Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Методи управління ресурсами в хмарних системах

(тема)

Виконав:
студент II курсу, групи СПм-23 - 1
Зборовський М.М.
(прізвище, ініціали)

Спеціальність
123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Системне програмування
(повна назва освітньої програми)

Керівник: проф. Волк М.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп'ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Зборовському Михайлу Михайловичу
(прізвище, ім'я, по батькові)

1. Тема роботи Методи управління ресурсами в хмарних системах

затверджена наказом по університету від " 22 " листопада 2024 р. № 1236ст

2. Термін подання студентом роботи до екзаменаційної комісії 20 січня 2025 р.

3. Вхідні дані до роботи

1. Технології управління ресурсами в хмарних системах
2. Технологія CloudOps.
3. Методи управління ресурсами в хмарних системах

4. Перелік питань, що потрібно опрацювати в роботі

1 Аналіз предметної області
2 Методи управління ресурсами в хмарних системах
3 Розгортання та дослідження
4 Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Демонстраційні матеріали. Плакати – 12 арк. ф. А4

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної області	25.11.24-27.11.24	
2	Розробка моделей	28.11.24-04.12.24	
3	Реалізація алгоритмів	05.12.24-15.12/24	
4	Розробка структури програмних засобів	16.12.24-25.12.24	
5	Розробка програмних модулів	25.12.24-28.12.24	
6	Оформлення матеріалів кваліфікаційної роботи	29.12.24-03.01.25	
7	Подання кваліфікаційної роботи керівникові та попередній захист	04.01.25-07.01.25	
8	Подання кваліфікаційної роботи на рецензування	08.01.25-20.01.25	

Дата видачі завдання 25 листопада 2024 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Волк М.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 56 с., 15 рис., 2 табл., 1 дод., 27 джерел.

ХМАРНІ СИСТЕМИ, МЕТОДИ РОЗПОДІЛЕННЯ РЕСУРСІВ, ВИСОКОПРОДУКТИВНІ ОБЧИСЛЕННЯ

У кваліфікаційній роботі досліджено методи управління ресурсами в хмарних системах з акцентом на оптимізацію обчислювальних процесів і автоматизацію операцій. Основна мета дослідження полягає в розробці ефективних підходів до управління ресурсами, які забезпечать надійність, масштабованість та мінімізацію енергоспоживання в умовах зростаючого навантаження на інформаційні системи.

У роботі описано сучасні архітектурні рішення, що дозволяють впроваджувати хмарні технології та використовувати багаторівневі стратегії для оптимізації роботи систем. Запропоновано нові підходи до управління віртуальними машинами та розподілу ресурсів, які ґрунтуються на принципах балансування навантаження та динамічного масштабування ресурсів відповідно до обчислювальних потреб.

Експериментальне моделювання показало високу ефективність запропонованих підходів у реальних хмарних системах, зокрема в умовах різного навантаження та обмеженості ресурсів. Результати симуляцій продемонстрували, що запропоновані алгоритми розподілу ресурсів знижують затримки, покращують продуктивність та зменшують витрати енергії.

Отримані результати підтверджують, що запропоновані методи та архітектурні рішення можуть бути використані для розгортання масштабованих хмарних систем, які забезпечують високу ефективність та надійність.

ABSTRACT

Master’s thesis: 56 pages, 15 figures, 2 tables, 1 appendice, 27 sources.

CLOUD SYSTEMS, RESOURCE ALLOCATION METHODS, HIGH-PERFORMANCE COMPUTING

The qualification work investigates methods of resource management in cloud systems with an emphasis on optimization of computing processes and automation of operations. The main goal of the research is to develop effective approaches to resource management that will ensure reliability and scalability and minimize energy consumption in conditions of increasing load on information systems.

The work describes modern architectural solutions that allow the implementation of cloud technologies and use multi-level strategies to optimize system operation. New approaches to virtual machine management and resource allocation are proposed, which are based on the principles of load balancing and dynamic scaling of resources in accordance with computing needs.

Experimental modelling has shown the high efficiency of the proposed approaches in real cloud systems, in particular under conditions of different load and resource limitations. The simulation results have demonstrated that the proposed resource allocation algorithms reduce delays, improve performance and reduce energy consumption.

The results obtained confirm that the proposed methods and architectural solutions can be used to deploy scalable cloud systems that provide high efficiency and reliability.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ВМ – віртуальна машина

ОС – операційна система

ПЗ – програмне забезпечення

AKS – Azure Kubernetes Service

API – Application Programming Interfase

DIS – Distributed Interactive Simulation

DEVS – Discrete Event System Specification

IaaS – Infrastructure as a Service

EKS – Elastic Kubernetes Service

ID – identity

FCFS – First-Come First-Served

FPGA – Field-Programmable Gate Array

GKE – Google Kubernetes Engine

HAL – Hardware Abstraction Layer

HLA – High Level Architecture

HPC – High Performance Computing

HPF – Highest Priority First

M&S – Modelling and Simulation

SaaS – Simulation as a Service

VM – Virtual Machine

VPN – Virtual Private Network

ВСТУП

Для ефективної архітектури системи управління хмарними ресурсами будь-якої хмарної системи дуже важливо визначити її обмеження, які необхідно подолати в процесі проєктування. Зазвичай, у цій парадигмі є множина центрів обробки даних, кожен зі своїм набором ресурсів. Лише один центр обробки даних повинен бути обраний для виконання кожної роботи протягом усього обчислювального процесу.

Завдання ускладняється, якщо ми маємо декілька завдань, які повинні одночасно виконуватися. Важливо знайти віртуальну машину, яка може впоратися з робочим навантаженням, не порушуючи умови з боку швидкості процесора, розміру пам’яті, місця для зберігання чи будь-якого іншого критерію, який може визначати стратегії управління.

Завдання не є превентивними, тобто після того, як вони призначені для віртуальної машини, очікується, що вони виконуватимуться повністю на цій віртуальній машині. Оскільки передбачається, що кожна віртуальна машина має одне ядро, конфлікт між запущеними процесами уникається, і віртуальна машина може зосереджуватися на одному завданні за раз. Усі процеси, що виконуються на даному ресурсі, не повинні потребувати більшої обчислювальної потужності, ніж може надати ресурс. Усі завдання, які виконуються на певному ресурсі, не повинні використовувати більше пам’яті, ніж дозволено для цього ресурсу.

Коли для хостів більше немає роботи, деякі з них буде вимкнено, щоб заощадити енергію, тому загальна кількість хостів і віртуальних машин не є фіксованою. Максимальна кількість віртуальних машин, які одночасно можуть бути розміщені на даному хості, обмежена його обчислювальною потужністю, яка сама по собі визначається кількістю доступних ядер ЦП і припущенням, що кожне ядро може розмістити лише одну віртуальну машину. Метод планування розглядає кожну роботу так, ніби вона була

повністю автономною, і вважає, що між ними не буде координації.

Так звана автоматизована робота та управління ресурсами стосуються великої кількості завдань, що виконуються у щоденних IT-операціях (від простих щоденних перевірок, змін конфігурації та інсталяції програмного забезпечення до організаційного планування всього процесу змін). Зараз спостерігається перехід від ручного виконання в минулому до стандартизованого, спрощеного і автоматизованого процесу під керуванням операцій.

Робота спрямована на оптимізацію корпоративних хмарних ресурсів і керування ними, використовуючи автоматизовані операції (autoOps) як фундаментальну стратегію. Оскільки галузі застосування інформаційних систем мають експоненціальне зростання та постійно з'являються інновації в IT-системах, складність управління ресурсами зростає.

Автоматизовані операції стали критично важливим компонентом, переходячи від ручного втручання до стандартизації, оптимізації робочого процесу та вдосконалення архітектури.

Завдяки розгортанню в реальних умовах і теоретичним основам, вони створюють ефективні стратегії оптимізації та управління хмарними ресурсами, тим самим підвищуючи ефективність, безпеку та стійкість інформаційних систем.

Кваліфікаційна робота організована таким чином. Аналіз предметної області наведено у розділі 1. Розділ 2 представляє опис методів, що лежать в основі роботи, детальний огляд процесу управління в хмарних системах. У розділі 3 представлено процес розгортання розробленого програмного забезпечення та експериментальні дослідження.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальна архітектура систем управління ресурсами в хмарних системах

Пропонується наступна архітектура, яка містить чотири основні модуля – агент споживача (Consumer Agents), агент брокера (Broker Agent), модуль менеджера (Manager Module) та модуль постачальника (Provider Modules). Вони утворюють базову архітектуру запропонованої моделі. Кожен із цих основних компонентів представляє окремий рівень у стеку хмарних обчислень і відповідає за окреме завдання, пов’язане з керуванням доступними ресурсами. На рівні SaaS Consumer Agents відповідають за взаємодію з користувачем.

Модуль менеджера рівня PaaS відповідає за вищезгадані функції узгодження та керування моделлю як для кінцевих користувачів, так і для основного постачальника інфраструктури. На рівні IaaS модуль постачальника контролює як віртуальне, так і фізичне обладнання хмари.

Нижче наведено детальний опис кожного модуля в цьому дизайні разом із відповідними ролями (рисунок 1.1). Consumer Agent – це агент на стороні споживача, який служить інтерфейсом, через який споживач подає завдання. Коли клієнти роблять запити, вони вказують параметри, зокрема скільки грошей вони можуть витратити, наскільки швидкими мають бути їхні комп’ютери, скільки місця їм потрібно для зберігання даних і коли вони їм потрібні. Потім Consumer Agent повідомляє Агента Брокера про невиконані обов’язки та будь-які передумови.

Для полегшення зв’язку між агентом споживача та модулем менеджера існує брокерський агент. Consumer Agent надає завдання Брокеру, а Брокер потім перетворює ці завдання на пропозиції перед тим, як надсилати їх до Модулю менеджера.

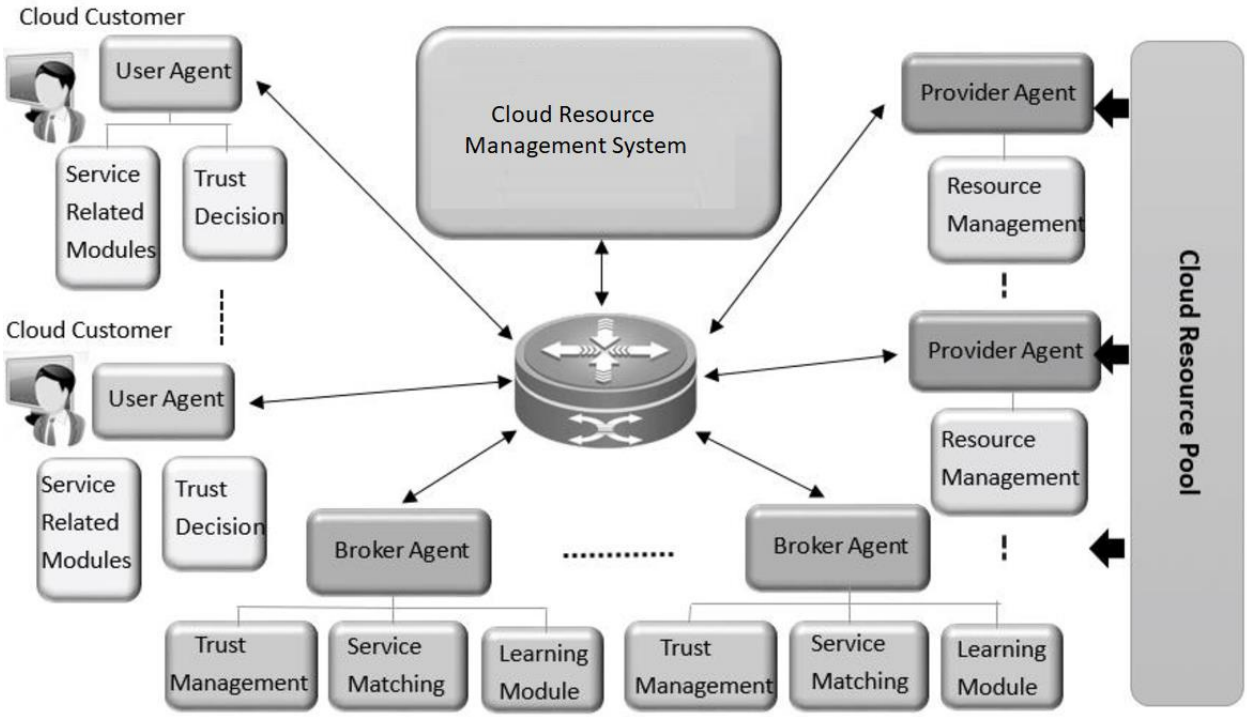

Рисунок 1.1 – Архітектура системи

Брокерський агент приймає завдання, надіслані споживачами, і використовує ці запити, щоб визначити, який рівень якості обслуговування (QoS) потрібен. Агент-споживач, який піклується про хід кожної роботи, отримує результати коментарів та результатів від Агента брокера.

