцент *(ім'я, ПРІЗВИЩЕ)*

Рецензент: _____
науковий ступінь, вчене звання *(ім'я, ПРІЗВИЩЕ)*

Київ 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Комп'ютерної інженерії
Ступінь вищої освіти «Магістр»

Спеціальність 123 Комп'ютерна інженерія
Освітньо-професійна прорама Комп'ютерні системи та мережі

ЗАТВЕРДЖУЮ
Завідувач кафедрою Комп'ютерної інженерії
Наталія ЛАЩЕВСЬКА
(ім'я, ПРІЗВИЩЕ)
"____" _____2023 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Голосун Христині Віталіївні
(прізвище, ім'я, по батькові здобувача)

1.Тема кваліфікаційної роботи: Підвищення продуктивності балансувальників навантаження у середовищах хмарних обчислень
керівник роботи Анастасія ВЄЧЕРКОВСЬКА, к.т.н., доцент
(ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від "19" 10 2023 р. №145
2. Строк подання кваліфікаційної роботи
3. Вихідні дані кваліфікаційної роботи:
3.1. Технології хмарних обчислень.
3.2. Балансувальники навантаження.
3.3. Науково-технічна література.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
4.1. Теоретичні аспекти підвищення продуктивності балансувальників навантаження у хмарних середовищах.
4.2. Розробка новогопідходу до балансування навантаження.
4.3. Практичне дослідження розроблюваного алгоритму.

5.Перелік ілюстраційного матеріалу: *презентація*

6. Дата видачі завдання "19" жовтня 2023р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Підбір технічної літератури	19.10.2023р. 26.10.2023р.	Виконано
2.	Аналіз основ хмарних обчислень та існуючих алгоритмів балансування	26.10.2023р. 05.11.2023р.	Виконано
3.	Ідентифікація ключових параметрів алгоритму	05.11.2023р. 19.11.2023р.	Виконано
4.	Моделювання алгоритму, його оцінка та експериментальне дослідження	19.11.2023р. 03.12.2023р.	Виконано
5.	Оформлення роботи, висновків	03.12.2023р. 08.12.2023р.	Виконано
6.	Розробка демонстраційного матеріалу, доповіді	08.12.2023р. 15.12.2023р.	Виконано

Здобувач вищої освіти _____ Христина ГОЛОСУН
(підпис) *(ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Анастасія ВЄЧЕРКОВСЬКА
(підпис) *(ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття ступня магістр: 63 стор., 4 рис., 75 джерел.

Мета роботи – розробка концепції нового алгоритму балансування навантаження, здатного підвищити продуктивність роботи у середовищі хмарних обчислень.

Об'єкт дослідження – балансування навантаження у різноманітних середовищах хмарних обчислень.

Предмет дослідження – алгоритм балансування навантаження у середовищах хмарних обчислень.

Короткий зміст роботи: У роботі досліджено сучасні засоби балансування навантаження у хмарних обчисленнях та проаналізовано інсуючі алгоритми балансування.

Виконуючи наступне завдання ідентифіковано ключові параметри алгоритму, розроблена концепція нового алгоритму, проведено моделювання та оцінювання.

У практичній частині, магістерської кваліфікаційної роботи, представлено експериментальне дослідження роботи алгоритму, проведено його аналіз та визначено напрямки подальшого вдосконалення.

КЛЮЧОВІ СЛОВА: ХМАРНІ ОБЧИСЛЕННЯ, БАЛАНСУВАННЯ, НАВАНТАЖЕННЯ, АЛГОРИТМ, ПАРАМЕТРИ, КОНЦЕПЦІЯ, МОДЕЛЮВАННЯ, ХМАРНІ СЕРЕДОВИЩА.

ABSTRACT

The text part of the qualification work for obtaining a master's degree: 63 pages, 4 figures, 75 sources.

The purpose of the work is to develop a concept of a new load balancing algorithm capable of increasing work performance in a cloud computing environment.

The object of research is load balancing in various cloud computing environments.

The subject of research is a load balancing algorithm in cloud computing environments.

Summary of the work: The work examines modern means of load balancing in cloud computing and analyzes the existing balancing algorithms.

Performing the following task, the key parameters of the algorithm were identified, the concept of the new algorithm was developed, and its simulation and evaluation was carried out.

In the practical part, the master's qualification work, an experimental study of the algorithm's work is presented, its analysis is carried out, and directions for further improvement are determined.

KEY WORDS: CLOUD COMPUTING, BALANCE, LOAD, ALGORITHM, PARAMETERS, CONCEPT, SIMULATION, CLOUD ENVIRONMENTS.

ЗМІСТ

ВСТУП

Сучасний світ хмарних обчислень відіграє ключову роль у цифровій трансформації бізнесу, науки, та повсякденного життя. Із стрімким розвитком інтернет-технологій та великим зростанням обсягів даних, важливість ефективного управління ресурсами хмарних сервісів стає все більш актуальною. Одним із основних викликів у цьому контексті є балансування навантаження - процес розподілу обробки запитів та обчислювальних ресурсів між серверами, щоб оптимізувати продуктивність та забезпечити безперервність хмарних послуг.

Традиційні підходи до балансування навантаження часто стикаються з проблемами неефективного розподілу ресурсів, особливо у випадках раптових змін у навантаженні або при роботі з великими обсягами даних. Це призводить до зниження продуктивності хмарних сервісів, підвищення часу відгуку та, як наслідок, зменшення задоволеності користувачів. Така ситуація вимагає розробки нових, більш ефективних алгоритмів балансування навантаження, які б забезпечували гнучкість та масштабованість у відповідь на динамічні умови хмарних середовищ.

Основною метою цієї роботи є розробка та аналіз нового алгоритму балансування навантаження, спрямованого на покращення продуктивності та надійності хмарних обчислень. Для досягнення цієї мети визначені наступні завдання:

а) Огляд та оцінка поточних методів балансування, їх переваг та недоліків;

б) Створення концептуальної моделі алгоритму, що враховує динамічність та різноманітність хмарних обчислень;

в) Проведення експериментального дослідження для оцінки продуктивності, надійності та адаптивності розробленого алгоритму;

г) Аналіз отриманих результатів для визначення ефективності алгоритму та виявлення потенційних напрямків для подальших удосконалень;

Розробка ефективного алгоритму балансування навантаження відкриває нові перспективи для оптимізації хмарних обчислень, що має велике значення для

підвищення продуктивності хмарних сервісів та задоволеності їхніх користувачів. Успішне впровадження такого алгоритму може сприяти зниженню витрат на обслуговування хмарних систем та підвищенню їхньої конкурентоспроможності.

РОЗДІЛ 1
ТЕОРЕТИЧНІ АСПЕКТИ ПІДВИЩЕННЯ ПРОДУКТИВНОСТІ БАЛАНСУВАЛЬНИКІВ НАВАНТАЖЕННЯ У ХМАРНИХ СЕРЕДОВИЩАХ

1.1 Основи хмарних обчислень

Хмарні обчислення [1] — це доступність ресурсів комп'ютерної системи на вимогу, особливо зберігання даних (хмарне сховище) і обчислювальна потужність, без прямого активного керування користувачем [2]. Великі хмари часто мають функції, розподілені в кількох місцях, кожне з яких є центром обробки даних. Хмарні обчислення спираються на спільне використання ресурсів для досягнення узгодженості та зазвичай використовують модель оплати за використання, яка може допомогти зменшити капітальні витрати, але також може призвести до несподіваних операційних витрат для користувачів [3].

У визначенні хмарних обчислень Національним інститутом стандартів і технологій Сполучених Штатів визначено «п'ять основних характеристик»:

а) Самообслуговування на вимогу. Споживач може в односторонньому порядку надати обчислювальні можливості, такі як серверний час і мережеве сховище, за потреби автоматично, не вимагаючи взаємодії людини з кожним постачальником послуг;

б) Широкий доступ до мережі. Можливості доступні через мережу та доступ до них через стандартні механізми, які сприяють використанню неоднорідними тонкими або товстими клієнтськими платформами (наприклад, мобільними телефонами, планшетами, ноутбуками та робочими станціями);

в) Об'єднання ресурсів. Обчислювальні ресурси постачальника об'єднуються для обслуговування кількох споживачів за допомогою моделі з кількома клієнтами, причому різні фізичні та віртуальні ресурси динамічно призначаються та перепризначаються відповідно до попиту споживачів;

г) Швидка еластичність. Можливості можна еластично надавати та вивільняти, у деяких випадках автоматично, для швидкого масштабування назовні та всередину відповідно до попиту. Для споживача можливості, доступні для забезпечення, часто здаються необмеженими і можуть бути використані в будь-якій кількості в будь-який час;

д) Розмірене обслуговування. Хмарні системи автоматично контролюють і оптимізують використання ресурсів, використовуючи можливості вимірювання на певному рівні абстракції, що відповідає типу послуги (наприклад, зберігання, обробка, пропускна здатність і активні облікові записи користувачів). Використання ресурсів можна відстежувати, контролювати та звітувати, забезпечуючи прозорість як для постачальника, так і для споживача використовуваної послуги [4];

Хмарні обчислення мають багату історію, яка сягає 1960-х років, коли початкові концепції розподілу часу стали популяризувати через дистанційне введення вакансій (RJE). У цю епоху переважно використовувалася модель «центру обробки даних», коли користувачі надсилали завдання операторам для виконання на мейнфреймах. Це був час досліджень і експериментів із способами зробити великомасштабну обчислювальну потужність доступною для більшої кількості користувачів за допомогою розподілу часу, оптимізації інфраструктури, платформи та програм, а також підвищення ефективності для кінцевих користувачів [5].

Використання метафори «хмара» для позначення віртуалізованих служб бере свій початок у 1994 році, коли General Magic використовувала її для опису всесвіту «місць», куди можуть потрапити мобільні агенти в середовищі Telescript. Цю метафору приписують Девіду Хоффману, співробітнику відділу комунікацій компанії General Magic, оскільки вона давно використовується в мережах і телекомунікаціях [6]. Вираз хмарних обчислень став більш відомим у 1996 році, коли Compaq Computer Corporation склала бізнес-план майбутнього обчислень та Інтернету. Амбіція компанії полягала в тому, щоб збільшити продажі за допомогою «додатків з підтримкою хмарних обчислень». У бізнес-плані передбачалося, що

онлайнове сховище файлів споживача, швидше за все, буде комерційно успішним. У результаті Compaq вирішила продати серверне обладнання постачальникам послуг Інтернету [7].

У 2000-х роках застосування хмарних обчислень почало формуватися зі створенням Amazon Web Services (AWS) у 2002 році, що дозволило розробникам створювати програми самостійно. У 2006 році була випущена бета-версія Google Docs, служба Amazon Simple Storage Service, відома як Amazon S3, і Amazon Elastic Compute Cloud (EC2), у 2008 році NASA розробила перше програмне забезпечення з відкритим кодом для розгортання приватних і гібридних хмар [8] - [9].

Наступне десятиліття стало початком різноманітних хмарних сервісів. У 2010 році Microsoft запустила Microsoft Azure, а Rackspace Hosting і NASA ініціювали проект хмарного програмного забезпечення з відкритим кодом OpenStack. IBM представила структуру IBM SmartCloud у 2011 році, а Oracle анонсувала Oracle Cloud у 2012 році. У грудні 2019 року Amazon запустила AWS Outposts, сервіс, який розширює інфраструктуру, служби, API та інструменти AWS до центрів обробки даних клієнтів, просторів спільного розміщення, або на території приміщення [10] - [11].

Після глобальної пандемії 2020 року хмарна технологія набула популярності завдяки рівню безпеки даних, яку вона пропонує, і гнучкості варіантів роботи, які вона надає всім співробітникам, особливо віддаленим [12].

Прихильники публічних і гібридних хмар стверджують, що хмарні обчислення дозволяють компаніям уникнути або мінімізувати початкові витрати на IT-інфраструктуру. Прихильники також стверджують, що хмарні обчислення дозволяють підприємствам швидше запускати свої додатки з покращеною керованістю та меншим обслуговуванням, і що це дозволяє IT-командам швидше налаштовувати ресурси відповідно до коливань і непередбачуваного попиту [13] - [15]. Забезпечення пакетних обчислювальних можливостей: висока обчислювальна потужність у певні періоди пікового попиту [16].

До додаткових цінних пропозицій хмарних обчислень належать:

а) Зменшення витрат. Модель надання публічної хмари перетворює капітальні витрати (наприклад, придбання серверів) на операційні витрати [17]. Це нібито знижує бар'єри для входу, оскільки інфраструктура зазвичай надається третьою стороною, і її не потрібно купувати для одноразових або рідкісних інтенсивних обчислювальних завдань. Ціноутворення на основі комунальних обчислень є «тонким» із варіантами виставлення рахунків на основі використання. Крім того, для реалізації проектів, які використовують хмарні обчислення, потрібно менше внутрішніх навичок IT [18]. Сучасний репозиторій проекту e-FISCAL [19] містить кілька статей, у яких більш детально розглядаються аспекти витрат, більшість із яких робить висновок, що економія коштів залежить від типу діяльності, що підтримується, і типу наявної внутрішньої інфраструктури;

б) Незалежність від пристрою та місця розташування [20], що дозволяє користувачам отримувати доступ до систем за допомогою веб-браузера незалежно від їхнього місцезнаходження чи пристрою, який вони використовують (наприклад, ПК, мобільний телефон). Оскільки інфраструктура знаходиться за межами сайту (як правило, надається третьою стороною) і доступ до неї здійснюється через Інтернет, користувачі можуть підключитися до неї з будь-якого місця [18];

в) Обслуговування хмарного середовища, що відбувається легше, оскільки дані розміщуються на зовнішньому сервері, який обслуговується постачальником, без необхідності інвестувати в обладнання центру обробки даних. IT-обслуговуванням хмарних обчислень керує команда IT-технічного обслуговування постачальника хмарних технологій, що зменшує витрати на хмарні обчислення порівняно з локальними центрами обробки даних;

г) Багатокористування, яке дозволяє розподіляти ресурси та витрати між великою групою користувачів, що, в свою чергу, дозволяє централізовувати інфраструктури в місцях з меншими витратами (наприклад, нерухомість, електроенергія тощо), збільшувати пропускну здатність пікового навантаження (користувачам не потрібно розробляти та платити за ресурси та обладнання, щоб відповідати найвищим можливим рівням навантаження), використовувати та

підвищувати ефективність систем, які часто використовуються лише на 10–20% [21] - [22];

д) Продуктивність, що контролюється IT-експертами постачальника послуг, а узгоджені та слабко пов’язані архітектури будуються з використанням веб-сервісів як системного інтерфейсу [18], [23]. Продуктивність можна підвищити, коли кілька користувачів можуть одночасно працювати з тими самими даними, а не чекати, поки вони будуть збережені та відправлені електронною поштою. Можна заощадити час, оскільки не потрібно повторно вводити інформацію, коли поля збігаються, а також користувачам не потрібно встановлювати оновлення прикладного програмного забезпечення на свій комп’ютер;

е) Доступність, котра покращується завдяки використанню кількох резервних сайтів, що робить добре розроблені хмарні обчислення придатними для безперервності бізнесу та аварійного відновлення [24];

ж) Масштабованість і еластичність, що за допомогою динамічного («на вимогу») надання ресурсів на детальній основі самообслуговування майже в режимі реального часу [25] - [26]. Зверніть увагу, час запуску віртуальної машини залежить від типу віртуальної машини, місцезнаходження, ОС і хмарні постачальники [25], без необхідності користувачам проектувати для пікових навантажень [27] - [29]. Це дає можливість збільшувати масштаб, коли потреба у використанні зростає, або зменшувати, якщо ресурси не використовуються [30]. Перевага хмарної масштабованості, що сприяє економії часу, також означає швидший час виходу на ринок, більшу гнучкість бізнесу та адаптивність, оскільки додавання нових ресурсів не займає стільки часу, скільки раніше [31]. Нові підходи до управління еластичністю включають використання методів машинного навчання, щоб запропонувати ефективні моделі еластичності [32];

и) Безпека, яка може покращитися завдяки централізації даних, збільшенню ресурсів, орієнтованих на безпеку, тощо, але може залишатися занепокоєння щодо втрати контролю над певними конфіденційними даними та відсутності безпеки для збережених ядер. Безпека часто така ж хороша, як і краща за інші традиційні системи, частково тому, що постачальники послуг можуть виділяти ресурси для

вирішення проблем безпеки, які багато клієнтів не можуть собі дозволити вирішити або для вирішення яких їм бракує технічних навичок [33]. Однак складність безпеки значно збільшується, коли дані розподіляються на більшій території або на більшій кількості пристроїв, а також у системах із кількома клієнтами, якими спільно користуються непов'язані користувачі. Крім того, доступ користувача до журналів аудиту безпеки може бути складним або неможливим. Встановлення приватної хмари частково мотивується бажанням користувачів зберегти контроль над інфраструктурою та уникнути втрати контролю над інформаційною безпекою;

Однією з головних проблем хмарних обчислень, порівняно з більш традиційними локальними обчисленнями, є безпека даних і конфіденційність. Користувачі хмари довіряють свої конфіденційні дані стороннім постачальникам, які можуть не мати належних заходів для захисту від несанкціонованого доступу, порушень або витоків. Користувачі хмари також стикаються з ризиками недотримання нормативних вимог, якщо їм потрібно дотримуватися певних правил або стандартів щодо захисту даних, таких як GDPR або HIPAA [34].

Іншою проблемою хмарних обчислень є зниження видимості та контролю. Користувачі хмари можуть не мати повного уявлення про те, як їхні хмарні ресурси керуються, налаштовуються чи оптимізуються їхніми постачальниками. Вони також можуть мати обмежені можливості налаштовувати або змінювати свої хмарні служби відповідно до своїх конкретних потреб або вподобань [34]. Повне розуміння всіх технологій може бути неможливим, особливо враховуючи масштаб, складність і навмисну непрозорість сучасних систем; однак існує потреба в розумінні складних технологій та їхніх взаємозв'язків, щоб мати в них силу та право власності [35]. Метафору хмари можна розглядати як проблематичну, оскільки хмарні обчислення зберігають ауру чогось ноуменального та нумінозного; це щось досвідчене без точного розуміння, що це таке або як це працює [36].

