МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження алгоритмів для управління хмарними ресурсами та оптимізації їх використання
(тема)

Виконав:
студент (ка) 2 курсу, групи ІПЗм-22-2

Бузько М.С.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-наукова

Керівник доц. Афанасьєва І.В.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
Кафедра Програмної інженерії
Рівень вищої освіти другий (магістерський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми освітньо-наукова програма
Освітня програма Інженерія програмного забезпечення

Курс 2 Группа ІПЗм-22-2 Семестр 2

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студенту Бузька Максима Сергійовича

1. Тема роботи: Дослідження алгоритмів для управління хмарними ресурсами та оптимізації їх використання

Затверджена наказом по університету від 29.03. 2024р. № 250 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 14.06.2024

3. Вихідні дані до роботи Середовище керування хмарними обчисленнями GCP, Основні типи віртуалізації, Система моделювання CloudSim

4. Перелік питань, що потрібно опрацювати в роботі *Аналіз Предметної Галузі, Методи Вирішення Проблеми, Розробка Алгоритмів Розподілення Віртуальних Машин За Хмарними Ресурсами, Реалізація Експерементів.*

КАЛЕНДАРНИЙ ПЛАН

Номер	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	23.01.2024 – 14.02.2024	*виконано*
2	Розробка постановки задачі	14.02.2024 – 21.03.2024	*виконано*
3	Розробка ПЗ	21.03.2024 – 02.04.2024	*виконано*
4	Експериментальні дослідження	02.04.2024 – 23.04.2024	*виконано*
5	Оформлення пояснювальної записки	01.05.2024 – 26.05.2024	*виконано*
6	Підготовка презентації та доповіді	26.04.2024 – 02.05.2024	*виконано*
7	Нормоконтроль, рецензування	02.06 – 11.06.24	*виконано*
8	Попередній захист	11.06.2024	*виконано*
9	Здача роботи у електронний архів	12.06.2024	*виконано*
10	Допуск до захисту у зав. кафедри	14.06.2024	*виконано*

Дата видачі завдання «19» січня 2024 р.

Студент _____ Бузько М.С.

(підпис)

Керівник роботи_____ доц. Афанасьєва І.В.

(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка містить 61 ст., 21 рис., 5 табл., 15 джерел.

АДАПТИВНІСТЬ АЛГОРИТМІВ, ЕФЕКТИВНІСТЬ ХМАРНИХ РЕСУРСІВ, ПРОДУКТИВНІСТЬ, УПРАВЛІННЯ РЕСУРСАМИ, ХМАРНІ ОБЧИСЛЕННЯ, AWS, AZURE, GCP, IAAS, PAAS, SAAS.

Об'єктом дослідження роботи є алгоритми для управління хмарними ресурсами та оптимізації їх використання. Це включає в себе вивчення та аналіз різних методів і стратегій, які можуть бути використані для ефективного управління ресурсами в хмарних обчисленнях.

Метою даної роботи є проведення комплексного аналізу та дослідження алгоритмів для управління хмарними ресурсами та оптимізації їх використання. Головні завдання включають вивчення існуючих стратегій управління ресурсами, аналіз їх ефективності, розробку критеріїв оцінки ефективності, та визначення оптимальних методів для покращення використання хмарних обчислювальних ресурсів.

В результаті проведеного дослідження буде отримано глибокий інсайт у сферу управління хмарними ресурсами та оптимізації їх використання. Ключові висновки та відкриття визначать ефективні алгоритми для керування ресурсами в хмарних обчисленнях, їх переваги та обмеження в конкретних сценаріях використання.

ADAPTABILITY OF ALGORITHMS, AWS, AZURE, EFFICIENCY OF CLOUD RESOURCES, GCP, CLOUD COMPUTING, IAAS, PAAS, PRODUCTIVITY, RESOURCE MANAGEMENT, SAAS.

The subject of research for the work are algorithms for managing wasteful resources and optimizing their resources. This includes learning and analysis of various

methods and strategies that can be used to effectively manage resources in low-income areas.

The purpose of this work is to conduct a comprehensive analysis and research of algorithms for managing wasteful resources and optimizing their use. The main tasks include the development of basic resource management strategies, analysis of their effectiveness, development of criteria for assessing effectiveness, and determination of optimal methods for reducing wasteful computing resources.

As a result of the investigation, deep insight will be obtained in the field of waste management and optimization of their resources. The key principles are to identify effective algorithms for resource management in heavy calculations, their advantages and sharing in specific resource scenarios.

Заява щодо самостійного виконання кваліфікаційної роботи та можливості її публікації в електронному архіві відкритого доступу ElArKhNURE.

Я, Бузько Максим Сергійович, студент гр. ІПЗм-22-2, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «ДОСЛІДЖЕННЯ АЛГОРИТМІВ ДЛЯ УПРАВЛІННЯ ХМАРНИМИ РЕСУРСАМИ ТА ОПТИМІЗАЦІЇ ЇХ ВИКОРИСТАННЯ», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений(на) з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

Вступ 9
1 Аналіз предметної галузі 10
1.1 Основні поняття та принципи 10
1.2 Види хмарних сервісів 11
1.3 Вагомі переваги та виклики використання хмарних ресурсів 13
1.4. Хмарні платформи 15
1.5 Постановка задачі 19
2 Методи вирішення проблеми 21
2.1 Алгоритми управління хмарними ресурсами 21
2.1.1 Алгоритми розподілу ресурсів 21
2.1.2 Алгоритми локалізації даних 22
2.1.3 Алгоритми енергозбереження 24
2.1.4 Алгоритми оркестрації та масштабування 25
2.2 Переваги та недоліки різних підходів 26
3 Розробка алгоритмів розподілення віртуальних машин за хмарними ресурсами 28
3.1 Огляд алгоритмів розв'язання проблем, пов'язаних із роботою 28
3.2 Збір даних про робоче навантаження 29
3.2.1 Параметри кластера Google 29
3.2.2 Характеристики кластера Bitbrains 29
3.2.3 Характеристики кластера Planet Lab 30
3.2.4 Характеристики моделі генерації ресурсів 31
4 Реалізація експерементів 32
4.1 Налаштування середовища GCP у експериментальному режимі 32
4.2 Розробка вигляду контейнера для клієнта GCP 33
4.2.1 Ініціалізація обгортки 34
4.2.2 Алгоритм прийняття рішень 35
4.2.3 Розміщення віртуальної машини 35
4.3 Монітор GCP 35

4.4 Алгоритми 36
4.4.1 Алгоритм першої відповідності 37
4.4.2 Алгоритм зменшення навантаження 38
4.4.3 Розширений алгоритм найкращої відповідності 38
4.5 Результати моделювання та експерементів 39
Висновки 44
Перелік джерел посилання 45
Додаток А Перелік джерел посилання науковими напрямами керівника та науковців кафедри Програмної інженерії 47
Додаток Б Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ 49
Додаток В Слайди презентації 50
Додаток Г Апробація у вигляді тез 58
Додаток Д Експертний висновок результатів перевірки кваліфікаційної роботи на відповідність оформлення вимогам ДСТУ 3008: 2015 61

ПЕРЕЛІК СКОРОЧЕНЬ

ВМ - Віртуальна Машина;

ОЗП - Оперативний Запам'ятовуючий Пристрій;

ЦП - Центральний Процесор;

API - Application Programming Interface;

AWS - Amazon Web Services;

CRM - Customer Relationship Management;

GCP - Google Cloud Platform;

IaaS - Infrastructure as a Service;

PaaS - Platform as a Service;

PM - Physical Machine;

SaaS - Software as a Service;

SLA - Service Level Agreement;

SQL - Structured Query Language;

VM - Virtual Machine;

YAML – yet another markup language (file format).

ВСТУП

Хмарні обчислення або скорочено хмара — модель забезпечення повсюдного та зручного доступу на вимогу через мережу до спільного пулу обчислювальних ресурсів, що підлягають налаштуванню (наприклад, до комунікаційних мереж, серверів, засобів збереження даних, прикладних програм та сервісів), і які можуть бути оперативно надані та звільнені з мінімальними управлінськими затратами та зверненнями до провайдера.

Хмарні обчислення стали ключовим елементом сучасного інформаційного ландшафту, надаючи організаціям та користувачам можливість доступу до розмаїття ресурсів через мережу Інтернет. За останні кілька років значущо зросла не тільки кількість доступних хмарних ресурсів, а й різноманіття послуг, що пропонуються. Від інфраструктури до програмного забезпечення, від сховищ даних до обчислювальної потужності - хмарні ресурси стали невід'ємною складовою сучасного IT.

Зростання попиту на хмарні ресурси породжує проблему ефективного управління цими ресурсами та оптимізації їх використання. В умовах зростаючої складності завдань, різноманіття вимог користувачів та динамічного середовища хмарних обчислень, виникає необхідність у розробці та вдосконаленні алгоритмів управління ресурсами.

Ця дослідницька робота призначена вивчити, аналізувати та порівнювати різні алгоритми управління хмарними ресурсами з метою розкриття їхніх переваг та недоліків. Розглядаючи аспекти розподілу ресурсів, локалізації даних, енергозбереження та оркестрації, ми прагнемо визначити оптимальні стратегії для різноманітних сценаріїв використання хмарних обчислень. Результати цього дослідження можуть служити основою для розробки більш ефективних та адаптивних систем управління хмарними ресурсами, що відповідають вимогам сучасного інформаційного середовища.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Основні поняття та принципи

Хмарні обчислення представляють собою парадигму обчислювальних послуг, що базуються на інтернет-технологіях [1]. Цей підхід дозволяє отримати доступ до обчислювальних ресурсів через мережу, без необхідності прямого володіння фізичними обладнаннями та інфраструктурою. У сутності, хмарні обчислення виступають як платформа, яка надає різноманітні послуги, такі як обчислювальна потужність, зберігання даних, мережеві ресурси та програмне забезпечення.