Важливою частиною запропонованого дизайну є менеджерський модуль. Він отримує дані від агенту Manager у модулі Cloud Provider і відповідно розподіляє ці ресурси. Для досягнення цілей запропонованої парадигми необхідна взаємодія між трьома підмодулями. Допоміжні компоненти менеджера включають агента моніторингу угоди про рівень обслуговування (SLA), агента узгодження та агента диспетчера завдань. Менеджер приймає ставки від агента менеджера у кожному центрі обробки даних і узгоджує угоди про рівень обслуговування з агентом-брокером через агента-переговорника за угодою SLA. На цьому етапі фокус модуля зміщується на фактичний процес переговорів, який ґрунтується на заздалегідь визначених цілях постачальників і клієнтів. Він також контролює процес виконання роботи споживача в центрах обробки даних хмарного

провайдера. Від агентів Manager він дізнається, як просувається кожна робота. Детальніше про спеціалізовані модулі Менеджера наведено нижче.

Агент з переговорів щодо рівня обслуговування (SLA Negotiator) відповідає за отримання всієї інформації, яку він збирає від агента-брокера та агентів-керівників, і узгодження умов обслуговування для всіх центрів обробки даних. Він починає переговори, вибираючи найкращу можливу пропозицію для кожної пропозиції, враховуючи наявний час і гроші. Він визначає, який центр обробки даних здатний виконувати операцію відповідно до викладених критеріїв. Процес зіставлення між кінцевими користувачами та центрами обробки даних здійснюється за допомогою паралельного узгодження PSO.

Після цього він складає договір і підписує його як клієнтом, так і постачальником послуг.

Після того, як агент узгодження SLA визначає ідентифікатор центру обробки даних для кожної роботи, завдання диспетчера завдань полягає в тому, щоб розподілити ці завдання відповідному об’єкту.

Робота агента моніторингу угоди про рівень обслуговування (SLA) полягає в тому, щоб збирати дані про SLA та стежити за нею в разі порушення. Він стежить за статусом усіх надісланих завдань і позначає будь-які потенційні порушення угоди про рівень обслуговування (SLA). У цій концепції пропуск строку вважається порушенням.

Модуль постачальника хмарних послуг, який виступає за інфраструктуру як сервісний рівень хмари, який складається з віртуальних машин і хостів. Щоб перевірити нашу модель із великою кількістю центрів обробки даних, ми припускаємо, що вона складається з кількох розосереджених центрів обробки даних – в роботі 10. Агент Manager, агент монітора хосту, планувальник завдань, агент балансування навантаження та агент менеджера віртуальної машини п’ять частин, які складають центр обробки даних, який є централізованим сховищем фізичних і віртуальних ресурсів.

Агент диспетчера центру обробки даних зв’язується між апаратним і програмним забезпеченням і диспетчерським модулем. У кожного центру обробки даних є свій «локальний менеджер». Агент менеджера DC кожного центру обробки даних зв’язуватиметься з агентом менеджера, щоб надавати пропозиції, які відповідають вимогам QoS споживача. Цей агент має кілька цілей; наприклад, він через регулярні проміжки часу передає інформацію про поточний стан хмари від агента моніторингу хосту до агента менеджера та отримує список дій, які потрібно виконати з модуля менеджера.

Агент HM або монітор хосту: він відстежує, наскільки зайнятий кожен хост, і попереджає про міграцію віртуальної машини, якщо робоче навантаження розподіляється нерівномірно. Відстежуються хости центру обробки даних і віртуальні машини (ВМ), а також такі показники, як використання ресурсів і витрати на енергію. Планувальник VM покладається на дані, надані цим модулем, для належного призначення хостів віртуальним машинам.

1.2 Хмарні системи та розподіл ресурсів

Хмарні обчислення – це глобальна платформа для високорозподіленої онлайн-обробки, комунікації та функцій керування даними. Хмарні обчислення забезпечують оперативні засоби онлайн; отже, Інтернет є засобом доступу до платформ хмарних обчислень. Організації, які керують хмарними платформами, називаються постачальниками хмарних послуг (CSP). CSP надають кінцевим користувачам різноманітні послуги від програмного забезпечення та проміжного ПЗ до високоскладної інфраструктури [1].

У наш час багато користувачів використовують хмарні обчислення, тому багато користувачів користуються все більшою кількістю його послуг і програм, де б вони не були. Хмарні обчислення також є системою резервного копіювання, тому будь-які файли з важливою інформацією можуть мати

резервні копії в хмарній мережі. Хмарні обчислення можуть обмінюватися файлами в будь-якій точці світу; користувачі можуть обмінюватися файлами швидко та безпечно без будь-яких вагань чи збоїв. Хмарні обчислення також мають резервну копію прикладного програмного забезпечення, щоб ми могли довірити його на випадок збою будь-якого програмного забезпечення, оскільки у нас є його резервні файли [2]. Хмарні обчислення можуть бути публічними та приватними. Публічна хмара відкрита для всіх користувачів, і постачальники публічних хмарних послуг надають її безкоштовно в Інтернеті. Постачальники приватної хмари надають доступ до приватної хмари певній кількості осіб [3]. Ці служби є системою мереж, які надають розміщені послуги.

З розвитком хмарних обчислень, великих даних і штучного інтелекту попит на хмарні ресурси швидко зростає, особливо на певні незрозумілі та нові потреби в ресурсах. Емерджентний режим не підтримується звичайними методами розподілу ресурсів у хмарі для своєчасного й оптимального розподілу ресурсів [4].

У службах багатохмарного середовища однією з критичних проблем безпеки є авторизація та автентифікація. У сценаріях авторизації ідентифікуйте користувача як такого, що має повноваження переглядати цю частину служби програми, а сценарії автентифікації визначають, чи є хмарний користувач автентичним. Різні програми мають різні життєві цикли та політики для використання методів автентифікації та авторизації. Більшість викликів безпеці гарантували, що правильні та справжні користувачі можуть отримати доступ до вашої хмарної програми [5].

У багатохмарному середовищі моніторинг є критичним викликом для безпеки та керування. Коли дані клієнтів зберігаються в хмарі, моніторинг є технікою, яка гарантує безпеку даних клієнтів. Хмарні послуги пропонують багато переваг для бізнес-використання, а інтерфейс програмування веб-додатків (API) підвищує ризики безпеки під час розгортання хмарних служб. Таким чином, більшість організацій хвилюються щодо зберігання особистих

даних у хмарі. Моніторинг дозволяє організаціям збалансувати ризики, скористатися перевагами хмари та зменшити складність критичних завдань [6].

У багатохмарному середовищі працювати зі сховищем завжди важко. При розробці хмарної системи необхідно пам’ятати, що безпека має важливе значення, що дозволяє зберігати дані з кількома хмарами. Більшість користувачів зберігають свої дані в хмарі замість того, щоб зберігати їх на локальних пристроях або жорстких дисках. Центри обробки даних піклуються про дані користувачів і забезпечують їх безпеку [7].

Захист системи – це техніка, яка використовується для зменшення ризиків безпеки та усунення атак на безпеку та поверхні системних атак. Аплікаційне зміцнення - це тип системного зміцнення. Захист програми надає можливість обмежити доступ користувача до програми на основі керування ролями. Під час зміцнення програми видаліть усі непотрібні функції. Також легко керувати паролями програм за допомогою завдань оновлення та скидання. Більшість організацій використовують найкращі методи, щоб зловживати шахрайством Інтернет-протоколу (IP) програми та запобігати зловмисникам. Методи посилення додатків включають обфускацію коду, антидебагінг, бінарне пакування та криптографію білого ящика [8].

Мультихмарні обчислення в першу чергу пов’язані з прибутковістю. Інституції з різними кремнієвими станціями (наприклад, розробки, тестування, виробництва та підтримки) рекомендується використовувати в окремих хмарні програми, щоб уникнути простоїв. Мультихмарний підхід дозволяє організаціям чітко покривати ризики збою обслуговування. Необхідно підвищити довіру до планів розповсюдження асоціації. Останнім часом компанії розробили багато власних хмарних додатків, головну рушійну силу оцінки хмар, і вирішили розмістити їх у кількох хмарах [9].

QoS є основним компонентом хмарних програм. Ресурси також мають відповідати Угоді про рівень обслуговування (SLA), визначеній кінцевим

користувачем і хмарними постачальниками. Належне використання ресурсів щодо потреб кінцевих користувачів збільшує прибуток постачальників хмарних послуг.

Управління QoS складається з проблеми розподілу ресурсів із такими службами, як продуктивність хмарних додатків, доступність і надійність. Метод керування QoS хмарних додатків автоматизує програмування розподілу ресурсів апаратного та програмного забезпечення хмарних обчислень [10].

У хмарних обчисленнях стратегії розподілу ресурсів гарантують віртуальні машини або розподіл фізичних ресурсів для користувача хмари з мінімальними ресурсами. Інші параметри відіграють важливу роль, коли SLA між постачальником послуг і користувачем хмари, як показано на рисунку 1.2.

Рисунок 1.2 – Стратегії розподілу ресурсів

Конкуренція за ресурси виникає, коли дві програми, розміщені на сервері, одночасно використовують ті самі ресурси. Подібним чином дефіцит ресурсів пов’язаний з дефіцитом ресурсів, який виникає, коли доступні обмежені ресурси. Як зазначено на наведеній вище діаграмі, фрагментація ресурсів є ще одним критичним параметром, який виникає, коли доступні ресурси обмежені, але недостатньо розумні для розподілу ресурсів потрібної програми. Надлишок і недостатність ініціалізації є останніми двома параметрами; надмірне забезпечення виникає, коли завдання ресурсів не відповідає фактичному попиту на ресурси з потребою.

Недостатня кількість ресурсів виникає, коли для завдання користувача виділяється кількість ресурсів, яка перевищує фактичну потребу [11].

Хмарні обчислення пропонують динамічно надані ресурси, які відображаються як один або кілька інтегрованих комп'ютерних ресурсів на основі обмежень. Провайдер додатків повинен визначити відповідну конфігурацію програмного та апаратного забезпечення протягом усього процесу хмарного надання, щоб гарантувати, що якість послуг (QoS) досягається без шкоди для використання та ефективності системи [12].

1.3 Традиційний режим експлуатації та обслуговування хмар

Загалом IT підприємства формувалися у три етапи. Перший етап – це ера мейнфреймів, яка характеризується записом і обробкою основних фінансових даних в одному місці, проблеми IT-системи не впливають на роботу. Другий етап — це інформаційна ера, яка характеризується записом і обробкою основних виробничих даних, а збій IT-систем призводив до припинення роботи деяких підприємств. Третій етап — це цифрова ера, яка характеризується записом і обробкою комплексних даних підприємств, зростанням обсягу даних у десятки разів, а також управлінням виробництвом і роботою підприємств цифровими засобами. IT системи серйозно впливають на нормальну роботу підприємств.

Традиційний режим роботи та технічного обслуговування - це в основному управління технічним обслуговуванням на першому та другому етапах, а новий режим експлуатації та технічного обслуговування повинен використовуватися на третьому етапі. Традиційний режим O&M здебільшого підтримує програми на основі архітектури chimney, тоді як новий режим O&M підтримує програми на основі розподіленої архітектури мікросервісів. [2] Концепція розповсюдження мікросервісів була висунута протягом 20 або 30 років, і теорія та практика є дуже зрілими. В останні роки під впливом цифрової економіки він процвітав і став основним способом

запуску програм у виробничих середовищах.

