НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

На правах рукопису
УДК 004.72

До захисту допущено
Завідувач кафедри ММСА
_____ Оксана ТИМОЩУК
«___» _____ 2024 р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-професійною програмою «Системний аналіз фінансового ринку»
за спеціальністю 124 «Системний аналіз»
на тему: «Моделі оптимального розподілу даних у гетерогенних базах даних у хмарному середовищі»

Виконав:
студент 2 курсу, групи КА-22мп
Шмідт Анатолій Євгенович _____

Керівник: професор кафедри ММСА,
д.т.н., проф. Мухін Вадим Євгенович _____

Рецензент: декан факультету
інформатики та обчислювальної техніки
КПІ ім. Ігоря Сікорського
д.т.н., Корнага Ярослав Ігорович _____

Засвідчую, що у цій магістерській дисертації
немає запозичень з праць інших авторів
без відповідних посилань
Студент (*підпис*): _____

Київ
2024

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Рівень вищої освіти — другий (магістерський)
Спеціальність — 124 «Системний аналіз»
Освітньо-професійною програмою «Системний аналіз фінансового ринку»

ЗАТВЕРДЖУЮ
Завідувач кафедри ММСА
_____ Оксана ТИМОЩУК
«___» _____ 2024 р.

ЗАВДАННЯ
на магістерську дисертацію студенту Шмідту Анатолію Євгеновичу

1. Тема дисертації: «Моделі оптимального розподілу даних у гетерогенних базах даних у хмарному середовищі», науковий керівник дисертації Мухін Вадим Євгенович, д.т.н., професор, затверджені наказом по університету від «08» грудня 2023 р. № 5200-с

2. Термін подання студентом дисертації: _____ 2024 р.

3. Об'єкт дослідження: розподіл даних у розподілених базах даних.

4. Предмет дослідження: математичні моделі оптимізації розподілу даних в розподілених базах даних.

5. Перелік завдань, які потрібно розробити:

1) провести огляд існуючої літератури щодо обраної теми;

2) сформулювати математичні моделі розподілу даних в розподілених базах даних;

3) створити програмний продукт, що реалізує математичні моделі оптимального розподілу даних;

4) продемонструвати результати роботи програмного продукту;

5) проаналізувати результати роботи програмного продукту.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу:

1) графіки результатів моделювання;

2) ілюстрації елементів інтерфейсу програми.

7. Орієнтовний перелік публікацій: Мухін В. Є., Шмідт А. Є. Моделі оптимального розподілу даних. Системні науки та інформатика : зб. доп. II науково-практ. конф. «Системні науки та інформатика», 4–8 груд. 2023 р. Київ, 2023. С. 192–197.

8. Дата видачі завдання: 01.09.2023 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Затвердження теми МД	01.09.2023-08.10.2023	виконано
2	Ознайомлення зі структурою МД згідно з Положенням про державну атестацію студентів НТУУ «КПІ ім. І. Сікорського»	09.10.2023-15.10.2023	виконано
3	Ознайомлення з ДСТУ 3008:2015 та стандартами ЄСПД	16.10.2023-22.10.2023	виконано
4	Проведення дослідження за темою МД під керівництвом керівника	23.10.2023-29.10.2023	виконано
5	Завершення роботи над першим варіантом частини МД	30.10.2023-06.11.2023	виконано
6	Проведення роботи над експериментальною частиною МД	07.11.2023-13.10.2023	виконано
7	Проведення роботи над програмним продуктом	14.11.2022-03.12.2022	виконано
8	Оформлення МД та аналіз отриманих результатів	04.12.2023-31.12.2023	виконано

Студент _____ Анатолій ШМІДТ

Науковий керівник дисертації _____ Вадим МУХІН

РЕФЕРАТ

Магістерська дисертація: 81 с., 22 рис., 20 табл., 1 додаток, 11 джерел.

РОЗПОДІЛ ДАНИХ, РОЗПОДІЛЕНА БАЗА ДАНИХ, МАТЕМАТИЧНА МОДЕЛЬ, МЕРЕЖЕВА СТРУКТУРА, ОПТИМІЗАЦІЯ

У цій роботі розглядаються математичні моделі оптимального розподілу даних у різних типах серверних мереж, створюється програмний продукт на мові програмування Python, що реалізує ці моделі.

Об’єкт дослідження – розподіл даних у розподілених базах даних.

Предмет дослідження – математичні моделі оптимізації розподілу даних в розподілених базах даних.

Мета роботи – розробка математичних моделей оптимізації розподілу даних у різноманітних структурах мережі, таких як ієрархічна, кільцева та решітчаста, розробка програмного продукту, що реалізує ці моделі.

Актуальність – у контексті зростаючого обсягу даних у сучасному світі, ефективний розподіл і управління ними стає невід'ємною складовою оптимального функціонування різноманітних інформаційних систем. Особливо важливим стає це завдання в рамках різних мережевих структур, таких як ієрархічні, кільцеві та решітчасті мережі. Дослідження спрямоване на створення інноваційних підходів, призначених вдосконалити процеси управління даними в різноманітних мережах.

Шляхи подальшого розвитку предмету дослідження – вдосконалення функціоналу, розгляд додаткових моделей, покращення інтерфейсу програми.

ABSTRACT

Master's thesis: 81 p., 22 fig., 20 tables, 1 appendix, 11 sources.

DISTRIBUTION OF DATA, DISTRIBUTED DATABASE, MATHEMATICAL MODEL, NETWORK STRUCTURE, OPTIMIZATION

This work explores mathematical models of optimal data distribution in various types of server networks and develops a software product using Python programing language that implements these models.

The object of the study is the distribution of data in distributed databases.

The subject of research is set of mathematical models for optimizing data distribution in distributed databases.

The purpose of the work is to develop mathematical models for optimizing data distribution in various network structures, such as hierarchical, ring, and mesh networks, and to develop a software product that implements these models.

Relevance lies in the fact that in the context of the growing volume of data in the modern world, efficient distribution and management of data become an integral component of the optimal functioning of various information systems. This task becomes particularly important within different network structures, such as hierarchical, ring, and mesh networks. The research is aimed at creating innovative approaches designed to improve data management processes in various networks.

Ways of further development of the research subject are improvement of functionality, consideration of additional models and enhancement of the program interface.

ЗМІСТ

ВСТУП

Основна проблема, що досліджується в цій магістерській дисертації – це оптимізація розподілу даних у розподілених базах даних. Ця проблема є актуальною у контексті збільшення обсягів та складності інформації. З кожним роком обсяги даних стрімко зростають. Підприємства та організації збирають величезні масиви інформації, і ефективне управління цими даними є важливою задачею.

Результатом цієї роботи є математичні моделі оптимального розподілу даних в таких типах серверної мережі: решітчастій, кільцевій та ієрархічній. Ці моделі реалізовані у хмарному середовищі програмування Google Colab за допомогою мови програмування Python.

Ця робота складається з п’яти розділів. У першому розділі наводиться аналіз існуючих рішень у сферах організації баз даних, гетерогенних баз даних та хмарних обчислень. Також розкривається проблема розподілу даних у гетерогенних базах даних. В кінці наводяться обмеження використання хмарних сервісів.

У другому розділі формується загальна математична задача розподілу даних. На основі цієї задачі формуються моделі оптимального розподілу даних для таких типів серверної мережі: решітчастої, кільцевої та ієрархічної.

У третьому розділі наводиться опис інструментів, що були використані для створення програмного продукту. Також ілюструється інтерфейс програми.

У четвертому розділі наводяться результати експериментів у реалізованому програмному продукті. Також проводиться аналіз цих результатів.

У п’ятому розділі розроблюється стартап-проект на базі створеної раніше програми.

РОЗДІЛ 1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ДЛЯ ОПТИМАЛЬНОГО РОЗПОДІЛУ ДАНИХ В ХМАРНОМУ СЕРЕДОВИЩІ

1.1 Характеристика моделей організації баз даних

Однією з ключових моделей є модель реляційної бази даних (RDBMS, Relational Database Management System). Основна її концепція базується на використанні реляційних таблиць для зберігання та управління даними. У цій моделі дані представлені у вигляді двовимірних таблиць, де кожен рядок відповідає запису, а кожна колонка - полю.

Головною характеристикою RDBMS є можливість встановлення взаємозв'язків між таблицями, що дозволяє ефективно організовувати та взаємодіяти з великими обсягами даних. Це досягається за допомогою унікальних ідентифікаторів (ключів), які дозволяють встановлювати зв'язки між записами різних таблиць.

Мова структурованих запитів SQL (Structured Query Language) використовується для взаємодії з базою даних. Запити дозволяють виконувати операції вставки, вибору, оновлення та видалення даних, а також виконувати складні операції з'єднання та агрегації.

Основні складові RDBMS включають таблиці з визначеними схемами, а також індекси для підвищення швидкодії запитів. Транзакційна підтримка гарантує консистентність бази даних у випадку збоїв або помилок.

Також, RDBMS визначає набір обмежень (constraints), таких як унікальність, цілісність даних, індексація, що сприяє надійності та правильності зберігання інформації.

Реляційні бази даних використовуються для різноманітних сфер, включаючи корпоративні системи, фінанси, телекомунікації та інші області, де необхідна структурована та зв'язана інформація.

Модель ключ-значення є іншим підходом до зберігання та організації даних у базах даних. Замість табличної структури, вона використовує простий,

але потужний підхід, де кожен елемент складається з ключа та відповідного йому значення. Цей ключ виступає унікальним ідентифікатором для збереженого значення.

Модель ключ-значення дозволяє використовувати швидкі та прості операції зчитування та запису, особливо в ситуаціях, коли дані можна представити у вигляді пар "ключ-значення". Використання ключів для ідентифікації даних дозволяє швидко виконувати операції пошуку та звернення до конкретних елементів. Це особливо корисно в великих обсягах даних, де ефективність та швидкодія є ключовими факторами.

Моделі ключ-значення є гнучкою та простою. Однак, в порівнянні з реляційними базами даних, ця модель може бути менш підходящою для завдань, де важливі зв'язки між різними наборами даних або потрібна складна структура.

Модель ключ-значення широко використовується у різних сучасних системах, таких як сховища даних для великих обсягів інформації, системи кешування в розподілених мережах, а також у деяких NoSQL базах даних.

Модель документно-орієнтованих баз даних є гнучкою та адаптивною системою зберігання інформації. У цій моделі дані представлені у вигляді документів, які можуть містити різні типи і структури даних, такі як текст, числа, масиви, вкладені документи і так далі.

Кожен документ у базі даних має унікальний ідентифікатор та зберігається у форматі, що легко читається і розуміється, такому як JSON (JavaScript Object Notation) або BSON (Binary JavaScript Object Notation). Це робить модель документно-орієнтованих баз даних ідеальною для сучасних додатків, де структура даних може змінюватися з часом.

Основна особливість цієї моделі - здатність до гнучкого зберігання та опрацювання даних. Документи можуть містити будь-яку кількість полів, а їхня структура може динамічно змінюватися, що робить їх ідеальними для представлення невизначених або різнорідних даних.

Одна з ключових переваг полягає у здатності виражати зв'язки між документами через вкладеність або використання посилань, що спрощує операції з'єднання та об'єднання даних.

Модель документно-орієнтованих баз даних активно використовується для зберігання та обробки даних у веб-розробці, системах керування контентом, аналітиці, де важлива гнучкість та швидкодія обробки різноманітних типів інформації.

Модель стовпцевих сховищ зберігає дані у вигляді стовпців, що забезпечує високий рівень ефективності при великих обсягах даних та операціях агрегації. Вона відрізняється від традиційних моделей тим, що дані групуються у стовпці замість рядка, що дозволяє великою мірою оптимізувати операції зчитування та запису.

Ця модель особливо ефективна при операціях агрегації та при роботі з великими обсягами даних, де необхідно отримувати доступ до певних стовпців без необхідності читання всього рядка. Іншою важливою рисою є можливість динамічно додавати нові стовпці до таблиці без перезавантаження існуючих даних.

Мова запитів у моделі стовпцевих сховищ зазвичай включає в себе операції CRUD (створення/create, читання/read, оновлення/update, видалення/delete), які можна виконувати для окремих стовпців або для групи стовпців, що дозволяє ефективно взаємодіяти з базою даних.

Такі бази даних часто використовуються в сценаріях, де важливо забезпечити швидкий доступ до конкретних частин інформації. Вони знаходять застосування в аналізі даних, системах зберігання для великих обсягів інформації, а також у вимогливих до швидкості розподілених системах.

1.2 Сутність гетерогенних баз даних

Сутність гетерогенних баз даних полягає у наявності різноманітних моделей, форматів та структур даних в межах однієї системи. Гетерогенні бази даних виникають, коли інформація зберігається та обробляється у різних джерелах, і кожне джерело може використовувати свою унікальну модель даних.

У гетерогенних базах даних можуть спостерігатися великі варіації у моделях (реляційні, документно-орієнтовані, ключ-значення, тощо), форматах (текстові файли, бінарні об'єкти, XML (EXtensible Markup Language), JSON) та мовах запитів. Це створює виклики у процесі інтеграції та обробки даних, оскільки різні частини системи можуть мати відмінні вимоги та особливості.

Однією з головних характеристик гетерогенних баз даних є необхідність використання спеціальних інструментів для інтеграції та узгодження різних джерел даних. Ці інструменти можуть включати середовища обміну даними, системи екстракції-трансформації-завантаження (ETL), або розподілені бази даних з розумінням різних моделей.

Необхідно дати визначення розподіленим базам даних та пояснити чому вони використовуються як інструмент узгодження різних джерел даних. Отже, розподілені бази даних - це системи управління базами даних, в яких дані фізично розташовані на різних комп'ютерах чи вузлах мережі. Ці системи забезпечують можливість зберігання, доступу та обробки даних через різні локації, що дозволяє оптимізувати швидкість доступу та забезпечити більшу доступність інформації.

Розподілені бази даних використовуються для гетерогенних баз даних з метою ефективної інтеграції та обробки різноманітних даних, які можуть мати відмінні моделі, формати та структури. Це дозволяє організувати інформацію з різних джерел у єдиний розподілений контекст, сприяючи забезпеченню єдиної точки доступу та управління для всієї системи.

Застосування розподілених баз даних дозволяє подолати проблеми гетерогенності шляхом створення зручного інтерфейсу для інтеграції різноманітних даних, забезпечуючи при цьому ефективність та стабільність в управлінні розподіленими та різнорідними ресурсами.

Проблеми гетерогенності можуть впливати на ефективність обробки та аналізу даних, а також ускладнювати процес управління та підтримки бази даних. У таких умовах, важливо розробляти стратегії стандартизації та нормалізації даних для спрощення інтеграції та підтримки єдиної моделі.

Однією з стратегій стандартизації даних є використання канонічної моделі даних. Ця модель визначає універсальні та стандартизовані структури даних, які можуть бути використані для представлення інформації незалежно від джерела. Вона служить відправною точкою для узгодження різноманітних форматів та моделей в єдиний, стандартний вигляд.

Ще однією стратегією є нормалізація даних. Цей процес включає в себе розбиття бази даних на менші, нормалізовані таблиці, що дозволяє уникнути дублювання інформації та підтримує цілісність даних. Нормалізація сприяє створенню однозначної та консистентної моделі, що полегшує її подальшу інтеграцію.

