НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

«На правах рукопису»
УДК _004.932_

До захисту допущено:
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2024 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою
«Інтегровані інформаційні системи»

зі спеціальності 126 «Інформаційні системи та технології»

на тему: «Інтелектуальна система управління ресурсами в хмарних середовищах на основі машинного навчання»

Виконав:
Студент 2 курсу, групи ІА-32мп
Улізько Владислав Олександрович _____

Керівник:
доцент каф. ІСТ, к.т.н., доц.
Шимкович Володимир Миколайович _____

Рецензент:
Доцент каф. ОТ, к.т.н., доц.
Волокита Артем Миколайович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти – другий (магістерський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інтегровані інформаційні системи»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2024 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Улізьку Владиславу Олександровичу

1. Тема дисертації «Інтелектуальна система управління ресурсами в хмарних середовищах на основі машинного навчання», науковий керівник дисертації Шимкович Володимир Миколайович, к.т.н., доц., затверджені наказом по університету від «08» 11 2024 р. № 5016-с

2. Термін подання студентом дисертації «09»12 2024 р.

3. Об’єкт дослідження: процеси управління ресурсами в хмарних середовищах, їх оптимізація за допомогою машинного навчання.

4. Вихідні дані: швидкість відповіді не більше 1 секунди, максимальне використання CPU не більше 80% під час виконання задач, оновлення інформації про стан ресурсів і виконання передбачення кожні 5 секунд.

5. Перелік завдань, які потрібно розробити: провести аналіз сучасних методів управління ресурсами в хмарних середовищах, оцінку застосування машинного навчання для управління ресурсами, зокрема для прогнозування навантажень, розроблення архітектури системи управління ресурсами в хмарних середовищах та моделі для автоматичного розподілу ресурсів на основі машинного навчання.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: робоча архітектура гібридної моделі, блок схема хмарного середовища, діаграма

захисту хмарного середовища, діаграма класів процесів віртуалізації, діаграма класів хмарного середовища, діаграма компонентів захисту хмарних систем, діаграма компонентів захисту хмарних систем, діаграма використання хмарного середовища.

7. Орієнтовний перелік публікацій: не заплановано

8. Дата видачі завдання: 02.09.2024 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Аналіз предметної області та огляд існуючих рішень	03.09.2024-20.09.2024	
2.	Аналіз методів машинного навчання для прогнозування навантажень у хмарних середовищах	23.09.2024-06.10.2024	
3.	Розробка архітектури інтелектуальної системи управління ресурсами	09.10.2024-20.10.2024	
4.	Реалізація моделі машинного навчання (SVM) та розробка алгоритмів прогнозування навантажень	23.10.2024-10.11.2024	
5.	Тестування та оцінка точності розробленої моделі машинного навчання	13.11.2024-20.11.2024	
6.	Оформлення пояснювальної записки	21.11.2024-24.11.2024	
7.	Попередній захист	25.11.2024	
8.	Захист		

Студент Владислав УЛІЗЬКО

Науковий керівник Володимир ШИМКОВИЧ

РЕФЕРАТ

Інтелектуальна система управління ресурсами в хмарних середовищах на основі машинного навчання: 116с., 22 табл., 24 рис., 9 дод., 49 джерел.

PYTHON, ХМАРНІ ТЕХНОЛОГІЇ, МАШИННЕ НАВЧАННЯ, НАВЧАННЯ МОДЕЛЕЙ, АВТОМАТИЗАЦІЯ, ХМАРНЕ УПРАВЛІННЯ РЕСУРСАМИ, ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ.

Хмарні обчислення – це нова модель віддалених обчислень, яка відкриває доступ до багатьох джерел. У сучасну епоху завдяки перевагам високої доступності, меншій вартості послуг, більшій доступності та масштабованості ресурсів хмарні обчислення стають найбільш підходящою платформою. Вони надають обчислювальні послуги на основі Інтернету, керуючи та використовуючи велику кількість ресурсів. Термін «управління ресурсами» вказує на можливості обчислювального середовища і фокусується на тому, як вони надають послуги іншим користувачам, додаткам і організаціям.

Метою роботи є підвищення ефективності систем управління ресурсами в хмарних середовищах на основі методів машинного навчання для оптимізації використання ресурсів. Завданням на роботу є аналіз сучасних методів управління ресурсами в хмарних середовищах, оцінка застосування машинного навчання для управління ресурсами, зокрема для прогнозування навантажень, розроблення архітектури системи управління ресурсами в хмарних середовищах та моделі для автоматичного розподілу ресурсів на основі машинного навчання.

Об’єктом дослідження є процеси управління ресурсами в хмарних середовищах, їх оптимізація за допомогою машинного навчання.

Предметом дослідження є алгоритми машинного навчання, що використовуються для прогнозування навантажень і оптимізації розподілу ресурсів у хмарних середовищах.

Результати дослідження можуть бути застосовані в різних галузях, таких як IT, телекомунікації та електронна комерція.

ABSTRACT

Intelligent Resource Management System in Cloud Environments Based on Machine Learning: 116 p., 22tab., 24draw., 9app., 49sources.

PYTHON, CLOUD TECHNOLOGIES, MACHINE LEARNING, MODEL TRAINING, AUTOMATION, CLOUD RESOURCE MANAGEMENT, INTELLIGENT SYSTEMS.

Cloud computing is a modern paradigm of remote computing that provides access to a wide range of resources. In today's era, due to its advantages such as high availability, lower service costs, increased accessibility, and resource scalability, cloud computing has become the most suitable platform. It delivers Internet-based computing services by managing and utilizing vast amounts of resources. The term "resource management" refers to the capabilities of the computing environment and focuses on how resources are allocated to other users, applications, and organizations.

The aim of this work is to develop an intelligent resource management system for cloud environments based on machine learning methods to optimize resource utilization. The tasks of this work include analyzing current methods of resource management in cloud environments, assessing the application of machine learning for resource management, particularly for workload prediction, and developing a system architecture for resource management in cloud environments and a model for automated resource allocation based on machine learning.

The object of the study is the processes of resource management in cloud environments and their optimization through machine learning.

The subject of the study is machine learning algorithms used for workload prediction and resource allocation optimization in cloud environments.

The research results can be applied in various fields such as IT, telecommunications, and e-commerce

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

AI – ArtificialIntelligence – штучний інтелект

API – Application Programming Interface – програмний інтерфейс додатків

CPU – CentralProcessingUnit – центральний процесор

GPU – Graphics Processing Unit – графічний процесор

IaaS – Infrastructure as a Service – інфраструктура як послуга

K^2– R-squared – коефіцієнт детермінації

LSTM – LongShort-TermMemory – нейронна мережа з довгою короткочасною пам’яттю

MAE – Mean Absolute Percentage Error – середня абсолютна похибка

MAPE – MeanAbsolute–середня абсолютна процентна похибка

ML – Machine Learning – машинне навчання

NaN – NotaNumber – відсутнє числове значення

PaaS – Platform as a Service – платформа як послуга

RAM – Random Access Memory – оперативна пам’ять

RF – RandomForest –штучний інтелект

RM – ResourceManagement – управління ресурсами

SaaS – SoftwareasaService – програмне забезпечення як послуга

ВСТУП

У липні 2020 року, згідно з демографічним дослідженням, проведеним компанією Statista, приблизно 4,57 мільярда користувачів були активними в Інтернеті, що охоплює 59% світового населення. Така величезна кількість користувачів свідчить про високу потребу у відмово стійких, безпечних та легко конфігурованих веб-додатках. До ери хмарних обчислень, з точки зору підприємств, витрати на утримання великих центрів обробки даних та розширення компанії були занадто високими. Хмарні обчислення, як надання віртуальних ресурсів з центральних систем, з'явилися в 1950-х роках в освітніх інститутах і компаніях, коли їх використання здійснювалося за допомогою серверів з великими обчислювальними можливостями і можливостями зберігання даних.

Артур Семюел у 1959 році ввів термін «машинне навчання» (Machine Learning, ML). Він був піонером у галузі комп'ютерних ігор та штучного інтелекту. Машинне навчання – це здатність машин навчатися на основі даних. МН дозволяє комп'ютерам дізнаватися, як виконувати завдання, не будучи запрограмованими в явному вигляді.

Він впевнено прокладає свій унікальний шлях у різних сферах корпоративних додатків, таких як виявлення шахрайства, бізнес-розвідка та підтримка клієнтів. За останні роки Microsoft, Amazon та Google зробили значні інвестиції в машинне навчання та штучний інтелект (ШІ), запустивши нові сервіси для проведення важливих реорганізацій, які стратегічно розмістять штучний інтелект в їхніх організаційних структурах. Навіть генеральний директор Google Сундар Пічаї зазначив, що Google переходить до «світу, де переважає штучний інтелект». Деякі автори зазначають, що на апаратне забезпечення впливають робочі навантаження на ШІ. Для того, щоб навчити модель розуміти мову або розпізнавати шаблон, потрібні значні паралельні обчислювальні ресурси, що може зайняти дні на традиційних комп'ютерах на базі центрального процесора. Можна сказати, що потужні графічні процесори (GPU) є першим вибором для

обчислювального блоку в багатьох робочих навантаженнях машинного навчання та штучного інтелекту завдяки значно скороченому часу обробки.

Предметом дослідження є алгоритми машинного навчання, що використовуються для прогнозування навантажень і оптимізації розподілу ресурсів у хмарних середовищах.

Об’єктом дослідження є процеси управління ресурсами в хмарних середовищах, їх оптимізація за допомогою методів машинного навчання.

Метою роботи є розробка інтелектуальної системи управління ресурсами в хмарних середовищах для оптимізації їх використання шляхом застосування методів машинного навчання, таких як прогнозування навантаження та динамічний розподіл ресурсів.

Завдання на роботу є проведення огляду сучасних систем управління ресурсами у хмарних середовищах, проведення аналізу методів машинного навчання в управлінні ресурсами, розроблення архітектури системи управління ресурсами в хмарних середовищах та моделі для автоматичного розподілу ресурсів на основі машинного навчання.

Результати дослідження можуть бути використані для підвищення продуктивності хмарних систем у різних галузях, зокрема в IT, телекомунікаціях та електронній комерції.

1 ОГЛЯД СУЧАСНИХ СИСТЕМ УПРАВЛІННЯ РЕСУРСАМИ В ХМАРНИХ СЕРЕДОВИЩАХ

1.1 Основні підходи до управління ресурсами в хмарних обчисленнях

За останні кілька років світ обчислень зазнав значного розвитку завдяки постійному зростанню попиту на високопродуктивні обчислювальні пристрої [1]. Ця еволюція призвела до появи нових обчислювальних парадигм, таких як кластерні обчислення, грід-обчислення та хмарні обчислення. Серед цих парадигм хмарні обчислення набули значної популярності [2]. Хмарні обчислення пропонуються у трьох формах, а саме:

– публічна хмара;
– приватна хмара та;
– гібридна хмара [1,3,4].

Крім того, надаються різні набори послуг, які можна умовно поділити на три категорії, а саме:

– програмне забезпечення як послуга (SaaS);
– інфраструктура як послуга (IaaS);
– платформа як послуга (PaaS) [1].

Хмарні обчислення базуються на сервіс-орієнтованій архітектурі (SOA), яка використовує концепції віртуалізації та розподілених обчислень [1]. У хмарних обчисленнях SOA доступ до спільного пулу ресурсів надається через мережу, а наявні ресурси можуть бути сконфігуровані відповідно до вимог користувачів [3].

Одним з ключових аспектів хмарних обчислень та віртуалізації є управління ресурсами (Resource Management, RM). RM – це процес, який займається закупівлею та вивільненням ресурсів [2]. Методи віртуалізації використовуються для гнучкого надання ресурсів на вимогу [3]. Для цього для кожного отриманого завдання створюється або нова ВМ, або воно розміщується на існуючій ВМ того ж користувача [2]. Після виконання завдання всі отримані ресурси звільняються і стають частиною пулу вільних ресурсів. Виділення ресурсів здійснюється на основі

угоди про рівень обслуговування (Service Level Agreement, SLA), яка узгоджується з постачальником послуг.

SLA містить детальну інформацію про рівень обслуговування, який потрібен орендарю. Крім того, він містить інформацію про процес оплати та штрафні санкції за порушення SLA [5].

Оскільки парадигма хмарних обчислень орієнтована на ринок, традиційні системно-орієнтовані архітектури РМ не підходять для таких систем [5]. Системно-орієнтовані архітектури не надають жодних стимулів постачальникам послуг і розглядають всі запити з однаковою важливістю. Отже, для хмарних середовищ розробляються чіткі ринково-орієнтовані методи, які здатні задовольнити потреби у наданні ресурсів на вимогу та розподілі ресурсів між користувачами. Такі методи RM забезпечують економічні стимули як для постачальників послуг, так і для клієнтів. Постачальники послуг ділять свої ресурси між кількома орендарями на основі оплати за використання.

RM розглядається як один з важливих аспектів хмарних обчислень для забезпечення ізоляції продуктивності та ефективного використання базового обладнання [6].У хмарних середовищах майже всі ресурси є віртуалізованими і поділяються між декількома користувачами [2]. Віртуалізація ставить деякі складні завдання, пов'язані з управлінням ресурсами. Основними дослідницькими проблемами/показниками RM є енергоефективність, порушення SLA, балансування навантаження, навантаження на мережу, максимізація прибутку, гібридні хмари та мобільні хмарні обчислення (МХО).

Запропоновані метрики RM супроводжуються численними новими викликами, наприклад, енергоефективність та навантаження на мережу призводять до порушень SLA. Аналогічно, зменшення порушень SLA збільшує споживання енергії та перешкоджає збільшенню прибутку. Крім того, RM, розроблені для публічних або приватних хмар, не можуть бути безпосередньо застосовані до гібридних та мобільних хмар через різницю в архітектурі сервісів.

Багатокритеріальна оптимізація, як правило, забезпечує найкраще можливе рішення шляхом одночасної оптимізації декількох метрик RM. Більше того,

багатокритеріальна оптимізація може додати нові виміри та виклики в RM, якщо розглянуті метрики є суперечливими. Це пов'язано з тим, що визначальні метрики взаємозалежні, і оптимізація однієї метрики погіршує продуктивність іншої метрики. Декілька прикладів конкуруючих показників: енергоефективність і порушення SLA, енергоефективність і навантаження на мережу, а також порушення SLA і максимізація прибутку.На рисунку 1.1 зображено таксономію методів РМ.

Рисунок 1.1 – Таксономія методів РМ[7]

Методи класифікуються на декілька категорій залежно від досліджуваної проблеми та використовуваної метрики. У випадку багатокритеріальних оптимізованих рішень, категорія метрики визначається на основі високофокусованої метрики.

Енергоефективність є одним з ключових питань, які необхідно вирішувати в хмарних системах [8]. Це пов'язано з тим, що споживання енергії не лише збільшує витрати на електроенергію постачальників послуг, але й відіграє певну роль у збільшенні викидів CO2 [9]. Більше того, машини та викопна нафта, що використовуються для виробництва електроенергії, є основними джерелами викидів CO2. Одним з потенційних шляхів вирішення цієї проблеми є консолідація робочого навантаження, коли споживання енергії мінімізується шляхом консолідації більшого робочого навантаження на меншій кількості серверів. Віртуальні машини, розміщені на серверах з низьким навантаженням, переміщуються на сервери з порівняно вищим навантаженням, а сервери, що

простоюють, вимикаються. Деякі з існуючих методів енергоефективної РМ обговорюються в наступному розділі.

Модифіковане найкраще відповідне зменшення (Modified Best Fit Decreasing, MBFD) [10] базується на алгоритмі Best Fit Decreasing (BFD), який використовується для бін-пакування. Алгоритм MBFD сортує віртуальні машини в порядку спадання на основі їхніх вимог до процесора. Після сортування всі ВМ призначаються хостам на основі моделі енергоспоживання. Модель енергоспоживання перевіряє зміну енергоспоживання (Pu) серверів і розміщує ВМ на сервері, який показує мінімальну зміну енергоспоживання. Крім того, вона використовує динамічні порогові значення, щоб утримувати використання сервера в межах діапазону та уникати порушень SLA.

Нижнє порогове значення вказує на те, що сервер недовикористовується, тоді як верхнє порогове значення попереджає постачальників послуг про можливе порушення SLA. Якщо будь-яке з порогових значень порушено, то віртуальна машина або набір віртуальних машин мігрує на інший сервер (сервери).

Традиційний BFD призначає ВМ або на сервери з мінімальною обчислювальною потужністю, або на сервери з максимальною невикористаною потужністю. Проблема алгоритму BFD полягає в тому, що він не враховує енергоспоживання серверів під час розміщення ВМ. Таким чином, ВМ може бути розміщена на сервері, який має низьку обчислювальну потужність, але споживає багато енергії. Отже, алгоритм BFD не є енергоефективним і потребує оптимізації. Для вирішення цієї проблеми автори в [11] пропонують енергоефективне рішення на основі BFD. BFD з урахуванням потужності та обчислювальної потужності (PCA-BFD) вирішує зазначену проблему, призначаючи ВМ на сервер, який має найбільшу обчислювальну потужність. Таким чином, PCA-BFD мінімізує споживання енергії та кількість використовуваних серверів.

У роботі [12] зосереджено увагу на архітектурі PaaS хмарних систем. Основна увага приділяється мінімізації споживання енергії та часу відгуку, а також максимізації доступності. Запропонована методика RM базується на розподіленій

ієрархічній структурі, описаній в [13], в якій нелінійна оптимізація ресурсів здійснюється за різними часовими шкалами.

Рішення RM приймаються щогодини, щоб мінімізувати накладні витрати, пов'язані з цими рішеннями. Рішення RM включають в себе увімкнення, вимкнення сервера та міграцію ВМ з одного сервера на інший, і споживають значну кількість енергії та мережевих ресурсів. Отже, ці операції не можуть виконуватися дуже часто. Всі вищеописані методи розроблені для того, щоб мінімізувати споживання енергії на стороні постачальника послуг. У роботі [11] алгоритм PCABFD зосереджуються лише на споживанні енергії і не розглядають інші проблеми PM, такі як порушення SLA, навантаження на мережу, максимізація прибутку та балансування навантаження. Однак у роботі [10] запропоновано алгоритм MBFD, який також спрямований на мінімізацію порушень SLA, що виникають через консолідацію робочого навантаження. У роботах [12,14] увагу зосереджено на максимізації доходу разом із оптимізацією енергоспоживання. Хоча вищезгадані методи працюють з багатьма проблемами RM, але все ще є можливості для вдосконалення.

Якість обслуговування (QoS) є одним з важливих питань, які слід враховувати при управлінні ресурсами [16]. Оскільки постачальники послуг мають намір надавати своїм клієнтам найкращі послуги, між обома сторонами підписується угода про рівень обслуговування (SLA). SLA містить інформацію про необхідний рівень обслуговування та ціну. SLA також містить положення про штрафні санкції, які застосовуються у разі порушення угоди. Постачальники послуг повинні уникати порушень і здійснювати контроль під час надання послуг клієнтам. Для вирішення цієї проблеми різні дослідники пропонують різні рішення, деякі з яких розглядаються далі.

На рисунку 1.2 зображено розподілену ієрархічну структуру.

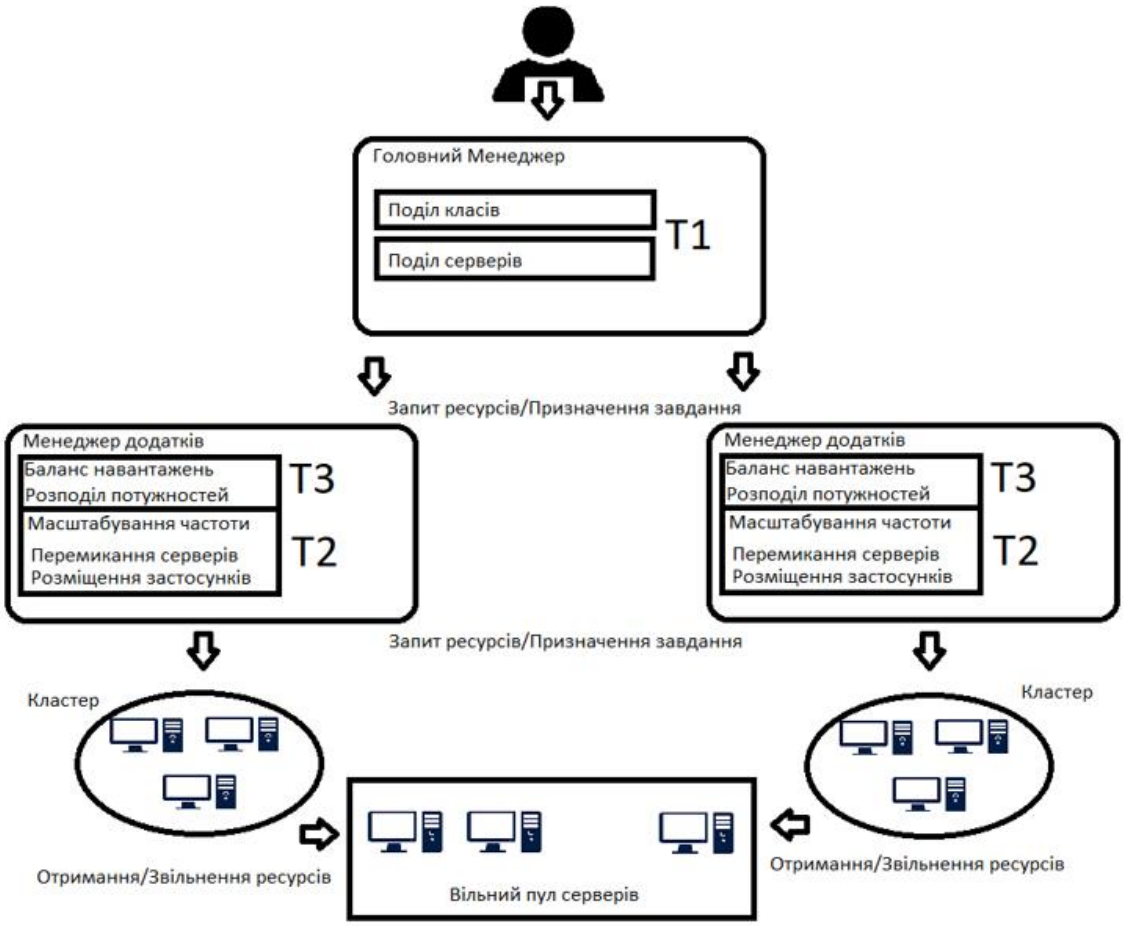

Рисунок 1.2 – Розподілена ієрархічна структура[15]

У роботі [17] запропоновано алгоритми розподілу потужностей для забезпечення SLA та обробки мінливих робочих навантажень. Запропоновані алгоритми взаємодіють з географічно розподіленими контролерами ресурсів, а також можуть перенаправляти навантаження при виникненні перевантаження в мережі. Більше того, за необхідності, додаток запускається на декількох віртуальних машинах, а робоче навантаження рівномірно розподіляється між ними. Оскільки робоче навантаження є мінливим, для прогнозування майбутніх вимог до навантаження використовується предиктор навантаження, і потужність змінюється на основі отриманих прогнозів. Крім того, порушення SLA вдається уникнути завдяки низькому часу відгуку під час взаємодії між віртуальними машинами. У випадку збільшення часу відгуку, ВМ мігрує на іншу фізичну машину.

Основна мета запропонованих алгоритмів полягає в тому, щоб задовольнити мінливі робочі навантаження, зберігаючи при цьому низький рівень порушень SLA.

У роботі [18] основна увага зосереджена на мінімізації ймовірності недостатнього або надлишкового надання ресурсів, спричиненого реактивним управлінням ресурсами (RM).Запропонований алгоритм базується на фреймворку, розглянутому у роботі [19], який розподіляє ресурси між різними завданнями. Алгоритм використовує різні агенти для виділення та припинення використання ресурсів. Крім того, він має модуль прогнозування, який прогнозує майбутні потреби сервісів на основі. Він також має можливість керувати ресурсами між різними сервісами, щоб уникнути непотрібного розподілу та вивільнення ресурсів.

1.2 Архітектура хмарних систем

У 1950-х роках вперше з'явилися прототипи хмарних систем в освітніх інститутах і компаніях, а його використання стало можливим завдяки серверам з великими обчислювальними можливостями і можливостями зберігання даних [18]. На рисунку 1.3 зображено основні категорії хмарних сервісів.

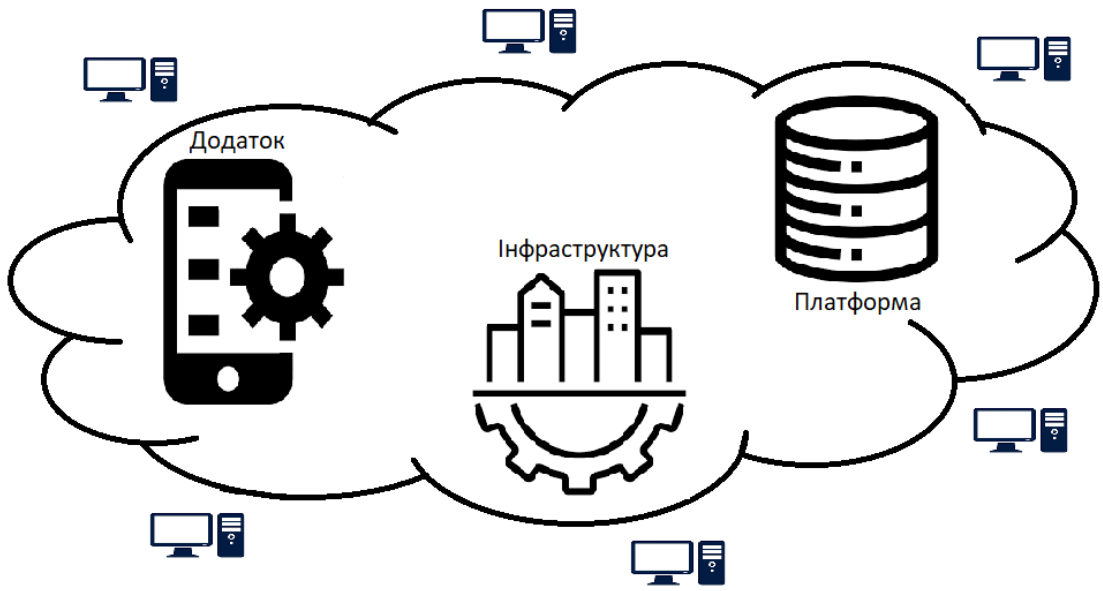

Рисунок 1.3 – Основні категорії хмарних сервісів[20]

Програмне забезпечення як послуга (SaaS): Існує вже багато програм, які надаються як SaaS, наприклад, CRM, ERP, Word тощо. Отже, основна концепція полягає в тому, щоб не встановлювати програмне забезпечення на комп'ютері або серверах клієнта, а отримати доступ до них через Інтернет. Таким чином, процес випуску та виправлення помилок буде простішим [17, 18].

