Аналіз можливостей Apache Kafka в рамках забезпечення стримінгу Big Data

Олег Дашкевич
кафедра Програмної Інженерії
Харківський національний університет
радіоелектроніки
Харків, Україна
oleh.dashkevych@nure.ua

Ігор Шубін
кафедра Програмної Інженерії
Харківський національний університет
радіоелектроніки
Харків, Україна
igor.shubin@nure.ua

Analysis of the Apache Kafka capabilities as part of providing Big Data Streaming

Oleh Dashkevych
Department of Software Engineering
Kharkiv National University
of Radio Electronics
Kharkiv, Ukraine
oleh.dashkevych@nure.ua

Igor Shubin
Department of Software Engineering
Kharkiv National University
of Radio Electronics
Kharkiv, Ukraine
igor.shubin@nure.ua

***Анотація*—Оскільки Інтернет речей займає більше опору в різних галузях промисловості, труднощі з обробкою всіх трьох «V» Big Data будуть зростати. Все більше і більше програм і організацій створюють файли та документи, а не лише цінності та цифри. Потокове передавання може здійснювати більшу частину обробки в режимі реального часу. Це призвело до третього (на момент написання: поточної) хвилі Big Data: переходу до повного обхвату потокового (іноді називається каппа-архітектура, як еволюція лямбда-архітектури)**

Abstract*—*As the Internet of Things takes more of a foothold in various industries, the difficulties in handling all three «V's» of Big Data will increase. More and more applications and organizations are generating files and documents rather than just values and numbers. Streaming could do most of our processing in real-time. This led to the third (at the time of writing: current) wave of big data: the move to a full embrace of streaming (sometimes referred to as the kappa architecture, as an evolution of lambda architecture).

Ключові слова*—*Big Data; Kafka; брокер; тема; реплікація; дані; ZooKeeper

Keywords*—*Big Data; Kafka; broker; topic; replication; data; ZooKeeper

I. Вступ

У новій моделі побудови архітектури систем, що накопичують великі дані, події транслюються безперервно, з припущенням, що вони необмежені - вони можуть ніколи не закінчуватися - таким чином, системи більше не можуть чекати, щоб отримувати "всі дані", а замість цього потрібно обробляти їх на льоту, як тільки ті надходять. Це вимагає нових методів: неможливо отримати повний перегляд всіх даних перед його обробкою, але ви повинні визначити своє вікно обробки - як слід групувати вхідні події і де "окреслити лінію" перед обробкою групи - і розрізняти час події (коли це відбувається) і час обробки (коли він обробляється). З потоковою передачею фундаментальний зсув рухається від "даних в стані спокою" до "даних в русі" - від відтворення до реального часу. Потокове передавання даних не тільки для вузькоспеціалізованих проектів.

Нові технології та архітектурні проекти дають змогу створювати гнучкі системи, які не тільки більш ефективні та прості у побудові, але також краще моделюють процес ведення бізнес-процесів. Це правда частково тому, що нові системи відокремлюють взаємозв'язок між процесами, які передають дані та процеси, які використовують дані. Дані з багатьох джерел можуть бути передані в сучасну платформу даних і використовуватися різними споживачами майже відразу або пізніше, коли це необхідно. Саме тому для вирішення данної проблеми був створений проект Apache Kafka.

II. KAFKA: ПОТОКОВИЙ АРХІТЕКТУРИНИЙ ПІДХІД

Kafka — це розподілена потокова платформа, що має три основні можливості:

- публікувати та підпис на потоки записів, схожі на чергу повідомлень або систему корпоративних повідомлень;
- зберігайти потоки записів у відмовостійкому та тривалому режимі;
- обробляти записи, як тільки ті вони створюються.

Kafka найчастіше використовується для потокового передавання даних в режимі реального часу в інші системи. Kafka є середнім шаром, щоб відокремити ваші канали передачі даних у реальному часі. Ядро Kafka не підходить для прямих обчислень, таких як агрегації даних або Комплексної Обробки Подій [1]. Потокове передавання Kafka, що входить до складу екосистеми Kafka, дає змогу здійснювати аналізи в реальному часі. Вона може використовуватися для живлення швидких смугових систем (систем реального часу та операційних даних), таких як Storm, Flink, Spark Streaming, а також ваші послуги та системи Комплексної Обробки Подій. Kafka також використовується для потокового передавання даних для аналізу пакетних даних. Kafka має можливість віддавати дані на Hadoop. Він передає їх на велику платформу даних або в RDBMS, Cassandra, Spark або навіть S3 для деякого майбутнього аналізу даних. Ці сховища даних часто підтримують аналіз даних, звітність, стиснення даних, аудит відповідності та резервне копіювання.