Інша стратегія - використання мови опису даних (DDL, Data Definition Language). Вона дозволяє визначати структуру даних та їхні взаємозв'язки в уніфікований спосіб. Застосування DDL спрощує процес визначення та розпізнавання структур даних незалежно від джерела.

Важливим аспектом є також забезпечення безпеки та захисту даних, оскільки різні джерела можуть використовувати різні механізми безпеки та автентифікації. Для вирішення цього питання в гетерогенних базах даних можуть використовуватись єдина система управління доступом та системи моніторингу та аудиту. Система управління доступом повинна враховувати різні механізми автентифікації та авторизації, що використовуються в різних джерелах даних, і забезпечувати їх універсальний контроль. Реалізація системи моніторингу та аудиту є важливим кроком для виявлення та

реагування на можливі загрози безпеки в різних джерелах даних. Аудит дозволяє відстежувати події та виявляти незвичайну активність, сприяючи вчасному реагуванню.

Таким чином, гетерогенність баз даних вимагає ретельного підходу до управління даними та інтеграції для забезпечення їх ефективного та безпечного використання.

1.3 Проблема розподілу даних в гетерогенних базах даних

Проблема розподілу даних в гетерогенних базах виникає з різноманітності структур, форматів та моделей даних між різними джерелами. Основним викликом є несумісність та розбіжність між цими різними системами, що ускладнює їх інтеграцію та узгодження.

Однією з центральних проблем є розбіжність у моделях даних. Різні джерела можуть використовувати різні структури та підходи до зберігання інформації, що ускладнює їхнє злиття та обробку в рамках єдиної системи. Ця проблема може бути вирішена шляхом впровадження мапінгу даних. Цей підхід передбачає створення проміжного рівня, який відображає взаємозв'язки та структури між різними моделями. Такий мапінг дозволяє перетворювати дані з одного формату у інший, забезпечуючи узгодженість та сумісність між різними джерелами.

Ще однією складністю є розбіжність у форматах даних. Джерела можуть використовувати різні типи файлів, стандарти обміну (наприклад, JSON, XML), або навіть використовувати бінарні формати. Це вимагає введення додаткових етапів трансформації та перетворення даних для досягнення взаємодії між різними джерелами. Ця проблема може бути вирішена за допомогою механізмів трансформації та перетворення даних. Вони включають в себе використання програмних інструментів або скриптів, які

конвертують дані з одного формату в інший. Це дозволяє стандартизувати інформацію та забезпечувати її сумісність між різними джерелами, що спрощує процес інтеграції та обробки даних.

Додатковою проблемою є розбіжність у мовах запитів. Різні системи можуть використовувати різні мови для взаємодії з базою даних. Це може вимагати від команд здійснювати переклад та адаптацію запитів для взаємодії з різними джерелами. Ця проблема може бути вирішена шляхом використання посередників. Ці інструменти дозволяють автоматично перетворювати запити з однієї мови до іншої, забезпечуючи узгодженість інтерфейсів взаємодії з різними джерелами даних. Такий підхід допомагає зменшити складність взаємодії та полегшити роботу з гетерогенними базами.

1.4 Основні концепції хмарних обчислень

Основні концепції хмарних обчислень базуються на наданні ресурсів та послуг через Інтернет, що дозволяє користувачам отримувати доступ до обчислювальної потужності, зберігання даних, мережевих ресурсів та інших послуг без необхідності володіти та управляти фізичним обладнанням.

Віртуалізація ресурсів виступає однією з ключових концепцій хмарних обчислень. Вона дозволяє ефективно використовувати фізичні ресурси та створювати віртуальні області, що полегшує розподіл обчислювальних та інших ресурсів між користувачами.

Віртуалізація серверів дозволяє запускати кілька віртуальних машин на одному фізичному сервері. Це сприяє більш ефективному використанню обчислювальної потужності та зменшенню необхідності великої кількості фізичних серверів.

У віртуалізації мережі, віртуальні мережеві ресурси можуть бути налаштовані та використовувані незалежно один від одного. Це спрощує

управління мережевою інфраструктурою та забезпечує гнучкість у розгортанні та конфігурації мережі для конкретних завдань.

Віртуалізація сховища даних дозволяє створювати віртуальні об'єкти для зберігання та керування даними, надаючи можливість легко масштабувати та розподіляти обсяги сховища відповідно до потреб користувачів.

Інша концепція - самообслуговування через API (Application Programming Interface). Вона надає можливість користувачам отримувати доступ до ресурсів та послуг через стандартні програмні інтерфейси, що дозволяє автоматизувати та полегшити управління інфраструктурою. API також служить інструментом для інтеграції хмарних рішень з існуючими системами та застосунками, дозволяючи передавати дані та керувати процесами в хмарному середовищі. Це полегшує використання хмарних сервісів як частини більших інформаційних систем.

Важливими характеристиками хмарних обчислень є масштабованість та еластичність. Вони дозволяють адаптувати обсяги ресурсів до потреб користувача. Це означає, що можливо збільшувати чи зменшувати кількість обчислювальних ресурсів в залежності від завдань та обсягу роботи.

Масштабованість та еластичність в хмарних обчисленнях досягаються завдяки можливості миттєво адаптувати обсяг обчислювальних ресурсів до змінних потреб користувача. Це забезпечується за допомогою гнучких механізмів автоматичного масштабування, які дозволяють збільшувати чи зменшувати кількість доступних ресурсів в залежності від завдань та обсягу роботи.

Ключові елементи цього підходу включають автоматичне виявлення навантаження та динамічне налаштування ресурсів для забезпечення оптимальної ефективності. Також використовуються різні стратегії розподілу ресурсів між різними користувачами та завданнями.

Розділення ресурсів та мультиарендність є також ключовими концепціями. Це дозволяє кільком користувачам чи організаціям використовувати та управляти ресурсами хмари незалежно один від одного.

Ці концепції можливі завдяки технічним та організаційним механізмам. Технічно це реалізується через використання віртуалізації. Організаційно розділення ресурсів та мультиарендність підтримуються системами управління та адміністрування хмарних обчислень. Ці системи надають інтерфейс для керування доступом до ресурсів, управління обліковими записами, та встановлення політик безпеки.

У підсумку, концепції хмарних обчислень, такі як віртуалізація, самообслуговування через API, масштабованість та еластичність, розділення ресурсів та мультиарендність, тісно взаємодіють, створюючи гнучку та ефективну інфраструктуру. Використання технічних засобів, таких як віртуалізація, дозволяє створювати ізольовані віртуальні середовища, тоді як системи управління дозволяють організаційно забезпечити ефективне розділення та управління доступом до ресурсів для різних арендарів. Цей синергетичний підхід робить хмарні обчислення потужним та динамічним інструментом для великої різноманітності користувачів та сценаріїв використання.

1.5 Взаємодія гетерогенних баз даних та хмарних обчислень

Взаємодія між гетерогенними базами даних та хмарними обчисленнями стає ключовою для оптимізації розподілу ресурсів в сучасному інформаційному середовищі. Хмарні обчислення виступають в ролі централізованої інфраструктури, що забезпечує доступ до обчислювальної потужності та зберігання даних за допомогою мережі, незалежно від локалізації та конфігурації бази даних.

Однією з ключових переваг взаємодії є забезпечення гнучкості та масштабованості. Хмарні обчислення дозволяють розширювати

обчислювальні ресурси та зберігання даних відповідно до потреб користувача, забезпечуючи при цьому оптимальну ефективність та економію витрат.

Ще один аспект - стандартизація та уніфікація даних. Хмарні сервіси часто використовують стандартизовані протоколи та формати для обміну даними, що спрощує взаємодію між різними гетерогенними базами. Це полегшує інтеграцію та обробку інформації з різних джерел.

Стандартизація визначає уніфіковані правила та норми для представлення, обробки та обміну даними. Це може включати в себе використання загальноприйнятих форматів даних, таких як JSON чи XML, або прийняття стандартів обміну інформацією, які регулюють структуру та семантику даних.

Уніфікація даних спрощує процес взаємодії між гетерогенними джерелами, встановлюючи спільний формат представлення інформації. Це дозволяє стандартизувати схеми баз даних, полегшуючи їхню інтеграцію та роботу як єдиного цілого.

Крім того, підвищується доступність та відмовостійкість. Використання різних центрів обробки даних та резервування інфраструктури дозволяє забезпечити високий рівень доступності сервісів. У випадку відмови в одному центрі, навантаження може бути автоматично перенаправлене на інший, що забезпечує неперервну роботу системи. Застосування резервного копіювання даних забезпечує відновлення інформації у випадку непередбачених подій чи аварій.

Також суттєвою є аналітика на основі штучного інтелекту (ШІ). Використання хмарних обчислень дозволяє ефективно впроваджувати аналітичні інструменти та алгоритми штучного інтелекту, що робить можливим отримання цінної інформації з гетерогенних джерел даних. ШІ дозволяє автоматизовано аналізувати та використовувати складні взаємодії в інформаційних потоках, що надходять з різних джерел.

Однією з ключових можливостей ШІ є розпізнавання патернів та тенденцій у великих обсягах даних. Алгоритми машинного навчання та

глибокого навчання дозволяють виявляти складні зв'язки та невидимі закономірності, що можуть залишитися непоміченими при традиційних методах аналізу.

ШІ також допомагає в прогнозуванні та оптимізації рішень. З використанням алгоритмів прогнозування та оптимізації, системи можуть аналізувати інформацію з різних джерел та надавати рекомендації для прийняття ефективних рішень.

Ще однією важливою характеристикою ШІ в аналітиці є автоматизація процесів виявлення аномалій та виявлення вразливостей в системах. Системи можуть автоматично реагувати на надзвичайні події, сповіщати про можливі загрози та приймати заходи для їх запобігання.

Узагальнюючи, взаємодія гетерогенних баз даних та хмарних обчислень створює сучасне та гнучке інформаційне середовище, що сприяє підвищенню продуктивності, стандартизації обробки даних та використанню передових аналітичних інструментів.

1.6 Обмеження використання хмарних сервісів

Хоча хмарні сервіси надають безліч переваг, їх використання також супроводжує ряд обмежень та потенційних проблем. Одним з основних обмежень є забезпечення безпеки та конфіденційності даних. Користувачі можуть відчувати нестабільність щодо того, як їхні дані зберігаються та обробляються в хмарі, особливо при великих обсягах конфіденційної інформації.

Ще однією проблемою є залежність від мережі та інтернет з'єднання. Для користувачів, що використовують хмарні сервіси, стає критично важливою доступність стабільного та швидкого інтернет з'єднання.

Відсутність з'єднання може призвести до втрати доступу до важливих ресурсів та даних.

Ще однією проблемою, з якою можуть зіткнутися користувачі, є питання щодо дотримання регуляторних вимог та стандартів. В різних країнах і галузях існують різні правила щодо зберігання та обробки даних, і не всі хмарні сервіси можуть відповідати усім цим вимогам.

До інших обмежень можна віднести необхідність великої пропускної здатності мережі для обробки великих обсягів даних, що може призвести до збільшення витрат на зв'язок для користувачів. Також, вартість хмарних послуг може бути проблемою, особливо для малих та середніх підприємств, які можуть зіткнутися з неочікувано високими витратами.

Також може бути актуальною проблема недостатнього контролю над інфраструктурою. Користувачі хмарних сервісів можуть відчувати втрату контролю над фізичними ресурсами, такими як сервери та зберігання даних, оскільки ці ресурси розміщені на інфраструктурі постачальника хмарних послуг.

Одним з потенційних обмежень використання хмарних сервісів є вартість переносу та витрати на трафік даних. Перенесення великих обсягів даних між локальними інфраструктурами та хмарними платформами може призвести до великих витрат через використання мережевих ресурсів та послуг.

Також, невизначеність щодо рівня обслуговування та доступності може бути обмеженням. Хоча багато хмарних постачальників гарантують високий рівень доступності, користувачам може бути важко контролювати або передбачити рівень обслуговування у певних ситуаціях.

Однією з проблем може бути непередбачувана вартість та модель ціноутворення. Зміна в розмірі даних чи обсягу транзакцій може призвести до неочікуваних витрат, що може бути проблематичним для користувачів.

Отже, використання хмарних сервісів вимагає уважного врахування цих обмежень та вибір стратегії, що враховує конкретні потреби та обмеження користувача.

1.7 Висновки до розділу

У цьому розділі роботи були досліджені питання, пов'язані з розподілом даних та гетерогенністю баз даних, а також їх взаємодію з хмарними технологіями. Було розглянуто основні характеристики моделей організації баз даних і визначено сутність гетерогенних баз даних. Також було висвітлено проблеми розподілу даних у гетерогенних базах даних.

Розкрито основні концепції хмарних обчислень, їх важливість для оптимізації роботи з даними. Була описана взаємодія гетерогенних баз даних та хмарних технологій. В кінці розділу розглянуто загальні переваги та обмеження використання хмарних сервісів, розкрито їхній вплив на розподіл даних в базах даних.

РОЗДІЛ 2 МАТЕМАТИЧНІ МОДЕЛІ РОЗПОДІЛУ ДАНИХ

2.1 Загальна математична задача

Математична задача полягає у знаходженні мінімальної кількості серверів в заданій мережі при якій час відповіді теж буде мінімальний. Зі зміною типу мережі змінюється середній шлях повідомлення (кількість переходів між вузлами). Для спрощення моделі припускаємо, що дані розподілені між вузлами мережі рівномірно та сервери мають однакову пропускну здатність.

Для заданої задачі доцільно сформувати таке рівняння [7]:

$$T(Ns) = \frac{N}{Ns} * Ts + L(Ns) * Tr \quad (2.1)$$

T(Ns) - це функція середнього часу відповіді від мережі в залежності від кількості серверів. Оскільки час для даної задачі не може бути нульовим чи від'ємним, множина значень функції має бути додатня. Ця функція підлягатиме мінімізації.

Ns – це кількість серверів. Це змінна, яка підлягає оптимізації. Ця зміна приймає натуральні значення більші 1, $Ns \in \mathbb{N}\backslash\{1\}$. Якщо при декількох значеннях Ns функція T(Ns) приймає мінімальне значення, то слід обрати найменшу кількість серверів Ns.

Ts – це час пошуку на одному сервері. Tr – це час переходу повідомлення між серверами. Оскільки в реальному житті ці значення перевищують 1 секунду лише в екстремальних випадках, то вважаємо, що $Ts, Tr \in (0,1)$.

L(Ns) - це функція середнього шляху повідомлення в мережі в залежності від кількості серверів. Оскільки час для даної задачі не може бути нульовим чи від'ємним, множина значень функції має бути додатня. Це

загальне рівняння, що описує загальну модель. Для кожного типу мережі необхідно визначити власну функцію середнього шляху L(Ns).

2.2 Кільцева мережа

У цьому підрозділі буде створено математичну модель кільцевої мережі серверів на основі загальної моделі, цільова функція для мінімізації описана у формулі (2.1). Цю модель буде модифіковано для задачі розподілу даних.

Для початку необхідно дати визначення кільцевій мережі. Кільцева мережа — це послідовний ланцюжок у замкнутому циклі. Дані переміщаються по кільцю в одному напрямку. Коли один вузол надсилає дані іншому, дані проходять через кожен проміжний вузол кільця, поки не досягнуть місця призначення. Проміжні вузли повторюють (ретранслюють) дані, щоб сигнал був сильним [8].