Крім того, важливою проблемою є хмарна міграція. Хмарна міграція – це процес переміщення даних, програм або робочих навантажень з одного хмарного середовища в інше або з локального середовища в хмару. Хмарна міграція може бути складною, довготривалою та дорогою, особливо якщо існують проблеми

несумісності між різними хмарними платформами чи архітектурами. Хмарна міграція також може спричинити простої, погіршення продуктивності або втрату даних, якщо її не спланувати та виконати належним чином [37].

Сервісно-орієнтована архітектура (SOA) просуває ідею «Все як послуга» (EaaS або XaaS, або просто aAsS) [38]. Ця концепція реалізується в хмарних обчисленнях за допомогою кількох моделей обслуговування, визначених Національним інститутом стандартів і технологій (NIST). Три стандартні моделі обслуговування: інфраструктура як послуга (IaaS), платформа як послуга (PaaS) і програмне забезпечення як послуга (SaaS) [4]. Вони зазвичай зображуються як шари в стеку, що забезпечує різні рівні абстракції. Однак ці шари не обов'язково взаємозалежні. Наприклад, SaaS можна надавати на «голому металі», минаючи PaaS і IaaS, а програма може працювати безпосередньо на IaaS, не будучи упакованою як SaaS.

«Інфраструктура як послуга» (IaaS) відноситься до онлайн-сервісів, які надають API високого рівня, які використовуються для абстрагування різних низькорівневих деталей базової мережевої інфраструктури, як-от фізичні обчислювальні ресурси, місцезнаходження, розподіл даних, масштабування, безпека, резервне копіювання тощо. гіпервізор запускає віртуальні машини як гості. Пули гіпервізорів у хмарній операційній системі можуть підтримувати велику кількість віртуальних машин і можливість масштабувати послуги відповідно до різноманітних вимог клієнтів. Контейнери Linux працюють в ізольованих розділах одного ядра Linux, яке працює безпосередньо на фізичному обладнанні. Контрольні групи та простори імен Linux — це базові технології ядра Linux, які використовуються для ізоляції, захисту та керування контейнерами. Використання контейнерів забезпечує вищу продуктивність, ніж віртуалізація, оскільки немає накладних витрат на гіпервізор. Хмари IaaS часто пропонують додаткові ресурси, такі як бібліотека образів дисків віртуальної машини, сховище необроблених блоків, сховище файлів або об'єктів, брандмауери, балансувальники навантаження, IP-адреси, віртуальні локальні мережі (VLAN) і комплекти програмного забезпечення [39].

Визначення хмарних обчислень NIST описує IaaS як «де споживач може розгортати та запускати довільне програмне забезпечення, яке може включати операційні системи та програми. Споживач не керує базовою хмарною інфраструктурою та не контролює її, але має контроль над операційними системами, сховищем, і розгорнуті додатки; і, можливо, обмежений контроль вибраних мережевих компонентів (наприклад, брандмауери хостів) [4].

Провайдери IaaS-хмари надають ці ресурси на вимогу зі своїх великих пулів обладнання, встановленого в центрах обробки даних. Для широкого підключення клієнти можуть використовувати або Інтернет, або хмари операторів (виділені віртуальні приватні мережі). Щоб розгорнути свої програми, користувачі хмари встановлюють образи операційної системи та прикладне програмне забезпечення в хмарній інфраструктурі. У цій моделі користувач хмари вносить виправлення та підтримує операційні системи та прикладне програмне забезпечення. Хмарні провайдери зазвичай виставляють рахунки за послуги IaaS на основі корисних обчислень: вартість відображає кількість виділених і спожитих ресурсів [40].

Платформа як послуга (PaaS). Розробники програм PaaS пропонують середовище розробки. Постачальник зазвичай розробляє набір інструментів і стандарти для розробки та каналів розповсюдження та оплати. У моделях PaaS хмарні постачальники надають обчислювальну платформу, яка зазвичай включає операційну систему, середовище виконання мовою програмування, базу даних і веб-сервер. Розробники додатків розробляють і запускають своє програмне забезпечення на хмарній платформі замість того, щоб безпосередньо купувати базові апаратні та програмні рівні та керувати ними. З деякими PaaS основний комп’ютер і ресурси зберігання автоматично масштабуються відповідно до вимог програми, тому користувачеві хмари не потрібно розподіляти ресурси вручну [41].

Деякі постачальники інтеграції та керування даними також використовують спеціалізовані програми PaaS як моделі доставки даних. Приклади включають iPaaS (інтеграційна платформа як послуга) і dPaaS (платформа даних як послуга). iPaaS дозволяє клієнтам розробляти, виконувати та керувати потоками інтеграції [42]. Відповідно до моделі інтеграції iPaaS клієнти керують розробкою та

розгортанням інтеграцій без встановлення та керування будь-яким апаратним чи проміжним програмним забезпеченням [43]. dPaaS надає продукти інтеграції та керування даними як повністю керовану послугу [44]. Згідно з моделлю dPaaS, постачальник PaaS, а не клієнт, керує розробкою та виконанням програм, створюючи програми даних для клієнта. Користувачі dPaaS отримують доступ до даних за допомогою інструментів візуалізації даних [45].

Програмне забезпечення як послуга (SaaS). У моделі програмного забезпечення як послуги (SaaS) користувачі отримують доступ до прикладного програмного забезпечення та баз даних. Хмарні постачальники керують інфраструктурою та платформами, на яких запускаються програми. SaaS іноді називають «програмним забезпеченням на вимогу», і зазвичай ціна визначається на основі плати за використання або з використанням абонентської плати [46]. У моделі SaaS хмарні провайдери встановлюють і експлуатують прикладне програмне забезпечення в хмарі, а користувачі хмари отримують доступ до програмного забезпечення з хмарних клієнтів. Користувачі хмари не керують хмарною інфраструктурою та платформою, на якій працює програма. Це позбавляє від необхідності встановлювати та запускати програму на власних комп’ютерах користувача хмари, що спрощує технічне обслуговування та підтримку. Хмарні програми відрізняються від інших програм своєю масштабованістю, якої можна досягти шляхом клонування завдань на кількох віртуальних машинах під час виконання, щоб задовольнити мінливий робочий попит [47]. Балансувальники навантаження розподіляють роботу по набору віртуальних машин. Цей процес прозорий для користувача хмари, який бачить лише одну точку доступу. Щоб забезпечити роботу великої кількості користувачів хмари, хмарні програми можуть бути мультитенантними, тобто будь-яка машина може обслуговувати більше однієї організації хмарних користувачів.

Модель ціноутворення для додатків SaaS зазвичай являє собою щомісячну або річну фіксовану плату за користувача [48], тому ціни стають масштабованими та змінними, якщо користувачів додають або видаляють у будь-який момент. Це також може бути безкоштовним [49]. Прихильники стверджують, що SaaS дає

компанії потенціал для зниження операційних витрат на IT за рахунок аутсорсингу обслуговування та підтримки апаратного та програмного забезпечення постачальнику хмарних технологій. Це дозволяє компанії перерозподіляти витрати на IT-операції з витрат на апаратне/програмне забезпечення та з витрат на персонал для досягнення інших цілей. Крім того, за допомогою програм, розміщених централізовано, оновлення можна випускати без необхідності користувачам інсталювати нове програмне забезпечення. Одним із недоліків SaaS є збереження даних користувачів на сервері хмарного провайдера. У результаті можливий несанкціонований доступ до даних [50]. Прикладами програм, які пропонуються як SaaS, є ігри та програмне забезпечення для підвищення продуктивності, наприклад Google Docs і Office Online. Програми SaaS можуть бути інтегровані з хмарним сховищем або службами розміщення файлів, як у випадку Google Docs, інтегрованого з Google Drive, і Office Online, інтегрованого з OneDrive [51].

Моделі розгортання

Приватна хмара — це хмарна інфраструктура, що експлуатується виключно для однієї організації, незалежно від того, керується вона внутрішньо чи третьою стороною, і розміщується як всередині, так і зовні [4]. Реалізація проекту приватної хмари вимагає значної участі у віртуалізації бізнес-середовища та вимагає від організації перегляду рішень щодо існуючих ресурсів. Це може покращити бізнес, але кожен крок у проекті викликає проблеми безпеки, які необхідно вирішити, щоб запобігти серйозним уразливостям. Самокеровані центри обробки даних [61], як правило, є капіталомісткими. Вони мають значний фізичний слід, вимагаючи виділення простору, апаратного забезпечення та контролю середовища. Ці активи необхідно періодично оновлювати, що призводить до додаткових капітальних витрат. Вони викликали критику, тому що користувачі «все ще повинні купувати, створювати та керувати ними» і, таким чином, не отримують переваги від меншого практичного управління [62], по суті «[відсутність] економічної моделі, яка робить хмарні обчислення такою інтригуючою концепцією» [63] - [64].

Громадська хмара. Хмарні послуги вважаються «загальнодоступними», коли вони надаються через публічний Інтернет, і вони можуть пропонуватися як платна підписка або безкоштовно [65]. Архітектурно відмінностей між загальнодоступними та приватними хмарними службами мало, але проблеми з безпекою значно зростають, коли послуги (програми, сховище та інші ресурси) використовуються спільно кількома клієнтами. Більшість постачальників загальнодоступних хмарних послуг пропонують послуги прямого підключення, які дозволяють клієнтам безпечно зв'язувати свої застарілі центри обробки даних зі своїми хмарними додатками [18], [66]. Кілька факторів, як-от функціональність рішень, вартість, інтеграційні та організаційні аспекти, а також безпека та безпека, впливають на рішення підприємств і організацій щодо вибору публічної хмари чи локального рішення [67].

Гібридна хмара — це композиція загальнодоступної хмари та приватного середовища, наприклад приватної хмари або локальних ресурсів [68], [69], які залишаються окремими об'єктами, але пов'язані разом, пропонуючи переваги кількох моделей розгортання. Гібридна хмара також може означати можливість підключення спільного розташування, керованих та/або виділених служб із хмарними ресурсами [4]. Gartner визначає гібридний хмарний сервіс як сервіс хмарних обчислень, який складається з деякої комбінації приватних, публічних і громадських хмарних сервісів від різних постачальників послуг [70]. Гібридна хмарна служба перетинає межі ізоляції та постачальників, тому її не можна просто віднести до однієї категорії приватної, загальнодоступної чи спільнотної хмарної служби. Це дозволяє розширити пропускну здатність або можливості хмарної служби шляхом агрегування, інтеграції або налаштування з іншою хмарною службою. Існують різноманітні випадки використання композиції гібридної хмари. Наприклад, організація може зберігати конфіденційні клієнтські дані в приватній хмарній програмі, але підключити цю програму до програми бізнес-аналітики, наданої в публічній хмарі як програмна служба [71]. Цей приклад гібридної хмари розширює можливості підприємства щодо надання певних бізнес-послуг шляхом додавання зовнішніх доступних публічних хмарних служб. Впровадження

гібридної хмари залежить від ряду факторів, таких як вимоги до безпеки та відповідності даних, рівень необхідного контролю над даними та програми, які використовує організація [72]. Іншим прикладом гібридної хмари є той, де IT-організації використовують публічні хмарні обчислювальні ресурси для задоволення тимчасових потреб у потужності, які не можуть бути задоволені приватною хмарою [73]. Ця можливість дозволяє гібридним хмарам використовувати розрив хмари для масштабування між хмарами.[4] Cloud bursting — це модель розгортання програми, за якої програма працює в приватній хмарі або центрі обробки даних і «вивертається» в загальнодоступну хмару, коли потреба в обчислювальній потужності зростає. Основна перевага розривної хмари та моделі гібридної хмари полягає в тому, що організація платить за додаткові обчислювальні ресурси лише тоді, коли вони потрібні [74]. Хмарний розрив дозволяє центрам обробки даних створювати внутрішню IT-інфраструктуру, яка підтримує середні робочі навантаження, і використовувати хмарні ресурси з загальнодоступних або приватних хмар під час сплесків вимог до обробки [75].

Хмарна архітектура, системна архітектура систем програмного забезпечення, задіяних у доставці хмарних обчислень, як правило, включає в себе кілька хмарних компонентів, які спілкуються один з одним через механізм слабкого зв’язку, такий як черга обміну повідомленнями. Пружне забезпечення означає розумне використання жорсткого або слабкого зчеплення, що стосується таких механізмів, як ці та інші.

Хмарна інженерія — це застосування інженерних дисциплін хмарних обчислень. Він забезпечує системний підхід до проблем високого рівня комерціалізації, стандартизації та управління в задумуванні, розробці, експлуатації та підтримці систем хмарних обчислень. Це міждисциплінарний метод, який охоплює внески з різних областей, таких як системи, програмне забезпечення, Інтернет, продуктивність, інженерія інформаційних технологій, безпека, платформа, ризики та інженерія якості.

Хмарні обчислення викликають проблеми з конфіденційністю, оскільки постачальник послуг може отримати доступ до даних, які знаходяться в хмарі, у

будь-який час. Це може випадково або навмисно змінити чи видалити інформацію. Багато хмарних провайдерів можуть ділитися інформацією з третіми сторонами, якщо це необхідно для цілей правопорядку без ордера. Це дозволено їх політикою конфіденційності, з якою користувачі повинні погодитися, перш ніж почати використовувати хмарні служби. Рішення конфіденційності включають політику та законодавство, а також вибір кінцевими користувачами щодо того, як зберігати дані. Користувачі можуть шифрувати дані, які обробляються або зберігаються в хмарі, щоб запобігти несанкціонованому доступу. Системи керування ідентифікацією також можуть запропонувати практичні рішення проблем конфіденційності в хмарних обчисленнях. Ці системи розрізняють авторизованих і неавторизованих користувачів і визначають обсяг даних, доступних для кожної організації. Системи працюють, створюючи та описуючи ідентифікаційні дані, записуючи дії та позбавляючись від невикористаних ідентифікаційних даних.

За даними Cloud Security Alliance, трійкою найбільших загроз у хмарі є незахищені інтерфейси та API, втрата та витік даних і збій обладнання, на які припадає 29%, 25% та 10% усіх збоїв у безпеці хмари відповідно. Разом вони утворюють спільні технологічні вразливості. На платформі хмарного провайдера, яка використовується різними користувачами, може існувати ймовірність того, що інформація, що належить різним клієнтам, знаходиться на одному сервері даних. Крім того, Євген Шульц, головний технічний директор Emagined Security, сказав, що хакери витрачають багато часу та зусиль на пошуки способів проникнення в хмару. «У хмарній інфраструктурі є справжні ахіллесові п’яти, які роблять великі діри, куди можуть потрапити погані хлопці». Оскільки дані сотень або тисяч компаній можуть зберігатися на великих хмарних серверах, хакери теоретично можуть отримати контроль над величезними сховищами інформації за допомогою однієї атаки — процес, який він назвав «гіперзломом». Деякі приклади цього включають порушення безпеки Dropbox і витік iCloud 2014. У жовтні 2014 року Dropbox було зламано, хакери вкрали понад 7 мільйонів паролів користувачів, намагаючись отримати від нього грошову вартість у біткойнах (BTC). Маючи ці

паролі, вони можуть читати особисті дані, а також індексувати ці дані пошуковими системами (роблячи інформацію загальнодоступною).

Однорангові (P2P) обчислення або мережа — це розподілена архітектура додатків, яка розподіляє завдання або навантаження між одноранговими користувачами. Однорангові вузли є однаково привілейованими, еквіпотентними учасниками мережі, які утворюють однорангову мережу вузлів [1].

Однорангові вузли роблять частину своїх ресурсів, таких як обчислювальна потужність, дискове сховище або пропускна здатність мережі, безпосередньо доступними іншим учасникам мережі, без необхідності централізованої координації серверами або стабільними хостами [2]. Однорангові партнери є як постачальниками, так і споживачами ресурсів, на відміну від традиційної моделі клієнт-сервер, у якій споживання та постачання ресурсів розділені [3].

Хоча системи P2P раніше використовувалися в багатьох сферах застосування [4], ця концепція була популяризована системами обміну файлами, такими як програма для обміну музикою Napster (спочатку випущена в 1999 році). Рівноправний рух дозволив мільйонам користувачів Інтернету підключатися «безпосередньо, формуючи групи та співпрацюючи, щоб стати створеними користувачами пошуковими системами, віртуальними суперкомп’ютерами та файловими системами» [6]. Основна концепція однорангових обчислень була передбачена в попередніх дискусіях про програмне забезпечення та мережеві системи, повертаючись до принципів, викладених у першому Запиті на коментарі, RFC 1 [7].

Бачення Всесвітньої павутини Тіма Бернерса-Лі було близьке до P2P-мережі, оскільки передбачалося, що кожен користувач мережі буде активним редактором і співавтором, створюючи та пов’язуючи вміст, щоб сформувати взаємопов’язану «павутину» посилань. Ранній Інтернет був більш відкритим, ніж сьогоднішній день, коли дві машини, підключені до Інтернету, могли надсилати пакети один одному без брандмауерів та інших заходів безпеки [6]. Це контрастує з подібною до трансляції структурою мережі, яка розвивалася протягом багатьох років [8] - [10]. Як попередник Інтернету, ARPANET була успішною одноранговою мережею,

де «кожен вузол-учасник міг запитувати та обслуговувати вміст». Однак ARPANET не була самоорганізованою, і їй не вистачало здатності «забезпечувати будь-які засоби для контекстної або контентної маршрутизації за межами «простої» адресної маршрутизації» [10].

Тому було створено Usenet, розподілену систему обміну повідомленнями, яку часто описують як ранню однорангову архітектуру. Вона була розроблена в 1979 році як система, яка забезпечує децентралізовану модель контролю [11]. Базова модель — це клієнт-серверна модель з точки зору користувача або клієнта, яка пропонує самоорганізований підхід до серверів груп новин. Однак сервери новин спілкуються один з одним як однорангові пристрої для поширення статей новин Usenet через всю групу мережевих серверів. Це ж міркування стосується електронної пошти SMTP у тому сенсі, що основна мережа ретрансляції електронної пошти агентів передачі електронної пошти має одноранговий характер, тоді як периферія клієнтів електронної пошти та їхні прямі з'єднання є суто клієнт-серверними відносинами.

У травні 1999 року, коли в Інтернеті з'явилися мільйони людей, Шон Фаннінг представив програму для обміну музикою та файлами під назвою Napster [10]. Napster був початком однорангових мереж, якими ми їх знаємо сьогодні, де «користувачі-учасники створюють віртуальну мережу, повністю незалежну від фізичної мережі, без необхідності підкорятися будь-яким адміністративним повноваженням чи обмеженням» [10].

