Sistemas distribuidos

Kafka vs RabbitMQ

Alumnos:
Bejamin Alonso - Juan Muñoz

24-5-2023

Índice

1. Resumen:

En este informe, se realizará una comparación exhaustiva entre Kafka y RabbitMQ, centrándose en su arquitectura, características clave, siendo como estas afectan a las características más importantes a la manera de ser usadas en la practica, basándose en nuestro propio buildeo de estos dispositivos y los contenidos pre-vistos en clase.

2. Introducción

:

Kafka y RabbitMQ son dos populares sistemas de mensajería utilizados en el ámbito de la tecnología de la información. Ambas plataformas desempeñan un papel fundamental en el manejo y procesamiento de datos en tiempo real, pero tienen enfoques arquitectónicos diferentes.

Kafka se basa en un modelo de publicación/suscripción, donde los mensajes son enviados por productores y luego consumidos por uno o más suscriptores interesados en recibir esos mensajes. Funciona como un intermediario confiable entre los productores y consumidores, garantizando la entrega de mensajes de manera eficiente y en orden. Es especialmente adecuado para escenarios en los que se manejan grandes volúmenes de datos y se requiere un alto rendimiento, como en la transmisión de eventos en tiempo real.

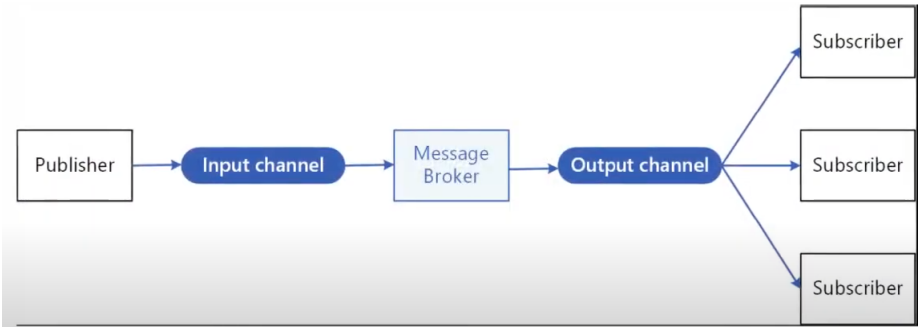

Figura 1: Diagrama de Kafka

Por otro lado, RabbitMQ se basa en un modelo de cola de mensajes. En este enfoque, los productores envían mensajes a una o varias colas de RabbitMQ, que luego los entrega a los consumidores según diferentes estrategias de enrutamiento y balanceo de carga. RabbitMQ se centra en la entrega confiable de mensajes, asegurando que los datos sean transmitidos de manera segura y sin pérdidas. Es especialmente útil

en aplicaciones donde se requiere una alta fiabilidad en la entrega de mensajes, como en sistemas de procesamiento de transacciones o comunicaciones asincrónicas entre diferentes componentes de un sistema.