Платформа як послуга (PaaS): Існує також схожа концепція, як і SaaS, але в PaaS як альтернатива необхідності купувати - платити за ліцензії на програмне забезпечення для платформ, таких як операційні системи, бази даних та проміжне програмне забезпечення, клієнт може зробити це, використовуючи платформу та інструменти [17, 18].

Інфраструктура як послуга (IaaS): IaaS має ще більші масштаби і більше застосування. Таким чином, замість того, щоб володіти і обслуговувати власні апаратні пристрої в центрі обробки даних, ви можете отримати доступ до них через Інтернет, увійшовши в систему з офіційними обліковими даними, і вони можуть бути розподілені по всій планеті. Ці апаратні пристрої можуть бути мережевими пристроями, віртуальними машинами, пристроями зберігання даних тощо [17,18,21].

Зобов'язання між постачальником послуг і клієнтом, що називається Угода про рівень обслуговування (SLA). У роботі [16] зазначено, що деякі аспекти якості послуг, відповідальності та доступності повинні бути узгоджені між користувачем і постачальником послуг. SLA зазвичай містить багато компонентів, від визначення послуг до розірвання угоди.

Крім того, SLA визначається на різних рівнях:

– представлено таксономію методів PM, яка базується на основних метриках PM. Метрики дослідження, які використовуються для надання рішень RM, включають енергоефективність, дотримання SLA, мінімізацію навантаження на мережу, балансування навантаження, максимізацію прибутку, гібридні хмарні обчислення та мобільні хмарні обчислення;

– багаторівневий SLA: це поділ на різні рівні, кожен з яких адресований різним групам клієнтів для одних і тих же послуг, в рамках одного SLA. Розглядаємо SLA корпоративного рівня, SLA на рівні клієнта та SLA на рівні сервісу;

– SLA на рівні клієнта: індивідуальна угода в групі клієнтів, яка містить всі послуги, якими вони користуються;

– SLA на рівні послуг: загальна угода з усіма клієнтами, які користуються послугами, що надаються постачальником послуг.

Якість обслуговування (QoS) – це вимірювання або опис загальної продуктивності послуги, наприклад, послуги хмарних обчислень або комп'ютерної мережі, зокрема, продуктивності, яку бачать користувачі мережі. Багато компонентів мережевої послуги часто розглядаються для вимірювання якості послуги, таких як доступність, затримка передачі, втрата пакетів тощо.

У роботах [22, 23] зазначено, що у 1959 році Артур Семюел ввів термін «машинне навчання» (Machine Learning, ML). Він був піонером у галузі комп'ютерних ігор та штучного інтелекту. Машинне навчання – це здатність машин навчатися на основі даних. МН дозволяє комп'ютерам дізнаватися, як виконувати завдання, не будучи запрограмованими в явному вигляді.

Він впевнено прокладає свій унікальний шлях у різних сферах корпоративних додатків, таких як виявлення шахрайства, бізнес-розвідка та підтримка клієнтів. За останні роки Microsoft, Amazon та Google зробили значні інвестиції в машинне навчання та штучний інтелект (ШІ), запустивши нові сервіси для проведення важливих реорганізацій, які стратегічно розмістять штучний інтелект в їхніх організаційних структурах. Навіть генеральний директор Google Сундар Пічаї зазначив, що Google переходить до «світу, де переважає штучний інтелект». Деякі автори зазначають, що навантаження на апаратне забезпечення впливає на роботу ШІ. Для того, щоб навчити модель розуміти мову або розпізнавати шаблон, потрібні значні паралельні обчислювальні ресурси, що може зайняти дні на традиційних комп'ютерах на базі центрального процесора. Можна сказати, що потужні графічні процесори (GPU) є першим вибором для

обчислювального блоку в багатьох завданнях машинного навчання та ШН завдяки значному скороченню часу обробки.

Надання ресурсів включає в себе розгортання, вибір та управління апаратними та програмними ресурсами під час виконання для забезпечення гарантованої продуктивності додатків. Хмарні провайдери пропонують два варіанти надання ресурсів. Перший – це короткострокове надання ресурсів на вимогу, а другий – довгострокові плани резервування. У великомасштабних розподілених системах, з якими пов'язане резервування ресурсів, існують важливі та складні проблеми. У роботі [18] зазначено, що деякі методи доставки ресурсів повинні відповідати параметрам якості обслуговування (QoS), таким як безпека, доступність, надійність, зокрема уникаючи порушень угоди про рівень обслуговування (SLA) та враховуючи довгострокові плани резервування.

У роботі [3] запропоновано метод прогнозування та вивчення моделей робочого навантаження в реальному часі на архітектурі мікросервісу, що використовує ML-моделі. Для демонстрації цієї архітектури автори використали платформу AWS та два алгоритми ML: логістичну регресію і багатокласову класифікацію. Моделі, застосовані для прогнозування, включали лінійну регресію та мультиноміальну логістичну регресію. Прогнозування в реальному часі було інтегровано в конфігурацію автоматичного масштабування для динамічного додавання або видалення ресурсів.

У роботі [24] запропоновано стратегії вимірювання та забезпечення ресурсів на основі прогнозування з використанням нейронних мереж (NN) та моделей лінійної регресії для задоволення майбутніх потреб у ресурсах. Автори також розробили методи прогнозування на основі набору даних, отриманих за допомогою TPC-W — еталонного показника, що добре зарекомендував себе для додатків електронної комерції.

Моделі прогнозування хмарних клієнтів за допомогою машинного навчання також широко досліджували у роботі [25] за допомогою веб-додатку TPC-W для надання ресурсів з використанням трьох різних технологій машинного навчання. Серед лінійної регресії (LR), нейронних мереж (NN) та опорних векторів

виявилося, що SVM є найкращою моделлю прогнозування, оскільки фокусується на завантаженні процесора та SLA (пропускна здатність та час відгуку) у вікні 9-12 хвилин.

На основі прогнозування обчислювального навантаження, яке може мати розподілений сервер, та оцінки відповідної кількості ресурсів, які необхідно забезпечити, створено систему прогнозування з автомасштабуванням, використовуючи методи ML для теорії черг та прогнозування часу [10]. Для прогнозування була використана SVM-регресійна модель, яка виявилася ідеальною для вхідних даних з лінійними та нелінійними моделями.

Згідно з наведеними вище дослідженнями, здається, що для прогнозування майбутніх потреб у ресурсах використовується комбіноване використання ресурсного забезпечення та ML. Найбільш поширеними методами ML, результати яких є перспективними та ефективними, є лінійна регресія (LR), нейронні мережі (NN) та машина опорних векторів (SVM).

1.3 Виклики та проблеми оптимізації ресурсів

Хмарні обчислення – це нова модель віддалених обчислень, яка відкриває доступ до багатьох джерел. У сучасну епоху завдяки перевагам високої доступності, меншій вартості послуг, більшій доступності та масштабованості ресурсів хмарні обчислення стають найбільш підходящою платформою. Вони надають обчислювальні послуги на основі Інтернету, керуючи та використовуючи велику кількість ресурсів. Термін «управління ресурсами» вказує на можливості обчислювального середовища і фокусується на тому, як вони надають послуги іншим користувачам, додаткам і організаціям. Ці ресурси надаються на вимогу через Інтернет за допомогою прикладних програм [11,12,26,27]. Хмарні обчислення базуються на розподілених і паралельних обчисленнях [14]. Для глобального розміщення ресурсів та їх економного використання використовуються розподілені обчислення. Коли попит користувачів змінюється, ресурси також змінюються відповідно. Через динамічну природу обчислювального

середовища стає дуже складно працювати з великою кількістю ресурсів. Хмарні обчислення мають багато проблем з розподілом ресурсів, включаючи інфраструктуру та відповідні сервіси. У роботі [1] зазначено, що планування ресурсів є основною проблемою хмарних обчислень.

Ще однією важливою проблемою в середовищі хмарних обчислень є створення методів управління ресурсами, які використовуються для визначення розміщення кожного модуля [10] з іншими пристроями для збільшення пропускної здатності та зменшення латентності. Центри обробки даних у хмарних обчисленнях [25] мають проблеми з реалізацією методів міграції та управління ресурсами.

Основними компонентами хмари є користувачі/бокери сервіси, сховище, додатки, платформа та інфраструктура [21, 22]. Потік розподілу ресурсів у хмарному середовищі починається з ідентифікації запитів від користувачів. Ці запити обробляються таким чином, щоб задовольнити певні цілі хмарного провайдера. Цілями в обчислювальних середовищах можуть бути оптимізація витрат [23] або оптимізація енергоспоживання [28] тощо. На основі інформації про ресурси, наприклад, моніторингу ресурсів, брокери (хмарний провайдер) запитують інформацію, а планувальник визначає рішення для розподілу ресурсів. Розподілювач ресурсів може забезпечити лише статичне та початкове виділення ресурсів після отримання будь-якого запиту або підтвердити як динамічне [29], так і статичне виділення [30] ресурсів для управління ними.

Планувальники можуть просто забезпечити початковий і статичний розподіл ресурсів [31] після надходження запиту або забезпечити як статичний, так і динамічний розподіл ресурсів для безперервного управління ресурсами, а також для подальшої оптимізації та коригування старих запитів. Більш широке впровадження технологій віртуалізації та хмарних обчислень призвело до розширення розмірів кластерів за рахунок збільшення кількості вузлів для малих та великих кластерних центрів відповідно [32].

Зважаючи на високе споживання електроенергії [33] та збільшення вуглецевого сліду [34], енергоефективність стає надзвичайно важливою для хмарних обчислень та центрів обробки даних. Тому вирішення питання розподілу

ресурсів в обчислювальному середовищі є великою проблемою [35]. Це питання також називають NP-важким [4, 36], що обговорюється в контексті хмарних обчислень. На рисунку 1.4 зображено розподіл ресурсів у хмарних середовищах.

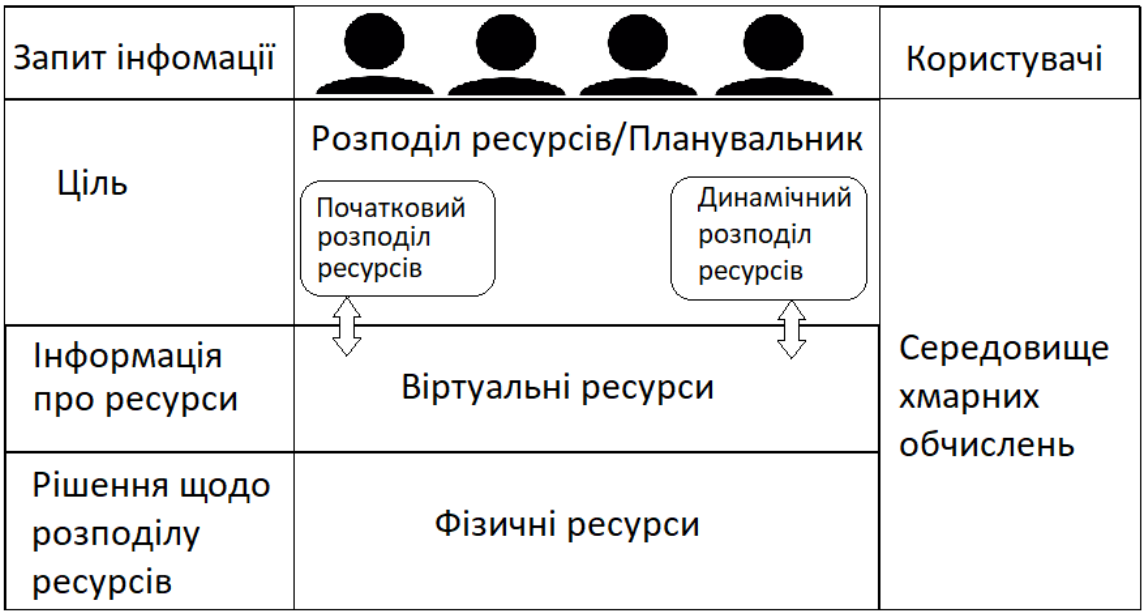

Рисунок 1.4 – Розподіл ресурсів у хмарних обчисленнях[37]

Однією з основних проблем у хмарних обчисленнях є розподіл ресурсів, оскільки через динамічну природу ресурсів [38] кінцеві користувачі можуть отримати доступ до ресурсів повсюдно (в будь-який час і в будь-якому місці). При цьому хмарні дата-центри надають велику кількість ресурсів, а обчислювальні моделі хмар здатні підтримувати адаптивний розподіл ресурсів на вимогу. Однак, така велика кількість ресурсів також призводить до неоптимального їх розподілу. Метою розподілу ресурсів для будь-якого хмарного брокера може бути або оптимізація додатків [39, 40], або ефективність використання енергії, або покращення використання ресурсів незалежно від того, які саме ІКТ-ресурси: еластична IP, блокові сховища та сервери [41] надаються кінцевим користувачам. Деякі з енергоефективних проблем з розподілом ресурсів [42, 43], які були виявлені в дослідженні, включають:

– представлено таксономію методів РМ, яка базується на основних метриках РМ. Метрики дослідження, які використовуються для надання рішень RM,

включають енергоефективність, дотримання SLA, мінімізацію навантаження на мережу, балансування навантаження, максимізацію прибутку, гібридні хмарні обчислення та мобільні хмарні обчислення;

– з різних робочих навантажень вибрати найбільш підходящу категорію завад і робочого навантаження, таких як споживання енергії, продуктивність і використання ресурсів;

– оцінка структури хмарної інфраструктури шляхом вдосконалення технологій та інструментів, топології та ресурсів хмари;

– з різних робочих навантажень вибрати найбільш підходящі перешкоди та категорії навантаження, такі як споживання енергії, продуктивність та використання ресурсів;

– сприяти консолідації ресурсів, оцінюючи взаємозалежності додатків для повернення інвестицій та зростання продуктивності;

– оцінка стандартизованих, об'єднаних та централізованих ресурсів центрів обробки даних для забезпечення та використання ресурсів під час роботи;

– відновлення після будь-яких збоїв: покращення доступності мережі, використання активів, ефективності енергоспоживання та скорочення часу використання ресурсів;

– планування практичної інфраструктури хмари завдяки підтримці безпеки бізнесу та еластичності для критично важливих заявок.

Термін «ресурс» у хмарних обчисленнях визначається як щось на зразок сховища, процесора, пропускної здатності та пам'яті (ІКТ-ресурси) [44]. Проблема розподілу ресурсів є дуже складною [45] і вимагає деяких припущень, таких як:

– вартість використаних ресурсів;

– зменшення втрат енергії;

– набір активних серверів.

Крім того, хмарні брокери розподіляють ресурси системи між процесорами і обмежують прийом вхідних запитів, виходячи з доступності ресурсів.

Багато функцій, таких як:

– спостереження за доступністю ресурсів;

– відстеження вимог сервісів

– спостереження за вимогами користувачів до сервісів;

– визначення вартості ресурсів;

– обробка запитів на надання послуг;

– відстеження фактичного розподілу ресурсів роблять це складним і важким завданням.

На рисунку 1.5 зображено проблеми розподілу ресурсів у хмарі.

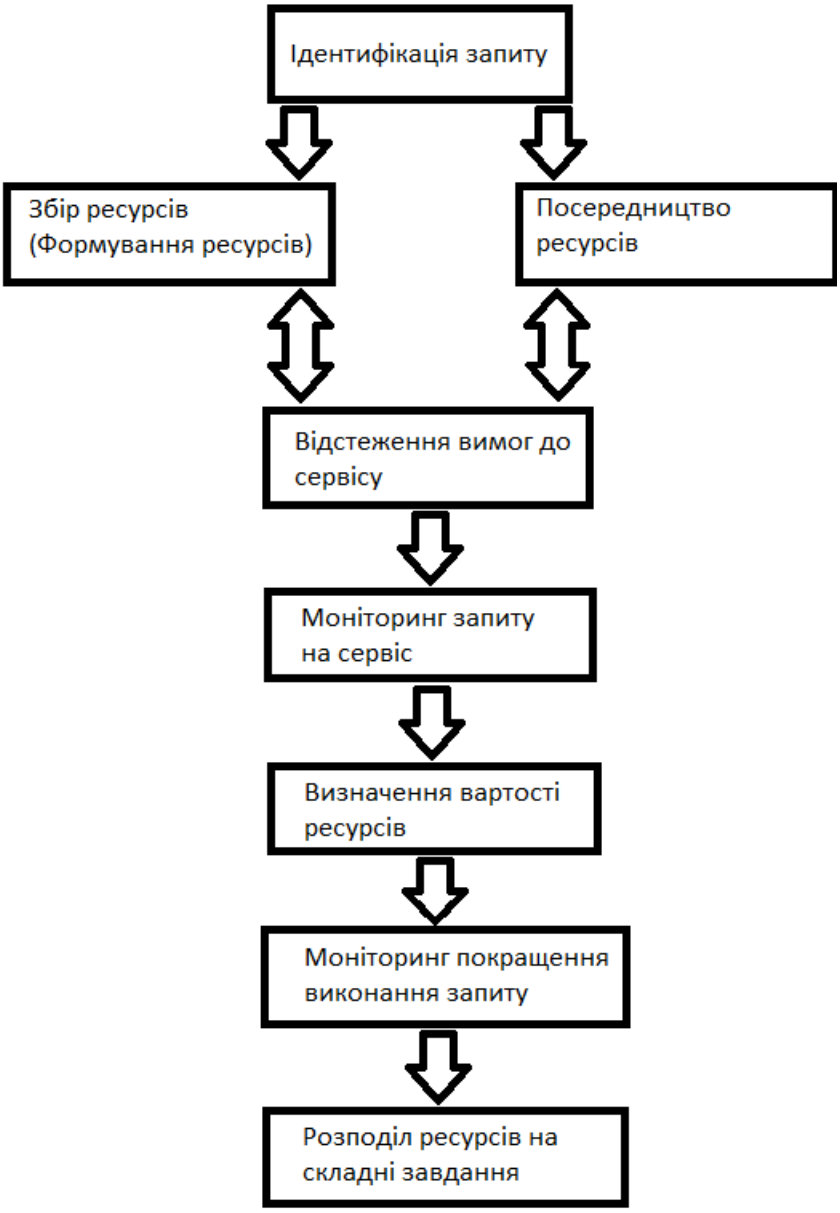

Рисунок 1.5 – Проблеми розподілу ресурсів у хмарі[46]

Першою перевагою планування ресурсів у хмарі є пошук або вибір відповідних ресурсів для планування відповідних робочих навантажень і підвищення ефективності використання ресурсів. Для кращого планування ресурсів потрібно найкраще відображення ресурсів. Друга мета планування ресурсів – знайти відповідне робоче навантаження для підтримки планування декількох робочих навантажень, щоб воно могло задовольнити різні вимоги сервісів, такі як надійність, завантаження процесора, доступність, безпека тощо для робочих навантажень у хмарі [13]. У хмарних обчисленнях управління ресурсами включає два етапи: планування ресурсів та надання ресурсів. Планування ресурсів визначається як відображення та реалізація робочого навантаження користувача хмари на основі номінованих ресурсів на основі виділених ресурсів. Натомість, виділення ресурсів – це визначення необхідних ресурсів для надання послуг на основі вимог, описаних користувачами хмари . На початку користувач передає запит на реалізацію робочого навантаження. Від імені цих вимог користувача, провайдер ресурсів визначає відповідні ресурси для робочого навантаження. Виходячи з вимог до якості обслуговування, він обмежує ймовірність та легкість надання ресурсів [19]. Після того, як ресурс ефективний, постачальник послуг хмарного сервісу відправляє ресурси для планування.

1.4 Порівняння різних методів управління ресурсами

Loud computing – віртуальні ресурси, що надаються клієнтам за моделлю «оплата за використання» під управлінням постачальників послуг. Вона включає в себе інфраструктуру як послугу (IaaS), платформу як послугу (PaaS) та програмне забезпечення як послугу (SaaS). Пріоритетом для постачальників послуг є задоволення очікувань клієнтів за конкурентними цінами. Управління ресурсами потребує оптимального використання віртуальних ресурсів, щоб реагувати у великих масштабах. Виділити відповідний ресурс для виконання різнорідних і динамічних завдань, які впливають на продуктивність і рівень обслуговування, досить складно. Якість обслуговування (QoS) можна забезпечити, враховуючи

доступність, масштабованість, використання, вартість, час, споживання енергії тощо.

Хмара має можливість надавати послуги відповідно до поведінки додатків. Такі організації, як банківська система, система охорони здоров'я, освітні платформи та електронна комерція використовують хмарні сервіси для зберігання та пошуку даних . Це усунуло необхідність купувати фізичні ресурси. Хмарні сервіси стали невід'ємною частиною нашого повсякденного життя, які задовольняють потреби в інформаційних технологіях (IT) завдяки економічній ефективності та зручності використання.

Щоб реалізувати це, існує проблема забезпечення гарантій вимог QoS та SLA шляхом управління гетерогенністю, динамічністю, складністю та невизначеністю ресурсів. Це може викликати недовіру у відносинах між споживачем і провайдером, коли цінова політика залежить від параметрів QoS.

Необхідно дослідити можливості хмарних обчислень для ефективного управління різноманітними вимогами додатків. Існуючі методи управління ресурсами не здатні в такому кастомізованому середовищі досягти важливих параметрів QoS та уникнути порушень SLA. Це опитування було проведено з метою надання практичної інформації в галузі управління хмарними ресурсами. В умовах непередбачуваності та невизначеності хмарного середовища проблему викликає неправильне налаштування. При цьому необхідно враховувати наступне:

Споживачі платять і очікують хмарний сервіс за розумну ціну і в розумні терміни. Але забезпечити очікувану продуктивність та дотримання SLA є складним завданням, яке поділяють на такі частини:Доступність ресурсів відіграє важливу роль для обробки миттєвого попиту користувачів. Але динамічний перерозподіл ресурсів призводить до перевантаження мережі та споживання енергії:

– хмарне середовище є динамічним і непередбачуваним. Керувати динамічністю та невизначеністю складно;

– хмарні обчислення складаються з різнорідних ресурсів. Вони можуть задовольнити різноманітні вимоги додатків. Але традиційні методи не здатні керувати гетерогенністю;

– хмарні дата-центри розташовані в різних географічних регіонах, що вимагає належного розподілу ізольованих ресурсів для більш ефективного використання.

Але існуючі методи управління ресурсами не здатні керувати диверсифікованими мережевими ресурсами.

Більше того, існуючі методи потребують розширення для налаштування нових платформ, таких як периферійні обчислення, контейнери та гібридні хмарні компоненти. Основним питанням у цьому контексті є підвищення доступності, використання та еластичності, що допомагає уникнути порушення SLA.

Попит на хмарні сервіси стрімко зростає. Щоб ефективно управляти поставками, захищаючи неприродні умови, потрібне краще рішення для управління ресурсами.

У роботі [4] присвячене стратегіям розподілу ресурсів, зосереджується на угоді про рівень обслуговування (SLA) для визначення використання ресурсів. Це поглиблене обговорення оптимального розподілу ресурсів для зміцнення хмарних сервісів. У роботі [23] представлено огляд методів управління ресурсами в хмарних обчисленнях з урахуванням QoS. Також обговорено важливість методів самоуправління з урахуванням оптимізації. Показано вплив параметрів QoS на SLA, а також надано пропозиції щодо подальшого розвитку стандартизації методів управління ресурсами. Мустафа та ін. представляють огляд і таксономію методів управління ресурсами.

У роботі[13] детально описано таксономію планування робочих процесів для ефективного управління ресурсами. Було обговорено проблеми розподілу ресурсів і планування для надання послуг з урахуванням витрат при управлінні додатками, що вимагають великих обсягів даних. У роботі [44] представлено таксономію схем планування ресурсів для надання економічно ефективних хмарних сервісів. У ній обговорюється метод, що враховує еволюцію, для належного використання ресурсів IaaS, PaaS і SaaS при одночасному управлінні енергією і прибутком.

Вони висвітлюють питання перерозподілу ВМ для різноманітних вимог додатків, зменшення порушення енергетичних SLA. Також обговорюються різні

метрики додатків і платформи для відповідного розміщення ВМ з метою управління використанням ресурсів і прибутком.

Багато досліджень присвячено SLA та його порушенню з точки зору параметрів QoS. У роботі [25] представлено «розподіл ресурсів з урахуванням енергоефективності» (EA-RA), процес, заснований на переговорах.Автори враховують параметри QoS та викиди парникових газів шляхом оптимізації з метою зменшення витрат та споживання енергії. У роботі [32] представлено метод, заснований на QoS, для покращення використання за допомогою технології Cloud Virtual, яка віртуалізує ресурси хмарного центру обробки даних, на яких запускаються екземпляри ресурсів або додатків. На одній фізичній машині запускається декілька служб додатків. Але виділення конкретного екземпляру віртуальної машини для виконання конкретного завдання ускладнюється через гетерогенність та динамічність середовища.

У цьому контексті у роботі [21] представлено метод високої доступності (HA) для управління хмарною інфраструктурою.У дослідницькій роботі обговорюються методи, розроблені для підвищення надійності та доступності в хмарі. В ній висвітлено проблеми та розроблено метод на основі бенчмаркінгу, який оцінює доступні для розгортання клієнту ресурси. У роботі [11] представлено метод підвищення доступності ресурсів за допомогою кластеризації віртуальних машин у центрах обробки даних.Було оптимізовано спільні ресурси для отримання кращих результатів щодо масштабованості та доступності. В роботі виконується автоматична міграція віртуальних машин для управління доступністю та балансуванням, що покращує використання.

У роботі [23] представлено огляд поточних рішень щодо доступності в хмарних розробках. Ця таксономія обговорювала форум доступності послуг, концепції та механізми базової оцінки. Були класифіковані запропоновані рішення, відмінності та подібності між різними рішеннями з точки зору доступності та їх впливу на продуктивність та SLA. У роботі [30] проведено дослідження щодо відображення ВМ на РМ і класифікувано методи, засновані на відображенні властивостей, що впливають на доступність. У цій дослідницькій роботі

обговорюються дії планування, тригери, цілі оптимізації, метрики, додатки та платформи, що використовуються для оцінки в різних методах. У роботі [29] представлено алгоритм «Енергоефективного розміщення віртуальних машин» (EE-VMsP) для оптимального мапування. Автори виконують конфігурацію та розміщення віртуальних машин для зменшення споживання енергії та тривалості роботи шляхом належної обробки гетерогенних та динамічних завдань.