Основні принципи хмарних обчислень:

- самообслуговування (Self-Service) - користувачі можуть самостійно здійснювати інтерактивний та автоматизований доступ до ресурсів без залучення адміністраторів;
- швидкість та масштабованість (Rapid Elasticity) - можливість швидко та ефективно змінювати обсяг обчислювальних ресурсів в залежності від змін в навантаженні без значних зусиль;
- доступність через мережу (Broad Network Access) - хмарні ресурси доступні через мережу і можуть бути легко досягнуті з різних пристроїв та місць;
- резервування ресурсів за засобами (Resource Pooling) - ресурси, такі як обчислювальна потужність чи зберігання даних, об'єднуються та виділяються користувачам відповідно до їх потреб;
- служба за запитом (On-Demand Self-Service) - користувачі можуть отримувати необхідні ресурси в режимі реального часу, коли це необхідно, і без зайвого часу на підготовку.

Хмарні обчислення є ключовим елементом інформаційних технологій у сучасному світі, надаючи ефективний та гнучкий доступ до ресурсів для організацій та користувачів. Цілеспрямоване використання хмарних обчислень

дозволяє вдосконалити інфраструктуру, знизити витрати та підвищити ефективність управління обчислювальними ресурсами.

1.2 Види хмарних сервісів

Хмарні обчислення надають різні рівні послуг, що відповідають різним потребам користувачів і організацій. Три основні види хмарних сервісів включають IaaS (Інфраструктура як послуга), PaaS (Платформа як послуга) та SaaS (Програмне забезпечення як послуга) [2].

Інфраструктура як послуга (IaaS):

Визначення: IaaS надає віртуальну інфраструктуру для користувачів через Інтернет. Це включає в себе віртуальні машини, сховище даних, мережеві ресурси та інші обчислювальні ресурси.

Застосування: Користувачі IaaS можуть створювати власні віртуальні машини, розгортати операційні системи та розробляти власні застосунки, зберігаючи при цьому повний контроль над операційною системою та конфігурацією.

Платформа як послуга (PaaS):

Визначення: PaaS забезпечує платформу для розробки та розгортання веб-додатків без необхідності управління обчислювальною інфраструктурою.

Застосування: Розробники використовують PaaS для створення веб-додатків, не звертаючи уваги на деталі інфраструктури. Платформа включає середовище для розробки, бази даних, середовище виконання та інші компоненти.

Програмне забезпечення як послуга (SaaS):

Визначення: SaaS надає готові до використання програми та сервіси через хмарні ресурси, доступні для користувачів через мережу.

Застосування: Користувачі можуть використовувати готові програми, такі як офісні пакети, облікові системи чи інші програми без необхідності їх встановлення чи оновлення.

Ці три рівні хмарних сервісів відображають різні рівні абстракції від інфраструктури до готових до використання застосунків, що дозволяє

користувачам та організаціям вибирати той рівень, який найкраще відповідає їх потребам та завданням (див. рис. 1.1).

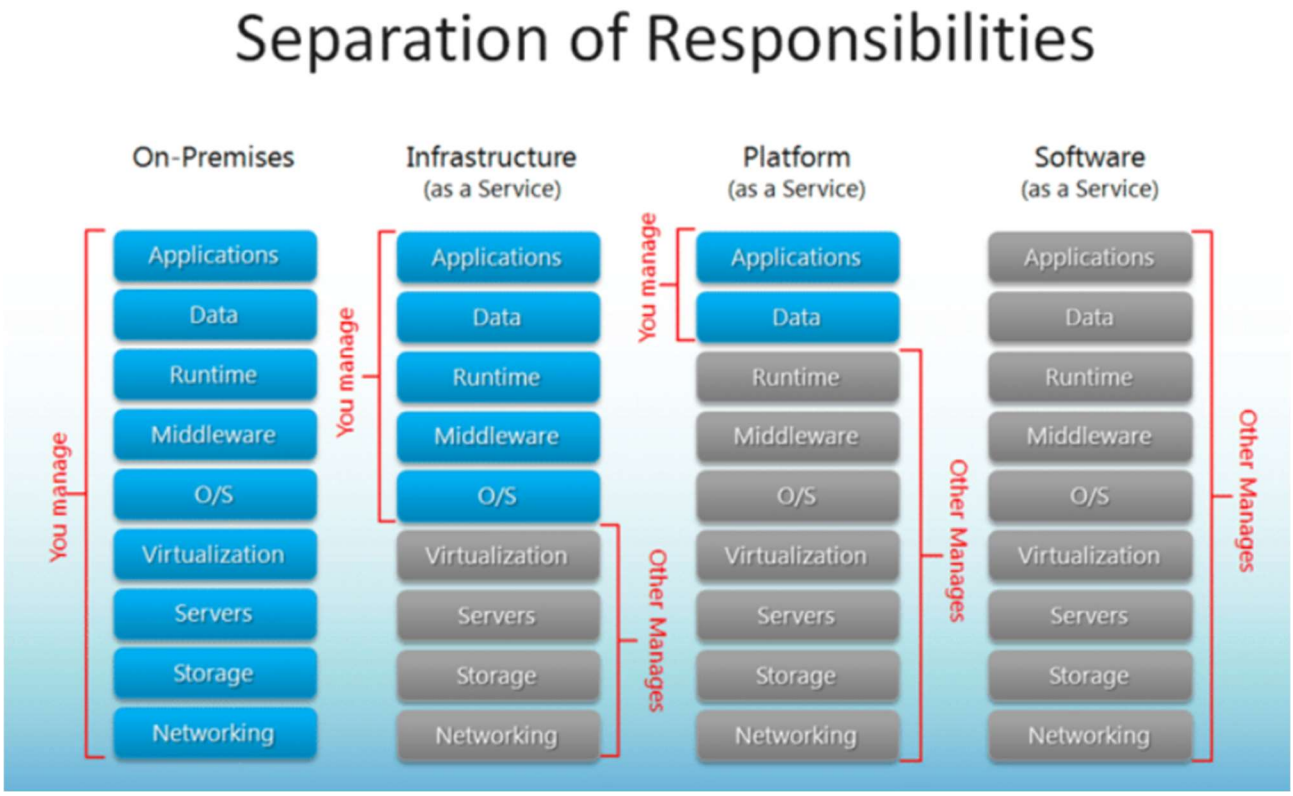

Рисунок 1.1 – Порівняння хмарних сервісів (за даними [3])

Загальною відмінністю також є поділ хмари на:

– загальнодоступний – користуватися ним може кожен, хто заплатить;
– приватний (On-Premise) – створюється на основі власної інфраструктури, до якої мають доступ конкретні люди, наприклад, співробітники даної компанії (простіше кажучи, приватну хмару можна розуміти, наприклад, як сервери компанії);
– гібридний – використовує як публічні, так і приватні хмарні ресурси.

CRM в моделі SaaS є надзвичайно простим варіантом, оскільки за певною платою ми отримуємо доступ до нашої CRM-системи, яку повністю підтримує постачальник послуг (необов'язково виробник). Останній несе відповідальність за оновлення та стабільність системи. CRM в моделі SaaS найзручніше для користувача, проте має обмеження щодо адаптації до конкретних бізнес-потреб і

відповідності вимогам зберігання особистих даних, що може бути особливо строгим у певних країнах та галузях.

Використання CRM у моделі PaaS передбачає можливість самостійно встановлювати систему та забезпечувати необхідну конфігурацію середовища для її належної роботи. Це рідкісна, але можлива ситуація, яка вимагає обережності для уникнення пошкоджень системи та водночас надає більше можливостей для модифікацій у порівнянні із SaaS.

Під CRM у моделі IaaS розуміється використання інфраструктури провайдера (часто з використанням віртуальних машин) для створення повного середовища, оптимізованого під систему CRM. Це вимагає значно більше IT-адміністративних знань, але дає практично необмежені можливості для оптимізації системи та адаптації до власних потреб, включаючи свободу вибору бази даних, операційної системи та інших складових середовища. Це також може бути використано для інших програм, наприклад, для автоматизації маркетингу, якщо необхідно.

Остання ситуація, яку ми розглянемо для кращого розуміння теми, це приватна хмара CRM. У цій ситуації, окрім хорошого адміністратора, нам також потрібна повна та правильно налаштована IT-інфраструктура (стосовно IaaS нам також потрібно подбати про конфігурацію мереж і серверів). Приватна хмара дає нам стільки ж можливостей для налаштування системи, скільки IaaS, і, крім того, ми отримуємо абсолютний контроль над потоком кожного байта наших даних CRM, що може бути особливо важливим у випадку компаній, які змушені проявляти надзвичайну турботу про безпека інформації, що зберігається в системі.

1.3 Вагомі переваги та виклики використання хмарних ресурсів

Використання хмарних ресурсів забезпечує компаніям низку значущих переваг [4]:

Ефективність витрат: Використання хмарних ресурсів дозволяє оптимізувати витрати, адже користувачі сплачують лише за фактично використані ресурси, уникаючи значних капіталовкладень у власні обчислювальні системи.

Масштабованість: Хмарні ресурси забезпечують можливість швидкого масштабування обчислювальної потужності відповідно до змін обсягу роботи чи навантаження, забезпечуючи ефективне реагування на змінні потреби.

Гнучкість та доступність: Забезпечення гнучкості та можливості доступу до ресурсів з будь-якого місця забезпечує мобільність та зручність роботи.

Швидкість впровадження: Хмарні ресурси дозволяють швидко розгортати та налаштовувати обчислювальні середовища без необхідності чекати на придбання та конфігурацію фізичного обладнання.

Автоматизація та моніторинг: Використання автоматизованих інструментів управління та моніторингу дозволяє оптимізувати використання ресурсів та надає можливість ефективно реагувати на зміни у реальному часі.

Ці переваги роблять хмарні ресурси привабливими для бізнесу та розробників, дозволяючи їм фокусуватися на стратегічних завданнях та розширювати свою інфраструктуру без великих витрат та технічних труднощів.

