НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №1

на тему: «Синхронна комунікація»

з предмету «Проектування розподілених систем»

Виконав:

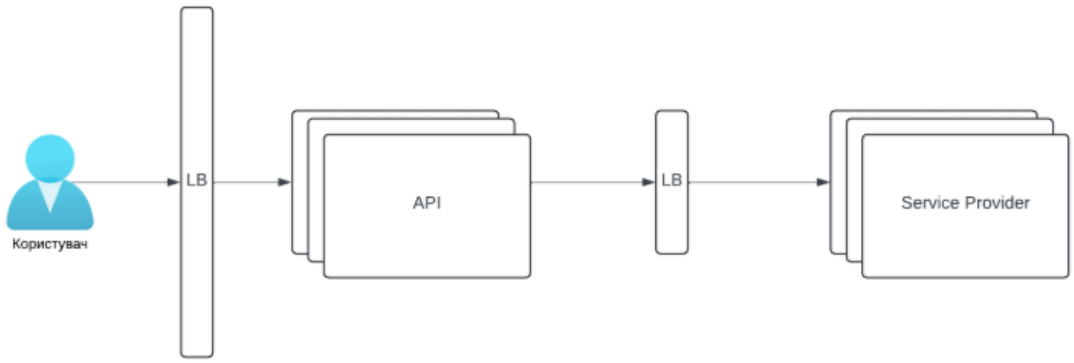
студент групи ІМ-31мн

Грибенко Єгор

Київ 2024

Завдання:

- Реалізувати синхронну комунікація між 2ма сервісами. Споживач Сервісу генерує завдання на обчислення і чекає відповіді від Постачальник Сервісу.
- Постачальник Сервісу має підраховувати час обчислення і логувати його для подальшого аналізу
- Споживач Сервісу має підраховувати час виконання запиту і логувати його для подальшого аналізу
- Розгорнути Load Balancer перед Споживачем Сервісу і/або Постачальником сервісу
- Опціонально: реалізувати протокол gRPC
- Опціонально: авторизація на рівні Споживача Сервісу
- Опціонально: авторизація на рівні Постачальника Сервісу



Хід роботи:

У ході роботи для реалізації серверів постачальника сервісу та споживача сервісу використаю засоби Python, а саме Flusk.