Існує проблема законного володіння даними (якщо користувач зберігає деякі дані в хмарі, чи може провайдер хмари отримати від цього прибуток?). У багатьох Умовах надання послуг нічого не стосується питання власності. Фізичний контроль комп'ютерного обладнання (приватна хмара) більш безпечний, ніж розміщення обладнання за межами підприємства та під чиїмось контролем (публічна хмара). Це дає чудовий стимул для постачальників послуг публічних хмарних обчислень визначати пріоритети створення та підтримки надійного управління безпечними послугами. Деякі малі підприємства, які не мають досвіду в IT-безпеці, можуть виявити, що для них безпечніше використовувати публічну хмару. Існує ризик

того, що кінцеві користувачі не розуміють проблем, пов’язаних із входом у хмарну службу (люди іноді не читають багато сторінок умов угоди про надання послуг і просто натискають «Прийняти», не читаючи). Це важливо зараз, коли хмарні обчислення є поширеними та потрібними для роботи деяких служб, наприклад, для інтелектуального персонального помічника (Apple Siri або Google Assistant). По суті, приватна хмара вважається більш безпечною з вищим рівнем контролю для власника, однак публічна хмара вважається більш гнучкою та вимагає менше часу та грошових інвестицій від користувача.

Однорангова мережа розроблена навколо поняття рівноправних однорангових вузлів, які одночасно функціонують як «клієнти» і «сервери» для інших вузлів мережі. Ця модель мережевої організації відрізняється від моделі клієнт-сервер, де зв’язок зазвичай здійснюється до центрального сервера та від нього. Типовим прикладом передачі файлів, яка використовує модель клієнт-сервер, є служба протоколу передачі файлів (FTP), у якій клієнтська та серверна програми відрізняються: клієнти ініціюють передачу, а сервери задовольняють ці запити.

Однорангові мережі зазвичай реалізують певну форму віртуальної накладеної мережі поверх топології фізичної мережі, де вузли в накладенні утворюють підмножину вузлів у фізичній мережі. Дані все ще обмінюються безпосередньо через базову мережу TCP/IP, але на прикладному рівні однорангові вузли можуть спілкуватися один з одним напряму через логічні накладені зв’язки (кожне з яких відповідає шляху через базову фізичну мережу). Накладення використовуються для індексації та виявлення однорангових пристроїв і роблять систему P2P незалежною від топології фізичної мережі. Виходячи з того, як вузли пов’язані один з одним у накладеній мережі, а також як ресурси індексуються та розташовані, ми можемо класифікувати мережі як неструктуровані та структуровані (або як гібридні між ними) [12] - [14].

Грід-обчислення — це використання широко розподілених комп’ютерних ресурсів для досягнення спільної мети. Обчислювальну мережу можна розглядати як розподілену систему з неінтерактивними робочими навантаженнями, які

включають багато файлів. Грід-комп'ютери відрізняються від звичайних високопродуктивних обчислювальних систем, таких як кластерні обчислення, тим, що грід-комп'ютери мають кожен вузол, налаштований на виконання іншого завдання/програми. Грід-комп'ютери також мають тенденцію бути більш неоднорідними та географічно розпорошеними (тому не пов'язаними фізично), ніж кластерні комп'ютери [1]. Хоча одна сітка може бути призначена для певної програми, зазвичай сітка використовується для різних цілей. Ґріди часто створюються за допомогою програмних бібліотек проміжного програмного забезпечення ґрід загального призначення. Розміри сітки можуть бути досить великими [2].

Грід – це форма розподілених обчислень, що складається з багатьох об'єднаних в мережу слабко пов'язаних комп'ютерів, які діють разом для виконання великих завдань. Для певних додатків розподілені або мережеві обчислення можна розглядати як особливий тип паралельних обчислень, який спирається на повні комп'ютери (з вбудованими процесорами, накопичувачем, джерелами живлення, мережевими інтерфейсами тощо), підключеними до комп'ютерної мережі (приватної чи публічної) через звичайний мережевий інтерфейс, наприклад Ethernet. Це відрізняється від традиційного уявлення про суперкомп'ютер, який має багато процесорів, з'єднаних локальною високошвидкісною комп'ютерною шиною. Ця технологія була застосована до обчислювально інтенсивних наукових, математичних і академічних проблем за допомогою волонтерських обчислень, і вона використовується в комерційних підприємствах для таких різноманітних застосувань, як відкриття ліків, економічне прогнозування, сейсмічний аналіз і обробка бек-офісних даних для підтримки електронної комерція та веб-сервіси.

Грід-комп'ютери об'єднують комп'ютери з кількох адміністративних доменів для досягнення спільної мети [3], щоб вирішити одне завдання, а потім можуть зникнути так само швидко. Розмір сітки може варіюватися від невеликого (обмеженого, наприклад, мережею комп'ютерних робочих станцій у корпорації) до великого публічного співробітництва між багатьма компаніями та мережами.

«Поняття обмеженої сітки також може бути відоме як взаємодія між вузлами, тоді як поняття більшої, ширшої сітки може, таким чином, стосуватися взаємодії між вузлами» [4].

Координація програм на Grid може бути складним завданням, особливо коли координується потік інформації між розподіленими обчислювальними ресурсами. Грід-системи робочих процесів були розроблені як спеціалізована форма системи керування робочими процесами, розробленої спеціально для створення та виконання серії обчислювальних кроків або кроків обробки даних або робочого процесу в контексті сітки.

«Розподілені» або «сіткові» обчислення загалом — це особливий тип паралельних обчислень, який спирається на повні комп'ютери (з вбудованими процесорами, накопичувачами, джерелами живлення, мережевими інтерфейсами тощо), під'єднані до мережі (приватної, загальнодоступної чи Інтернету). за допомогою звичайного мережевого інтерфейсу, що виробляє товарне обладнання, порівняно з нижчою ефективністю проектування та створення невеликої кількості спеціальних суперкомп'ютерів. Основний недолік продуктивності полягає в тому, що різні процесори та локальні області зберігання не мають високошвидкісних з'єднань. Таким чином, це розташування добре підходить для додатків, у яких кілька паралельних обчислень можуть відбуватися незалежно, без необхідності передавати проміжні результати між процесорами [5]. Висока масштабованість територіально розосереджених мереж, як правило, є сприятливою через низьку потребу в з'єднанні між вузлами порівняно з пропускною здатністю публічного Інтернету [6].

Туманне обчислення [1] - [2] або туманна мережа, також відома як туманне обчислення, [3] - [4] – це архітектура, яка використовує периферійні пристрої для виконання значного обсягу обчислень (граничних обчислень), зберігання та зв'язку локально та маршрутизовано через мережу Інтернет.

У 2011 році виникла необхідність розширити хмарні обчислення за допомогою туманних обчислень, щоб впоратися з величезною кількістю пристроїв IoT і великими обсягами даних для додатків у реальному часі з низькою затримкою

[5]. Туманні обчислення, які також називають периферійними обчисленнями, призначені для розподілених обчислень, де численні «периферійні» пристрої підключаються до хмари. Слово «туман» стосується його хмароподібних властивостей, але ближче до «землі», тобто пристроїв Інтернету речей [6]. Багато з цих пристроїв генеруватимуть об’ємні необроблені дані (наприклад, із датчиків), і замість того, щоб пересилати всі ці дані на хмарні сервери для обробки, ідея туманного обчислення полягає в тому, щоб виконати якомога більше обробки за допомогою обчислювальних блоків, розташовані разом із пристроями, що генерують дані, тому пересилаються оброблені, а не необроблені дані, і вимоги до пропускної здатності зменшуються. Додаткова перевага полягає в тому, що оброблені дані, швидше за все, знадобляться тим самим пристроям, які згенерували дані, тому завдяки локальній обробці, а не віддаленій, затримка між введенням і відповіддю мінімізується. Ця ідея не зовсім нова: у сценаріях нехмарних обчислень апаратне забезпечення спеціального призначення (наприклад, мікросхеми обробки сигналів, які виконують швидке перетворення Фур’є) давно використовується для зменшення затримки та зменшення навантаження на ЦП.

Мережа Fog складається з площини керування та площини даних. Наприклад, на площині даних туманне обчислення дозволяє обчислювальним службам розташовуватися на межі мережі на відміну від серверів у центрі обробки даних. Порівняно з хмарними обчисленнями, туманні обчислення підкреслюють наближеність до цілей кінцевих користувачів і клієнтів (наприклад, операційні витрати, політика безпеки [7], використання ресурсів), щільне географічне розподілення та усвідомлення контексту (що стосується обчислювальних ресурсів і ресурсів Інтернету речей), зменшення затримки і економія магістральної пропускної здатності для досягнення кращої якості обслуговування (QoS) [8] і периферійної аналітики/потокового аналізу, що забезпечує чудову взаємодію з користувачем[9] і резервування в разі збою, а також його можна використовувати в сценаріях Assisted Living [10] - [15].

Мережа Fog підтримує концепцію Інтернету речей (IoT), згідно з якою більшість пристроїв, що використовуються людьми щодня, будуть підключені

один до одного. Приклади включають телефони, переносні пристрої для моніторингу стану здоров’я, під’єднані транспортні засоби та доповнену реальність за допомогою таких пристроїв, як Google Glass [16] - [20]. Пристрої IoT часто мають обмежені ресурси та обмежені обчислювальні можливості для виконання криптографічних обчислень. Туманний вузол може забезпечити безпеку для пристроїв IoT, виконуючи натомість ці криптографічні обчислення [21].

SPAWAR, підрозділ ВМС США, створює прототип і тестує масштабовану, безпечну мережеву мережу, стійку до збоїв, для захисту стратегічних військових активів, як стаціонарних, так і мобільних. Програми, що керують машиною, що працюють на вузлах мережі, «захоплюють», коли втрачається підключення до Інтернету. Варіанти використання включають Інтернет речей, напр. розумні зграї дронів [22].

Університет Мельбурна вирішує проблеми збору й обробки даних із камер, пристроїв ЕКГ, ноутбуків, смартфонів і пристроїв Інтернету речей за допомогою свого проекту FogBus 2, який використовує edge/fog і Oracle Cloud Infrastructure для обробки даних у режимі реального часу [23].

ISO/IEC 20248 надає метод, за допомогою якого дані об’єктів, ідентифікованих граничними обчисленнями з використанням носіїв даних автоматичної ідентифікації (AIDC), штрих-коду та/або мітки RFID, можна зчитувати, інтерпретувати, перевіряти та робити доступними для «туману» та на «Край», навіть коли тег AIDC змінився [24].

Існують також деякі відмінності між програмуванням і МК. Писати програми, які можуть працювати в середовищі суперкомп’ютера, який може мати спеціальну операційну систему або вимагати, щоб програма вирішувала проблеми паралелізму, може бути дорогим і важким. Якщо проблему можна адекватно розпаралелити, «тонкий» рівень «сіткової» інфраструктури може дозволити звичайним, автономним програмам, з урахуванням іншої частини тієї самої проблеми, працювати на кількох машинах. Це дає змогу писати та налагоджувати на одній звичайній машині та усуває ускладнення, пов’язані з тим, що кілька

екземплярів однієї програми одночасно виконуються в одній спільній пам’яті та просторі для зберігання.

Атаки, які можуть бути здійснені на системи хмарних обчислень, включають атаки «людина посередині», фішингові атаки, атаки автентифікації та атаки зловмисного програмного забезпечення. Однією з найбільших загроз вважаються атаки зловмисних програм, таких як троянські коні. Нещодавні дослідження, проведені у 2022 році, показали, що метод ін’єкції троянського коня є серйозною проблемою зі шкідливим впливом на системи хмарних обчислень.

За даними International Data Corporation (IDC), глобальні витрати на послуги хмарних обчислень досягли 706 мільярдів доларів США, а до 2025 року очікується, що вони досягнуть 1,3 трильйона доларів США. У той час як Gartner підрахувала, що глобальні витрати кінцевих користувачів публічних хмарних сервісів досягнуть 600 мільярдів доларів до 2023 року. Згідно зі звітом McKinsey & Company, важелі хмарної оптимізації витрат і орієнтовані на цінності бізнес-випадки використання прогнозують понад 1 трильйон доларів США в середньому показнику EBITDA для компаній зі списку Fortune 500 у 2030 році. За даними Gartner, у 2022 році витрати на корпоративні інформаційні технології перевищили 1,3 трлн доларів США внаслідок переходу на хмару, а у 2025 році вони зросли майже до 1,8 трлн доларів США.

Мета хмарних обчислень полягає в тому, щоб дозволити користувачам скористатися всіма цими технологіями, не потребуючи глибоких знань або досвіду роботи з кожною з них. Хмара спрямована на скорочення витрат і допомагає користувачам зосередитися на своєму основному бізнесі замість того, щоб їм заважали IT-перешкоди. Основною технологією хмарних обчислень є віртуалізація. Програмне забезпечення віртуалізації розділяє фізичний обчислювальний пристрій на один або кілька «віртуальних» пристроїв, кожний з яких можна легко використовувати та керувати для виконання обчислювальних завдань. Завдяки віртуалізації на рівні операційної системи, яка по суті створює масштабовану систему з кількох незалежних обчислювальних пристроїв, незадіяні обчислювальні ресурси можна розподіляти та використовувати ефективніше.

Віртуалізація забезпечує гнучкість, необхідну для прискорення IT-операцій, і знижує витрати за рахунок збільшення використання інфраструктури. Автономне обчислення автоматизує процес, за допомогою якого користувач може надавати ресурси на вимогу. Зводячи до мінімуму участь користувачів, автоматизація прискорює процес, зменшує витрати на робочу силу та зменшує ймовірність людських помилок.

Хмарні обчислення використовують концепції комунальних обчислень, щоб забезпечити показники використовуваних послуг. Хмарні обчислення намагаються вирішити QoS (якість обслуговування) і проблеми надійності інших моделей грід-обчислень.

Існують такі форми хмарних обчислень, як:

а) Клієнт-серверна модель – клієнт-серверні обчислення загалом позначають будь-яку розподілену програму, яка розрізняє постачальників послуг (сервери) і запитувачів послуг (клієнти).

б) Грід-обчислення – форма розподілених і паралельних обчислень, за допомогою яких «супер-і віртуальний комп'ютер» складається з кластера об'єднаних в мережу слабко пов'язаних комп'ютерів, які діють узгоджено для виконання дуже великих завдань.

в) Туманні обчислення – розподілена обчислювальна парадигма, яка надає послуги даних, обчислень, зберігання та додатків ближче до клієнта або пристроїв, які знаходяться поблизу користувача, наприклад мережевих маршрутизаторів. Крім того, fog computing обробляє дані на мережевому рівні, на інтелектуальних пристроях і на стороні клієнта кінцевого користувача (наприклад, на мобільних пристроях), замість того, щоб надсилати дані у віддалене місце для обробки.

г) Комунальне обчислення – «упаковка обчислювальних ресурсів, таких як обчислення та зберігання, як послуги з вимірюванням, подібні до традиційних комунальних послуг, таких як електроенергія».

д) Одноранговий зв'язок – розподілена архітектура без необхідності центральної координації. Учасники є як постачальниками, так і споживачами ресурсів (на відміну від традиційної клієнт-серверної моделі).

е) Хмарна пісочниця – живе ізольоване комп’ютерне середовище, у якому програма, код або файл можуть працювати, не впливаючи на програму, у якій вони працюють.

1.2 Балансування навантаження в хмарних обчисленнях

В обчислювальній техніці балансування навантаження — це процес розподілу набору завдань на набір ресурсів (обчислювальних блоків) з метою підвищення ефективності їх загальної обробки. Балансування навантаження може оптимізувати час відгуку та уникнути нерівномірного перевантаження деяких обчислювальних вузлів, коли інші обчислювальні вузли залишаються бездіяльними.

Алгоритм балансування навантаження завжди намагається вирішити конкретну проблему. Серед іншого, необхідно враховувати характер завдань, алгоритмічну складність, апаратну архітектуру, на якій будуть працювати алгоритми, а також необхідну стійкість до помилок. Тому потрібно знайти компроміс, щоб найкращим чином відповідати вимогам конкретної програми.

Ефективність алгоритмів балансування навантаження критично залежить від характеру завдань. Отже, чим більше інформації про завдання доступно на момент прийняття рішення, тим більший потенціал для оптимізації.

Досконале знання часу виконання кожного із завдань дозволяє досягти оптимального розподілу навантаження (див. алгоритм префіксної суми).[1] На жаль, насправді це ідеалізований випадок. Знати точний час виконання кожного завдання - вкрай рідкісна ситуація.

З цієї причини існує кілька методів, щоб отримати уявлення про різні часи виконання. Перш за все, у вдалому сценарії наявності завдань відносно однорідного розміру можна вважати, що кожне з них вимагатиме приблизно середнього часу виконання. Якщо, з іншого боку, час виконання дуже нерегулярний, слід використовувати більш складні методи. Одним з методів є додавання деяких метаданих до кожного завдання. Залежно від попереднього часу виконання для подібних метаданих, можна зробити висновки для майбутнього завдання на основі статистики [2].

У деяких випадках завдання залежать одне від одного. Ці взаємозалежності можна проілюструвати орієнтованим ациклічним графом. Інтуїтивно зрозуміло, що деякі завдання не можуть розпочатися, доки не будуть завершені інші.

Припускаючи, що необхідний час для кожного із завдань відомий заздалегідь, оптимальний порядок виконання повинен призвести до мінімізації загального часу виконання. Хоча це NP-складна проблема, тому її може бути важко розв’язати точно. Існують алгоритми, такі як планувальник завдань, які обчислюють оптимальний розподіл завдань за допомогою метаевристичних методів.

Іншою особливістю завдань, критично важливих для розробки алгоритму балансування навантаження, є їх здатність розбиватися на підзадачі під час виконання. Алгоритм «Деревоподібне обчислення», представлений пізніше, використовує цю специфіку.