Хмарне середовище надає економічно ефективні послуги за моделлю оплати за використання. Таким чином, провайдери повинні забезпечити надання послуг відповідно до попиту та моніторинг виділених ресурсів. Моніторинг є важливим аспектом управління хмарними ресурсами. Хмара стикається з проблемами управління послугами на вимогу, масштабованістю, еластичністю та невизначеністю.

У роботі [31] проведено огляд сучасних інструментів моніторингу та дослідили конструкції та проблеми моніторингу хмарних технологій. Булозроблено висновок, що хмара потребує розподілених інструментів підтримки моніторингу. Ці інструменти повинні охоплювати широкий спектр моніторингу ресурсів для управління масштабованістю, еластичністю, продуктивністю на вимогу споживачів. У роботі [1] підкреслено важливість моніторингу та факторів, що впливають на продуктивність: масштабованість, еластичність та міграція. Було показано, що моніторинг повинен здійснюватися на різних рівнях для використання хмарних ресурсів. У роботі [24] проведено дослідження з моніторингу, яке розглядає важливість доступності ресурсів на рівні IaaS. Дослідження показує, що доступність може покращити якість послуг та їх використання. У роботі підкреслюється, що доступність і моніторинг допомагають підвищити якість і продуктивність послуг за рахунок поліпшення їх використання.

У роботі [12] запропоновано «самоадаптивний підхід до моніторингу» (SA-MA) для оцінки поточного стану системи, ступеня аномалії та прогнозування виявлення несправностей за допомогою аналізу головних компонент. Автори ефективно виявляють ненормальний стан і знижують накладні витрати на моніторинг.

Моніторинг виділених ресурсів може підвищити доступність і зручність використання. Традиційні методи RM не здатні забезпечити таке середовище для ідентифікації робочого навантаження під час виконання, яке може відстежувати стан ресурсів. Опитування показує, що було проведено обмежену роботу з моніторингу для управління еластичним масштабуванням та очікуваною якістю послуг.

Технологія Cloud Virtual віртуалізує ресурси хмарного центру обробки даних, які запускають екземпляри ресурсів або програм. Одна фізична машина запускає кілька служб додатків. Але виділення конкретного екземпляра віртуальної машини для конкретного завдання є складним через неоднорідність і динамічне середовище.

У цьому контексті у роботі[21] представлено метод високої доступності (HA) для управління хмарною інфраструктурою. У дослідницькій роботі обговорювалися методи, розроблені для підвищення надійності та доступності в хмарі. У ньому висвітлюються проблеми та розроблений метод на основі порівняльного аналізу, який оцінює доступні ресурси для розгортання клієнта. У роботі представлено метод високої доступності ресурсів за допомогою кластеризації віртуальних машин у центрах обробки даних. Автори оптимізують спільні ресурси для отримання кращих результатів щодо масштабованості та доступності. Робота виконувала автоматичну міграцію віртуальної машини для керування доступністю та балансуванням, що покращує використання. У роботі [23] представлено опитування щодо поточних рішень доступності в хмарі. Ця таксономія обговорювала форум доступності служби, концепції та механізми базової оцінки. Автори класифікували запропоновані рішення, відмінності та подібності між різними рішеннями з точки зору доступності та їх впливу на продуктивність і SLA. У роботі [31]автори роблять опис щодо відображення віртуальних машин у PM та класифікації методів на основі доступності впливу властивостей відображення. У цій дослідницькій роботі обговорювалися планування дій, тригери, цілі оптимізації, показники, додатки та платформи, які використовуються для оцінювання в різних техніках. У роботі [29] представлено

алгоритм «Енергоефективне розміщення віртуальних машин» (EE-VMsP) для оптимального відображення. Виконано конфігурацію та розміщення віртуальних машин, щоб зменшити споживання енергії та забезпечити належну обробку різнорідних та динамічних завдань.

Висновки до розділу 1

У розділі проведено огляд сучасних підходів до управління ресурсами в хмарних середовищах, зокрема моделей розгортання хмарних обчислень (публічних, приватних і гібридних), а також сервісів SaaS, PaaS та IaaS. Виявлено ключові аспекти архітектури хмарних систем та виклики, пов’язані з управлінням ресурсами, такі як енергоефективність, дотримання SLA та балансування навантаження.

Проведено аналіз проблем оптимізації ресурсів, включаючи адаптивний розподіл ресурсів та динамічне планування. Визначено основні недоліки традиційних методів управління, що не враховують особливості гібридних та мобільних хмарних середовищ. Встановлено, що багатокритеріальна оптимізація є ефективним підходом для вирішення суперечливих задач, таких як енергоефективність і максимізація прибутку.

Розглянуто сучасні підходи до управління ресурсами, включаючи моделі консолідації робочого навантаження та енергоефективні алгоритми, що дозволяють оптимізувати використання серверів і зменшити витрати на електроенергію.

Результати цього розділу формують основу для впровадження інтелектуальних систем управління ресурсами в хмарних середовищах, що враховують динамічність середовища, енергоспоживання та потреби користувачів.

2 АНАЛІЗ МЕТОДІВ МАШИННОГО НАВЧАННЯ В УПРАВЛІННІ РЕСУРСАМИ

2.1 Підходи до прогнозування навантаження на системи

Розподіл віртуальних машин (ВМ) є ключовим завданням у хмарних середовищах, яке впливає на ефективність та економічність роботи хмари. Постачальники послуг прагнуть підвищити ефективність використання ресурсів, в той час як клієнти шукають економічно ефективний розподіл ресурсів. Однак різні робочі навантаження та різні вимоги до якості обслуговування (QoS) призводять до коливань попиту на ресурси. Затримка в доступності ресурсів ще більше ускладнює ситуацію. Ефективне надання достатньої кількості ресурсів із гарантованою якістю обслуговування має важливе значення для хмарних провайдерів та користувачів [1]. Одним з підходів до вирішення цієї проблеми є точне прогнозування майбутньої поведінки робочого навантаження. Аналіз даних про використання в минулому дозволяє проводити прогнозний аналіз і приймати кращі рішення в управлінні ресурсами. Прогнозування робочого навантаження є особливо важливим для ефективності виконання додатків у хмарах [3]. Підвищене навантаження на процесор негативно впливає на продуктивність і надійність. Тому для ефективного розподілу ресурсів необхідні автоматизовані та адаптивні підходи для прогнозування попиту на навантаження. Однак прогнозуванню навантаження для декількох віртуальних машин у хмарних середовищах приділяється менше уваги, незважаючи на взаємопов'язану природу сучасного хмарного програмного забезпечення [6].

У порівнянні з традиційними моделями машинного навчання [16], гібридна модель поєднує LSTM та AdaBoost для вирішення проблем, пов'язаних з прогнозуванням навантаження на ВМ. Традиційні моделі машинного навчання часто намагаються вловити складні часові залежності та закономірності в даних про навантаження. На відміну від них, запропонована гібридна модель ефективно використовує LSTM для фіксації та моделювання цих залежностей, що призводить до більш точного прогнозування навантаження. Моделі глибокого навчання, такі

як LSTM, добре вловлюють часові залежності, але вони стикаються з труднощами при обробці зашумлених або викидних точок даних. Гібридна модель долає це обмеження шляхом інтеграції AdaBoost, який чудово справляється з зашумленими даними, призначаючи більшу вагу складним для прогнозування вибіркам, що призводить до більш точних прогнозів навантаження навіть за наявності нерівномірностей у даних. Крім того, гібридна модель забезпечує більшу гнучкість та інтерпретованість порівняно з моделями глибокого навчання.

Модель на основі генетичного алгоритму для оцінки використання процесора та пам'яті для віртуальних машин була запропонована у роботі [10], і порівняно з сірою моделлю, результати методу показують більш точні прогнози, особливо для стабільності та нестабільності. У роботі [25] представлена модель прогнозування, яка враховує короткострокове та довгострокове використання ресурсів процесора/пам'яті у віртуальних середовищах. Автори використовують байєсівські мережі для виявлення взаємозв'язків і залежностей між змінними, що дозволяє більш точно прогнозувати бажані результати. У роботі [26] представлена модель прогнозування, заснована на алгоритмі нейронної мережі зворотного поширення (BPNN), названу RVLBPNN (Random Variable Learning Rate Backpropagation Neural Network – нейронна мережа зі швидкістю навчання випадкових змінних). Моделі, які враховують навантаження на процесор та пам'ять і використовують внутрішні взаємозв'язки між навантаженнями, перевершують за точністю прогнозування приховані марковські моделі та наївні байєсівські класифікатори. У роботі [27] запропоновано метод прогнозування погоди з використанням мережі глибокого переконання (DBN), який підвищує точність прогнозу за допомогою таких методів, як ортогональний експериментальний дизайн і метод «Знайди різницю» (Spot the Difference). Порівняння з моделлю ARIMA показує, що точність прогнозування попиту на процесори та оперативну пам'ять значно зросла. Однак методи машинного навчання часто є складними і вимагають багато даних для вилучення та навчання моделей, що може бути тривалим і обмежує їх використання для забезпечення якості обслуговування (QoS).

Прогнозування для хмарних операцій, що підкреслює важливість розуміння поведінки користувачів для фактично точного відображення реальності. Щоб підвищити точність прогнозу, скоротивши при цьому час розрахунку, запропонований метод інтегрує моделі ARIMA з алгоритмами EEMD та RT. Для порівняння EEMD-ARIMA з моделями ARIMA враховано такі фактори, як похибка оцінки, достовірність та оцінка часових витрат. Результат показує покращення прогнозування продуктивності, що дозволяє покращити розподіл ресурсів та оптимізацію QoS у хмарних системах. Модель у роботі [12] поєднує різні методи прогнозування, такі як авторегресійна модель, модель на основі ann та svm моделі. Вага кожного методу динамічно змінюється відповідно до відносної похибки для досягнення найкращої середньої продуктивності. Метод емпіричної декомпозиції мод (Empirical Mode Decomposition, EEMD) використовується для контролю нестаціонарних хостів, таким чином підвищуючи точність прогнозу та зменшуючи час обчислень.

Для вирішення проблеми оцінювання в контексті розподілу хмарних ресурсів у роботі [14] представлена нечітка логіко-орієнтована модель для прогнозування навантаження додатків в Інтернеті. У роботі [17] досліджується використання доповненого навчання за допомогою штучних нейронних мереж (ШНМ) для оптимізації конфігурації процесора та пам'яті віртуальних машин (ВМ) з метою підвищення продуктивності додатків. Однак ці рішення, як правило, зосереджені на управлінні ресурсами процесора і не включають інші аспекти розподілу та оптимізації ресурсів. У роботі [18] було запропоновано метод зіставлення шаблонів для виявлення подібних шаблонів використання в минулих траєкторіях використання користувачами хмарних сервісів. Інтерпольовано і прогнозовано майбутні ціни на основі подібних моделей. У роботі [21] використана модель та регресія опорних векторів, щоб була представлена модель для відображення взаємозв'язку між розподілом ресурсів та ефективністю у хмаро-віртуалізованому середовищі. Модель призначена для прогнозування ресурсів, необхідних для досягнення цільових показників продуктивності. Однак цей підхід передбачає статичну функціональність і стабільну поведінку використання, що обмежує його

застосовність у хмарному середовищі. Крім того, автономна робоча модель не може своєчасно реагувати на зміни навколишнього середовища та навантаження.

Запропонована гібридна модель успішно фіксує часові кореляції та складні патерни навантаження в даних про навантаження на віртуальні машини, поєднуючи переваги мереж з довгою та короткою пам'яттю (LSTM) з AdaBoost. Гібридна модель забезпечує підвищену точність прогнозування навантаження на віртуальні машини завдяки двоетапній процедурі навчання LSTM для вилучення корисних ознак та навчання AdaBoost для агрегування прогнозів від численних слабких регресорів. Він ефективно керує нелінійними взаємодіями та використовує переваги як LSTM, так і AdaBoost для отримання більш точних оцінок навантаження. Для отримання точних та ефективних прогнозів гібридна модель LSTM та AdaBoost для прогнозування навантаження на віртуальні машини в хмарі використовує двофазну методологію.

На першому етапі запропонована модель спочатку використовує можливості LSTM для виявлення часових залежностей і довгострокових тенденцій у даних про навантаження на ВМ. Збираються та попередньо обробляються історичні дані, які в подальшому подаються на вхід LSTM-моделі, оскільки LSTMA-модель – це особливий вид рекурентної нейронної мережі, яка добре запам'ятовує послідовні шаблони та зберігає їх у пам'яті протягом тривалого часу. Щоб спрогнозувати навантаження на віртуальну машину на наступний часовий крок, вона навчається на попередньо оброблених даних. Використовуючи модель LSTM, на цьому етапі основна увага приділяється виявленню основних залежностей і тенденцій у даних про навантаження на віртуальну машину.

На другому етапі запропонована модель інтегрує алгоритм AdaBoost для подальшого покращення можливостей прогнозування LSTM-моделі. AdaBoost - це метод бустінгу, який об'єднує декілька поганих класифікаторів в один потужний класифікатор. Для навчання кожного слабкого класифікатора використовується окрема підмножина даних. Слабкі класифікатори можна розглядати як альтернативні моделі в контексті прогнозування навантаження, які створюють прогнози на основі різних характеристик даних. На основі їх продуктивності

алгоритм AdaBoost розподіляє ваги між цими слабкими класифікаторами, а остаточний прогноз виводиться шляхом об'єднання прогнозів всіх слабких класифікаторів з використанням цих ваг.

Для того, щоб підвищити точність прогнозування навантаження на віртуальну машину, ця фаза має на меті скористатися перевагами різноманітності слабких класифікаторів та успішно об'єднати їхні прогнози. На рисунку 2.1 зображено робочу архітектуру гібридної моделі.

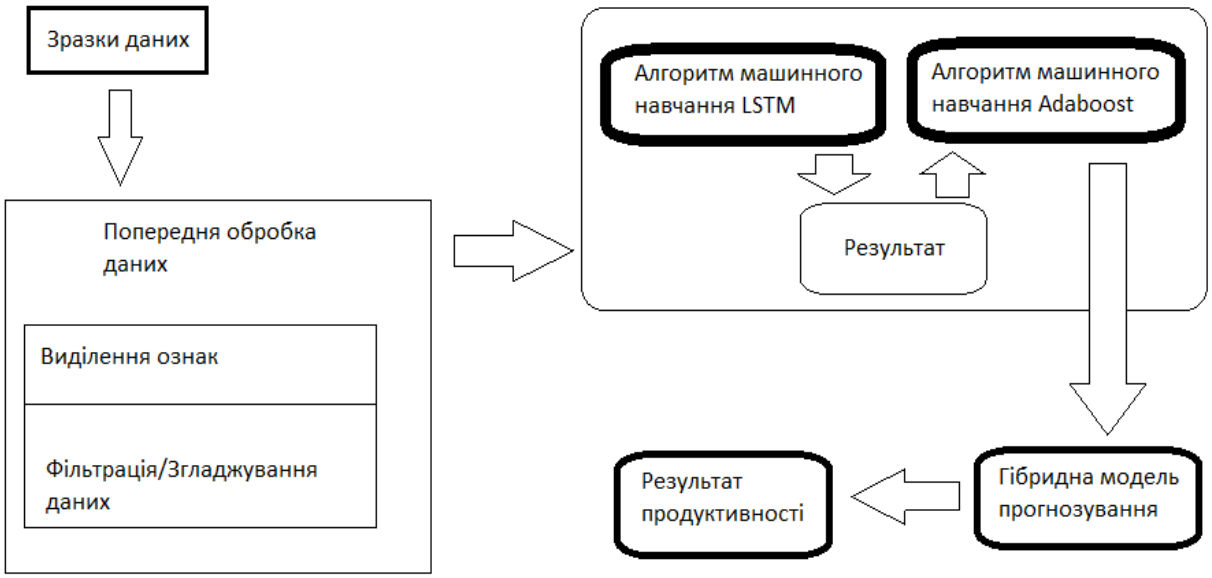

Рисунок 2.1 – Робоча архітектура гібридної моделі[47]

Результати показали, що гібридний підхід перевершує ці алгоритми в точності прогнозування навантаження на віртуальну машину.

2.2 Алгоритми оптимізації ресурсів на основі машинного навчання

За визначенням Національного інституту стандартів і технологій (NIST), «хмарні обчислення» – це структура, яка забезпечує широкий, комфортний, на вимогу міжмережевий магістральний доступ до узгодженого пулу ресурсів, які можуть бути швидко розподілені та ініційовані з мінімальною координацією з боку постачальника послуг» [30]. Розподіл віртуалізованих ресурсів інформаційно-

комунікаційних технологій в парадигмі хмарних обчислень є складною проблемою для вирішення, оскільки існують різні робочі навантаження додатків (MapReduce, розподіл контенту та мережеві веб-додатки) з суперечливими вимогами до розподілу ресурсів інформаційно-комунікаційних технологій (час відгуку, час виконання, використання ресурсів і т.д.). Через обмеженість наявних ресурсів та зростаючі вимоги клієнтів, завдання розподілу ресурсів стає дедалі складнішим. В результаті було розроблено кілька нових моделей і стратегій для ефективного розподілу ресурсів. Деякі технології використовують динамічні методи і моделі розподілу ресурсів, які зосереджують свою увагу на різноманітних обмеженнях або цілях для оптимізації розподілу ресурсів.

Прогнозування пропускної здатності мережі на основі досліджень трафіку в режимі реального часу є основною перешкодою для підвищення ефективності хмарних обчислень [1].

Це пов'язано з тим, що основна увага при управлінні мережею зосереджена на мережевих ресурсах, ігноруючи при цьому доступність хмарних ресурсів. Ці недоліки можна зменшити, використовуючи технології глибокого навчання (DL) і машинного навчання (ML) для автоматизації самоконфігурації мережі та управління несправностями. У більшості досліджень, присвячених глибинному навчанню і машинному навчанню для оптичних мереж, використовуються методи з автономним контролем. Згідно з основним припущенням, моделі проходять навчання на минулих даних, перш ніж вони будуть застосовані до реальних подій. Це обмеження часто не є актуальним для глобальних мереж (WAN) через те, як швидко можуть змінюватися моделі трафіку [23, 33]. Інноваційні аналітичні стратегії необхідні для успішного видобутку цього величезного обсягу мережевих даних для отримання релевантної інформації. Є припущення, що концептуальні основи машинного навчанняі глибинного навчання можуть дати дієві відповіді для обробки мережевих даних.

Автоматизований динамічний розподіл ресурсів використовується для коригування способу використання хмарних ресурсів, щоб краще відповідати оптимізаційній меті постачальника хмарних послуг, яка полягає в максимальному

використанні обчислювальних ресурсів. Це досягається шляхом зміни способу використання хмарних ресурсів.

Цей підхід аналізує евристичні дані про використання ресурсів, коли клієнти використовують певні програми, і надає оптимальний ресурс для цієї програми з урахуванням конфігурації користувача. Такий розподіл ресурсів забезпечує додатковий ресурс, коли необхідно підтримувати оптимальне використання ресурсів, коли невикористані ресурси хмари звільняються для перерозподілу.

Хмарні обчислення стали повсюдними невдовзі після запуску Amazon. Elastic Compute Cloud Product у 2006 році. Це відкрило двері для інших великих провайдерів послуг для впровадження хмарних обчислень і побудови хмарних системних мереж з підвищеною відмовостійкістю. Хмарні обчислення є інтригуючим проривом завдяки своїй моделі ціноутворення «платити по мірі використання» та універсальності. Хмарніобчислювальні рішення функціонують шляхом розгортання великого центрального сервера в різних географічних точках, а потім розподіляють ресурси з серверів на основі попиту. З появою більш досконалих інструментів зростає попит на конкретні функції хмарних обчислень. Промисловість та організації завжди шукають високопродуктивну мережу з легкодоступними пристроями зберігання даних, які б дозволили вести бізнес на недорогих ПК. Через всепроникну природу бізнесу в наш час спостерігається стрімке зростання використання хмарних обчислень. Наприклад, у 2019 році завдяки хмарній віртуалізації та індивідуальній архітектурі Linux отримав широке розповсюдження і став доступним для багатьох платформ. Центри обробки даних забезпечують основу для всіх цих процесів, розміщуючи програмне забезпечення з інтенсивними потребами в обробці даних.

Незважаючи на те, що хмарні обчислення набувають все більшої популярності в індустрії інформаційних технологій завдяки багатьом перевагам, які вони пропонують, на шляху хмарних інновацій все ще існують серйозні перешкоди. Деякі з цих перешкод включають управління, відповідність даних, проблеми безпеки, невизначеність щодо енергоефективності та труднощі зі

стратегією впровадження. Ці проблеми викликають занепокоєння, і ми намагаємося знайти ефективні способи їх вирішення.

Термін «хмарні обчислення» (ХО) означає парадигму, яка останнім часом стала найбільш відомою і часто використовуваною в галузі інформаційних технологій і телекомунікацій (ІКТ). Користувачі хмарних сервісів не завжди усвідомлюють цінність хмарних інновацій, навіть якщо вони прямо чи опосередковано підтримують свої повсякденні пошукові сервіси через діяльність в Інтернеті. Через свою важливість у світі комп'ютерів та інженерії хмарні обчислення стали популярним терміном у комунікації. Послуги хмарних обчислень дають змогу країнам, що розвиваються, та малозабезпеченим країнам отримувати необхідні послуги без обмежень, сприяючи швидкому економічному прогресу [35]. До періоду хмарних інновацій створення компанією традиційного центру обробки даних було складним процесом через потребу в грошових коштах як на утримання, так і на початкові інвестиції в інфраструктуру.

На противагу цьому, зараз ми використовуємо хмарні сервіси, в яких комп'ютерний товар можна просто орендувати залежно від потреби, а програму можна розгорнути без зайвих зусиль. Багато компаній (великих і малих) намагаються збалансувати свої операційні витрати, одночасно отримуючи доступ до інструментів підвищення ефективності (таких як платформи, інфраструктура та власне програмне забезпечення), тому оптимізація інноваційних послуг КК стає неминучою через численні переваги, які відповідають потребам бізнесу. Користувачі мають простий, послідовний і масштабований доступ до спільного пулу програмованих мережевих активів, коли підходи до продуктивності хмарних обчислень ґрунтуються на комерційній моделі, що базується на корисності. Ця концепція є основою хмарних обчислень. Клієнти можуть використовувати канали на свій вибір, виходячи з конкретних потреб, завдяки надійному та стійкому до збоїв процесу, який стає можливим завдяки віртуальним машинам, розміщеним у хмарі.

Використовуючи генетичні алгоритми (ГА) та легкі симулятори, у роботі [17] розроблено те, що називається розподілом ресурсів з урахуванням топології

(Topology Aware Resource Allocation), модель, яка може прогнозувати розподіл ресурсів у середовищі IaaS (TARA).Метою цієї моделі була оптимізація Map Reduce, і вона зафіксувала скорочення часу виконання завдання на 50% у порівнянні з незалежним від додатків розподілом.

У роботі [48] розроблено систему розподілу ресурсів (RAS) для моделі хмарного сервісу, яка базувалася на концепції інфраструктури як послуги (IaaS). Це було зроблено для покращення ціни та прибутковості бізнесу їхніх клієнтів. Ця RAS використовує запропоновану політику для покращення використання ресурсів за рахунок залучення ресурсів від інших постачальників послуг, які не використовуються. У роботі [9] застосували інший підхід до моделі хмарного сервісу IaaS, щоб оптимізувати обчислення для кращого екологічного використання обчислень. Це досягається шляхом впровадження алгоритму асиметрії, який вимірює невідповідність ресурсів у багатовимірному використанні ресурсів, інтегруючи в їх систему набір евристик для запобігання перевантаженню системи.

Концепція балансування навантаження полягає в справедливому розподілі робочого навантаження між усіма доступними інформаційно-технологічними ресурсами. Навіть якщо сервіс не працює, головна мета - забезпечити його роботу, надаючи процедурам прийнятне використання ресурсів. Балансування навантаження також фокусується на зменшенні затримки завдань та оптимізації використання ресурсів, що призводить до економічно ефективної, покращеної продуктивності системи. Воно також пропонує універсальність і гнучкість для використання з різними розмірами, які можуть змінитися в майбутньому, що вимагатиме використання більшої кількості IT-ресурсів. Інші цілі включають зменшення споживання енергії та викидів вуглецю, а також уникнення перевантажень за рахунок постачання ресурсів та дотримання стандартів якості обслуговування [16, 27]. Як наслідок, це вимагає відповідного механізму планування навантаження, який враховує численні заходи.

Балансування навантаження – це механізм розподілу навантаження багатьох користувачів на одне або кілька з'єднань, серверів, терміналів або інших IT-

ресурсів [10]. Ця хмарна технологія відрізняється від традиційної архітектури справжнього балансування навантаження. У хмарній індустрії багато науковців по всьому світу досліджують і розробляють різні типи оптимальних методів розподілу ресурсів.

Підхід використовує дисперсію часу виконання для належного балансування IT-ресурсів та підвищення продуктивності. На додаток до балансування навантаження, ми маємо різні додаткові проблеми, такі як час виконання, продуктивність ВМ, економія енергії, міграція ВМ, викиди вуглекислого газу, управління якістю обслуговування та ресурсами тощо [13], [28]. У роботі [4] досліджували евристичні підходи і використовували різні типи навантаження для покращення робочого процесу в хмарному середовищі. Ці навантаження включали мережеве, процесорне, пам'ять та інші. У роботі [3] проведено детальне дослідження різних стратегій планування завдань і визначили заходи, які підходять для хмарного середовища. Спочатку література була зосереджена на методологіях, параметрах і застосуваннях. Деякі дослідники застосовували заходи безпеки [45] до різних метрик, що використовуються в контексті балансування навантаження.

Незважаючи на те, що хмарні обчислення привернули до себе багато уваги, вони все ще мають кілька недоліків, одним з яких є балансування навантаження. Деякі з проблем, з якими стикається балансування навантаження в хмарних обчисленнях.