Загальний вигляд цієї мережі можна побачити на рисунку 2.1.

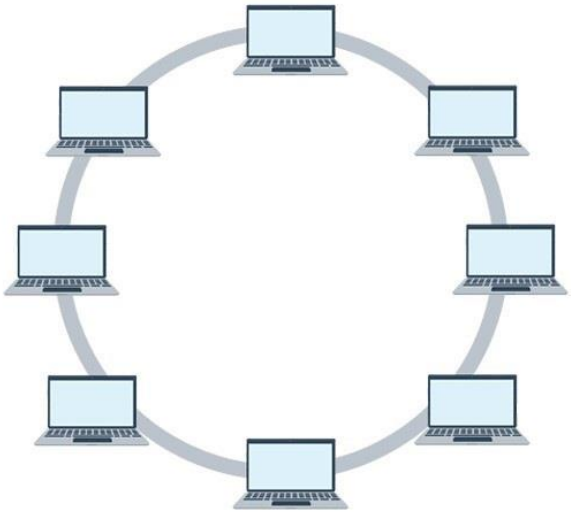


Рисунок 2.1 – Приклад кільцевої топології

2.2.1 Опис математичної моделі

Для побудови моделі необхідно знайти функцію середнього шляху L(Ns). Треба знайти L(Ns) для парної кількості серверів. Вважатимемо, що запит виконаний, коли повідомлення з потрібної інформацію повернеться до початкового сервера. Через це шлях подвоюється. Вважаємо, що повідомлення передається в обидва боки.

$$L(2) = 2 * 1 = 2$$

$$L(4) = 2\left(\frac{2}{4} + 2 * \frac{2}{4}\right) = 3$$

$$L(6) = 2\left(\frac{2}{6} + 2 * \frac{2}{6} + 3 * \frac{2}{6}\right) = 4$$

Очевидно, що для парних значень Ns справедливо наступне:

$$L(Ns) = Ns/2 + 1 \quad (2.2)$$

Необхідно знайти L(Ns) для непарної кількості серверів.

$$L(3) = 2(\frac{2}{3} + 2 * \frac{1}{3}) = \frac{8}{3}$$

$$L(5) = 2\left(\frac{2}{5} + 2 * \frac{2}{5} + 3 * \frac{1}{5}\right) = \frac{18}{5}$$

$$L(7) = 2\left(\frac{2}{7} + 2 * \frac{2}{7} + 3 * \frac{2}{7} + 4 * \frac{1}{7}\right) = \frac{32}{7}$$

Для парних значень Ns справедливо наступне:

$$L(Ns) = \frac{2(Ns-1)*(Ns-1)}{Ns} \quad (2.3)$$

Використовуючи формули (2.2) та (2.3), можна сформувати кінцеву формулу L(Ns):

$$L(Ns) = \begin{cases} Ns/2 + 1, & Ns > 1, Ns - \text{парне} \\ \frac{2(Ns-1)*(Ns-1)}{Ns}, & Ns > 1, Ns - \text{непарне} \end{cases} \quad (2.4)$$

2.2.2 Модифікація моделі для задачі розподілу даних

У цьому підрозділі буде проведено мінімізацію функції (2.1), використовуючи функцію середнього шляху повідомлення з формули (2.4). Спробуємо знайти мінімум, використовуючи опуклу оптимізацію. Для цього розширимо область значень Ns до $Ns \in [2, +\infty)$. Очевидно, ця множина значень є опуклою множиною. Перевіримо на опуклість функцію (2.1) для функції середнього шляху з формули (2.2) та окремо для формули (2.3). Щоб перевірити опуклість функції, подивимось на значення другої похідної:

$$T(Ns) = \frac{N}{Ns} * Ts + \left(\frac{Ns}{2} + 1\right) * Tr \quad (2.5)$$

$$\left(T(Ns)\right)' = \frac{Tr}{2} - \frac{Ts\ N}{Ns^2} \quad (2.6)$$

$$(T(Ns))'' = \frac{2\ Ts\ N}{Ns^3} \quad (2.7)$$

Розглянемо другу похідну (2.7). Згідно з постановкою загальної задачі, $Ts > 0$ та $N > 0$. За умовою розширення області, $Ns \geqslant 2$. Звідси випливає, що друга похідна функції (2.5) приймає лише додатні значення на заданій множині. А отже, опуклість функції (2.5) доведено.

Знайдемо мінімум функції (2.5) на заданій множині. З опуклості функції та множини, на якій вона задана, випливає, що локальний мінімум є глобальним. Тобто, щоб знайти мінімум функції на заданій множині,

достатньо знайти розв’язок рівняння (T(Ns))' = 0 (використовуючи формулу (2.6)). Знайдемо цей розв’язок пам’ятаючи, що Ns ∈ [2, +∞).

$$\left(\mathrm{T(Ns)}\right)' = \frac{\mathrm{Tr}}{2} - \frac{\mathrm{Ts\ N}}{\mathrm{Ns}^2} = 0$$

$$\mathrm{Ns}^{1*} = \sqrt{2\ \frac{\mathrm{Ts\ N}}{\mathrm{Tr}}} \tag{2.8}$$

Перевіримо на опуклість наступну функцію:

$$\mathrm{T(Ns)} = \frac{\mathrm{N}}{\mathrm{Ns}} * \mathrm{Ts}\ + \left(\frac{2(Ns-1)*(Ns-1)}{Ns}\right) * \mathrm{Tr} \tag{2.9}$$

$$\left(\mathrm{T(Ns)}\right)' = 2\ \mathrm{Tr} - \frac{(2\ \mathrm{Tr} + \mathrm{Ts}*\mathrm{N})}{\mathrm{Ns}^2} \tag{2.10}$$

$$(\mathrm{T(Ns)})'' = \frac{2*(2\ \mathrm{Tr} + \mathrm{Ts}*\mathrm{N})}{\mathrm{Ns}^3} \tag{2.11}$$

Розглянемо другу похідну (2.11). Згідно з постановкою загальної задачі, Ts > 0, Tr > 0 та N > 0. За умовою розширення області, Ns ⩾ 2. Звідси випливає, що друга похідна функції (2.9) приймає лише додатні значення на заданій множині. А отже, опуклість функції (2.9) доведено.

Знайдемо мінімум функції (2.9) на заданій множині. З опуклості функції та множини, на якій вона задана, випливає, що локальний мінімум є глобальним. Тобто, щоб знайти мінімум функції на заданій множині, достатньо знайти розв’язок рівняння (T(Ns))' = 0 (використовуючи формулу (2.10)). Знайдемо цей розв’язок пам’ятаючи, що Ns ∈ [2, +∞).

$$\left(\mathrm{T(Ns)}\right)' = 2\ \mathrm{Tr} - \frac{(2\ \mathrm{Tr} + \mathrm{Ts}*\mathrm{N})}{\mathrm{Ns}^2} = 0$$

$$\mathrm{Ns}^{2*} = \sqrt{\frac{\mathrm{Ts\ N}}{2\ \mathrm{Tr}}} \tag{2.12}$$

Для знаходження розв’язку загальної задачі, важливою умовою якої є $Ns \in \mathbb{N}\backslash\{1\}$, знайдемо значення Ns^{1*} та Ns^{2*}, використовуючи формули (2.8) та (2.12) відповідно. В загальному випадку ці значення не будуть цілими. Тому для кожного значення знайдемо два найближчі цілі числа. Для N^{s1*} позначимо ці значення як Ns^{1-} та Ns^{1+} як менше ціле число та більше ціле число відповідно. Аналогічно позначимо Ns^{2-} та Ns^{2+}. Якщо Ns^{1*} - ціле число, то $Ns^{1-} = Ns^{1+} = Ns^{1*}$. Аналогічно для Ns^{2*}. Підставимо ці значення у загальну формулу (2.1), що використовує функцію (2.4). Серед отриманих значень функції виберемо найменше, згідно з властивостями опуклих функцій, це і буде мінімум цільової функції. Якщо декілька значень Ns є точками мінімуму функції, то обираємо найменше з них.

2.3 Решітчаста мережа

У цьому підрозділі буде створено математичну модель решітчастої мережі серверів на основі загальної моделі, цільова функція для мінімізації описана у формулі (2.1). Цю модель буде модифіковано для задачі розподілу даних.

Для початку необхідно дати визначення решітчастій мережі. Плоска решітчаста топологія передбачає, що кожен її вузол сполучений із найближчим сусідом [9]. У цій роботі моделюється багаторівнева плоска решітчаста мережа, де на кожному рівні 4 вузла.

Загальний вигляд цієї мережі можна побачити на рисунку 2.2.

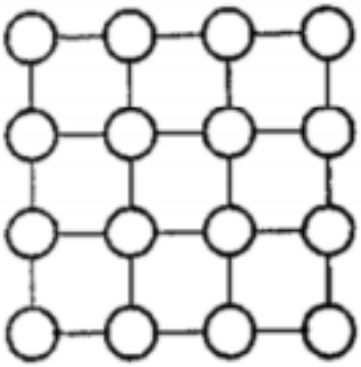


Рисунок 2.2 – Приклад решітчастої топології [9]

2.3.1 Опис математичної моделі

Для побудови моделі необхідно знайти функцію середнього шляху. Спробуємо знайти L(R), де R – кількість шарів. Вважатимемо, що запит виконаний, коли повідомлення з потрібною інформацією повернеться до початкового сервера. Через це шлях подвоюється.

Знайдемо закономірність для L(R), послідовно підставляючи R = 1, R = 2, R = 3 і так далі.

$$\mathrm{L}(1) = 2(\frac{2}{4} + 2 * \frac{2}{4}) = \frac{12}{4}$$

$$\mathrm{L}(2) = 2\left(\frac{3}{8} + 2 * \frac{3}{8} + 3 * \frac{1}{8}\right) = \frac{24}{8}$$

$$\mathrm{L}(3) = 2\left(\frac{3}{12} + 2 * \frac{4}{12} + 3 * \frac{3}{12} + 4 * \frac{1}{12}\right) = \frac{48}{12}$$

Для значень L(R) справедливо наступне:

$$\mathrm{L}(R) = (R + 2)$$

Визначимо загальну формулу для рівнів решітчастої мережі:

$$\mathrm{T}(R) = \frac{\mathrm{N}}{4\mathrm{R}} * \mathrm{Ts} + (\mathrm{R} + 2) * \mathrm{Tr} \quad (2.13)$$

2.3.2 Модифікація моделі для задачі розподілу даних

У цьому підрозділі проведемо мінімізацію функції (2.13). Спробуємо знайти мінімум, використовуючи опуклу оптимізацію. Для цього розширимо область значень Ns до $\mathrm{Ns} \in [2, +\infty)$, $\mathrm{R} \in [1, +\infty)$. Очевидно, ця множина значень є опуклою множиною. Перевіримо на опуклість функцію (2.13). Знайдемо на значення другої похідної:

$$\big(\mathrm{T}(R)\big)' = \mathrm{Tr} - \frac{\mathrm{Ts\ N}}{4\ \mathrm{R}^2} \quad (2.14)$$

$$(\mathrm{T}(\mathrm{R}))'' = \frac{\mathrm{Ts\ N}}{2\mathrm{R}^3} \quad (2.15)$$

Розглянемо другу похідну (2.15). Згідно з постановкою загальної задачі, Ts > 0, Tr > 0 та N > 0. За умовою розширення області, $\mathrm{R} \geqslant 1$. Звідси випливає, що друга похідна функції (2.13) приймає лише додатні значення на заданій множині. А отже, опуклість функції (2.13) доведено.

Знайдемо мінімум функції (2.13) на заданій множині. З опуклості функції та множини, на якій вона задана, випливає, що локальний мінімум є глобальним. Тобто, щоб знайти мінімум функції на заданій множині, достатньо знайти розв'язок рівняння (T(R))' = 0 (використовуючи формулу (2.14)). Знайдемо цей розв'язок пам'ятаючи, що $\mathrm{R} \in [1, +\infty)$.

$$\big(\mathrm{T}(\mathrm{R})\big)' = \mathrm{Tr} - \frac{\mathrm{Ts\ N}}{4\ \mathrm{R}^2} = 0$$

$$\mathrm{R}^* = \sqrt{\frac{\mathrm{Ts\ N}}{4\mathrm{Tr}}} \quad (2.16)$$

Для знаходження розв’язку загальної задачі, важливою умовою якої є $R \in \mathbb{N}$, знайдемо значення R*, використовуючи формулу (2.16). В загальному випадку це значення не буде цілим. Тому знайдемо два найближчі цілі значення. Позначимо ці значення як R^{*-} та R^{*+} як менше ціле число та більше ціле число відповідно. Якщо R* - ціле число, то $R^{*-} = R^{*+} = R^*$.. Підставимо ці значення у загальну формулу (2.13). Серед отриманих значень функції виберемо найменше, згідно з властивостями опуклих функцій, це і буде мінімум цільової функції.

2.4 Ієрархічна мережа

У цьому підрозділі буде створено математичну модель ієрархічної мережі серверів на основі загальної моделі, цільова функція для мінімізації описана у формулі (2.1). Цю модель буде модифіковано для задачі розподілу даних.

Для початку дамо визначення ієрархічній мережі або топології дерева. Топологія дерева — це мережа, в якій кожен вузол пов’язаний з іншими в ієрархії. Її також називають ієрархічною топологією, оскільки в цій топології всі елементи розташовані як гілки дерева. У топології дерева будь-які два пов’язані вузли можуть мати лише одне взаємне з’єднання, отже, між ними може бути лише одне зв’язок [10].

Загальний вигляд цієї мережі можна побачити на рисунку 2.3.

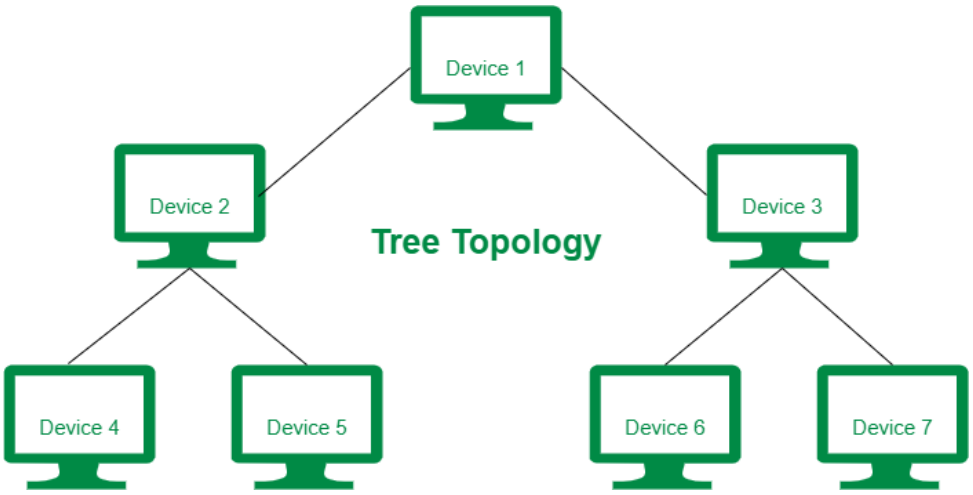


Рисунок 2.3 – Приклад ієрархічної мережі [11]

2.4.1 Опис математичної моделі

Для побудови моделі необхідно знайти функцію середнього шляху. Спробуємо знайти L(Ns), для ієрархії з повністю заповненими рівнями. Для цього необхідно ввести параметр I, що визначатиме кількість підпорядкованих вузлів для кожного вузла вищого рангу, $I \in \mathbb{N}\backslash\{1\}$. Вважатимемо, що запит виконаний, коли повідомлення з потрібною інформацією повернеться до початкового сервера. Через це шлях подвоюється.