Використання хмарних ресурсів може стикатися з рядом викликів, таких як:

- безпека та конфіденційність: зростаючі загрози безпеки та обмеження конфіденційності можуть становити ризики для даних, особливо при їх зберіганні на віддалених серверах;
- залежність від мережі: недоступність інтернет-з'єднання може призвести до тимчасової втрати доступу до обчислювальних ресурсів та послуг хмари;
- ліцензування та вартість: складнощі у керуванні ліцензіями та неочікувані витрати на різні хмарні ресурси можуть призвести до фінансових труднощів;
- обмежена адаптованість: деякі хмарні рішення можуть бути обмеженими в можливості адаптації до специфічних потреб користувачів чи вимагати компромісів;

- локалізація даних: вимоги щодо локалізації даних можуть вимагати дотримання специфічних правил та обмежити використання хмарних ресурсів у певних регіонах або галузях;
- менеджмент та контроль: виникнення складнощів у керуванні та контролі над обчислювальними ресурсами, які знаходяться в хмарному середовищі.

Розуміння цих викликів є важливим для організацій при прийнятті рішення щодо використання хмарних ресурсів та розробки стратегій їх впровадження.

1.4. Хмарні платформи

Amazon, Microsoft та Google займають лідируючі позиції на ринку хмарних провайдерів, що надають безпечні, гнучкі та надійні хмарні послуги. Їх відповідні хмарні платформи, AWS, Azure та GCP, надають користувачам доступ до обчислюваних ресурсів та аналітичних інструментів, а також до зберігання даних, серверів, ПО тощо [5].

На даний момент AWS випереджає своїх конкурентів, Azure і GCP, з точки зору функціональності та доступності. Однак, Azure і GCP також не відстають, щоб довести своє панування на ринку (див. табл. 1).

Amazon Web Services (AWS) є найбільшою та найбільш розповсюдженою хмарною платформою у світі. Вона надає величезний набір сервісів, які охоплюють всі аспекти інфраструктури, включаючи обчислення, зберігання, бази даних, мережі та багато іншого. AWS використовується багатьма великими корпораціями та стартапами для забезпечення масштабованих і надійних рішень.

Microsoft Azure є хмарною платформою від Microsoft, яка надає широкий спектр послуг. Azure включає в себе різні обчислювальні ресурси, бази даних, аналітику, штучний інтелект та інші сервіси. Його інтеграція з продуктами Microsoft, такими як Windows Server, Active Directory та Office 365, робить його популярним серед корпорацій та організацій, що використовують екосистему Microsoft.

Google Cloud Platform (GCP) від Google надає послуги обчислення, аналітики, штучного інтелекту та інфраструктури. GCP вирізняється своєю сильною фокусірованістю на машинному навчанні та аналітиці даних. Крім того, вона використовує технології, розроблені Google, такі як Kubernetes для оркестрації контейнерів.

Таблиця 1.1 – Порівняння AWS з Azure і GCP (за даними [6])

Features	AWS	Azure	GCP
Compute Services	AWS Beanstalk Amazon EC2 Amazon EC2 Auto-Scaling Amazon Elastic Container Registry Amazon Elastic Kubernetes Service Amazon Lightsail AWS Serverless Application Repository VMware Cloud for AWS AWS Batch AWS Fargate AWS Lambda AWS Outposts Elastic Load Balancing	Platform-as-a-service (PaaS) Function-as-a-service (FaaS) Service Fabric Azure Batch Cloud Services Container Instances Batch Azure Container Service (AKS) Virtual Machines Compute Engine Virtual Machine Scale Sets	App Engine Docker Container Registry Instant Groups Compute Engine Graphics Processing Unit (GPU) Knative Kubernetes Functions
Storage Services	Simple Storage Service (S3) Elastic Block Storage (EBS) Elastic File System (EFS) Storage Gateway Snowball Snowball Edge Snowmobile	Blob Storage Queue Storage File Storage Disk Storage Data Lake Store	Cloud Storage Persistent Disk Transfer Appliance Transfer Service

Продовження таблиці 1.1

Features	AWS	Azure	GCP
AI/ML	SageMaker Comprehend Lex Polly Rekognition Machine Learning Translate Transcribe DeepLens Deep Learning AMIs Apache MXNet on AWS TensorFlow on AWS	Machine Learning Azure Bot Service Cognitive Services	Cloud Machine Learning Engine Dialogflow Enterprise Edition Cloud Natural Language Cloud Speech API Cloud Translation API Cloud Video Intelligence Cloud Job Discovery (Private Beta)
Database Services	Aurora RDS DynamoDB ElastiCache Redshift Neptune Database Migration Service	SQL Database Database for MySQL Database for PostgreSQL Data Warehouse Server Stretch Database Cosmos DB Table Storage Redis Cache Data Factory	Cloud SQL Cloud Bigtable Cloud Spanner Cloud Datastore
Backup Services	Glacier	Archive Storage Backup Site Recovery	Nearline (frequently accessed data) Coldline (infrequently accessed data)
Serverless computing	Lambda Serverless Application Repository	Functions	Google Cloud Functions
Strengths	Dominant market position Extensive, mature offerings Support for large organizations Global reach Flexibility and a wider range of services	Second largest provider Integration with Microsoft tools and software Broad feature set Hybrid cloud Support for open source Ideal for startups and developers	Designed for cloud-native businesses Commitment to open source and portability Flexible contracts DevOps expertise Complete container-based model Most cost-efficient
Caching	Elastic Cache	Redis Cache	Cloud CDN
File Storage	EFS	Azure Files	ZFS and Avere
Networking	Amazon Virtual Private Cloud (VPC)	Azure Virtual Network (VNET)	Cloud Virtual Network

кінець таблиці 1.1

Features	AWS	Azure	GCP
Security	AWS Security Hub	Azure Security Center	Cloud Security Command Center
Location	77 availability zones within 24 geographic regions	Presence in 60+ regions across the world	Presence in 24 regions and 73 zones. Available in 200+ countries and territories
Documentation	Best in class	High quality	High quality
DNS Services	Amazon Route 53	Azure Traffic Manager	Cloud DNS
Notifications	Amazon Simple Notification Service (SNS)	Azure Notification Hub	None
Load Balancing	Elastic Load Balancing	Load Balancing for Azure	Cloud Load Balancing
Automation	AWS Opsworks	Azure Automation	Compute Engine Management
Compliance	AWS CloudHSM	Azure Trust Center	Google Cloud Platform Security

Відмінності між трьома основними хмарними службами можна побачити, оцінивши їх за допомогою різних параметрів, таких як обчислювальна потужність, хмарне сховище, баз даних, покриття та документація.

Обчислювальна потужність: AWS використовує Elastic Compute Cloud (EC2), що відповідає за всі обчислювальні послуги, керуючи віртуальними машинами, які мають попередньо налаштовані параметри, а також за потреби можуть бути налаштовані користувачами. З іншого боку, Azure пропонує Virtual Machine Scale Sets в якості центру обробки даних, тоді як GCP використовує Google Compute Engine (GCE), який виконує ті ж функції.

Хмарне сховище: Amazon S3 – найкращий варіант для зберігання з обширною документацією, перевіреною технологією та належною підтримкою спільноти. Microsoft Azure Storage та Google Cloud Storage також пропонують надійні послуги зберігання з високою продуктивністю і надійним захистом.

Бази даних: Усі основні провайдери пропонують кілька інструментів та варіантів послуг, що стосуються баз даних. Служба реляційних баз даних Amazon (RDS) підтримує основні бази даних, такі як Oracle та PostgreSQL, і керує всім –

від оновлення до виправлення. База даних SQL Azure пропонує функції обробки бази даних SQL для Azure, а Cloud SQL – для GCP.

Покриття: Amazon має 77 зон доступності, Azure присутня у 60+ регіонах, а Google – у 33 країнах, нові регіони додаються регулярно.

Документація: Усі три постачальники пропонують високоякісну документацію, хоча AWS трохи випереджає Azure та GCP.

1.5 Постановка задачі

Для досягнення поставленої мети, а саме аналізу сучасних алгоритмів управління хмарними ресурсами та надання рекомендацій щодо їх використання треба:

- визначити переваги та недоліки кожного алгоритму в контексті оптимізації використання ресурсів;
- розглянути можливість застосування кожного алгоритму для різних сценаріїв використання хмарних сервісів;
- порівняти ефективність різних алгоритмів управління ресурсами за різних умов;
- сформулювати рекомендації щодо вибору оптимального алгоритму управління ресурсами для конкретних завдань у хмарних обчисленнях.

Завдання дослідження:

- провести огляд літератури та існуючих підходів до управління ресурсами у хмарних обчисленнях;
- вивчити та проаналізувати основні алгоритми розподілу ресурсів, локалізації даних, енергозбереження та оркестрації та масштабування;
- встановити критерії оцінки ефективності кожного алгоритму;

- провести експериментальне дослідження з використанням реальних або симулюваних даних для розробки алгоритму автоматичного розподілення навантаження системи;
- підготувати звіт із результатами дослідження та сформулювати рекомендації для вибору оптимального алгоритму управління ресурсами в хмарних обчисленнях.

Очікувані результати - дослідження дозволить отримати глибше розуміння різних алгоритмів управління ресурсами в хмарних обчисленнях та їх застосування в різних сценаріях. Результати дослідження зможуть надати корисні вказівки та рекомендації для вибору оптимального підходу до управління ресурсами в хмарному середовищі залежно від конкретних вимог та умов проекту.

2 МЕТОДИ ВИРІШЕННЯ ПРОБЛЕМИ

2.1 Алгоритми управління хмарними ресурсами

Управління хмарними ресурсами передбачає ефективне розподілення та використання обчислювальних, мережевих та зберігальних ресурсів у хмарному середовищі. Алгоритми управління грають ключову роль у забезпеченні оптимального використання ресурсів та виконанні завдань користувачів [7].