Статичні та динамічні алгоритми

Алгоритм балансування навантаження є «статичним», коли він не враховує стан системи для розподілу завдань. Таким чином, стан системи включає такі показники, як рівень завантаження (і іноді навіть перевантаження) певних процесорів. Замість цього заздалегідь робляться припущення про загальну систему, наприклад час надходження та вимоги до ресурсів для вхідних завдань. Крім того, відомо кількість процесорів, їх відповідна потужність і швидкість передачі даних. Таким чином, статичне балансування навантаження має на меті пов’язати відомий набір завдань із доступними процесорами, щоб мінімізувати певну функцію продуктивності. Хитрість полягає в концепції цієї функції продуктивності.

Методи статичного балансування навантаження зазвичай зосереджені навколо маршрутизатора або Master, який розподіляє навантаження та оптимізує функцію продуктивності. Ця мінімізація може врахувати інформацію, пов’язану із завданнями, які потрібно розподілити, і отримати очікуваний час виконання.

Перевага статичних алгоритмів полягає в тому, що вони прості в налаштуванні та надзвичайно ефективні у випадку досить регулярних завдань (наприклад, обробка HTTP-запитів із веб-сайту). Проте все ще існує певна статистична різниця у розподілі завдань, яка може призвести до перевантаження деяких обчислювальних блоків.

На відміну від алгоритмів статичного розподілу навантаження, динамічні алгоритми враховують поточне навантаження кожного з обчислювальних блоків (також званих вузлами) у системі. У цьому підході завдання можна динамічно переміщувати з перевантаженого вузла на недостатньо завантажений, щоб отримати швидшу обробку. Хоча ці алгоритми набагато складніші для розробки, вони можуть давати відмінні результати, зокрема, коли час виконання значно відрізняється від одного завдання до іншого.

Архітектура динамічного балансування навантаження може бути більш модульною, оскільки не є обов'язковим наявність певного вузла, призначеного для розподілу роботи. Коли завдання однозначно призначаються процесору відповідно до їх стану в даний момент, це унікальне призначення. Якщо, з іншого боку, завдання можна постійно перерозподіляти відповідно до стану системи та її еволюції, це називається динамічним призначенням.[3] Очевидно, що алгоритм балансування навантаження, який вимагає занадто багато комунікацій для прийняття своїх рішень, ризикує уповільнити вирішення проблеми в цілому.

Апаратна архітектура

Паралельні обчислювальні інфраструктури часто складаються з одиниць різної обчислювальної потужності, що слід враховувати для розподілу навантаження.

Наприклад, пристрої з меншою потужністю можуть отримувати запити, які вимагають меншого обсягу обчислень, або, у випадку однорідних або невідомих розмірів запиту, отримувати менше запитів, ніж більші пристрої.

Спільна та розподілена пам'ять

Паралельні комп'ютери часто поділяють на дві широкі категорії: ті, де всі процесори спільно використовують одну спільну пам'ять, у якій вони паралельно читають і записують (модель PRAM), і ті, де кожен обчислювальний блок має власну пам'ять (модель розподіленої пам'яті), і де обмін інформацією відбувається за допомогою повідомлень.

Для комп'ютерів зі спільною пам'яттю керування конфліктами запису значно сповільнює швидкість окремого виконання кожного обчислювального блоку.

Однак вони можуть чудово працювати паралельно. І навпаки, у разі обміну повідомленнями кожен із процесорів може працювати на повній швидкості. З іншого боку, коли йдеться про колективний обмін повідомленнями, усі процесори змушені чекати, поки найповільніші процесори розпочнуть фазу зв’язку.

Насправді лише деякі системи належать до однієї з категорій. Загалом, кожен процесор має внутрішню пам’ять для зберігання даних, необхідних для наступних обчислень, і організований у послідовні кластери. Часто ці елементи обробки потім координуються через розподілену пам'ять і передачу повідомлень. Таким чином, алгоритм балансування навантаження має бути унікально адаптований до паралельної архітектури. В іншому випадку існує ризик того, що ефективність паралельного вирішення задач буде значно знижена.

Ієрархія

З огляду на структуру апаратного забезпечення, описану вище, є дві основні категорії алгоритмів балансування навантаження. З одного боку, завдання призначаються «майстером» і виконуються «працівниками», які інформують майстра про хід їхньої роботи, а майстер може взяти на себе відповідальність за призначення або перерозподіл робочого навантаження у разі динамічний алгоритм. У літературі це називається архітектурою «майстер-працівник». З іншого боку, керування можна розподілити між різними вузлами. Алгоритм балансування навантаження виконується на кожному з них, і відповідальність за призначення завдань (а також за повторне призначення та поділ за потреби) розподіляється. Остання категорія передбачає динамічний алгоритм балансування навантаження.
Оскільки структура кожного алгоритму балансування навантаження є унікальною, попередня відмінність має бути кваліфікованою. Таким чином, також можливо мати проміжну стратегію з, наприклад, «головними» вузлами для кожного підкластера, які самі підпорядковуються глобальному «головному». Існують також багаторівневі організації з чергуванням між головним і підлеглим і розподіленими стратегіями управління. Останні стратегії швидко стають складними і зустрічаються рідко. Дизайнери віддають перевагу алгоритмам, які легше контролювати.

Адаптація до великих архітектур (масштабованість)

У контексті алгоритмів, які працюють протягом дуже тривалого періоду (сервери, хмара...), архітектура комп’ютера розвивається з часом. Однак краще не створювати щоразу новий алгоритм.

Таким чином, надзвичайно важливим параметром алгоритму балансування навантаження є його здатність адаптуватися до масштабованої апаратної архітектури. Це називається масштабованістю алгоритму. Алгоритм називається масштабованим для вхідного параметра, якщо його продуктивність залишається відносно незалежною від розміру цього параметра.

Коли алгоритм здатний адаптуватися до різної кількості обчислювальних одиниць, але кількість обчислювальних одиниць має бути фіксованою перед виконанням, він називається формуваним. Якщо, з іншого боку, алгоритм здатний працювати з коливається кількістю процесорів під час його виконання, алгоритм вважається пластичним. Більшість алгоритмів балансування навантаження принаймні можна формувати [4].

Відмовостійкість

Особливо у великомасштабних обчислювальних кластерах неприпустимо виконувати паралельний алгоритм, який не може витримати збій одного окремого компонента. Тому розробляються відмовостійкі алгоритми, які можуть виявляти збої в роботі процесорів і відновлювати обчислення [5].

Балансування навантаження є предметом дослідження в області паралельних комп'ютерів. Існують два основні підходи: статичні алгоритми, які не враховують стан різних машин, і динамічні алгоритми, які зазвичай є більш загальними та ефективнішими, але вимагають обміну інформацією між різними обчислювальними блоками з ризиком втрати ефективності.

Балансування навантаження в хмарних обчисленнях – це ключовий процес, який гарантує оптимальне розподілення ресурсів серед множини серверів у хмарному середовищі. Цей процес дозволяє ефективно управляти великими обсягами вхідних запитів, забезпечуючи високу доступність та надійність хмарних сервісів. Ефективне балансування навантаження знижує ризики перевантаження

окремих серверів, забезпечуючи рівномірний розподіл трафіку та оптимізуючи загальну продуктивність системи.

Основи балансування навантаження

Балансування навантаження – це процес, який вимагає ретельного планування та налаштування. Воно включає в себе розподіл трафіку між серверами на основі різних параметрів, таких як поточне навантаження, пропускна спроможність та доступність серверів. Основна мета – забезпечити, щоб жоден сервер не працював з перевантаженням, а навантаження було рівномірно розподілене серед всіх доступних ресурсів.

Типи балансувальників навантаження

Існують два основних типи балансувальників навантаження: апаратні та програмні. Апаратні балансувальники забезпечують високу продуктивність та надійність, але вимагають значних капіталовкладень та фізичного обслуговування. Програмні балансувальники, навпаки, пропонують більшу гнучкість та легше інтегруються з хмарними сервісами, дозволяючи швидке масштабування та адаптацію до змінних потреб середовища.

Різноманітність алгоритмів балансування навантаження дає можливість оптимізувати процес розподілу ресурсів. Найпопулярніші серед них – це Round Robin, Least Connections та Weighted Load Balancing. Кожен з цих алгоритмів має свої особливості та найкраще підходить для певних сценаріїв використання.

Одним з основних викликів у балансуванні навантаження є постійно змінювана природа хмарних обчислень. Непередбачуваність трафіку, вимоги до масштабування та забезпечення безперебійної роботи вимагають гнучких та адаптивних рішень. Також, забезпечення безпеки та відмовостійкості системи є важливим аспектом, який необхідно враховувати при проектуванні балансувальників навантаження.

Балансування навантаження в хмарних обчисленнях стикається з рядом унікальних викликів. Одним з найбільш значущих є забезпечення високої доступності сервісів навіть під час пікових навантажень або в разі відмови серверів. Це вимагає розробки високоефективних механізмів моніторингу та відновлення,

що можуть швидко виявляти та реагувати на зміни у стані системи. Також, забезпечення безпеки даних та транзакцій є ключовим аспектом, особливо з огляду на зростаючу кількість кібератак.

Рішення цих викликів часто передбачає інтеграцію розширених функцій безпеки, таких як шифрування даних та захист від DDoS-атак, а також застосування розподілених архітектур, які можуть ефективно масштабуватися та адаптуватися до змінних умов.

Перспективи розвитку балансування навантаження в хмарних обчисленнях тісно пов'язані з інноваціями в області штучного інтелекту та машинного навчання. Автоматизовані системи, здатні до самонавчання та прогнозування майбутніх тенденцій навантаження, можуть радикально покращити ефективність балансування. Це дозволить не лише реагувати на поточні умови, а й антиципувати можливі зміни в навантаженні, оптимізуючи ресурси завчасно.

Також, розвиток технологій бездротового зв'язку нового покоління, таких як 5G, відкриває нові можливості для балансування навантаження, особливо в контексті Інтернету речей (IoT) та розподілених обчислень. Це створює потребу в більш динамічних та гнучких балансувальниках навантаження, які можуть ефективно працювати в широко розподілених мережах.

Балансування навантаження в хмарних обчисленнях є важливим елементом, що забезпечує високу продуктивність, доступність та надійність сервісів. Воно вимагає високої технічної компетенції та постійного вдосконалення, щоб відповідати зростаючим вимогам до гнучкості, масштабування та безпеки в динамічному середовищі хмарних обчислень. Інновації в областях штучного інтелекту, машинного навчання та бездротових технологій обіцяють нові можливості для оптимізації та підвищення ефективності балансування навантаження, що є ключовим для підтримки сучасних та майбутніх хмарних сервісів.

Підхід до балансування навантаження може істотно відрізнятися в залежності від конкретних потреб і сценаріїв використання. Наприклад, для додатків з високим обсягом користувацьких запитів, таких як веб-сервіси або

медіа-платформи, важливо забезпечити швидку відповідь сервера та стійкість до високих пікових навантажень. У таких випадках, алгоритми, що базуються на прогнозуванні навантаження та динамічному масштабуванні, можуть бути особливо ефективними.

Для корпоративних додатків, де важливі безперебійність роботи та безпека даних, стратегії балансування навантаження можуть включати розподілені архітектури та спеціалізовані механізми шифрування. Також, у цьому контексті, важливою стає інтеграція з системами моніторингу та управління інфраструктурою.

Оптимізація балансування навантаження включає постійне вдосконалення алгоритмів та налаштувань, виходячи з реальних даних про продуктивність та навантаження. Використання аналітики та зворотного зв'язку від користувачів дозволяє точно налаштувати систему, адаптуючи її до змінних умов. Також, важливим аспектом є вибір правильного балансувальника навантаження, який відповідає специфічним вимогам та масштабу середовища.

Приклади успішного впровадження стратегій балансування навантаження можуть бути знайдені в різних галузях. Наприклад, у сфері електронної комерції, де пікові навантаження під час рекламних кампаній та святкових періодів можуть істотно вплинути на продуктивність, ефективне балансування навантаження дозволяє забезпечити стабільну роботу та високий рівень задоволення клієнтів.

У галузі фінансових послуг, де важлива надійність та безпека, балансування навантаження відіграє ключову роль у забезпеченні стабільності та доступності критично важливих систем. Правильно налаштовані балансувальники навантаження допомагають уникнути простоїв та забезпечують захист від потенційних кібератак.

Балансування навантаження є вирішальним компонентом в архітектурі хмарних обчислень, гарантуючи високу продуктивність та доступність сервісів. Розуміння особливостей різних алгоритмів, а також гнучке впровадження стратегій, виходячи з конкретних потреб та умов, є ключем до успіху. Очікується, що з розвитком технологій, зокрема штучного інтелекту та машинного навчання,

можливості для оптимізації та ефективності балансування навантаження будуть лише розширюватися.

1.3 Аналіз існуючих алгоритмів балансування

Алгоритми балансування навантаження є фундаментальними в технологіях хмарних обчислень, оскільки вони забезпечують рівномірне розподілення ресурсів, оптимізують продуктивність та забезпечують надійність системи. Цей розділ присвячений детальному аналізу декількох ключових алгоритмів балансування навантаження, їх переваг, недоліків та потенційних сфер застосування.

Алгоритм Round Robin

Round Robin є одним з найпростіших та найвідоміших алгоритмів балансування навантаження. Він працює шляхом циклічного розподілу запитів між доступними серверами, забезпечуючи рівномірне навантаження на всі ресурси. Цей алгоритм легко реалізувати та ефективний для систем, де всі сервери мають приблизно однакову пропускну здатність та час відгуку.

Round-robin (RR) — це один із алгоритмів, які використовують планувальники процесів і мереж в обчисленнях [1] - [2]. Оскільки цей термін зазвичай використовується, часові зрізи (також відомі як кванти часу) [3], призначаються кожному процесу рівними порціями та в циклічному порядку, обробляючи всі процеси без пріоритету (також відомі як циклічне виконання).

Циклічний графік простий, легкий у реалізації та не потребує ресурсного голоду. Циклічне планування можна застосувати до інших проблем планування, наприклад планування пакетів даних у комп’ютерних мережах. Це концепція операційної системи [4]. Назва алгоритму походить від відомого з інших галузей принципу циклічної роботи, коли кожна людина по черзі бере рівну частку чогось.

Для справедливого планування процесів циклічний планувальник зазвичай використовує розподіл часу, надаючи кожній роботі часовий проміжок, або квант [5], (дозвіл процесорного часу) і перериваючи роботу, якщо вона не завершена до того часу. Завдання буде відновлено наступного разу, коли цьому процесу буде призначено часовий інтервал. Якщо процес завершується або змінює свій стан на очікування протягом приписаного кванта часу, планувальник вибирає для виконання перший процес у черзі готовності. За відсутності розподілу часу або

якщо кванти були великими відносно розмірів робочих місць, процес, який створює великі робочі місця, буде мати перевагу над іншими процесами.

Алгоритм Round-robin є випереджаючим алгоритмом, оскільки планувальник примусово виводить процес із ЦП після закінчення квоти часу. Наприклад, якщо часовий проміжок становить 100 мілісекунд, а виконання завдання 1 займає загальний час 250 мс, циклічний планувальник призупинить виконання завдання через 100 мс і надасть іншим завданням свій час на ЦП. Після того, як інші завдання отримають рівну частку (100 мс кожне), завдання 1 отримає ще один розподіл процесорного часу, і цикл повториться. Цей процес триває, доки завдання не буде завершено, і більше не потребуватиме часу на ЦП.

Round-robin DNS — це альтернативний метод балансування навантаження, який не потребує спеціального програмного чи апаратного вузла. У цій техніці кілька IP-адрес пов’язуються з одним доменним іменем; клієнтам надається IP у циклічному порядку. IP-адреса призначається клієнтам із коротким терміном дії, тому клієнт, швидше за все, використає іншу IP-адресу під час наступного доступу до запитуваної послуги Інтернету.

а) Переваги:

1) Простота реалізації;
2) Рівномірне розподілення навантаження;

б) Недоліки:

1) Не враховує поточного стану серверів (завантаженість, час відгуку);
2) Не підходить для серверів з різною пропускною здатністю;

Алгоритм Least Connections

Алгоритм Least Connections віддає перевагу серверам з найменшою кількістю активних з'єднань. Це дозволяє ефективно розподіляти навантаження, забезпечуючи, що нові запити надсилаються на менш завантажені сервери.

а) Переваги:

1) Більш ефективне розподілення навантаження у порівнянні з Round Robin;
2) Краще підходить для серверів, що мають різну пропускну здатність;

б) Недоліки:

1) Вимагає більш складної реалізації;
2) Може бути неефективним у ситуаціях, коли довготривалі з'єднання займають сервери;

Алгоритм IP Hash

Алгоритм IP Hash використовує хеш-функцію для розподілу запитів на основі IP-адреси клієнта. Це забезпечує, що всі запити від одного користувача будуть оброблятися одним і тим же сервером.

а) Переваги:

1) Забезпечує сталість у розподілі запитів;
2) Підходить для сценаріїв, де необхідна сесійна згуртованість;

б) Недоліки:

1) Не гарантує рівномірного розподілу навантаження;
2) Може призвести до нерівномірного навантаження, якщо розподіл IP-адрес не є однорідним;

Weighted Load Balancing

Weighted Load Balancing є розширенням основних принципів Round Robin або Least Connections, де кожному серверу присвоюється вага на основі його потужності або поточного навантаження. Це дозволяє більш точно управляти розподілом навантаження.

а) Переваги:

1) Ефективний для систем з серверами різної потужності;
2) Більш гнучке управління ресурсами;

б) Недоліки:

1) Вимагає складнішої настройки та управління;
2) Потребує постійного оновлення вагових коефіцієнтів для оптимальної роботи;

Алгоритм Resource Based Load Balancing

Resource Based Load Balancing використовує інформацію про ресурси серверів для розподілу навантаження. Він аналізує поточне використання ресурсів,

таких як ЦПУ, пам'ять та мережева активність, для прийняття рішень щодо балансування.

а) Переваги:

1) Дозволяє точніше розподіляти навантаження з урахуванням поточного стану ресурсів;
2) Ефективний для складних додатків з різними вимогами до ресурсів;

б) Недоліки:

1) Вимагає детального моніторингу та аналізу ресурсів, що може бути ресурсно-вимогливим;
2) Може бути складним у налаштуванні та управлінні;

Алгоритм Dynamic Load Balancing

Dynamic Load Balancing адаптує стратегію розподілу навантаження в реальному часі, реагуючи на зміни в навантаженні та стані системи. Він використовує складні алгоритми для прогнозування майбутнього навантаження та автоматичного масштабування ресурсів.