Враховуючи вищесказане, для ієрархічної мережі доцільно сформувати наступну функцію середнього шляху:

$$L(Ns) = 2\log_I Ns$$

Визначимо загальну формулу для ієрархічної мережі:

$$T(Ns) = \frac{N}{Ns} * Ts + 2\log_I Ns * Tr \quad (2.17)$$

2.4.2 Модифікація моделі для задачі розподілу даних

У цьому підрозділі проведемо мінімізацію функції (2.17). Спробуємо знайти мінімум, використовуючи опуклу оптимізацію. Для цього розширимо область значень Ns до $Ns \in [2, +\infty)$. Очевидно, ця множина значень є опуклою множиною. Перевіримо на опуклість функцію (2.17). Знайдемо на значення другої похідної:

$$\left(T(Ns)\right)' = \frac{2\,Tr}{Ns \log I} - \frac{Ts\,N}{Ns^2} \tag{2.18}$$

$$(T(Ns))'' = \frac{\left(2\left(Ts\,N - \frac{Tr\,Ns}{\log I}\right)\right)}{Ns^3} \tag{2.19}$$

Розглянемо другу похідну (2.19). Згідно з постановкою загальної задачі, Ts > 0, Tr > 0 та N > 0. За умовою розширення області, $Ns \geqslant 2$. При умові, що $Ts \cdot N \neq (Tr \cdot Ns)/\log(I)$,·друга похідна функції (2.17) приймає лише додатні значення на заданій множині. При цій умові функція (2.17) опукла.

Знайдемо мінімум функції (2.17) на заданій множині. З опуклості функції та множини, на якій вона задана, випливає, що локальний мінімум є глобальним. Тобто, щоб знайти мінімум функції на заданій множині, достатньо знайти розв'язок рівняння (T(Ns))' = 0 (використовуючи формулу (2.18)). Знайдемо цей розв'язок пам'ятаючи, що $Ns \in [2, +\infty)$.

$$\left(T(Ns)\right)' = \frac{2\,Tr}{Ns \log I} - \frac{Ts\,N}{Ns^2} = 0$$

$$Ns^* = \frac{Ts\ N \log I}{2\,Tr} \tag{2.20}$$

Для знаходження розв'язку загальної задачі, важливою умовою якої є $Ns \in \mathbb{N}\backslash\{1\}$, знайдемо значення Ns* використовуючи формулу (2.20). В

загальному випадку це значення не буде цілим. Тому знайдемо два найближчі цілі значення. Позначимо ці значення як Ns^{*-} та Ns^{*+} як менше ціле число та більше ціле число відповідно. Якщо Ns* - ціле число, то $Ns^{*-} = Ns^{*+} = Ns^*$. Підставимо ці значення у загальну формулу (2.17). Серед отриманих значень функції виберемо найменше, згідно з властивостями опуклих функцій, це і буде мінімум цільової функції.

Щоб знайти оптимальне значення кількості рівнів ієрархічної мережі, підставимо Ns* в наступну формулу:

$$R_I^* = C(\log_I Ns^*),$$

де C – це функція округлення до більшого цілого числа.

2.5 Висновки до розділу

У даному розділі було викладено процес розробки математичних моделей різних типів мереж та модифікації цих моделей для задачі розподілу даних. Була сформована загальна модель, описана оптимізаційна задача, а також описані моделі для ієрархічної, кільцевої, решітчастої типів мереж.

Методами опуклої оптимізації доведено, що для кожної моделі можна знайти мінімум на заданій множині. Також наведено усі необхідні формули та розрахунки для розв’язання оптимізаційної задачі для трьох згаданих вище моделей.

РОЗДІЛ 3 РОЗРОБКА ПРОГРАМИ ДЛЯ МОДЕЛЮВАННЯ ОПТИМАЛЬНОГО РОЗПОДІЛУ ДАНИХ

3.1 Хмарне середовище розробки Google Colab

Для створення програми було обране веб середовище розробки Google Colab.

Google Colaboratory або Colab - це хмарний сервіс від Google Research. Це веб середовище розробки, яке дозволяє будь-якому користувачеві писати вихідний код у своєму редакторі та запускати його з браузера. Зокрема, редактор підтримує мову програмування Python і орієнтований на завдання машинного навчання, аналіз даних, навчальні проекти тощо [1].

Переваги Google Colab:

1) безкоштовний;

2) хмарний сервіс;

3) проста колаборація;

4) підтримка відеопроцесорів;

5) вбудовані бібліотеки [2].

Далі буде дана інструкція використання Google Colab.

1. Відкрити Google Colab: Щоб скористатися Google Colab, треба відкрити свій веб-браузер та перейти за посиланням «https://colab.research.google.com/» і опинитись на сторінці Google Colab. Перед цим користувачеві слід зайти в акаунт Google.

2. Створити новий блокнот: Після входу в обліковий запис можна створити новий блокнот, натиснувши кнопку "New Notebook".

3. Написати код: можна писати свій код у комірках блокноту. Також можна додавати текстові комірки для документування коду. Google Colab підтримує багато мов програмування, але основним чином використовується для Python.

4. Запустити код: Щоб запустити код, треба натиснути на кнопку "play" біля комірки з кодом, скористатися комбінацією клавіш "Shift+Enter" або перейти в меню "runtime" і вибрати відповідні опції, такі як "run all", "run before", "run after" і т. д. Google Colab виконає код і відобразить результат нижче комірки.

5. Зберегти свій блокнот: Після написання коду можна зберегти свій блокнот, натиснувши "File", а потім "Save". Блокнот буде збережено у обліковому записі Google Drive. Код буде автоматично збережено, і також буде можливість перевірити історію змін [2].

3.2 Мова програмування Python та додаткові бібліотеки

Для створення програми було обрана мова програмування Python версії 3.10.12. Для спрощення процесу розробки були використані наступні додаткові бібліотеки: numpy (версія 1.23.5), matplotlib (версія 3.7.1).

Python - це високорівнева інтерпретована скриптова мова програмування, яка була розроблена наприкінці 1980-х років Гвідо ван Россумом в Інституті математики і комп'ютерних наук Нідерландів. Перша версія була опублікована в новинній групі alt.sources у 1991 році, а версія 1.0 була випущена у 1994 році [3].

Далі будуть перечислені п’ять основних переваг мови програмування Python.

1. Легка читаність коду – при реалізації високонавантажених проєктів, над якими працюють відразу кілька програмістів, читаність коду грає вирішальне значення, тому що кожен розробник повинен розуміти, що роблять його колеги. Python диктує суворі вимоги до оформлення коду і влаштована таким чином, щоб мінімізувати кількість рядків. Тому читати її завжди зручно і легко.

2. Широка область застосування – Python використовується для розробки онлайн і мобільних додатків, машинного навчання, при створенні ігор і зручна для автоматизації математичних розрахунків. Тому її можна використовувати для вирішення практично будь-яких завдань.

3. Гарне ком’юніті і стійкість – мова Python може похвалитися великим фан-клубом розробників і підтримується гігантами IT-індустрії, такими, як Google, Facebook і Spotify.

4. Портативність – у багатьох мовах програмування, для запуску програми на різних платформах потрібно вносити зміни в код. Python працює по іншому. Написавши код один раз, розробник зможе запускати його на будь-яких пристроях без додаткових коригувань.

5. Велика база бібліотек – працюючи з Python, розробник не залежить від зовнішніх бібліотек. Мова оснащена потужною стандартною базою функцій, які можна використовувати для вирішення комерційних завдань, що істотно спрощує роботу програміста [4].

Можна виділити два основних недоліки мови Python: невисока швидкість роботи та неефективне витрачання пам’яті.

NumPy – це фундаментальний пакет для наукового обчислення в Python. Це бібліотека Python, яка надає об'єкт багатовимірного масиву, різноманітні похідні об'єкти, а також набір інструментів для швидких операцій над масивами, включаючи математичні, логічні, маніпуляції формою, сортування, вибір, введення/виведення, дискретні перетворення Фур'є, основну лінійну алгебру, основні статистичні операції, випадкові симуляції та багато іншого [5].

Matplotlib – це крос-платформена бібліотека для візуалізації даних та побудови графіків (гістограми, точкові діаграми, стовпчасті діаграми тощо) для Python та його розширення NumPy. Таким чином, вона є життєздатною відкритою альтернативою MATLAB. Розробники також можуть використовувати API (інтерфейси програмування застосунків) бібліотеки

Matplotlib для вбудовування графіків у графічні користувацькі інтерфейси програм [6].

3.3 Користувацький інтерфейс програми

Програма відкривається у браузері. Щоб відкрити, її потрібно перейти по спеціальному посиланню. При переході має відкритися сторінка Google Colab (рис. 3.1).

Рисунок 3.1 – Частина початкового вікна у браузері

Сторінка містить декілька комірок з параметрами. Перша комірка виділена червоним на рисунку 3.2. Кожна комірка це частина програми, що відповідає за певну модель. Наприклад, на рисунку 3.2 виділена комірка з моделлю кільцевої мережі.

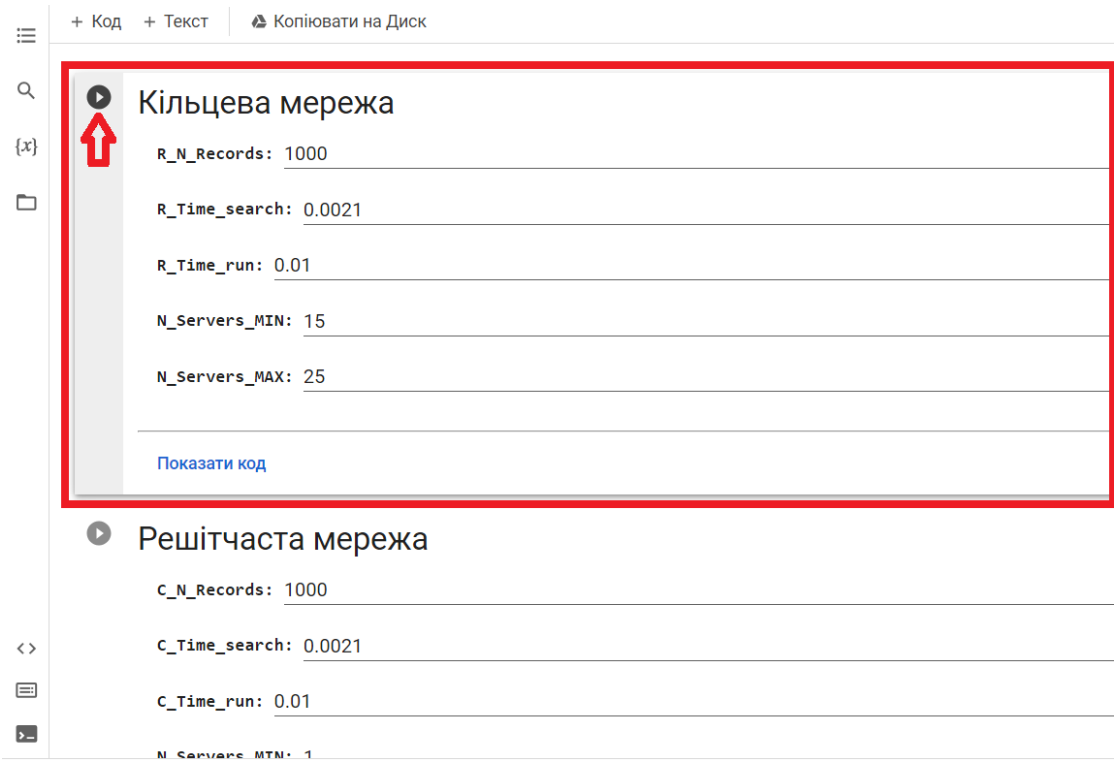


Рисунок 3.2 – Комірка «Кільцева мережа»

У комірці перечислені параметри для моделювання (рис. 3.3). Параметри задані за замовчуванням, але користувач може змінити кожен з них.

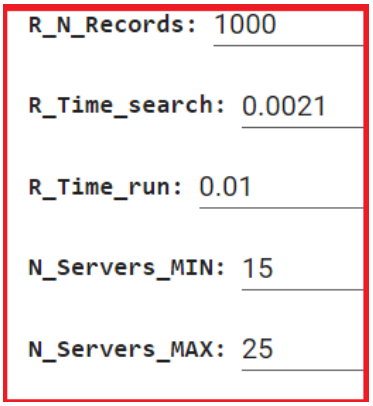


Рисунок 3.3 – Параметри комірки «Кільцева мережа»

Параметри кільцевої мережі (рис. 3.3).

1. Параметр «R_N_Records» - відповідає за загальну кількість записів для кільцевої мережі.
2. Параметр «R_Time_search» - відповідає часу пошуку для 1 запису на 1 сервері для кільцевої мережі.
3. Параметр «R_Time_run» - відповідає часу переходу 1 запиту через 1 канал зв’язку між серверами для кільцевої мережі.
4. Параметри «N_Servers_MIN» та «N_Servers_MAX» визначають мінімальну та максимальну кількість серверів на графіку відповідно.

Після встановлення параметрів для запуску моделі необхідно запустити комірку. Для цього необхідно натиснути кнопку запуску, що знаходиться ліворуч від назви комірки. Вона позначена на рисунку 3.4. Якщо користувач не зайшов в обліковий запис Google, то треба буде це зробити щоб виконати програму.