2.1.1 Алгоритми розподілу ресурсів

Хмарні обчислення надають користувачам гнучкість та ефективність у використанні обчислювальних ресурсів. Алгоритми розподілу ресурсів в хмарних обчисленнях виконують ключову роль у забезпеченні оптимального використання обчислювальних, мережевих та зберігальних можливостей. Розглянемо деякі основні алгоритми:

- Раунд-робін (Round Robin):

Опис: Цей алгоритм розподілу ресурсів працює за принципом чергування. Кожному користувачеві або віртуальній машині в змінному порядку надається доступ до ресурсів. Це сприяє рівномірному розподілу навантаження.

Переваги: Простий у реалізації, використовується для рівномірного розподілу ресурсів.

Недоліки: Не враховує різні рівні важливості або типи завдань.

- Множинне Чергування (Multilevel Queue):

Опис: Ресурси розділяються на декілька рівнів черги, де кожен рівень представляє собою окремий пріоритет чи тип ресурсів. Завдання переміщуються між рівнями в залежності від їхнього пріоритету та характеристик.

Переваги: Дозволяє краще управляти різними типами завдань та пріоритетами.

Недоліки: Складніше у впровадженні та налаштуванні.

- Перш Найменший Термін (Shortest Job Next, SJN):

Опис: Також відомий як алгоритм оптимального розподілу. Вибирає завдання з найменшим часом виконання для обробки на наступному кроці.

Переваги: Мінімізує час очікування та оптимізує загальний час виконання.

Недоліки: Вимагає точної інформації про час виконання завдань, що може бути непрактичним.

- Динамічне Призначення Пріоритетів (Dynamic Priority Assignment):

Опис: Пріоритети завдань динамічно призначаються в залежності від їхнього стану та пріоритету. Завданням із високим пріоритетом може бути виділено більше ресурсів.

Переваги: Забезпечує гнучкість та реагує на зміни у стані системи.

Недоліки: Вимагає додаткових обчислень для динамічного визначення пріоритетів.

- Фарм Хмарних Ресурсів (Cloud Resource Farming):

Опис: Завдання розподіляються між різними хмарними ресурсами, щоб оптимізувати їхнє використання. Може використовуватися для масштабування та забезпечення високої доступності.

Переваги: Використовується для оптимізації ресурсів та підтримки масштабування.

Недоліки: Вимагає додаткового управління та координації.

Ці алгоритми становлять основу для ефективного управління та оптимізації використання ресурсів у хмарних обчисленнях, забезпечуючи гнучкість, продуктивність та ефективність систем.

2.1.2 Алгоритми локалізації даних

Алгоритми локалізації даних в хмарних обчисленнях грають ключову роль у забезпеченні швидкого та ефективного доступу до інформації в різних областях хмарної інфраструктури. Розглянемо деякі з них:

– алгоритми реплікації даних: забезпечують створення кількох копій даних та їх розміщення в різних частинах хмарної інфраструктури. Користувач має можливість отримати доступ до найближчої копії даних, що підвищує швидкість доступу та надійність;

- алгоритми розміщення даних з урахуванням частоти доступу: базуються на ідеї розміщення найбільш часто використовуваних даних ближче до користувачів. Це зменшує час доступу до часто запитуваних ресурсів та оптимізує продуктивність;
- алгоритми керування розподілом даних: визначають оптимальні механізми для розподілу даних між різними вузлами хмарної інфраструктури. вони можуть враховувати фактори, такі як відстань, пропускна спроможність мережі та наявність резервних засобів;
- алгоритми динамічного розміщення даних: забезпечують можливість автоматичного перенесення даних між різними вузлами в залежності від змін у навантаженні чи стані системи. Це дозволяє оптимізувати використання ресурсів;
- алгоритми шардування (sharding): розбивають великі об'єкти даних на менші фрагменти, які називаються шардами, і розподіляють їх між різними серверами чи вузлами. Це покращує паралельність та швидкість доступу до конкретних фрагментів даних;
- алгоритми міграції даних: забезпечують можливість переміщення даних між різними ресурсами хмарної інфраструктури. Це може бути використано для оптимізації роботи та забезпечення балансу навантаження;
- алгоритми збереження даних з урахуванням географічних обмежень: враховують правила збереження даних відповідно до законодавства щодо захисту даних у конкретних географічних областях. Це дозволяє дотримуватися вимог до конфіденційності та безпеки.

Ці алгоритми визначають стратегії розподілу та управління даними в хмарних обчисленнях, забезпечуючи оптимальний доступ, надійність та відповідність вимогам конкретних застосунків.

2.1.3 Алгоритми енергозбереження

Енергозбереження в хмарних обчисленнях стає все більш важливим аспектом для забезпечення сталого розвитку та оптимізації використання ресурсів. Розглянемо деякі алгоритми, які спрямовані на мінімізацію витрат енергії:

- алгоритми керування ресурсами: адаптують кількість використовуваних ресурсів в залежності від поточного навантаження та вимог до обчислювальної потужності. Це дозволяє вимкнути або зменшити потужність неактивних ресурсів;
- алгоритми управління розміщенням ресурсів: визначають оптимальне розміщення віртуальних машин та обчислювальних ресурсів для зменшення витрат енергії. Можуть використовувати динамічне переміщення завдань до ресурсів із меншим споживанням енергії;
- алгоритми планування робочих навантажень: оптимізують розподіл робочих завдань для максимізації використання ресурсів та мінімізації енергоспоживання. Вони можуть враховувати зони навантаження з високою ефективністю;
- алгоритми керування температурою та охолодженням: моніторять та керують температурою в обчислювальних центрах. Забезпечують ефективне використання систем охолодження та регулюють потужність устаткування для зменшення витрат енергії;
- алгоритми резервного живлення: оптимізують використання резервних джерел енергії, таких як акумулятори чи системи неперервного живлення. Вони можуть активувати резервне живлення в періоди пікового навантаження чи аварій;
- алгоритми призупинення та відновлення ресурсів: дозволяють тимчасово призупиняти роботу неактивних ресурсів та відновлювати їхню роботу при потребі. Це дозволяє економити енергію в періоди малого навантаження;

– алгоритми динамічного споживання енергії: адаптуються до змін споживання енергії залежно від обчислювальних вимог. Наприклад, зменшують тактову частоту процесора або вимикати окремі компоненти обладнання в періоди невеликого навантаження.

Ці алгоритми спрямовані на оптимізацію витрат енергії у хмарних обчисленнях, забезпечуючи баланс між продуктивністю та сталістю. Ефективне впровадження цих стратегій допомагає знижувати енергоспоживання та покращувати сталість хмарних інфраструктур.

2.1.4 Алгоритми оркестрації та масштабування

Оркестрація та масштабування є ключовими аспектами управління ресурсами та забезпечення ефективності хмарної інфраструктури. Розглянемо деякі алгоритми, які використовуються для цих цілей:

– алгоритми оркестрації: оркестрація включає в себе координацію та управління різними компонентами хмарної інфраструктури. Алгоритми оркестрації дозволяють автоматизовано керувати розгортанням, моніторингом, та оптимізацією ресурсів.
– алгоритми динамічного масштабування: автоматично регулюють кількість ресурсів у залежності від змін в обсязі роботи або навантаження системи. Можуть використовувати горизонтальне чи вертикальне масштабування.
– алгоритми розподілення навантаження: розподіляють робочі завдання між різними вузлами хмарної інфраструктури для забезпечення рівномірного навантаження та уникнення перевантаження окремих ресурсів.
– алгоритми збільшення та зменшення ресурсів: автоматично виявляють потреби в збільшенні або зменшенні кількості ресурсів в системі і вживають відповідних заходів.

- алгоритми автоматичного завершення ресурсів: визначають ресурси, які залишаються неактивними або не використовуються, і автоматично їх завершують для економії ресурсів.
- алгоритми керування завданнями: визначають оптимальний спосіб розподілу завдань для досягнення максимальної продуктивності та швидкості виконання.
- алгоритми забезпечення високої доступності: забезпечують постійну доступність хмарної інфраструктури шляхом автоматичного виявлення та виправлення відмов.

Ці алгоритми дозволяють ефективно управляти та оптимізувати хмарні ресурси, забезпечуючи масштабованість та стабільність в залежності від змінних вимог та умов використання.

2.2 Переваги та недоліки різних підходів

Нижче подано порівняльну таблицю 2.1 для алгоритмів управління хмарними ресурсами, включаючи алгоритми розподілу ресурсів, локалізації даних, енергозбереження та оркестрації та масштабування.

Таблиця 2.1 – Порівняння алгоритмів управління хмарними ресурсами (створено самостійно)

Алгоритм	Переваги	Недоліки
Алгоритми Розподілу Ресурсів	Ефективне використання ресурсів. Можливість балансування завдань. Зменшення перевантаження та недоексплуатації ресурсів.	Потребує детального моніторингу та оцінки ресурсів. Складність у визначенні оптимального розподілу.

Кінець таблиці 2.1

Алгоритм	Переваги	Недоліки
Алгоритми Розподілу Ресурсів	Ефективне використання ресурсі. Можливість балансування завдань. Зменшення перевантаження та недоексплуатації ресурсів.	Потребує детального моніторингу та оцінки ресурсів. Складність у визначенні оптимального розподілу.
Алгоритми Локалізації Даних	Зниження затримок у доступі до даних. Покращення безпеки та конфіденційності даних. Зменшення мережевого трафіку.	Потребує ефективної стратегії розподілу даних. Потребує додаткових ресурсів для локалізації даних.
Алгоритми Енергозбереження	Зниження витрат енергії та вартості експлуатації. Сприяє сталому розвитку та екологічній ефективності. Можливість адаптації до змінного навантаження.	Може впливати на продуктивність при вимкненні ресурсів. Складність у визначенні оптимального рівня енергозбереження.
Алгоритми Оркестрації та Масштабування	Автоматизація та управління розгортанням ресурсів. Забезпечення масштабованості за потребою. Забезпечення високої доступності та стійкості.	Складність у реалізації та конфігурації. Ризик втрати продуктивності при неправильній конфігурації. Потребує адекватного моніторингу для ефективності.