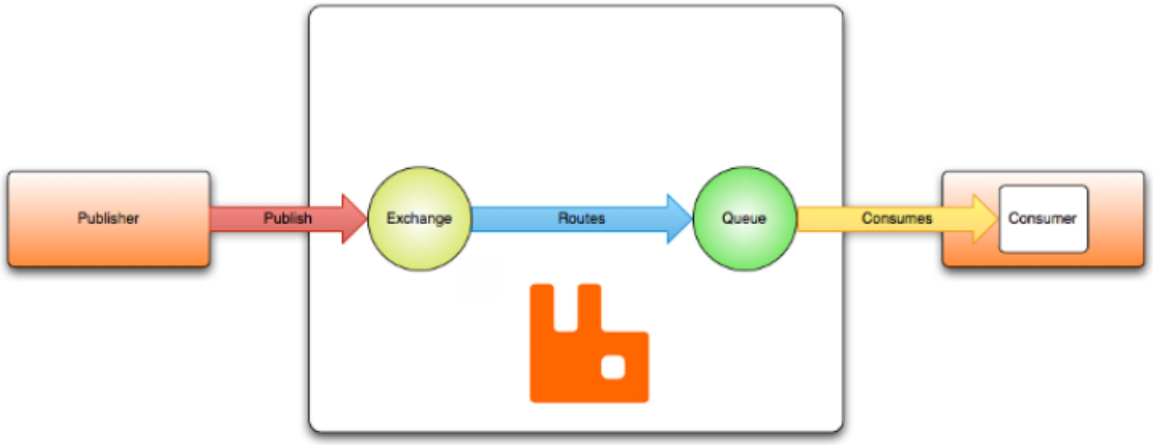

Figura 2: Diagrama de RabbitMQ

3. Planificación del sistema

En un mundo cada vez más conectado, la empresa multinacional UDPrecios se ha establecido como líder en el mercado de servicios y productos de consumo. Con su enfoque innovador y orientado al cliente, UDPrecios ha logrado una expansión significativa en todo el mundo. Sin embargo, con el crecimiento de su negocio, la empresa se ha enfrentado a desafíos en la gestión eficiente de sus recursos informáticos.

UDPrecios opera una infraestructura de red global con numerosos servidores que requieren un monitoreo constante del rendimiento y la disponibilidad. Además, la empresa ha experimentado un aumento sustancial en las ventas en línea a través de su plataforma de comercio electrónico, donde los clientes realizan pedidos en tiempo real. La creciente demanda de sus servicios y productos ha llevado a UDPrecios a considerar una expansión en sus recursos informáticos y a la necesidad de establecer dos canales de transmisión de datos separados para satisfacer sus requerimientos específicos.

El primer canal de transmisión de datos está destinado a monitorear el rendimiento de los servidores en su infraestructura de red. UDPrecios reconoce la importancia de recopilar datos en tiempo real sobre el uso de recursos, la carga de trabajo y el rendimiento de los servidores para garantizar una experiencia óptima para sus

clientes. Este canal permitirá la captura continua de información crítica y la entrega confiable de los datos a los equipos de administración y soporte técnico de UDPrecios.

El segundo canal de transmisión de datos es esencial para la plataforma de comercio electrónico de UDPrecios, donde los clientes realizan pedidos en línea. Con una base de clientes en constante crecimiento, UDPrecios necesita una solución que garantice una comunicación fluida y confiable entre los diferentes componentes del sistema de procesamiento de pedidos. Este canal será responsable de transmitir eventos de pedidos en tiempo real, desde el momento en que se realiza una compra hasta su procesamiento y envío.

En este escenario, UDPrecios se encuentra en una encrucijada y debe tomar una decisión estratégica: ¿utilizar Kafka o RabbitMQ en uno o en ambos canales de transmisión de datos? La elección adecuada determinará la eficiencia, la escalabilidad y la confiabilidad de la infraestructura tecnológica de UDPrecios.

El equipo de tecnología de UDPrecios se ha embarcado en un proceso exhaustivo de evaluación y comparación entre Kafka y RabbitMQ. Analizarán aspectos clave como la arquitectura, la escalabilidad, la durabilidad y la entrega confiable de mensajes de ambas plataformas. Considerarán cómo cada una se adapta a las necesidades específicas de UDPrecios en términos de monitoreo del rendimiento de los servidores y procesamiento de pedidos en línea.

La elección correcta no solo garantizará la eficiencia y la calidad de los servicios de UDPrecios, sino que también sentará las bases para un crecimiento futuro sin problemas. Con tanto en juego, UDPrecios está decidida a tomar la decisión correcta para asegurar su posición como líder en el mercado global y ofrecer una experiencia excepcional a sus clientes. La competencia está en aumento y UDPrecios está lista para enfrentar este desafío al elegir la solución de transmisión de datos adecuada.

4. Análisis de datos

4.0.1. Escalabilidad

Kafka está diseñado para la escalabilidad horizontal desde el principio. Puede manejar grandes volúmenes de datos y soportar altas tasas de transferencia de mensajes al distribuir particiones en múltiples nodos (en nuestro caso, el aumento de hilos) de Kafka. Esto permite agregar más nodos y particiones según sea necesario para escalar horizontalmente y aumentar la capacidad de procesamiento y almacenamiento.

Por otro lado, RabbitMQ también se puede escalar horizontalmente, pero su enfoque es más tradicional en comparación con Kafka. La escalabilidad horizontal en RabbitMQ se logra mediante la configuración de clústeres, donde múltiples nodos

RabbitMQ se agrupan para distribuir la carga. Sin embargo, la escalabilidad de RabbitMQ puede ser más complicada de configurar y administrar en comparación con Kafka.