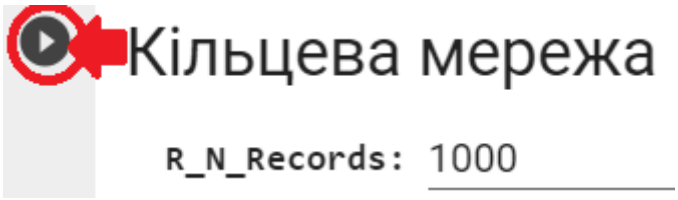
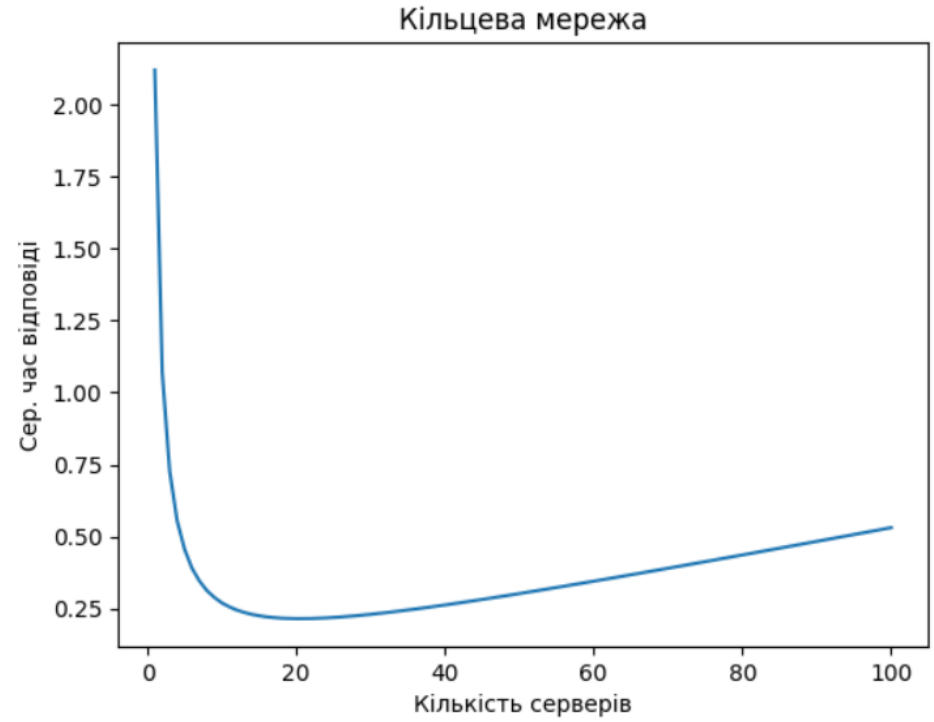


Рисунок 3.4 – Кнопка запуску комірки

Приклад результатів виконання програми для кільцевої мережі можна побачити на рисунку 3.5. Вони мають з’явитися під відповідною коміркою. Результати складаються з двох частин: зверху знаходиться графік залежності середнього часу відповіді від кількості серверів відповідної мережі (в цьому прикладі кільцевої мережі); знизу, під графіком, знаходиться оптимальна кількість серверів відповідної мережі.




```
Оптимальна кількість серверів: 20
```

Рисунок 3.5 – Приклад результатів виконання програми

Результати зберігаються, якщо відкрити сторінку повторно через якийсь час, вони залишаються. Щоб отримати результати для інших параметрів необхідно запусти комірку знову з відповідними параметрами. Результати зберігаються у акаунті Google, тому для кожного користувача вони різні.

Далі будуть коротко описані інші комірки.

Параметри решітчастої мережі (рис. 3.6).

1. Параметр «C_N_Records» - відповідає за загальну кількість записів для решітчастої мережі.
2. Параметр «C_Time_search» - відповідає часу пошуку для 1 запису на 1 сервері для решітчастої мережі.
3. Параметр «C_Time_run» - відповідає часу переходу 1 запиту через 1 канал зв’язку між серверами для решітчастої мережі.
4. Параметри «N_Servers_MIN» та «N_Servers_MAX» визначають мінімальну та максимальну кількість серверів на графіку відповідно.

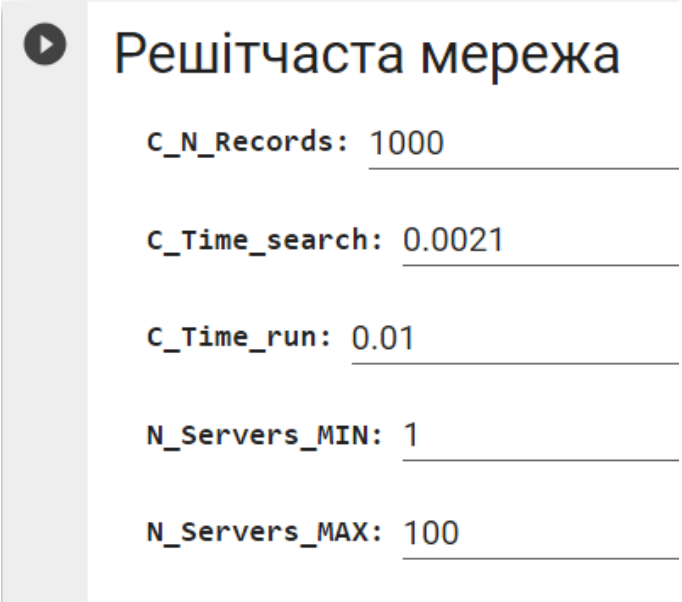


Рисунок 3.6 – Комірка «Решітчаста мережа»

Параметри ієрархічної мережі (рис. 3.7).

1. Параметр «I_N_Records» - відповідає за загальну кількість записів для ієрархічної мережі.
2. Параметр «I_Time_search» - відповідає часу пошуку для 1 запису на 1 сервері для ієрархічної мережі.
3. Параметр «I_Time_run» - відповідає часу переходу 1 запиту через 1 канал зв’язку між серверами для ієрархічної мережі.
4. Параметр «I_N_Links» - відповідає кількості зв’язків на 1 рівні ієрархічної мережі.
5. Параметри «N_Servers_MIN» та «N_Servers_MAX» визначають мінімальну та максимальну кількість серверів на графіку відповідно.

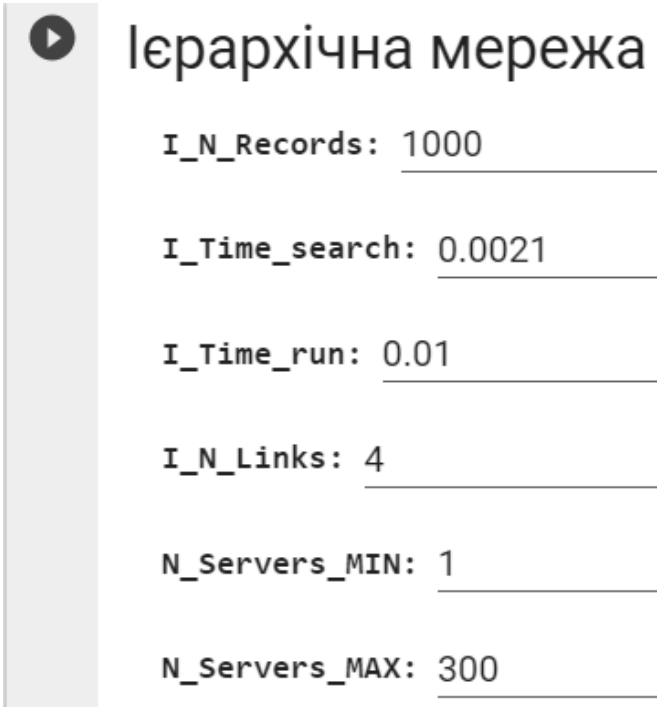


Рисунок 3.7 – Комірка «Ієрархічна мережа»

Параметри для порівняння мереж (рис. 3.8).

1. Параметр «N_Records» - відповідає за загальну кількість записів для всіх вищеперерахованих мереж.
2. Параметр «Time_search» - відповідає часу пошуку для 1 запису на 1 сервері для всіх вищеперерахованих мереж.
3. Параметр «Time_run» - відповідає часу переходу 1 запиту через 1 канал зв’язку між серверами для всіх вищеперерахованих мереж.
4. Параметр «I_N_Links» - відповідає кількості зв’язків на 1 рівні ієрархічної мережі.
5. Параметри «N_Servers_MIN» та «N_Servers_MAX» визначають мінімальну та максимальну кількість серверів на графіку відповідно.

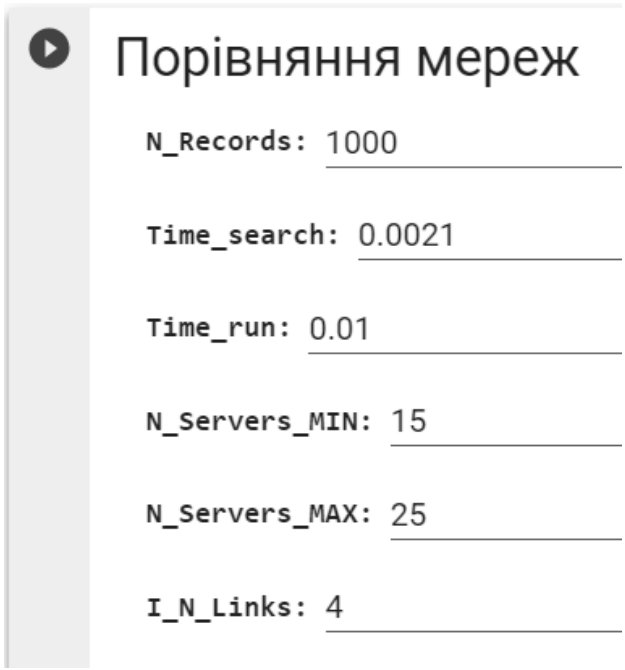
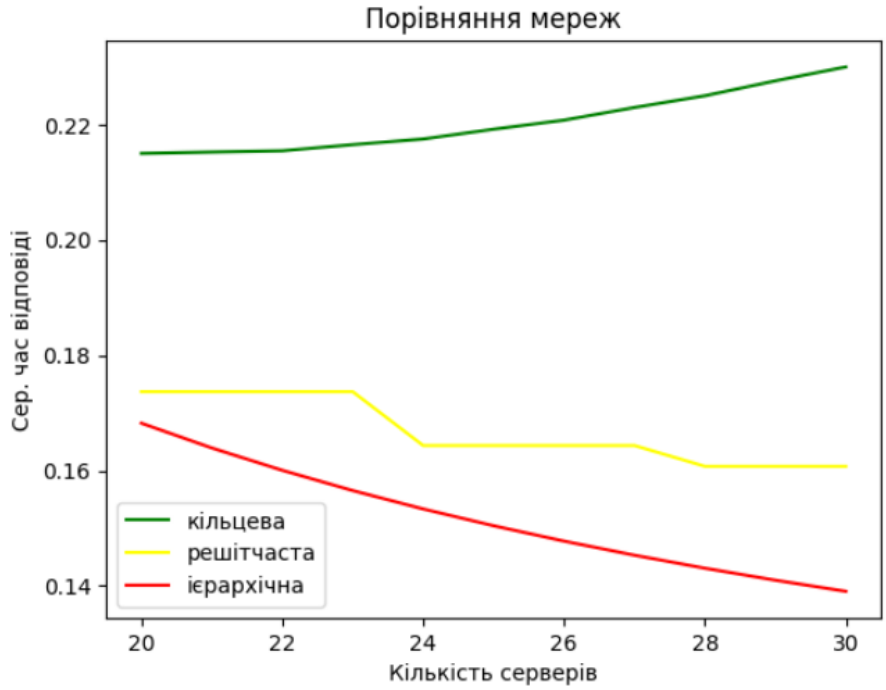


Рисунок 3.8 – Комірка «Порівняння мереж»

На рисунку 3.9 зображені результати порівняння вищеперерахованих мереж. Можна побачити графіки відповідних мереж на заданому відрізку графіка. Кожна мережа має свій колір графіка. Кільцева мережа – зелена, решітчаста – жовта, ієрархічна – червона. Знизу відображені оптимальні значення кількості серверів та кількості рівнів для кожної мережі.




```
Оптимальна кількість серверів (кільцева): 21
Оптимальна кількість рівнів мережі (решітчаста): 8
Оптимальна кількість серверів (ієрархічна): 146
Оптимальна кількість рівнів мережі(ієрархічна): 4
```

Рисунок 3.9 – Результати комірки «Порівняння мереж»

3.4 Висновки до розділу

У даному розділі було розглянуто процес розробки програмного продукту, спрямованого на оптимізацію розподілу даних у розподілених базах даних.

Середовищем розробки було обрано Google Colab. Google Colab - це безкоштовне хмарне середовище розробки, яке має велику колекцію вбудованих бібліотек. Мова Python – це основна мова програмування, що підтримується в Google Colab. Тому вона була обрана як мова програмування для розробки програмного продукту.

В кінці був наданий детальний опис інтерфейсу користувача для створеного програмного продукту, а також описана інструкція користувача.

РОЗДІЛ 4 АНАЛІЗ МОДЕЛЕЙ ОПТИМАЛЬНОГО РОЗПОДІЛУ ДАНИХ З ВИКОРИСТАННЯМ РОЗРОБЛЕНОЇ ПРОГРАМИ

4.1 Кільцева мережа

Щоб отримати результати кільцевої мережі для заданих параметрів, треба ввести параметри у відповідну комірку та запустити її. Для цього експерименту були використані наступні параметри (рис. 4.1):

1) кількість записів (N) – 1000;

2) час пошуку (T_s) – 0.0021 с;

3) час переходу (T_r) – 0.01 с.

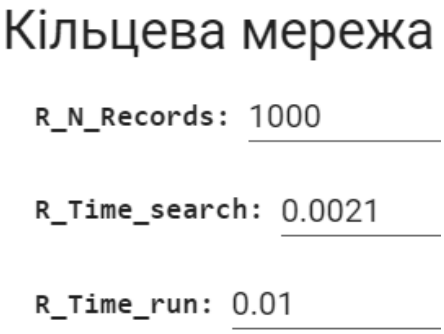
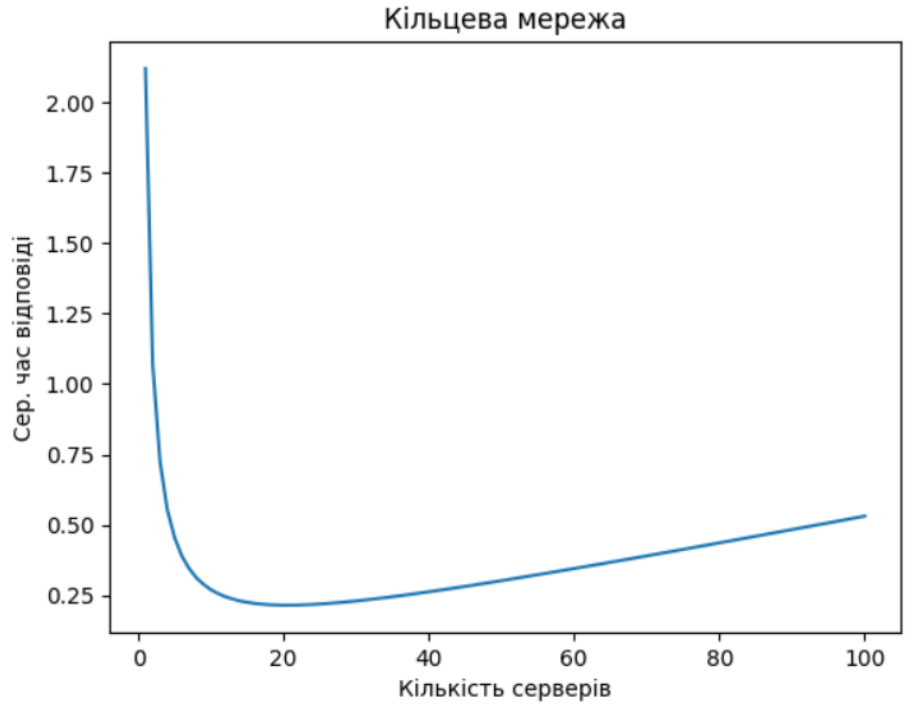


Рисунок 4.1 – Параметри для першого експерименту

Ці параметри будуть використані і для інших мереж.