Таке порівняння дозволяє краще зрозуміти переваги та недоліки кожного алгоритму у контексті управління ресурсами в хмарних обчисленнях.

3 РОЗРОБКА АЛГОРИТМІВ РОЗПОДІЛЕННЯ ВІРТУАЛЬНИХ МАШИН ЗА ХМАРНИМИ РЕСУРСАМИ

3.1 Огляд алгоритмів розв'язання проблем, пов'язаних із роботою.

Пакування контейнерів зазвичай являє собою складне завдання NP. Тому цей кваліфікаційний тест розділений на дві частини. Теоретичний підхід (використання CloudSim) і практичний підхід (реалізація в GCP).

Етапи реалізації цих підходів

Етап 1: моделювання алгоритмів на основі розроблених підходів на різних наборах даних у симуляторі CloudSim.

Етап 2: встановлення порогових значень і запуск скрипта-обгортки на C#.

Етап 3: локальне зберігання метрик вузлів. Зберігайте метрики вузлів як локальні (VM) і глобальні (Host) параметри в базі даних.

Крок 4. вибирається найкращий алгоритм пакування для розгортання ВМ у середовищі GCP.

Дуже важливо протестувати модель на реальних даних, щоб отримати відповідні результати. За замовчуванням симулятор CloudSim використовує алгоритм white-labelling для розміщення ВМ. Розширте клас PowerVmAllocationPolicyMigrationAbstract optimizeAllocation, щоб реалізувати алгоритм розміщення Bin у симуляторі CloudSim. Нелегко порівнювати всі алгоритми один з одним через змінні дані. Тому ми протестуємо цей алгоритм на різних наборах даних з використанням фіксованих параметрів. Основна мета цієї роботи - мінімізувати використання ресурсів PM без шкоди для SLA. Для досягнення цієї мети політика міграції під час розгортання фіксується на "мінімальному часі міграції".

У симуляторі CloudSim за замовчуванням увімкнено функцію відстеження робочого навантаження. Для обох кластерів Google параметри Bitbrain та Index відрізняються для обох кластерів Google. Тому набори даних кластера Google і Bitbrain необхідно перетворити в необхідний формат YAML. Після перетворення наборів даних вихідні файли надаються симулятору. Для того щоб порівняти

алгоритми, необхідно зафіксувати параметри конфігурації хоста і віртуальної машини.

3.2 Збір даних про робоче навантаження

3.2.1 Параметри кластера Google

Набір даних робочого навантаження кластера Google зібрано з обчислювального ресурсу Google (кластера екземплярів kubernetes). Цей кластер Google являє собою набір машин, упакованих у стійки і підключених до високошвидкісної кластерної мережі. Набір даних містить 29 днів відстеження робочого навантаження на 12 500 машинах. Нижче описано параметри, які містяться в наборі даних у таблиці подій завдання. Параметри в наборі даних:

- ідентифікатор;
- час старту;
- кількість ядер;
- використання процесора (МГц);
- використання оперативної пам'яті (КБ);
- доступна пам'ять (КБ);
- тип процесора, що використовується.

3.2.2 Характеристики кластера Bitbrains

Як базу даних використовували розподілений центр обробки даних компанії Bitbrains (голландський сервіс-провайдер), яка займається хостингом і бізнес-обчисленнями для підприємств (1 750 віртуальних машин). Відстеження.

Відстеження робочих навантажень у Bitbrain складається з двох частин: швидкого відстеження робочих навантажень і rnd-відстеження робочих навантажень. Швидке відстеження робочого навантаження зібрало 1250 віртуальних машин за один місяць, загалом 4057 ядер. Відстеження робочого навантаження Rnd, з іншого боку, зібрало дані про 500 віртуальних машин за три місяці, отримавши 1444 ядра. Параметри для кожної віртуальної машини зберігаються окремо у форматі .csv [8].

Параметри набору даних:

– маркер часу;
– ядра процесора;
– передбачена ємність центрального процесора;
– продуктивність центрального процесора (в мегагерцах);
– обсяг пам'яті (в кілобайтах);
– використана пам'ять (в кілобайтах);
– швидкість читання диска (в кілобайтах на секунду);
– швидкість запису на диск (в кілобайтах на секунду);
– отримана пропускна здатність мережі (в кілобайтах на секунду);
– пропускна здатність мережі (в кілобайтах на секунду).

3.2.3 Характеристики кластера Planet Lab

Набір трасувань використання віртуальних машин у лабораторії Planet (1000 віртуальних машин), зібраний за 10 випадкових днів у березні та квітні 2024 року. За замовчуванням моніторинг робочого навантаження в лабораторії ввімкнено в CloudSim. Дані містять характеристики використання процесора для тисяч віртуальних машин у більш ніж 500 місцях по всьому світу. У цьому наборі даних завантаження процесора кожної віртуальної машини вимірюється протягом п'яти хвилин.

Параметри в наборі даних:

- ідентифікатор;
- ядра центрального процесора;
- потужність центрального процесора;
- обсяг оперативної пам'яті (КБ);
- пропускна здатність (КБ/с);
- об'єм дискового простору (КБ).

3.2.4 Характеристики моделі генерації ресурсів

Цей набір даних було згенеровано додатком на базі мови C#. У цьому наборі даних завантаження процесора було згенеровано з постійними значеннями, такими як 50 % і 30 %; набір даних може бути згенеровано з такими параметрами, як оперативна пам'ять, ядра і потужність процесора.

Параметри в набір даних:

- ідентифікатор;
- ядра центрального процесора;
- потужність центрального процесора;
- обсяг оперативної пам'яті (КБ);
- пропускна здатність (КБ/с);
- обсяг дискового простору (КБ).

3.2.5 Характеристики сталої бази даних

Він також має ті самі параметри, що й постійна база даних. Набір даних генерує динамічні зміни в політиці використання з циклічними, випадковими, постійними та спадними патернами. Параметри набору даних включають ідентифікатор, ядра центрального процесора, потужність центрального процесора, оперативну пам'ять (КБ), пропускну здатність (КБ/с) і місткість диска (КБ).

4 РЕАЛІЗАЦІЯ ЕКСПЕРЕМЕНТІВ

4.1 Налаштування середовища GCP у експериментальному режимі

Експеримент щодо оцінки упаковки Bin установки віртуальних машин передбачав використання 5 вузлів, які були надані через платформу Mirantis. Специфікації кожного сервера були наступними:

- обчислювальні вузли;
- 1 вузол контролера;
- процесор з 8 ядрами;
- 16 ГБ оперативної пам'яті;
- операційна система Ubuntu 16.04 (Xenial);
- 256 ГБ пам'яті.

GCP - це дуже потужне хмарне середовище. Здебільшого воно підрозділяється на обчислювальні вузли, вузли контролера і мережеві вузли. У цьому практичному підході обчислювальні вузли нагадують фізичні машини, а всередині віртуальних екземплярів знаходяться віртуальні машини. Нарешті, віртуальні машини об'єднуються відповідно до алгоритму розгортання.

У цій реалізації використовуються два типи сховищ: на кожному обчислювальному вузлі розгорнуто сховище даних Datastore-1, яке може збирати дані про використання процесора для кожної віртуальної машини і для вузла гіпервізора в режимі реального часу. Інформація з цього сховища даних надходить у центральну базу даних для розрахунку показників продуктивності. Ця центральна база даних розташована на вузлі контролера GCP. Дані зберігаються у вигляді середнього мегагерца, споживаного кожною віртуальною машиною в останній момент часу t вимірювання. На практиці використання функції CPU C(t1, t2) для віртуальної машини описано нижче. де t1 - початковий час використання процесора віртуальною машиною, а t2 - кінцевий час. Це використання ЦП кожного вузла, що зберігається відповідно до часової мітки. Розрахунок виконується за такою формулою:

$$\overline{C_i^v}(t_1,t_2)=\frac{n_i^v F\left(\tau_i^v(t_2)-\tau_i^v(t_1)\right)}{t_2-t_1} \qquad (4.1)$$

Завантаження процесора C є функцією кількості віртуальних машин і часового інтервалу між t1 і t2 від кількості ядер n віртуальної машини, де F - частота одноядерної віртуальної машини, а τ - процесор, який використовується віртуальною машиною [9].

Жива міграція віртуальних машин - це міграція віртуальних машин з одного обчислювального вузла на інший без будь-якого простою. Така жива міграція віртуальних машин впливає на продуктивність програмних систем; Вільям [10] описує зниження продуктивності додатків і час простою. Очікуваний середній час простою [11] становить 10 % від завантаження процесора. Згідно із запропонованим алгоритмом, для усунення порушень SLA необхідно виконувати мінімальну міграцію віртуальних машин.

4.2 Розробка вигляду контейнера для клієнта GCP

GCP надає дуже широку підтримку в клієнтському API мовою C#. Для виконання поставленої задачі знадобились такі nuGet пакети:

- Google.Apis.Auth
- Google.Apis.Compute.v1,
- Google.Cloud.Logging.Log4Net.

Цей клауд провайдер підтримує операції міграції та обслуговування ресурсів [12] у середовищах GCP (див. рис. 4.1). Процес завантаження коду до хмари складається з трьох етапів.

Етап 1: Ініціалізація контейнера.

Етап 2: виконання алгоритмів прийняття рішень.

Етап 3: розгортання ВМ.