За допомогою віртуалізації ціла макросхема може сприйматися як файл або серія файлів, а віртуальну машину також можна перемістити між фізичними комп'ютерами, щоб зменшити навантаження на перевантажену реальну макросхему. Рівномірний розподіл робочого навантаження по всьому центру обробки даних або кластеру центрів обробки даних є головним пріоритетом. Чи існує спосіб динамічного розподілу навантаження в системах хмарних обчислень для запобігання виникненню вузьких місць? Це питання актуальне для процесу переміщення віртуальних макмашин;

Еластичність хмарних обчислень, яка дозволяє миттєво призначати та звільняти ресурси, є однією з найпривабливіших їхніх особливостей. Які найкращі

способи використання або вивільнення хмарних ресурсів при збереженні звичайної продуктивності системи та оптимальному використанні ресурсів;

За останнє десятиліття обсяги даних, що зберігаються в мережі, зростали експоненціальними темпами, а управління зберіганням даних стало критичною проблемою для хмарних обчислень, навіть для організацій, що укладають субпідрядні договори на зберігання даних, або для приватних осіб;

Мікро-центри обробки даних можуть бути дешевшими, енергоефективнішими та кориснішими, ніж великі центри обробки даних. Малий бізнес може запропонувати надання багатьох хмарних послуг, що зробить можливими обчислення в географічному розрізі. Забезпечити достатній час реакції за допомогою ефективного розподілу ресурсів, балансування навантаження стане проблемою глобального масштабу;

Економія від масштабу є одною з важливих переваг використання хмарних технологій. Енергозбереження має важливе значення в глобальній економіці, оскільки обмежена кількість світових ресурсів підтримується обмеженою кількістю компаній, які працюють швидше, ніж кожна з них.

Завдяки ефективному плануванню роботи та розподілу ресурсів, кілька сучасних методів планування можуть підтримувати баланс навантаження та забезпечувати кращі результати. Ефективне використання ресурсів є життєво важливим для максимізації доходів за допомогою оптимальних алгоритмів балансування навантаження.

Щоб запобігти зациклюванню на локальному оптимумі, у роботі [32] представили новий метод балансування навантаження, який включає алгоритм оптимізації на основі навчання (TLBO) та генетично зважену оптимізацію (GWO), щоб збалансувати навантаження на всі віртуальні машини, максимізуючи при цьому пропускну здатність (BM). На 11 тестових функціях гібридні результати оцінювалися за допомогою оптимізації роєм (PSO), оптимізації на основі біогеографії (BBO) та генетично зваженої оптимізації (GWO). Для перевірки запропонованого підходу до балансування навантаження було проведено симуляцію гібридного алгоритму. Низька фінансова вигода провайдерів послуг

пояснюється неефективним використанням ресурсів та енергії. Як наслідок, центри обробки даних могли б використовувати ефективну стратегію управління ресурсами. З огляду на це, у роботі [12] розроблено нову архітектуру балансування навантаження, яка дозволяє максимально ефективно використовувати ресурси дата-центру, зменшуючи при цьому операційні витрати. Для реалізації найкращого розподілу віртуальних машин між локальними комп'ютерами, фреймворк використовував модифікований генетичний алгоритм.

Результати тестування показали, що запропонований фреймворк перевершує поточні та три інші поширені евристичні технології розміщення ВМ на 45,21%, 84,49%, 119,93% та 113,96% з точки зору споживання ресурсів. Самокероване прогнозування навантаження (SDWF) – це метод, запропонований у роботі [11], яка використовує різницю між фактичним та прогнозованим навантаженням для кращого передбачення майбутніх навантажень. Нейронні мережі в моделі навчаються за допомогою покращеної евристики, заснованої на виникненні чорних дір. Запропонований метод було протестовано з використанням шести різних потоків реальних даних. Точність порівнювали з найсучаснішою моделлю, побудованою за допомогою таких інструментів, як глибоке навчання, еволюційні алгоритми та зворотне розповсюдження. Цей підхід зменшив середньоквадратичну помилку прогнозу на 99,9% порівняно зі звичайним методом. Для оцінки системи прогнозування були використані критерії Фрідмана та Вілкоксона.

Планування завдань є ключовим елементом для ефективного балансування навантаження в хмарних середовищах. Воно дозволяє оптимізувати використання ресурсів і забезпечити їх рівномірний розподіл серед обчислювальних одиниць, що допомагає уникнути перевантажень на окремих серверах і знижує ризик затримок у виконанні завдань. Однією з основних цілей планування завдань є дотримання Угоди про рівень обслуговування (SLA), яка є контрактом, що визначає вимоги до надання послуг споживачам, включаючи такі аспекти, як доступність, швидкість обробки, та максимальні допустимі затримки.

Висновки до розділу 2

У розділі проведено аналіз методів машинного навчання, що використовуються для управління ресурсами в хмарних середовищах. Особливу увагу приділено підходам до прогнозування навантаження на системи, які дозволяють заздалегідь визначати потреби у ресурсах, мінімізуючи ризик перевантаження.

Розглянуто алгоритми оптимізації ресурсів, серед яких моделі на основі Support Vector Machines (SVM), LSTM та AdaBoost, які демонструють високу точність прогнозування завдяки здатності обробляти великі обсяги даних і враховувати часові залежності. Запропоновані методи дозволяють ефективно розподіляти ресурси, забезпечуючи адаптивність системи до змінних умов.

Також оцінено переваги використання гібридних підходів, що поєднують різні алгоритми для підвищення точності прогнозів та покращення якості обслуговування (QoS). Виявлено, що застосування методів машинного навчання сприяє оптимізації витрат і підвищенню ефективності роботи хмарних систем.

Результати дослідження формують базу для впровадження інтелектуальних систем управління ресурсами, що забезпечують динамічний розподіл ресурсів, мінімізацію затримок і адаптацію до змін у навантаженнях.

3 АРХІТЕКТУРА СИСТЕМИ УПРАВЛІННЯ РЕСУРСАМИ У ХМАРНОМУ СЕРЕДОВИЩІ

3.1 Постановка задачі

Для бездротового Інтернету речей, оскільки радіоресурси за своєю природою є дефіцитними, управління ресурсами стає дуже складним завданням у практичних мережах, особливо коли кількість користувачів, що конкурують між собою, або смарт-пристроїв дуже велика. Загальна продуктивність будь-якої бездротової мережі залежить від того, наскільки ефективно і динамічно управляються гіпервимірні ресурси, такі як часові інтервали, частотні діапазони і ортогональні коди, а також від того, наскільки добре в проекті мережі враховані змінні навантаження трафіку і коливання бездротового каналу, щоб забезпечити зв'язок між користувачами з різними вимогами до якості обслуговування (QoS). Розробка нових додатків IoT вимагає поліпшення спектральної ефективності, високої швидкості передачі даних і пропускної здатності, а також врахування контексту і персоналізації послуг IoT, і, отже, проблема управління ресурсами набуває вирішального значення [4, 5].

Коли велика кількість пристроїв одночасно отримує доступ до бездротового каналу, це призводить до перевантаження каналу (наприклад, фізичний канал випадкового доступу (Physical Random Access Channel – PRACH) в мережах LTE). Крім того, масове розгортання пристроїв Інтернету речей створює проблеми, пов'язані з перевантаженням мережі, мережевою архітектурою та архітектурою зберігання даних для розумних пристроїв, а також ефективними протоколами передачі даних [2, 3].

Рішення для боротьби з перевантаженням каналу доступу для пристроїв Інтернету речей здебільшого ґрунтуються на використанні різних ймовірностей доступу та пріоритетів для різних класів. Управління ресурсами стільникових мереж IoT повинно враховувати методи балансування навантаження і управління доступом для підтримки великої пропускної здатності і зв'язку при ефективному використанні мережевих ресурсів.

У щільній (і великомасштабній) мережі Інтернету речей проблеми внутрішніх і міжстільникових перешкод стають критично важливими, а отже, потрібні складні методи розподілу потужності і управління перешкодами. Багато додатків IoT можуть мати гетерогенну природу з потенційно різною топологією мережі, трафіком, а також характеристиками каналів. Тому дуже складно динамічно вибирати потужність передачі у відповідь на зміну фізичних умов каналу і мережі.

Розподіл віртуалізованих ресурсів інформаційно-комунікаційних технологій в парадигмі хмарних обчислень є складною проблемою для вирішення, оскільки існують різні робочі навантаження додатків (MapReduce, розподіл контенту та мережеві веб-додатки) з суперечливими вимогами до розподілу ресурсів інформаційно-комунікаційних технологій (час відгуку, час виконання, використання ресурсів і т.д.). Через обмеженість наявних ресурсів та зростаючі вимоги клієнтів, завдання розподілу ресурсів стає дедалі складнішим.

Хмарні обчислення – це нова модель віддалених обчислень, яка відкриває доступ до багатьох джерел. У сучасну епоху завдяки перевагам високої доступності, меншій вартості послуг, більшій доступності та масштабованості ресурсів хмарні обчислення стають найбільш підходящою платформою. Вони надають обчислювальні послуги на основі Інтернету, керуючи та використовуючи велику кількість ресурсів. Термін «управління ресурсами» вказує на можливості обчислювального середовища і фокусується на тому, як вони надають послуги іншим користувачам, додаткам і організаціям. Ці ресурси надаються на вимогу [26] через Інтернет [11, 12, 27] та прикладними програмами [11, 12, 27]. Хмарні обчислення базуються на розподілених і паралельних обчисленнях [14]. Для глобального розміщення ресурсів та їх економного використання використовуються розподілені обчислення. Коли попит користувачів змінюється, ресурси також змінюються відповідно. Через динамічну природу обчислювального середовища стає дуже складно працювати з великою кількістю ресурсів. Хмарні обчислення мають багато проблем з розподілом ресурсів, включаючи

інфраструктуру та відповідні сервіси. Згідно з [1], планування ресурсів є основною проблемою хмарних обчислень.

Розподіл віртуальних машин (ВМ) є ключовим завданням у хмарних середовищах, яке впливає на ефективність та економічність роботи хмари. Постачальники послуг прагнуть підвищити ефективність використання ресурсів, в той час як клієнти шукають економічно ефективний розподіл ресурсів. Однак різні робочі навантаження та різні вимоги до якості обслуговування (QoS) призводять до коливань попиту на ресурси. Затримка в доступності ресурсів ще більше ускладнює ситуацію. Ефективне надання достатньої кількості ресурсів із гарантованою якістю обслуговування має важливе значення для хмарних провайдерів та користувачів [1]. Одним з підходів до вирішення цієї проблеми є точне прогнозування майбутньої поведінки робочого навантаження. Аналіз даних про використання в минулому дозволяє проводити прогнозний аналіз і приймати кращі рішення в управлінні ресурсами. Прогнозування робочого навантаження є особливо важливим для ефективності виконання додатків у хмарах [3]. Підвищене навантаження на процесор негативно впливає на продуктивність і надійність. Тому для ефективного розподілу ресурсів необхідні автоматизовані та адаптивні підходи для прогнозування попиту на навантаження. Однак прогнозуванню навантаження для декількох віртуальних машин у хмарних середовищах приділяється менше уваги, незважаючи на взаємопов'язану природу сучасного хмарного програмного забезпечення.

Розподіл віртуальних ресурсів і прогнозування навантаження є ключовими для забезпечення ефективності хмарних систем. Для досягнення високої продуктивності необхідно використовувати адаптивні методи, які дозволяють динамічно розподіляти ресурси, знижуючи ймовірність перевантажень і покращуючи якість обслуговування.

3.2 Архітектура системи управління

Хмарні обчислення є багатообіцяючою новою парадигмою, яка надає клієнтам економічно ефективні послуги аутсорсингу на вимогу та на основі оплати за споживання[1]. Ці послуги пропонуються через хмару. Цифровий світ, що постійно розширюється, визначається поширенням онлайн-сервісів , додатків, що керуються даними, і нестримним зростанням контенту, створеного користувачами, призвело до безпрецедентного попиту на обчислювальні ресурси в середовищах хмарних обчислень.[2]. Постачальники хмарних послуг (CSP) стали наріжним каменем цифрової ери, надаючи рішення «Інфраструктура як послуга» (IaaS), щоб задовольнити ненаситне бажання отримати обчислювальну потужність, сховище та мережеві можливості.

Система управління ресурсами у хмарних середовищах складається з ключових модулів, кожен із яких виконує певну роль у забезпеченні ефективного управління ресурсами, структурна схема системи управління ресурсами у хмарних середовищах наведена в додатку Б. Інтерфейс користувача забезпечує доступ адміністратора до системи, дозволяючи отримувати візуалізовані дані про стан ресурсів та передавати команди управління. Основна інформація для інтерфейсу надходить із модуля моніторингу, який збирає метрики про стан ресурсів із хмарних серверів. Ці дані надсилаються до бази даних для збереження та створення історичних записів, а також передаються в модуль збору та підготовки даних для подальшої обробки.

Модуль збору та підготовки даних виконує очищення та нормалізацію сирих даних, отриманих від модуля моніторингу, після чого підготовлені дані передаються до модуля прогнозування навантаження. Останній, використовуючи алгоритми машинного навчання, прогнозує майбутнє навантаження на ресурси та передає результати у модуль масштабування**.** Модуль масштабування, у свою чергу, аналізує прогнози та поточний стан ресурсів, приймає рішення про масштабування (горизонтальне чи вертикальне) і передає автоматизовані команди до модуля управління ресурсами.

Модуль управління ресурсами виконує команди щодо додавання, видалення або зміни конфігурації ресурсів у хмарних середовищах. Цей модуль безпосередньо взаємодіє з хмарними серверами, реалізуючи всі рішення, прийняті системою. Крім того, він записує виконані дії у базу даних для аудиту та збереження історії змін. База даних відіграє центральну роль у системі, забезпечуючи зберігання оперативних метрик, історичних даних, параметрів стану ресурсів та виконаних команд.

Основна логіка роботи системи полягає в тому, що вона працює у режимі реального часу, збираючи дані про стан ресурсів, обробляючи їх, прогнозуючи майбутнє навантаження та виконуючи необхідні дії для автоматичного масштабування ресурсів. Усі команди передаються до хмарних серверів, що є кінцевим об'єктом управління. Завдяки такій структурі система забезпечує оптимальне використання ресурсів, підтримує стабільність роботи хмарного середовища та відповідає сучасним вимогам автоматизації.

Однак у міру того, як ринок хмарних послуг зростає, завдання CSP стає дедалі складнішим: виконання зобов’язань щодо QoS при одночасному ефективному управлінні динамічними та різноманітними потребами в обчислювальних ресурсах [3].

Діаграма прецедентів відображає ключові функції системи управління ресурсами в хмарних середовищах та їх взаємодію з відповідними акторами, діаграма прецедентів системи управління ресурсами у хмарних середовищах наведена в додатку В.

Адміністратор системи виконує основні операції з керування ресурсами. Зокрема, адміністратор взаємодіє із функцією "Моніторинг стану системи", яка дозволяє отримувати актуальну інформацію про поточний стан ресурсів. Це забезпечує можливість прийняття обґрунтованих рішень щодо подальшого управління ресурсами. Також адміністратор має доступ до функції "Управління ресурсами", через яку здійснюється ручне налаштування ресурсів, включаючи додавання, видалення або модифікацію параметрів хмарних серверів..

Автоматизовану частину системи представляє Системний монітор, заснований на моделі машинного навчання. Цей монітор працює на основі даних, отриманих через "Моніторинг стану системи", що є ключовою складовою для забезпечення актуальності прогнозів. Системний монітор виконує функцію "Прогнозування навантаження та автоматичне масштабування", яка відповідає за аналіз поточного навантаження та прийняття рішень щодо автоматичного масштабування ресурсів у хмарних середовищах.

Таким чином, діаграма охоплює ключові аспекти роботи системи, розподіляючи відповідальність між адміністратором, який здійснює ручне управління, і автоматизованими компонентами, які забезпечують ефективну обробку даних та адаптивність системи в умовах змінного навантаження.

У результаті цієї змінної динаміки поняття об’єднаних хмарних обчислень перетворилося на переконливу парадигму. Постачальники хмарних послуг об’єднують зусилля в рамках цього об’єднаного підходу, створюючи екосистему спільної роботи, де вони об’єднують і ділять свої невикористовувані комп’ютерні ресурси. Отримане об’єднання, відоме як розподілені хмарні обчислення, має низку переваг, зокрема підвищену доступність і надійність[4]. Цей спільний підхід дозволяє федерації подолати властиві обмеження, з якими стикаються окремі CSP, коли мова йде про підтримку QoS, особливо в періоди високого попиту на ресурси або низького використання. Об’єднана хмарна архітектура — це гетерогенний і розподілений підхід, який надає пов’язану з хмарою інфраструктуру шляхом об’єднання багатьох постачальників IaaS. Ця модель також відома як об’єднана хмара[5].

Діаграма компонентів відображає основні модулі системи управління ресурсами, їхні функції та взаємозв’язки, відповідна діаграма компонентів системи управління ресурсами у хмарних середовищах наведена в додатку Г.

Система складається з восьми ключових компонентів, кожен із яких виконує певну роль у забезпеченні роботи всієї системи.

Модуль моніторингу відповідає за збір даних про стан ресурсів у хмарному середовищі, він отримує метрики, такі як завантаження CPU, використання RAM і

дискового простору, а також час останнього оновлення цих даних. Модуль передає сирі метрики до модуля збору та підготовки даних для подальшої обробки, а також записує оперативні метрики до бази даних. Крім того, метрики доступні для перегляду адміністратором через інтерфейс користувача.

Модуль збору та підготовки даних очищує та нормалізує дані, отримані від модуля моніторингу, готуючи їх для подальшого аналізу. Сирі метрики обробляються для видалення шуму та заповнення пропусків, після чого очищені дані передаються в модуль прогнозування навантаження.

Модуль прогнозування навантаження аналізує підготовлені дані для створення прогнозів майбутнього навантаження на систему, використовуючи алгоритми машинного навчання. Прогнозовані дані передаються до модуля масштабування для прийняття рішень про зміну ресурсів. Також результати прогнозів зберігаються у базі даних.

Модуль управління ресурсами є кінцевою ланкою, яка реалізує всі команди щодо додавання, видалення чи зміни конфігурації ресурсів у хмарному середовищі. Він передає відповідні команди до хмарного середовища та записує журнал виконаних команд у базу даних.

Інтерфейс користувача забезпечує доступ адміністратора до системи. Через цей інтерфейс адміністратор може переглядати метрики, надані модулем моніторингу, а також надсилати ручні команди для управління ресурсами.

База даних являється централізованим сховищем, яке зберігає історію метрик, прогнозів навантаження, масштабувань та виконаних команд. Вона забезпечує доступ до історичних даних для аналізу та звітності.

Хмарне середовище, компонент який є кінцевим об’єктом управління. Він отримує команди від модуля управління ресурсами та реалізує їх у вигляді змін конфігурації ресурсів, серед них є додавання серверів і змінення потужностей.

Маємо такі взаємозв’язки між компонентами:

– модуль моніторингу отримує метрики від хмарного середовища та передає їх до модуля збору й підготовки даних, бази даних і інтерфейсу користувача;

– модуль збору та підготовки даних очищує метрики та передає їх до модуля прогнозування навантаження;

– модуль прогнозування навантаження аналізує дані та надсилає прогнози до модуля масштабування, а також записує результати у базу даних;

– модуль масштабування приймає рішення про зміни ресурсів та передає команди до модуля управління ресурсами;

– модуль управління ресурсами виконує команди змін конфігурацій у хмарному середовищі та зберігає виконані дії у базі даних.

Ця структура забезпечує ефективне управління ресурсами у хмарному середовищі, оптимізуючи використання ресурсів і адаптуючи систему до змінного навантаження.

За цих обставин вибір найкращого хмарного постачальника послуг і його розгортання за низькою ціною є цікавою роботою, яку потрібно виконати. Здатність об’єднаних хмарних обчислень забезпечувати ефективне використання комп’ютерних ресурсів, навіть коли загальний попит низький, є особливою особливістю, яка заслуговує на увагу[6]. Ця унікальна можливість підкреслює необхідність впровадження відповідних стратегій управління ресурсами для послуг IaaS, що надаються CSP хмарної федерації. Об’єднані хмарні обчислення IaaS підтримують багато типів центрів обробки даних . Загальнодоступні хмарні центри обробки даних, якими керують основні постачальники хмарних послуг, такі як Amazon Web Services (AWS), Microsoft Azure та Google Cloud Platform (GCP), є типовими прикладами таких центрів обробки даних. Важливу роль також відіграють приватні центри обробки даних, які належать і обслуговуються окремими організаціями чи корпораціями. У об’єднаних хмарних конфігураціях також широко поширені гібридні центри обробки даних, які поєднують як публічну, так і приватну хмарну технологію.

Діаграма діяльності відображає основні етапи роботи системи управління ресурсами у хмарному середовищі, відповідна діаграма діяльності для автоматичного масштабування у системі управління ресурсами наведена в додатку Д. Процес розпочинається зі збору даних про стан ресурсів. На першому етапі

"Модуль моніторингу" отримує метрики, такі як завантаження CPU, використання RAM і дискового простору, а також час останнього оновлення. Ці метрики записуються у базу даних як оперативні дані, що дозволяє створювати історичні записи для подальшого аналізу.

Після цього дані передаються до "Модуля збору та підготовки даних", де вони очищуються від шумів, нормалізуються і готуються для аналізу. Далі ці підготовлені дані надходять до "Модуля прогнозування навантаження", який за допомогою алгоритмів машинного навчання створює прогноз майбутнього навантаження. Прогнозовані значення також зберігаються у базі даних для подальшого використання.

На основі прогнозу система переходить до прийняття рішень. Спочатку перевіряється, чи прогнозоване навантаження перевищує заданий поріг. Якщо ні, система повертається до моніторингу і продовжує збір нових метрик. Якщо прогнозоване навантаження перевищує порогове значення, проводиться додаткова перевірка: чи є доступні ресурси для масштабування. Якщо ресурси недоступні, система також повертається до моніторингу.

У разі доступності ресурсів виконується масштабування. "Модуль масштабування" генерує команди, які передаються до "Модуля управління ресурсами". Ці команди надсилаються до хмарного середовища, де відбуваються зміни конфігурації ресурсів, такі як додавання серверів чи збільшення потужностей. Усі виконані дії записуються у базу даних як результати масштабування.

Після виконання масштабування система знову повертається до моніторингу, завершуючи цикл управління ресурсами. Діаграма діяльності демонструє ефективну взаємодію між модулями системи, узгоджену з базою даних і хмарним середовищем. Вона забезпечує гнучке управління ресурсами, дозволяючи системі адаптуватися до змінного навантаження, мінімізуючи ризики перевантаження та забезпечуючи оптимальну продуктивність.

Взаємозв’язані центри обробки даних і периферійні центри обробки даних, розташовані ближче до кінцевих споживачів, можуть покращити продуктивність

об’єднаних хмарних служб і швидкість реагування. Через різноманітність типів центрів обробки даних об’єднані хмарні системи можуть надавати широкий спектр послуг і ресурсів для задоволення різноманітних потреб.

Діаграма послідовностей описує основні етапи взаємодії компонентів системи управління ресурсами в хмарному середовищі, відповідна Діаграма потоків даних у системі управління ресурсами у хмарних середовищах наведена в додатку Е. Процес починається зі збору даних про стан ресурсів через модуль моніторингу. Цей модуль отримує метрики від хмарного середовища (використання CPU, RAM, дискового простору тощо) та передає ці дані до бази даних для збереження історичних записів і подальшого аналізу.

Після цього зібрані дані передаються до модуля очищення та нормалізації, де вони обробляються для усунення шумів і приведення до уніфікованого формату. Оброблені дані надходять до модуля прогнозування навантаження, який за допомогою алгоритмів машинного навчання створює прогнози майбутнього використання ресурсів.

На основі цих прогнозів модуль масштабування аналізує стан системи та приймає рішення про необхідність змін у конфігурації ресурсів. Якщо прогнозоване навантаження перевищує порогові значення, перевіряється наявність доступних ресурсів для масштабування. У разі наявності ресурсів генерується команда, яка передається модулю управління ресурсами.

Модуль управління ресурсами виконує масштабування, додаючи нові ресурси або оптимізуючи використання існуючих. Результати масштабування, як і всі інші дії, зберігаються в базі даних для забезпечення історичних записів.

Система працює циклічно: після виконання масштабування або за відсутності необхідності в ньому, система повертається до етапу моніторингу для збору нових метрик. Така структура забезпечує адаптивність системи, дозволяє мінімізувати затримки та оптимально використовувати ресурси, підтримуючи стабільність роботи хмарного середовища.

Хмарне середовище, швидше за все, стикається з нездатністю хмарних провайдерів обробляти дані користувачів, недостатньою прозорістю та ризиками

неправильного управління, що все може мати значний негативний вплив на репутацію хмарних провайдерів[24].

Як результат, дуже необхідно встановити довіру, щоб хмарне середовище виглядало надійним. Дослідження об’єднаних хмар стало більш поширеним протягом останніх кількох років як засіб вирішення проблем, які потребують обчислень і даних у широкому масштабі. Через складність моделей надання послуг довірче управління є абсолютною потребою в хмарних службах, які працюють у децентралізованому режимі. Щоб забезпечити успішне розгортання у відкритому, динамічному та непередбачуваному об’єднаному хмарному середовищі[16], дуже важливо, щоб між користувачами, постачальниками хмарних послуг і хмарними провайдерами були встановлені довірчі відносини. Існує постійна потреба в інноваційних протоколах та інструментах, які можуть покращити та оцінити рівень безпеки, що забезпечується послугою хмарних обчислень [10], його брокерів або постачальника послуг.

Діаграма станів ілюструє логіку роботи системи управління ресурсами в хмарному середовищі, розкриваючи основні етапи її функціонування, відповідна діаграма станів для системи управління ресурсами у хмарних середовищах наведена в додатку Ж. Процес розпочинається зі стану моніторингу, де система збирає дані про стан ресурсів, таких як завантаження CPU, RAM, дискового простору та мережеве навантаження. Зібрані дані передаються на етап очищення, де видаляються шуми та некоректні значення. Після цього очищені дані нормалізуються, що дозволяє привести їх до єдиного формату, придатного для обробки. На етапі прогнозування дані аналізуються за допомогою моделі машинного навчання, яка визначає можливе майбутнє навантаження на ресурси. Якщо прогнозоване навантаження перевищує встановлений поріг, система переходить до перевірки доступності ресурсів. У випадку, якщо навантаження нижче порогового значення, система повертається до моніторингу. Під час перевірки ресурсів здійснюється аналіз їх доступності. Якщо ресурси доступні, система переходить до стану масштабування, де виконується оптимізація ресурсів: додаються нові сервери, перерозподіляються задачі або змінюються параметри

існуючих ресурсів. У разі виникнення невдачі передбачено повторні спроби масштабування до досягнення ліміту. Якщо всі спроби масштабування вичерпані, система переходить до стану помилки, де фіксується невдача у журналі для подальшого аналізу, після чого повертається до моніторингу. Після успішного масштабування система оновлює відповідні записи в базі даних, зберігаючи інформацію про виконані зміни, та повертається до моніторингу, завершуючи цикл. Ця діаграма демонструє адаптивний і циклічний характер роботи системи, який забезпечує її ефективність навіть у динамічних умовах.