Початкова мета - підвищення ефективності експлуатації та обслуговування за допомогою автоматизованих засобів. [3] Неочікувано це завадило покращенню ефективності експлуатації та технічного обслуговування. За останні два роки багато підприємств вирішили повторно запровадити режим експлуатації та технічного обслуговування, який більше відповідає сучасним автоматичним методам інтелекту, і досягти оновлення звичайного експлуатації та технічного обслуговування до технології та технічного обслуговування CloudOps.

1.4 Існуючі методи оптимізації розподілу ресурсів

Алгоритми для оптимізації розв’язку задачі є складним інструментом для досягнення найкращого можливого результату. Алгоритм – це математична процедура для отримання результатів із вхідних даних за певних умов (Blum and Roli, 2003). Цільова функція — це показник, який використовується в методології оптимізації для визначення того, наскільки добре досягаються цілі в певних межах.

Алгоритми для оптимізації проблеми спрямовані на максимізацію або мінімізацію цільової функції (Yang, 2010). Програма використовує необхідні математичні методи для дослідження простору оптимальних рішень після визначення проблеми оптимізації. Можна вибрати між пошуком найкращої відповіді, що може зайняти багато часу, та майже оптимальним рішенням, яке можна знайти за менший час.

Немає універсально застосовного ефективного методу використання в розподілених системах через велику кількість параметрів, які необхідно враховувати. Як альтернатива, алгоритм оптимізації призначений для досягнення набору заздалегідь визначених цілей. Методи оптимізації можна розбити на кілька широких груп відповідно до їх визначальних характеристик. Виходячи з природи алгоритму та підходу до пошуку

рішення, їх часто класифікують як детерміновані або стохастичні (Yang, 2010). Стохастичні алгоритми базуються на непередбачуваності шляху та змінних, тоді як детерміновані алгоритми слідують заздалегідь визначеному шляху та набору змінних. У генетичних алгоритмах, наприклад, популяція рішень змінюється від циклу до циклу, оскільки пошук найкращого рішення залежить від випадковості. З іншого боку, стохастичні алгоритми забезпечують подібні результати, незважаючи на те, що для кожної популяції використовуються однакові шляхи. Методи, які поєднують детерміновані та стохастичні алгоритми, спрямовані на те, щоб уникнути недоліків обох типів алгоритмів, одночасно пожинаючи переваги обох.

Визначення основних методів у багатохмарному середовищі для розподілу ресурсів і керування ними [13]. У хмарних обчисленнях один постачальник хмарних послуг із певною кількістю послуг для кінцевого користувача є успішною централізованою установкою, яка забезпечує хорошу продуктивність. Спільне використання ресурсів у багатохмарному середовищі можна налаштувати централізовано або децентралізовано. Мультихмарність поділяється на різні типи на основі цих конфігурацій і стає частиною поступово розробленої мультихмарної архітектури.

Мультихмарна платформа з’явилася в пошуках оптимальних переваг. Її можна класифікувати за двома широкими можливими конфігураціями: централізована і децентралізована. Багатохмарне середовище можна налаштувати в централізований спосіб, який може мати всі ресурси під контролем центральної організації.

Централізація, безумовно, забезпечує більше контролю та вимагає відчутних інвестицій у розвиток центральної інфраструктури для обслуговування всіх користувачів хмари та їхніх відповідних запитів. У децентралізованій конфігурації мультихмари управління розподіляється між декількома об’єктами в мультихмарі на основі потужності та доступності ресурсів з різними зацікавленими сторонами, які можуть керувати своїми ресурсами відповідно до своїх локальних вимог і об’єднувати ресурси разом

із службами адміністрування. На основі розробки стратегії та політики в децентралізованій конфігурації учасники матимуть або єдину політику щодо послуг і цін, або учасники матимуть свою структуру бізнес-транзакцій, оскільки ця конфігурація жодним чином не обмежує [14].

На практиці кінцеві користувачі також відображають необхідні послуги для всіх учасників і аналізують затримку на основі близькості розташування, щоб отримати оптимальні результати. Будь-який учасник із кращим рівнем затримки та продуктивністю відповідає за надання послуг [15]. Концепція віртуалізації дуже застосовна для створення віртуальної хмарної структури, яка може надавати обчислювальні ресурси, сховище та мережеві ресурси кінцевому користувачеві для активації послуг із складових хмар у віртуальному середовищі виконання. Найбільш працездатна та стійка мультихмарна архітектура — це поєднання кількох SDC (невеликих центрів обробки даних), що представляють локалізовані ресурси з кожним учасником. Масштабованість такого розташування також можлива в інкапсульованому середовищі, не заважаючи кінцевому користувачеві [16]. Точка зору та потреби кінцевого користувача вирішують, яка централізована чи децентралізована конфігурація є більш придатною та продуктивною для кінцевого користувача [17].

Децентралізована хмара поєднує в собі переваги парадигм хмарних обчислень і однорангових (P2P) [18]. Безпосередньою перевагою децентралізації є покращена конфіденційність і безпека, оскільки централізований контроль над даними відсутній. Беручи участь у децентралізованій хмарі, малі хмарні гравці отримають канал для формування бізнес-груп і підтримки один одного. Кінцеві користувачі хмари отримають більше впевненості у використанні цієї моделі завдяки її новим функціям.

Участь SDC більш стабільна в децентралізованих хмарах, ніж безперервне приєднання та вихід у системі P2P [19]. Однак між децентралізованою хмарою та моделлю P2P є три ключові відмінності. По-

перше, партнери в децентралізованій хмарі – це переважно невеликі центри обробки даних, які є юридичними та економічними соціальними утвореннями, а не фізичними особами. Таким чином, стимули постачальників хмарних послуг до співпраці спрямовані не лише на максимізацію прибутку, але й на інші соціальні фактори, такі як правові обмеження, близькість розташування, довіра та лояльність. По-друге, порівняно з великою популяцією аналогів, кількість хмарних провайдерів відносно невелика. По-третє, обчислювальне середовище не таке високодинамічне, як у моделі P2P [20].

Варто зазначити, що децентралізована хмара не замінить централізовану хмару, але вони доповнюють одна одну. Централізовані хмари вже продемонстрували свою потужність у веб-хостингу та додатках із інтенсивним обчисленням. Однак децентралізована хмара є найкращим вибором, якщо додаткам потрібна висока конфіденційність інформації або кращий контроль над даними [21].

2 МЕТОДИ УПРАВЛІННЯ РЕСУРСАМИ В ХМАРНИХ СИСТЕМАХ

2.1 Розробка структури системи управління ресурсами в хмарних системах

Запропонована структура містить три сегменти у мультихмарі для розподілу ресурсів і управління ними. Раніше показано, що мультихмарне середовище розширюється в корпоративному секторі та на рівні CSP для надання спеціалізованих послуг, пов’язаних із мультихмарним середовищем. Три фактори пов'язані з цим процесом. Постачальник послуг переважно надає послуги в будь-якій конкретній конфігурації, тобто приватній, загальнодоступній, гібридній або спільнотній хмарі. У той же час, другий постачальник ресурсів має фізичну структуру та розширює об’єкт для інших постачальників ресурсів. Третім важливим фактором у цьому процесі є кінцевий користувач, який працює в багатохмарному середовищі.

Постачальник ресурсів має певну обчислювальну систему, сховище та мережу. Ці ресурси надаються кінцевому користувачеві через віртуальні машини, докери, контейнери та інші засоби, якими володіють постачальники хмарних послуг. Послуги надаються на ресурсах відповідно до певної стратегії ціноутворення та QoS у різних випадках.

Існує дев’ять кроків для параметрів QoS для розподілу ресурсів у хмарному середовищі. На першому кроці клієнт надсилає запит на сервер через запит у віртуальному середовищі. Конфігурація та тестування програми виконуються з цілями QoS у віртуальному середовищі. На другому кроці необхідний запит надсилається до керування кластером для кластерних служб у багатохмарних середовищах через різні конфігурації з параметрами QoS, як показано на рисунку 2.1. Параметри QoS включають порогову конфігурацію, тестові логічні контролери та покроковий план. На етапах 4, 5 і 6 модель тестового стану та аналіз моделі аналізуються на різних

конфігураціях і перевіряються, чи задовольняється QoS. На сьомому кроці, якщо параметр QoS задовольняється, виконується надання ресурсу з пулу ресурсів. На кроці 8 виконуються рівні моніторингу та бізнес-логіки та передаються у віртуальне середовище. Нарешті всі результати та витрати передаються клієнту.

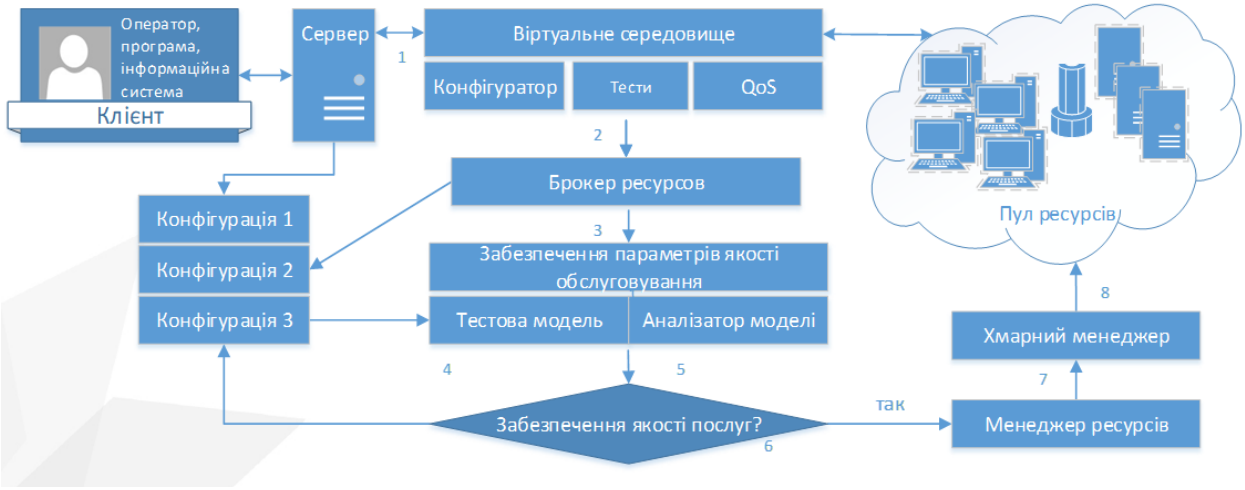

Рисунок 2.1 – Запропонована структура та порядок надання послуг

Запропонована структура враховує три основні параметри для QoS і розподілу ресурсів у багатохмарному середовищі, як показано в Таблиці 2.1.

Ці параметри взято з літератури та різних хмарних моделей, зокрема варіантів багатохмарних середовищ. Структура запропонованої системи базується на цих трьох параметрах, щоб розширити інші процеси та логіку для динамічного та швидкого розподілу ресурсів із покращенням QoS у багатохмарній платформі.

Важливо відзначити, що описова структура має три основні параметри, необхідні для розробки будь-якого типу мультихмари. Перший параметр, співпраця, є ключовим елементом у розробці основи для оптимізації та спільного використання. Як обговорювалося, багатохмарність — це об’єднання різних сервісів від різних постачальників хмарних сервісів, високий рівень неоднорідності від структурного до сервісного рівня робить цей параметр фундаментальним і вирішальним. Підмножина цього параметра

включає такі параметри, як стимули та стратегії. Подальша деталізація цих параметрів надала змінні, пов’язані з фінансами, соціальними факторами та ефективністю SLA. Крім того, ці параметри є життєво важливими для розподілу ресурсів із очевидним QoS.

Таблиця 2.1 – Параметри QoS

Multi-cloud	Cooperation	Incentives	Financial SLA performance
		Strategies	Workload base SLA base
	Optimization	Objective	Revenue Improve QoS
		Model	Constraint base Game theory
		Evaluation	Process cost SLA base
	Data sharing	Inter cloud	Storage Optimization
		Intra cloud	Duplication Summarisation

Запропонована структура показує, що співпраця неможлива без стимулів і стратегії. Хороша стратегія може призвести до мінімальних стимулів і максимальних результатів. Однак стимули не обов’язково є фіскальними. Ці стимули можуть стосуватися соціальних факторів або можуть призвести до SLA на основі ефективності. Тому постачальники ресурсів потребують співпраці на основі стимулів і ретельно розробленої стратегії. Стратегії розподілу ресурсів пов’язані зі стимулами, особливо в багатохмарному сценарії, де постачальники ресурсів мають свої активи та хочуть максимізувати свої стимули, як показано на рисунку 2.2.