Tanto Kafka como RabbitMQ ofrecen escalabilidad, pero Kafka está especialmente diseñado para manejar grandes volúmenes de datos y proporcionar una alta capacidad de transferencia de mensajes. Kafka es conocido por su rendimiento y escalabilidad horizontal.

4.0.2. Tolerancia al fallo

Kafka es conocido por su alta disponibilidad y tolerancia a fallos. Está diseñado para ser altamente resistente y proporcionar una replicación de datos confiable y distribuida. Kafka logra esto a través de poder distribuir sus recursos y su enfoque de registros inmutable. Los datos se almacenan en registros de forma secuencial, lo que permite la replicación y la recuperación de fallos en caso de que un nodo falle. Kafka también permite la configuración de un factor de replicación para garantizar la disponibilidad de datos incluso en caso de fallo de un nodo.

RabbitMQ es considerado como un bróker de mensajería más tradicional. Se basa en el patrón publicador-subscriptor, aunque puede tratar la comunicación de manera síncrona o asíncrona, según se establezca en la configuración. También garantiza la entrega y el orden de los mensajes entre los productores y los consumidores. Es adecuado para muchos protocolos de mensajería, flexibilidad y plugins disponibles, que puede lograr ser conectado con herramientas de varios desarrolladores.

Por esto, Kafka tiene un rendimiento y escalabilidad superior a RabbitMQ, por lo que es mucho más usado en proyectos en los que se desea una alta tolerancia a fallos y procesamiento en tiempo real. RabbitMQ es un sistema más flexible y con un enrutado de mensajes más completo.

4.0.3. Latencia y rendimiento

Kafka proporciona la latencia más baja con rendimientos más altos, al tiempo que proporciona una gran durabilidad y alta disponibilidad. RabbitMQ puede lograr una latencia de extremo a extremo más baja que Kafka, pero solo con rendimientos significativamente más bajos (30 000 mensajes/seg. frente a 200 000 mensajes/seg. para Kafka), después de lo cual su latencia se degrada significativamente. [**1**]

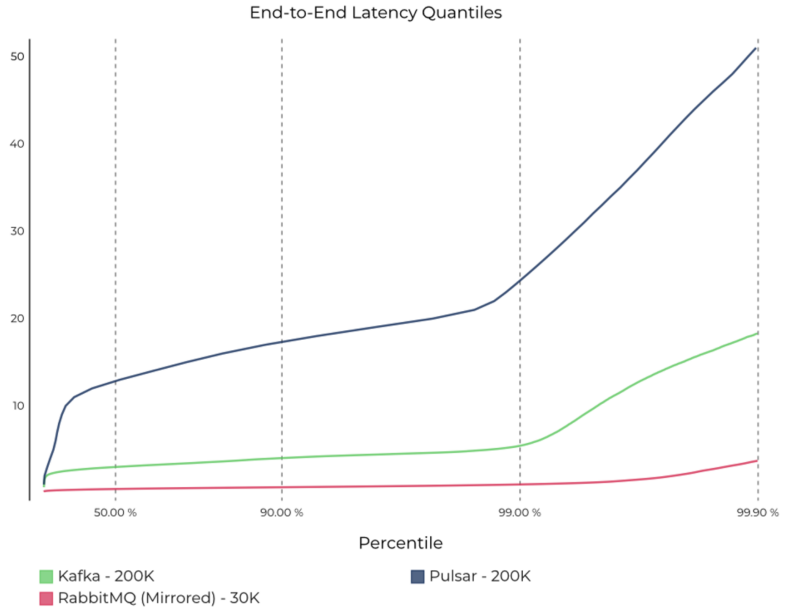

Figura 3: Kafka vs Pulsar vs RabbitMQ - Latency

4.0.4. Persistencia de mensajes

Apache Kafka:

1. Almacenamiento persistente: Kafka almacena los mensajes de manera persistente en su sistema de archivos, lo que permite que los mensajes se retengan incluso después de que hayan sido consumidos. Esto asegura que los mensajes estén disponibles para su recuperación y procesamiento posterior.

2. Retención de mensajes: Kafka permite configurar un tiempo de retención para los mensajes en los temas (topics). Esto significa que los mensajes se mantendrán durante un período específico de tiempo, independientemente de si han sido consumidos o no. Esta característica facilita la recuperación de mensajes históricos y la construcción de sistemas de eventos basados en el tiempo.