Безпека об’єднаної хмарної служби має охоплювати широкий спектр тем, включаючи автентифікацію , авторизацію, захист даних тощо. Це фундаментальні цілі безпеки, які разом складають принципи безпеки, і вони стають абсолютно необхідними під час переходу до хмари. Щоб належним чином вирішити ці проблеми безпеки стосовно хмарних служб до вибору, хмарне середовище вимагає використання інструменту, який може проводити оцінку ризиків. Коли йдеться про досягнення безпеки в архітектурі, довіра є компонентом, який можна оцінити на будь-якому рівні[25].

Коли порівнюються дві сутності, між ними визначається довіра; репутація суб'єкта, з іншого боку, є всеохоплюючою оцінкою цього суб'єкта. Тому на практиці життєво важливо вибирати постачальника з широкого діапазону ознак як доказів для винесення судження про довіру, навіть якщо немає конкретних критеріїв.

Діаграма послідовностей ілюструє процес взаємодії ключових модулів системи управління ресурсами в хмарному середовищі, відповідна діаграма послідовності для автоматичного масштабування ресурсів у хмарних середовищах наведена в додатку И. Процес починається з того, що "Системний монітор" збирає дані про стан ресурсів, такі як завантаження CPU, RAM, дискового простору, мережевого навантаження тощо. Ці дані передаються до "Модуля прогнозування", який за допомогою алгоритмів машинного навчання аналізує отриману інформацію та генерує прогнозоване навантаження на ресурси..

Після цього "Модуль прогнозування" передає прогнозоване навантаження до "Модуля управління ресурсами", який виконує оцінку необхідності масштабування. Для цього "Модуль управління ресурсами" звертається до "Бази даних" із запитом історичних даних про стан ресурсів. Використовуючи історичні дані та прогнозоване навантаження, модуль приймає рішення щодо масштабування.

У разі потреби в масштабуванні "Модуль управління ресурсами" передає відповідну команду до "Модуля масштабування". Останній виконує масштабування, додаючи нові ресурси або оптимізуючи використання існуючих, а потім оновлює інформацію про ресурси в "Базі даних". Після завершення масштабування "Модуль масштабування" надсилає підтвердження виконання змін до "Модуля управління ресурсами", а той, у свою чергу, передає цю інформацію до "Системного монітора".

Цикл завершується поверненням до "Системного монітора", що дозволяє системі продовжувати процес моніторингу та підтримувати стабільність роботи в хмарному середовищі. Ця діаграма послідовностей чітко демонструє взаємодію між компонентами системи, підкреслюючи логіку процесу управління ресурсами.

Об’єднані хмари забезпечують більшу кількість ресурсів, що допомагає підвищити як економічну ефективність, так і якість. Це один із способів підвищення економічної ефективності. Це включає покращення як для користувача, так і для постачальника, наприклад, зменшення кількості часу, необхідного для виконання завдання за певної вартості, збільшення пропускної здатності, яку може впоратися з системою, або оптимізація способу використання ресурсів[26].

Коли хмара виявляє, що її центр обробки даних недостатньо використовується в певний момент, вона має можливість вирішити, чи зробити свої ресурси доступними для інших хмар. Сам центр обробки даних неможливо вимкнути. Розташування в різних частинах земної кулі: компанії хмарних послуг створюють свої центри обробки даних по всьому світу. У результаті з’являється можливість розподілу навантаження, а також підвищення продуктивності.

Хмарний клієнт може легко уникнути прив’язки до одного постачальника, використовуючи багато хмар і зберігаючи здатність плавно переміщувати робочі навантаження між ними[27].

Якщо постачальник вносить зміни у свою політику або ціни, які негативно впливають на його клієнтів, ці клієнти можуть швидко переключитися на іншу послугу. Через конкуренцію постачальник хмарних послуг може забезпечити кращі угоди про рівень обслуговування (SLA) для своїх клієнтів. Гарантована продуктивність через обмежені ресурси, доступні одному постачальнику хмарних послуг, раптове збільшення робочого навантаження може призвести до зниження продуктивності[11]. Продуктивність може бути гарантована за допомогою кількох постачальників хмарних послуг. Хмарна федерація може подолати цей недолік, орендуючи ресурси у міжнародних постачальників хмарних послуг, таким чином забезпечуючи якість обслуговування, про яку було домовлено раніше. Гарантована доступність: хмарна система зможе відновити послуги шляхом об’єднання з іншими постачальниками хмарних послуг у незмінній формі у разі непередбачених природних катастроф.

Модель бази даних розроблена для забезпечення ефективного управління ресурсами в хмарному середовищі та включає шість основних таблиць: "Користувачі", "Ресурси", "Прогнози", "Масштабування", "Журнал подій" і "Системні параметри", відповідна схема бази даних для системи управління ресурсами у хмарних середовищах наведена в додатку Й. Таблиця "Користувачі" містить інформацію про користувачів системи, включаючи їх унікальний ідентифікатор, роль, ім’я користувача, електронну пошту та хешований пароль, і пов’язана з таблицею "Журнал подій", що дозволяє фіксувати дії користувачів у системі, зокрема ініціацію масштабування або зміну конфігурацій. Таблиця "Ресурси" зберігає дані про ресурси хмарного середовища, включаючи ідентифікатор ресурсу, тип ресурсу (наприклад, CPU, RAM, Disk), поточне навантаження, статус ресурсу (доступний або зайнятий), локацію, власника ресурсу та дату створення. Ця таблиця пов’язана з таблицями "Прогнози" та "Масштабування", що дозволяє аналізувати використання ресурсів і проводити їх

оптимізацію. Таблиця "Прогнози" використовується для збереження результатів прогнозування навантаження і містить атрибути, такі як ідентифікатор прогнозу, ідентифікатор ресурсу, прогнозоване навантаження, модель прогнозування, точність прогнозу, відхилення від реальних даних, час прогнозу та дату створення. Зв’язок із таблицею "Масштабування" дозволяє відстежувати, які прогнози стали причиною змін у ресурсах. Таблиця "Масштабування" містить дані про виконані операції масштабування, зокрема ідентифікатор масштабування, тип масштабування, ідентифікатор ресурсу, результат виконання (успішно/не успішно), ідентифікатор прогнозу, тип ресурсу, причину масштабування та дату/час виконання. Вона пов’язана з таблицями "Ресурси" та "Прогнози" для відстеження ефективності змін. Таблиця "Журнал подій" слугує для аудиту системи та зберігає записи про всі події, включаючи ідентифікатор події, тип події (наприклад, помилка, зміна параметрів, прогнозування), час події, ідентифікатор користувача, ідентифікатор прогнозу або масштабування (якщо подія пов’язана з цими таблицями), а також деталі події. Такий підхід забезпечує можливість аналізу та діагностики роботи системи. Таблиця "Системні параметри" містить глобальні налаштування системи, такі як порогові значення для ресурсів (наприклад, CPU > 80%), конфігураційні параметри ML-моделі та час між перевірками стану ресурсів. Зв’язки між таблицями інтегрують дані: "Ресурси" пов’язані з "Прогнозами" та "Масштабуванням", "Користувачі" мають зв’язок із "Журналом подій" і "Масштабуванням", а "Журнал подій" дозволяє відстежувати ключові події в системі.

Така структура моделі бази даних гарантує надійність, ефективність та зручність аналізу інформації в системі управління ресурсами..

Однак існує значний проміжок у дослідженнях у створенні комплексних моделей довіри та механізмів безпеки, здатних адаптуватися до постійно змінюваних загроз і вразливостей, притаманних хмарним системам . Ефективний розподіл ресурсів у мультихмарних і об’єднаних хмарних установках залишається складністю. Нинішній сукупності досліджень бракує складності, необхідної для розробки вдосконалених алгоритмів і методів для оптимального розподілу

ресурсів, налаштованих для задоволення різних потреб користувачів, уникаючи при цьому марнування ресурсів. Забезпечення узгодженого та надійного QoS для багатьох постачальників хмарних послуг є серйозною проблемою. Щоб вирішити цю проблему, існує нагальна потреба в дослідницьких зусиллях, спрямованих на створення методів, здатних передбачати, контролювати та забезпечувати гарантії якості обслуговування в динамічній та різноманітній екосистемі хмарних обчислень. На щастя, запропонований хмарний дизайн IaaS усуває ці прогалини в дослідженнях, роблячи значний внесок у ці ключові сфери.

Хмарний дизайн IaaS представляє новаторську концепцію довіри з точки зору довіри та безпеки. Ця методологія динамічно оцінює надійність постачальників хмарних послуг, використовуючи дані в реальному часі про продуктивність постачальників, події безпеки та дані клієнтів для постійного оновлення показників довіри.

Крім того, архітектура включає передові методи безпеки, такі як шифрування, виявлення вторгнень і обмеження доступу для захисту даних користувачів і робочих навантажень, успішно подолаючи розрив, забезпечуючи адаптовану та стійку структуру довіри та безпеки. Запропоноване рішення включає вдосконалений алгоритм розподілу ресурсів, здатний максимізувати використання ресурсів у різних хмарах. Щоб динамічно призначати ресурси, він враховує різноманітні критерії, такі як моделі робочого навантаження, проблеми з витратами та можливості постачальника.

Цей комплексний метод зменшує марну витрату ресурсів і підвищує загальну ефективність, заповнюючи прогалину в дослідженнях щодо оптимального розподілу ресурсів. З точки зору гарантії QoS, хмарний дизайн IaaS містить модуль прогнозування та моніторингу QoS, який постійно перевіряє продуктивність постачальників хмарних послуг. Цей модуль прогнозує та забезпечує рівні QoS шляхом поєднання історичних даних , вимірювань у реальному часі та визначених користувачем параметрів. Крім того, архітектура має методи для автоматизованого переміщення робочого навантаження для забезпечення якості обслуговування під час неочікуваних збоїв постачальника.

Висновки до розділу3

У розділі розроблено архітектуру системи управління ресурсами в хмарних середовищах та поставлено завдання створення ефективного механізму управління для забезпечення якості обслуговування (QoS). Було визначено ключові вимоги до системи, включаючи динамічний розподіл ресурсів, адаптацію до змін у навантаженні та мінімізацію затримок.

Розроблено архітектуру системи, яка інтегрує сучасні методи управління ресурсами із застосуванням хмарних технологій. Представлено UML-діаграми, які демонструють функціонування основних компонентів системи, таких як автоматичне масштабування, управління навантаженням та забезпечення безпеки даних.

Запропонована система базується на модульному підході, що забезпечує її масштабованість, гнучкість і адаптивність до різних умов роботи. Особливу увагу приділено інтеграції принципів довіри та безпеки в управлінні ресурсами, що дозволяє підвищити надійність хмарних середовищ і підтримувати високу якість обслуговування навіть у динамічних умовах.

Результати цього розділу є основою для розробки інтелектуальної системи управління ресурсами, яка сприятиме ефективному використанню ресурсів у хмарних середовищах, підвищенню продуктивності та зниженню витрат.

4 РОЗРОБКА МОДЕЛІ УПРАВЛІННЯ РЕСУРСАМИ З ВИКОРИСТАННЯМ МАШИННОГО НАВЧАННЯ

4.1 Вибір моделі машинного навчання

Метою цього підрозділу є вибір найбільш підходящої моделі машинного навчання для оптимізації управління ресурсами в хмарних середовищах.

Як відомо, аналіз величезних масивів даних, доступних сьогодні, став рушійною силою для бізнесу. Паралельна обробка на товарних кластерах стала де-факто способом впоратися з масштабом і складністю таких додатків, що вимагають великих обсягів даних. Для прискорення роботи додатків на декількох машинах використовується модель масово синхронно-паралельних розподілених обчислень.

У масивно-синхронно-паралельній моделі обчислювальне завдання, яке ставиться перед кластером комп'ютерів, спочатку розбивається на етапи, а потім кожен етап розбивається на кілька завдань. Завданням на етапі призначаються однакові обчислення або різні блоки вхідних даних. Завдання, що належать до кожного етапу, можуть виконуватися паралельно. Планувальник кластера призначає ці завдання машинам (вузлам), де вони виконуються паралельно.

Обчислювальна робота синхронізується на межі різних етапів: вихідні дані попереднього етапу використовуються як вхідні для наступного. Таке змішування проміжних даних зазвичай виконується завданнями попереднього етапу, що записують дані на локальний диск, і завданнями наступного етапу, що зчитують їх.

Робота завершується, коли всі її завдання завершили виконання. Ключовою особливістю таких фреймворків є те, що вони автоматично обробляють збої (які більш ймовірні в кластері стандартних комп'ютерів, схильних до збоїв) без додаткових зусиль з боку програміста.

Навчання з учителем – це гілка алгоритмів машинного навчання, у якій будуються прогностичні моделіз відомих пар вхідних і цільових даних, які називаються позначений набір даних. Загалом, заданий набір даних із мітками розділяється навчальний набір, набір перевірки та тестовий набір. Алгоритми навчання під наглядом спрямовані на навчання функція, яка також називається

моделлю, яка може зіставляти вхідні дані з відповідними цільовими значеннями в навчальний набір даних. Вивчені моделі перевіряються на перевірочному наборі, який модель не бачила під час фази навчання. Модель, яка демонструє найкращі показники ефективності, наприклад точність прогнозування, на валідаційному наборі даних, обирається як результат етапу навчання. На етапі тестування обрана модель використовується для прогнозування цільових значень (міток) у тестовому наборі. Істинні мітки у тестовому наборі (ground truth) використовуються для оцінки точності та продуктивності навченої моделі.

Контрольоване навчання включає дві категорії алгоритмів:

– класифікація: для категорійних значеннях міток;

– регресія: для міток із дійсними значеннями.

Машини опорних векторів (SVM) стали популярним готовим алгоритмом навчання під наглядом протягом останніх кількох десятиліть, особливо завдяки його кращим емпіричним показникам порівняно з іншими алгоритми. SVM використовує теорію машинного навчання, щоб максимізувати точність прогнозування в автоматичному режим уникнення надмірної відповідності навчальним даним. Автор роботи [11] Вапник розробив основи SVM з використанням мінімізації структурного ризику (SRM), або регуляризований принцип мінімізації емпіричного ризику (ERM) [43], який дозволяє SVM узагальнювати добре для невидимих наборів даних.

Використовуючи прогнози з моделей, побудованих раніше, модель-інформований планувальник потім вибірково затримує початок виконання завдання, якщо цей вузол передбачено створений відстаючим. Завдання відкладається лише за умови впевненості у відповідному прогнозі перевищує мінімально необхідну впевненість. Це дозволяє уникнути перевантаження вузлів, тим самим зменшуючи їх шанси на створення відстаючих. Основний підхід полягає в тому, щоб відкласти призначення завдання до моменту, який із найбільшою ймовірністю зможе завершити його вчасно.

Це рішення запровадило затримку запуску завдання до моменту, коли обраний вузол перестане бути перевантаженим, або знайдено інший менш завантажений вузол.

Для вирішень задач прогнозування дуже добре показує свою ефективність модель RandomForestRegressor, яка є частиною бібліотеки Scikit-learn[49]. Ця модель використовує набір дерев рішень для прогнозування, об’єднуючи їх результати шляхом усереднення. Завдяки цій техніці модель демонструє стійкість до перенавчання, ефективно працює з великими наборами даних та враховує взаємозалежності між ознаками.

Обрана модель RandomForestRegressor стане основою для реалізації планувальника задач у хмарному середовищі, дозволяючи прогнозувати навантаження та ефективно розподіляти ресурси. У наступному підрозділі буде розглянуто методи її реалізації.

4.2 Реалізація алгоритмів та тестування моделі машинного навчання для управління ресурсами

4.2.1 Огляд необхідних інструментів для розробки

Psutil (processandsystemutilities) – кросплатформна бібліотека для отримання інформації про запущені процеси та використання системи (процесор, пам'ять, диски, мережа, датчики) у Python. Вона корисна в основному для моніторингу системи, профілювання та обмеження ресурсів процесів, а також для управління запущеними процесами. Він реалізує багато функціональних можливостей класичних інструментів командного рядка UNIX, таких як ps, top, iotop, lsof, netstat, ifconfig, free та інші. psutil наразі підтримує наступні платформи:

- Linux;
- Windows;
- macOS;
- FreeBSD, OpenBSD, NetBSD;
- Sun Solaris;

– AIX.

На рисунку 4.1 зображено бібліотеки використані в роботі

Рисунок 4.1 – Бібліотеки використані в роботі

На рисунку 4.2 зображено процес написання коду проекту

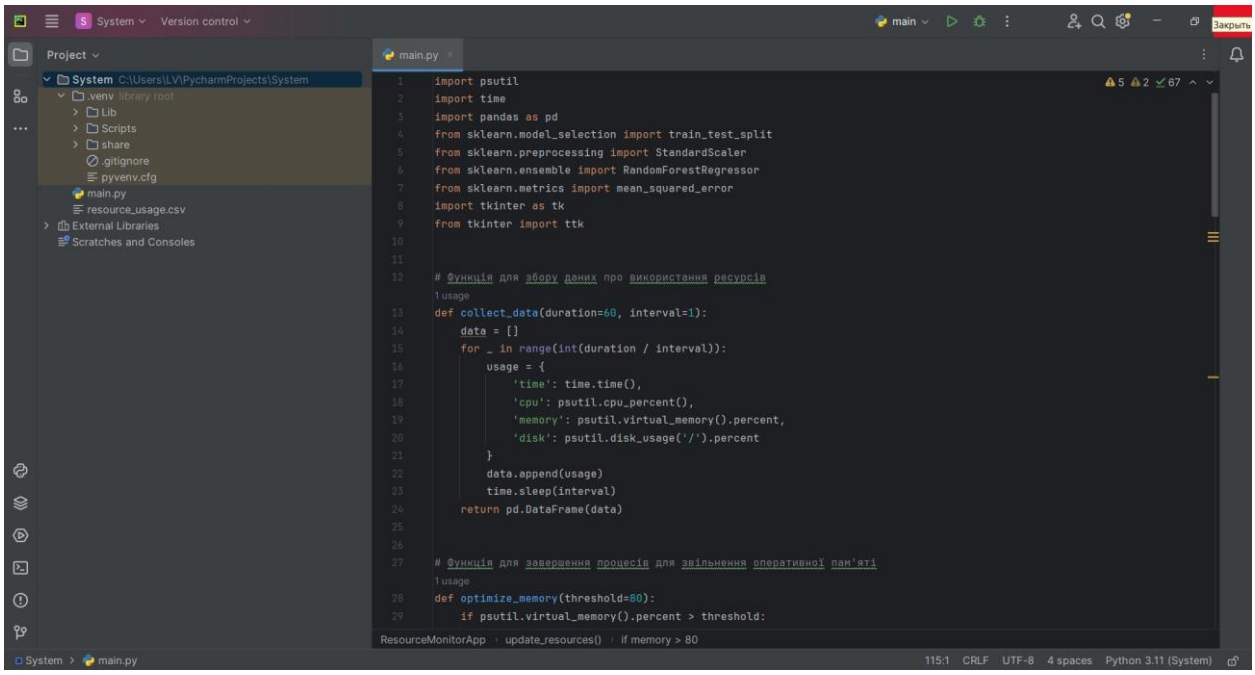

Рисунок 4.2 – Процес написання коду проекту

Pandas – це пакет Python, який надає швидкі, гнучкі та виразні структури даних, призначені для того, щоб зробити роботу з «реляційними» або «розміченими» даними простою та інтуїтивно зрозумілою. Він має на меті стати фундаментальним високорівневим будівельним блоком для виконання практичного аналізу реальних даних мовою Python. Крім того, він має ширшу мету – стати найпотужнішим і найгнучкішим інструментом аналізу та маніпулювання

даними з відкритим вихідним кодом, доступним будь-якою мовою. Pandas вже активно використовується для досягнення цієї мети. Ось деякі з основних можливостей, які pandas робить добре, а саме:

– легка обробка відсутніх даних (представлених як NaN, NA або NaT) у форматі з плаваючою комою, а також даних без плаваючої коми;

– змінність розмірів: стовпці можна вставляти та видаляти з DataFrame та об'єктів більшої розмірності.

На рисунку 4.3 зображено файли проекту

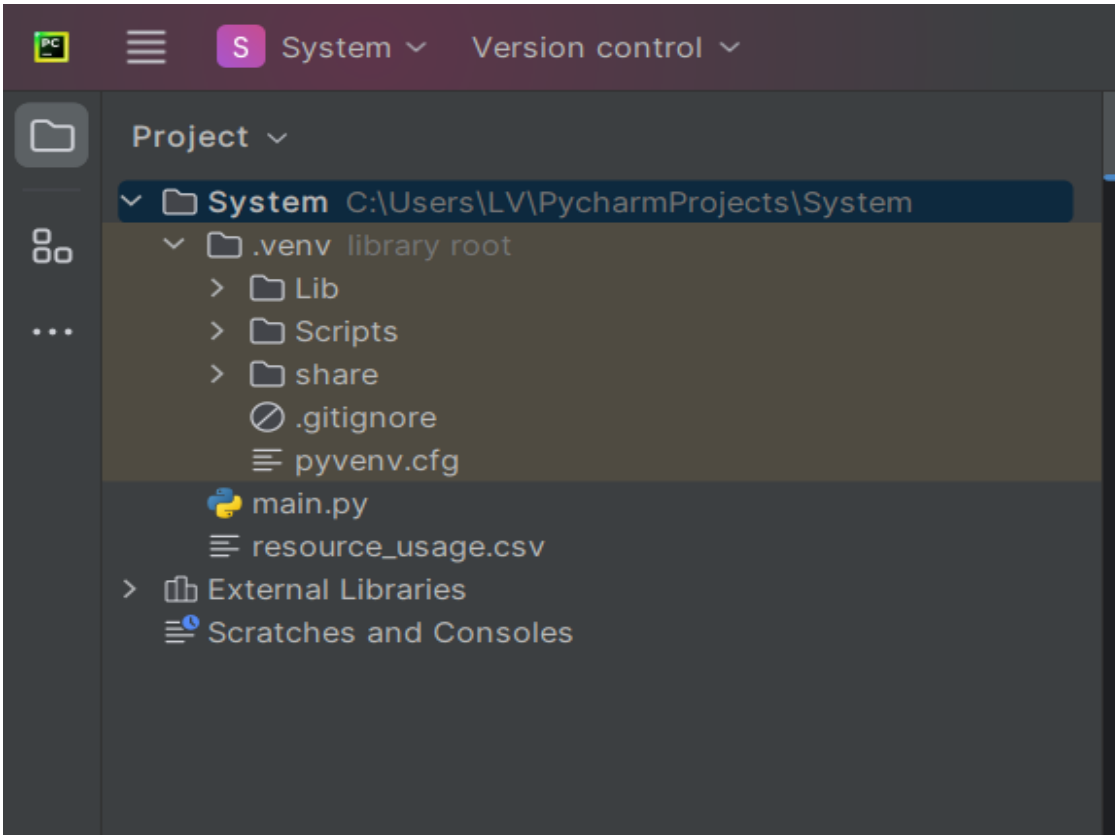

Рисунок 4.3 – Файли проекту

Автоматичне та явне вирівнювання даних: об'єкти можна явно вирівняти за набором міток, або користувач може просто ігнорувати мітки і дозволити Series, DataFrame тощо автоматично вирівнювати дані під час обчислень.

Потужні, гнучкі групи за функціональністю для виконання операцій розділення, застосування та об'єднання наборів даних, як для агрегування, так і для перетворення даних.

Спрощення перетворення розрізнених, по-різному індексованих даних в інших структурах даних Python та NumPy в об'єкти DataFrame.

Інтелектуальна нарізка на основі міток, фантазійне індексування та підмножини великих наборів даних.

Інтуїтивне злиття та приєднання наборів даних.

Гнучка зміна форми та обертання наборів даних.

Scikit-learn, також відомий як sklearn, — це бібліотека машинного навчання та моделювання даних з відкритим кодом для Python. Він містить різні алгоритми класифікації, регресії та кластеризації, включаючи машини опорних векторів, випадкові ліси, посилення градієнта, k-середні та DBSCAN, і розроблений для взаємодії з бібліотеками Python, NumPy та SciPy.

Scikit-learn був вперше випущений у 2010 році, і з тих пір він зайняв помітне місце в екосистемі машинного навчання Python . Він реалізує численні алгоритми моделювання даних і машинного навчання, а також надає послідовні API Python. Він підтримує стандартизований і стислий інтерфейс моделі для всіх моделей. Наприклад, Scikit-learn використовує просту модель робочого процесу підгонки/прогнозування для своїх алгоритмів класифікації.

Scikit-learn добре інтегрується з багатьма іншими бібліотеками Python, такими як matplotlib і plotly для побудови графіків, NumPy для векторизації масивів, фрейми даних Pandas, SciPy та багато інших. Ви можете передавати масиви NumPy і фрейми даних Pandas безпосередньо в алгоритми Scikit-learn.

Він надає повний набір контрольованих і неконтрольованих алгоритмів навчання , що охоплює такі сфери, як:

– перевірка наявності балансу на системному гаманці, що використовується для операцій.

– класифікація: визначення категорії, до якої належить об’єкт;

– регресія: передбачення атрибута безперервного значення, пов’язаного з об’єктом;

– кластеризація: автоматичне групування схожих об’єктів у набори з такими моделями, як k-середні;

– зменшення розмірності: зменшення кількості атрибутів у даних для підсумовування, візуалізації та вибору функцій за допомогою таких моделей, як аналіз основних компонентів (PCA);

– вибір моделі: порівняння, перевірка та вибір параметрів і моделей;

– попередня обробка: виділення та нормалізація функцій, включаючи визначення атрибутів у даних зображення та тексту.

Scikit-learn здебільшого написаний на Python і широко використовує NumPy для високопродуктивної лінійної алгебри та операцій з масивами. Деякі основні алгоритми написані на Python для підвищення продуктивності.

4.2.2 Огляд основних етапів роботи моделі

Одним із основних етапів роботи системи є збір даних про використання ресурсів. Для цього у програмі використовується функція collect_data, яка виконує моніторинг таких показників, як завантаженість процесора (CPU), використання оперативної пам'яті (RAM) та заповненість диску. Ця функція періодично збирає дані протягом заданого інтервалу часу, створюючи структуру даних у вигляді таблиці (DataFrame)

Функція працює у циклі, де через кожний інтервал часу (interval) вона зчитує актуальні показники системи за допомогою бібліотеки psutil. Зібрані дані додаються у список, який пізніше перетворюється на DataFrame для подальшої обробки. Такий підхід дозволяє отримувати інформацію у реальному часі, що є критично важливим для ефективного управління ресурсами.