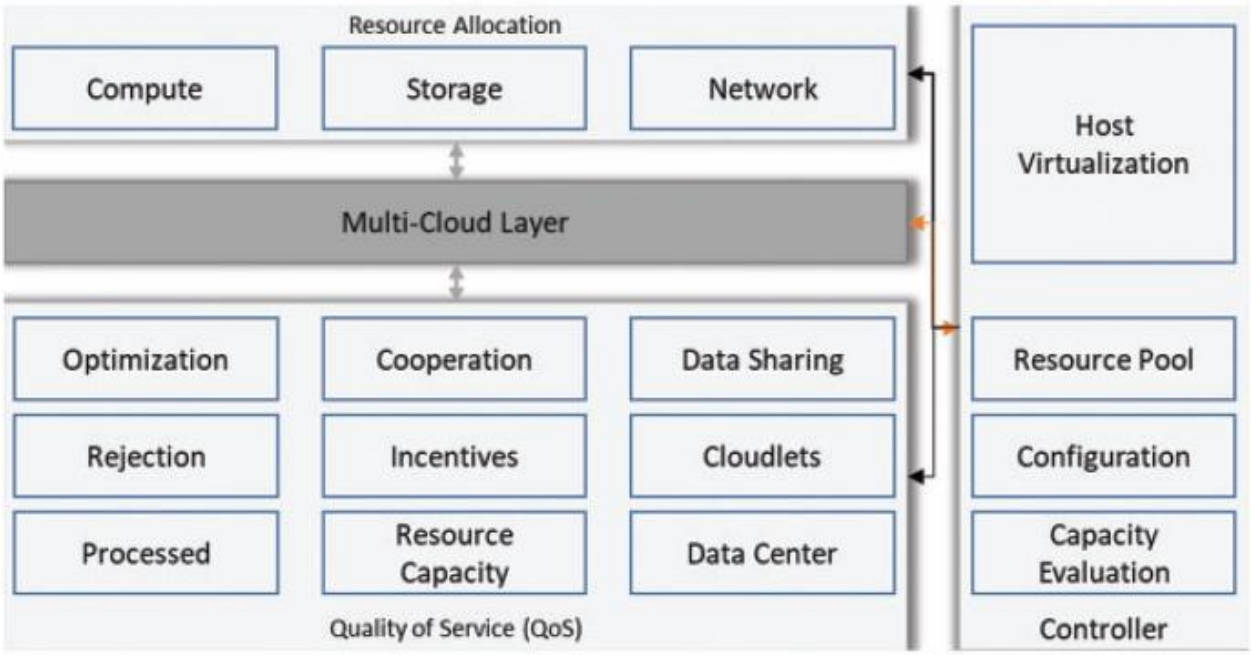

Рисунок 2.2 – Параметри для розподілу ресурсів у мультихмарі

Розвиток цієї співпраці є складним через гетерогенну природу мультихмарності. Стимули можуть бути різними з різними цілями для постачальника послуг. Стабільна пропускна здатність і менша затримка можуть бути метою, пов'язаною із стимулами. Іншим фактором, який розглядається, є зв'язок між оптимізацією та ефективністю співпраці, що призводить до досягнення послуги та ефективності цілі постачальника ресурсів і постачальника послуг. Тому пропонується мати динамічну перспективу після використання цієї основи як частину структури.

Існує два явних сценарії співпраці для розподілу ресурсів у багатохмарному середовищі. Перший сценарій передбачає наявність центральної сутності, тоді як другий сценарій передбачає децентралізовану багатохмарну конфігурацію. Перший сценарій простий щодо розподілу ресурсів, оскільки всі учасники надають послуги та ресурси центральному органу для виконання конкретних завдань. Виконання завдань є головною турботою всіх учасників, щоб вони могли переходити до наступного завдання. Центральний орган здійснив основну координацію, і жоден із постачальників ресурсів не шукає індивідуальної вигоди. Таким чином,

стимули стають визнанням продуктивності, рейтингом, а конкурентне середовище завжди існує в централізованому багатохмарному середовищі.

Іншими словами, співпрацю легко та просто встановити в централізованій мультихмарі, не маючи глибокої стратегії для розробки умов співпраці та рівнів стимулів, як показано на рисунку 2.2.

Оскільки жодна центральна організація не контролює та не організовує співпрацю між усіма зацікавленими сторонами, співпраця з розподілом ресурсів є дуже складною в децентралізованому багатохмарному середовищі.

Тому ключове питання полягає в тому, чи можливо створити середовище співпраці для постачальників ресурсів. У децентралізованому багатохмарному середовищі вивчення потенційних сценаріїв і змінних, на яких постачальники ресурсів базуватимуть свої рішення щодо спільного розподілу ресурсів, має вирішальне значення. Завдяки двом різним типам стимулів і тактик, які використовуються для класифікації процесів, існує дві підмножини співпраці.

Мультихмарна платформа — це середовище, яке надає всі можливості для тестування технологій і бізнес-теорій. Розподіл хмарних ресурсів із першим рівнем взаємодії є основним параметром для успішного багатохмарного середовища. Збереження інтересів або стимулів постачальника ресурсів незмінним означає довшу та стабільнішу домовленість про надання послуг. Після досягнення співпраці за допомогою стимулів і стратегії з усіма підмножинами, наступним логічним кроком є оптимізація. Необхідною умовою для працездатності оптимізації є повний набір інформації, пов’язаної з ресурсами та багатохмарною структурою.

Спосіб використання оптимізації з повною інформацією або без неї – зберегти оптимізацію як централізований процес (з вичерпною інформацією, пов’язаною з усіма постачальниками ресурсів, ресурсами та типами хмари та конфігураціями). У разі недостатнього набору інформації оптимізація може бути локалізованим процесом у кожного постачальника ресурсу. Однак він буде ефективним лише на локальному рівні і не матиме справу з

мультихмарністю як платформою. Ми розділили оптимізацію на цільову функцію запропонованої моделі, обмеження, модель і критерії оцінки.

Після розробки цільової функції та її параметрів модель для оптимізації та розподілу ресурсів багатохмарної платформи базується на двох основних параметрах, тобто моделі, пов'язані з обмеженнями або базою правил, а інший параметр – теорія ігор (аналіз переваг для ситуації, яка виграє).

Місткість ресурсів повинна бути між загальними ресурсами та загальними виділеними ресурсами, наприклад, обчислення, зберігання та пропускна здатність. Усі ресурси повинні працювати в рамках виділеного бюджету з точки зору потужності, часу та доступності. Усі виділені/доступні ресурси повинні виконувати завдання згідно з графіком; відкладені завдання можна пропустити та перепланувати. У мультихмарі фактор надійності є частиною бази правил; випадки збою з точки зору доступності, доступності, реплікації тощо безпосередньо впливають на надійність. База правил відповідає законодавчим і нормативним вимогам геологічного розташування, де експлуатується хмара. Ця концепція підходить для всіх типів кількох хмар у централізованих і децентралізованих конфігураціях. Багато моделей намагалися сформулювати механізми розподілу ресурсів за допомогою теорії ігор.

Переважно ці моделі беруть ціну та QoS як ключові елементи для розробки відповідного моделювання.

Для запропонованої структури критерії оцінки для аналізу методів оптимізації та розподілу послуг базуються на таких моментах:

- обчислювальна цінність моделі є важливим сегментом, який слід враховувати;

- вартість процесу включає загальний час, необхідний для виконання конкретних завдань, і затримки, середній очікуваний час і фактичні витрати часу;

- фінансовий прибуток також є важливим параметром для відображення ефективності будь-якої моделі розподілу ресурсів;

- коефіцієнт витрат пов’язаний з обчисленнями, пропускною здатністю та сховищем, а також налаштуваннями мультихмарної конфігурації, наприклад централізованої чи децентралізованої;

- пропускна здатність мережі з точки зору пропускної здатності та пропускної здатності є стандартним параметром для оцінки якості обслуговування;

- пропускна здатність мережі пов’язана з використанням цієї потужності в режимі реального часу хмарними службами, різниця полягає в значенні коефіцієнта затримки та продуктивності;

- доступність даних у реплікації в різних геологічних місцях для зниження вартості обробки;

- балансування робочого навантаження шляхом управління отриманими запитами та виконання запитів у різних центрах обробки даних;

- SLA орієнтовані на оцінку продуктивності та параметрів оптимізації.

Випадки порушення вказівок SLA є ключовими параметрами, які слід враховувати для визначення якості обслуговування.

Мета полягає в тому, щоб планувати обмін і переміщення даних у дисциплінований спосіб з використанням різних інструментів оптимізації, тоді як постачальники хмарних послуг також надають небагато. Міжхмарне планування позитивно впливає на керування затримками та масову передачу даних. У цьому методі пропускна здатність зв’язку між двома вузлами розраховується за допомогою кількох протоколів, щоб збільшити пропускну здатність і мінімізувати час транзакції; він також враховує обчислювальну потужність і швидкість зберігання, оскільки це впливає на масову передачу. Він підтримується алгоритмом оптимізації для керування транзакціями під час передачі та використання інших протоколів для отримання максимальної пропускної здатності та якості обслуговування [22].

Цей метод допомагає оптимізувати дані в стані спокою, тобто перед передачею даних краще максимізувати дані, щоб забезпечити швидке та рентабельне передавання. Дані перетворюються на менші фрагменти; метод

без втрати інформації використовує стиснення, зберігаючи всю інформацію метаданих у вихідному наборі даних. Інший процес, який підтримує оптимізацію, називається дедуплікацією, яка забезпечує передачу даних один раз, і ресурси не повинні використовуватися повторно для тієї самої передачі даних. Таким чином, метадані містять інформацію, пов’язану з усіма реплікаціями, позначеними як «передані», щоб уникнути марної витрати ресурсів. Оптимізація також пов'язана з оптимізацією протоколу; CloudOpt – це перевірена платформа, яка надає численні служби для стиснення даних, дедуплікації, оптимізації, налаштування пропускної здатності та оптимізації протоколу [23].

2.2 Застосування технології CloudOps

CloudOps – це фактично автоматизована робота та обслуговування в хмарі, CloudOps = Cloud x DevOps, підкреслюючи, що це повне використання характеристик самої хмари для кращої політики [16] DevOps, прискорення швидкої та стабільної доставки. Основний момент полягає в тому, щоб підкреслити характеристики самої хмари, без необхідності нашої повторної розробки. Характеристики самої хмари включають високу еластичність, високу стандартизацію, високий рівень автоматизації та режим самообслуговування хмари, що означає, що користувачі можуть отримати доступ до неї відповідно до власних потреб, не покладаючись на будь-яку іншу підтримку можливостей.

CloudOps визначає п’ять вимірів, на яких підприємства зосереджуються в процесі побудови хмари та керування хмарою, і це повторює п’ять загальних проблем клієнтів хмари. Це вартість, автоматизація, надійність, еластичність і безпека, скорочено CARES [4]. Наприклад, інструмент оптимізації витрат вирішує проблему витрат, можливість автоматизації вирішує проблему ефективності автоматизованої роботи та технічного обслуговування, можливість надійності може бути

використана для підвищення стабільності бізнесу та скорочення часу втрати обслуговування, здатність еластичності вирішує проблема доступності додатків, а можливість відповідності вимогам безпеки покращує безпеку бізнесу. [5] Тому CloudOps — це не лише концепція експлуатації та технічного обслуговування, але й являє собою загальний термін для хмарних постачальників, який надає вам набір стандартизованих інструментів для роботи та технічного обслуговування.

Корпоративна стратегія є основою всієї корпоративної діяльності. Стратегічні цілі — це довгострокові цілі, а бізнес-цілі — короткострокові. Це вимагає, щоб ресурси підприємства відповідали цілям організації не тільки для забезпечення відповідності та раціональності довгострокової стратегії постачання ресурсів, але й для забезпечення достатньої гнучкості для досягнення короткострокових цілей. Відповідно до передумови чіткого та чіткого консенсусу, обов’язки та повноваження організації мають бути уточнені, а показники бюджету потенціалу бізнес-системи мають бути реалізовані. Таким чином, управління організаційними ресурсами повинно мати можливість створювати багаторівневі організації на основі ролей або користувачів, а також може розподіляти квоти ресурсів і квоти витрат для кожної організації. У той же час, коли організація або ресурси змінюються через бізнес-причини, вона повинна мати можливість гнучко регулювати організаційну структуру та розподіл ресурсів.

