НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №2

з предмету «Проектування розподілених систем»

Виконав:

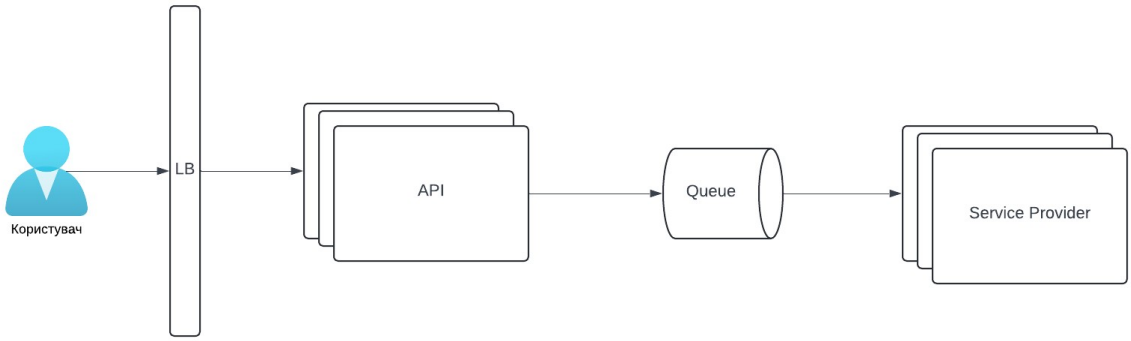
студент групи ІМ-31мн

Рекечинський Дмитро

Київ 2024

Завдання

- Реалізувати асинхронну комунікацію між Постачальником Сервісу і Споживачем Сервісу за допомогою Брокера Повідомлень
- 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
- Реалізувати горизонтальне масштабування засобами Брокера Повідомлень
- Реалізувати чергу с пріоритетами
- Реалізувати request-reply паттерн в асинхронній комунікації
- Порівняти результати синхронної і асинхронної комунікації



Виконання завдання

Для виконання завдання було створено 4 типи сервісів (всього 6 інстансів):

- Споживач сервісу consumer-service
  - Інстанс consumer-service-1
  - Інстанс consumer-service-2
- Постачальник сервісу provider-service
  - Інстанс provider-service-1
  - Інстанс provider-service-2
- Load balancer для consumer-service (lb_consumer)
- Брокер повідомлень RabbitMQ (rabbitmq)

RabbitMQ ідеально підходить в якості брокера повідомлень, тим паче, що у ньому реалізована концепція черги з пріоритетами.

Доволі тяжкою частиною було забезпечення того, що спочатку запуститься готовий для підключень сервіс RabbitMQ, а після нього — усі залежні сервіси (споживач сервісу та постачальник сервісу).

Втім, рішення знайшлось: це healthcheck. Суть така: вписуємо команду, яка перевіряє, чи готовий сервіс до роботи. Якщо ні, перевіримо ще раз через вказану кількість секунд. Якщо так, оповіщуємо про це всі залежні сервіси, і вони після цього запускаються.

Конфігурація healthcheck для RabbitMQ виглядає таким чином:

```
healthcheck:
  test: rabbitmq-diagnostics check_port_connectivity
  interval: 10s
  timeout: 5s
  retries: 10
  start_period: 5s
```

Для того, щоб враховувати статус healthcheck для залежних сервісів, слід змінити конфігурацію з такої форми:

```
depends_on:
  - rabbitmq
```

на таку:

```
depends_on:
  rabbitmq:
    condition: service_healthy
```

Якщо не вказувати condition явно, то його значення за замовчуванням є service_started, що буквально означає «як тільки сервіс запустився». У випадку із RabbitMQ це значення не підходить, оскільки ми орієнтуємось саме на статус готовності до підключень.

Демонстрація результатів:

```
~
❯ curl -X POST "http://localhost/add_task" -H "Content-Type: application/json" -
d '{"task": 42, "priority": 1}' | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   170  100   143  100    27    176     33 --:--:-- --:--:-- --:--:--   210
{
  "response": {
    "result": "110225727627443182468826115224975585937 5",
    "computation_time": 0.804,
    "provider": "provider-service1"
  },
  "request_time": 0.806
}
```

Рис. 1 — Виконання асинхронної комунікації за допомогою брокера повідомлень

```
~
❯ curl "http://localhost/generate_task?input_data=42" -H "Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjoiN2U5Zjc0MWYtZjM1Zi00YmY1LWExOWI
tMDZiMmM1ZGM3NmU5IiwiaWF0IjoxNzM0NTgxMTQ0LCJleHAiOjE3MzQ1ODE3NDR9.zChAUL8BVMfKRi
kEIlJU5pMIiiRFwKxf7_RXNdXGOJE" | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   172  100   172    0     0    225      0 --:--:-- --:--:-- --:--:--   225
{
  "response": {
    "result": "19711764481763478680787648613972589936 64",
    "computation_time": 0.715,
    "name": "provider-service_2"
  },
  "request_time": 0.752,
  "consumer": "consumer-service_1"
}
```

Рис. 2 — Виконання синхронної комунікації за допомогою балансувальника

Ефективність можна легко визначити за різницею між часом обчислення та часом на запит. У синхронній комунікації різниця складає 37 мс, в той же час, як брокер повідомлень зайняв 2 мс, і це лише для виконання одного запиту. При використанні більших навантажень ця різниця стане більш суттєвою.

Повна версія коду проекту розміщена за веб-адресою: https://github.com/rocket111185/distribution-systems/tree/release/lab2