На рисунку 4.4 зображений код функції для збору даних про використання ресурсів з середовища.

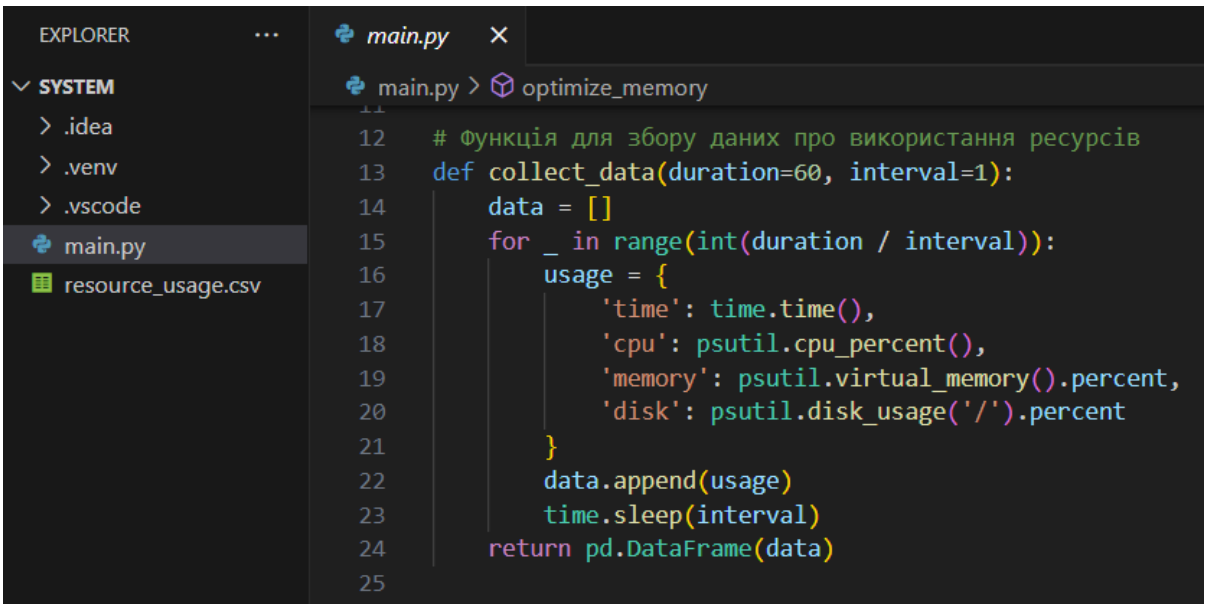

Рисунок 4.4 – Код функції для збору даних про використання ресурсів з середовища

Для уникнення перевантаження системи за умов високого використання оперативної пам'яті передбачено механізм її оптимізації. Функція optimize_memory перевіряє, чи перевищує завантаженість пам'яті заданий поріг (наприклад, 80%). У разі перевищення вона намагається завершити процеси, які споживають надмірні ресурси.

На рисунку 4.5 зображений код функції для завершення процесів і звільнення оперативної пам’яті.

Рисунок 4.5 – Код функції для завершення процесів і звільнення оперативної пам’яті

Ця функція виконує перевірку всіх процесів у системі за допомогою методу process_iter і завершує ті, що використовують значну кількість пам'яті. Такий підхід до вирішення наявної проблеми дозволяє швидко звільнити ресурси без потреби у ручному втручанні користувача.

Після збору даних вони використовуються для побудови моделі машинного навчання, яка прогнозує використання дискового простору на основі завантаженості процесора та пам'яті. Для цього використовується алгоритм RandomForestRegressor, що є частиною бібліотеки sklearn.

На рисунку 4.6 зображений код для підготовки та тренування моделі машинного навчання.

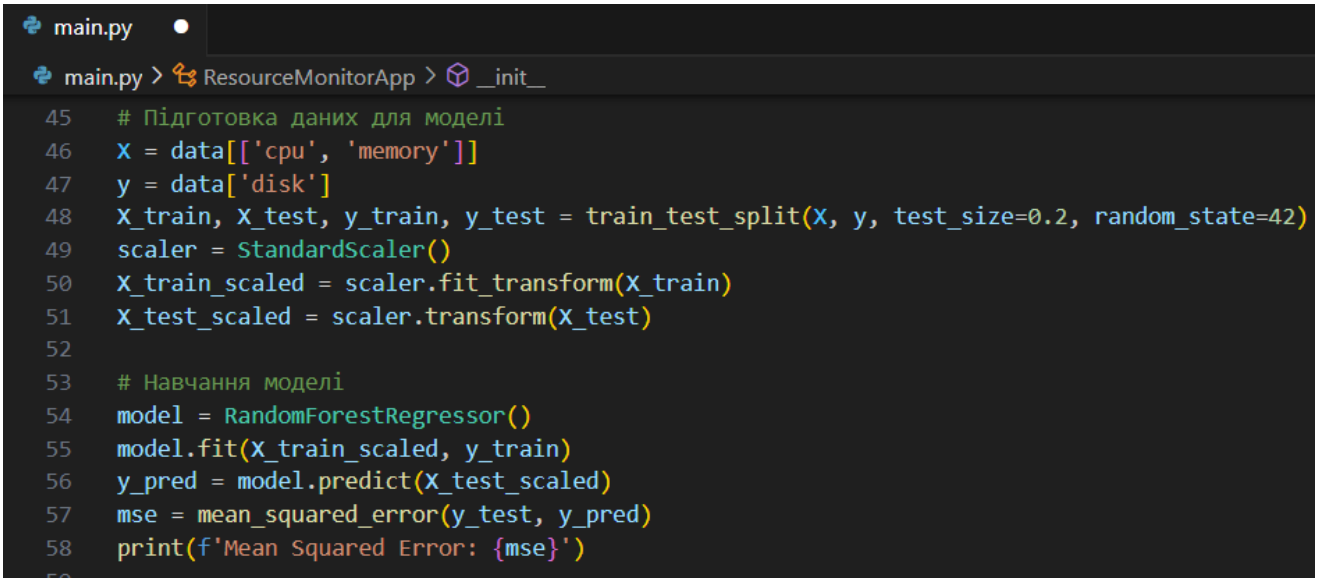

Рисунок 4.6 – Код для підготовки та тренування моделі машинного навчання

На першому етапі здійснюється розподіл даних на навчальний та тестовий набори. Для підвищення точності модель масштабує вхідні дані, використовуючи StandardScaler. Навчання моделі здійснюється на основі навчального набору, а точність оцінюється за допомогою метрики середньоквадратичної помилки (Mean Squared Error). Такий підхід забезпечує високу точність прогнозів.

Для забезпечення зручності взаємодії з системою розроблено графічний інтерфейс користувача (GUI) за допомогою бібліотеки tkinter. Інтерфейс дозволяє відображати поточний стан ресурсів, а також прогнозовані показники.

На рисунку 4.7 зображений фрагмент коду відповідального за графічний інтерфейс.

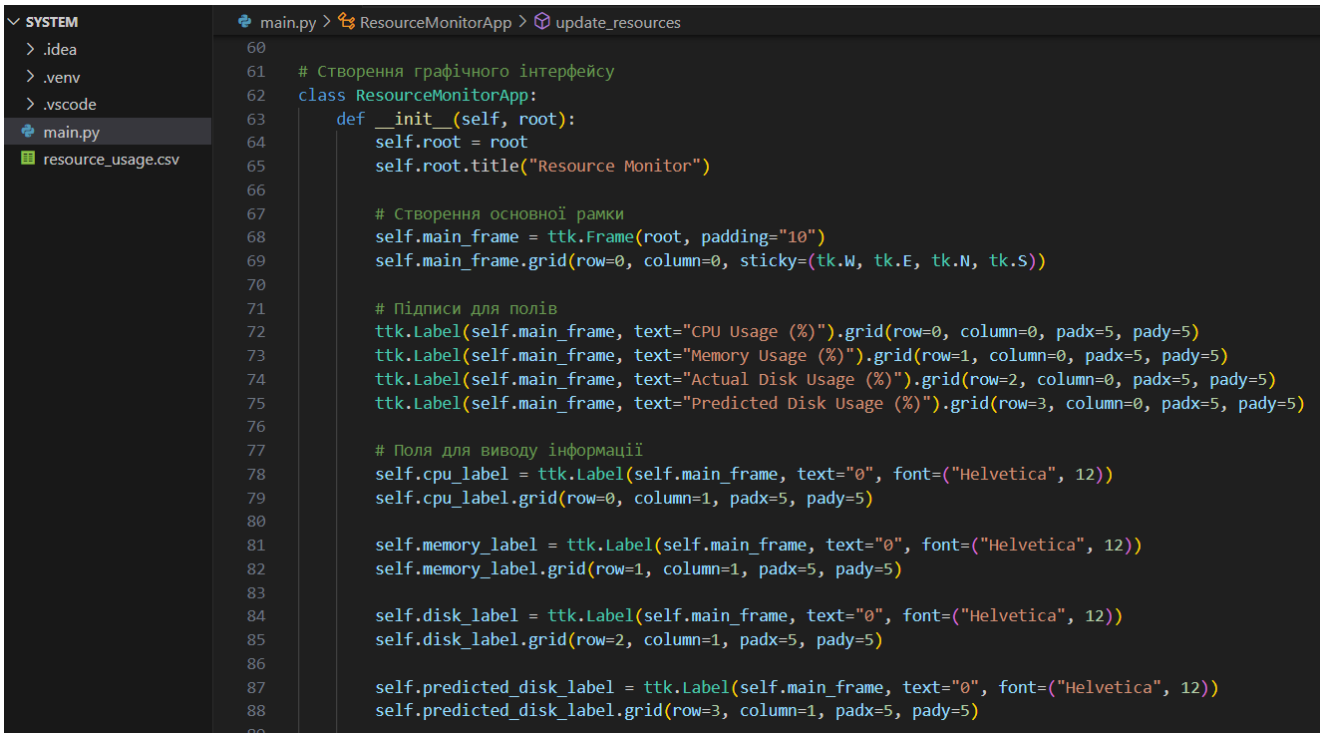

Рисунок 4.7 – Фрагмент коду відповідального за графічний інтерфейс

Графічний інтерфейс інтерактивно оновлюється кожні 5 секунд, показуючи актуальні дані про використання ресурсів, що зручно для моніторингу в реальному часі.

На рисунку 4.8 зображений фрагмент коду, що періодично оновлює інтерфейс користувача.

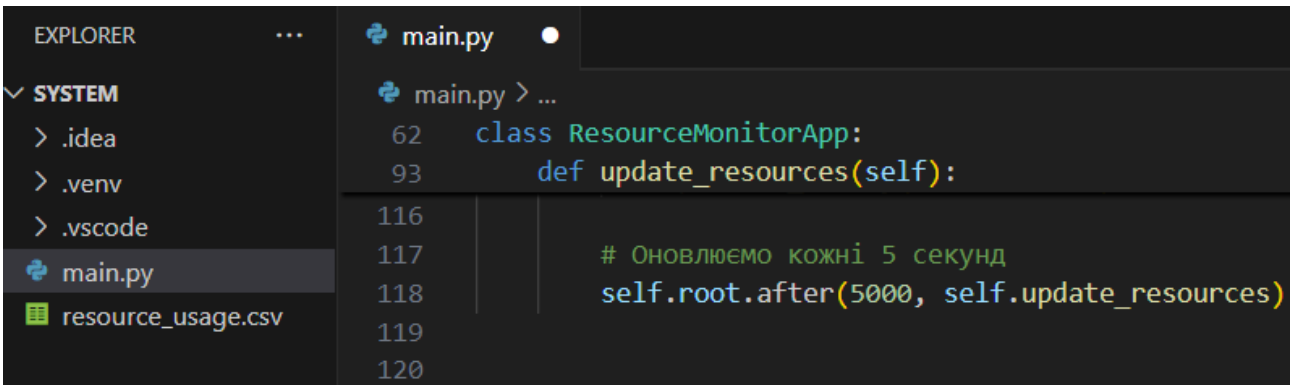

Рисунок 4.8 – Фрагмент коду, що періодично оновлює інтерфейс користувача

Однією з ключових функцій програми є метод update_resources, який забезпечує оновлення даних про використання ресурсів і прогнозування показників у реальному часі.

На рисунку 4.9 зображений код методу оновлення даних про використання ресурсів і прогнозування показників у реальному часі.

```
main.py
main.py > ResourceMonitorApp > update_resources
62  class ResourceMonitorApp:
93      def update_resources(self):
94          cpu = psutil.cpu_percent()
95          memory = psutil.virtual_memory().percent
96          disk = psutil.disk_usage('/').percent
97
98          # Створюємо DataFrame з іменами колонок
99          features = pd.DataFrame([[cpu, memory]], columns=['cpu', 'memory'])
100
101         # Масштабуємо дані
102         features_scaled = scaler.transform(features)
103
104         # Передбачаємо використання диску
105         predicted_disk_usage = model.predict(features_scaled)[0]
106
107         # Оновлюємо поля з інформацією
108         self.cpu_label.config(text=f"{cpu}%")
109         self.memory_label.config(text=f"{memory}%")
110         self.disk_label.config(text=f"{disk}%")
111         self.predicted_disk_label.config(text=f"{predicted_disk_usage:.2f}%")
112
```

Рисунок 4.9 – Код методу оновлення даних про використання ресурсів і прогнозування показників у реальному часі

Цей метод збирає актуальні показники системи, масштабує дані, здійснює прогнозування використання диску та оновлює відповідні поля в інтерфейсі

Такий підхід до управління ресурсами дозволяє автоматизувати моніторинг і оптимізацію, знижуючи ймовірність перевантаження та забезпечуючи ефективне використання системних ресурсів.

4.2.3 Навчання моделі машинного навчання

Моделі ML можна навчити використовувати виробничі процеси кількома способами. Здатність моделей ML обробляти великі обсяги даних може допомогти виробникам виявити аномалії та перевірити кореляції під час пошуку шаблонів у каналі даних. Він може забезпечити виробників можливостями прогнозованого технічного обслуговування та мінімізувати заплановані та незаплановані простої.

На рисунку 4.10 зображений перший результат роботи.

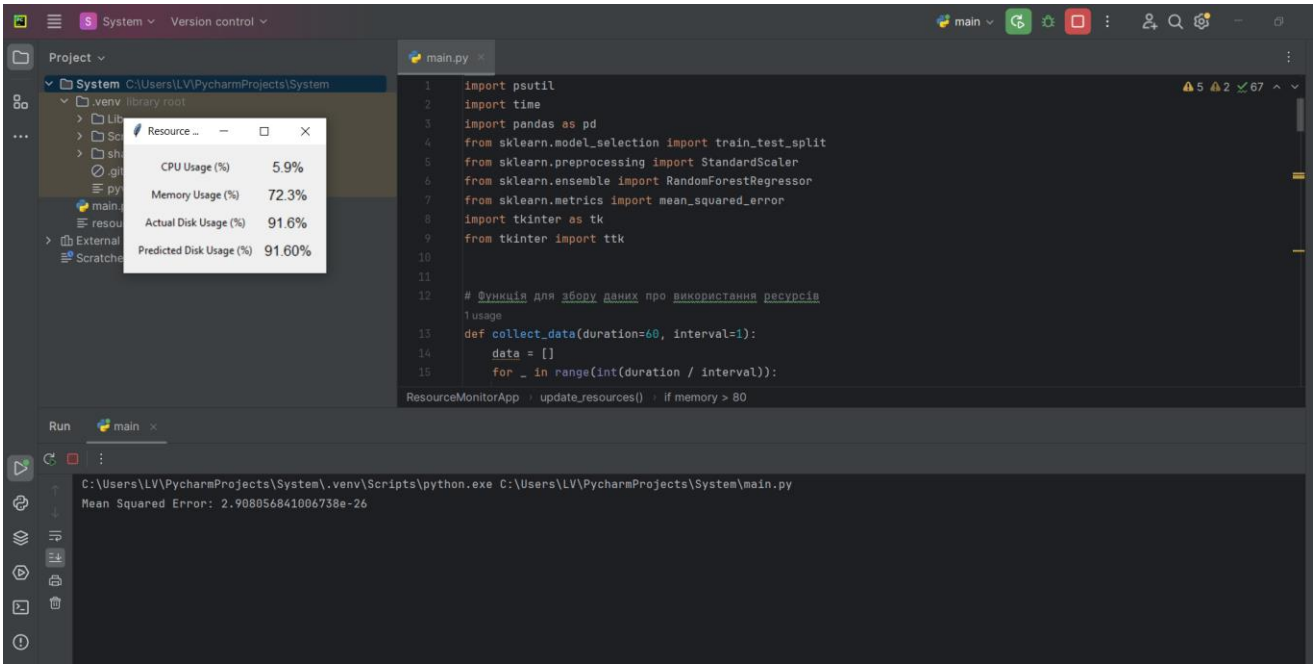

Рисунок 4.10 – Результат – 1

Навчальна модель — це набір даних, який використовується для навчання алгоритму ML. Він складається з вибірки вихідних даних і відповідних наборів

вхідних даних, які впливають на вихід. Навчальна модель використовується для проходження вхідних даних через алгоритм, щоб співвіднести оброблений вихід із зразком. Результат цієї кореляції використовується для модифікації моделі.

Цей ітеративний процес називається «припасуванням моделі». Точність навчального набору даних або набору даних перевірки має вирішальне значення для точності моделі.

Навчання моделі машинною мовою – це процес подачі даних в алгоритм ML, щоб допомогти визначити та вивчити правильні значення для всіх задіяних атрибутів. Існує кілька типів моделей машинного навчання, з яких найпоширенішими є контрольоване та неконтрольоване навчання.

Контрольоване навчання можливе, коли навчальні дані містять як вхідні, так і вихідні значення. Кожен набір даних, який має вхідні та очікувані вихідні дані, називається контрольним сигналом. Навчання виконується на основі відхилення обробленого результату від задокументованого результату, коли вхідні дані вводяться в модель.

На рисунку 4.11 зображений другий результат роботи.

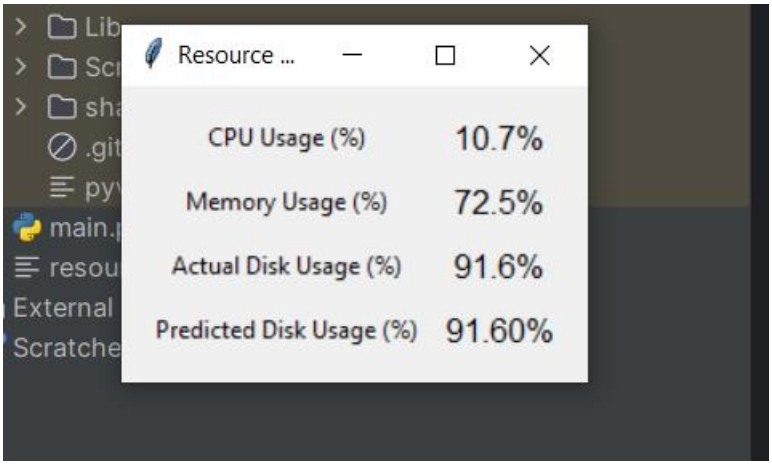

Рисунок 4.11 – Результат – 2

Навчання без нагляду передбачає визначення закономірностей у даних. Потім додаткові дані використовуються для підгонки шаблонів або кластерів. Це також ітераційний процес, який покращує точність на основі кореляції з

очікуваними шаблонами або кластерами. У цьому методі немає еталонного вихідного набору даних.

Після визначення постановки проблеми необхідно дослідити та зібрати дані, які можна використовувати для живлення машини. Це важливий етап у процесі створення моделі ML, оскільки кількість і якість використаних даних вирішуватимуть, наскільки ефективною буде модель. Дані можна зібрати з уже існуючих баз даних або створити з нуля

Під час підготовки даних, дані профілюються, форматуються та структуруються відповідно до потреб, щоб вони були готові для навчання моделі. Це етап, на якому вибираються відповідні характеристики та атрибути даних. Цей етап, ймовірно, матиме прямий вплив на час виконання та результати. Це також на етапі, коли дані класифікуються на дві групи – одна для навчання моделі ML, а інша – для оцінки моделі. На цьому етапі також проводиться попередня обробка даних шляхом нормалізації, усунення дублікатів і виправлення помилок.

На рисунку 4.12 зображений третій результат роботи.

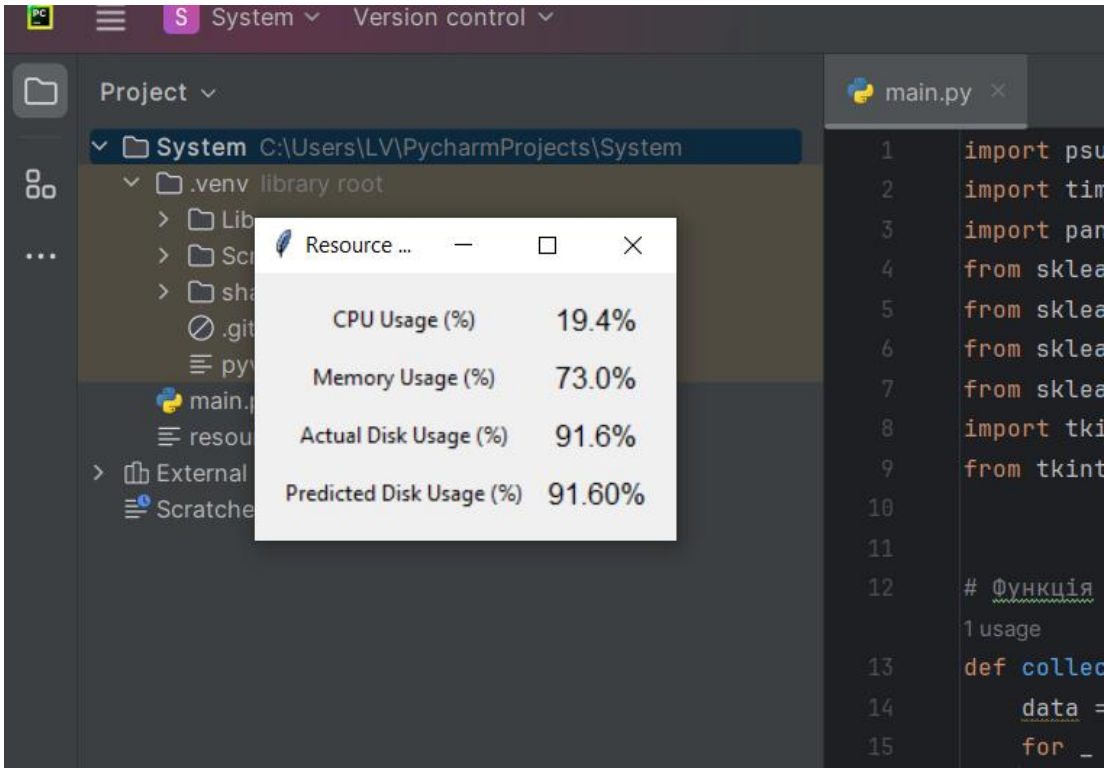

Рисунок 4.12 – Результат – 3

Вибір і призначення моделі або протоколу має здійснюватися відповідно до мети, яку прагне досягти модель ML. Існує кілька моделей, з яких можна вибрати, як-от лінійна регресія, k-середні та баєсова. Вибір моделей багато в чому залежить від типу даних, які використовуються. Наприклад, згорткові нейронні мережі обробки зображень були б ідеальним вибором, а k-середні найкраще підійшли б для сегментації.

Для побудови моделі прогнозування використовується метод RandomForestRegressor, який є інструментом для створення ансамблевої моделі на основі методу випадкових лісів[49]. Цей підхід поєднує кілька дерев рішень, де кожне дерево навчається на випадковій підмножині даних і атрибутів. Результатом моделі є середнє значення прогнозів усіх дерев, що дозволяє зменшити ймовірність переобчислення та покращити точність передбачень.

Цей алгоритм добре підходить для задач регресії, які потребують врахування великої кількості змінних і нелінійних залежностей. У системі він використовується для прогнозування навантаження на ресурси, зокрема CPU, RAM і дисковий простір, що дозволяє здійснювати масштабування на основі аналітичних даних.

Наступним йде етап навчання моделі машини або «Навчання моделі», на якому алгоритм ML навчається шляхом подачі наборів даних. Це етап, на якому відбувається навчання. Послідовне навчання може значно покращити швидкість прогнозування моделі ML. Ваги моделі повинні бути ініціалізовані випадковим чином. Таким чином алгоритм навчиться відповідно коригувати ваги.

Модель машини потрібно перевірити на «набір даних перевірки». Це допомагає оцінити точність моделі. Визначення показників успіху на основі того, чого має досягнути модель, має вирішальне значення для обґрунтування кореляції.

Вибір правильного параметра, який буде змінено для впливу на модель ML, є ключовим для досягнення точної кореляції. Набір параметрів, які вибираються на основі їх впливу на архітектуру моделі, називають гіперпараметрами. Процес ідентифікації гіперпараметрів шляхом налаштування моделі називається налаштуванням параметрів. Параметри для кореляції повинні бути чітко визначені

таким чином, щоб точка зменшення віддачі для валідації була максимально близькою до 100% точності.

4.2.4 Алгоритм роботи ПЗ на основі машинного навчання

Розглянемо алгоритм роботи програми. Програма використовує бібліотеки: psutil для моніторингу системних ресурсів, time для таймера, pandas для роботи з даними, sklearn для побудови моделі машинного навчання, tkinter для створення графічного інтерфейсу.

collect_data(duration=60, interval=1):

Виконується збір даних про завантаження CPU, використання оперативної пам'яті та диску з інтервалом в 1 секунду протягом 60 секунд.

Дані збираються у форматі словника і зберігаються у DataFrame для подальшої обробки та збереження у файл CSV.

optimize_memory(threshold=80):

Перевіряє рівень використання пам'яті, і якщо він перевищує 80%, намагається завершити процеси, які використовують більше 1% пам'яті, щоб звільнити ресурси.

Зібрані дані зберігаються у файл resource_usage.csv для подальшого аналізу.

Вибираються колонки cpu та memory як предиктори (X), а disk — як цільова змінна (y).

Дані діляться на тренувальний та тестовий набори (80% для навчання, 20% для тестування).

Виконується масштабування даних за допомогою StandardScaler для покращення роботи моделі.

Модель RandomForestRegressor була навчена на тренувальних даних для прогнозування використання дискового простору на основі параметрів системи, таких як завантаженість процесора та використання оперативної пам’яті. На рисунку 4.13 зображено моделі даних.