Таким чином, основою корпоративної хмари є використання хмарних ресурсів, а ресурси в хмарі мають «готові» та «гнучкі» способи оплати. Як керувати ресурсами – це розглянути питання про те, як використовувати ресурси зручно та як використовувати ресурси безпечно. [6] Після зловживання ресурсами може виникнути серія серйозних наслідків, як-от ситуації з-під контролю та втрата даних. У сфері IT-менеджменту ресурси займають важливе місце, а ресурси тісно пов’язані з ідентифікацією, повноваженнями, фінансовими витратами, дотриманням вимог аудиту тощо. Це закладає основу для вивчення технології та обслуговування CloudOps, яка

використовує автоматизовані операції у хмарі для оптимізації управління ресурсами та підвищення ефективності роботи в корпоративному хмарному середовищі.

2.3 Моделювання запропонованих рішень

Основні аспекти реалізації запропонованого методу та первинна конфігурація компонентів моделювання, які будуть використані для створення необхідних модулів, будуть розглянуті в цьому розділі. Також визначимо, як будуть оцінюватися етапи розподілу ресурсів і які критерії використовуватимуться. Однак оцінка та унікальне налаштування кожного алгоритму будуть розглянуті в наступних розділах.

Спочатку обговорюються переваги моделювання. Апаратне забезпечення, програмне забезпечення та основна мережа присутні в налаштуваннях хмарних обчислень. Крім того, клієнти мають різні та, можливо, суперечливі очікування QoS. Розробка та оцінка показників продуктивності моделі за різних налаштувань може бути складною справою в реальному хмарному середовищі, наприклад Amazon EC2 або Microsoft Azure.

Це необхідно, оскільки запропонований підхід вимагає тестування багатьох архітектур центрів обробки даних із різними вимогами. Крім того, процес оцінювання обмежений, коли використовуються реальні параметри через обмеження інфраструктури, і повторна оцінка випробувань стає дуже складною з точки зору вимірювання. Це робить повторне тестування неймовірно складним і вимагає значних модифікацій середовища та інфраструктури тестування. Однак перенастроювання параметрів порівняльного аналізу для зміни додатків і робочих навантажень для проведення більшої кількості тестів займає багато часу, грошей. Оскільки реальне середовище іноді надто складне для еталонних досліджень і оцінок, розробники та дослідники часто вдаються до використання симуляторів

замість цього. Коли справа доходить до тестування нових налаштувань інфраструктури, розробники можуть заощадити час і зусилля, застосувавши інструменти та середовища, призначені для моделювання. Використання симуляторів також може підвищити адаптивність моделей, оскільки це дозволяє розробнику будувати структуру, зручну для користувача та вимог. У світлі цих міркувань модель буде перевірено за допомогою симулятора.

CloudSim, GreenCloud і MDCSim є лише кількома прикладами інструментів моделювання, доступних для хмарних систем. У цьому дослідженні використана платформа CloudSim для реалізації та оцінки нашого методу. Це пов'язано з рядом факторів. Для початку CloudSim – це безкоштовний і загальнодоступний симулятор на основі Java. Крім того, він має різні підмодулі, які імітують основні елементи та рівні хмарних середовищ. Це дозволяє легко адаптувати симуляцію до конкретних конструкцій шляхом додавання або видалення компонентів за потреби. У цьому дослідженні ми використовували CloudSim 3.0.3. Воно дозволяє змоделювати черги та обробку подій, впровадити нові елементи хмарної системи, включаючи хости, центри обробки даних, брокери та віртуальні машини, а також міжкомпонентний зв’язок і керування часом.

Далі наведено список основних класів CloudSim.

Центр обробки даних представляє основну апаратну архітектуру хмари та знаходиться під контролем провайдерів.

Термін «брокер» означає модуль брокера, який керує взаємодією між клієнтами та постачальниками послуг.

«Хост» означає, що він імітує сервер у центрі обробки даних.

Реалізація віртуальної машини (VM), яка працює на хості та виконує обов’язки.

Cloudlet модель рівня SaaS: хмарні додатки та сервіси моделей (у цій моделі це позначено терміном task).

Розподіл віртуальних машин: політика, надана в характеристиках центру обробки даних, яка визначає процес призначення віртуальних машин

хостам.

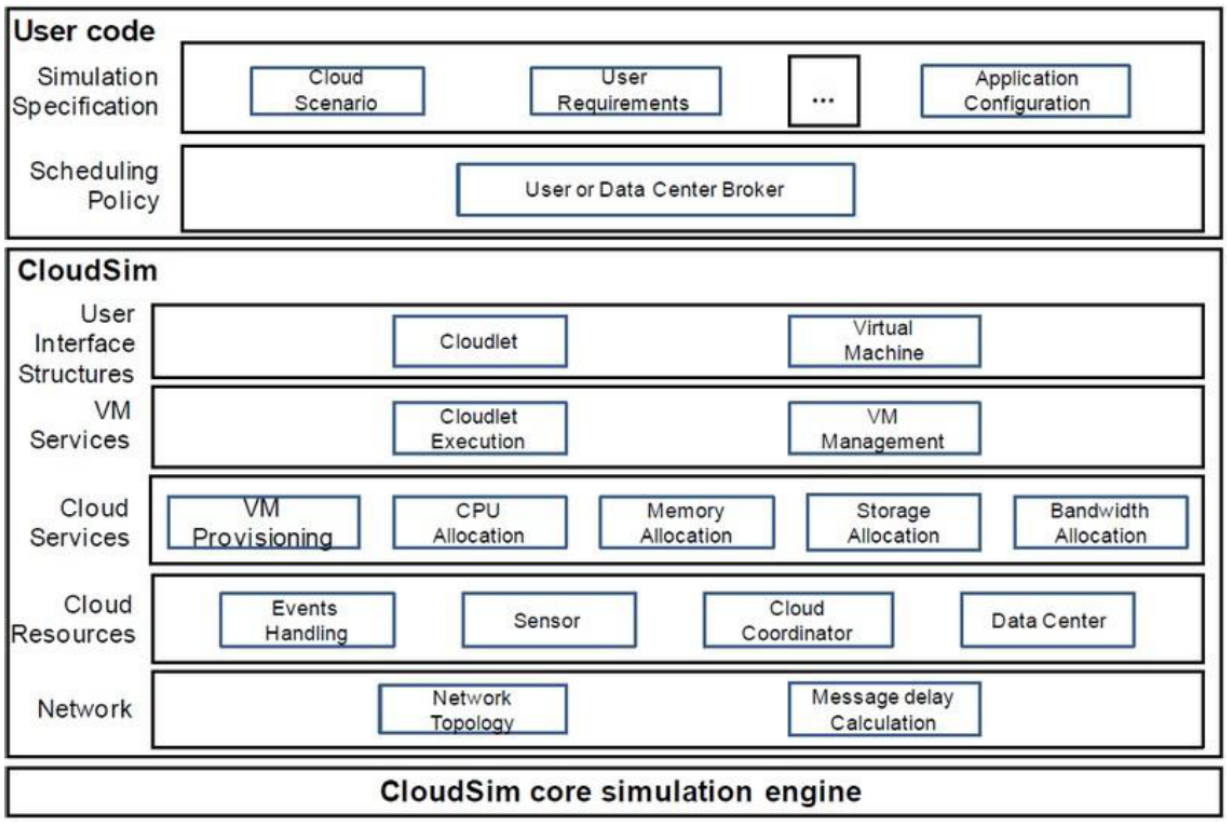

Рисунок 2.3 – Архітектура CloudSim

VmScheduler — це метод, за допомогою якого ресурси ЦП хоста розподіляються між кількома віртуальними машинами. Це програма, яка встановлюється на кожному фізичному хості в центрі обробки даних і відповідає за розподіл обчислювальної потужності між віртуальними машинами.

Cloudlet Scheduler — це унікальна стратегія віртуальної машини (VM) для розподілу процесорного часу між хмарлетами.

Було внесено кілька змін до класів симулятора CloudSim, щоб реалізувати модель і зробити її застосовною до даної проблеми. З цією метою було оновлено попередні класи та створено нові, щоб допомогти в таких сферах, як обчислення навантаження, кластеризація хостів і узгодження.

3 РОЗГОРТАННЯ ТА ДОСЛІДЖЕННЯ

3.1 Управління хмарними ресурсами

У системі хмарних технологій контейнеризація стала першим вибором для розробників для розгортання програм, а Kubernetes є кращою системою оркестровки контейнерів і планування. Незважаючи на те, що контейнеризація та Kubernetes значно спростили розгортання додатків, керування послугами все ще вимагає глибокої участі розробників. Основна концепція сервісної сітки полягає в маршрутизації запитів між мікросервісами на рівні інфраструктури за допомогою проксі-серверів, які працюють паралельно з кожним сервісом, утворюють мережу мережевого формату та взаємодіють із мікросервісами.

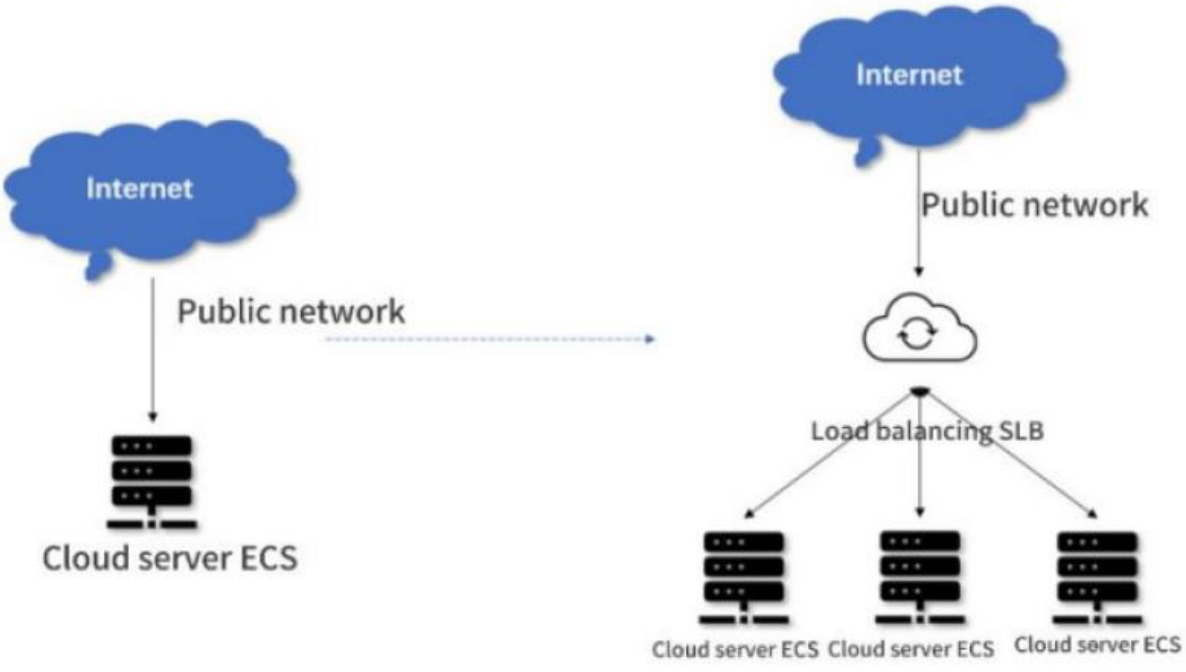

Рисунок 3.1 – Сервер управління хмарними ресурсами

Service Mesh, як рівень інфраструктури, який займається зв’язком між службами, допомагає розробникам позбутися дилеми проблем зв’язку між службами та доручає гріду важку роботу з логіки керування зв’язком, тому

деякі люди називають це другим поколінням мікросервісів. Балансування навантаження розподіляє трафік доступу між декількома внутрішніми серверами на основі політик пересилання для досягнення високого рівня паралелізму та підвищення продуктивності обробки. У процесі автоматизації O&M підприємства балансування навантаження стає необхідною частиною для врахування зростаючої кількості сервісів і користувачів.