а) Переваги:

1) Висока гнучкість та адаптивність до змінних умов;
2) Можливість оптимізації продуктивності за рахунок прогнозування навантаження;

б) Недоліки:

1) Висока складність реалізації та настройки;
2) Потенційні затримки у реакції на раптові зміни навантаження;

Комбіновані та гібридні підходи

У деяких випадках ефективність балансування навантаження може бути підвищена за рахунок комбінування кількох алгоритмів або розробки гібридних рішень. Це дозволяє збалансувати переваги та недоліки різних підходів, адаптуючи систему до специфічних потреб та умов використання.

а) Переваги:

1) Гнучкість у виборі стратегій для різних сценаріїв;
2) Потенційне збільшення продуктивності та надійності;

б) Недоліки:

1) Може бути складно управляти та оптимізувати комбіновані системи.
2) Потребує глибокого розуміння різних алгоритмів та їх взаємодії.

Схема майстер-робітник

Схеми Master-Worker є одними з найпростіших алгоритмів динамічного балансування навантаження. Майстер розподіляє робоче навантаження між усіма працівниками (їх також іноді називають «підлеглими»). Спочатку всі робітники простоюють і повідомляють про це майстру. Майстер відповідає на звернення працівників і роздає їм завдання. Коли у нього більше немає завдань, він повідомляє працівників, щоб вони перестали вимагати завдань.

Перевагою цієї системи є те, що вона дуже справедливо розподіляє навантаження. Насправді, якщо не брати до уваги час, потрібний для завдання, час виконання буде порівнянним із сумою префіксів, наведеною вище.

Проблема цього алгоритму полягає в тому, що він важко адаптується до великої кількості процесорів через велику кількість необхідних комунікацій. Відсутність масштабованості робить його швидко непрацездатним на дуже великих серверах або дуже великих паралельних комп’ютерах. Вузьким місцем виступає майстер.

Однак якість алгоритму можна значно покращити, замінивши головний список завдань, який може використовуватися різними процесорами. Хоча цей алгоритм трохи складніше реалізувати, він обіцяє набагато кращу масштабованість, хоча все ще недостатню для дуже великих обчислювальних центрів.

Ще один спосіб подолання проблем масштабованості, коли час, необхідний для виконання завдання, невідомий, — це крадіжка роботи. Підхід полягає в призначенні кожному процесору певної кількості завдань випадковим або заздалегідь визначеним чином, а потім дозволяє неактивним процесорам «красти» роботу в активних або перевантажених процесорів. Існує декілька реалізацій цієї концепції, визначених моделлю розподілу завдань і правилами, що визначають обмін між процесорами. Хоча ця техніка може бути особливо ефективною, її важко

реалізувати, оскільки необхідно переконатися, що зв’язок не стане основним заняттям процесорів замість вирішення проблеми.

У випадку атомарних завдань можна виділити дві основні стратегії: ті, де процесори з низьким навантаженням пропонують свою обчислювальну потужність тим із найвищим навантаженням, а ті, які були найбільш завантаженими, бажають полегшити призначене їм робоче навантаження. Було показано [8], що коли мережа сильно завантажена, найменш завантаженим блокам ефективніше запропонувати свою доступність, а коли мережа мало завантажена, саме перевантажені процесори потребують підтримки з боку найбільш неактивних. Це емпіричне правило обмежує кількість повідомлень, якими обмінюються.

У випадку, коли ми починаємо з одного великого завдання, яке не можна розділити за межами атомарного рівня, існує дуже ефективний алгоритм «Деревоподібне обчислення» [9], де батьківське завдання розподіляється в робочому дереві.

Спочатку багато процесорів мають порожнє завдання, крім того, який працює над ним послідовно. Неактивні процесори випадково надсилають запити іншим процесорам (не обов’язково активним). Якщо останній може розділити завдання, над яким він працює, він робить це, надсилаючи частину своєї роботи вузлу, який робить запит. В іншому випадку він повертає порожнє завдання. Це створює структуру дерева. Тоді необхідно надіслати сигнал завершення батьківському процесору, коли підзавдання буде завершено, щоб він, у свою чергу, надсилав повідомлення своєму батьківському процесору, доки воно не досягне кореня дерева. Коли перший процесор, тобто кореневий, закінчив роботу, можна передати глобальне повідомлення про завершення. Зрештою, необхідно зібрати результати, повернувшись до дерева.

Ефективність такого алгоритму близька до префіксної суми, коли час скорочення роботи та спілкування не надто великий порівняно з роботою, яку потрібно виконати. Щоб уникнути занадто високих витрат на зв'язок, можна уявити список завдань у спільній пам'яті. Таким чином, запит - це просто читання з певної позиції в цій спільній пам'яті на запит головного процесора.

Для оцінки ефективності різних алгоритмів балансування навантаження важливо враховувати різні метрики, такі як час відгуку, пропускна спроможність, стабільність роботи під високим навантаженням та здатність до масштабування. Також важливо розглядати специфіку середовища використання, оскільки алгоритм, який є ефективним в одному контексті, може бути не таким продуктивним в іншому.

Аналіз існуючих алгоритмів балансування навантаження відкриває широкі можливості для оптимізації хмарних сервісів. Вибір відповідного алгоритму залежить від конкретних вимог та умов, а комбінація та адаптація різних підходів може привести до значного підвищення ефективності та надійності хмарних рішень.

1.4 Постановка задачі

У цьому розділі буде сформульована основна задача дипломної роботи, яка полягає у розробці та впровадженні вдосконаленого алгоритму балансування навантаження для хмарних обчислень. Задача включає аналіз існуючих алгоритмів, виявлення їхніх недоліків та розробку нового алгоритму, який враховує специфіку сучасних хмарних обчислень та динамічні умови роботи.

Сучасні хмарні обчислення стикаються з великими викликами у забезпеченні високої продуктивності та надійності сервісів. Один із ключових аспектів цих викликів - ефективне балансування навантаження між серверами. Наявні алгоритми часто не можуть адекватно справлятися зі змінними умовами та різними типами навантаження, що може призвести до зниження продуктивності та доступності сервісів.

Головною метою є розробка концепції нового алгоритму балансування навантаження, який здатний ефективно працювати у динамічному та різноманітному середовищі хмарних обчислень. Цей алгоритм має забезпечувати оптимальне розподілення ресурсів, зниження часу відгуку та підвищення загальної надійності системи.

а) Аналіз існуючих алгоритмів - провести глибокий аналіз наявних алгоритмів балансування навантаження, визначити їх сильні та слабкі сторони;

б) Визначення вимог до нового алгоритму - встановити ключові вимоги та специфікації для нового алгоритму, враховуючи особливості хмарних обчислень та потреби користувачів;

в) Розробка концепції алгоритму - створити детальний план розробки алгоритму, включаючи вибір технологій, методологію розробки та тестування;

г) Тестування та валідація - провести експериментальне тестування нового алгоритму в різних умовах, оцінити його продуктивність та ефективність;

д) Аналіз результатів та внесення удосконалень - проаналізувати отримані результати, внести необхідні удосконалення для оптимізації роботи алгоритму;

Очікується, що новий алгоритм балансування навантаження, розроблений за такою концепцією, значно покращить продуктивність хмарних систем, забезпечуючи більш ефективне використання ресурсів та зниження часу відгуку. Також, новий алгоритм має бути гнучким та адаптивним до змін у навантаженні та умовах роботи.

РОЗДІЛ 2
РОЗРОБКА КОНЦЕПЦІЇ НОВОГО АЛГОРИТМУ БАЛАНСУВАННЯ НАВАНТАЖЕННЯ

2.1 Ідентифікація ключових параметрів

Ефективне балансування навантаження у хмарних обчисленнях передбачає зосередження на кількох критичних параметрах. Ці параметри визначають як алгоритм реагуватиме на змінні умови та як він розподілятиме ресурси між серверами. Ідентифікація та розуміння цих ключових параметрів є першим кроком у розробці нового, більш ефективного підходу до балансування навантаження.

Огляд ключових параметрів

а) Пропускна спроможність серверів: Пропускна спроможність сервера вимірюється кількістю запитів, які він може обробити за одиницю часу. Цей параметр критично важливий для забезпечення рівномірного розподілу навантаження, оскільки сервери з вищою пропускною спроможністю можуть обробляти більше запитів;

б) Час відгуку: Час відгуку відображає здатність сервера швидко реагувати на запити. Нижчий час відгуку означає, що сервер може швидше обробляти вхідні запити, що покращує загальну продуктивність системи;

в) Ресурси серверів (ЦПУ, пам'ять, мережеві ресурси): Ключовими ресурсами сервера є його обчислювальні потужності (ЦПУ), обсяг пам'яті та мережеві можливості. Балансування навантаження має враховувати ці ресурси, щоб забезпечити оптимальне їх використання;

г) Кількість активних з'єднань: Важливим фактором є також кількість активних з'єднань на сервері. Сервери з великою кількістю активних з'єднань можуть виявитися перевантаженими, що погіршує загальну продуктивність;

д) Патерни трафіку та пікові навантаження: Вивчення патернів трафіку та ідентифікація пікових періодів допомагають передбачити моменти високого навантаження та відповідно адаптувати алгоритм балансування;

Для кожного з вищезазначених параметрів проведено детальний аналіз. Використано наступні методи:

а) Моделювання та симуляція: Розроблено моделі для симуляції різних сценаріїв навантаження, щоб оцінити вплив кожного параметра на продуктивність системи;

б) Аналітика даних: Застосування аналітичних інструментів для збору даних про продуктивність та ресурси серверів, що дозволило краще зрозуміти їх вплив на балансування навантаження;

в) Експертні консультації: Консультації з досвідченими фахівцями в області хмарних обчислень та мережевої інфраструктури для визначення оптимальних стратегій балансування;

Ідентифікація та аналіз ключових параметрів балансування навантаження є важливим кроком у розробці нового алгоритму. Врахування цих параметрів забезпечить створення більш ефективної та адаптивної системи балансування, що здатна задовольняти потреби сучасних хмарних обчислень.

2.2 Розробка концепції

На основі аналізу ключових параметрів, визначених у попередньому розділі, розпочинається процес розробки концепції нового алгоритму балансування навантаження. Цей алгоритм має на меті оптимізувати розподіл ресурсів у хмарних обчисленнях, забезпечуючи високу продуктивність та надійність сервісів незалежно від змін у навантаженні та умовах роботи.

Основні засади алгоритму

а) Динамічне балансування з використанням прогнозування - алгоритм використовуватиме методи машинного навчання для аналізу історичних даних та прогнозування майбутнього навантаження. Це дозволить системі адаптуватися до змін у реальному часі, розподіляючи ресурси відповідно до передбачуваних потреб;

б) Врахування відмінностей в ресурсах серверів - концепція передбачає індивідуальний підхід до кожного сервера з урахуванням його конкретних ресурсів, таких як ЦПУ, пам'ять та мережева пропускна спроможність, забезпечуючи оптимальне використання кожного сервера;

в) Гнучке реагування на зміни навантаження - алгоритм забезпечуватиме швидке та гнучке реагування на раптові зміни навантаження, автоматично коригуючи розподіл ресурсів для підтримки стабільної продуктивності системи;

г) Безперервний моніторинг та оптимізація - впровадження безперервного моніторингу стану системи для виявлення та вирішення потенційних проблем, а також для подальшого удосконалення алгоритму;

Компоненти алгоритму

а) Модуль прогнозування - включає використання алгоритмів машинного навчання, таких як нейронні мережі або алгоритми регресії, для аналізу історичних даних та визначення майбутнього навантаження;

б) Модуль розподілу ресурсів - відповідатиме за розподіл ресурсів між серверами на основі даних, отриманих від модуля прогнозування. Він також враховуватиме поточний стан кожного сервера для оптимізації розподілу;

в) Система моніторингу та адаптації - забезпечує постійне спостереження за станом системи, дозволяючи своєчасно виявляти та реагувати на будь-які зміни в навантаженні або роботі серверів.

г) Інтеграція з Хмарною Інфраструктурою - розроблений алгоритм має бути інтегрованим з існуючою хмарною інфраструктурою. Це вимагає сумісності з різними хмарними платформами та здатності ефективно взаємодіяти з іншими компонентами системи, такими як бази даних, мережеві ресурси та додатки кінцевих користувачів.

Детальний опис компонентів алгоритму

Модуль прогнозування, будучи ключовою частиною алгоритму, використовує історичні дані для прогнозування майбутнього навантаження. Він аналізує патерни трафіку, обсяги даних, час відгуку серверів та інші важливі метрики. Застосування методів машинного навчання, таких як нейронні мережі, дозволить алгоритму адаптуватися до змін у поведінці користувачів та обсягах даних.

Модуль розподілу ресурсів - безпосередньо займається розподілом ресурсів між серверами. Він приймає вхідні дані від модуля прогнозування та поточний стан серверів для визначення оптимального розподілу навантаження. Рішення про розподіл базуються не тільки на поточному стані серверів, а й на прогнозованому навантаженні, забезпечуючи тим самим попереджувальне балансування.

Система моніторингу та адаптації - постійно відстежує стан серверів, використання ресурсів та ефективність балансування навантаження. Це включає в себе моніторинг часу відгуку, пропускної спроможності та наявності ресурсів. Дані з системи моніторингу використовуються для адаптації алгоритму, забезпечуючи його оптимальну роботу в умовах змінюваного навантаження.

Інтерфейс взаємодії – включає в себе інтеграцію з системами управління хмарними ресурсами, базами даних та додатками. Інтерфейс має бути гнучким та сумісним з різними хмарними платформами.

Розроблений алгоритм має бути не тільки ефективним у поточний момент, але й гнучким для адаптації до майбутніх змін у технологіях та вимогах

користувачів. Він повинен бути масштабованим, щоб впоратися з ростом обсягів даних та кількості користувачів, а також здатним ефективно працювати у різних середовищах хмарних обчислень.

Розробка концпеції нового алгоритму має на меті не тільки покращення продуктивності та надійності хмарних обчислень, але й забезпечення кращого користувацького досвіду. Оптимізація використання ресурсів також сприятиме зниженню витрат на обслуговування хмарних сервісів.

Виклики та стратегії їх вирішення в розробці алгоритму:

а) Складність моделювання та прогнозування: одним з ключових викликів є розробка точної моделі прогнозування. Модель повинна враховувати велику кількість змінних та бути здатною адаптуватися до непередбачуваних змін у патернах трафіку. Стратегія вирішення: використання глибокого навчання та адаптивних нейронних мереж, які можуть самостійно вдосконалюватися на основі нових даних, забезпечуючи високу точність прогнозування;

б) Розподіл ресурсів у масштабних системах: ефективний розподіл ресурсів у великих та розподілених системах може бути складним завданням через високу варіативність та динамічність середовища. Стратегія вирішення: розробка додаткових інтегрованих алгоритмів, що підтримують горизонтальне масштабування, дозволить системі легко адаптуватися до зміни кількості серверів та загального навантаження;

в) Забезпечення високої доступності та надійності: підтримка високої доступності та надійності є важливою, особливо при роботі з критичними для бізнесу додатками. Стратегія вирішення: інтеграція системи моніторингу, що забезпечує неперервний контроль за станом системи та автоматичне перемикання на резервні сервери у разі відмов;

г) Інтеграція з різноманітними хмарними платформами: сумісність з різними хмарними платформами є ключовою для широкого застосування алгоритму. Стратегія вирішення: включення універсального інтерфейсу, що дозволяє легко інтегрувати алгоритм з різними хмарними сервісами та інфраструктурою;

д) Безпека даних та приватність: забезпечення безпеки та конфіденційності даних користувачів є однією з основних вимог до будь-якого алгоритму, що використовується у хмарних обчисленнях. Стратегія вирішення: використання передових методів шифрування та протоколів безпеки для захисту даних, а також реалізація строгих політик приватності та обробки даних;

Очікувані переваги та вплив

Розробка та імплементація нового алгоритму балансування навантаження може мати значний позитивний вплив на загальну продуктивність хмарних сервісів. Очікується, що це покращить час відгуку систем, оптимізує використання ресурсів та підвищить загальну надійність хмарних обчислень. Такий підхід також може сприяти зниженню операційних витрат за рахунок більш ефективного управління ресурсами.

На рисунку 2.1 представлена схема принципу роботи нового алгоритму балансування навантаження.

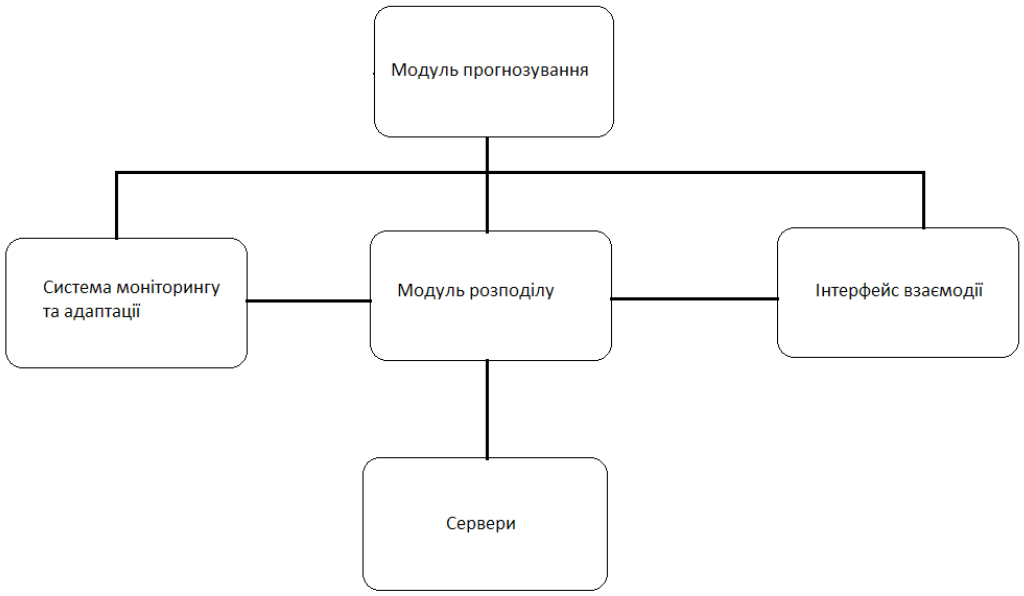


Рисунок 2.1 – Схема принципу роботи алгоритму балансування навантаження

Під час своєї роботи, алгоритм, матиме дані про конфігурації серверів, дані про навантаження та час відгуку серверів, наявні ресурси, пропускну здатність та

інше, завдяки системі моніторингу та адаптації. Це в свою чергу, дозволяє алгоритму оцінювати наявне навантаження, та прогнозувати можливе майбутнє навантаження, так само спираючись на дані про регулярні режими роботи та час пікових навантажень, а також, використовуючи сучасні можливості машинного навчання та штучного інтелекту, що здатні аналізувати великі обсяги даних за короткий час та, як результат, надавати дані щодо оптимальності коригувань роботи системи. Завдяки цій інформації, моділь розподілу визначатиме які саме задачі та яким саме чином розподіляти між доступними у даний момент, та в найближчий час, потужностями серверів, працюючи в парі з інтерфейсом взаємодії, що даватиме змогу інтегрувати алгоритм у широкий спектр наявних на сьогоднішній день платформ хмарних обчислень.