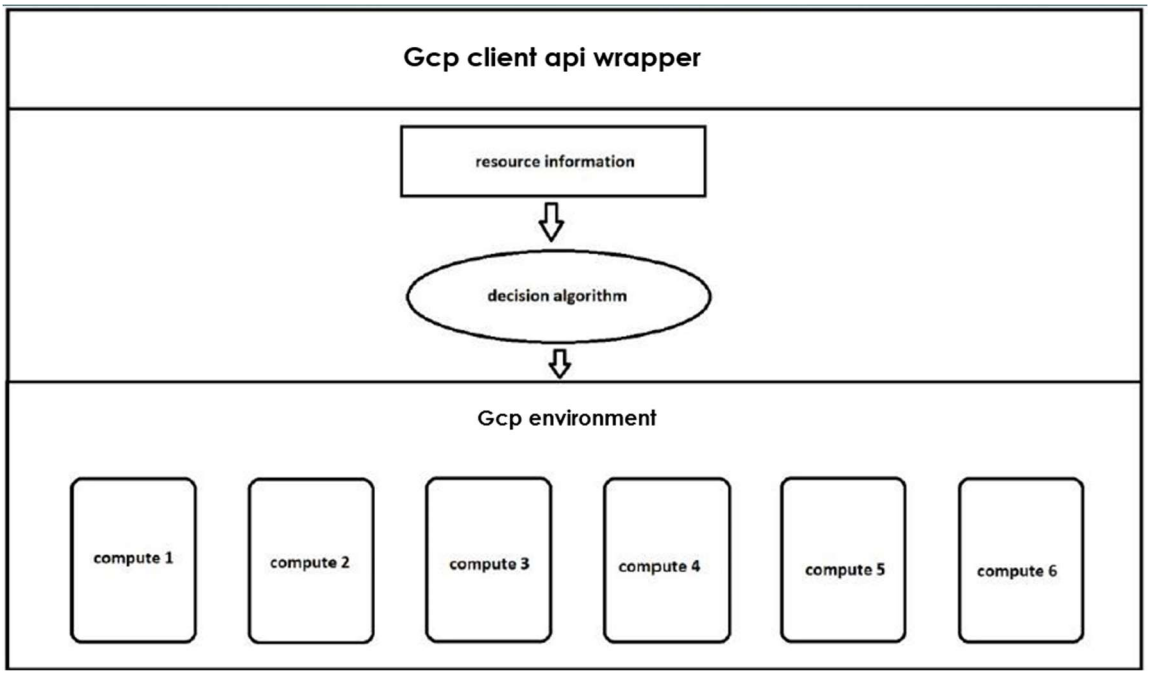

Рисунок 4.1 – Схема втілення GCP (рисунок створено самостійно)

Далі буде наведено більш детальний опис кожного етапу розгортання додатку в хмарі.

4.2.1 Ініціалізація обгортки

Обгортка здебільшого розроблена з використанням GCP і C# API. Для запуску екземпляра в середовищі GCP необхідно передати такі параметри, як ім'я образу, ім'я тому, ім'я ключа, ідентифікатор мережі та зона доступності обчислень. Зона доступності обчислень. За замовчуванням планувальник GCP Nova використовує циклічний алгоритм для розміщення екземплярів на обчислювальних вузлах. Тому перед запуском екземпляра в середовище GCP ця обгортка використовується для застосування внутрішніх розрахунків з використанням алгоритму прийняття рішень. Ця обгортка надає всю необхідну інформацію про ресурси алгоритму прийняття рішень.

4.2.2 Алгоритм прийняття рішень

Розрахувавши доступність ресурсів на кожному обчислювальному вузлі, цей алгоритм [13] ухвалення рішення видає ім'я обчислювального вузла, на якому буде запущено екземпляр.

Він надає ім'я вузла. У цій частині використовуються алгоритми First fit, Best Fit і Enhanced Best Fit.

4.2.3 Розміщення віртуальної машини

Існує два типи розташування ВМ [14]. До них належать.

– звичайне розміщення;

– порогове розгортання.

Звичайне розгортання виконується під час кожного запуску екземпляра. Порогові значення для кожного обчислювального вузла задаються користувачем (наприклад, максимальне використання процесора, пам'яті тощо). Коли використання вузла перевищує порогове значення, автоматично виконується сценарій для живої міграції екземпляра між обчислювальними вузлами; для вибору нового вузла для віртуальної машини використовується алгоритм пакування Bin.

Налаштування BIOS для кожного ПМ мають відповідати технології Wake-on-LAN. Це дає змогу керувати параметрами живлення обчислювального вузла.

Після завершення розгортання ВМ сценарій автоматично відключає невикористовувані обчислювальні вузли, використовуючи їхні MAC-адреси.

Контейнер інтегрується з Cloud Monitor для отримання уявлення про використання ресурсів у середовищі GCP у режимі реального часу.

4.3 Монітор GCP

GCP Monitor насамперед призначений для забезпечення видимості стану хмарного середовища в реальному часі. Інструмент було розроблено з використанням GCP Client API і клієнта OpenTelemetry; OpenTelemetryDB - це

база даних часових рядів, що означає, що всі метрики можна ефективно зберігати з використанням тимчасових міток (див. рис. 4.2).

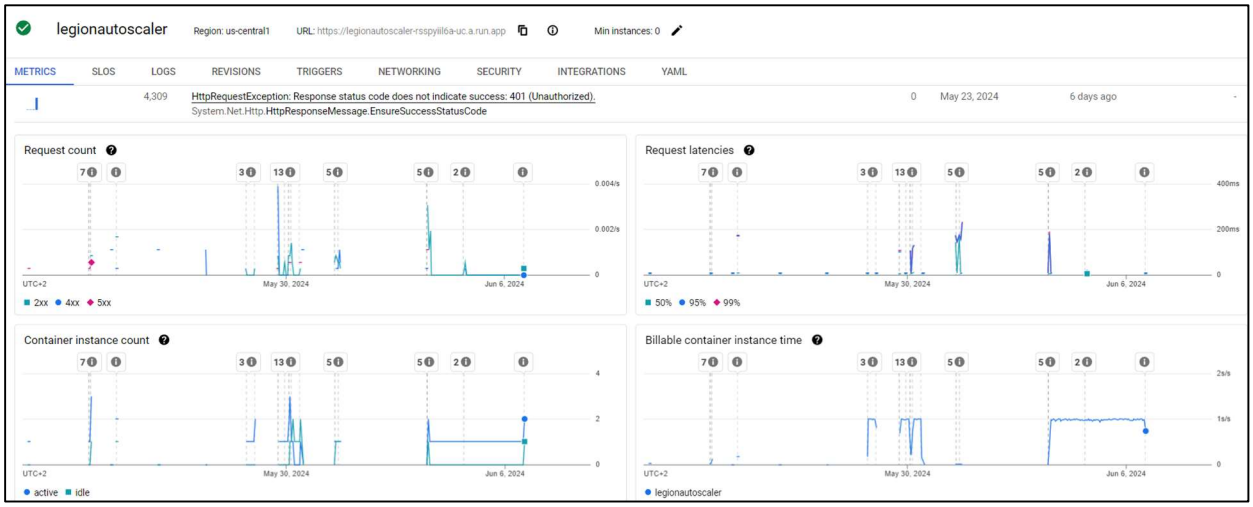

Рисунок 4.2 – Візуалізація дисплею панелі Cloud Monitor (рисунок створено самостійно)

Знання інформації про всі метрики дає змогу зрозуміти, що саме дає збій під час тестування. Інструмент завантажує всі дані метрик із бази даних -2 (обчислювальний вузол) у клієнт OpenTelemetry.

Основними функціями цього монітора GCP є:

– використання процесора обчислювальних вузлів і контролерів;
– кількість VCPU, розгорнутих на вузлах; і
– дисковий простір, що використовується кожним вузлом;
– використання пам'яті вузла.

4.4 Алгоритми

Ефективність пакування [15] постійно змінюється залежно від розмірів контейнерів. Тому алгоритми пакування мають бути протестовані зі змінним об'ємом контейнерів за коротший час із високою ефективністю.

$$PackingEfficiency = \frac{sum\ of\ VM\ resources}{Used\ PM\ resources} \tag{4.2}$$

де sum of VM resources – це сума ресурсів віртуальних машин, а used PM resources – використані ресурси реальної машини.

Алгоритми пакування контейнерів описані таким чином:

– алгоритм, що скорочує першу відповідність та вибирає перший елемент;

– найбільш підходящий є алгоритм скорочення;

– покращений варіант найкращого підбору.

4.4.1 Алгоритм першої відповідності

Алгоритм FIFO (першої відповідності) вважається одним із найстаріших управлінських алгоритмів для роботи з пам'яттю. Реалізація цього алгоритму показана на малюнку 4.3. Він розміщує контейнери в порядку, в якому вони надходять (FIFO - "перший прийшов, перший обслуговується").

Algorithm 1: First fit decreasing algorithm

```
Input: Host list, VM list
Output: map_to_VM
1. Sort VM list in order of decreasing CPU utilization
2. For VM in VM list do
3.    for host in Host list do
4.       if host has enough resources to VM then
5.          map_host ← Host
6.    if  map_host != NULL then
7.       add (map_host, VM) to map_to_VM
8.    return map  to  VM
```

Рисунок 4.3 – Алгоритм першої відповідності (за даними [14])

Цей алгоритм може бути розділений на два типи:

– перша підгонка;

– перша підгонка зменшується.

4.4.2 Алгоритм зменшення навантаження

В алгоритмі first-reduce усі завдання спочатку сортуються відповідно до розміру завдання (див. рис. 4.4). Тому в більшості випадків він забезпечує вищу ефективність пакування, ніж звичайний алгоритм first-fit.

```
Algorithm 2: Best fit decreasing algorithm
Input: Host list, VM list
Output: map_to_VM
1. Sort VM list in order of decreasing CPU utilization
2. For VM in VM list do
3.    for host in Host list do
4.       if (0 < host utilization > threshold (host, VM))
5.          power _difference ← power (host, VM) - power (host)
6.          If power _difference > max(power) then
7.             map_host ← host
8.             max(power) ← power _difference
9.    if map_host != NULL then
10.       add (map_host, VM) to map_to_VM
11.   return map_to_VM
```

Рисунок 4.4 – Алгоритм зменшення навантаження (за даними [14])

Обидва алгоритми схожі, за винятком сортування списку ВМ на початковому кроці.

