### **Міністерство освіти і науки України**

### **Національний технічний університет України «Київський**

### **політехнічний інститут імені Ігоря Сікорського"**

### **Факультет інформатики та обчислювальної техніки**

### **Кафедра інформатики та програмної інженерії**

### **Звіт**

### з лабораторної роботи № 5

### з дисципліни

### «Компоненти програмної інженерії частина 3. Архітектура програмного

### забезпечення»

### «Масштабування системи»

### **Виконав** ІП-21 Скрипець О., Мунтяну А.

### **Перевірив** Зубик Л. В.

### **Київ 2024**

**Лабораторна робота №5**

**Тема:** Масштабування системи.

**Мета:** Масштабувати кожен елемент системи.

**Завдання:**

1. Зробити високорівневий capacity planning для системи:
   1. наприклад: скільки запитів може обслуговувати один інстанс мікросервіса, яке навантаження потягне база даних тощо
2. Вкажіть для кожного елементу системи, який тип масштабування буде використовуватись в кожному випадку:
   1. мікросервіси
   2. система обміну повідомленнями
   3. кеш (якщо є)
   4. база даних
   5. система потокової обробки даних (якщо є).
3. Вкажіть причини, з яких ви застосовуєте масштабування взагалі і кожний тип масштабування зокрема.
4. Вказати для кожного елементу системи, як відбувається балансування навантаження у випадку горизонтального масштабування (як відбувається диспетчеризація)

**1. Зробити високорівневий capacity planning для системи:**

Система обслуговує 10000 активних користувачів.

Пікове навантаження становить 2000 одночасних запитів. При цьому бажаний час відповіді становить не більше 2 секунд.

Очікуване навантаження на кожен елемент системи:

1. **API Gateway:** API Gateway робить пересилання запитів до відповідних мікросервісів. Оскільки вся кількість запитів пройде через API Gateway, кількість RPS для нього буде такою ж, як і пікове навантаження системи: 2000 RPS.
2. **Навантаження на кожен мікросервіс:**

Середнім навантаженням було б значення 2000 RPS / 7 мікросервісів ≈ 286 RPS на кожен мікросервіс. Враховуватимемо це число, виконуючи припущення для кожного мікросервісу.

* Сервіс пошуку ментора:
* Очікувана кількість запитів: 800 RPS, так як це одна з основних функцій платформи, доступна в тому числі незареєстрованим користувачам.
* Навантаження на базу даних: високе
* Сервіс аутентифікації та авторизації:
* Очікувана кількість запитів: 200 RPS
* Навантаження на базу даних: помірне
* Сервіс управління користувачами:
* Очікувана кількість запитів: 100 RPS
* Навантаження на базу даних: помірне
* Сервіс кабінету ментора:
* Очікувана кількість запитів: 150 RPS
* Навантаження на базу даних: середнє
* Сервіс бронювань:
* Очікувана кількість запитів: 400 RPS
* Навантаження на базу даних: дуже високе
* Сервіс консультацій:
* Очікувана кількість запитів: 100 RPS
* Навантаження на базу даних: помірне
* Сервіс оцінювання та відгуків:
* Очікувана кількість запитів: 200 RPS
* Навантаження на базу даних: середнє

3) **База даних:** 3000 RPS

4) **Message Broker (Apache Kafka):** 2000 RPS

5) **Інтеграція з зовнішними системами:** Інтеграція з сервісом оплати та notification-сервісом вимагатиме додаткових ресурсів мережі та серверів, а також спеціального врахування безпеки і продуктивності API.

**2. Вкажіть для кожного елементу системи, який тип масштабування буде використовуватись в кожному випадку:**

* 1. **мікросервіси - горизонтальне масштабування.**

При зростанні кількості користувачів і менторів доцільно використовувати горизонтальне масштабування для додавання нових екземплярів мікросервісів, щоб впоратися з навантаженням без перерв у роботі системи. Якщо один сервіс виявляється недоступним, інші можуть продовжувати працювати, забезпечуючи безперервну доступність функціоналу.

* 1. **система обміну повідомленнями - горизонтальне масштабування.**

У наслідок event-driven підходу, система обміну повідомленнями може стикатися з великим обсягом подій. Шляхом розподілення обробки подій між кількома серверами, горизонтальне масштабування забезпечує високу доступність та надійність системи, оскільки один вузол може продовжувати роботу у разі відмови іншого. Розподіл обробки подій між багатьма серверами також дозволяє підвищити пропускну здатність системи, щоб вона могла ефективно впоратися зі зростаючим навантаженням.