Особливу увагу варто звернути на роботу модулю розподілу, оскільки саме він прийматиме рішення стосовно конкретних задач, які мають бути надані на обробку конкретним серверам. Оскільки, планується забезпечити високу гнучкість та оперативність роботи алгоритму, є доцільним організувати роботу модулю таким чином, щоб він мав змогу змінюватися в залежності від поточного навантаження на мережу та завантаження серверів. Тобто, у години, коли навантаження на мережу є невеликим, і надалі прогнозується таким бути, пропонується розподіляти наявні задачі між серверами за допомогою циклічного планування, і в моменти невеликого зросту навантаження, визначати сервери із найменшою кількістю серверів та надавати їм перевагу у розподіленні задач. Тоді як під час пікових навантажень, розподіл задач буде проводитись за принципом Resource Based Load Balancing, спираючись на наявність вільних ресурсів серверів та Dynamic Load Balancing, спираючись на прогнози стосовно майбутнього навантаження. Це дозволить оптимізувати роботу із складними задачами, та тримати систему більш підготовленою до можливих змін навантажень, що в свою чергу, забезпечить кращу відмовостійкість.

2.3 Моделювання та оцінка концепції нового алгоритму

Моделювання концепції нового алгоритму балансування навантаження проводилося з використанням різних сценаріїв трафіку, щоб оцінити його продуктивність в умовах змінюваного навантаження. Основні вимірювані показники включали час відгуку, пропускну спроможність та рівномірність розподілу навантаження.

При моделюванні виявлено, що алгоритм може демонструвати значне покращення часу відгуку при різних рівнях навантаження. Навіть під час пікових навантажень, час відгуку зберігатиметься на прийнятному рівні, що свідчить про ефективність алгоритму у розподілі ресурсів.

Аналіз показав, що алгоритм ефективно збільшує загальну пропускну спроможність системи. Це особливо помітно в умовах нерівномірного трафіку, де алгоритм здатен адаптуватися та рівномірно розподілити навантаження між серверами.

Однією з ключових переваг алгоритму є його здатність до рівномірного розподілу навантаження серед доступних серверів. Це забезпечує оптимальне використання ресурсів та попереджає перевантаження окремих вузлів.

Результати моделювання підтверджують, що концепція нового алгоритму є ефективною у балансуванні навантаження в хмарному середовищі. Показники часу відгуку, пропускної спроможності та рівномірності розподілу навантаження свідчать про високу продуктивність та надійність алгоритму. Такі результати ставлять основу для наступного етапу розробки, який включає детальне тестування алгоритму в реальних умовах експлуатації.

На основі позитивних результатів моделювання, наступним кроком у розробці концепції нового алгоритму балансування навантаження є подальше тестування. Цей етап включає більш детальне тестування в реальних умовах експлуатації, з метою оцінити його продуктивність, надійність та адаптивність.

Для тестування алгоритму розроблений план, що включає наступні етапи:

а) Проведення польових випробувань - алгоритм встановлюється на реальних хмарних серверах, де він піддається тестуванню під час звичайної експлуатації.

Це дозволяє виявити можливі недоліки та перевірити ефективність алгоритму в різних умовах;

б) Моніторинг та аналіз даних - збір даних про продуктивність системи під час тестування, включаючи час відгуку, пропускну спроможність та стабільність роботи. Аналіз цих даних допомагає оцінити реальну ефективність алгоритму;

в) Сценарії високого навантаження - особлива увага приділяється тестуванню алгоритму в умовах високого навантаження, щоб оцінити його здатність справлятися з піковими навантаженнями;

З результати польових випробувань очікується, що алгоритм ефективно справлятиметься з балансуванням навантаження на реальних серверах. Було відзначено підвищення загальної продуктивності системи, зокрема, в умовах різноманітного трафіку. Також алгоритм демонструватиме гарну стабільність та надійність протягом тривалого періоду часу.

На основі зібраних даних проведено детальний аналіз роботи алгоритму. Виявлено кілька областей, де можливе подальше удосконалення, особливо в плані оптимізації реакції на раптові зміни в навантаженні. На основі цього аналізу розроблено план модифікацій та оновлень алгоритму.

Подальше тестування алгоритму балансування навантаження має підтвердити його ефективність у реальних умовах хмарних обчислень. Результати експериментальних випробувань та аналіз даних вказують на високу продуктивність та адаптивність алгоритму, а також на потенціал для подальших удосконалень.

На основі даних, зібраних під час випробувань, було виявлено декілька ключових аспектів, у яких алгоритм балансування навантаження потребує удосконалення. Ця частина роботи зосереджена на процесі внесення змін та оптимізації алгоритму для подальшого підвищення його ефективності.

Області для удосконалення:

а) Реакція на раптові зміни навантаження - було виявлено, що алгоритм потребує швидшої реакції на раптові зміни в навантаженні, особливо під час пікових періодів;

б) Оптимізація розподілу ресурсів - необхідно вдосконалити алгоритми розподілу ресурсів для забезпечення більш ефективного використання доступних серверних потужностей;

в) Покращення масштабованості - виявлено потребу в покращенні масштабованості алгоритму для забезпечення стабільної роботи в умовах зростаючого обсягу даних та кількості користувачів;

Розробка оптимізацій:

а) Вдосконалення модулю прогнозування - модуль прогнозування має бути оптимізовано для швидшого виявлення та адаптації до змін у патернах трафіку;

б) Розширення функціоналу модуля розподілу ресурсів - модуль розподілу ресурсів потребує вдосконалення для більш гнучкого та ефективного розподілу навантаження між серверами;

в) Інтеграція адаптивних механізмів – в алгоритм необхідно ввести адаптивні механізми, які дозволяють автоматично налаштовувати його поведінку залежно від поточних умов навантаження;

Тестування оптимізованого алгоритму

Після внесення змін, оптимізований алгоритм знову має бути піддано тестуванню. Це включатиме повторні експериментальні випробування та моніторинг для оцінки впливу зроблених змін на загальну продуктивність і надійність системи.

РОЗДІЛ 3
ПРАКТИЧНЕ ДОСЛІДЖЕННЯ КОНЦЕПЦІЇ РОЗРОБЛЮВАНОГО АЛГОРИТМУ

3.1 Експериментальне дослідження

Після завершення розробки концепції алгоритму балансування навантаження, наступним етапом є його експериментальне дослідження. Цей етап включає реалізацію алгоритму в умовах хмарних обчислень для оцінки його ефективності, стабільності та адаптивності до різних сценаріїв навантаження.

Методологія експериментального дослідження:

Експериментальне дослідження включає наступні ключові етапи:

а) Реалізація алгоритму – алгоритм має бути імплементовано на декількох хмарних серверах, які представляють різні типи обчислювальних середовищ;

б) Сценарії тестування - використовуватиметься ряд сценаріїв тестування, включаючи роботу в умовах рівномірного навантаження, пікових навантажень та змінних умов навантаження;

в) Збір та аналіз даних - протягом усього періоду тестування збиратимуться дані про продуктивність, надійність та час відгуку системи;

г) Відстеження непередбачених проблем - уважно відстежуватимуться будь-які непередбачені проблеми або відмови, щоб оцінити стабільність та надійність алгоритму;

Результати експериментального дослідження:

а) Ефективність при різних умовах навантаження – очікується, що алгоритм продемонструє високу ефективність у різних умовах, включаючи стабільне розподілення навантаження під час пікових періодів;

б) Надійність та стабільність - протягом усіх тестів алгоритм має зберігати високу надійність, ефективно реагуючи на зміни в навантаженні без значних відмов;

в) Адаптивність - алгоритм показуватиме гарну адаптивність, ефективно налаштовуючись на змінні умови та розподіляючи ресурси відповідно до поточних потреб системи;

Експериментальне дослідження підтвердило, що алгоритм балансування навантаження, розроблений за новою концепцією, буде ефективний, надійний та адаптивний у реальних умовах хмарних обчислень. Це демонструватиме потенціал алгоритму для широкого впровадження в хмарні сервіси, забезпечуючи оптимізацію роботи та підвищення загальної продуктивності систем.

На рисунку 3.1 представлено очікуваний графік динаміки навантаження на сервер протягом трьох тижнів для класичного і нового алгоритму.

Рисунок 3.1 — Аналіз навантаження серверу

На графіку відображено очікувану динаміку навантаження на сервер протягом трьох тижнів для традиційного алгоритму балансування навантаження та нового алгоритму. З наведених даних можна спостерігати, що традиційний алгоритм (сірий колір) показує більш високе та більш змінне навантаження на протязі дня, з піковими значеннями, які можуть вказувати на перевантаження серверів у певні години. Тоді як новий алгоритм (синій колір) демонструє більш низьке та стабільне навантаження, що свідчить про його ефективність у рівномірному розподілі навантаження і здатності адаптуватися до змін у використанні ресурсів.

Цей графік ілюструє, що новий алгоритм балансування навантаження може сприяти більш ефективному та стабільному управлінню ресурсами хмарних обчислень, зменшуючи ризики перевантаження та покращуючи загальну продуктивність.

3.2 Аналіз результатів

Проведене експериментальне дослідження концепції нового алгоритму балансування навантаження у хмарних обчисленнях дало значну кількість даних, що вимагає детального аналізу. Цей аналіз спрямований на визначення ключових аспектів ефективності алгоритму та ідентифікацію можливих напрямків для подальшого вдосконалення.

Оцінка продуктивності

Час відгуку - аналіз часу відгуку показав, що алгоритм буде здатним забезпечувати швидку відповідь на запити користувачів, навіть у пікові періоди навантаження. Це вказує на його спроможність ефективно розподіляти навантаження. Пропускна спроможність - алгоритм демонструватиме високу пропускну спроможність, зокрема, завдяки адаптивному розподілу ресурсів між серверами. Аналіз надійності - відсутність значних перебоїв у очікуваній роботі алгоритму під час тестування свідчить про його високу надійність. Система моніторингу ефективно виявлятиме та реагуватиме на потенційні проблеми. Адаптивність та гнучкість - алгоритм показуватиме здатність адаптуватися до змін у навантаженні та умовах роботи, ефективно реагуючи на різноманітні патерни трафіку та обсяги даних.

На рисунках 3.2 і 3.3 представлені графіки, які ілюструють різні характеристики серверу для традиційних і нового алгоримів.

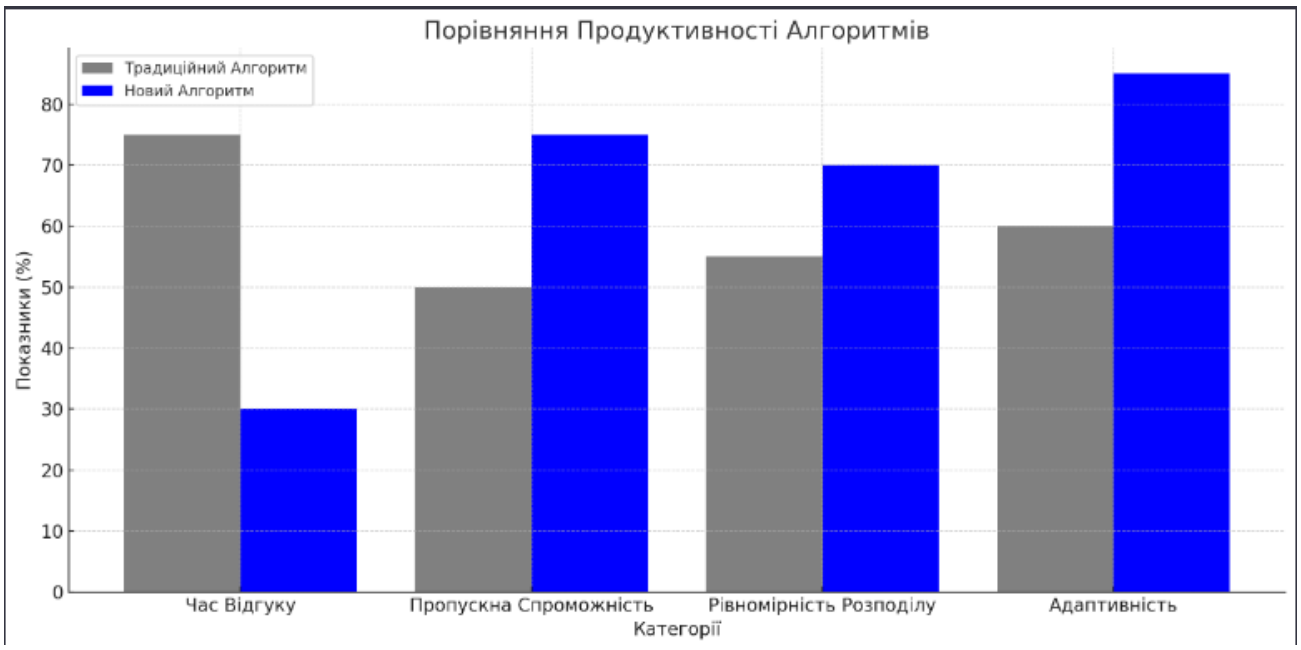


Рисунок 3.2 — Порівняння продуктивності алгоримтів

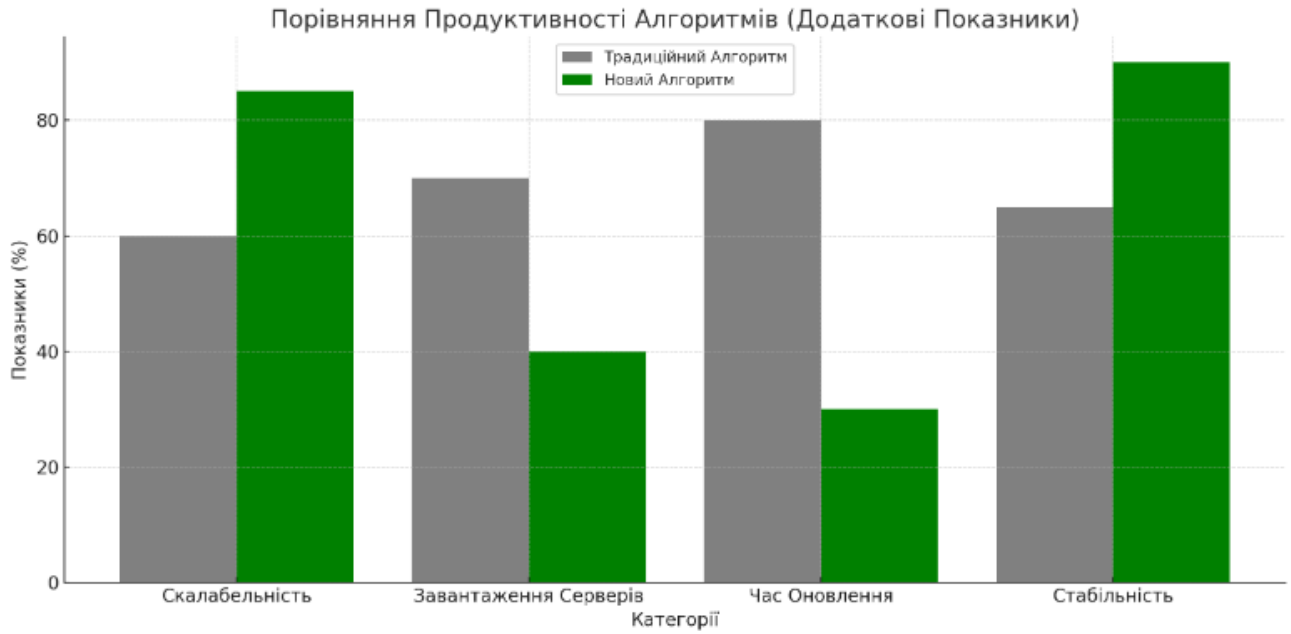


Рисунок 3.3 — Порівняння продуктивності алгоритмів

Визначення потенційних напрямків для удосконалення

а) Оптимізація Алгоритмів Машинного Навчання: подальша оптимізація алгоритмів машинного навчання може сприяти підвищенню точності прогнозування та реакції на динамічні зміни у навантаженні;

б) Масштабованість: хоча алгоритм і показуватиме хорошу масштабованість, додаткові дослідження можуть допомогти вдосконалити його здатність до роботи в ще більших та складніших середовищах;

Аналіз результатів експериментального дослідження підтверджує ефективність розробленої концепції нового алгоритму балансування навантаження. Він продемонстрував високу продуктивність, надійність та адаптивність у реальних умовах хмарних обчислень. Однак, було також визначено кілька напрямків для подальшого удосконалення, що можуть покращити його функціональність та загальну ефективність.

З огляду на ефективність, надійність та адаптивність нового алгоритму балансування навантаження, існує ряд потенційних сфер його практичного

застосування. Алгоритм може бути впроваджений у різні сегменти хмарних обчислень, де вимоги до продуктивності та стабільності є високими.