4.4.3 Розширений алгоритм найкращої відповідності

Цей алгоритм мало чим відрізняється від алгоритму зменшення продуктивності (див. рис. 4.5). В алгоритмі найкращого скорочення ресурси ВМ спочатку сортуються, а завдання розміщуються відповідно до різниці в потужності між ресурсами після міграції ВМ і до міграції хоста ВМ.

```
Algorithm 3: Enhanced Best fit algorithm
Input: Host list, VM list
Output: map_to_VM
1. Sort VM list in order of decreasing CPU utilization
2. For VM in VM list do
3. Sort host list in order of decreasing CPU utilization
4.    for host in Host list do
5.       if (0 < host utilization > threshold (host, VM))
6.             map_host ← host
7.    if map_host ! = NULL then
8.       add (map_host, VM) to map_to_VM
9.    return map_to_VM
```

Рисунок 4.5 – Розширений алгоритм найкращої відповідності (за даними [14])

Однак у цьому підході за ітерацію сортуються як ВМ, так і хости.

4.5 Результати моделювання та експерементів

У даному розділі потрібно здійснити порівняння різних параметрів для алгоритмів "Найкращий підхід", "Зменшення найкращого підходу", "Зменшення першої підгонки" та "Перша підгонка" з використанням набору даних Planet lab, Google compute, Bitbrain і Raw. Таким чином, ми провели моделювання 16 разів, щоб охопити всі наші тестові сценарії. Порівнюємо показники для алгоритмів розміщення:

- споживання енергії;
- кількість міграцій ВМ;
- кількість вимкнених хостів;
- середній час міграції;
- порушення SLA.

Споживання енергії залежить від того, скільки потужності обчислювальний модуль (РМ) використовує під час роботи, і воно також змінюється залежно від використання ресурсів РМ. З рисунку 4.6 видно, що покращений алгоритм найкращої відповідності дає найкращі результати у всіх тестових випадках, оскільки ми сортуємо ресурси хоста та ресурси віртуальної машини на кожній ітерації.

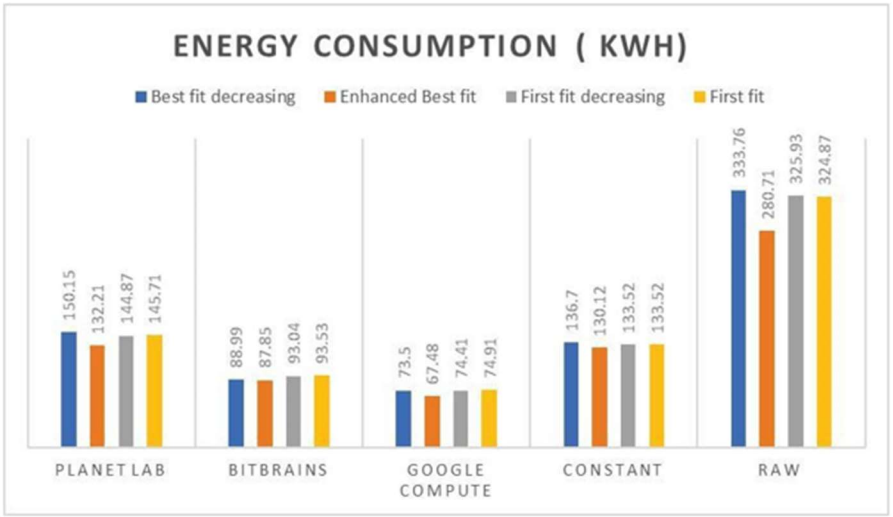

Рисунок 4.6 – Порівняння використання енергії (у кіловат-годинах) (рисунок створено самостійно)

У наведеній таблиці 4.1 подано середні значення та довірчі інтервали для кожного набору даних.

Таблиця 4.1 – Результати експериментів: порівняння витраченої енергії. (створено самостійно)

Набір даних	Середні витрати (kWh)	Стандартне відхилення	95% ДІ
Planet Lab	143,24	7.71	135 до 151
Bitbrains	90,85	2,85	88.1 до 93.7
Google compute	72,58	3.45	69,2 до 76
Постійний набір даних	133,47	2.69	130-136
Набір необроблених даних	316,32	24.1	292 до 340

Менший обсяг міграцій віртуальних машин вказує на більш ефективну консолідацію, менше трафіку в хмарній мережі та менше порушення рівня обслуговування для міграції віртуальних машин. У цьому сценарії вдосконалений алгоритм найкращого підходу забезпечує кращі результати порівняно з іншими алгоритмами, за винятком постійної бази даних. У випадку постійного набору даних зменшення найкращої відповідності приводить до кращих результатів.

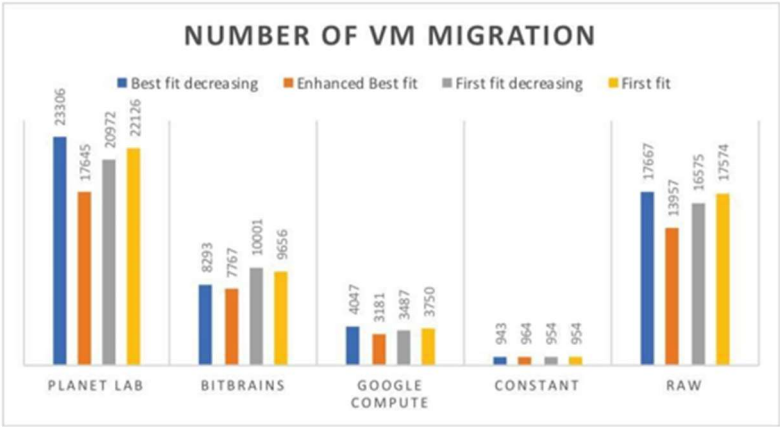

Рисунок 4.7 – Порівняння кількості міграцій віртуальних машин (рисунок створено самостійно)

У таблиці 4.2 наведено середні значення та довірчі інтервали для кожного набору даних.

Таблиця 4.2 – Результати експериментів: кількість міграцій (створено самостійно)

Набір даних	Середній	Стандартне відхилення	95% ДІ
Planet Lab	21012.25	2440	18600 до 23400
Bitbrains	8929,25	1070	7880 до 9980
Google compute	3616,25	369	3260 до 3980
Постійний набір даних	953,75	8.59	946 до 962
Набір необроблених даних	16443,25	1730	14700 до 18100

При зміні вимкнених хостів мінімальна кількість вимкнень відноситься до оптимального алгоритму. Як видно з рисунку 4.8, кількість вимкнених хостів, які найкраще підходять, значно зменшує кількість методів. Це свідчить про те, що коли хост переходить у сплячий режим, він залишається там протягом тривалого часу, що підтверджує ефективність наших алгоритмів.

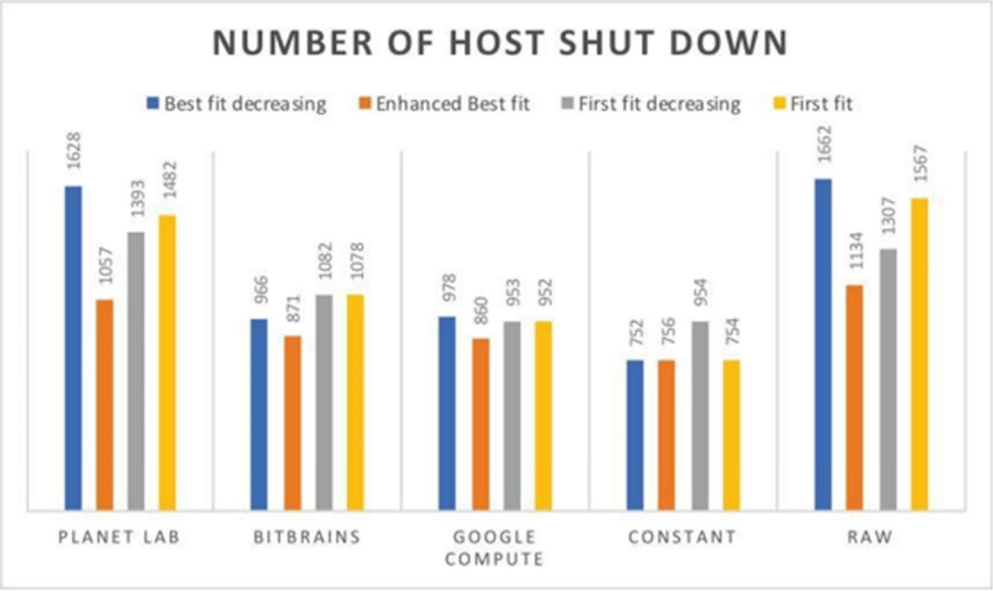

Рисунок 4.8 – Порівняння кількості вимкнених серверів. (рисунок створено самостійно)

У цьому сценарії вдосконалений алгоритм найкращого підходу показує кращі результати, ніж інші алгоритми, за винятком баз даних із постійним завантаженням. У випадку постійного набору даних алгоритм першого підбору демонструє кращу ефективність. Середнє порушення SLA є важливим показником для оцінки ефективності упаковки в PM. Як вже обговорювалося, мінімальне порушення SLA визначається як ефективний алгоритм. Таким чином, у цій симуляції ми використовуємо мінімальний час міграції як бажану політику розподілу в CloudSim. Результати цього розділу майже не відрізняються від інших алгоритмів (рисунок 4.9).