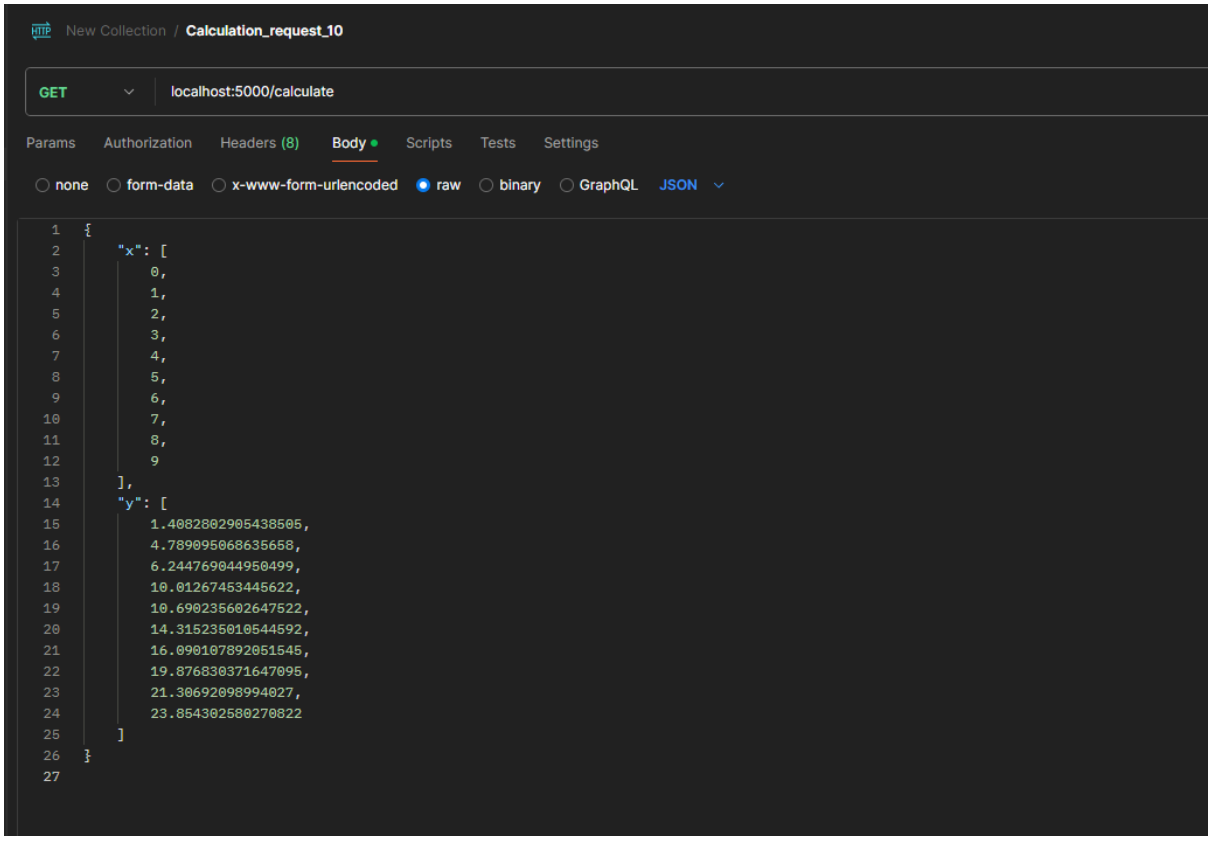
Для початку, відображу docker-compose файл, в якому можна побачити архітектуру проекту:

```yaml
services:

  nginx-api:
    container_name: lb-api
    image: nginx:latest
    volumes:
      - ./api/nginx.conf:/etc/nginx/nginx.conf
    depends_on:
      - api-1
      - api-2
    ports:
      - "5000:5000"

  api-1:
    container_name: api-1
    build: ./api
    environment:
      - FLASK_ENV=development
    volumes:
      - ./api:/app

  api-2:
    container_name: api-2
    build: ./api
    environment:
      - FLASK_ENV=development
    volumes:
      - ./api:/app

  lb-provider:
    container_name: lb-provider
    image: nginx:latest
    volumes:
      - ./service/nginx.conf:/etc/nginx/nginx.conf
    depends_on:
      - provider-1
      - provider-2

  provider-1:
    container_name: provider-1
    build: ./service
    environment:
      - FLASK_ENV=development
    volumes:
      - ./service:/app

  provider-2:
    container_name: provider-2
    build: ./service
    environment:
      - FLASK_ENV=development
    volumes:
      - ./service:/app
```

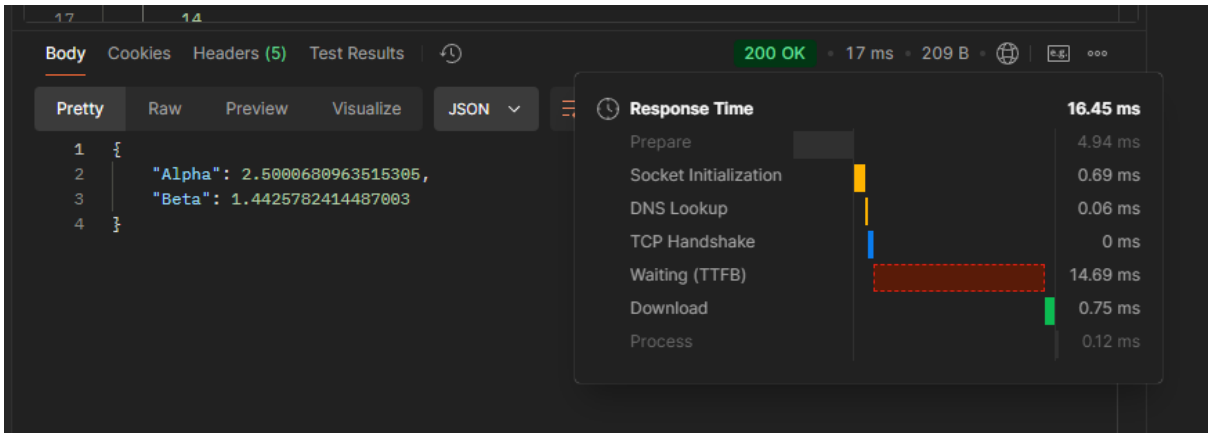
Бачимо, що в docker-compose я створив по два сервери-консьюмери (апі) та сервери-провайдери. При цьому є два контейнери Nginx, що балансують навантаження перед апі та перед сервіс провайдерами.

Як обчислення я обрав просту задачу лінійної регресії. Обчислення передбачає масиви абсцис та ординат на вхід, та віддає коефіцієнт нахилу та зміщення на вихід.

