УДК 004.41

Т. Ланевич
(Тернопільський національний технічний університет імені Івана Пулюя)

APACHE KAFKA ТА RABBITMQ: ПОРІВНЯННЯ АРХІТЕКТУР ТА МОЖЛИВОСТЕЙ

T. Lanevych

APACHE KAFKA AND RABBITMQ: COMPARING ARCHITECTURES AND CAPABILITIES

Архітектура, керована подіями (Event-driven architecture, EDA) – це патерн проєктування програмного забезпечення, який дозволяє організації виявляти «події» або важливі бізнес-моменти і реагувати на них у реальному часі або в режимі, близькому до реального. Ця модель замінює традиційну архітектуру «запит/відповідь», де сервіси повинні були чекати на відповідь, перш ніж перейти до наступного завдання. Використання Apache Kafka та RabbitMQ є популярними підходами для реалізації такої архітектури у масштабованих і гнучких проєктах, оскільки вони підтримують високу доступність, ефективність обміну повідомленнями і низьку затримку.

RabbitMQ – це потужний і гнучкий брокер повідомлень, заснований на протоколі AMQP (Advanced Message Queuing Protocol). Він використовується для керування маршрутизацією, доставкою та зберіганням повідомлень у розподілених системах. RabbitMQ дозволяє використовувати різні типи обміну повідомлень, що робить його хорошим вибором для багатьох різних додатків.

Apache Kafka – це платформа для потокової обробки подій, що дозволяє ефективно обробляти величезні обсяги даних у реальному часі. Kafka є розподіленою системою, де події передаються між виробниками (producers) та споживачами (consumers) через так звані теми (topics).

Основні відмінності між RabbitMQ та Apache Kafka:

1. Архітектура RabbitMQ базується на чергах, де повідомлення тимчасово зберігаються до моменту їх використання. Кожна черга підключена до біржі, яка спрямовує повідомлення до потрібної черги на основі визначених правил. Ця архітектура добре підходить для систем, які потребують складної маршрутизації та надійної доставки повідомлень. Kafka використовує розподілену архітектуру, засновану на незмінних журналах (immutable logs). Повідомлення записуються в теми і залишаються там, навіть після того, як їх буде прочитано. Це дозволяє декільком користувачам читати одні й ті ж повідомлення без шкоди для продуктивності, що робить Kafka дуже масштабованою і придатною для великих обсягів даних.

2. RabbitMQ часто є найкращим вибором для додатків, які потребують надійної доставки повідомлень і підтримки складних шаблонів обміну повідомленнями. Він широко використовується в системах електронної комерції, де важливо забезпечити надійну і своєчасну доставку замовлень, транзакцій і повідомлень. Kafka – це правильний інструмент для систем, які потребують обробки великих обсягів даних у режимі реального часу, наприклад, для моніторингу конвеєрів даних або потокової передачі даних. Його архітектура дозволяє ефективно обробляти безперервні потоки даних, що робить його

ідеальним для таких програм, як моніторинг інфраструктури, аналіз даних у реальному часі та інтеграція даних між різними системами.

3. Хоча RabbitMQ можна масштабувати горизонтально, він має деякі обмеження в проєктах з дуже високою пропускною здатністю. Його модель черги та обміну є потужною, але може мати проблеми у сценаріях з надзвичайно великими обсягами повідомлень та низькими вимогами до затримок. Kafka був розроблений для високої масштабованості. Він може обробляти петабайти даних без втрати продуктивності завдяки розподіленій архітектурі та управлінню журналами. Це робить Kafka найкращим вибором для додатків, які потребують масштабного горизонтального масштабування.

4. Обидві системи забезпечують збереження повідомлень, але різними способами. RabbitMQ дозволяє зберігати повідомлення на диску, гарантуючи, що вони не будуть втрачені у випадку збою. Однак, існують деякі обмеження на те, як цими повідомленнями керувати і видаляти їх з черг. Kafka зберігає повний журнал подій у своїх темах. Повідомлення залишаються в журналі протягом певного часу, навіть якщо вони були використані. Це забезпечує більшу довговічність і дозволяє користувачам із затримкою отримати доступ до попередніх повідомлень без впливу на продуктивність системи.

Використання RabbitMQ найкраще підходить для застосукнів, в яких важливо мати:

1. Складну маршрутизацію повідомлень на основі певних правил.
2. Гарантовану та надійну доставка повідомлень.
3. Легку та швидку інтеграцію в проєктах з невеликим обсягом повідомлень і низькою затримкою.

Apache Kafka краще підійде для:

1. Обробки великих обсягів даних у реальному часі.
2. Застосунків, які потребують високої масштабованості та низької затримки.
3. Сценаріїв, коли декілька споживачів мають доступ до одних і тих самих повідомлень, наприклад, у конвеєрах даних.

Висновок. Вибір між Apache Kafka та RabbitMQ залежить від характеру системи та вимог до обробки подій. Kafka більше підходить для великих, масштабованих і високонавантажених систем, де потрібно обробляти потоки даних в реальному часі. RabbitMQ є чудовим вибором для більш традиційних систем з фокусом на асинхронну обробку подій з низькою затримкою та високою гнучкістю в маршрутизації.

Література

1. Implementing event-driven architecture with Apache Kafka [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/abb-bank/implementing-event-driven-architecture-with-apache-kafka-2eaa26181917.

2. RabbitMQ vs Kafka: A Comparative Analysis of Mechanism, Concepts, Features, & Use Cases [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@byanalytixlabs/rabbitmq-vs-kafka-a-comparative-analysis-of-mechanism-concepts-features-use-cases-4420dafc8280.

3. RabbitMQ vs. Kafka: Which One to Choose for Your Event-Driven Architecture? [Електронний ресурс] – Режим доступу до ресурсу: https://dev.to/guilhermesiqueira/rabbitmq-vs-kafka-which-one-to-choose-for-your-event-driven-architecture-3enm.