Kafka використовує ZooKeeper для вибору головного брокера Kafka та порціювання розділів теми. Kafka використовує ZooKeeper, щоб керувати пошуком сервісів для брокерів Kafka, які утворюють кластер. ZooKeeper надсилає зміни до топології до Kafka, тому кожен вузол в кластері знає, коли приєднується новий брокер, брокер помирає, тема видаляється або додана тема, тощо. ZooKeeper забезпечує синхронізований перегляд конфігурації кластера Kafka [2].

Тема — це назва або категорія. Теми в Kafka завжди є мульти-абонентом - тобто тема може мати нуль, той, на який написано. Для кожної теми кластер Kafka підтримує журнал розділів (рис. 1)

Виробники Kafka пишуть дані на теми. Kafka споживачі читають з теми. Тема пов'язана з журналом, який є структурою даних на диску. Kafka додає записи від виробника (ів) до кінця журналу тем [3].

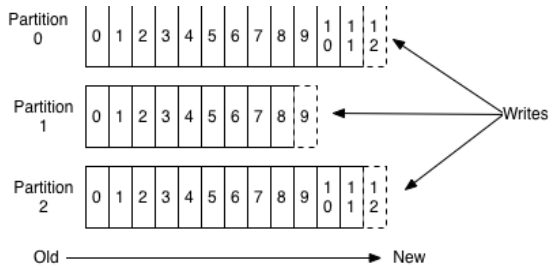

Рис.1. Журнали розділів для тем

Журнал тегів складається з багатьох розділів, які розповсюджуються на декілька файлів, які можуть бути розповсюджені на декількох вузлах кластера Kafka. [3] Споживачі читають з тематики Kafka у своїй каденції і можуть вибрати, де вони (зсув) у журналі тем. Кожна група споживачів відстежує, де вони перестали читати. Kafka поширює розділи журналу теми на різні вузли в кластері для високої продуктивності з горизонтальною масштабованістю. Розповсюдження розділів допомагає швидко записувати дані. Розділи журналу тематик — це шлях Kafka до відокремленого читання та запису до журналу тем. Крім того, потрібні розділи для одночасного роботи декількох споживачів групи споживачів (рис. 2). Kafka повторює розділи для багатьох вузлів, щоб забезпечити відмову.

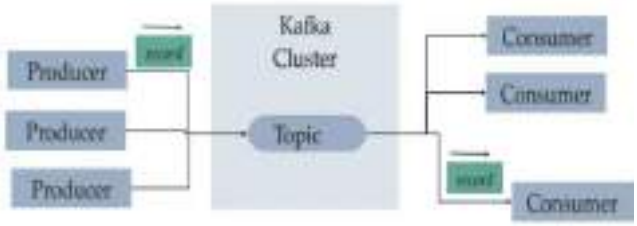

Рис.2. Схема взаємодії споживачів та продюсерів з кластером Kafka

III. ОСНОВНІ КОМПОНЕНТИ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ

A. Брокери Kafka

Група Kafka складається з декількох брокерів Kafka. Кожен Kafka Брокер має унікальний ідентифікаційний номер. Брокери Kafka містять розділи журналу тем. Підключення до одного брокера завантажує клієнта до всього кластера Kafka. Для аварійного завершення роботи необхідно розпочати з мінімум від трьох до п'яти брокерів. У кластері Kafka можуть бути 10, 100 або 1000 брокерів в кластері, якщо це необхідно.

B. Kafka Streams для обробки потоку

API Kafka Stream будується на основі ядра Kafka та має власний життєвий цикл [4]. Kafka Streams дозволяє обробляти потоки в реальному часі та підтримує потокові

процесори. Поточний процесор приймає постійні потоки записів з вхідних тем, виконує деяку обробку, перетворення, агрегацію на вхід, і виробляє один або декілька вихідних потоків. Наприклад, додаток для відеопрогравача може приймати потік вхідних подій переглянутих відеозаписів, призупинити відтворення відео та виводити потоки користувацьких уподобань, а потім передавати нові рекомендації щодо відео, виходячи з останніх дій користувачів або сукупної активності багатьох користувачів, щоб побачити, що нового відео гаряче. API Kafka Stream вирішує важкі проблеми із запитами поза обліковим записом, об'єднуючись у кількох потоках, об'єднуючи дані з декількох потоків, дозволяючи виконувати обчислення за станом [5].