Маємо приклад запиту до сервісу:

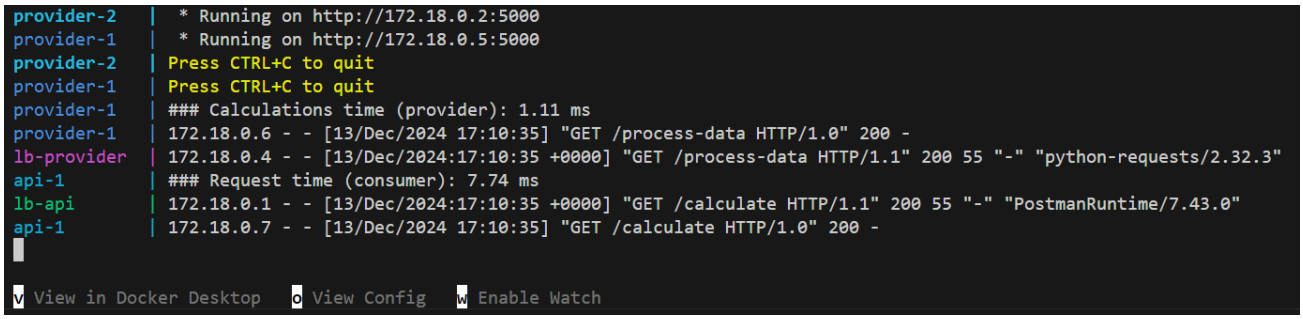


Отже, перевіримо виконання запиту на послідовність із 1000 точок:



Запит виконано за 17мс, 14мс з яких займає очікування відповіді від api.

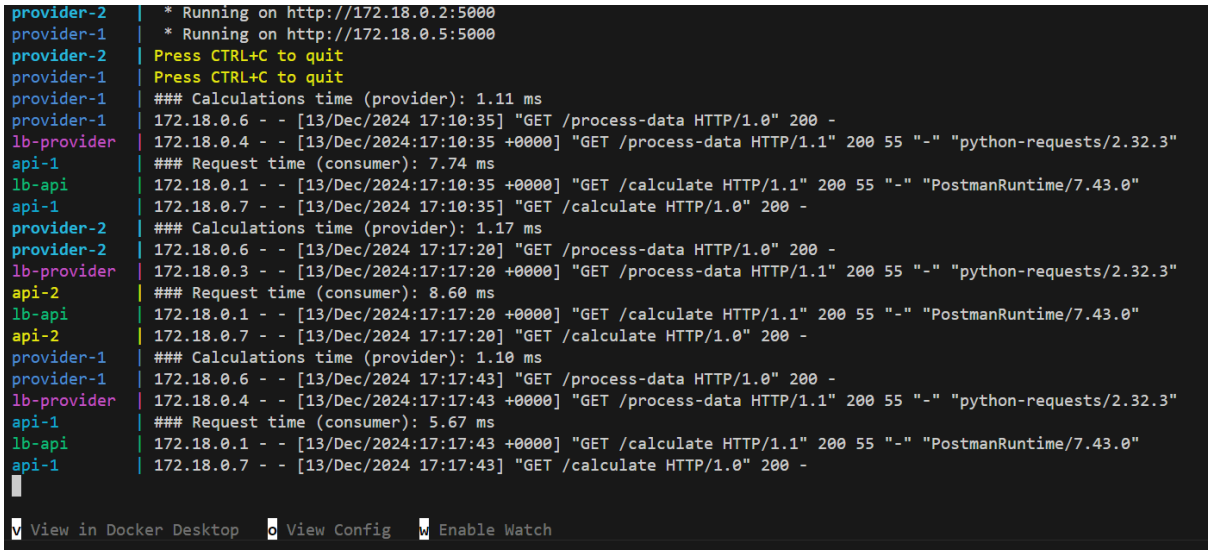
В логах можемо побачити більше про хід виконання операції.