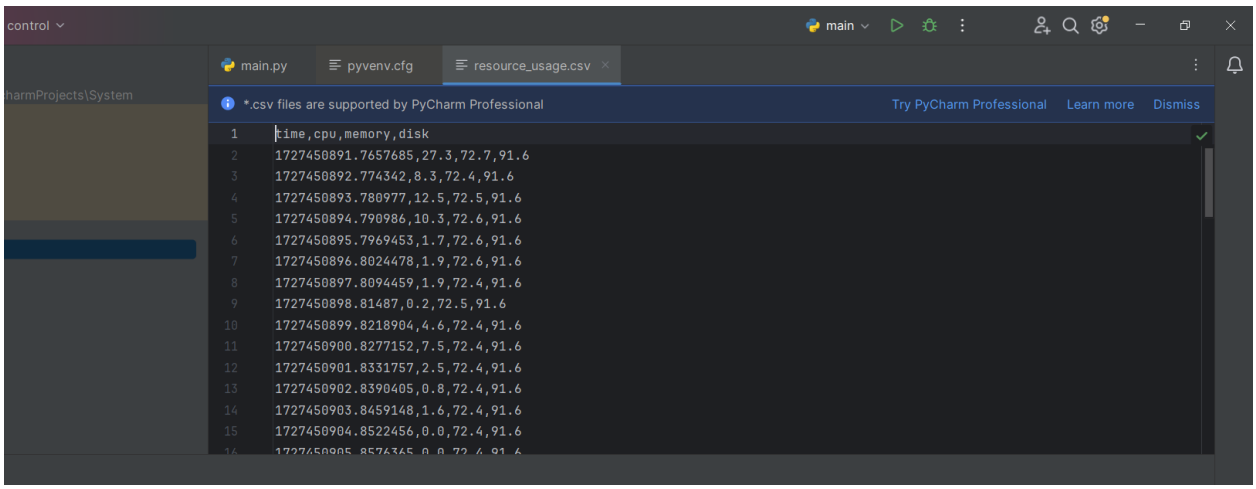

Рисунок 4.13 – Моделі даних

Обчислюється середньоквадратична похибка (MSE) для оцінки точності моделі.

За допомогою tkinter створюється простий інтерфейс для моніторингу використання ресурсів у реальному часі.

Виводиться інформація про поточне використання CPU, пам'яті, реального дискового простору та передбаченого використання диску.

Функція update_resources() кожні 5 секунд:

Оновлює значення CPU, пам'яті та диску.

Виконує передбачення використання диску на основі моделі.

Викликає функцію optimize_memory() у разі перевищення використання пам'яті понад 80%.

Створюється головне вікно Tk та запускається цикл роботи інтерфейсу mainloop().

Розглянемо сценарій роботи системи, в якому в певний момент виникає перевищення порогового значення завантаження пам’яті, а саме 80%.

Спочатку у нас йде етап ініціалізації, система працює в стандартному режимі, збирає метрики використання ресурсів. На рисунку 4.14 зображено етап ініціалізації

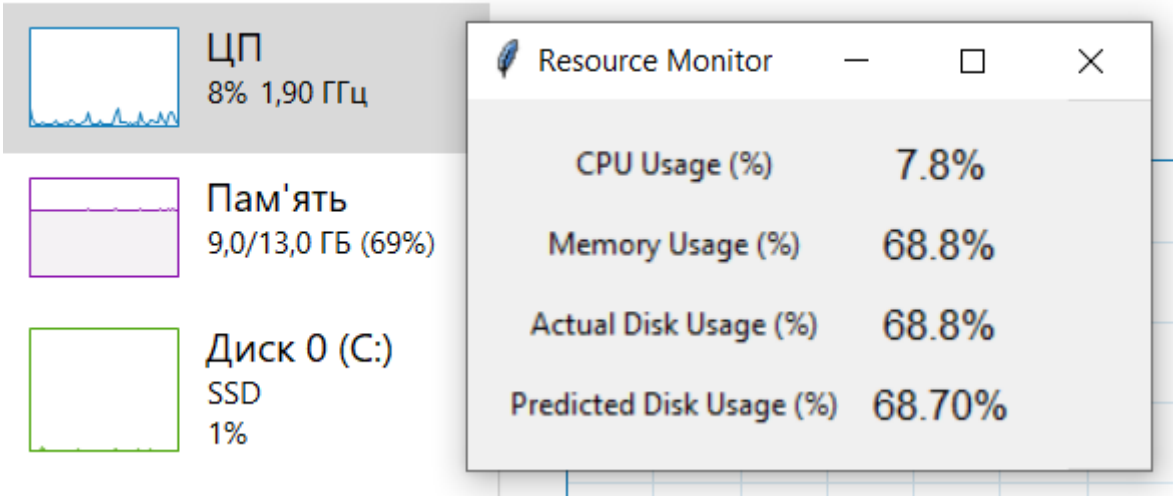

Рисунок 4.14 – Етап ініціалізації

Далі йде етап потокових оновлень даних, система продовжує працювати в стандартному режимі, але через кожні 5 секунд зчитує актуальні дані про ресурси. На кожній ітерації дані передаються на модель прогнозування. На рисунку 4.15 зображено стан системи через 5 секунд роботи в стандартному режимі.

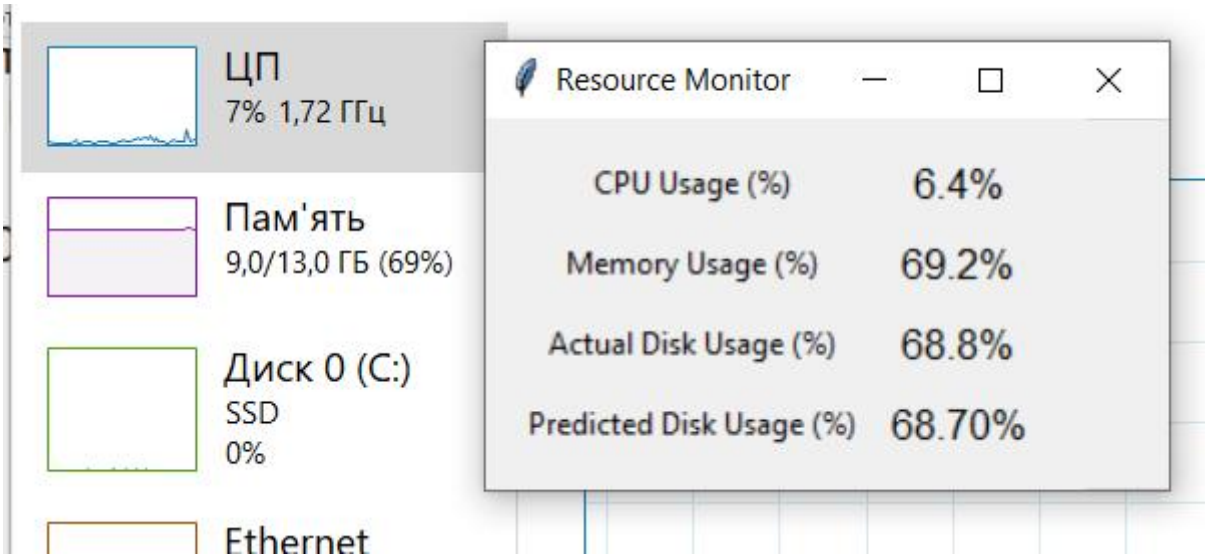

Рисунок 4.15 – Стан системи через 5 секунд роботи в стандартному режимі

Тепер розглянемо ситуацію відпрацювання системи при появі критичної ситуації. Система на певний етап ітерації почала спостерігати критичну ситуацію, а саме використання пам’яті 80%. На рисунку 4.16 зображено критичний стан системи.

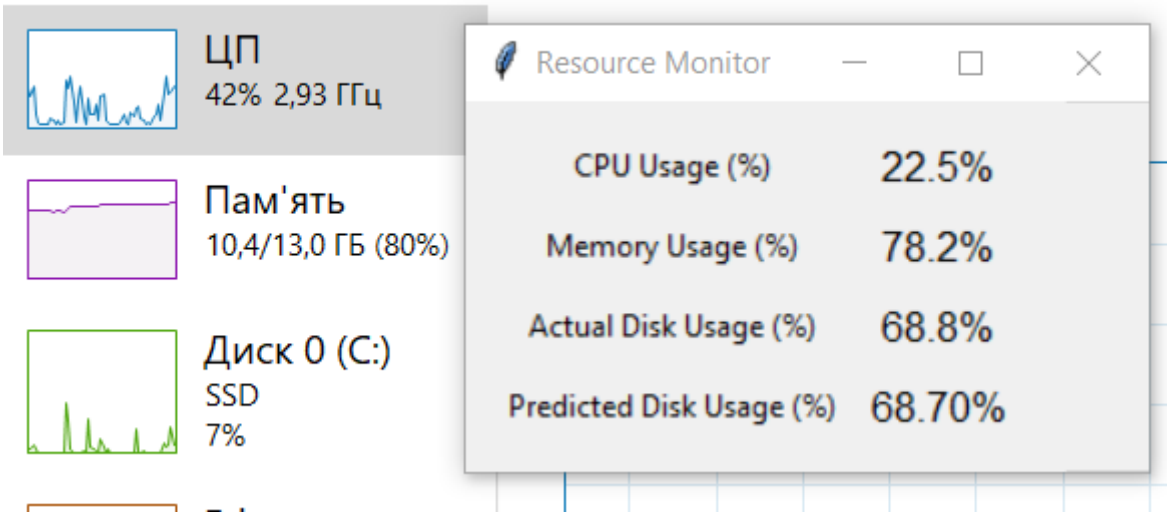

Рисунок 4.16 – Критичний стан системи

І в момент коли система бачить що використання пам’яті сягає 80%, спрацьовує функція optimize_memory(). У цей момент задіюється інтеграція з алгоритмом прогнозування, який аналізує історичні дані щодо використання пам’яті. Алгоритм здійснює оцінку поточного стану навантаження, прогнозує його подальший розвиток і формує рішення про необхідність оптимізації. На основі цього аналізу обираються процеси, які мають низький пріоритет або тимчасово неактивні, для їх завершення з метою вивільнення ресурсів. Такий підхід дає змогу звільнити пам’ять та відновити стабільну роботу системи, підтримуючи її ефективність. На рисунку 4.17 зображено вікно завершення роботи одного з процесів

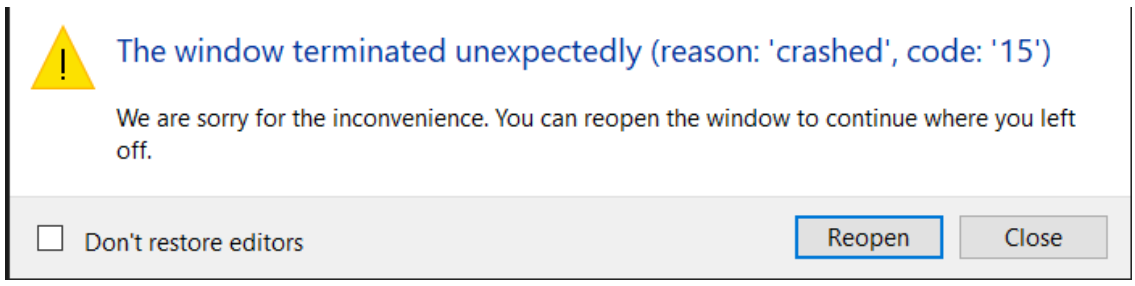

Рисунок 4.17 – Вікно завершення роботи одного з процесів

Після оптимізації система повертається до штатного режиму роботи, продовжуючи оновлювати дані кожні 5 секунд. На рисунку 4.18 зображено стан системи після відпрацювання критичної ситуації.

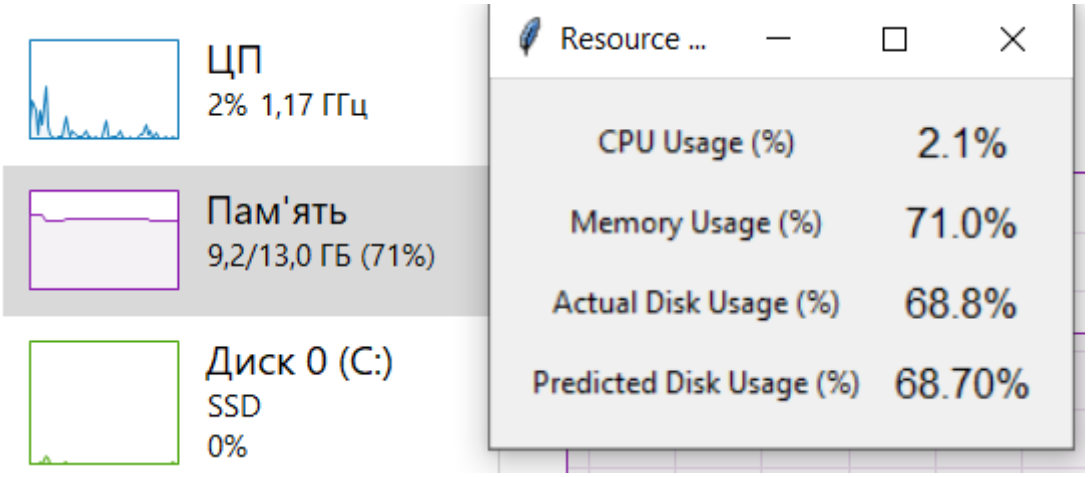

Рисунок 4.18 – Стан системи після відпрацювання критичної ситуації

Цей приклад демонструє ефективність системи управління ресурсами, яка підтримує стабільну роботу навіть у критичних ситуаціях.

4.2.5 Подальші перспективи покращення системи

Розроблена модель машинного навчання для управління ресурсами в хмарному середовищі є ефективним рішенням для моніторингу та оптимізації використання ресурсів. Проте, для забезпечення ще більшої функціональності, адаптивності та надійності, існує низка напрямків, які можуть бути реалізовані в майбутньому.

Першим значним напрямом буде інтеграція з іншими алгоритмами машинного навчання. Для системи був використаний алгоритм RandomForestRegressor, який демонструє високу точність прогнозування використання дискового простору. В подальшому можна розглядати використання інших моделей, таких як градієнтний бустинг (XGBoost, LightGBM) або глибокі нейронні мережі (Deep Neural Networks), які можуть підвищити точність за рахунок

кращої обробки складних залежностей у даних. Крім того, використання ансамблевих методів може забезпечити підвищення стабільності прогнозів.

Наступним не менш значним напрямом буде розширення функціоналу моніторингу. Наразі система аналізує CPU, оперативну пам’ять і дисковий простір. Подальший розвиток передбачає додавання метрик, які можуть бути корисними для більш комплексного управління, таких як:

– моніторинг мережевої активності (вхідний/вихідний трафік);

– стан системних журналів для виявлення потенційних помилок або аномалій.

Інтеграція механізмів прогнозування не лише для використання диску, але й для інших параметрів системи дозволить системі заздалегідь визначати потенційні перевантаження або проблеми. Наприклад, за допомогою рекурентних нейронних мереж (LSTM) можна прогнозувати завантаженість CPU на кілька годин наперед, що дасть змогу вчасно адаптуватися до змін навантаження. Тому використання прогнозової оптимізації буде гарним подальшим рішенням.

Підтримка більшого масштабування системи буде ще одним варіантом напрямку покращень системи. Адже система може бути адаптована для роботи з більшими об’ємами даних та більшою кількістю моніторингових точок. Це особливо актуально для великих організацій чи дата-центрів, де необхідно одночасно аналізувати десятки чи сотні серверів. Для цього варто інтегрувати розподілену архітектуру, наприклад, використовуючи Apache Kafka чи RabbitMQ для збору даних, і хмарні сервіси для зберігання та обробки.

Хоча функція оптимізації пам’яті вже реалізована, її можна вдосконалити. Наприклад, замість завершення процесів, які споживають найбільше пам’яті, можна запровадити динамічне зменшення їхнього пріоритету або тимчасове призупинення. Також варто додати алгоритми розподілу ресурсів на основі пріоритетів додатків, що враховують критичність завдань.

Також можна розглянути інтеграцію із системами безперервного моніторингу. Розроблена система може допускає розширенння для інтеграції з популярними інструментами моніторингу, такими як Prometheus, Nagios або

Zabbix. Це забезпечить автоматизований аналіз даних та швидке реагування на інциденти. Візуалізація метрик у реальному часі за допомогою Grafana або інших аналогів може суттєво підвищити зручність використання.

Наступним важливим кроком буде забезпечення більшої безпеки даних. Оскільки система працює з чутливими даними про використання ресурсів, варто додати механізми безпеки, такі як:

– шифрування даних під час їхнього зберігання та передачі;

– вбудований механізм автентифікації та авторизації користувачів;

– виявлення аномалій для захисту від потенційних атак або неправомірного використання ресурсів.

А для забезпечення більш надійної та стабільної роботи можна впровадити механізми автоматичного масштабування ресурсів на основі прогнозованого навантаження. Наприклад, система може запускати додаткові віртуальні машини або додаткові обчислювальні ядра в разі високого попиту.

Передостаннім перспективним напрямом буде використання найновіших архітектур машинного навчання. Розвиток архітектур, таких як трансформери, відкриває нові можливості для моделювання складних залежностей у даних. Інтеграція цих підходів може покращити прогнози використання ресурсів у різноманітних сценаріях роботи системи управління ресурсами у хмарних середовищах.

Останнім та не менш важливим напрямом для покращення системи буде адаптація для інших платформ. Система може бути адаптована для роботи в інших середовищах, таких як мобільні пристрої, IoT або розподілені обчислення. Це дозволить застосовувати її для більш широкого спектра завдань, включаючи управління ресурсами в периферійних обчисленнях (Edge Computing).

Усі ці напрямки для подальшого перспективного розвитку зможуть зробити систему більш гнучкою, масштабованою та адаптованою до потреб сучасного інформаційного середовища. Інтеграція запропонованих покращень забезпечить не лише підвищення ефективності управління ресурсами, але й збільшення її застосування в реальних умовах ринку.

Висновки до розділу 4

У розділі здійснено вибір моделі машинного навчання для оптимізації управління ресурсами в хмарних середовищах. Обрано модель RandomForestRegressor, яка забезпечує високу точність прогнозування завдяки використанню регуляризації ризиків та ефективному узагальненню даних. Реалізація алгоритму базується на застосуванні бібліотек Python, таких як psutil для моніторингу ресурсів та pandas для обробки даних, що дозволило створити ефективний механізм аналізу та управління.

Тестування алгоритму в реальних умовах показало його здатність передбачати навантаження на ресурси, уникати перевантаження та забезпечувати ефективний розподіл ресурсів. Це значно підвищує якість обслуговування (QoS) та зменшує енергоспоживання системи. Запропонована модель демонструє практичну цінність для автоматизації управління ресурсами в хмарних середовищах, адаптуючись до змінних потреб користувачів і сприяючи зниженню витрат.

Результати дослідження, розробки та розгляду подальших перспектив розвитку будуть формувати основу для подальшого впровадження автоматизованих систем управління ресурсами в IT, телекомунікаціях та електронній комерції, що забезпечить більш ефективне використання хмарних технологій.

5 СТАРТАП ПРОЕКТ

5.1 Особливості проекту

В останні роки широко використовуються хмарні обчислення. Хмарні обчислення відносяться до централізованих обчислювальних ресурсів, користувачі через доступ до централізованих ресурсів виконують обчислення, центр хмарних обчислень повертає користувачеві результати обробки програми. Хмарні обчислення призначені не тільки для індивідуальних користувачів, але і для корпоративних користувачів. Купуючи хмарний сервер, користувачам не потрібно купувати велику кількість комп'ютерів, заощаджуючи витрати на обчислення. Постачальники хмарних послуг можуть динамічно планувати обчислювальні ресурси відповідно до вимог доступу користувачів, щоб максимізувати ефективність використання обчислювальних ресурсів. Згідно зі звітом China Economic News Network, масштаби хмарних обчислень у Китаї досягли 209,1 мільярда юанів. Наразі найбільш зрілими постачальниками хмарних послуг у Китаї є Ali Cloud, Baidu Cloud, Huawei Cloud тощо.

Раціональний розподіл ресурсів відіграє вирішальну роль у хмарних обчисленнях. При розподілі ресурсів хмарних обчислень центр хмарних обчислень має обмежені хмарні ресурси, і користувачі прибувають послідовно. Кожен користувач просить центр хмарних обчислень використовувати певну кількість хмарних ресурсів у певний час. Розподіл ресурсів хмарних обчислень повинен враховувати різні потреби користувачів.

Різні користувачі мають різні вимоги до хмарних ресурсів. Це дослідження в основному розглядає призначення користувачів на сервери, які знаходяться близько, щоб покращити якість обслуговування користувачів. Однак, якщо користувачі призначаються на сервери, які знаходяться близько до них, деякі сервери можуть бути перевантажені, і час очікування буде довгим. Тому в цьому дослідженні розглядається метод розподілу хмарних ресурсів, заснований на глибокому навчанні з підкріпленням. Глибоке навчання з підкріпленням може визначити динамічний розподіл ресурсів відповідно до поточного стану системи,

таким чином максимізуючи ефективність використання хмарних ресурсів та зменшуючи час очікування користувачів.

Для побудови стартапу було проаналізовано тему та побудовано таблиці (5.1 – 5.22).

Таблиця 5.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Розробка моделі управління ресурсами з використанням машинного навчання. Основна мета – створення системи, яка здатна моніторити та оптимізувати використання ресурсів (CPU, оперативна пам'ять, дисковий простір) в реальному часі. Система використовує алгоритми машинного навчання для прогнозування навантаження на системні ресурси та динамічної оптимізації шляхом виявлення і закриття непотрібних процесів.	IT-інфраструктура: оптимізація серверів і хмарних ресурсів для зниження витрат на енергію та збільшення продуктивності. Програмне забезпечення: інтеграція в додатки для управління ресурсами на рівні ОС або віртуальних машин. Мобільні пристрої: контроль ресурсів у смартфонах для продовження часу автономної роботи.	Покращення продуктивності автоматизоване управління ресурсами забезпечує стабільну роботу системи і зменшує затримки. Зменшення витрат за допомогою оптимізації використання ресурсів знижується споживання енергії та необхідність оновлення обладнання. Прогнозування навантажень передбачення пікових навантажень допомагає уникнути збоїв у роботі систем. Автоматизація користувач не потребує ручного втручання для оптимізації роботи.

Таблиця 5.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

п/п	Техніко-економічні характеристики ідеї	Потенційні товари/концепції конкурентів	W	N	S	Мій проект	Конкурент 1	Конкурент 2	Конкурент 3
	Використання машинного навчання для прогнозування навантажень на ресурси системи	Моделі управління ресурсами без прогнозування	W: висока складність впровадження ML алгоритмів на початкових етапах	N: потреба у великому обсязі даних для навчання моделей	S: можливість точного прогнозування пікових навантажень	Прогнозування ресурсів	Автоматичний моніторинг ресурсів (ручне управління)	Просте управління ресурсами без навчання	Використання базових методів управління без ML
	Автоматична оптимізація ресурсів (CPU, RAM, диск) в реальному часі	Ручне управління ресурсами або напівавтоматичні рішення	W: ризики неправильного завершення критичних процесів	N: потреба в постійному моніторингу стану системи	S: повністю автоматизоване управління без необхідності втручання користувача	Автоматична оптимізація	Напівавтоматична оптимізація	Ручна оптимізація ресурсів	Автоматизація без точного управління
	Мінімізація енергоспоживання через оптимізацію навантаження	Стандартні методи енергозбереження або без оптимізації	W: потреба в складній інтеграції з енергозберігаючими системами	N: базові засоби економії енергії, не пов'язані з оптимізацією навантаження	S: можливість значно зменшити витрати енергії за рахунок розумного управління.	Зниження енергоспоживання	Зниження за рахунок обмеження функціоналу	Використання без фокусування на економії	Стандартні енергозберігаючі рішення без оптимізації

Таблиця 5.3 – Технологічна здійсненність ідеї проекту

п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
	Розробка моделі управління ресурсами з використанням машинного навчання	1. Алгоритми машинного навчання (наприклад, регресія, класифікація)	Наявні, але потребують налаштування для специфічних задач	Доступні через бібліотеки (scikit-learn, TensorFlow, Keras)
	Моніторинг системних ресурсів у реальному часі	2. Бібліотека psutil для збору інформації про ресурси	Наявна	Доступна для використання в Python
	Інтерфейс користувача для виводу даних	3. Tkinter або PyQt для створення GUI	Наявні	Легко доступні для реалізації на Python
	Оптимізація процесів з використанням даних	4. Системи управління процесами	Потребують інтеграції з існуючими системами	Можливо реалізувати, потребує дослідження API
	Автоматизоване навчання моделей	5. Автоматизовані фреймворки для ML (наприклад, AutoML)	Наявні, але можуть потребувати адаптації	Доступні, але з обмеженнями на використання

Таблиця 5.4 – Попередня характеристика потенційного ринку стартап-проекту

п/п	Показники стану ринку (найменування)	Характеристика
	Кількість головних гравців, од	5-10
	Загальний обсяг продаж, грн/ум.од	5 000 000 грн/рік
	Динаміка ринку (якісна оцінка)	Зростає
	Наявність обмежень для входу (вказати характер обмежень)	Високі технологічні вимоги та необхідність інвестицій у розробку
	Специфічні вимоги до стандартизації та сертифікації	ISO 9001 для управління якістю, специфікації безпеки даних
	Середня норма рентабельності в галузі (або	15-25%

п/п	Показники стану ринку (найменування)	Характеристика
	по ринку), %	

Таблиця 5.5 – Характеристика потенційних клієнтів стартап-проекту

п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
	Оптимізація використання системних ресурсів	IT-компанії, хостинг-провайдери, підприємства з великими обсягами даних	IT-компанії можуть вимагати високої точності прогнозування, в той час як малі підприємства шукають простоту використання	- Простота інтеграції з існуючими системами - Висока точність алгоритмів - Гнучкість у налаштуваннях
	Зниження витрат на енергоспоживання	Бізнеси, які активно використовують сервери, дата-центри	Великі підприємства шукають рішення для зниження витрат, а малі компанії можуть бути зацікавлені в швидкій економії	- Наявність звітності про енергоспоживання - Підтримка енергоефективних практик
	Підвищення продуктивності	Освітні установи, дослідницькі організації	Освітні установи можуть бути чутливі до бюджету, в той час як дослідницькі організації зацікавлені у передових технологіях	- Доступ до підтримки і документації - Можливість кастомізації під специфічні потреби
	Автоматизація управління ресурсами	Малі та середні підприємства, стартапи	Малі підприємства потребують простих у використанні	- Інтуїтивний інтерфейс - Наявність

п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
			рішень, стартапи можуть шукати інноваційні та нові підходи	демонстраційних версій або безкоштовних пробних періодів

Таблиця 5.6 – Фактори загроз

п/п	Фактор	Зміст загрози	Можлива реакція компанії
	Конкуренція	Зростання числа конкурентів, які пропонують подібні рішення	Проведення аналізу конкурентів, вдосконалення продукту, поліпшення обслуговування клієнтів
	Зміни в технологіях	Швидкий розвиток технологій, що може зробити продукт застарілим	Інвестування в дослідження та розробки, адаптація до нових технологій
	Регуляторні зміни	Введення нових законів або стандартів, які можуть вплинути на діяльність	Адаптація продукту до нових вимог, консультації з експертами
	Економічна нестабільність	Ризики, пов'язані з економічною ситуацією (рецесія, інфляція)	Оптимізація витрат, диверсифікація продуктового портфеля, пошук нових ринків

Таблиця 5.7 – Фактори можливостей

п/п	Фактор	Зміст можливості	Можлива реакція компанії
	Розвиток ринку машинного навчання	Зростаючий попит на рішення в галузі управління ресурсами	Розширення асортименту продуктів, активна реклама та маркетинг
	Партнерства з великими компаніями	Можливість укладення угод з великими замовниками для масштабування	Створення стратегічних альянсів, пропозиція спеціалізованих рішень
	Інновації в технологіях	Нові технології, що можуть поліпшити продукт або знизити витрати	Інвестування в нові технології, покращення існуючих рішень

п/п	Фактор	Зміст можливості	Можлива реакція компанії
	Зростання екологічної свідомості	Попит на енергоефективні та екологічні рішення	Розробка та просування продуктів, що відповідають екологічним вимогам

Таблиця 5.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Вказати тип конкуренції	Монополістична: на ринку є кілька основних гравців з диференційованими продуктами.	Розробка унікальних функцій продукту для залучення клієнтів; активне маркетингове просування.
2. За рівнем конкурентної боротьби	Національний: конкуренція на всій території країни.	Розширення каналів збуту; покращення логістики та обслуговування клієнтів.
3. За галузевою ознакою	Внутрішньогалузева: конкуренція між компаніями в одній галузі (управління ресурсами).	Аналіз конкурентів для виявлення їхніх слабких сторін; покращення продуктивності та ефективності.
4. Конкуренція за видами товарів	Товарно-видова: конкуренція на основі специфікацій продуктів.	Вдосконалення товарів; розробка нових функцій, що відрізняють продукт від конкурентів.
5. За характером конкурентних переваг	Нецінова: конкуренція на основі якості, бренду та сервісу.	Підвищення якості обслуговування; створення бренду з позитивним іміджем.
6. За інтенсивністю	Марочна: конкуренція на основі впізнаваності бренду.	Активне просування бренду, участь у виставках та конференціях для підвищення впізнаваності.