3. Replicación y tolerancia a fallos: Kafka ofrece replicación de datos en clústeres para garantizar la disponibilidad y durabilidad de los mensajes. Los datos se replican en múltiples nodos, lo que proporciona tolerancia a fallos y alta disponibilidad. En caso de que un nodo falle, los mensajes aún estarán disponibles en los nodos restantes.

4. Commit logs: Kafka utiliza un sistema de registros de confirmación (commit logs) para almacenar los mensajes. Esto permite una recuperación rápida y eficiente en caso de fallos o reinicios del sistema.

RabbitMQ:

1. Almacenamiento persistente: RabbitMQ también ofrece almacenamiento persistente de mensajes, lo que significa que los mensajes se guardan en el disco y se conservan incluso después de un reinicio del servidor. Esto asegura que los mensajes no se pierdan y puedan ser recuperados más tarde.
2. Colas duraderas: RabbitMQ permite la creación de colas duraderas, lo que significa que las colas y los mensajes en ellas sobreviven a reinicios del servidor. Esto garantiza que los mensajes se conserven y puedan ser recuperados después de una interrupción del sistema.
3. Acknowledgements (confirmaciones): RabbitMQ proporciona un mecanismo de confirmaciones para garantizar la entrega segura de mensajes. Los consumidores pueden confirmar la recepción de los mensajes, lo que asegura que no se pierdan en caso de fallas en el procesamiento.
4. Exchange y enrutamiento: RabbitMQ ofrece un sistema de intercambio (exchange) y enrutamiento de mensajes, lo que permite dirigir los mensajes a colas específicas según ciertos criterios. Esto brinda flexibilidad en la recuperación de mensajes según las necesidades del sistema.

4.0.5. Facilidad de uso y administración

Tanto a nivel de búsqueda de internet, documentación y material proporcionado por la universidad, Kafka cuenta con mayor material de apoyo para su creación y ejecución a comparación de RabbitMQ. Esto puede llegar a ser un ventaja al momento de escogerlo como un dispositivo para 'Empezar', pero RabbitMQ tiene la ventaja que es más flexible, con capacidad de agregar plugins y ser compatible con más protocolos.

5. Análisis de la experiencia

En resumen, al comparar Kafka y RabbitMQ, se puede concluir que Kafka presenta varias ventajas y es más fácil de implementar, mientras que RabbitMQ puede considerarse una opción más compleja pero también más flexible debido a las herramientas que ofrece.