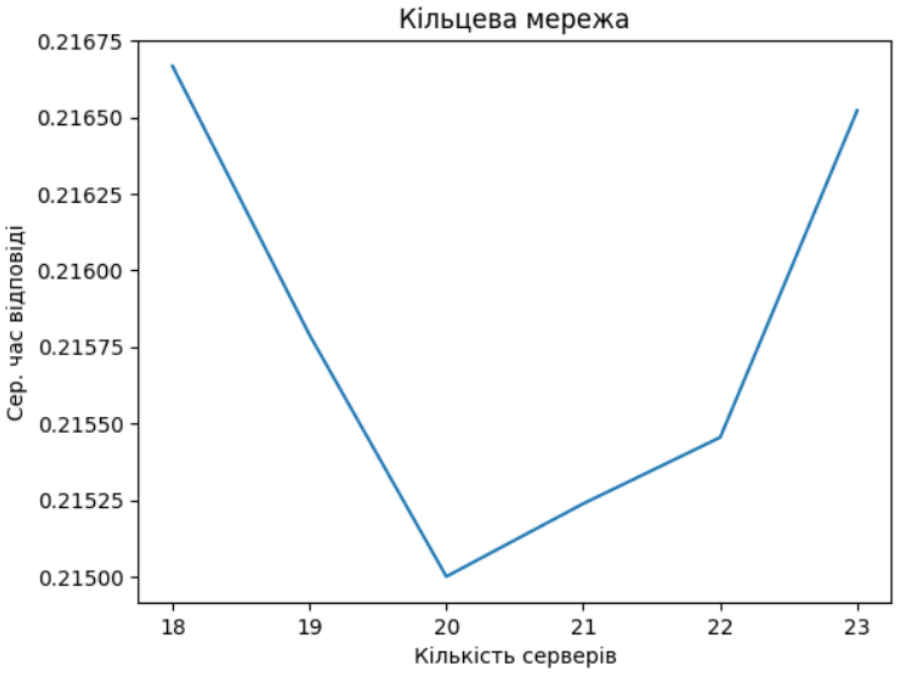
Можна провести аналіз отриманих результатів (рис. 4.2). Середній час відповіді T зменшується до певного моменту, до оптимального значення кількості серверів N_c^*, а далі середній час збільшується. При збільшенні кільця збільшується час передачі, тому після досягнення оптимального значення N_c^* сенсу додавати нові сервери немає.

Графік на меншому відрізку можна побачити на рисунку 4.3. Також можна побачити відрізок, в який входить N_c^*=20. Мінімальне значення середнього часу відповіді T^* досягається при N_c^*, T^* = 0.2152.




```
Оптимальна кількість серверів: 20
```

Рисунок 4.2 – Результати першого експерименту для кільцевої мережі




```
Оптимальна кількість серверів: 20
```

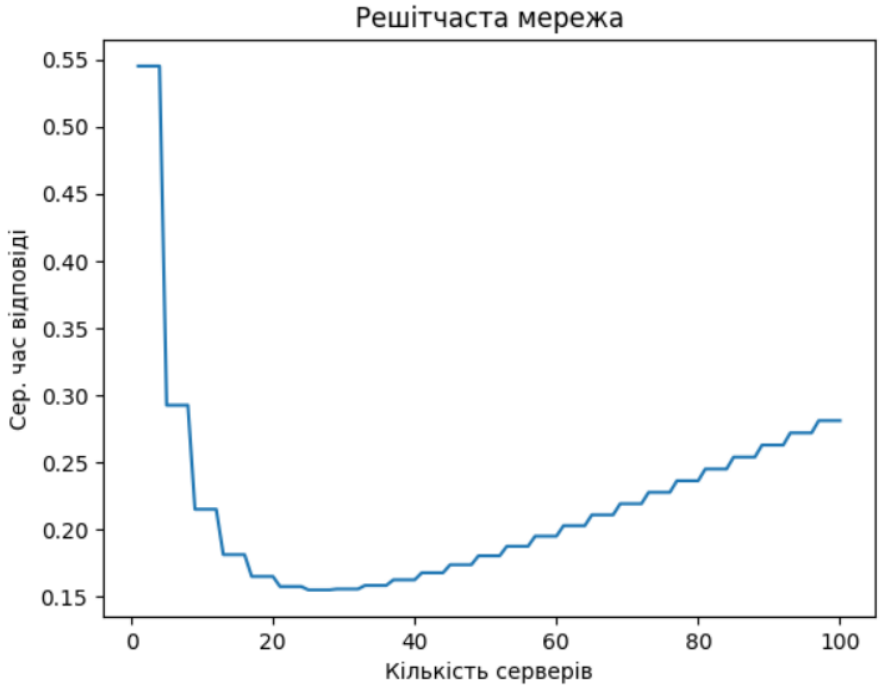
Рисунок 4.3 – Менший графік для кільцевої мережі

4.2 Решітчаста мережа

Можна провести аналіз отриманих результатів (рис. 4.4). Середній час відповіді T зменшується до певного моменту, до оптимального значення кількості серверів N_c^*, а далі середній час збільшується.

Розглянемо графік на меншому відрізку (рис. 4.5) в який входить N_c^*=36. На цьому графіку краще помітно, що значення T є однаковим на одному рівні. Рівень складається з 4 серверів.

Мінімальне значення середнього часу відповіді T^* досягається при N_c^*, T^* = 0.1582. Порівняно з кільцевою мережею оптимальний середній час відповіді менший, але і серверів потрібно значно більше.




```
Оптимальна кількість рівнів мережі: 9
```

Рис 4.4 – Результати першого експерименту для решітчастої мережі

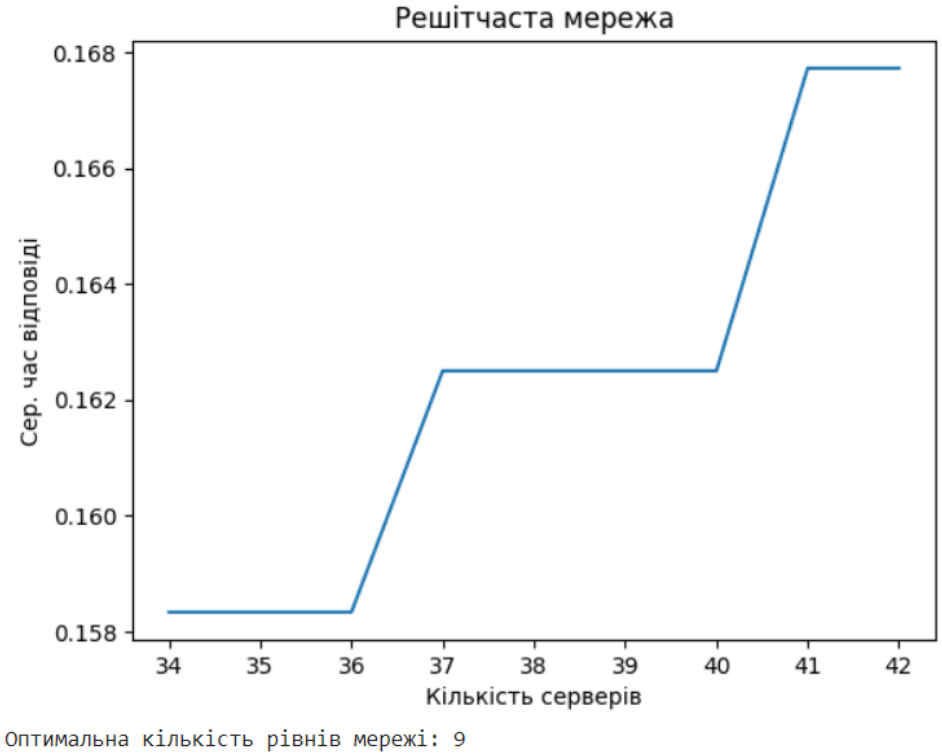


Рисунок 4.5 – Менший графік для решітчастої мережі

4.3 Ієрархічна мережа

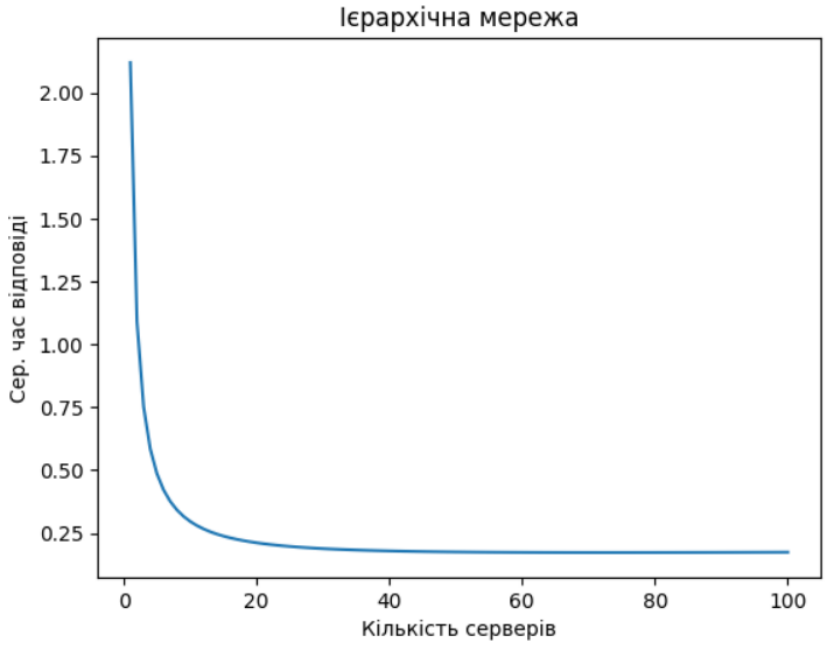
Згідно встановлених параметрів сервери в ієрархії мають по два підпорядковані сервери меншого рангу.

Можна провести аналіз отриманих результатів (рис. 4.6). Середній час відповіді T досягає оптимального значення кількості серверів N_c^*, а далі більше оптимального.

Розглянемо графік на меншому відрізку (рис. 4.7) в який входить N_c^*=73. На графіку помітно, що при збільшенні кількості серверів збільшується середній час відповіді T. Це викликано тим, що збільшується кількість переходів, що в свою чергу збільшує час переходу Tr.

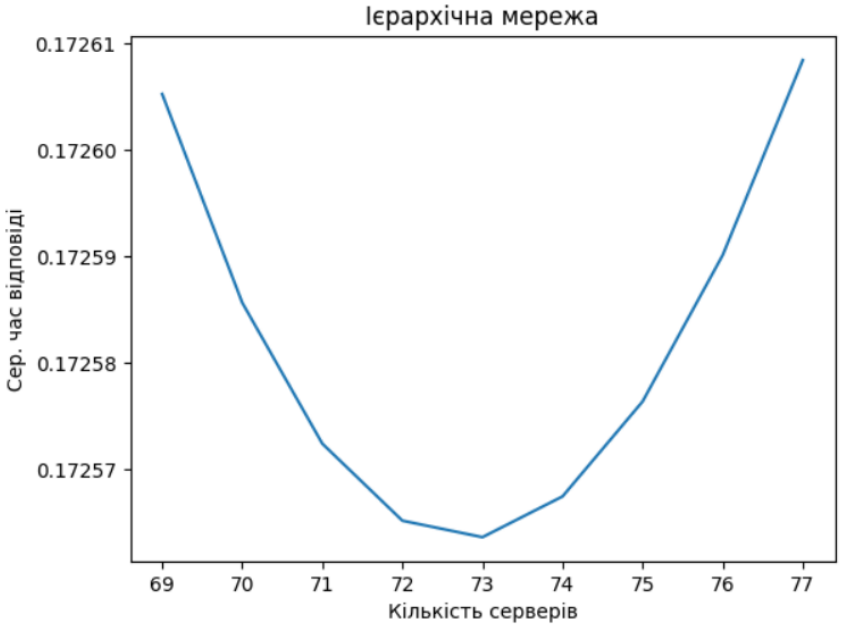
Мінімальне значення середнього часу відповіді T^* досягається при N_c^*, T^* = 0.1724. Порівняно з кільцевою мережею оптимальний середній час

відповіді менший, але і серверів потрібно значно більше. Решітчаста мережа показала кращі результати по обом показникам.




```
Оптимальна кількість серверів: 73
Оптимальна кількість рівнів мережі: 7
```

Рисунок 4.6 – Результати першого експерименту для ієрархічної мережі




```
Оптимальна кількість серверів: 73
Оптимальна кількість рівнів мережі: 7
```

Рисунок 4.7 – Менший графік для ієрархічної мережі

4.4 Порівняння мереж

У попередніх підрозділах було проведено порівняння при наступних параметрах: кількість записів (N) – 1000; час пошуку (T_s) – 0.0021 с; час переходу (T_r) – 0.01 с. Для цього експерименту оптимальною за середнім часом відповіді виявилась решітчаста мережа.

Проведемо експеримент для вдвічі більшої кількості записів. Отже, параметри наступні (рис. 4.8): кількість записів (N) – 2000; час пошуку (T_s) – 0.0021 с; час переходу (T_r) – 0.01 с; кількість зв'язків ієрархічної мережі – 2.

Найменший оптимальний середній час відповіді $T^* = 0.1928$ досягається ієрархічною мережею (рис. 4.9, рис. 4.10), але за рахунок великої кількості серверів [7].

