УДК 004.415:004.2

ПОРІВНЯЛЬНИЙ АНАЛІЗ АРХІТЕКТУРИ ОБРОБКИ СПОВІЩЕНЬ НА СТОРОНІ BACK-END: СИНХРОННОГО REST API ТА АСИНХРОННОГО ПІДХОДУ З APACHE KAFKA

Старіченко В.С., Лановий О.Ф.
e-mail: vladyslav.starichenko@nure.ua, oleksiy.lanovyy@nure.ua
Харківський національний університет радіоелектроніки, каф. ПІ
м. Харків, Україна

This work is devoted to a comparative analysis of synchronous REST API and asynchronous Apache Kafka approaches for notification processing architecture on the back-end side. The study examines key aspects such as performance, scalability, error-handling mechanisms, and fault tolerance. The results demonstrate that Kafka provides better fault tolerance and scalability under high loads, while REST API offers simpler implementation and immediate feedback. The research provides practical recommendations for choosing the optimal architectural approach depending on system requirements.

У сучасному світі цифрових технологій системи обробки сповіщень стають все більш критичними компонентами програмного забезпечення [1]. Згідно з дослідженнями, 70% розподілених систем використовують гібридні архітектури, комбінуючи синхронні та асинхронні патерни комунікації [2]. За даними аналітиків, обсяг оброблюваних сповіщень у корпоративних системах зростає на 40% щорічно, що створює нові виклики для архітекторів програмного забезпечення [3].

Метою даної роботи є порівняння двох підходів до обробки сповіщень на стороні back-end – синхронного (REST API) та асинхронного (Apache Kafka). Це дослідження покликане визначити переваги та недоліки кожного підходу, а також оптимальні умови їхнього використання.

Синхронний підхід з використанням REST API є одним із найбільш поширених способів організації взаємодії між мікросервісами. Архітектура включає наступні компоненти: Notification Service – головний компонент для обробки запитів; REST Sender – компонент для передачі даних; PostgreSQL – база даних для зберігання сповіщень. Для зменшення складності системи використовується окрема таблиця для зберігання нотифікацій, що дозволяє ізолювати логіку роботи з повідомленнями від основних бізнес-таблиць.

Асинхронний підхід з Apache Kafka базується на розподіленій платформі потокової обробки даних [3]. Фундаментальним поняттям в архітектурі Kafka є топік (topic), який розділяється на партиції для забезпечення паралелізму. Архітектура Kafka складається з брокерів, producers, consumers та ZooKeeper. Kafka реалізує механізм реплікації для забезпечення відмовостійкості та підтримує різні рівні гарантій доставки повідомлень.

На рис. 1 представлено порівняння потоків даних у двох досліджуваних архітектурах.