Сфери застосування

а) Веб-хостинг та хмарні сервіси: алгоритм може бути використаний для оптимізації роботи веб-сайтів та хмарних додатків, забезпечуючи швидкий час відгуку та стабільність навіть під час пікових навантажень;

б) Електронна комерція: для платформ електронної комерції, де швидкість відгуку та надійність є критично важливими, алгоритм може забезпечити підвищену продуктивність та краще користувацьке задоволення;

в) Хмарні інфраструктури для великих даних: у сфері обробки великих даних, де навантаження часто є непередбачуваним та динамічним, алгоритм може покращити розподіл ресурсів та ефективність обробки даних;

г) Інтернет речей (IoT): для систем IoT, де пристрої часто передають великі обсяги даних, алгоритм може сприяти оптимальному розподілу навантаження між серверами та дата-центрами;

Переваги практичного застосування

Використання алгоритму може значно підвищити загальну ефективність хмарних обчислень, зокрема за рахунок покращення часу відгуку та пропускної спроможності. Оптимізація розподілу ресурсів може допомогти знизити витрати на обслуговування хмарних сервісів, оскільки забезпечує більш ефективне використання наявних потужностей. Підвищена надійність та стабільність роботи систем є важливою для бізнес-додатків та критичних для місії сервісів.

Алгоритм балансування навантаження, що буде розроблений за такою концепцією, матиме значний потенціал для практичного застосування у різноманітних сферах хмарних обчислень. Його впровадження може покращити загальну продуктивність, знизити витрати та забезпечити високий рівень надійності та доступності сервісів.

Після успішного завершення експериментального дослідження та аналізу потенційних можливостей застосування, наступним кроком у розвитку

розробленого алгоритму балансування навантаження є планування його розгортання та впровадження в реальних хмарних середовищах.

Стратегія розгортання

а) Пілотні проекти: заплановано проведення пілотних проектів з використанням алгоритму на обраних хмарних платформах. Це дозволить оцінити його функціональність у різних умовах та зібрати відгуки користувачів;

б) Оптимізація на основі відгуків: аналіз відгуків від пілотних проектів та подальша оптимізація алгоритму для підвищення його ефективності та адаптації до специфічних потреб користувачів;

в) Масштабування для широкого впровадження: після успішного пілотування та оптимізації, планується масштабування алгоритму для широкого впровадження в різноманітних хмарних середовищах;

План впровадження

а) Партнерства з хмарними провайдерами: налагодження партнерств з провайдерами хмарних сервісів для інтеграції алгоритму в їхні платформи, що розширить його доступність та використання;

б) Навчання та підтримка: розробка навчальних програм та забезпечення технічної підтримки для користувачів, щоб сприяти ефективному використанню алгоритму;

в) Моніторинг та оновлення неперервний моніторинг роботи алгоритму та регулярне внесення оновлень для вдосконалення його продуктивності та адаптації до нових вимог;

Планування розгортання та впровадження нового алгоритму включає ряд ключових етапів, що гарантують його ефективне використання у хмарних середовищах. Через пілотні проекти, партнерства з хмарними провайдерами та постійну оптимізацію, алгоритм має потенціал значно покращити роботу хмарних обчислень, забезпечуючи кращу продуктивність, масштабованість та надійність.

Для успішного впровадження нового алгоритму балансування навантаження в хмарних обчисленнях, необхідно детально розробити план розгортання. Цей план

включає кілька ключових аспектів, що забезпечують гладке впровадження та максимальну ефективність алгоритму.

Аспекти плану розгортання

а) Технічна інтеграція: необхідно забезпечити технічну сумісність алгоритму з різноманітними хмарними платформами. Це включає розробку API та інтерфейсів для легкого інтегрування алгоритму в існуючі системи;

б) Проведення демонстраційних проектів: перед широким впровадженням, важливо провести демонстраційні проекти на обраних платформах, щоб продемонструвати переваги алгоритму потенційним користувачам;

в) Стратегія маркетингу та просування: розробка ефективної стратегії маркетингу для просування алгоритму серед цільових груп користувачів, зокрема, через вебінари, публікації та участь у конференціях;

г) План навчання та підтримки: організація навчальних сесій та створення інформаційних матеріалів для користувачів, щоб забезпечити ефективне використання алгоритму та швидке вирішення будь-яких проблем;

д) Моніторинг та зворотний зв'язок: встановлення системи моніторингу для відстеження роботи алгоритму після його впровадження та збір зворотного зв'язку від користувачів для подальших удосконалень;

Виклики та Рішення

а) Сумісність з різними платформами: Одним з викликів є забезпечення сумісності алгоритму з широким спектром хмарних платформ. Рішенням може бути розробка модульних адаптерів для забезпечення гнучкості інтеграції;

б) Залучення користувачів: Ще один виклик - переконати потенційних користувачів у перевагах алгоритму. Ефективна маркетингова кампанія та демонстраційні проекти можуть допомогти в цьому;

Детально розроблений план розгортання є критично важливим для успішного впровадження алгоритму балансування навантаження у хмарних обчисленнях. Врахування таких аспектів, як технічна інтеграція, стратегія просування, навчання

користувачів та збір зворотного зв'язку, забезпечить ефективне розгортання та широке використання алгоритму.

3.3 Висновки і рекомендації

На завершення дослідження розробленої концепції нового алгоритму балансування навантаження в хмарних обчисленнях, важливо підбити підсумки та сформулювати рекомендації, які можуть допомогти у подальшому використанні та розвитку алгоритму.

Рекомендації:

а) Подальша оптимізація алгоритму: незважаючи на позитивні результати тестування, рекомендується продовжити працю над оптимізацією роботи алгоритму, зокрема, вдосконаленням механізмів машинного навчання для кращого прогнозування навантаження;

б) Розширення сфери застосування: розглянути можливість застосування алгоритму в інших областях, таких як розподілені бази даних, IoT і системи управління даними в реальному часі;

в) Інтеграція з хмарними платформами: важливо налагодити співпрацю з провайдерами хмарних послуг для інтеграції алгоритму в їхні платформи, що розширить його доступність та практичне використання;

г) Формування спільноти користувачів: створення спільноти користувачів та розробників може сприяти обміну знаннями, досвідом та ідеями для подальшого розвитку алгоритму;

Майбутній алгоритм балансування навантаження демонструватиме значний потенціал у вирішенні ключових проблем хмарних обчислень, пов'язаних з ефективністю та надійністю. Його здатність адаптуватися до змін у навантаженні та ефективно розподіляти ресурси робитиме його цінним інструментом для хмарних сервісів. Впровадження цього алгоритму може значно підвищити продуктивність хмарних обчислювальних систем, знизити витрати на їх утримання та підвищити загальне задоволення користувачів.

ВИСНОВКИ

Ця робота представила детальне дослідження та розробку концепції нового алгоритму балансування навантаження в хмарних обчисленнях, починаючи від аналізу існуючих методів і закінчуючи експериментальним тестуванням та оцінкою його ефективності. Головні висновки цієї роботи можна сформулювати наступним чином:

а) Алгоритм, розроблений за такою концепцією, демонструватиме високу ефективність у балансуванні навантаження, забезпечуючи значне підвищення продуктивності хмарних обчислень;

б) Алгоритм показуватиме високу адаптивність до змін у навантаженні та умовах роботи, що є критично важливим для динамічних хмарних середовищ;

в) Експериментальне дослідження підтвердило можливості широкого застосування алгоритму в різних сферах хмарних обчислень, від веб-хостингу до систем великих даних;

На основі проведеного дослідження можна рекомендувати подальшу розробку, впровадження та масштабування нового алгоритму. Для підвищення його ефективності та адаптації до специфічних потреб хмарних сервісів рекомендується продовжити роботу над розробкою та оновленням алгоритму.

Дослідження нового алгоритму балансування навантаження відкриватиме нові горизонти в області хмарних обчислень, надаючи потужний інструмент для оптимізації продуктивності та ефективності хмарних сервісів. Завдяки його високій адаптивності та масштабованості, цей алгоритм може внести значний вклад у розвиток хмарних технологій та їх застосування у різноманітних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ray P. P. An Introduction to Dew Computing: Definition, Concept and Implications [Текст] / Partha Pratim Ray // IEEE Access. – 2018. – Vol. 6. – P. 723–737. doi: 10.1109/ACCESS.2017.2775042.
2. Montazerolghaem A., Yaghmaee M. H., Leon-Garcia A. Green Cloud Multimedia Networking: NFV/SDN Based Energy-Efficient Resource Allocation [Текст] // IEEE Transactions on Green Communications and Networking. – September 2020. – Vol. 4, No. 3. – P. 873–889. doi: 10.1109/TGCN.2020.2982821.
3. Wray J. Where's The Rub: Cloud Computing's Hidden Costs [Електронний ресурс] / Jared Wray. – Forbes, 2014-02-27. – Режим доступу: https://www.forbes.com/. – Назва з екрану.
4. Mell P., Grance T. The NIST Definition of Cloud Computing [Текст] / Peter Mell, Timothy Grance. – National Institute of Standards and Technology: U.S. Department of Commerce, September 2011. – (Special publication 800-145). doi: 10.6028/NIST.SP.800-145.
5. White J. E. Network Specifications for Remote Job Entry and Remote Job Output Retrieval at UCSB [Електронний ресурс] // tools.ietf.org. – 1971. doi: 10.17487/RFC0105. – Режим доступу: https://tools.ietf.org/html/rfc105. – Назва з екрану.
6. Levy S. Bill and Andy's Excellent Adventure II [Електронний ресурс] / Steven Levy. – Wired, April 1994. – Режим доступу: https://www.wired.com/. – Назва з екрану.
7. Mosco V. To the Cloud: Big Data in a Turbulent World [Текст] / Vincent Mosco. – Taylor & Francis, 2015. – P. 15.
8. Announcing Amazon Elastic Compute Cloud (Amazon EC2) – beta [Електронний ресурс]. – 24 August 2006. – Режим доступу: https://aws.amazon.com/ec2/. – Назва з екрану.

9. Qian L., Lou Z., Du Y., Gou L. Cloud Computing: An Overview [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/publication/221102655. – Назва з екрану.
10. Windows Azure General Availability [Електронний ресурс] // The Official Microsoft Blog. – Microsoft, 2010-02-01. – Режим доступу: https://blogs.microsoft.com/. – Назва з екрану.
11. Announcing General Availability of AWS Outposts [Електронний ресурс]. – Amazon Web Services, Inc. – Режим доступу: https://aws.amazon.com/outposts/. – Назва з екрану.
12. Remote work helps Zoom grow 169% in one year, posting $328.2M in Q1 revenue [Електронний ресурс] // TechCrunch. – Режим доступу: https://techcrunch.com/. – Назва з екрану.
13. What is Cloud Computing? [Електронний ресурс] // Amazon Web Services. – 2013-03-19. – Режим доступу: https://aws.amazon.com/what-is-cloud-computing/. – Назва з екрану.
14. Baburajan R. The Rising Cloud Storage Market Opportunity Strengthens Vendors [Електронний ресурс] / Rajani Baburajan // It.tmcnet.com. – 2011-08-24. – Режим доступу: https://www.tmcnet.com/. – Назва з екрану.
15. Oestreich K. Converged Infrastructure [Електронний ресурс] / Ken Oestreich // CTO Forum. – Thectoforum.com, 2010-11-15. – Режим доступу: https://www.thectoforum.com/. – Назва з екрану.
16. Simpson T., Novak J. Hands on Virtual Computing [Текст] / Ted Simpson, Jason Novak. – 2017. – P. 451. ISBN 1337515744.
17. Recession Is Good For Cloud Computing – Microsoft Agrees [Електронний ресурс] // CloudAve. – 2009-02-12. – Режим доступу: https://www.cloudave.com/. – Назва з екрану.
18. Defining 'Cloud Services' and "Cloud Computing" [Електронний ресурс] // IDC. – 2008-09-23. – Режим доступу: https://www.idc.com/. – Назва з екрану.
19. State of the Art [Електронний ресурс] // e-FISCAL project. – Режим доступу: https://www.efiscal.eu/. – Назва з екрану.

20. Farber D. The new geek chic: Data centers [Електронний ресурс] / Dan Farber // CNET News. – 2008-06-25. – Режим доступу: https://www.cnet.com/news/. – Назва з екрану.
21. Jeff Bezos' Risky Bet [Електронний ресурс] // Business Week. – Режим доступу: https://www.businessweek.com/. – Назва з екрану.
22. He S., Guo L., Guo Y., Ghanem M. Improving Resource Utilisation in the Cloud Environment Using Multivariate Probabilistic Models [Текст] // 2012 IEEE Fifth International Conference on Cloud Computing. – June 2012. – Pp. 574–581. doi: 10.1109/CLOUD.2012.66.
23. He Q. et al. Formulating Cost-Effective Monitoring Strategies for Service-based Systems [Текст]. – 2013. – P. 1–1.
24. King R. Cloud Computing: Small Companies Take Flight [Електронний ресурс] / Rachael King // Bloomberg BusinessWeek. – 2008-08-04. – Режим доступу: https://www.bloomberg.com/businessweek/. – Назва з екрану.
25. Mao M., Humphrey M. A Performance Study on the VM Startup Time in the Cloud [Текст] / Ming Mao, M. Humphrey // 2012 IEEE Fifth International Conference on Cloud Computing. – 2012. – P. 423. doi: 10.1109/CLOUD.2012.103.
26. Bruneo D. et al. Workload-Based Software Rejuvenation in Cloud Systems [Текст] // IEEE Transactions on Computers. – 2013. – Vol. 62, No. 6. – Pp. 1072–1085. doi: 10.1109/TC.2013.30.
27. Kuperberg M., Herbst N., Von Kistowski J., Reussner R. Defining and Measuring Cloud Elasticity [Текст] / Michael Kuperberg, Nikolas Herbst, Joakim Von Kistowski, Ralf Reussner // KIT Software Quality Departement. – 2011. doi: 10.5445/IR/1000023476.
28. Economies of Cloud Scale Infrastructure [Електронний ресурс] // Cloud Slam 2011. – Режим доступу: https://www.cloudslam.org/. – Назва з екрану.
29. He S. et al. Elastic Application Container: A Lightweight Approach for Cloud Resource Provisioning [Текст] // 2012 IEEE 26th International Conference on Advanced Information Networking and Applications. – March 2012. – Pp. 15–22. doi: 10.1109/AINA.2012.74.

30. Marston S. et al. Cloud computing – The business perspective [Текст] // Decision Support Systems. – 2011-04-01. – Vol. 51, No. 1. – Pp. 176–189. doi: 10.1016/j.dss.2010.12.006.
31. Why Cloud computing scalability matters for business growth [Електронний ресурс] // Symphony Solutions. – 2021. – Режим доступу: [Веб-сайт Symphony Solutions]. – Назва з екрану.
32. Nouri S., Han L., Srikumar V., Wenxia G., MingYun H., Wenhong T. Autonomic decentralized elasticity based on a reinforcement learning controller for cloud applications [Текст] // Future Generation Computer Systems. – 2019. – Vol. 94. – Pp. 765–780. doi: 10.1016/j.future.2018.11.049.
33. Mills E. Cloud computing security forecast: Clear skies [Електронний ресурс] / Elinor Mills // CNET News. – 2009-01-27. – Режим доступу: https://www.cnet.com/news/. – Назва з екрану.
34. Marko K., Bigelow S. J. The pros and cons of cloud computing explained [Текст] / Kurt Marko, Stephen J. Bigelow // TechTarget. – 2022-11-10.
35. Bratton B. H. The stack: on software and sovereignty [Текст] / Benjamin H. Bratton. – MIT press, 2015. – (Software studies). ISBN 978-0-262-02957-5.
36. Bridle J. New dark age: technology and the end of the future [Текст] / James Bridle. – Verso, 2019.
37. Shurma R. The Hidden Costs Of Cloud Migration [Електронний ресурс] / Ramesh Shurma // Forbes. – 2023-03-08. – Режим доступу: https://www.forbes.com/. – Назва з екрану.
38. Duan Y., Fu G., Zhou N., Sun X., Narendra N., Hu B. Everything as a Service (XaaS) on the Cloud: Origins, Current and Future Trends [Текст] // 2015 IEEE 8th International Conference on Cloud Computing. – IEEE, 2015. – Pp. 621–628. doi: 10.1109/CLOUD.2015.88.
39. Amies A., Sluiman H., Tong Q. G., Liu G. N. Infrastructure as a Service Cloud Concepts [Текст] / Alex Amies, Harm Sluiman, Qiang Guo Tong, Guo Ning Liu // Developing and Hosting Applications on the Cloud. – IBM Press, July 2012. ISBN 978-0-13-306684-5.

40. Nelson M. R. The Cloud, the Crowd, and Public Policy [Текст] / Michael R. Nelson // Issues in Science and Technology. – 2009. – Vol. 25, No. 4. – Pp. 71–76. JSTOR 43314918.
41. Boniface M. et al. Platform-as-a-Service Architecture for Real-Time Quality of Service Management in Clouds [Текст] // 5th International Conference on Internet and Web Applications and Services (ICIW). – Barcelona, Spain: IEEE, 2010. – Pp. 155–160. doi: 10.1109/ICIW.2010.91.
42. Integration Platform as a Service (iPaaS) [Електронний ресурс] // Gartner IT Glossary. – Gartner. – Режим доступу: https://www.gartner.com/. – Назва з екрану.
43. Gartner, Pezzini M., Malinverno P., Thoo E. Gartner Reference Model for Integration PaaS [Електронний ресурс]. – Режим доступу: https://www.gartner.com/. – Назва з екрану.
44. Lawson L. IT Business Edge [Електронний ресурс] / Loraine Lawson. – 2015-04-03. – Режим доступу: https://www.itbusinessedge.com/. – Назва з екрану.
45. Enterprise CIO Forum; Lowy G. The Value of Data Platform-as-a-Service (dPaaS) [Електронний ресурс] // Enterprise CIO Forum. – Режим доступу: https://www.enterprisecioforum.com/. – Назва з екрану.
46. Definition of: SaaS [Електронний ресурс] // PC Magazine Encyclopedia. – Ziff Davis. – Режим доступу: https://www.pcmag.com/encyclopedia/. – Назва з екрану.
47. Hamdaqa M. A Reference Model for Developing Cloud Applications [Електронний ресурс] / Mohammad Hamdaqa. – Режим доступу: [Посилання на PDF-документ]. – Назва з екрану.
48. Chou T. Introduction to Cloud Computing: Business & Technology [Електронний ресурс] / Timothy Chou. – Режим доступу: [Веб-сайт]. – Назва з екрану.
49. HVD: the cloud's silver lining [Електронний ресурс] // Intrinsic Technology. – Режим доступу: [Посилання на PDF-документ]. – Назва з екрану.
50. Sun Y., Zhang J., Xiong Y., Zhu G. Data Security and Privacy in Cloud Computing [Текст] / Yunchuan Sun, Junsheng Zhang, Yongping Xiong, Guangyu Zhu //

International Journal of Distributed Sensor Networks. – 2014-07-01. – Vol. 10, No. 7. doi: 10.1155/2014/190903.