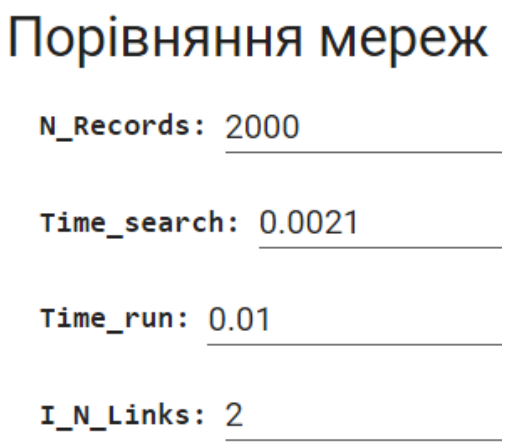
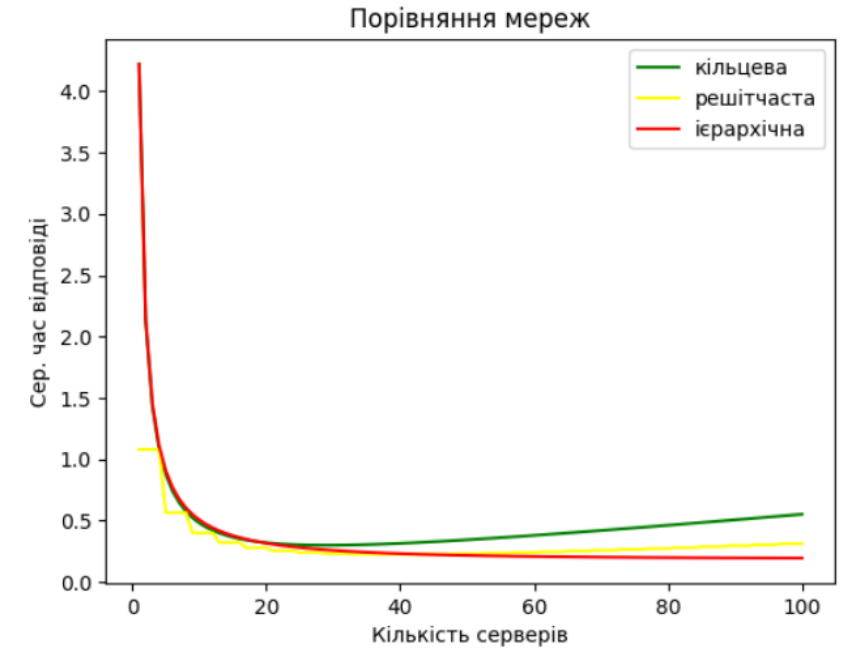


Рисунок 4.8 – Параметри для всіх мереж при другому експерименті




```
Оптимальна кількість серверів (кільцева): 30
Оптимальна кількість рівнів мережі (решітчаста): 10
Оптимальна кількість серверів (ієрархічна): 146
Оптимальна кількість рівнів мережі(ієрархічна): 8
```

Рисунок 4.9 – Результати другого експерименту

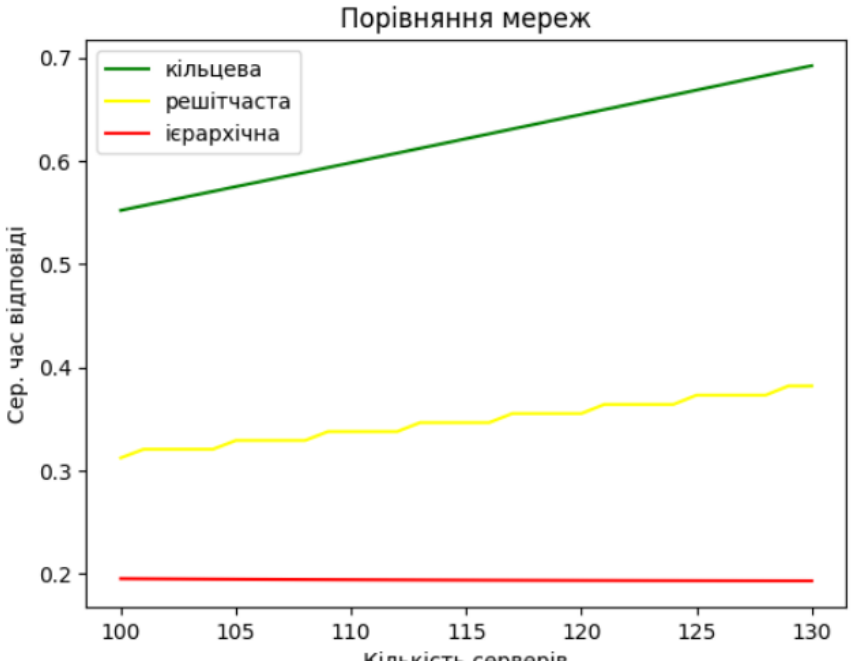


Рисунок 4.10 – Розширення графіку

Якщо замість збільшення вдвічі кількості запитів, збільшити час пошуку, то отримаємо результати ідентичні другому експерименту.

4.5 Висновки до розділу

В цьому розділі було проведено два експерименти в реалізованому програмному продукті. Параметри при першому експерименті:

1) кількість записів (N) – 1000;

2) час пошуку (Ts) – 0.0021 с;

3) час переходу (Tr) – 0.01 с;

4) кількість зв’язків ієрархічної мережі (I) – 2.

Моделі порівнювалися за оптимальним середнім часом відповіді T*. Результати були наступні:

1) для кільцевої мережі T* = 0.2152 с;

2) для решітчастої мережі T* = 0.1582 с;

3) для ієрархічної мережі T* = 0.1724 с.

При другому експерименті кількість записів була подвоєна, N = 2000. Результати були наступні:

1) для кільцевої мережі T* = 0.3 с;

2) для решітчастої мережі T* = 0.225 с;

3) для ієрархічної мережі T* = 0.1928 с.

У першому експерименті найкращий результат надала решітчаста модель. У другому – ієрархічна модель. Можна зробити наступний загальний висновок: кільцева та решітчаста моделі дають найкращі результати при малих значеннях параметру N (кількість записів), а ієрархічна модель дає найкращі результати при великих значеннях N.

РОЗДІЛ 5 РОЗРОБКА ВЛАСНОГО СТАРТАП ПРОЕКТУ

Зараз у сучасному світі існує велика потреба ефективного використання баз даних. Наразі є можливість застосування розподілених баз даних. Для ефективної роботи з ними – важливо оптимально розподіляти дані на серверах. Тому в результаті дослідження отримуємо моделі, що допомагають оцінити середній час відповіді в залежності від структури мережі. Все це означає, що проблема, яку вирішує стартап, являється актуальною і може стати успішним на ринку та стати основним вибором для переважної кількості цільової аудиторії.

5.1 План розробки стартапу та масштабування його на ринок

Наведемо план розробки стартапу та виведення його на ринок. Спочатку треба провести маркетинговий аналіз, який включає в себе:

- конкурентний аналіз, щоб зрозуміти, якими методами вирішення проблем вже користуються люди;
- формування ідеї самого проекту та виділення цільової аудиторії;
- розробити стратегію виведення товару на ринок, базуючись на аналізі ринкового середовища.

Наступним кроком являється організація самого стартапу. На цьому етапі мають бути:

- складений весь план та побудований таймлайн розробки та запуску продукту;
- запланований обсяг виробництва та оцінений потенційний обсяг ресурсу, який буде потрібен для виконання плану;

- розраховані витрати, необхідні для реалізації проекту, та витрати на запуск проекту.

Далі необхідно виконати фінансово-економічний аналіз та оцінити ризики стартап-проекту, в межах якого:

- визначити обсяг інвестиційних втрат;
- розрахувати основні фінансово-економічні показники проекту (собівартість, ціну продукту/послуги, податковий збір та чистий прибуток) та визначити показники інвестиційної привабливості проекту (рентабельність продажів, період окупності проекту);
- визначити основні ризики проекту та способи для їх запобігання.

Фінальним кроком являється розробка заходів з комерціалізації продукту. Цей крок являється важливим для масштабування та збільшення розмірів продукту.

Для того, щоб залучити інвесторів та знайти різні способи фінансування проекту, необхідно:

- провести дослідження на предмет інтересів потенційних інвесторів та бізнесів;
- скласти інвестиційну пропозицію, яка включає в себе як опис самого продукту та його теперішні розміри, так і можливі шляхи розширення та розвитку;
- обрати канали комунікації із потенційно зацікавленими персонами.

Далі наведемо результати виконання кожного з описаних кроків.

5.2 Опис ідеї стартап-проекту

Стартап-проект полягає у оцінці середнього часу відповіді у гетерогенних розподілених базах даних. Суть продукту стартапу полягає у

тому, що потрібно якомога краще прогнозувати даний час і обирати необхідну структуру мережі, враховуючи навантаження користувачів на систему.

У таблиці 5.1 наведена інформаційна карта стартапу.

Таблиця 5.1 – Інформаційна карта стартап-проекту

Назва проекту	WiseDataSharing
Автори проекту	Шмідт Анатолій Євгенович
Коротка анотація	Надати бізнесу та спеціалістам якісний інструмент для оцінки середнього часу відповіді у гетерогенних розподілених базах даних.
Термін реалізації проекту	10 місяців
Необхідні ресурси	Приміщення з комп'ютерами, доступом до Інтернету, доступ до електромережі Програмне забезпечення для розробки, хмарне сховище для даних, антивірусні програми Фінансові кошти на оплату заробітної плати виконавцям на термін 10 місяців, а також на такі витрати як: оренда приміщення, комунальні послуги, оренда хмарного сховища тощо
Опис проблеми, яку вирішує проект	Продукт вирішує задачу розподілу даних у гетерогенних базах даних.

Продовження таблиці 5.1

Головні цілі та завдання проекту	Метою проекту є створення програмного продукту, який на основі даних про гетерогенну розподілену базу даних запропонує оптимальну модель розподілу даних.
Очікувані результати	Привернення технологічних компаній до нашого стартапу, покращення розподілу даних для гетерогенних баз даних

5.3 Технологічний аудит ідеї проекту

Тепер можна розібрати ідею стартапу та провести конкурентний аналіз. У таблиці 5.2 наведений опис ідеї стартапу.

Таблиця 5.2 – Опис ідеї стартапу

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Основна ідея - створення комплексної системи, яка буде приймати на початкові значення та повертатиме та результат моделювання	1. Оцінка часових витрат на запити до гетерогенних баз даних	1. Можна буде ефективно планувати робочі процеси, оптимізовувати час відповіді
	2. Система збору відгуків про якість моделей для подальшого покращення системи	2. Система буде оновлюватись і якість прогнозування буде збільшуватись

Далі необхідно провести порівняльний аналіз конкурентів проекту та наведемо результати у таблиці 5.3.

Таблиця 5.3 – Порівняльний аналіз конкурентів проекту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів		W	N	S
		Власний проект	Аналог від конкурентів			
1.	Ефективність підбору моделі	Достатня ефективність	Достатня ефективність		+	
2.	Доступність по ціні	Доступна ціна	Невелика ціна		+	
3.	Персоналізація під регіон	Присутня	Відсутня	+		

Далі необхідно проаналізувати можливість технічно здійснити ідею проекту (таблиця 5.4).

Таблиця 5.4 – Технологічна здійсненність продукту

№ п/п	Ідея проекту	Технології і реалізації	Наявність технологій	Доступність технологій
1	Створення комплексної системи, яка буде приймати на початкові значення та повертатиме та результат моделювання	Використання мови програмування Python	Наявні	Доступні
2		Використання мови програмування C++	Наявні, необхідні допрацювання	Доступні
3		Використання мови програмування C#	Наявні, необхідні допрацювання	Доступні
Обрана технологія реалізації ідеї проекту: Python				

5.4 Аналіз ринкових можливостей запуску стартап-проекту

Далі необхідно провести попередній аналіз ринку для запуску стартап-проекту (таблиця 5.5).

Таблиця 5.5 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	1
2	Загальний обсяг продаж, грн/ум.од	4000
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Відсутні
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі (або по ринку), %	15 %

Тепер необхідно провести характеристику потенційних клієнтів, які можуть бути зацікавлені в проекті (таблиця 5.6).

Таблиця 5.6 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреби, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Пошук оптимальної моделі розподілу даних гетерогенних базах даних у хмарному середовищі	Користувачі баз даних	Цікавить загалом час відповіді від мережі	Простота використання
2	Оптимізація витрат	Системні адміністратори / DEV OPS	Необхідність бачити можливості для оптимізації ресурсів, при чому не втрачаючи продуктивність	Можливість вносити зміни до структури

Обрахуємо фактори загроз (таблиця 5.7) та можливостей (таблиця 5.8). Необхідно проаналізувати загрози, щоб зрозуміти можливі перешкоди при запуску продукт на ринок. Фактори можливостей же треба обрахувати, щоб знати усі сприятливі умови та по можливості ними скористатися.

Таблиця 5.7 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкуренція	Хоча ринок є відкритим і неосвоїним, на ньому вже є кілька великих гравців, які відомі на ринку, які вже мають свою цільову групу покупців	Знайти точки додаткової цінності для користувача
2	Ціна збуту	Конкуренти можуть коштувати менше через нижчу якість	Сфокусуватися на якості роботи моделі
3	Якість аналізу	Через комплексність задачі, моделі можуть мати проблеми на певних предметних областях	Мати достатній штаб і ресурси, для побудови різних моделей для різних предметних областей

Таблиця 5.8 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Універсальність	Продукт не залежить від конкретної структури гетерогенних баз даних	Зробити акцент при маркетингу, продовжувати розвиток як окремого продукту

Продовження таблиці 5.8

2	Простота у використанні	Від користувача треба лише ввести початкові параметри	Реалізувати зручний інтерфейс для вивчення
3	Якість та гарантії	Надавати найбільш якісні послуги та сервіси	Відповідність міжнародним стандартам

Далі необхідно розглянути питання конкуренції, а саме визначити її тип та рівень (таблиця 5.9).

Таблиця 5.9 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	У чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Вказати тип конкуренції: недосконала конкуренція	Представлено мало хороших методології та експертів	Зробити максимальним збут застосунку
2. За рівнем конкурентної боротьби: міжнародний	Наявні проекти, розроблені та можуть бути доступні у всьому світі	Розширити цільову аудиторію, розробити підхід на різних мовах

Продовження таблиці 5.9

3. За галузевою ознакою: внутрішньогалузева	Можуть працювати з різними галузями	Покращити персоналізацію
4. Конкуренція за видами товарів: товарно-родова	Конкуренція з аналізами інших систем та експертів	Підтримувати та покращувати якість існуючих функцій
5. За характером конкурентних переваг: неціновa	Різні компанії пропонують різну якість	Розробляти якісніші алгоритми і моделі
6. За інтенсивністю: марочна	Вже представлені компанії із сильним брендом	Предметно створити комунікаційну стратегію для вибудови свого бренду

Далі необхідно виконати аналіз конкуренції за моделлю 5 сил конкуренції Майкла Портера (таблиця 5.10).

Таблиця 5.10 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти у галузі	Потенційні конкуренти	Постачальники	Клієнти	Товаровиробники
	Інші існуючі методології та продукти	Якість, ціни, кількість користувачів, капіталовкладення	Фактори сили постачальників	Контроль якості, порівняння цін	Сила бренду, якість, ціна, масштаби
Висновки	Конкуренція з невеликою інтенсивністю, а також підігрітий ринок	Можливості входження на ринок, нові потенційні конкуренти	Постачальники відсутні	Клієнти не диктують умови роботи на ринку	Товаровиробники відсутні

Маючи результати аналізу конкуренції (таблиця 5.10), характеристики ідеї стартап-проекту (таблиця 5.5), характеристики потенційних клієнтів і їх вимоги до продукту (таблиця 5.6) та фактори ринкового середовища (таблиці 5.7 і 5.8) було сформульовано та обґрунтовано перелік факторів конкурентоспроможності (таблиця 5.11).

Таблиця 5.11 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Універсальність	Продукт не залежить від апаратної платформи як у більшості конкурентів
2	Простота у використанні	Від користувача треба лише дотримуватися підходу
3	Якість та гарантії	Надавати найбільш якісні послуги та сервіси

Тепер можна провести аналіз сильних та слабких сторін продукту (таблиця 5.12).

Таблиця 5.12 – Порівняльний аналіз сильних та слабких сторін системи

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів						
			-3	-2	-1	0	1	2	3
1	Універсальність	20	+						
2	Простота у використанні	16				+			
3	Якість та гарантії	11			+				

Далі проведено SWOT-аналіз продукту (таблиця 5.13).

Таблиця 5.13 – SWOT-аналіз стартап-проекту

Сильні сторони Універсальність Простота у використанні Якість та гарантії	Слабкі сторони Відсутність сильного бренду Не сформована база клієнтів Не підключені альтернативні канали маркетингу
Можливості Покращення системи Персоналізація	Загрози Нові системи та експерти Збут

Після проведення SWOT-аналізу, можна визначити сильні та слабкі сторони, можливості та загрози, пов'язані з конкуренцією та плануванням стартап-проекту. Далі буде спроектовано альтернативну ринкову поведінку для інтеграції стартап-проекту на ринок та приблизний час реалізації системного комплексу, з урахуванням потенційних проектів, що можуть бути виведені на ринок та наведемо результати у таблиці 5.14.