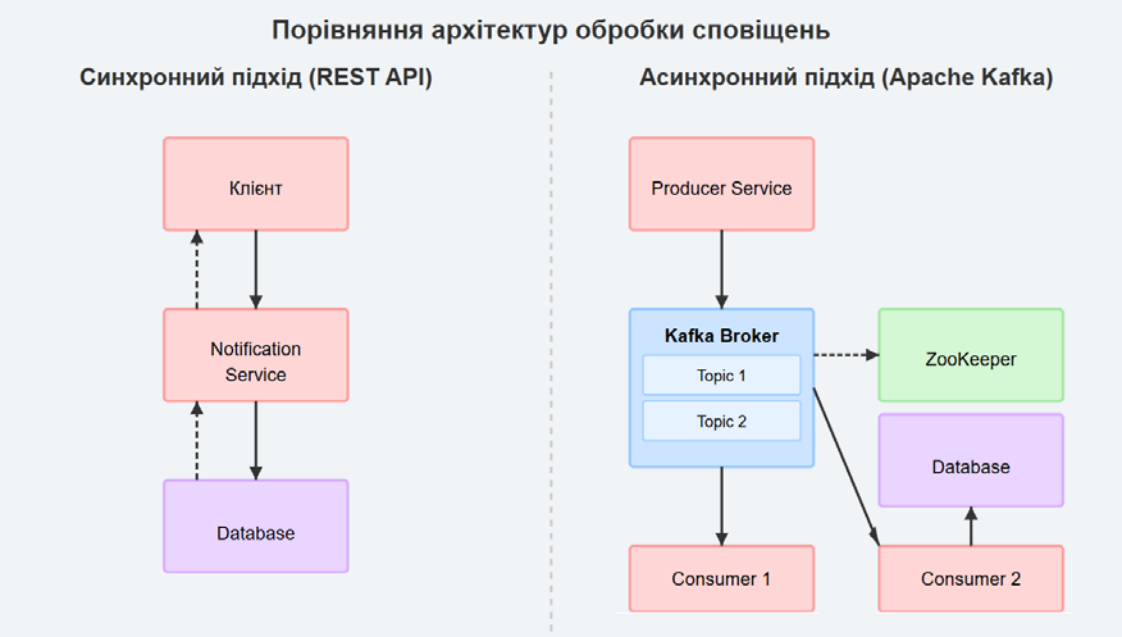

Рисунок 1 – Порівняння потоків даних у синхронній (REST API) та асинхронній (Apache Kafka) архітектурах обробки сповіщень на стороні back-end

Проведений порівняльний аналіз механізмів обробки помилок показав, що в синхронній REST API архітектурі обробка помилок базується на HTTP статус кодах, що дозволяє швидко реагувати на проблеми, але створює тісну зв'язність між сервісами. В асинхронній архітектурі з Apache Kafka використовуються інструменти автоматичних повторних спроб доставки, Dead Letter Queue (DLQ) та механізм підтвердження обробки, що забезпечує вищу надійність [3].

Експериментальне тестування продуктивності, проведене за допомогою Apache JMeter на тестовому середовищі з 3 серверами (4 vCPU, 16GB RAM), показало, що при однаковій максимальній пропускній здатності (~1000 повідомлень/с), Kafka демонструє значно нижчу латентність при пікових навантаженнях (50-100 мс проти 200-500 мс у REST API) та ефективніше використання ресурсів CPU (60-70% проти 85-95%). Виміри горизонтального масштабування при збільшенні кількості екземплярів сервісів з 1 до 10 показали, що Kafka демонструє зростання продуктивності, яке перевищує лінійне (на 20% більше очікуваного), що можна охарактеризувати як високу ефективність масштабування.

При дослідженні надійності та відмовостійкості, з використанням Chaos Monkey для симуляції відмов компонентів, встановлено, що синхронний підхід з REST API забезпечує базовий рівень надійності, але при виникненні збоїв повідомлення можуть бути втрачені. Асинхронний

підхід з Kafka надає вбудовані механізми реплікації даних, збереження повідомлень та автоматичного відновлення роботи, що забезпечує високу відмовостійкість системи.

В рамках проведення досліджень було виконано комплексне тестування обох підходів, що включає базовий тест (100 паралельних запитів, 30 хвилин), стрес-тест (до 1000 користувачів, 1 година) та довготривалий тест відмовостійкості (500 паралельних користувачів, симуляція відмов, 2 години).

На підсумку проведених досліджень можна зробити висновки, що REST API доцільно використовувати для систем з невеликим навантаженням, де важлива простота впровадження, а Apache Kafka є більш оптимальним для високонавантажених систем з високими вимогами до надійності та масштабованості [1]. Для досягнення балансу між перевагами обох підходів може бути доцільним використання гібридної архітектури при розробці back-end додатків.

Список використаних джерел:

1. Newman S. Building Microservices: Designing Fine-Grained Systems. O’Reilly Media, 2021. 280 p. URL: https://www.oreilly.com/library/view/building-microservices-2nd/9781492034018/ (дата звернення: 21.02.2025).

2. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. O’Reilly Media, 2017. 616 p. URL: https://www.oreilly.com/library/view/designing-data-intensive-applications/9781491903063/ (дата звернення: 21.02.2025).

3. Narkhede N., Shapira G., Palino T. Kafka: The Definitive Guide: Real-Time Data and Stream Processing at Scale. O’Reilly Media, 2017. 322 p. URL: https://www.oreilly.com/library/view/kafka-the-definitive/9781491936153/ (дата звернення: 21.02.2025).