Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації та управління

## Лабораторна робота № 8 з дисципліни «ПАРАЛЕЛЬНІ ТА РОЗПОДІЛЕНІ ОБЧИСЛЕННЯ»

Тема: «\_ Розробка алгоритмів для розподілених систем клієнт-серверної архітектури.\_»

Виконав: Перевірив:

студент групи ІТ-02

Терешкович Максим.

Дата здачі \_\_\_\_\_\_\_\_\_\_\_\_

Захищено з балом \_\_\_\_\_\_\_\_\_\_\_

Київ 2023

**5.8 Завдання до комп’ютерного практикуму 8 «Розробка алгоритмів для розподілених систем клієнт-серверної архітектури»**

A picture containing text, font, line, screenshot

Description automatically generated

Щоб реалізувати дану лабораторну роботу було прийнято рішення використовувати мову програмування Python так, як кількість коду, яка потрібна для реалізації алгоритму – мінімальна.

Для розробки веб-застосування клієнт-серверної архітектури, що реалізує алгоритм множення матриць на стороні сервера, можна використати фреймворк Flask. Для виконання матричних операцій можна використати бібліотеку NumPy.

Фреймворк Flask та бібліотека NumPy є популярними і зручними інструментами для розробки веб-застосувань та матричних операцій відповідно.

Flask є легким та гнучким фреймворком для створення веб-застосувань, що дозволяє розробнику створити веб-застосування швидко та ефективно. Він має досить простий синтаксис, що дозволяє розробникам швидко створювати веб-сторінки та інші функції, що потрібні для веб-застосування.

NumPy, з іншого боку, є бібліотекою для наукових обчислень з особливим акцентом на маніпулювання масивами та матрицями. Вона надає швидку та ефективну обробку числових даних та дозволяє розробникам виконувати матричні операції без необхідності вручну програмувати алгоритми для цього. NumPy має багато вбудованих функцій для роботи з масивами та матрицями, що дозволяє розробникам швидко виконувати складні операції з даними.

Таким чином, використання Flask та NumPy в розробці клієнт-серверного веб-застосування для множення матриць забезпечує швидку та ефективну розробку, а також ефективну обробку числових даних та матриць на стороні сервера.

Зазначу зразу – я впевнений на 100 відсотків, що алгоритм буде працювати швидше тоді коли дані вже на сервері, бо не потрібно витрачати час на їх пересилку туди.

Для початку в нас є два файли які відповідають за сервер та його запуск.

A black background with white text

Description automatically generated with low confidenceЯкщо говорити коротко то якщо дані для обчислення вже знаходяться на сервері, то все що потрібно – відправити звичайний запит на нього для обчислення матриці. Сам сервер буде отримувати це запит і виконувати множення з даних, які вже задані в ньому. А на вихід ми отримаємо файл Json та час витрачений на алгоритм.

Файл on\_server.py:

A screen shot of a computer program

Description automatically generated with medium confidence

Цей код визначає програму Flask, яка очікує HTTP POST запити на кінцевій точці '/multiply'. Очікується, що запит міститиме матрицю випадкових значень з плаваючою комою розміром 200x200 у форматі JSON. Потім програма розбиває матрицю на менші частини і використовує багатопроцесорну обробку для обчислення точкового добутку кожної частини з іншою випадковою матрицею 200x200. Результати об'єднуються і повертаються у вигляді JSON-об'єкта, що містить результуючу матрицю. Нарешті, час, необхідний для завершення обчислень, виводиться на консоль.

Використання багатопроцесорної обробки дозволяє розділити обчислення на декілька ядер процесора, що може значно скоротити час, необхідний для завершення обчислень.

Файл on\_server\_client.py:

A picture containing text, screenshot, font, line

Description automatically generated

Цей код надсилає HTTP POST-запит на URL-адресу 'http://localhost:5000/multiply'. Після відправлення запиту код намагається розібрати JSON-відповідь за допомогою методу response.json(), припускаючи, що відповідь від сервера має формат JSON.