Таблиця 5.14 – Альтернативи ринкового впровадження стартап проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Вихід на ринок з нижче якістю	70%	4 місяці
2	Пропонувати одразу платне використання	50%	6 місяців

Продовження таблиці 5.14

3	Представлення користувачам системи без інтерфейсу	60%	5 місяці

У даному пункті був проведений детальний аналіз ринку та продукту. Також відповідно до результатів проведеного конкурентного аналізу, визначених факторів ринку та його сприятливість, описання ідеї та характеристик стартап-проекту, можна зробити висновок, що існують дуже сприятливі умови для виходу продукту на ринок.

5.5 Розроблення ринкової стратегії стартап-проекту

Для розробки ринкової стратегії продукту, у першу чергу, необхідно проаналізувати цільову аудиторію проекту (таблиця 5.15).

Таблиця 5.15 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит у межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Користувачі баз даних	Висока	25%	Висока	Середня
2	Великі бізнеси	Середня	20%	Середня	Середня

Продовження таблиці 5.15

3	Системні адміністратори/ DEV OPS	Середня	30%	Середня	Середня
Які цільові групи обрано: 1, 3					

Маючи аналіз цільових груп, далі визначимо базову стратегію розвитку продукту (таблиця 5.16).

Таблиця 5.16 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
1	1 та 3	Диференційованого маркетингу	Масштабування та максимізація	Оптимальних витрат

Для роботи в обраних сегментах ринку сформовано базову стратегію розвитку (таблиці 5.17, 5.18).

Таблиця 5.17 – Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
Ні	Так	Ні	Виклику лідера

Таблиця 5.18 – Визначення стратегії позиціонування

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)
Універсальність Простота у використанні Якість результатів	Оптимальних витрат	Універсальність Простота у використанні Якість та гарантії	Система, яка краще всіх забезпечує цілісність, конфіденційність та доступність інформації

5.6 Розроблення маркетингової програми стартап-проекту

Після проведеного комплексного аналізу, можна повноцінно описати ключові переваги концепції потенційного товару (таблиця 5.19) та побудувати концепцію маркетингових комунікацій (таблиця 5.20).

Таблиця 5.19 – Ключові переваги концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Якісна генерація	Збереження конфіденційних даних	Постійне покращення методики, які зможуть вберегти від сучасних кібератак
2	Універсальність	Методологія не залежить від апаратної платформи	Такого виду системний підхід може використовувати будь який користувач
3	Простий інтерфейс	Системний підхід дуже простий у використанні	Системний підхід із інтуїтивно зрозумілими пунктами для збереження конфіденційності, доступності та цілісності інформації

Таблиця 5.20 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламно о повідомлення	Концепція рекламного звернення
1	Пошук спеціалізованих систем	b2b продажі Зв’язок через теплі контакти Таргетована реклама у соціальних мережах Публікація в спеціалізованих виданнях, журналах	Точність Якість Універсальність	Поєднати повідомлення про те, що це якісна методологія яка є незалежною	Таргетована реклама на цільову аудиторію
2	Пошук доступного та дешевого продукту	Рекламні банери в Інтернеті, форуми, реклами від інфлюєнсерів	Простота	Вселити довіру у бренд та продукт	Реклама у лідерів думок Вивіски в публічних місцях Таргетована реклама на цільову аудиторію

5.7 Висновки до розділу

У даному розділі були розглянуті ключові аспекти розробки та масштабування стартапу, спрямованого на оптимальне використання розподілених баз даних. Починаючи з плану розробки та завершуючи аналізом ринкових можливостей, досліджено кожен етап становлення проекту.

Перш за все, план розробки надав структуру та логічний порядок дій для досягнення успішного впровадження ідеї стартапу. Технологічний аудит ідеї проекту розкрив ключові аспекти технічної реалізації та визначає можливі технічні виклики. Аналіз ринкових можливостей дозволив зрозуміти, наскільки вигідним та конкурентоспроможним стартап може бути на сучасному ринку. Розроблення ринкової стратегії та маркетингової програми стартапу визначило шляхи просування та взаємодії з цільовою аудиторією.

У цілому, розділ надає повний огляд стратегії розвитку стартапу.

ВИСНОВКИ

Метою цієї магістерської дисертації була розробка математичних моделей оптимізації розподілу даних у різноманітних структурах мережі, таких як ієрархічна, кільцева та решітчаста, а також розробка програмного продукту, що реалізує ці моделі.

Перед розробкою математичних моделей був проведений аналіз існуючих рішень для оптимального розподілу даних. Також була досліджена проблематика хмарних обчислень: наведені основні концепції та обмеження.

Була сформована загальна математична задача розподілу даних. Була проведена оптимізація моделей методами опуклої оптимізації для наступних типів мережі: кільцевої, решітчастої та ієрархічної.

Було створено програмний продукт на мові програмування Python у хмарному середовищі розробки Google Colab. Був наданий короткий опис мови програмування, середовища розробки, а також використаних додаткових бібліотек. Опис інтерфейсу і інструкція користувача були надані для полегшення роботи з програмою.

Проведено декілька експериментів та представлені результати отриманих моделей з використанням реалізованого програмного продукту. Всі експерименти супроводжувались ілюстраціями графіків для відповідних моделей.

В кінці був проведений аналіз стартап-проекту на основі створеної програми. В процесі аналізу створено план розробки та загальний опис проекту. Також були розкриті такі теми як технологічний аудит, ринкова та маркетингова стратегія цього стартап проекту.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Google Colab або Google Colaboratory: що це таке. Hardware libre. URL: https://www.hwlibre.com/uk/google-colaboratory/ (дата звернення: 20.10.2023).
2. Shaik K. Introduction to Google Colab. linkedin. URL: https://www.linkedin.com/pulse/introduction-google-colab-khaleel-shaik (дата звернення: 20.10.2023).
3. Real Python. Introduction to Python 3 – Real Python. Python Tutorials – Real Python. URL: https://realpython.com/python-introduction/ (дата звернення: 20.10.2023).
4. Python: плюси і мінуси мови, які завдання вирішує і чи варто вивчати. Avada Media. URL: https://avada-media.ua/ua/services/python-plyusy-i-minusy-yazyka-kakiye-zadachi-reshayet-i-stoit-li-izuchat/ (дата звернення: 20.10.2023).
5. NumPy documentation – NumPy v1.26 Manual. NumPy. URL: https://numpy.org/doc/stable/ (дата звернення: 20.10.2023).
6. What Is Matplotlib In Python? How to use it for plotting? - ActiveState. ActiveState. URL: https://www.activestate.com/resources/quick-reads/what-is-matplotlib-in-python-how-to-use-it-for-plotting/ (дата звернення: 20.10.2023).
7. Мухін В. Є., Шмідт А. Є. Моделі оптимального розподілу даних. Системні науки та інформатика: зб. доп. II науково-практ. конф. «Системні науки та інформатика», 4–8 груд. 2023 р. Київ, 2023. С. 192–197.
8. Networking complete. 2nd ed. San Francisco : Sybex, 2001. 877 p.

9. Комп’ютерні системи : методичні вказівки до лабораторних робіт / укл.: Баловсяк С. В., Одайська Х. С. Чернівці : Чернівецький національний університет ім. Ю. Федьковича, 2021. 72 с.
10. What is Tree Topology? Definition and Explanation - javatpoint. www.javatpoint.com. URL: https://www.javatpoint.com/what-is-tree-topology (дата звернення: 29.11.2023).
11. Difference between Ring Topology and Tree Topology - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/difference-between-ring-topology-and-tree-topology/ (дата звернення: 28.11.2023).

ДОДАТОК А ЛІСТИНГ ПРОГРАМНОГО ПРОДУКТУ

Комірка «Кільцева мережа»

```
#@markdown # Кільцева мережа

import matplotlib.pyplot as plt
import numpy as np
import math
R_N_Records = 1000 # @param {type:"integer"}
R_Time_search = 0.0021 # @param {type:"number"}
R_Time_run = 0.01 # @param {type:"number"}
N_Servers_MIN = 18 # @param {type:"integer"}
N_Servers_MAX = 23 # @param {type:"integer"}

R_NS_Opt =  round(((2 * (R_N_Records * R_Time_search - R_Time_run))
/R_Time_run)**(1/2))
R_Results=np.zeros((2,N_Servers_MAX))
for n in range(R_Results[0].size):
  R_Results[0][n]=n+1
   R_Results[1][n]=(R_N_Records  *  R_Time_search  /  (n+1))  +
(math.floor(((n+2)**2)/2))  / (n+1)* R_Time_run
plt.title('Кільцева мережа')
plt.plot(R_Results[0][N_Servers_MIN-
1:],R_Results[1][N_Servers_MIN-1:])
plt.xlabel('Кількість серверів')
plt.ylabel('Сер. час відповіді')
plt.show()
print("Оптимальна кількість серверів:",R_NS_Opt)
#@markdown ---
```

Комірка «Решітчаста мережа»

```
#@markdown # Решітчаста мережа
```

```
import matplotlib.pyplot as plt
import numpy as np
import math
C_N_Records = 1000 # @param {type:"integer"}
C_Time_search = 0.0021 # @param {type:"number"}
C_Time_run = 0.01 # @param {type:"number"}
N_Servers_MIN = 1 # @param {type:"integer"}
N_Servers_MAX = 100 # @param {type:"integer"}

def C_Func(l):
 return ((C_N_Records * C_Time_search / (4*l)) + (l+2)*C_Time_run)

C_L_Opt = round(C_Time_search * C_N_Records
/(4*C_Time_run)**(1/2))
C_Results=np.zeros((2,N_Servers_MAX))
for n in range(C_Results[0].size):
  C_Results[0][n]=n+1
  l=math.ceil((n+1)/4)
  C_Results[1][n]=C_Func(l)
plt.title('Решітчаста мережа')
plt.plot(C_Results[0][N_Servers_MIN-
1:],C_Results[1][N_Servers_MIN-1:])
plt.xlabel('Кількість серверів')
plt.ylabel('Сер. час відповіді')
plt.show()
if(C_L_Opt>1):
  if(C_Func(C_L_Opt)>C_Func(C_L_Opt-1)):
    C_L_Opt=C_L_Opt-1
print("Оптимальна кількість рівнів мережі:", C_L_Opt)
#@markdown ---
```

Комірка «Ієрархічна мережа»

```
#@markdown # Ієрархічна мережа
```

```
import matplotlib.pyplot as plt
import numpy as np
import math
I_N_Records = 1000 # @param {type:"integer"}
I_Time_search = 0.0021 # @param {type:"number"}
I_Time_run = 0.01 # @param {type:"number"}
I_N_Links = 2 # @param {type:"integer"}
N_Servers_MIN = 69 # @param {type:"integer"}
N_Servers_MAX = 77 # @param {type:"integer"}

def I_Func(n):
 return (I_N_Records * I_Time_search / n) + (((math.log(n)
/ math.log(I_N_Links)) + 1) * 2 * I_Time_run)

I_NS_Opt = math.ceil(I_N_Records*I_Time_search
*math.log(I_N_Links)
/(2*I_Time_run))
I_L_Opt = math.ceil(math.log(I_NS_Opt)/math.log(I_N_Links))
I_Results=np.zeros((2,N_Servers_MAX))
for n in range(I_Results[0].size):
  I_Results[0][n]=n+1
  I_Results[1][n]=I_Func(n+1)
plt.title('Ієрархічна мережа')
plt.plot(I_Results[0][N_Servers_MIN-
1:],I_Results[1][N_Servers_MIN-1:])
plt.xlabel('Кількість серверів')
plt.ylabel('Сер. час відповіді')
plt.show()

print("Оптимальна кількість серверів:", I_NS_Opt)
print("Оптимальна кількість рівнів мережі:", I_L_Opt)
#@markdown ---
```

Комірка «Порівняння мереж»

```
#@markdown # Порівняння мереж

import matplotlib.pyplot as plt
import numpy as np
import math
N_Records = 1000 # @param {type:"integer"}
Time_search = 0.0021 # @param {type:"number"}
Time_run = 0.01 # @param {type:"number"}
I_N_Links = 4 # @param {type:"integer"}
N_Servers_MIN = 20 # @param {type:"integer"}
N_Servers_MAX = 30 # @param {type:"integer"}

R_N_Records=C_N_Records=I_N_Records=N_Records
R_Time_search=C_Time_search=I_Time_search=Time_search
R_Time_run=C_Time_run=I_Time_run=Time_run

R_NS_Opt =  round(((2 * (R_N_Records * R_Time_search - R_Time_run))
/R_Time_run)**(1/2))+1
R_Results=np.zeros((2,N_Servers_MAX))
for n in range(R_Results[0].size):
  R_Results[0][n]=n+1
   R_Results[1][n]=(R_N_Records  *  R_Time_search  /  (n+1))  +
(math.floor(((n+2)**2)/2))  / (n+1)* R_Time_run

def C_Func(l):
 return ((C_N_Records * C_Time_search / (4*l)) + (l+2)*C_Time_run)

C_L_Opt =  round(
((5*C_Time_run*C_Time_run + 4 * C_N_Records * C_Time_run
* C_Time_search)**(1/2) + C_Time_run)/
(4 * C_Time_run))
C_Results=np.zeros((2,N_Servers_MAX))
for n in range(C_Results[0].size):
```

```
        C_Results[0][n]=n+1
        l=math.ceil((n+1)/4)
        C_Results[1][n]=C_Func(l)
      if(C_L_Opt>1):
        if(C_Func(C_L_Opt)>C_Func(C_L_Opt-1)):
          C_L_Opt=C_L_Opt-1

      def I_Func(n):
       return (I_N_Records * I_Time_search / n) + (((math.log(n)
      / math.log(I_N_Links)) + 1) * 2 * I_Time_run)

      I_NS_Opt       =       math.ceil(I_N_Records       *I_Time_search
*math.log(I_N_Links)
      /(2*I_Time_run))
      I_L_Opt = math.ceil(math.log(I_NS_Opt)/math.log(I_N_Links))
      I_Results=np.zeros((2,N_Servers_MAX))
      for n in range(I_Results[0].size):
        I_Results[0][n]=n+1
        I_Results[1][n]=I_Func(n+1)



      plt.title('Порівняння мереж')
      plt.plot(R_Results[0][N_Servers_MIN-
1:],R_Results[1][N_Servers_MIN-1:], color='green', label ='кільцева')
      plt.plot(C_Results[0][N_Servers_MIN-
1:],C_Results[1][N_Servers_MIN-1:],        color='yellow',        label
='решітчаста')
      plt.plot(I_Results[0][N_Servers_MIN-
1:],I_Results[1][N_Servers_MIN-1:], color='red', label ='ієрархічна')
      plt.xlabel('Кількість серверів')
      plt.ylabel('Сер. час відповіді')
      plt.legend(loc='best')
      plt.show()
```

```
print("Оптимальна кількість серверів (кільцева):",R_NS_Opt)
print("Оптимальна кількість рівнів мережі (решітчаста):", C_L_Opt)
print("Оптимальна кількість серверів (ієрархічна):", I_NS_Opt)
print("Оптимальна кількість рівнів мережі(ієрархічна):", I_L_Opt)

#@markdown ---
```