* 1. **кеш - горизонтальне масштабування.**

Кеш служить для збереження та швидкого доступу до даних, що часто використовуються. Горизонтальне масштабування дозволяє розділити кеш між різними серверами для збільшення обсягу даних, які можна зберігати, і підвищення швидкодії доступу до них. Окрім цього, з використанням горизонтального масштабування можна легко налаштувати резервне копіювання та відновлення кешу, що забезпечить надійність та стійкість до відмов системи.

* 1. **база даних - горизонтальне & вертикальне масштабування.**

Зі зростанням кількості користувачів, запитів та операцій в базі даних необхідно забезпечити достатню пропускну здатність для обробки всіх цих даних. Комбінація шардування та реплікації дозволить ефективно масштабувати базу даних.

Горизонтальне масштабування бази даних буде корисним для забезпечення масштабованості, надійності та високої доступності бази даних за рахунок розподілення навантаження між декількома серверами, кожен з яких містить лише частину даних.

Вертикальне масштабування відбувається за рахунок збільшення ресурсів на одному сервері. У системі воно буде використовуватися для бази даних менторів, адже до неї очікується найбільша кількість запитів і вона містить найбільше різноманітних даних.

**3.** **Вкажіть причини, з яких ви застосовуєте масштабування взагалі і кожний тип масштабування зокрема.**

Масштабування використовується для забезпечення продуктивності, надійності та доступності системи при зростанні навантаження. Воно дозволяє системі ефективно впоратися зі збільшенням кількості користувачів, запитів або даних.

1. **Горизонтальне масштабування** використовується, коли необхідно додати нові екземпляри системи для обробки збільшеного навантаження. Це дозволяє системі розподілити навантаження між декількома серверами, що забезпечує високу доступність та надійність. Якщо один сервер стає недоступним, інші можуть продовжувати працювати, забезпечуючи безперервну доступність функціоналу. Горизонтальне масштабування використовується для мікросервісів, системи обміну повідомленнями, кешу та бази даних.
2. **Вертикальне масштабування** використовується, коли необхідно збільшити ресурси наявного сервера для обробки збільшеного навантаження. Це може включати збільшення обсягу пам’яті, процесорного часу або місця на диску. Вертикальне масштабування використовується для бази даних.
3. **Комбіноване масштабування** використовується, коли один тип масштабування не є достатнім для задоволення потреб системи. Наприклад, база даних може використовувати комбінацію горизонтального та вертикального масштабування для забезпечення високої продуктивності, доступності та надійності. Горизонтальне масштабування дозволяє розподілити навантаження між декількома серверами, а вертикальне масштабування дозволяє збільшити ресурси кожного сервера. Комбінація цих двох типів масштабування дозволяє системі ефективно впоратися зі зростаючим навантаженням.

**4. Вказати для кожного елементу системи, як відбувається балансування навантаження у випадку горизонтального масштабування (як відбувається диспетчеризація)**

Давайте розглянемо, як відбувається балансування навантаження у випадку горизонтального масштабування для кожного елементу системи:

1. **Мікросервіси**:
   * Балансування навантаження для мікросервісів зазвичай виконується за допомогою балансувальника навантаження, який розподіляє вхідні запити між різними екземплярами мікросервісів.
   * Це може бути просте кругове балансування, де запити розподіляються послідовно між серверами, або більш складне балансування на основі навантаження, де балансувальник навантаження враховує поточне навантаження на кожен сервер при вирішенні, куди направити новий запит.
   * Це дозволяє системі розподілити навантаження між декількома серверами, що забезпечує високу доступність та надійність. Якщо один сервер стає недоступним, інші можуть продовжувати працювати, забезпечуючи безперервну доступність функціоналу.
2. **Система обміну повідомленнями**:
   * В системах обміну повідомленнями, таких як Kafka або RabbitMQ, балансування навантаження зазвичай виконується на рівні споживачів.
   * Кожен споживач в групі споживачів отримує повідомлення з певного набору розділів, що забезпечує рівномірне розподілення навантаження.
   * Це дозволяє системі розподілити навантаження між декількома серверами, що забезпечує високу доступність та надійність. Якщо один сервер стає недоступним, інші можуть продовжувати працювати, забезпечуючи безперервну доступність функціоналу.
3. **База даних**:
   * Для баз даних горизонтальне масштабування зазвичай досягається за допомогою шардування, де дані розподіляються між різними серверами.
   * Балансування навантаження може виконуватися за допомогою балансувальника навантаження, який розподіляє запити до бази даних між різними серверами на основі різних стратегій, таких як кругове балансування або балансування на основі навантаження.
   * Це дозволяє системі розподілити навантаження між декількома серверами, що забезпечує високу доступність та надійність. Якщо один сервер стає недоступним, інші можуть продовжувати працювати, забезпечуючи безперервну доступність функціоналу.