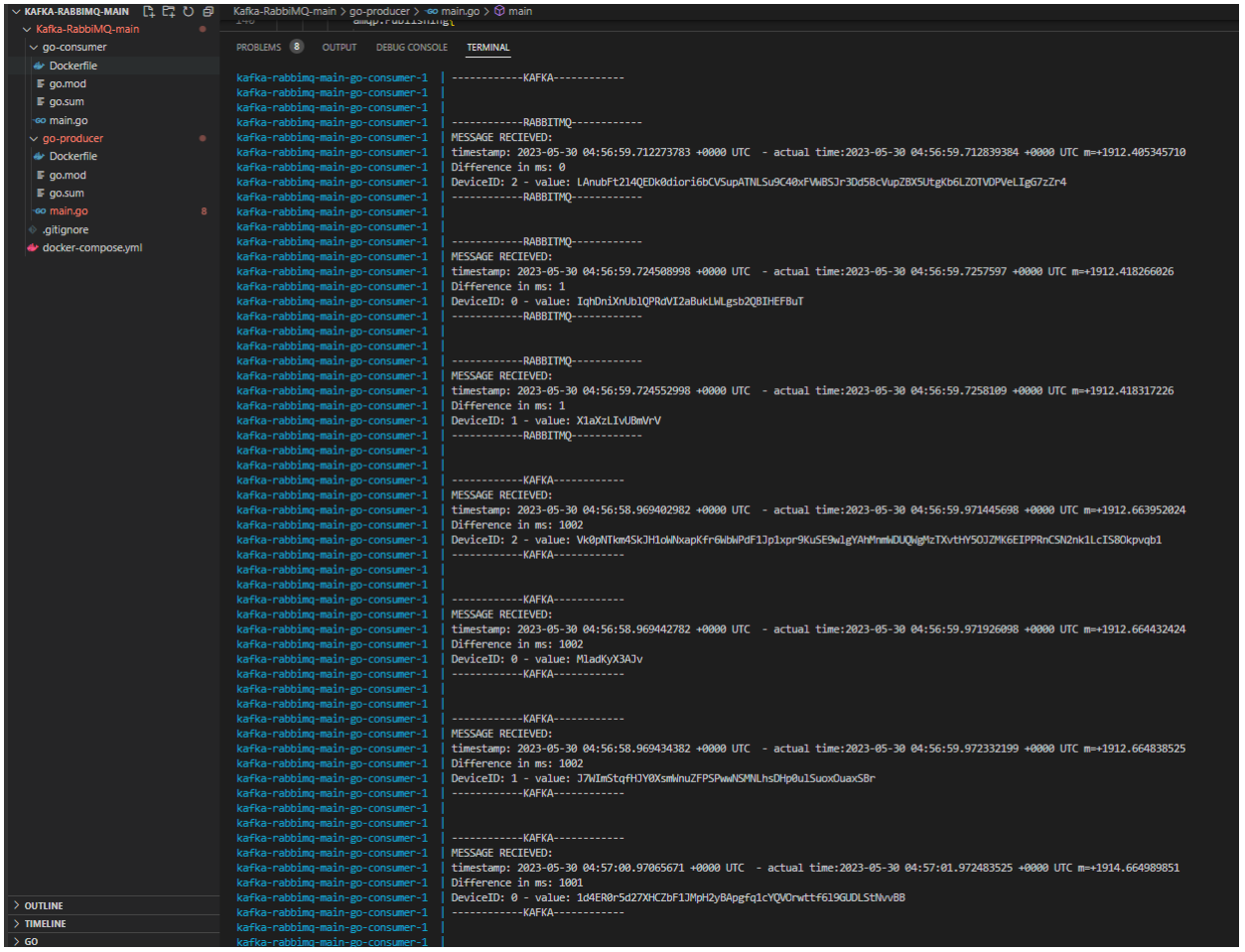

Figura 4: Mensajes

Volviendo a nuestro programa, la decisión queda de la siguiente forma:

la plataforma de comercio electrónico donde los clientes realizan pedidos en línea, para garantizar una comunicación fluida y confiable entre los diferentes componentes del sistema, puedes implementar RabbitMQ de la siguiente manera:

1. Productores: Cuando los clientes realizan pedidos, se generan eventos que contienen detalles del pedido, como el artículo comprado, la cantidad y la dirección de envío. Estos eventos se envían a una cola en RabbitMQ.

2. Consumidores: Puedes tener diferentes consumidores que procesen estos eventos de pedido. Por ejemplo, uno de los consumidores podría encargarse de verificar el inventario para asegurarse de que hay suficiente stock disponible. Otro consumidor podría ser responsable de generar la factura y enviarla al

cliente por correo electrónico.

3. RabbitMQ: Actúa como intermediario de mensajes, encolando y entregando los eventos de pedido a los consumidores correspondientes. RabbitMQ garantiza la entrega confiable de mensajes y permite una comunicación asincrónica entre los diferentes componentes del sistema.

Por otro lado, el sistema de análisis de datos en tiempo real para monitorear el rendimiento de servidores de la empresa, ya que los servidores generan constantemente datos sobre el uso de recursos, la carga de trabajo y el rendimiento puede ser mayor, por eso acá Kafka puede desempeñar un papel fundamental.

1. Productores: Cada servidor puede actuar como un productor de datos, enviando mensajes al tema correspondiente en Kafka. Estos mensajes pueden incluir información como el uso de la CPU, la memoria, las solicitudes entrantes, etc.
2. Consumidores: Puedes tener varios consumidores interesados en analizar y procesar esos datos. Por ejemplo, uno de los consumidores podría generar métricas en tiempo real y enviar alertas si se detectan anomalías en el rendimiento. Otro consumidor podría almacenar los datos en una base de datos para su posterior análisis.
3. Kafka: Actúa como un intermediario de mensajería, almacenando y entregando los mensajes de manera eficiente. Kafka permite una distribución paralela de mensajes y garantiza la tolerancia a fallos y la escalabilidad.

6. Referencias

1. Kafka vs Pulsar - Performance, Features, and Architecture Compared — DE. (s. f.). Confluent. https://www.confluent.io/de-de/kafka-vs-pulsar/
2. When to use RabbitMQ or Apache Kafka - CloudAMQP. (2022, 13 diciembre). CloudAMQP. https://www.cloudamqp.com/blog/when-to-use-rabbitmq-or-apache-kafka.html