PRÁCTICA FUENTES DE DATOS: APACHE KAFKA

Lucian Iacob

Lab. 4 images

- ### Replication

```
smartedge00 :: ~/SW/kafka_2.13-2.8.0 » bin/kafka-topics.sh --describe --zookeeper 192.168.0.2:2181 --topic my-replicated3-topic1
Topic: my-replicated3-topic1    TopicId: 4N9jZ9iUQ9GI16WJAMaqPw PartitionCount: 1       ReplicationFactor: 3    Configs:
        Topic: my-replicated3-topic1    Partition: 0    Leader: 1       Replicas: 1,2,0 Isr: 1,2,0
smartedge00 :: ~/SW/kafka_2.13-2.8.0 » bin/kafka-topics.sh --describe --zookeeper 192.168.0.2:2181 --topic my-replicated3-topic3
Topic: my-replicated3-topic3    TopicId: oTX4-I-bSMuGDaDToCZnUw PartitionCount: 3       ReplicationFactor: 3    Configs:
        Topic: my-replicated3-topic3    Partition: 0    Leader: 1       Replicas: 1,2,0 Isr: 1,2,0
        Topic: my-replicated3-topic3    Partition: 1    Leader: 2       Replicas: 2,0,1 Isr: 2,0,1
        Topic: my-replicated3-topic3    Partition: 2    Leader: 0       Replicas: 0,1,2 Isr: 0,1,2
smartedge00 :: ~/SW/kafka_2.13-2.8.0 » bin/kafka-topics.sh --describe --zookeeper 192.168.0.2:2181 --topic my-replicated1-topic3
Topic: my-replicated1-topic3    TopicId: u1kaCqNMTN-hhhMl7pO4Hg PartitionCount: 3       ReplicationFactor: 1    Configs:
        Topic: my-replicated1-topic3    Partition: 0    Leader: 0       Replicas: 0     Isr: 0
        Topic: my-replicated1-topic3    Partition: 1    Leader: 1       Replicas: 1     Isr: 1
        Topic: my-replicated1-topic3    Partition: 2    Leader: 2       Replicas: 2     Isr: 2
smartedge00 :: ~/SW/kafka_2.13-2.8.0 »
```

- ### Fault Tolerance

```
smartedge00 :: ~/SW/kafka_2.13-2.8.0 »
[1]  + 5157 exit 143   nohup bin/kafka-server-start.sh config/server1.properties > /dev/null 2>&1
smartedge00 :: ~/SW/kafka_2.13-2.8.0 » bin/kafka-topics.sh --describe --zookeeper 192.168.0.2:2181 --topic fault-tolerance
Topic: fault-tolerance  TopicId: Ivx458khQ4Oey1D4JAnYIg PartitionCount: 2       ReplicationFactor: 2    Configs:
        Topic: fault-tolerance  Partition: 0    Leader: 0       Replicas: 1,0   Isr: 0
        Topic: fault-tolerance  Partition: 1    Leader: 0       Replicas: 0,1   Isr: 0
smartedge00 :: ~/SW/kafka_2.13-2.8.0 » bin/kafka-console-consumer.sh --bootstrap-server localhost:9092, localhost:9093 --from-beginning --
topic fault-tolerance
XX
YY
^CProcessed a total of 2 messages
smartedge00 :: ~/SW/kafka_2.13-2.8.0 »                                                                                    130 ↵
```

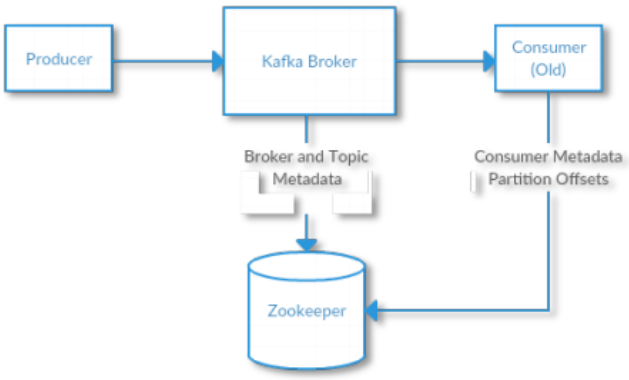
Questions

1 What is a broker, consumer and producer?

El broker es el corazón de un sistema Kafka, son pipes que van desde los producers hasta los consumers acumulando y distribuyendo los datos. Es lo que actúa como controlador del sistema.

Un producer es la aplicación encargada de publicar/escribir mensajes en Kafka mientras que el consumer es el encargado de suscribirse a esos eventos para leerlos y procesarlos. Lo esencial es que estos dos servicios están desacoplados, no dependen el uno del otro.

Para hacernos una idea de cómo funcionan estos componentes en conjunto adjuntamos esta imagen de los apuntes:

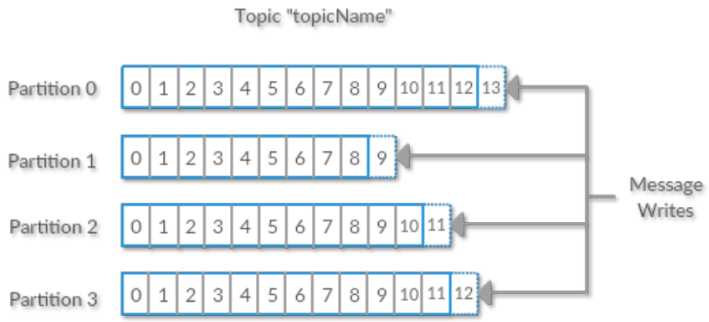


The Definitive Guide, 1st Edition by Neha Narkhede; Gwen Shapira; Todd Palino.
Published by O'Reilly Media, Inc.