Великий трафік розподіляється на декілька серверів у фоновому режимі для обробки, щоб впоратися з високими проблемами одночасного виконання. В управлінні експлуатацією та обслуговуванням обладнання підприємства ефективне управління робочою силою, запасними частинами, технологіями та ресурсами даних є ключовим для зниження витрат на експлуатацію та технічне обслуговування та підвищення загальної ефективності роботи. Людські ресурси, запасні частини, технічні ресурси та ресурси даних відіграють важливу роль в ефективному управлінні автоматизованими хмарними активами, і завдяки розумному плануванню та управлінню цими ресурсами можна досягти ефективного, розумного та сталого розвитку експлуатації та обслуговування обладнання.

У хмарній екосистемі, яка характеризується високим рівнем паралелізму та розподіленими архітектурами, визначення пріоритетів критичних робочих процесів має важливе значення для підтримки безперервності та продуктивності бізнесу. Без належного розподілу ресурсів і керування робочим навантаженням основні процеси можуть затримуватись або збої через конкуренцію ресурсів або надмірну фрагментацію.

Значна частина щоденної роботи з експлуатації та технічного обслуговування включає управління конфігурацією та підтримку стану послуг. В даний час управління конфігурацією на основі стану (стан системи, стан коду, стан конфігурації та стан процесу) значно розвинене та досягло значного прогресу в експлуатації та обслуговуванні. Поява нових інструментів є нескінченною та вражаючою, і в практичних застосуваннях, незалежно від того, замінюють ці інструменти чи об’єднують ці інструменти,

розуміння кожної конкретної сцени та вибору буде іншим і, зрештою, може відображатися в зовсім іншій формі. Багато великих IT-компаній використовувати маріонетку для керування та розгортання програмного забезпечення в кластерах. Переваги та недоліки полягають у тому, що веб-інтерфейс користувача створює звіти про обробку, списки ресурсів, керує вузлом у реальному часі. Недоліки полягають у тому, що процес інсталяції є більш складним, ніж інші інструменти, і вимагає вивчення Puppet DSL або Ruby. Процес інсталяції не перевіряє помилки та створює звіти про помилки.

Кожен інструмент O&M використовується лише для допомоги персоналу в O&M. Кожен інструмент має свої переваги. Puppet застосовний для автоматичного налаштування та розгортання програмного забезпечення. SaltStack призначений для управління інфраструктурою, і його можна запустити за лічені хвилини, легко керуючи тисячами серверів і досить швидко. Ansible використовується для пакетного налаштування операційної системи, пакетного розгортання програми та пакетного запуску команд.

3.2 Опис експериментів

Запропоновану структуру було оцінено за допомогою симулятора CloudSim, набору інструментів для моделювання хмарних обчислень з різними перспективами для QoS, оптимізації та інших відповідних аспектів. Під час моделювання було створено три основні компонента, тобто децентралізовану мультихмару, брокер та менеджер навантаження. Децентралізована хмара спрямована на імітацію SDC, віртуальних машин і спільного використання в SDC. Роль брокера полягає в тому, щоб передати навантаження на віртуальні машини та запитати розподіл ресурсів, щоб вимагати якості обслуговування. Диспетчер навантаження відображає запити та досягнення відповідно до кінцевого користувача [24].

Кілька постачальників хмарних послуг стягують з користувачів плату за різні послуги, тому найняти хмарні служби для експериментів складно.

Тому дослідження, впровадження та розробки, пов’язані з хмарою, в основному виконуються за допомогою симуляторів хмари. Для цілей перевірки в цьому дослідженні CloudSim використовувався для моделювання великомасштабних хмарних обчислень за допомогою віртуального сервера та створення індивідуальної віртуальної машини. CloudSim [25] також використовується для оцінки моделей із змодельованими робочими навантаженнями та центрами обробки даних. Було використано наступну стратегію для моделювання.

CloudSim не має графічного інтерфейсу користувача (GUI); замість цього він працює з різними інтегрованими середовищами розробки (IDE), такими як Eclipse і NetBeans. Ця симуляція використовує Eclipse з CloudSim. Конфігурація CloudSim базується на хмарних програмах, хостах, віртуальних машинах і рандомізації для імітації різних конфігурацій на різних віртуальних машинах, як показано в Таблиці 3.1.

Таблиця 3.1 – Конфігурація середовища виконання експериментів

Засоби	Опис
Apache Tomcat 9.0.48	Webserver for hosting DSpace
DSpace 5.10	Database Simulation host
JDK8	Development environment
PstgreSQL 9.6.22	Database Simulation
Dataset	KDD2020

Google Cloud Services (GCP) використовується як хост-платформа для моделювання, тоді як Microsoft Azure використовується для порівняльної перевірки.

Ця симуляція оцінює різні конфігурації в кількох налаштуваннях хмари, щоб переконатися в дійсності параметрів, згаданих у запропонованій структурі, тобто оптимізації, співпраці та спільного використання. Всі три параметри представляють якість обслуговування. Віртуальні машини

рандомізовано з п’ятьма пристроями, як показано на знімку екрана конфігурації. Симуляція активується з мінімальною конфігурацією замість великої кількості ресурсів, щоб зрозуміти умови робочого навантаження та збої. Віртуальна машина має 1 ГБ пам’яті.

Замість використання фізичного набору даних DSpace використовується для моделювання наборів даних у діапазоні від 1000 до 10000 із параметрами одночасних запитів для оцінки керування потужністю запропонованої структури.

Як показано на рисуноку 3.2, конфігурація віртуальної машини маэ сховище на рівні 10000 з оперативною пам’яттю на рівні 512, а швидкість мільйонів інструкцій за секунду (MIPS) зафіксована на рівні 250.

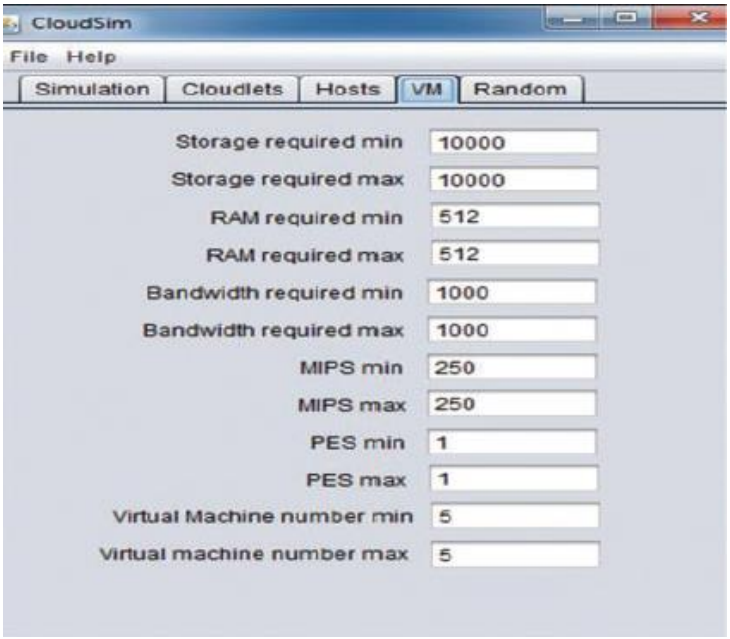

Рисунок 3.2 – Конфігурація віртуальної машини

Максимальний статус і кількість віртуальних машин – п’ять на мінімальному та максимальному рівнях. Рандомізація використовує ці п'ять машин для створення кількох епізодичних транзакцій для обчислень, мережі та зберігання. У той же час, хмара налаштована для забезпечення кількох імітованих центрів обробки даних для віртуальних машин для імітації кількох ресурсів і кількох екземплярів сховища, як показано на рисунку 3.3.

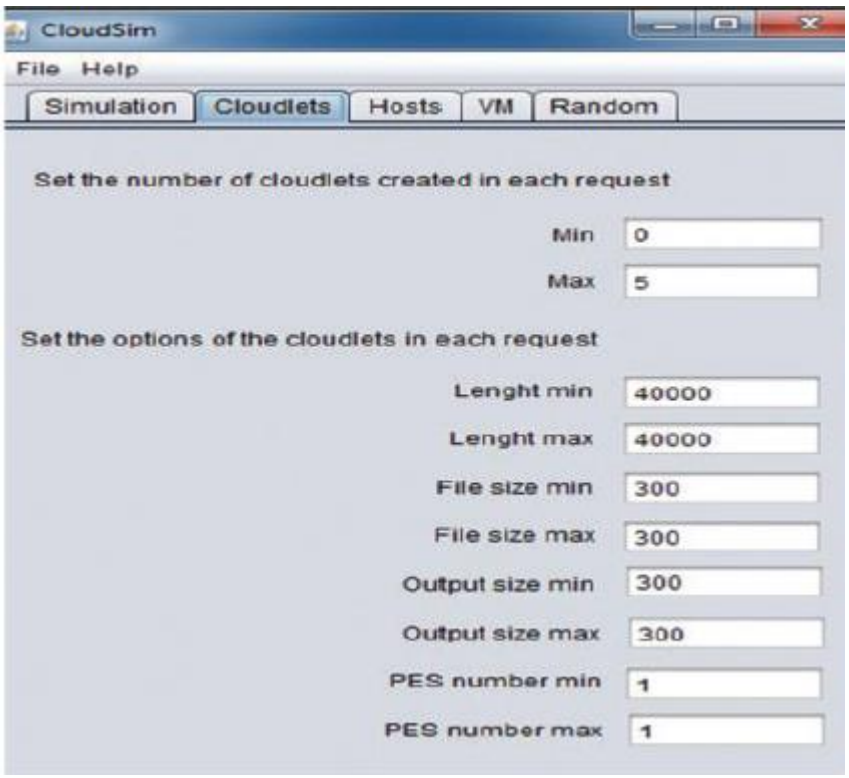

Рисунок 3.3 – Конфігурація Cloudlets

Використовується максимум 5 хмарних програм, що означає, що кожна віртуальна машина моделюватиме 5 різних хмарних програм. Загалом моделюється 25 екземплярів, щоб оцінити запропоновану структуру з використанням обчислень, сховища та мережі.

Розподіл ресурсів із різноманітною доступністю постачальників хмарних послуг відіграє важливу роль, як зазначено в постановці проблеми та запропонованій структурі в мультихмарі. Таким чином, конфігурація віртуальних машин визначається із середньою потужністю, але з проблемою обробки в термінах MIPS на рівні 250 з мінімальною та максимальною кількістю 5 віртуальних машин в одному екземплярі, як показано на рисунку 3.4. Комп’ютер, мережа та сховище розподіляються в цьому блоці. Подальша конфігурація CloudSim має складний сценарій перевірки запропонованого фреймворку.

Перший експеримент базується на загальних ресурсах, тобто обчисленні, сховищі та мережі на SDC та робочому навантаженні в певну

годину пік, як показано на рисунку 3.5. Помітно, що все моделювання виконується в децентралізованому багатохмарному середовищі.