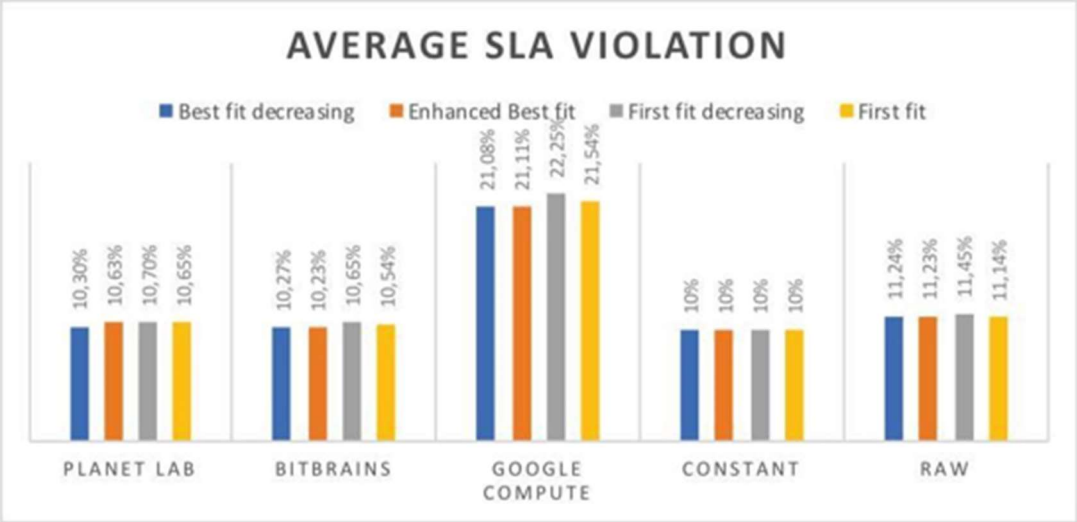

Рисунок 4.9 – Порівняння середніх відхилень від рівня обслуговування (SLA) (рисунок створено самостійно)

У наведеній нижче таблиці подані середні значення та інтервали надійності для кожного набору даних (таблиця 4.3).

Таблиця 4.3 – Дані експериментів: порушення SLA (створено самостійно)

Набір даних	Середній	Стандартне Відхилення	95% ДІ
Planet Lab	10.6	0,182	10.4-10.8
Bitbrains	10.4	0,205	10.2-10.6
Google compute	21.5	0,545	21-22

кінець таблиці 4.3

Постійний набір даних	10	0	10 до 10
Набір необроблених даних	11.3	0,131	11.2-11.4

Час, необхідний для виконання, називається часом моделювання для розміщення всіх віртуальних машин. У цьому випадку найкращим варіантом буде вказати мінімальний час виконання, оскільки він закінчує процес розміщення всіх операцій міграції віртуальної машини в Datacenter (див. рис. 4.10).

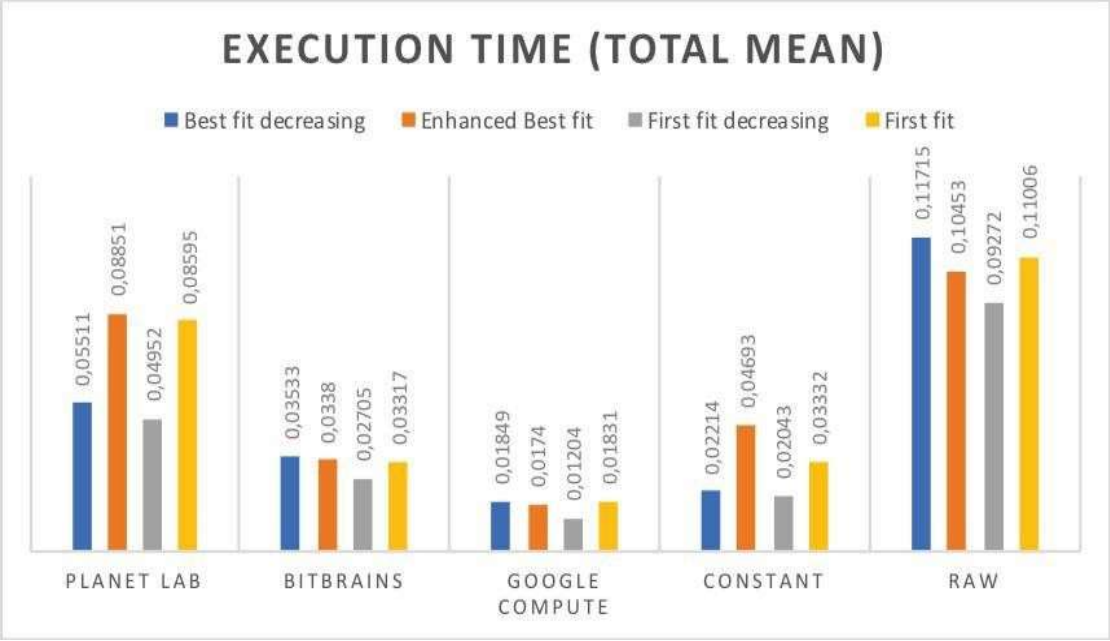

Рисунок 4.10 – Оцінка часу виконання (у секундах) (рисунок створено самостійно)

У цьому варіанті розширений метод скорочення першого відповідності показує кращу ефективність у порівнянні з іншими методами.

ВИСНОВКИ

В результаті роботи над кваліфікаційною роботою було виконано всі поставлені задачі, а саме:

- проведено огляд літератури та існуючих підходів до управління ресурсами у хмарних обчисленнях;
- вивчено та проаналізувано основні алгоритми розподілу ресурсів, локалізації даних, енергозбереження та оркестрації та масштабування;
- встановлено критерії оцінки ефективності кожного алгоритму;
- проведено експериментальне дослідження з використанням реальних або симулюваних даних для розробки алгоритму автоматичного розподілення навантаження системи;
- підготувано звіт із результатами дослідження та сформулювано рекомендації для вибору оптимального алгоритму управління ресурсами в хмарних обчисленнях.

Для практичного підходу було використано платформу GCP та розроблено C#-оболонку для інтеграції цієї моделі в середовище GCP за допомогою C#-клієнта. Наша програма забезпечує переміщення операцій віртуальної машини між обчислювальними вузлами. Для вибору обчислювальних вузлів було використано вдосконалений алгоритм оптимізації.

Очікувані результати дослідження дозволили отримати глибше розуміння різних алгоритмів управління ресурсами в хмарних обчисленнях та їх застосування в різних сценаріях. Результати дослідження зможуть надати корисні вказівки та рекомендації для вибору оптимального підходу до управління ресурсами в хмарному середовищі залежно від конкретних вимог та умов проекту.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Understanding the basics of cloud computing URL - https://www.lucidchart.com/blog/cloud-computing-basics (дата звернення: 02.05.2024).

2. Types of cloud computing URL - https://www.redhat.com/en/topics/cloud-computing/public-cloud-vs-private-cloud-and-hybrid-cloud (дата звернення: 02.05.2024).

3. Яка різниця між CRM у моделі SaaS і CRM у хмарі? URL - https://evolpe.com.ua/ (дата звернення: 02.05.2024).

4. Advantages and Disadvantages of Cloud Computing URL - https://cloud.google.com/learn/advantages-of-cloud-computing#:~:text=The%20cloud%20delivers%20more%20flexibility,use%20cases%20into%20their%20strategies (дата звернення: 02.05.2024).

5. Top cloud providers: AWS, Microsoft Azure, and Google Cloud, hybrid, SaaS players URL - https://www.zdnet.com/article/the-top-cloud-providers-of-2021-aws-microsoft-azure-google-cloud-hybrid-saas/ (дата звернення: 02.05.2024).

6. Порівняння табличка: AWS vs Azure vs GCP URL - https://smartnet.ua/porivnjannja-tablichka-aws-vs-azure-vs-gcp/ (дата звернення: 04.05.2024).

7. Resource Management Models in Cloud Computing URL - https://www.geeksforgeeks.org/resource-management-models-in-cloud-computing/ (дата звернення: 04.05.2024).

8. Волк М. О., Саранча С. М., Гвоздецький Д. П., Ольшанська Т. І. Методи розподілення віртуальних машин за хмарними ресурсами. XII міжнародна науково-технічна конференція “Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління”. Баку, Харків, Жиліна. Том 1., 27-28 квітня 2022р., с. 70

9. M. Mishra and A. Sahoo, “On Theory of VM Placement: Anomalies in Existing Methodologies and Their Mitigation Using a Novel Vector Based Approach,” in 2011 IEEE 4th International Conference on Cloud Computing, 2011, pp. 275–282.

10. M. Mishra and U. Bellur, “Whither Tightness of Packing? The Case for Stable VM Placement,” IEEE Transactions on Cloud Computing, vol. 4, no. 4, pp. 481–494, Oct. 2016.

11. Bezsmertnyi, O., Golian, N., Golian, V., Afanasieva, I., Behavior Driven Development Approach in the Modern Quality Control Process 2020 IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020 - Proceedings, 2021, стр. 217–220, 9467891 DOI 10.1109/PICST51311.2020.9467891

12. Alkilani M., Kobziev V., Enhancing E-government Services by Using Cloud Computing / CEUR Workshop Proceedings, 2019, 2683, pp. 66-69.

13. Yerokhin, A., Semenets, V., Nechyporenko, A., Turuta, O., Babii, A., F-transform 3D Point Cloud Filtering Algorithm Yerokhin, A., Semenets, V., Nechyporenko, A., Turuta, O., Babii, A. Proceedings of the 2018 IEEE 2nd International Conference on Data Stream Mining and Processing, DSMP 2018, 2018, стр. 524–527, 8478581 DOI: 10.1109/DSMP.2018.8478581

14. A. Beloglazov and R. Buyya, “Managing Overloaded Hosts for Dynamic Consolidation of Virtual Machines in Cloud Data Centers under Quality of Service Constraints,” IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 7, pp. 1366–1379, Jul. 2013.

15. A. Beloglazov and R. Buyya, “Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers,” Concurrency Computat.: Pract. Exper., vol. 24, no. 13, pp. 1397–1420, Sep. 2012.