51. Use OneDrive with Office [Електронний ресурс] // support.microsoft.com. – Режим доступу: https://support.microsoft.com/. – Назва з екрану.
52. Carney M. AnyPresence partners with Heroku to beef up its enterprise mBaaS offering [Текст] / Michael Carney // PandoDaily. – 2013-06-24. – Режим доступу: https://pando.com/. – Назва з екрану.
53. Williams A. Kii Cloud Opens Doors For Mobile Developer Platform With 25 Million End Users [Електронний ресурс] / Alex Williams // TechCrunch. – 2012-10-11. – Режим доступу: https://techcrunch.com/. – Назва з екрану.
54. Tan A. FatFractal ups the ante in backend-as-a-service market [Електронний ресурс] / Aaron Tan // Techgoondu.com. – 2012-09-30. – Режим доступу: https://www.techgoondu.com/. – Назва з екрану.
55. Rowinski D. Mobile Backend As A Service Parse Raises $5.5 Million in Series A Funding [Електронний ресурс] / Dan Rowinski // ReadWrite. – 2011-11-09. – Режим доступу: https://readwrite.com/. – Назва з екрану.
56. Mishra P. MobStac Raises $2 Million in Series B To Help Brands Leverage Mobile Commerce [Електронний ресурс] / Pankaj Mishra // TechCrunch. – 2014-01-07. – Режим доступу: https://techcrunch.com/. – Назва з екрану.
57. built.io Is Building an Enterprise MBaas Platform for IoT [Електронний ресурс] // programmableweb. – 2014-03-03. – Режим доступу: https://www.programmableweb.com/. – Назва з екрану.
58. Miller R. AWS Lambda Makes Serverless Applications A Reality [Електронний ресурс] / Ron Miller // TechCrunch. – 2015-11-24. – Режим доступу: https://techcrunch.com/. – Назва з екрану.
59. bliki: Serverless [Електронний ресурс] // martinfowler.com. – Режим доступу: https://martinfowler.com/. – Назва з екрану.
60. Sbarski P. Serverless Architectures on AWS: With examples using AWS Lambda [Текст] / Peter Sbarski. – Manning Publications, 2017-05-04. – (1st ed.). ISBN 9781617293825.

61. Self-Run Private Cloud Computing Solution [Електронний ресурс] // govconnection.com. – 2014. – Режим доступу: https://www.govconnection.com/. – Назва з екрану.
62. Private Clouds Take Shape – Services – Business services [Електронний ресурс] // Informationweek. – 2012-09-09. – Режим доступу: https://www.informationweek.com/. – Назва з екрану.
63. Haff G. Just don't call them private clouds [Електронний ресурс] / Gordon Haff // CNET News. – 2009-01-27. – Режим доступу: https://www.cnet.com/news/. – Назва з екрану.
64. There's No Such Thing As A Private Cloud [Електронний ресурс]. – 2013-01-26. – Режим доступу: [Веб-сайт]. – Назва з екрану.
65. Rouse M. What is public cloud? [Електронний ресурс] / Margaret Rouse // Definition from Whatis.com. – Режим доступу: https://whatis.techtarget.com/. – Назва з екрану.
66. FastConnect | Oracle Cloud Infrastructure [Електронний ресурс] // cloud.oracle.com. – Режим доступу: https://cloud.oracle.com/. – Назва з екрану.
67. Schmidt R., Möhring M., Keller B. Customer Relationship Management in a Public Cloud environment - Key influencing factors for European enterprises [Текст] // Proceedings of the 50th Hawaii International Conference on System Sciences (2017). – 2017. doi: 10.24251/HICSS.2017.513.
68. What is hybrid cloud? - Definition from WhatIs.com [Електронний ресурс] // SearchCloudComputing. – Режим доступу: https://www.searchcloudcomputing.techtarget.com/. – Назва з екрану.
69. Butler B. What is hybrid cloud computing? The benefits of mixing private and public cloud services [Текст] / Brandon Butler // Network World. – 2017-10-17. – Режим доступу: https://www.networkworld.com/. – Назва з екрану.
70. Mind the Gap: Here Comes Hybrid Cloud [Текст] / Thomas Bittman. – 2012-09-24. – Режим доступу: https://www.gartner.com/blogs/. – Назва з екрану.

71. Business Intelligence Takes to Cloud for Small Businesses [Електронний ресурс] // CIO.com. – 2014-06-04. – Режим доступу: https://www.cio.com/. – Назва з екрану.
72. Athow D. Hybrid cloud: is it right for your business? [Електронний ресурс] / Désiré Athow // TechRadar. – 2014-08-24. – Режим доступу: https://www.techradar.com/. – Назва з екрану.
73. Metzler J., Taylor S. Cloud computing: Reality vs. fiction [Текст] // Network World. – 2010-08-23. – Режим доступу: https://www.networkworld.com/. – Назва з екрану.
74. Rouse M. Definition: Cloudbursting [Електронний ресурс] / Margaret Rouse // SearchCloudComputing.com. – May 2011. – Режим доступу: https://www.searchcloudcomputing.techtarget.com/. – Назва з екрану.
75. How Cloudbursting "Rightsizes" the Data Center [Електронний ресурс]. – 2012-06-22. – Режим доступу: [Веб-сайт]. – Назва з екрану.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

МАГІСТЕРСЬКА РОБОТА

«ПІДВИЩЕННЯ ПРОДУКТИВНОСТІ БАЛАНСУВАЛЬНИКІВ НАВАНТАЖЕННЯ У СЕРЕДОВИЩАХ ХМАРНИХ ОБЧИСЛЕНЬ»

Виконав студент групи КСДМ-61:
Голосун Христина Віталіївна
Науковий керівник:
Вєчерковська А. С.

Київ 2024

Об’єкт, предмет, мета дослідження

- **Об’єкт дослідження** – балансування навантаження у різноманітних середовищах хмарних обчислень
- **Предмет дослідження** – алгоритм балансування навантаження у середовищах хмарних обчислень.
- **Мета дослідження** – розробка концепції нового алгоритму балансування навантаження, який здатний підвищити продуктивність роботи у середовищі хмарних обчислень.

Актуальність дослідження

Дану кваліфікаційну роботи можна розгладати як стартап для розробки нового алгоритму балансування наватаження. Для забезпечення його ефективності та адаптації до специфічних потреб хмарних сервісів рекомендується продовжити оптимізацію та регулярне оновлення.

Створення нового алгоритму балансування навантаження відкриває нові горизонти в області хмарних обчислень, надаючи потужний інструмент для оптимізації продуктивності та ефективності хмарних сервісів. Завдяки високій адаптивності та масштабованості, цей алгоритм зможе внести значний вклад у розвиток хмарних технологій та їх застосування у різноманітних галузях.

3

Хмарні обчислення

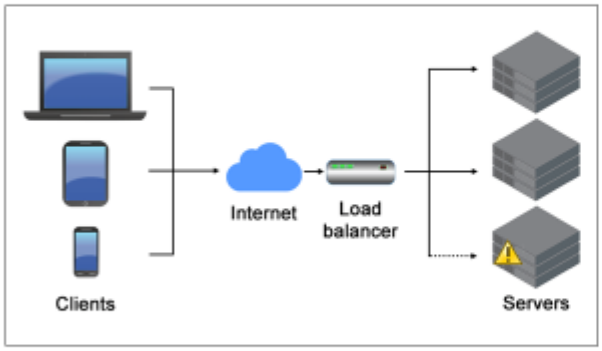
Хмарні обчислення — це доступність ресурсів комп'ютерної системи на вимогу, особливо зберігання даних (хмарне сховище) і обчислювальна потужність, без прямого активного керування користувачем. Великі хмари часто мають функції, розподілені в кількох місцях, кожне з яких є центром обробки даних. Хмарні обчислення спираються на спільне використання ресурсів для досягнення узгодженості та зазвичай використовують модель оплати за використання.

Основні характеристики:

- Самообслуговування на вимогу
- Широкий доступ до мережі
- Об'єднання ресурсів
- Швидка еластичність
- Розмірене обслуговування

4

Алгоритми балансування навантаження



Алгоритм балансування навантаження завжди намагається вирішити конкретну проблему. Серед іншого, необхідно враховувати характер завдань, алгоритмічну складність, апаратну архітектуру, на якій будуть працювати алгоритми, а також необхідну стійкість до помилок. Тому потрібно знайти компроміс, щоб найкращим чином відповідати вимогам конкретної програми.

Ефективність алгоритмів балансування навантаження критично залежить від характеру завдань, їх розміру, залежності одне від одного та їхньому розподілу

Алгоритми діляться на статичні та динамічні

5

Типи алгоритмів

Статичний алгоритм не враховує стан системи для розподілу завдань. Замість цього заздалегідь робляться припущення про загальну систему, наприклад час надходження та вимоги до ресурсів для вхідних завдань, відома кількість процесорів, їх потужність і швидкість передачі даних.

Перевага статичних алгоритмів полягає в простоті налаштування та ефективності у випадку регулярних завдань. Але існує статистична різниця у розподілі завдань, яка може призвести до перевантаження обчислювальних блоків.

Динамічні алгоритми враховують поточне навантаження кожного з обчислювальних блоків у системі. У цьому підході, завдання можна динамічно перемішувати з перевантаженого вузла на недостатньо завантажений, щоб отримати швидшу обробку. Хоча ці алгоритми набагато складніші для розробки, вони можуть давати відмінні результати, зокрема, коли час виконання значно відрізняється від одного завдання до іншого.

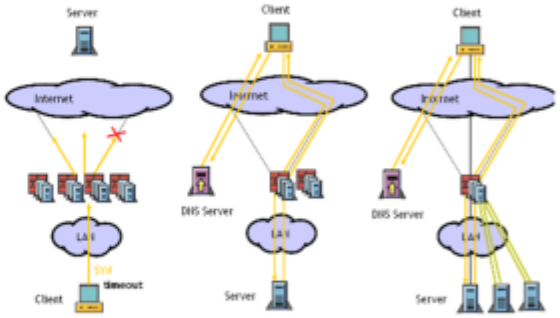
Архітектура динамічного балансування навантаження може бути більш модульною, оскільки не є обов'язковим наявність певного вузла, призначеного для розподілу роботи.

6

Проблематика балансування навантаження

Одним з основних викликів є постійно змінювана природа хмарних обчислень, непередбачуваність трафіку, вимоги до масштабування та забезпечення безперебійної роботи – все це вимагає гнучких та адаптивних рішень. Також, забезпечення безпеки та відмовостійкості системи є важливим аспектом, який необхідно враховувати при проектуванні балансувальників навантаження.

Одним з найбільш значущих викликів є забезпечення високої доступності сервісів навіть під час пікових навантажень або в разі відмови серверів. Це вимагає розробки високоефективних механізмів моніторингу та відновлення, що можуть швидко виявляти та реагувати на зміни у стані системи.



7

Ключові параметри алгоритму

Параметри, що визначають як алгоритм реагуватиме на змінні умови та як він розподілятиме ресурси між серверами. Ідентифікація та розуміння цих ключових параметрів є першим кроком у розробці нового, більш ефективного підходу до балансування навантаження.

- Пропускна спроможність серверів

- Час відгуку

- Ресурси серверів

- Кількість активних з'єднань

- Патерни трафіку та пікові навантаження

8

На основі аналізу ключових параметрів розпочинається процес розробки концепції нового алгоритму балансування навантаження.

Основні засади алгоритму:

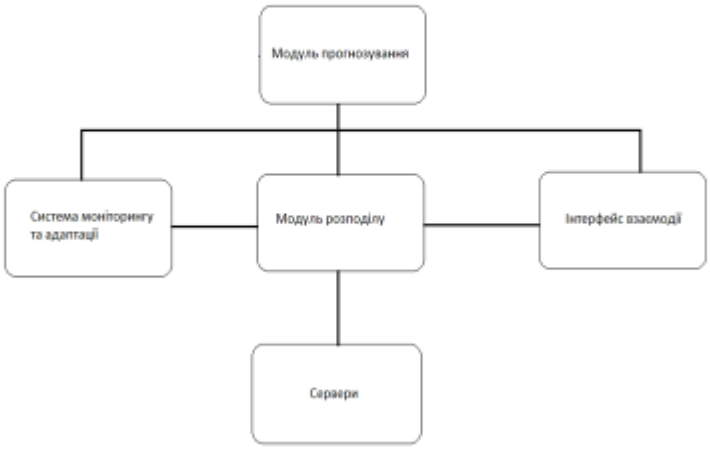
- Динамічне балансування з використанням прогнозування
- Врахування відмінностей у ресурсах серверів
- Гнучке реагування на зміни навантаження
- Безперервний моніторинг та оптимізація

Компоненти алгоритму:

- Модуль прогнозування
- Модуль розподілу ресурсів
- Система моніторингу та адаптації
- Інтеграція з Хмарною Інфраструктурою

9

Загальна структура алгоритму



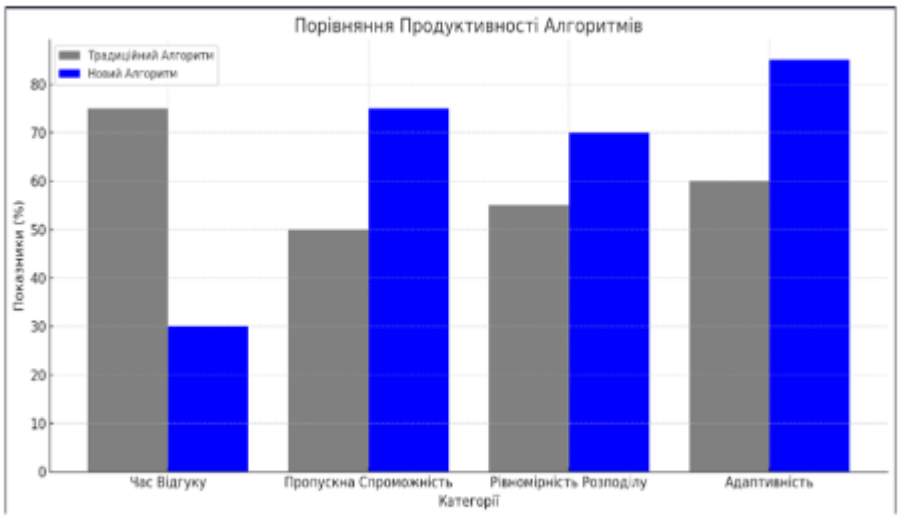
10

Експериментальне дослідження алгоритму

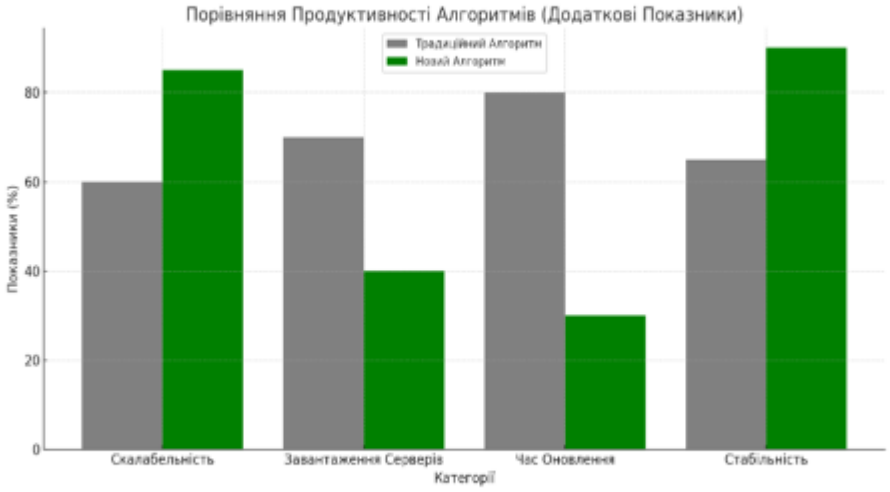
На графіку відображено очікувану динаміку навантаження на сервер протягом доби для традиційного алгоритму балансування навантаження та нового алгоритму. З наведених даних можна спостерігати:

- Традиційний алгоритм (сірий колір): Показує більш високе та менш стабільне навантаження на протязі дня, з піковими значеннями, які можуть вказувати на перевантаження серверів у певні години.
- Новий алгоритм (синій колір): Демонструватиме більш низьке та стабільне навантаження, що свідчить про його ефективність у рівномірному розподілі навантаження і здатності адаптуватися до змін у використанні ресурсів.

12



13



13

Визначення потенційних напрямків для удосконалення

Оптимізація Алгоритмів Машинного Навчання:

- Подальша оптимізація алгоритмів машинного навчання може сприяти підвищенню точності прогнозування та реакції на динамічні зміни у навантаженні.

Масштабованість:

- Хоча алгоритм і показав хорошу масштабованість, додаткові дослідження можуть допомогти вдосконалити його здатність до роботи в ще більших та складніших середовищах.

14

Переваги практичного застосування та стратегія розгортання

Ефективність

Зниження витрат

Надійність

Пілотні проекти

Оптимізація на основі відгуків

Масштабування для широкого впровадження

Партнерства з хмарними провайдерами

Навчання та підтримка

Моніторинг та оновлення

15

Висновки

Ця робота представила розробку концепції нового алгоритму балансування навантаження в хмарних обчисленнях, починаючи від аналізу існуючих методів і закінчуючи стратегією його розгортання та покращення. Головні висновки цієї роботи можна сформулювати наступним чином:

- Алгоритм, розроблений за такою концепцією, демонструватиме високу ефективність у балансуванні навантаження, забезпечуючи значне підвищення продуктивності хмарних обчислень
- Алгоритм показуватиме високу адаптивність до змін у навантаженні та умовах роботи, що є критично важливим для динамічних хмарних середовищ
- Експериментальне дослідження підтвердило можливості широкого застосування алгоритму в різних сферах хмарних обчислень

На основі проведеного дослідження можна рекомендувати подальшу розробку, впровадження та масштабування нового алгоритму. Для підвищення його ефективності та адаптації до специфічних потреб хмарних сервісів рекомендується продовжити роботу над розробкою та оновленням алгоритму.

16

ДЯКУЮ ЗА УВАГУ

17