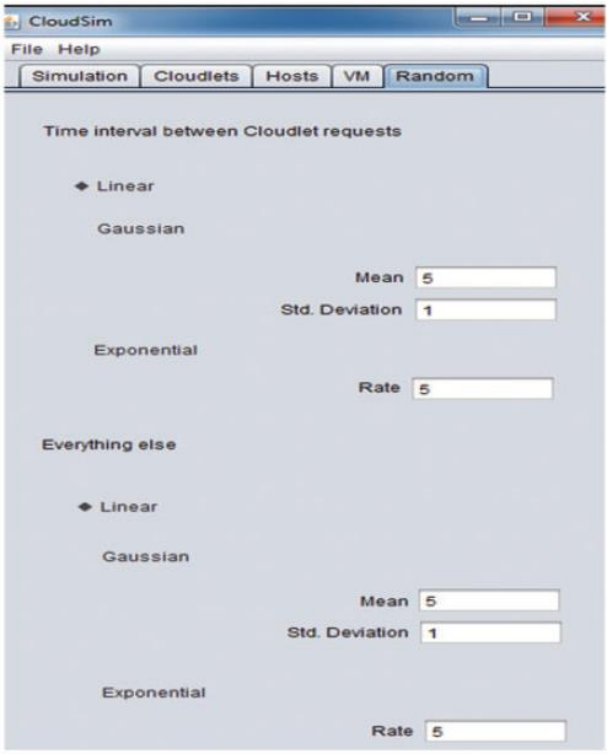

Рисунок 3.4 –Конфігурація рандомізації

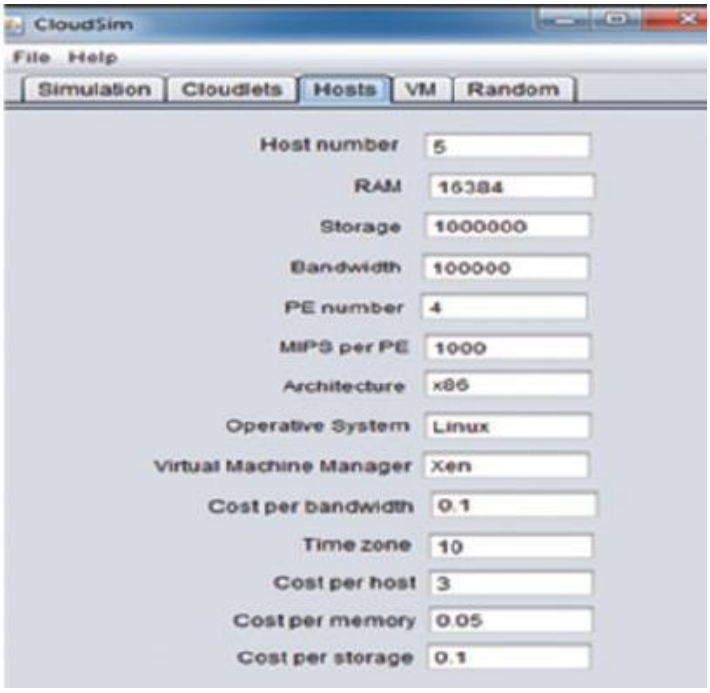

Рисунок 3.5 – Конфігурація хоста

Початковий цикл перевірки моделюється без запропонованої структури та параметрів, які ми визначили в структурі. Результат показує зниження ресурсної потужності зі збільшенням навантаження та інтервалу часу. SDC та відповідні ресурси не змогли оптимізувати ресурс для підвищення якості обслуговування. Конфігурація моделювання, як показано на рисунку 3.6 значить, що фактичний прогін із повтором моделювання становить 30 із початковим значенням 42.

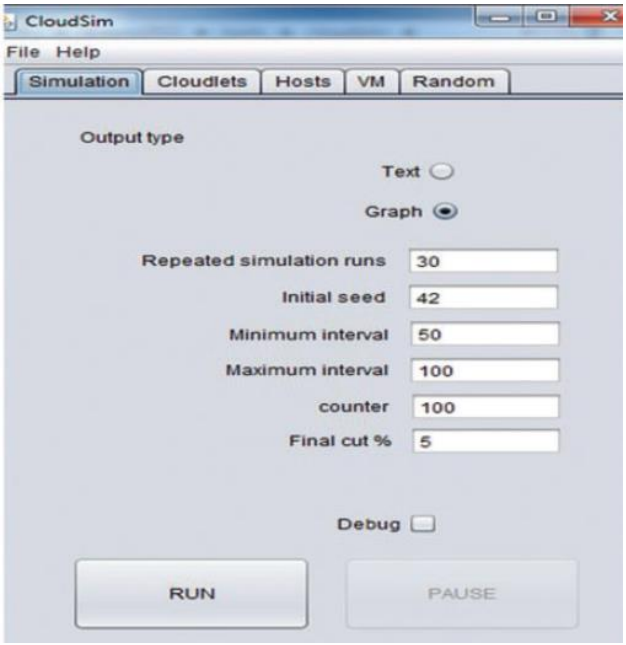

Рисунок 3.6 – Конфігурація моделювання

3.3 Результати експериментів та їх обговорення

Безперервність моделювання забезпечує ще один сплеск QoS, який досягає максимального значення та знову починає знижуватися, як показано на рисунку 3.7. Таким чином, децентралізована мультихмара зі стандартною конфігурацією коливається щодо якості обслуговування та ресурсної потужності. Коли ємність починає вичерпуватися, якість обслуговування починає знижуватися, і жодний вбудований протокол оптимізації не впливає

на це. Симулятор використовує параметри та значення для створення децентралізованої мультихмари на основі різних перевірених сценаріїв.

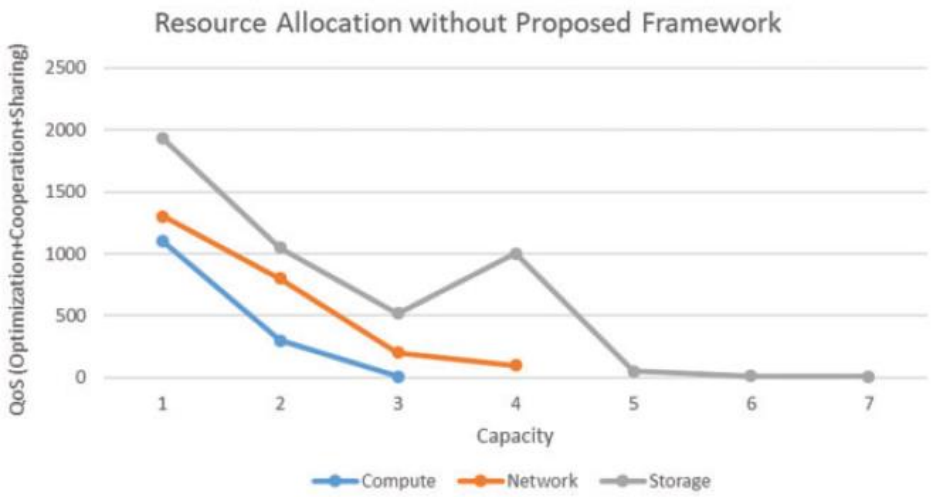

Рисунок 3.7 – Розподіл ресурсів без оптимізації

Другий цикл моделювання зосереджений на тому ж експерименті, але включає запропоновану структуру та відповідні параметри, які обговорюються в розділі методології. Симуляція знову надає ті самі три модулі, тобто децентралізовану мультихмару, посередника та менеджер навантаження, з тими самими параметрами та налаштуваннями SDC, як показано на рисунку 3.8.

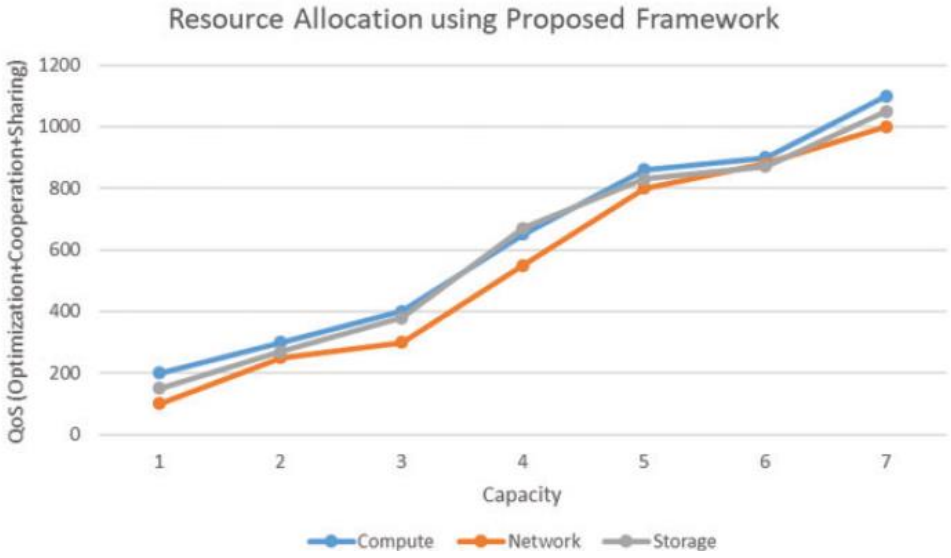

Рисунок 3.8 – поділ ресурсів з оптимізацією запропонованим методом

Те саме навантаження моделюється з ідентичним таймфреймом. Результати такі, як на графіку. Різке зростання помітне за всіма показниками, тобто, обчислення, зберігання та мережа. Запропонована структура використовує сценарій для підвищення якості обслуговування шляхом оптимізації виділених ресурсів у заданий період часу. Значення співпраці, оптимізації та доступності даних приймаються як змінні, які коливаються від 1 до 10. Результати моделювання є дуже важливими для підтримки об’єктивних та дослідницьких питань цієї структури, тобто, оскільки співпраця зростає, якість обслуговування також збільшення; так само виконуються посередницькі фактори хмарної оптимізації замість локальної оптимізації, що також демонструє позитивний вплив. Нарешті, дані доступні за допомогою стратегічної реплікації, щоб мінімізувати втрату ресурсів і затримку, як показано на рисунку 3.9.

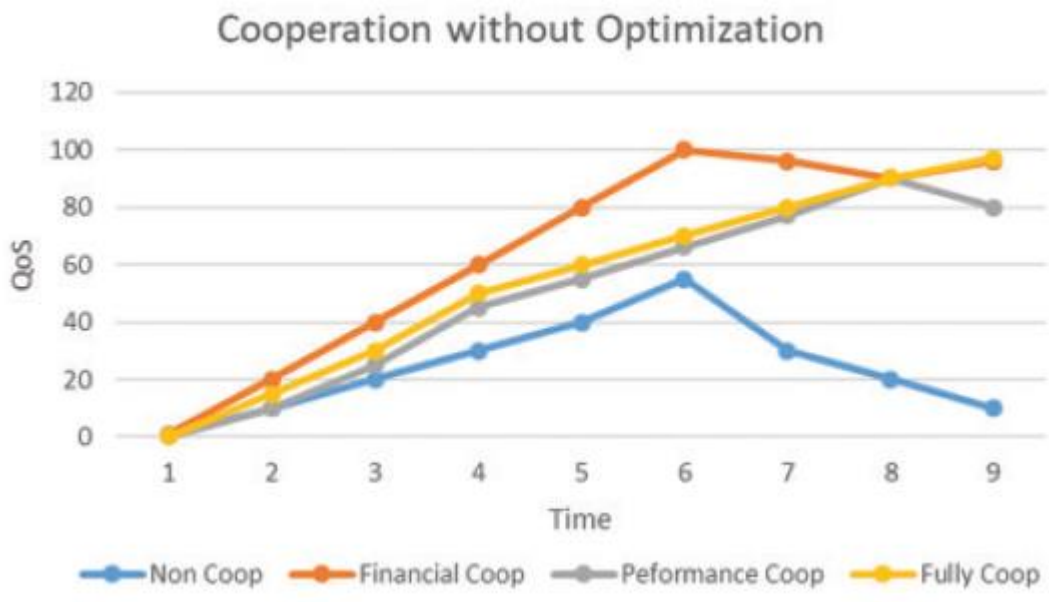

Рисунок 3.9 – Аналіз впливу співпраці

Також спостерігається роль SDC і втрати запиту. Результати показують, що втрата запитів зменшується після оптимізації, тоді як використання без запропонованої структури також зменшує втрату запитів. Тим не менш, вплив збільшується на 67% із включенням запропонованого

методу. Також важливо відзначити, що симуляція налаштована на години пік і велике навантаження. Тому можна з упевненістю сказати, що запропонована структура та параметри, визначені в цій структурі, є дійсними та працездатними для досягнення цілей дослідження та вирішення постановки проблеми.

Вплив оптимізації на співпрацю також аналізується за параметрами, що включають відмову від співпраці, фінансову співпрацю, співпрацю в продуктивності та повну співпрацю. Оскільки важливо оцінити, чи вплив оптимізації, який спостерігався в попередньому експерименті на розподіл ресурсів, на збільшення співпраці, відсутність співпраці тут є змінною через відсутність стратегії, обізнаності та зв'язку, що ізолює постачальників ресурсів. Результати показано на рисунку 3.10.

