When to choose Kafka versus traditional messaging technologies

Red Hat

Apache Kafka is well-known for its capability to retain, integrate, and stream large amounts of data with low overhead. However, Apache Kafka is designed with less consideration for the characteristics and uniqueness of a message. In contrast, Apache ActiveMQ treats each message as unique and prioritizes delivery rather than volume.

Executive summary

As integration approaches evolve, and the volume of data increases, IT leaders and architects need to pay close attention to the way messaging capabilities are designed in their organizations. Each integration approach comes with a set of tools and capabilities that determine which messaging technologies are best suited for specific use cases. Evaluating messaging technologies can become a tedious task. This e-book summarizes two distinct approaches: Apache Kafka for modern messaging solutions and Apache ActiveMQ for point-to-point messaging solutions.

Understanding message-oriented middleware

Messaging is a method of communication between applications. A messaging system takes care of sending, receiving, and processing messages. There are various messaging system approaches for application to application communication, such as Remote Procedure Call (RPC), object request broker (ORB), and message-oriented middleware (MOM). The focus of this article is on MOM, which spotlights solutions that require a message broker to manage communication between applications.

One of the most popular traditional MOM solutions on the market is Active Message Queuing, better known as ActiveMQ or AMQ. This open source message broker solution was introduced nearly 20 years ago, and is currently part of the Apache community. It offers transaction support for Java™ Message Service (JMS) and extended architecture (XA) transactions, allowing the support of a variety of use cases.

With the surge of technologies like containers, container orchestration, nonstructural databases, microservices architectures, and stream data processing, a new catalog of technologies and architectural patterns has emerged. Apache Kafka is among these new technologies, and provides faster processing, larger scalability, high availability and improved fault tolerance.

Apache Kafka is well-known for its capability to retain, integrate, and stream large amounts of data with low overhead. However, Apache Kafka is designed with less consideration for the characteristics and uniqueness of a message. In contrast, ActiveMQ treats each message as unique and prioritizes delivery rather than volume.

What is Apache Kafka and why is it valuable?

Apache Kafka is an asynchronous messaging service that uses the publish-subscribe method to move data between microservices, cloud-native or traditional applications, and other systems. The publish-subscribe messaging pattern is mostly used when you need an event or a message to trigger multiple actions within an architecture or organization.

Apache Kafka differentiates itself from other asynchronous messaging solutions because of its ability to deliver data in near-real time, stream vast amounts of data, and replay data in the order it was received. Because of its capabilities, Apache Kafka is sometimes considered more than a messaging solution, as it also supports stream processing and data integration.

The technical benefits of Apache Kafka

Apache Kafka product technical benefits are difficult to synthesize on a short list, but they can bring clarity when deciding about which technologies best suit the requirements and needs within an organization.

- **Record storage with fault tolerance**
  Publish-subscribe messaging solutions like Apache Kafka allow for storing data into brokers from where consumers and producers can access the records or messages. Apache Kafka's architecture allows brokers to be replicated into different machines for insync replica configurations. This configuration helps protect against failure at the hardware or application level, since all brokers would be available on their replicas. You can automatically replace a failed component with its replica and ensure no loss of service, which is essential for high availability.

- **Horizontal scalability**
  Each broker in the Apache Kafka architecture can contain multiple topics, and each topic can be broken into multiple partitions. This configuration allows consumers to access topics and multiple partitions in parallel, which improves scalability. Horizontally scaling requires you to add more brokers to an existing Apache Kafka cluster. The cluster will require more nodes to balance out the load, and ultimately allow the cluster to serve more requests. Consumers don't need to wait for messages to be delivered, instead accessing messages as soon as they are recorded on a topic.

- **Processing streams in real time**
  Because Apache Kafka does not require the acknowledgement of messages received or recorded, message writing can occur at a very high rate, thus functioning like real-time message delivery. Message acknowledgement and the rate for receiving messages can be configured to ensure that messages are not creating buffers, data is not lost in transit, or messages are not recorded in the wrong order. This ability allows organizations to move large volumes of messaging data, mitigating latency and bandwidth limitations.

- **Log compaction and data retention**
  Apache Kafka features retention and log policies that differentiate it from other messaging technologies. Both features allow Apache Kafka to retain messages even after they have been received by the intended consumer. Message retention is configurable per broker, allowing for messages to be stored for longer periods of times. Apache Kafka also allows messages to be written in the order they were received by the broker. This capability allows Apache Kafka to process and reprocess data many times, replying via a series of streams when required.

What is ActiveMQ and why is it valuable?

ActiveMQ is a message-oriented middleware that supports asynchronous communication between client and server. This technology is known as a traditional messaging solution. ActiveMQ is a broker that delivers point-to-point messages via queues and publish-subscribe messages via topics.

As a traditional messaging solution, ActiveMQ supports message delivery. Messages are stored until received and acknowledged before they can be deleted from the queue or topic. The broker plays a central role in distributing and receiving messages as it contains the logic for retention, persistence, and delivery.