Приклад реалізації:

1)Запускаємо сервер(з матрицею 200\*200):

A picture containing text, screenshot, font

Description automatically generated

2)Робимо запит і отримуємо вирішену матрицю:

A picture containing screenshot, text

Description automatically generated

3)Перевіряємо час затрачений на виконання:

A screen shot of a computer

Description automatically generated with low confidence

Як бачимо – час виконання дорівнює 1.36 мс для матриці 200\*200 розташованої на сервері. Тепер перейдемо до завдання де матрицю треба відправляти на сервер.

Файл off\_server.py:

A screen shot of a computer program

Description automatically generated with low confidence

Тут майже все теж саме але тепер очікується, що запит буде містити дві матриці, MatrixA і MatrixB, обидві у форматі JSON. Потім програма виконує матричне множення цих двох матриць, використовуючи NumPy та мультипроцесорну обробку. Зокрема, кожен рядок MatrixA паралельно множиться на транспоновану матрицю MatrixB за допомогою методу Pool.map з модуля мультипроцесорної обробки. Отримані вектори рядків потім склеюються для формування результуючої матриці, яка повертається як JSON-об'єкт, що містить результуючу матрицю.

Код містить деякі оптимізації, такі як транспонування матриціB перед початком обчислень, щоб зробити обчислення точкового добутку більш ефективним, та використання багатопроцесорної обробки для розпаралелювання обчислень на декількох ядрах процесора.

Файл off\_server\_client.py

A screen shot of a computer

Description automatically generated with medium confidence

Тут ми генеруємо рандомні матриці розміром n x n за допомогою NumPy, а потім будує словник даних, що містить ці дві матриці у вигляді списків списків. Параметр json методу requests.post використовується для надсилання словника даних у вигляді JSON-об'єкта в тілі HTTP POST-запиту на кінцеву точку <http://localhost:5000/multiply>.

Приклад реалізації:

1)Запускаємо сервер:

A picture containing text, font, screenshot

Description automatically generated

2)Робимо запит і отримуємо вирішену матрицю:

A picture containing text, screenshot, software, multimedia software

Description automatically generatedA black screen with white text

Description automatically generated with low confidence3)Перевіряємо час затрачений на виконання:

A screen shot of a computer

Description automatically generated with low confidence

Як бачимо – час виконання дорівнює 1.66 мс для матриці 200\*200 не розташованої на сервері. Отже я був правий, що алгоритм працює швидше з даними, які вже на сервері і які не потрібно відправляти. Але проведемо ще декілька експериментів з матрицями різної розмірності, щоб впевнитися в цьому.

A screenshot of a computer

Description automatically generated with low confidence

Час замірений в мілісекундах.

1. Порівняти реалізацію алгоритму в клієнт-серверній системі та в розподіленій системі з рівноправними процесорами. **20 балів.**

Обчислення матриць у клієнт-серверній системі здійснюються на сервері, куди клієнти надсилають запити, а отримані результати повертаються їм. Продуктивність сервера можна покращити, використовуючи паралельне програмування, щоб обробляти запити від багатьох клієнтів одночасно.

У розподіленій системі з рівноправними процесорами кожен процесор обчислює свою частину результату, використовуючи технології, такі як MPI, OpenMP або CUDA. Це значно підвищує продуктивність системи, оскільки кожен процесор працює над своєю задачею паралельно з іншими процесорами.

Залежно від потреб і можливостей, можна використовувати клієнт-серверну або розподілену систему для обчислення матриць. Розподілені системи, як правило, потребують більшої складності в настройці та обслуговуванні, але можуть бути корисними для складних завдань. Клієнт-серверні системи можуть бути підходящим варіантом для більшості простих завдань.

Найбільше прискорення в роботі алгоритмів спостерігалося на розмірі матриць в 2000\*2000 = 2.1

***\*РОБОТУ НЕ ПЛАНУЮ ЗАХИЩАТИ – ОЦІНІТЬ БУДЬ ЛАСКА ЗВІТ\****