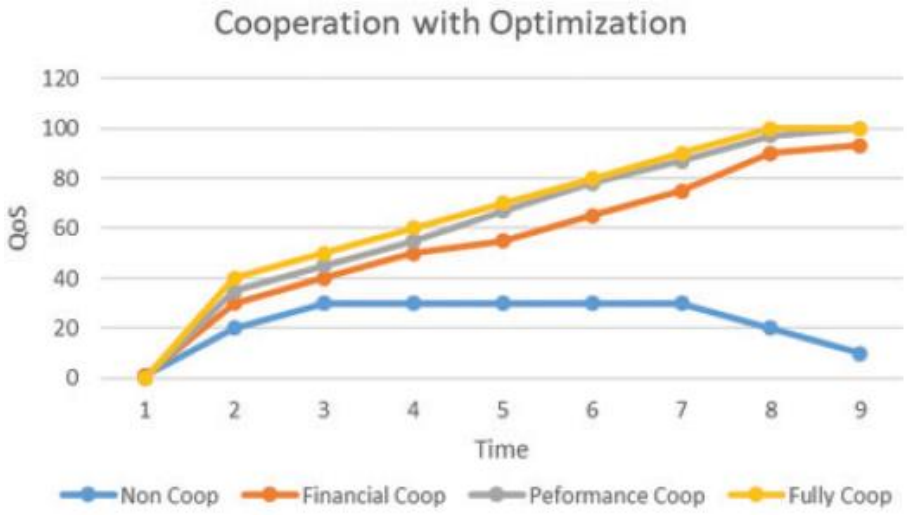

Рисунок 3.10 – Аналіз впливу співпраці з оптимізацією

Відмова від співпраці з часом зростала, але як тільки було досягнуто рівнязрілості, вона почала зменшуватися. Захоплюючим результатом є те, що співпраця завдяки фінансовим стимулам різко зростає, тоді як інші параметри, тобто співпраця на основі ефективності та повна співпраця, майже однакові. Це також підтверджує дослідження попереднього розділу про те, що фінансові стимули є найпоширенішою причиною залучення постачальників ресурсів до співпраці.

ВИСНОВКИ

Планування та розподіл ресурсів є основними визначальними факторами якості послуг у хмарних обчисленнях. Оскільки розподіл ресурсів в одному хмарному середовищі є складним, хмарні обчислення потребують ефективного модуля розподілу ресурсів. У той же час розподіл ресурсів у багатохмарному сценарії ще більше ускладнює процеси розподілу. Дотепер вимоги до завдань використовувалися для визначення того, як призначати ресурси з багатохмарного середовища. Процес оптимізації розподілу ресурсів базується на оптимальному використанні потужності ресурсів. Однак основні параметри використовувалися без підмножин, тобто обчислень, мережі та пам’яті для моделювання.

Результати моделювання виявляють роль різних змінних і факторів, які відіграють значну роль у якості обслуговування. Співпраця між постачальниками ресурсів для створення моделі функціонування, яка служить джерелом оптимальної якості обслуговування, базується на самій природі децентралізованої мультихмари, яка є другим компонентом процесу оптимізації. Результат цього моделювання також дав значні результати щодо параметрів співпраці та її впливу на якість обслуговування. Експериментальні результати демонструють, що наш алгоритм може збалансувати споживання всіх типів ресурсів, одночасно швидко й оптимально розподіляючи ресурси для неочікуваних потреб.

Управління корпоративними хмарними ресурсами відіграє ключову роль у узгодженні організаційних цілей із використанням ресурсів, забезпечуючи як довгострокове стратегічне узгодження, так і короткострокову операційну гнучкість. У міру того, як компанії стикаються зі складнощами оцифровки, ефективне управління ресурсами стає першорядним, що вимагає чітких організаційних структур, рольового контролю доступу та гнучких механізмів розподілу ресурсів. Еволюція

практики експлуатації та технічного обслуговування від традиційних до сучасних автоматизованих методів відображає мінливі вимоги цифрової ери. Хоча початковий ентузіазм щодо автоматизованих проектів з експлуатації та технічного обслуговування дав цінну інформацію, такі проблеми, як фрагментація інструментів і проблеми масштабованості, вимагали переходу до більш розумних і ефективних підходів. Технологія CloudOps стає ключовим інструментом, який використовує автоматизовані операції в хмарі для оптимізації управління ресурсами та підвищення ефективності роботи в корпоративному хмарному середовищі.

Таким чином, прагнення до автоматизації експлуатації та технічного обслуговування є невід'ємною частиною реалізації повного потенціалу IT-операцій підприємства. Застосовуючи стандартизовані підходи, передові технології та потужність даних, організації можуть орієнтуватися в складних умовах сучасних хмарних середовищ, одночасно підвищуючи ефективність, стійкість і конкурентоспроможність своїх операцій.

У цій роботі пропонується парадигма розподілу ресурсів у хмарних обчисленнях з метою покращення узгодження угоди про рівень обслуговування (SLA), планування завдань і розподілу віртуальних машин (VM). Ці модулі були оптимізовані за допомогою багатьох форм оптимізації роїв частинок, адаптованих до особливостей кожної ситуації. Під час узгодження угод про рівень обслуговування (SLA) між користувачами та кількома віддаленими центрами обробки даних із різною потужністю для покращення процесу використовується паралельний PSO. Завдяки застосуванню паралельного алгоритму PSO переговори можна автоматизувати, щоб заощадити час і зусилля, але при цьому отримати рішення прийнятної якості. У порівнянні з PSO та SPPSO, запропонований алгоритм для узгодження SLA скорочує час очікування приблизно на 30% і 20% відповідно. Якщо порівняти його з алгоритмом PSO, можна побачити приріст пропускної здатності приблизно на 20%. У порівнянні з PSO відсоток порушень SLA зменшується приблизно на чверть [26, 27].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. N. Tabassum, A. Ditta, T. Alyas, S. Abbas and M. A. Khan, “Prediction of cloud ranking in a hyperconverged cloud ecosystem using machine learning,” Computers, Materials & Continua, vol. 67, no. 3, pp. 3129–3141, 2021.

2. B. Rad, B. Bhatti and H. Ahmadi, “An introduction to docker and analysis of its performance,” International Journal of Computer Science and Network Security (IJCSNS), vol. 17, no. 3, pp. 228, 2017.

3. A. Martin, A. Raponi, S. Combe and D. Pietro, “Docker ecosystem-vulnerability analysis,” Computer Communications, vol. 122, no. 4, pp. 30–43, 2018.

4. K. Ye, H. Shen, Y.Wang and C. Xu, “Multi-tier workload consolidations in the cloud: Profiling, modeling and optimization,” IEEE Transactions on Cloud Computing, vol. 71, no. 3, pp. 1–9, 2020.

5. M. Shifrin, R. Mitrany, E. Biton and O. Gurewitz, “VM scaling and load balancing via cost optimal MDP solution,” IEEE Transactions on Cloud Computing, vol. 71, no. 3, pp. 41–44, 2020.

6. M. Ciavotta, G. Gibilisco, D. Ardagna, E. Nitto, M. Lattuada et al., “Architectural design of cloud applications: A performance-aware cost minimization approach,” IEEE Transactions on Cloud Computing, vol. 71, no. 3, pp. 110–116, 2020.

7. P. Kryszkiewicz, A. Kliks and H. Bogucka, “Small-scale spectrum aggregation and sharing,” IEEE Journal Selected Areas Communication, vol. 34, no. 10, pp. 2630–2641, 2016.

8. G. Levitin, L. Xing and Y. Xiang, “Reliability vs. vulnerability of N-version programming cloud service component with dynamic decision time under co-resident attacks,” IEEE Transaction Server Computing, vol. 1374, no. 3, pp. 1–10, 2020.

9. M. Aslanpour, M. Ghobaei and A. Nadjaran, “Auto-scaling web

applications in clouds: A cost-aware approach," Journal Network Computer Application, vol. 95, pp. 26–41, 2017.

10. T. He, A. N. Toosi and R. Buyya, "Performance evaluation of live virtual machine migration in SDNenabled cloud data centers," Journal of Parallel Distribution Computing, vol. 131, no. 3, pp. 55–68, 2019.

11. M. A. Altahat, A. Agarwal, N. Goel and J. Kozlowski, "Dynamic hybrid-copy live virtual machine migration: Analysis and comparison," Procedia Computer Science, vol. 171, no. 2019, pp. 1459–1468, 2019.

12. O. Alrajeh, M. Forshaw and N. Thomas, "Using virtual machine live migration in trace-driven energyaware simulation of high-throughput computing systems," Sustainable Computing Informatics System, vol. 29, no. August (12), pp. 100468, 2021.

13. S. Padhy and J. Chou, "MIRAGE: A consolidation aware migration avoidance genetic job scheduling algorithm for virtualized data centers," Journal of Parallel Distribution Computing, vol. 3, no. 12, pp. 1043–1055, 2021.

14. Z. Li, S. Guo, L. Yu and V. Chang, "Evidence-efficient affinity propagation scheme for virtual machine placement in the data center," IEEE Access, vol. 8, pp. 158356–158368, 2020.

15. T. Fukai, T. Shinagawa and K. Kato, "Live migration in bare-metal clouds," IEEE Transaction of Cloud Computing, vol. 9, no. 1, pp. 226–239, 2021.

16. Y. Chapala and B. E. Reddy, "An enhancement in restructured scatter-gather for live migration of the virtual machine," in Proc. of 6th Int. Conf. Invention Computing Technology ICICT 2021, NewYork, USA, no. 6, pp. 90–96, 2021.

17. N. Naz, S. Abbas, M. Adnan and M. Farrukh, "Efficient load balancing in cloud computing using multilayered mamdani fuzzy inference expert system," International Journal of Advanced Computer Science and Applications (IJACSA), vol. 10, no. 3, pp. 569–577, 2019.

18. L. Heilig, E. Lalla-Ruiz and S. Voß, "Modeling and solving cloud service purchasing in multi-cloudenvironments," Expert System Application, vol.

147, no. 3, pp. 113165, 2020.

19. M. Alaluna, E. Vial, N. Neves and F. M. V. Ramos, “Securemulti-cloud network virtualization,” Computer Networks, vol. 161, no. 4, pp. 45–60, 2019.

20. R.Rahim, “Comparative analysis of membership function on Mamdani fuzzy inference system for decision making,” Journal of Physics, vol. 930, no. 1, pp. 012029, 2017.

21. B. Liu, X. Chang, Z. Han, K. Trivedi and R. J. Rodríguez, “Model-based sensitivity analysis of IaaS cloud availability,” Future Generation Computer System, vol. 83, no. 7, pp. 1–13, 2018.

22. T. He, A. N. Toosi and R. Buyya, “SLA-aware multiple migration planning and scheduling in SDN-NFVenabled clouds,” Journal of Systems and Software, vol. 176, no. 4, pp. 110943–110950, 2021.

23. N. Iqbal, S. Abbas, M. A. Khan, T. Alyas, A. Fatima et al., “An RGB image cipher using chaotic systems, 15-puzzle problem, and DNA computing,” IEEE Access, vol. 7, pp. 174051–174071, 2019.

24. T. Alyas, I. Javed, A. Namoun, A. Tufail, S. Alshmrany et al., “Live migration of virtual machines using a mamdani fuzzy inference system,” Computers, Materials & Continua, vol. 71, no. 2, pp. 3019–3033, 2022.

25. N. Tabassum, T. Alyas, M. Hamid, M. Saleem and S. Malik, “Hyper-convergence storage framework for ecocloud correlates,” Computers, Materials & Continua, vol. 70, no. 1, pp. 1573–1584, 2022.

26. Volk Maksym, Buhrii Andrii, Kovtun Evgenii, Zhuravel Denys, Zborovskyi Mykhailo. Simulation and management of fog computing for IoT. 7-th International Scientific and Technical Conference "COMPUTER AND INFORMATION SYSTEMS AND TECHNOLOGIES". Kharkiv:NURE.–2024.− P.11-12

27. Волк М.О., Бугрій А.М., Ковтун Є.В., Самойлов І.А., Зборовський М.М. Моделі та методи ефективного управління ресурсами в системах хмарних обчислень. Проблеми інформатизації: Матеріали дванадцятої міжнародної науково-технічної конференції. –Баку – Харків – Бельсько-Бяла , 21 – 22 листопада 2024 року. С.50. doi: https://doi.org/10.32620/PI.24.t2