C. Кластер Kafka, відмовостійкість, ISRs

Kafka підтримує реплікацію для підтримки відмови. Kafka використовує ZooKeeper для формування брокерів Kafka в кластері, і кожен вузол в кластері Kafka називається брокером Kafka. Тематичні розділи можуть бути відтворені в декількох вузлах для відмови від переходу. Тема повинна мати коефіцієнт реплікації, більший за 1. Наприклад, якщо ви працюєте в AWS, ви хотіли б, щоб передача працювала в разі затримки однієї зони доступності AWS. Якщо один Kafka Брокер перестає працювати, то інший, який є ISR (синхронізована репліка), може обслуговувати дані [6].

D. Kafka Failover та аварійне відновлення Kafka

Kafka використовує реплікацію (рис. 3) в разі відмови [7]. Тиражування розділів журналу тематики Kafka дозволяє відмовитися від стійки або зони доступності AWS (AZ). Потрібен коефіцієнт реплікації, щонайменше 3, для виживання однієї аварії AZ.

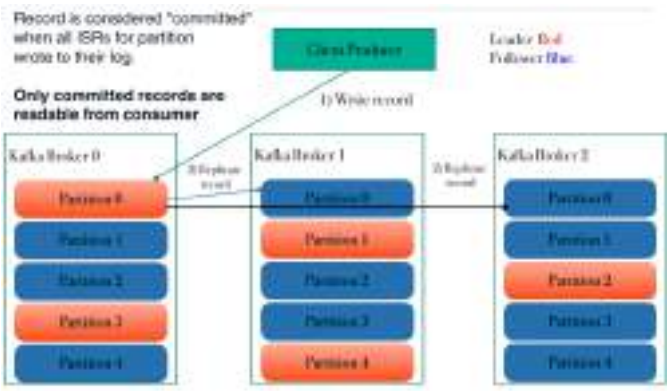

Рис.3. Схема реплікації данних

Для аварійного відновлення потрібно використовувати Mirror Maker - утиліту Kafka, котра поставляється з його ядром. Mirror Maker повторює кластер Kafka в інший центр обробки даних або регіон AWS. Mirror Maker відзеркалює, тому його не варто плутати з реплікацією.

Зауважте, що не існує жорсткого та швидкого правила щодо того, як потрібно самостійно налаштувати кластер Kafka. Наприклад, можна налаштувати весь кластер в одному AZ, щоб використовувати розширені налаштування мережі AWS і групи місць розташування для більш високої пропускної спроможності, а потім використовувати Mirror Maker, щоб відобразити кластер на іншому AZ у тому самому регіоні в режимі очікування.

IV. Висновок

Провівши огляд технології в рамках адаптації її для трансферингу BigData незалежно від їх доменної області, можна зробити висновок щодо релевантності Kafka, як брокера подій. По перше, Kafka послідовно записує файлові системи, що є швидко. Завдяки сучасному швидкому накопичувачу, Kafka може легко записати до 700 МБ або більше даних в секунду. По друге, Kafka є масштабуємою, пише і зчитує, розділяючи журнали тем у розділи. Згадані журнали тем можуть бути розділені на декілька розділів, які можна зберігати на декількох різних серверах, і ці сервери можуть використовувати кілька дисків. Кілька виробників можуть писати на різні розділи тієї ж теми. Кілька споживачів з декількох груп споживачів можуть ефективно читати з різних розділів. Це вирішує проблему BigData «V» — обсяг та швидкість.

Література References

[1] Jean-Paul Azar (2017). “What Is Kafka? - DZone Big Data” [Online]. Available: https://dzone.com/articles/what-is-kafka

[2] Jean-Paul Azar. (2017). “Kafka Architecture - DZone Big Data” [Online]. Available: https://dzone.com/articles/kafka-architecture

[3] Cloudurable. (2017). “Kafka Architecture: Consumers” [Online]. Available: http://cloudurable.com/blog/kafka-architecture-consumers/index.html

[4] Jean-Paul Azar (2017). “Kafka Detailed Design and Ecosystem - DZone Big Data” [Online]. Available: https://dzone.com/articles/kafka-detailed-design-and-ecosystem

[5] Stéphane Maarek (2017). “How to use Apache Kafka to transform a batch pipeline into a real-time one” [Online]. Available: https://medium.com/@stephane.maarek/how-to-use-apache-kafka-to-transform-a-batch-pipeline-into-a-real-time-one-831b48a6ad85

[6] Neha Narkhede (2015). “Hands-free Kafka Replication: A lesson in operational simplicity - Confluent” [Online]. Available: https://www.confluent.io/blog/hands-free-kafka-replication-a-lesson-in-operational-simplicity/

[7] Marcio Andrada (2017). “2-Way Replication With Apache Kafka _ LinkedIn” [Online]. Available: https://www.linkedin.com/pulse/2-way-replication-apache-kafka-marcio-andrada