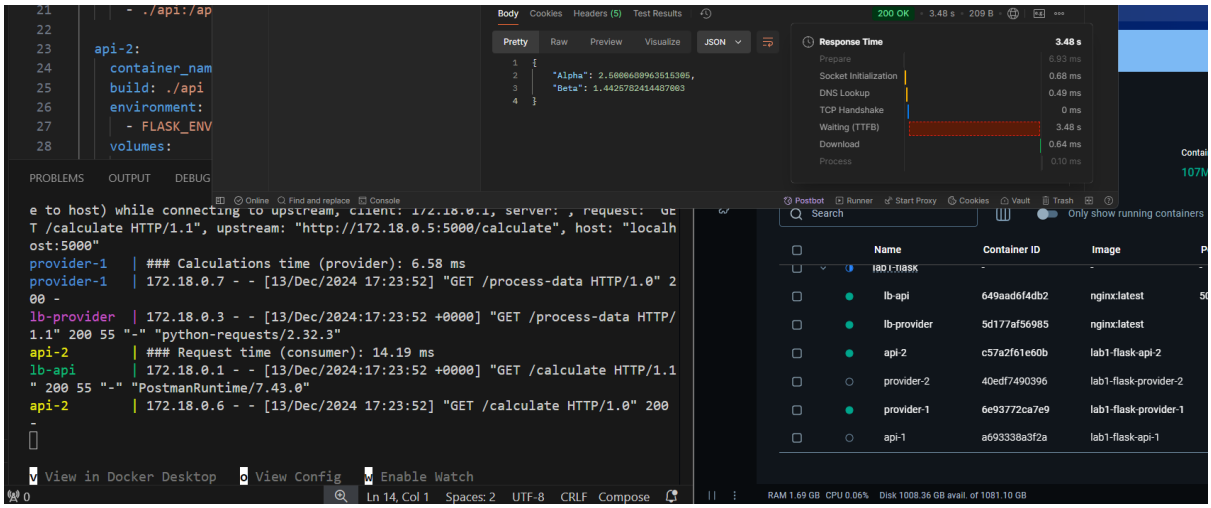
В логах бачимо, що запит пройшов через ноди api-1, лоад балансер api, лоад балансер сервіс-провайдерів та провайдер-1.

Самі обчислення зайняли 1.11 мілісекунди, а час запиту від api до сервера – 7.74 мілісекунд.

Тепер перевіримо, чи дійсно працюють лоад балансери. Якщо трафік перенаправляється між різними api/провайдерами, побачимо це у логах:

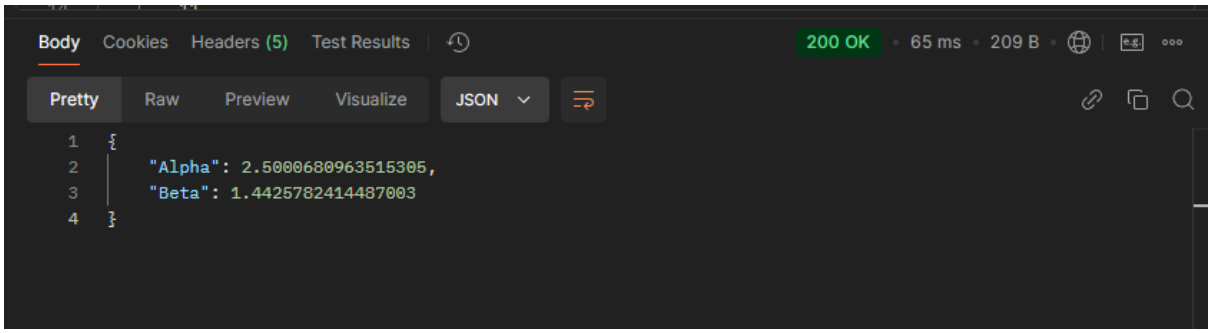


Бачимо, що api та провайдери почергово змінюються. Також можна помітити, що nginx одночасно змінює і api, і сервіс провайдера, тому маємо пари api-1, провайдер-1 та api-2, і провайдер-2. Логіку балансування можна змінити в конфігурації nginx, але зараз просто дропнемо контейнери api-1 та провайдер-2, щоб подивитися, чи nginx зможе працювати в таких умовах.



У вікні докера зправа бачимо, що контейнери, згадані раніше покладені. При цьому зверху бачимо, що реквест пройшов, але аж за 3 секунди.

Зліва по логах лачимо, що запит виконався через апі-2 та провайдер-1, отже лоад-балансер направив конекшн по доступним сервісам. При цьому наступні конекшни ідуть швидше тим самим шляхом (65 мс):



Отже, обидва лоад балансери працюють.

Посилання на репозиторій:

https://github.com/YehorHrybenko/lab1-rozpsys

Висновок:

В цій роботі я створив проект, що реалізує синхронну комунікацію між двома серверами – сервіс консьюмером та сервіс провайдером. Було розгорнуто два лоад балансери – перед споживачем сервісу та перед постачальниками сервісу. Після цього роботу балансувальників навантаження було перевірено і підтверджено. Час виконання запитів на споживачі сервісу та час виконання обчтиислень на стороні провайдера сервісу було виведено у логах.