Таблиця 5.9 – Аналіз конкуренції в галузі за М. Портеро

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Навести перелік прямих конкурентів	Компанія А, Компанія Б, Компанія В	Стартапи в галузі, нові технологічні компанії	Постачальники технологій, сервісні компанії	Клієнти середнього та великого бізнесу	Програмне забезпечення з аналогічною функціональністю
Визначити бар’єри входження в ринок	Високі капіталовкладення, необхідність сертифікації	Середні бар’єри, потреба в інвестиціях та дослідженнях	Наявність альтернативних постачальників	Доступність продуктів-конкурентів	Легкий доступ до дешевших аналогів
Визначити фактори сили постачальників	Обмежена кількість постачальників, високі витрати на зміну постачальника	Низька загроза, багато альтернатив	Залежність від кількох великих постачальників	Клієнти можуть вибрати постачальника	Конкуренція між постачальниками
Визначити фактори сили споживачів	Висока чутливість до цін, можливість переходу до конкурентів	Зростання вимог до якості та обслуговування	Високі вимоги до якості та сервісу	Високі вимоги до адаптації продукту	Споживачі можуть легко знайти альтернативи
Фактори загроз з боку замінників	Наявність дешевих альтернатив	Висока загроза через нові технології	Вплив нових постачальників замінників	Зміна уподобань споживачів	Постійна загроза через нові розробки

Таблиця 5.10 – Обґрунтування факторів конкурентоспроможності

п/п	Фактор конкурентоспроможності	Обґрунтування
	Якість продукту	Висока якість забезпечує задоволення потреб клієнтів, що призводить до повторних покупок і рекомендацій.
	Інноваційність	Впровадження нових технологій та функцій робить продукт привабливішим для споживачів і відрізняє його від конкурентів.
	Цінова політика	Конкурентоспроможні ціни можуть залучити більше клієнтів, особливо в умовах жорсткої конкуренції.
	Обслуговування клієнтів	Високий рівень обслуговування покращує взаємовідносини з клієнтами і формує позитивний імідж компанії.
	Гнучкість	Можливість швидко адаптуватися до змін у попиті та ринку дозволяє компанії залишатися конкурентоспроможною.
	Розширення асортименту	Широкий асортимент продуктів задовольняє різноманітні потреби клієнтів і зменшує ризики втрати клієнтів.
	Брендова репутація	Сильний бренд сприяє довірі споживачів та може знижувати чутливість до цін.
	Дослідження та розробки	Інвестиції в R&D сприяють створенню унікальних продуктів і рішень, які важко повторити конкурентам.

Таблиця 5.11 – Порівняльний аналіз сильних та слабких сторін

п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з готовим продуктом
	Якість продукту	18	+2
	Інноваційність	15	+1
	Цінова політика	16	0
	Обслуговування клієнтів	17	+3
	Гнучкість	14	+1
	Розширення асортименту	12	-1

п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з готовим продуктом
	Брендова репутація	19	+2
	Дослідження та розробки	15	+1

5.2 SWOT-аналіз

SWOT-аналіз (сильні сторони, слабкі сторони, можливості та загрози) — це система, яка використовується для оцінки конкурентної позиції компанії та розробки стратегічного планування.

SWOT-аналіз — це техніка для оцінки ефективності, конкуренції, ризику та потенціалу бізнесу, а також його частини, наприклад продуктової лінії чи підрозділу, галузі чи іншої організації.

Використовуючи внутрішні та зовнішні дані , ця методика може скеровувати підприємства до стратегій, які, швидше за все, будуть успішними, і відволікатися від тих, у яких вони були або ймовірно будуть менш успішними. Незалежні SWOT-аналітики, інвестори чи конкуренти також можуть зорієнтувати їх щодо того, чи може компанія, лінійка продуктів чи галузь бути сильною чи слабкою та чому.

Таблиця 5.12 – SWOT-аналіз стартап-проекту

Сильні сторони	Слабкі сторони
1. Висока якість продукту	1. Обмежені фінансові ресурси
2. Інноваційні технології	2. Недостатня обізнаність про ринок
3. Досвідчена команда	3. Обмежений асортимент продуктів
4. Гнучкість у адаптації до змін	4. Відсутність брендової впізнаваності
Можливості	Загрози
1. Розширення ринку	1. Сильна конкуренція
2. Зростання попиту на технології	2. Зміни в законодавстві

Можливості	Загрози
3. Партнерство з великими компаніями	3. Економічні коливання
4. Нові технології, що з'являються на ринку	4. Зміна споживчих вподобань

Таблиця 5.13 – Альтернативи ринкового впровадження стартап-проекту

п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
	Запуск пілотного проекту в обмеженій географічній області	Висока	6 місяців
	Партнерство з великими компаніями для спільного просування	Середня	1 рік
	Участь у виставках та конференціях для підвищення обізнаності	Висока	3-6 місяців
	Проведення маркетингової кампанії через соціальні мережі	Висока	1-3 місяці
	Розробка програми лояльності для залучення клієнтів	Середня	6-12 місяців

Таблиця 5.14 – Вибір цільових груп потенційних споживачів

п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
	Малі та середні підприємства в IT-індустрії	Висока	Середній	Висока	Середня
	Стартапи, що займаються розробкою програмного забезпечення	Висока	Високий	Середня	Висока
	Корпорації, що впроваджують	Середня	Низький	Висока	Низька

п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
	нові технології				
	Навчальні заклади та освітні організації	Висока	Середній	Низька	Висока
	Фрілансери та індивідуальні підприємці	Висока	Високий	Середня	Висока

Таблиця 5.15 – Визначення базової стратегії розвитку

п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
	Запуск пілотного проекту в обмеженій географічній області	Нішевий маркетинг	Висока якість продукту, індивідуальний підхід до клієнтів	Розширення через фокус на специфічному сегменті
	Партнерство з великими компаніями	Спільні пропозиції	Спільні ресурси, доступ до нових ринків, розподіл ризиків	Розвиток через стратегічні альянси
	Проведення маркетингової кампанії через соціальні мережі	Масовий маркетинг	Висока обізнаність, залучення молодої аудиторії	Збільшення впізнаваності бренду та залучення нових клієнтів
	Розробка програми лояльності	Утримання існуючих клієнтів	Підвищення задоволеності клієнтів, зниження плинності	Поглиблення відносин з наявними клієнтами

Таблиця 5.16 – Визначення базової стратегії конкурентної поведінки

п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забиратиме існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
	Так	Шукатиме нових споживачів	Ні, буде розробляти унікальні функції, не копіюючи товар конкурентів	Інноваційна стратегія
	Ні	Забирати існуючих споживачів у конкурентів	Так, адаптує популярні функції, щоб відповідати потребам цільової аудиторії	Стратегія «переслідування»
	Так	Компанія зосередиться на обох напрямках — пошуку нових та забиранні існуючих	Ні, планує впроваджувати нові технології та рішення	Стратегія диверсифікації
	Ні	Переважно шукатиме нових споживачів	Так, копіюватиме певні елементи дизайну та функціоналу	Стратегія «контроль над ринком»

Таблиця 5.17 – Визначення стратегії позиціонування

п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
	Висока якість та надійність	Інноваційна стратегія	Унікальні функції, що відповідають потребам користувачів	Інновації, якість, надійність
	Доступна ціна	Стратегія «переслідування»	Конкурентоспроможна ціна, хороше обслуговування	Доступність, ефективність, економія
	Простота у використанні	Стратегія диверсифікації	Інтуїтивний інтерфейс, підтримка користувачів	Простота, зручність, підтримка
	Сумісність з іншими системами	Стратегія «контроль над ринком»	Висока інтеграція, партнерство з провідними постачальниками	Сумісність, партнерство, інтеграція

5.3 Розрахунок вартості проекту та його елементів

Таблиця 5.18 – Визначення ключових переваг концепції потенційного товару

п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
	Ефективне управління ресурсами	Зниження витрат та оптимізація процесів	Унікальні алгоритми, що забезпечують гнучкість і адаптивність у використанні ресурсів
	Підвищення продуктивності	Автоматизація рутинних завдань	Інтуїтивний інтерфейс та швидкість обробки даних
	Безпека даних	Високий рівень захисту та конфіденційності	Використання передових технологій шифрування та захисту інформації
	Легкість інтеграції	Сумісність з іншими платформами	Розробка модулів для легкої інтеграції з існуючими системами
	Доступність та простота використання	Інтуїтивно зрозумілий процес налаштування	Підтримка користувачів через навчальні матеріали та консультації

Таблиця 5.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
I. Товар за задумом	Базова потреба споживача, яку задовольняє товар: - Ефективне управління ресурсами - Оптимізація витрат і підвищення продуктивності. Основна функціональна вигода: зниження витрат, економія часу, покращення контролю над ресурсами.
II. Товар у реальному виконанні	Властивості/характеристики: 1. Інтуїтивний інтерфейс 2. Висока продуктивність 3. Забезпечення безпеки даних М/Нм: - Мобільний доступ - Налаштування під потреби користувача Вр/Тх /Тл/Е/Ор: - Вартість - Технічні характеристики - Екологічні характеристики - Орієнтація на потреби клієнтів

Рівні товару	Сутність та складові
	Якість: Стандарти, нормативи, параметри тестування, що підтверджують надійність і ефективність продукту. Пакування: Сучасне, екологічне, з інформативними елементами. Марка: Назва організації-розробника + назва товару, що підкреслює якість і інноваційність.
III. Товар із підкріпленням	До продажу: Пропозиції навчальних матеріалів, демонстраційні версії, консультації. Після продажу: Технічна підтримка, регулярні оновлення, програми лояльності. За рахунок чого потенційний товар буде захищено від копіювання: - Патенти на технології - Унікальний алгоритм обробки даних - Система шифрування, що забезпечує безпеку інформації - Брендова репутація та лояльність клієнтів.

Таблиця 5.20 – Визначення меж встановлення ціни

п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
	5000 грн	7000грн	Середній дохід: 10,000 грн/місяць	Верхня межа: 7500 грн Нижня межа: 4500 грн
	4500 грн	6500 грн	Високий дохід: 15,000 грн/місяць	Верхня межа: 8000 грн Нижня межа: 5000 грн
	6000 грн	7500 грн	Низький дохід: 5,000 грн/місяць	Верхня межа: 7000 грн Нижня межа: 4000 грн

Таблиця 5.21 – Формування системи збуту

п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
	Середній споживач шукає якість та ціну	Інформування про продукт, технічна підтримка, навчання	Прямий продаж через веб-сайт	Комбінована система (онлайн + офлайн)
	Корпоративні клієнти з високими вимогами до обслуговування	Індивідуальний підхід, адаптація пропозицій, післяпродажне обслуговування	Прямий і непрямий продаж	Прямий продаж з дистриб'юторами
	Малі бізнеси, що шукають доступність та швидкість	Швидка доставка, гнучкі умови оплати, простота замовлення	Непрямий продаж через реселлерів	Мультиканальна система (онлайн + реселлери)

Таблиця 5.22 – Концепція маркетингових комунікацій

п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
	Орієнтовані на дослідження та аналіз	Соціальні мережі, вебінари, професійні форуми	Інноваційність, ефективність, простота використання	Інформування про переваги продукту	Використання графіки, кейсів успішного впровадження
	Залежність від відгуків інших споживачів	Відгуки, блоги, відеоогляди	Надійність, підтримка клієнтів	Створення довіри, підкреслення позитивного досвіду	Наголос на відгуках клієнтів та експертів
	Швидкість ухвалення рішення	Email-розсилки, інформаційні бюлетені	Доступність, конкурентна ціна	Залучення уваги, створення терміновості	Використання акцій та обмежених пропозицій

Висновки до розділу 5

Проведений аналіз показав, що на ринку існує стабільний попит на рішення, що пропонуються проектом. Динаміка ринку свідчить про зростання інтересу до автоматизації управління ресурсами, що підвищує рентабельність роботи на даному сегменті. Середня рентабельність галузі відповідає очікуванням і створює умови для успішного виходу на ринок.

Потенційні групи клієнтів демонструють зацікавленість у продуктах, що забезпечують ефективне управління ресурсами. Однак на ринку присутні певні бар'єри входження, зокрема, вимоги до стандартизації та сертифікації. Конкуренція є високою, але проект має чіткі конкурентні переваги, що дозволяють йому зайняти свою нішу.

З огляду на результати проведеного аналізу, подальша імплементація проекту є доцільною. Своєчасні інвестиції в розвиток продукту та маркетингову стратегію дозволять закріпити позиції на ринку та забезпечити стабільний прибуток.

ВИСНОВКИ

В магістерській дисертації вирішено актуальну науково-практичну задачу підвищення ефективності систем управління ресурсами в хмарних середовищах на основі методів машинного навчання для оптимізації використання ресурсів.

В даній роботі було проведено всебічний аналіз підходів питань управління ресурсами у хмарних середовищах, зокрема моделей розгортання хмарних обчислень(публічних, приватних і гібридних), а також сервісів SaaS, PaaS та IaaS. В результаті було виявлено, що майже всі існуючі методи, які були розглянуті, стикаються з різноманітними проблемами динамічності та перевантаження. Тому для їх вирішення вимагається впровадження новітніх технологій для покращення таких систем.

В межах роботи було вивчено можливості застосування алгоритмів машинного навчання для архітектури систем управління ресурсами у хмарних середовищах. Виявлено переваги сучасних алгоритмів що дозволяють зменшити кількість активних серверів, знизити витрати на енергоспоживання та покращити загальну ефективність системи. В результаті було проаналізовано і виявлено алгоритми, що здатні знижувати затримки роботи подібних систем та сприяють більш ефективному управлінню ресурсами.

Також було розроблено архітектуру системи управління ресурсами, яка здатна враховувати сучасні проблеми динамічності роботи у хмарних середовищах. І було запропоноване рішення що забезпечить гнучке масштабування, достатній рівень адаптивності до ймовірних змін та економічний та ефективний розподіл ресурсів.

Було створено модель на основі машинного навчання для системи управління ресурсами, що дозволяє ефективно слідкувати за ресурсами у хмарних середовищах та ефективно балансувати навантаження. Завдяки цій моделі забезпечується висока якість обслуговування (QoS).

Результати дослідження можуть бути застосовані в різних галузях, таких як IT, телекомунікації та електронна комерція.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sharma, S., Pariha, D.: A review on resource allocation in cloud computing. Int. J. Adv. Res. Ideas Innov. Technol. 1, 1–7 (2014)

2. Ngenzi, A., Nair, S.R.: Dynamic resource management in Cloud datacenters for Server consolidation. arXiv preprint arXiv:1505.00577 (2015)

3. Magurawalage, C.S., Yang, K., Patrik, R., Georgiades, M., Wang, K.: A resource management protocol for mobile cloud using auto-scaling. arXiv preprint arXiv:1701.00384 (2017)

4. Chen, X., Li, W., Lu, S., Zhou, Z., Fu, X.: Efficient resource allocation for on-demand mobile-edge cloud computing. IEEE Trans. Veh. Technol. 67(9), 8769–8780 (2018)

5. Nguyen, T., Bao, L.L.: Joint computation offloading and resource allocation in cloud based wireless HetNets. In: GLOBECOM 2017 IEEE Global Communications Conference. IEEE (2017)

6. Nguyen, T.T., Long, B.L.: Joint computation offloading and resource allocation in cloud based wireless HetNets. arXiv preprint arXiv:1812.04711 (2018)

7. Mustafa S., Nazir B., Hayat A., Khan A. ur R., Madani S. A. Resource Management in Cloud Computing: Taxonomy, Prospects and Challenges // Computers and Electrical Engineering. 2015. Vol. 47. P. 71–85. URL: https://www.researchgate.net/figure/Taxonomy-of-RM-Techniques_fig1_280918541 (дата звернення: 19.12.2024)

8. Grewal RK, Pateriya PK. A rule-based approach for effective resource provisioning in hybrid cloud environment. Adv Intell Syst Comput 2013;203:41–57.

9. Choudhury K, Dutta D, Sasmal K. RM in a hybrid cloud infrastructure. Int J Comput Appl 2013;79(12):41–5.

10. Saraswathi, A.T., Kalaashri, Y.R., Padmavathi, S.: Dynamic resource allocation scheme in cloud computing. Procedia Comput. Sci. 47, 30–36 (2015)

11. Wang, L., et al.: Cloud computing: a perspective study. New Gener. Comput. 28(2), 137– 146 (2010)

12. Wang, L., Fu, C.: Research advances in modern cyber infrastructure. New Gener. Comput.28(2), 111–112 (2010)

13. Altmann J, Kashef MM. Cost model based service placement in federated hybrid clouds. Future Gener Comput Syst 2014;41:79–90.

14. Voorsluys, W., Broberg, J., Buyya, R.: Introduction to cloud computing. In: Cloud computing, pp. 1–41 (2011)

15. Mustafa S., Nazir B., Hayat A., Khan A. ur R., Madani S. A. Resource Management in Cloud Computing: Taxonomy, Prospects and Challenges // Computers and Electrical Engineering. 2015. Vol. 47. P. 71–85. URL: https://www.researchgate.net/figure/Distributed-hierarchical-framework_fig2_280918541 (дата звернення: 19.12.2024)

16. Ghobaei-Arani, M., Khorsand, R., Ramezanpour, M.: An autonomous resource provisioning framework for massively multiplayer online games in cloud environment. J. Netw. Comput. Appl. 142, 76–97 (2019)

17. Younge, A.J., Von, L.G., Wang, L., Lopez-Alarcon, S., Carithers, W.: Efficient resource management for cloud computing environments. In: International Conference on Green Computing, pp. 357–364. IEEE (2010)

18. Shyamala, K., Rani, T.S.: An analysis on efficient resource allocation mechanisms in cloud computing. Indian J. Sci. Technol. 8(9), 814 (2015)

19. Farahabady MRH, Lee YC, Zomaya AY. Pareto-optimal cloud bursting. IEEE Trans Parallel Distrib Syst 2014;25(10).

20. Tsakalidou V. N., Mitsou P., Papakostas G. Machine Learning for Cloud Resources Management – An Overview. 2021. URL: https://www.researchgate.net/figure/Basic-cloud-service-categories_fig1_348861169 (дата звернення: 19.12.2024)

21. Liu, N., et al.: A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning. In: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), pp. 372–382. IEEE (2017)

22. Arfeen, M.A., Pawlikowski, K., Willig, A.: A framework for resource allocation strategies in cloud computing environment. In: 2011 IEEE 35th Annual Computer Software and Applications Conference Workshops, pp. 261–266. IEEE (2011)

23. Singh, P., Talwariya, A., Kolhe, M.: Demand response management in the presence of renewable energy sources using Stackelberg game theory. In: IOP Conference Series: Materials Science and Engineering, vol. 605, 1, no. 1, p. 012004. IOP Publishing (2019)

24. Li, Z., Chu, T., Kolmanovsky, I.V., Yin, X., Yin, X.: Cloud resource allocation for cloud- based automotive applications. Mechatronics 50, 356–365 (2018)

25. Beloglazov, A., Abawajy, J., Buyya, R.: Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Fut. Gener. Comput. Syst. 28(5), 755–768 (2012)

26. 1Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Fut. Gener. Comput. Syst. 25(6), 599–616 (2009)

27. Wang, L., Kunze, M., Tao, J., von Laszewski, G.: Towards building a cloud for scientific applications. Adv. Eng. Softw. 42(9), 714–722 (2011)

28. Mohan, N., Kangasharju, J.: Placing it right!: optimizing energy, processing, and transport in Edge-Fog clouds. Ann. Telecommun. 73(7–8), 463–474 (2018)

29. Addis B, Ardagna D, Panicucci B, Squillante MS, Zhang L. A hierarchical approach for the RM of very large cloud platforms. IEEE Trans Dependable Secure Comput 2013;10(5):253–72.

30. Ardagna D, Panicucci B, Trubian M, Zhang L. Energy-aware autonomic resource allocation in multitier virtualized environments. IEEE Trans Serv Comput 2012;5(1):2–19.

31. Ardagna D, Casolari S, Colajanni M, Panicucci B. Dual time-scale distributed capacity allocation and load redirect algorithms for cloud systems. J Parallel Distrib Comput 2012;72(6):796–808.

32. Wei Y, Blake MB, Saleh I. Adaptive RM for service workflows in cloud environments. In: 2nd international workshop on workflow models, systems, services and applications in the cloud; 2013.

33. Ergu D, Kou G, Peng Y, Shi Yong, Shi Yu. The analytic hierarchy process: task scheduling and resource allocation in cloud computing environment’’. J Supercomput 2013;64(3):835–48.

34. García AG, Espert IB, García VH. SLA-driven dynamic cloud resource management. Future Gener Comput Syst 2014;31,1–11.

35. Zaman S, Grosu D. A combinatorial auction-based mechanism for dynamic VM provisioning and allocation in clouds. IEEE Trans Cloud Comput 2013;1(2),129–41.

36. Chunlin L, Layuan L. Multi-Layer RM in cloud computing. J Network Syst Manage 2013. http://dx.doi.org/10.1007/s10922-012-9261-1. (Дата звернення 16.09.2024)

37. Puthal D., Nepal S., Ranjan R., Chen J. "Threats to Networking Cloud and Edge Datacenters in the Internet of Things" // IEEE Cloud Computing. 2019. Vol. 6, No. 2. P. 62–71. URL: https://www.researchgate.net/figure/Resource-allocation-in-cloud-computing_fig1_341258906 (дата звернення: 19.12.2024)

38. Ali S, Jing Si-Yuan, Kun S. Profit-aware DVFS enabled RM of IaaS cloud. Int J Comput Sci Issues (IJCSI) 2013;10(2):237–47.

39. Goudarzi H, Pedram M. Profit-maximizing resource allocation for multi-tier cloud computing systems under service level agreements. Large scale network-centric computing systems, Wiley series on parallel and distributed computing; 2013.

40. Ban Y, Chen H, Wang Z. EALARM: an ENHANCE autonomic load-aware RM for P2P key-value stores in cloud. In: 7th IEEE international symposium service-oriented system engineering; 2013.

41. Al Sallami NM, Al Daoud A. Load balancing with neural network. Int J Adv Comput Sci Appl (IJACSA) 2013;4(10):138–45.

42. Kokilavani T, George Amalarethinam DI. Load balanced min–min algorithm for static meta-task scheduling in grid computing. Int J Comput Appl 2011;20(2).

43. Ye D, Chen J. Non-cooperative games on multidimensional resource allocation. Future Gener Comput Syst 2013;29:1345–52.

44. Jung G, Sim KM. Agent-based adaptive resource allocation on the cloud computing environment. 40th international conference on parallel processing workshops (ICPPW) 2011:345–51.

45. He L, Zou D, Zhang Z, Chen C, Jin H, Jarvis SA. Developing resource consolidation frameworks for moldable virtual machines in clouds. Future Gener Comput Syst 2014;32:68–81.

46. Puthal D., Nepal S., Ranjan R., Chen J. Challenges of Resource Allocation Assignment in Cloud // ResearchGate. 2020. URL: https://www.researchgate.net/figure/Challenges-of-resource-allocation-assignment-in-cloud_fig2_341258906 (дата звернення: 19.12.2024)

47. Sharma S. K., Simmhan Y., Peddoju S. K. Performance Analysis of Machine Learning Centered Workload Prediction Models for Cloud // IEEE Transactions on Parallel and Distributed Systems. 2023. Vol. 34, No. 4. P. 1313–1330. URL: https://www.computer.org/csdl/journal/td/2023/04/10029931/1Kmz3aPX2eY (дата звернення: 19.12.2024)

48. Malik S, Huet F, Caromel D. Latency based group discovery algorithm for network aware cloud scheduling. Future Gener Comput Syst 2014;31:28–39.

49. Scikit-learn. RandomForestRegressor. URL: https://scikit-learn.org/1.5/modules/generated/sklearn.ensemble.RandomForestRegressor.html (дата звернення: 20.12.2024)