The technical benefits of ActiveMQ

The technical benefits of implementing ActiveMQ message brokers are well known in the industry, but sometimes understated, and include:

- **High performance broker providing use case flexibility.**
  ActiveMQ was designed as message-oriented middleware with an intelligent broker. This broker takes charge of scalability, partitioning, retention, persistence, and processing, which supports and handles complex message distribution on behalf of the clients, providing a simple setup for a broad range of messaging use cases.
- **Multiple messaging, protocols, and application programming interfaces (APIs).**
  ActiveMQ brokers support anycast and multicast messaging scenarios, multiple protocol plugins, and many clients. ActiveMQ broker is among the most multifunctional messaging solutions available for supporting multiple use cases.
- **High availability and flexible message persistence.**
  ActiveMQ's flexible message persistence and disk storage play a key role in this solution's high availability. ActiveMQ allows the creation of a source-replica scheme for storing messages, an architecture that reacts to failures when clients lose connection to a broker. On the other side, disk storage defines the overall broker performance. ActiveMQ also supports message persistence, which can increase performance and minimize the disk storage used.

Apache Kafka or ActiveMQ?

When deciding on the best messaging technology, the decision should be informed by the organization's use case, because every messaging solution will not serve the organization's requirements and use cases the same way. There are several technical features to consider when choosing a messaging solution, such as message persistence, data volume, and broker performance. When combined correctly, these elements can provide the balance required for high availability, scalability, and throughput, regardless of the type of messaging solution you choose.

Apache Kafka has been presented many times as a significant change in the industry, sometimes called a next-generation messaging solution, delivering high throughput and availability. This technology excels at the ability to ingest, log, process, and move vast amounts of data. These characteristics make Apache Kafka a great option for the following three use cases.

Real-time streams processing scenarios

Apache Kafka is a great candidate for use cases that require data to be processed as soon as it becomes available. Scenarios that require real-time data processing streams include cybersecurity, predictive maintenance, Internet of Things (IoT)-like solutions, and financial IT systems monitoring stock data.

Application activity tracking

One area where Apache Kafka excels is in its capacity for ingesting, logging, processing, and moving vast amounts of data. Apache Kafka can create large streaming pipelines for circulating data from multiple sources at high performance and availability. Real-world use cases include streaming data between data lakes functioning as an ingestion pipeline for data platforms, website activity tracking, and cluster or systems monitoring.

Microservices communication

Apache Kafka can easily support or behave as the broker system between multiple microservices. Therefore, if an organization is thinking of modernizing existing applications or creating new applications using microservices, then using Apache Kafka as the broker would be the optimal decision. For example, the Apache Kafka ecosystem can provide connectors to legacy systems, open source platforms, analytics engines, and search engines solutions. Its integration capabilities also help solve the challenges with microservices orchestration such as data synchronization, security, and logging.

In contrast, ActiveMQ is a robust traditional messaging solution that provides you with the ability to treat messages as unique entities, prioritizing the message header, content, and delivery.

Transactional messaging

ActiveMQ is well suited for complex IT infrastructures that require using granular messaging configuration to send messages from one system to many, such as transactional messaging. ActiveMQ is especially useful when sending information that is highly sensitive, making it a suitable choice for a variety of use cases in financial services and banking.

General purpose asynchronous messaging

Another great use case for ActiveMQ is when implementing web- or Java-based style services. ActiveMQ allows you to target general purpose asynchronous messaging, which generally works for use cases that only transmit a small number of messages per day. The low volume allows for local message storage until delivery, providing unique support per message and ensuring messages are delivered in the order they were sent.

Finally, ActiveMQ supports simpler development and maintenance when compared with Apache Kafka. Apache Kafka technology offered as managed or Software-as-a-Service (SaaS) solutions provides a lower barrier for entry, development, and long-term maintenance compared to do-it-yourself (DIY) solutions.

Key takeaways

Newer messaging technologies are key for accelerating application modernization either through a migration approach or through new development. Technologies like Apache Kafka open up a new range of applications that otherwise would have been difficult to develop or implement a few years ago.

Apache Kafka can help with ingesting, logging, and streaming vast amounts of data in near-real time—something that is becoming increasingly important in a world where data is becoming an increasingly valuable but unwieldy resource. Collecting and streaming data for further analysis by intelligent applications is a common use case seen in the industry. Apache Kafka can become a fundamental building block in an artificial intelligence and machine learning (AI/ML) platform by serving as a connection to multiple data sources.

As with any technology, there is a catch. Designing the right configuration for your Apache Kafka implementation can be complex, as it highly affects the deployment architecture in production. Things like proxy configurations, number of partitions, and load balancing are deciding factors on how well an Apache Kafka instance performs. The complexity of Apache Kafka could challenge teams as they dedicate resources to configure, manage, and maintain the infrastructure. Handing over some of the complexities to experts can be of great help as IT teams and developers build their messaging architecture, which makes managed Apache Kafka instances a great choice for many organizations.

Messaging solutions at Red Hat

Red Hat provides a complete technology portfolio, proven expertise, and in-house experts to help you achieve your messaging and connectivity goals. Our messaging portfolio provides a comprehensive set of integration and messaging technologies to connect applications and data across hybrid cloud environments.

Red Hat AMQ broker

Red Hat® AMQ broker is a pure-Java multiprotocol message broker with persistence and advanced high-availability modes. Based on the Apache ActiveMQ project, this broker allows information to be delivered reliably.

Red Hat AMQ streams

Red Hat AMQ streams is a self-managed massively scalable, distributed, and high-performance data streaming platform based on the Apache Kafka project. It offers a distributed backbone that allows microservices and other applications to share data with high throughput and low latency, allowing real-time integration and connecting the IoT.

Learn more about Red Hat messaging solutions

Visit Red Hat Application Foundations for more information on Red Hat messaging solutions.