2 Which sentences are the right one:

A partition is split by topics or a topic is split by partitions?

Lo correcto es lo segundo “a topic is split by partitions”. En Kafka tenemos un topic del tema que sea “web.logs”, “network twitter” por ejemplo y cada uno de estos se dividirá en particiones a las que irán los mensajes según la clave que lleven.



The Definitive Guide, 1st Edition by Neha Narkhede; Gwen Shapira; Todd Palino.
Published by O'Reilly Media, Inc.

Kafka preserves the order of messages within a partition or Kafka preserves the order of messages within a topic?

Lo correcto es lo primero “*Kafka preserves the order of messages within a partition”. Esto lo podemos ver en la propia definición de partition que se da en la documentación:*

“Each partition is an ordered, immutable sequence of records that is continually appended to — a structured commit log.”

Se puede ver en la figura superior también cómo la ordenación va a nivel de partición. En caso de necesitar todos los mensajes de un topic ordenados, lo que necesitamos es usar una sola partición.

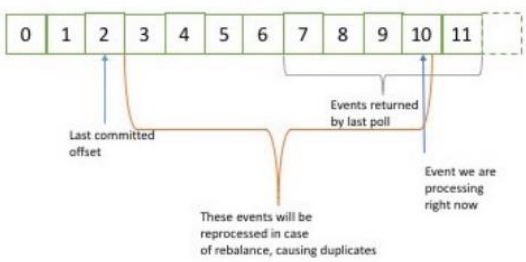
3. If a rebalancing happens at consumers side, what two situations do we have to take in mind? Could we avoid it? Could the performance be impacted? Why?

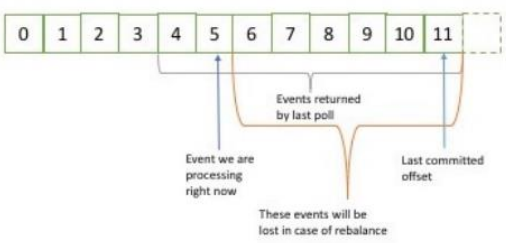
Hay 2 casos posibles según el tamaño del offset (trackeo de la posición de los consumers, es decir, hasta dónde ha leído).

- El offset commiteado es más pequeño que el offset del último mensaje que ha procesado el cliente. En este caso, los mensajes entre el último offset commiteado y el último procesado por el cliente serán procesados dos veces (Izquierda).

- El offset commiteado es más grande que el offset del último mensaje ha procesado el cliente. En este caso, los mensajes de entre medias serán saltados por el cliente y no se procesarán (Derecha).

Por commitear queremos decir actualizar la posición actual dentro de una partición.





The Definitive Guide, 1st Edition by Neha Narkhede; Gwen Shapira; Todd Palino.
Published by O'Reilly Media, Inc.

La forma de solucionar estos problemas es dejando de lado los auto commits por defecto y pasando a los commiteos síncronos, en los cuales se produce un bloqueo hasta que se confirme el procesamiento del mensaje. Esto claramente afecta al rendimiento pues para un hilo de ejecución hasta que se confirme la llegada, por ese hilo no se mueve nada hasta entonces.

4. Provide a brief description of Kafka Connect, Kafka Stream, KSQL and Schema-Registry. Provides a use case and why it would be a good choice incorporate it into the final solution for each one.

- Kafka Connect es una herramienta destinada a la comunicación entre Apache Kafka y otros sistemas de datos. Es un entorno de trabajo a gran escala que permite integración de los datos en tiempo real, simplificando la adopción de conectores para el traspaso de datos. Provee una gran cantidad de conectores open source y permite la creación de conectores privados.

- Kafka Stream es una librería de Java que actúa como una capa inmediatamente superior a Kafka orientada al procesamiento de datos en streaming consumiendo y produciendo datos en los topics de Kafka. Permite realizar filtrado, mapeos, agregaciones e incluso joins en operaciones con o sin estado mediante una API de alto nivel o incluso la creación de topologías siguiendo un comportamiento similar a Apache Storm.

- KSQL es simplemente una capa de abstracción que aporta una interfaz SQL para procesar datos en SQL evitando escribir scripts para obtención de información de los datos.

- Es un proceso standalone externo a los Kafka brokers cuyo trabajo es mantener una base de datos con el histórico de los esquemas que han ido teniendo y tienen los mensajes. Esta bbdd es persistida dentro de un topic de Kafka. Nos soluciona problemas de evolución de los esquemas de los mensajes conforme pasa el tiempo y evoluciona el cliente (empresa) que esté usando Kafka. Si tenemos un mensaje viejo y queremos entender su estructura, procedemos a mirar el histórico de esquemas gracias a Schema Registry.

Un ejemplo de todo en conjunto sería el deseo de monitorizar los logs de cómo se conectan los distintos usuarios a una aplicación que tengo levantada en la empresa (una de gestión de presupuestos en los distintos departamentos por dar un ejemplo) mediante Elastic Search. Para ello necesitaré de un conector proveído por Kafka Connect. Si además de ello, deseo hacer consultas sobre la cantidad de usuarios conectados del departamento X, o el número de usuarios conectados entre las 9:00 y las 15:00, necesitaría de un servicio como KSQL que me permita hacerlo. Como todos estos datos vienen en tiempo real se necesitaría un servicio como Kafka Streams.

Si los logs van cambiando con el tiempo al surgir nuevas capacidades en la aplicación de gestión de presupuestos (añadir varios pasos de autenticación y que surja un campo nuevo en el log que diga dónde ha fallado la autenticación) o por cambios a nivel administrativo en la empresa (surgen nuevos departamentos que tengan que conectarse o por el estilo), necesitaremos guardar el esquema de los logs que van surgiendo y por lo tanto necesitaremos también del servicio de Schema Registry.