Introducción a kafka stream

Kafka stream es como un consumidor con muchas funciones para el procesamiento de datos, como transformación, enriquecimiento, agregación, etc.
Al mismo tiempo, es un productor que publica datos en otro topic para que el consumidor final pueda consumir los datos transformados directamente desde el topic de salida.

- Kafka stream es una biblioteca de procesamiento de flujos para realizar transformaciones dentro del propio Kafka.
- Kafka stream consume y transforma cada dato, y luego lo publica en otro topic.
- Salió a la venta en 2017.
- Es una buena alternativa a los marcos de procesamiento de flujos como Spark o Flink.

Procesamiento de streams

¿Qué es un Stream?

- Un stream o flujo es una secuencia de datos que vienen en orden.
- Los datos llegan continuamente con el tiempo.
- No sabemos cuándo terminará.
- Cada dato de un flujo también se denomina evento.

En el contexto de los flujos de datos, un evento es un cambio significativo en el estado de un sistema o un suceso de interés para el sistema.
Estos eventos son los componentes básicos de los flujos de datos.
Cada dato es inmutable y no puede modificarse, pero podemos reproducir la secuencia de datos. Kafka topic es un buen ejemplo de stream.

¿Cada cuánto se procesan los datos?

Micro Batching

El **micro batching** consiste en procesar pequeños lotes de datos a intervalos frecuentes, a menudo de segundos, para mantener el ritmo de la alta velocidad de los datos.

Esos datos se guardan en el almacén de datos que puede ser un simple archivo de texto, una base de datos o un clúster de big data.
A continuación, procesamos los datos, por ejemplo, transformándolos, realizando cálculos, agregando datos, combinándolos, filtrando los datos erróneos, etc.
Normalmente, dejamos los datos originales intactos y creamos un nuevo archivo, tabla o cualquier otra cosa para conservar la salida.

Ejemplo: Cálculo de tipos de interés.

Por ejemplo, el cálculo de los tipos de interés puede hacerse diariamente.
Sin embargo, la comprobación de si el material del almacén sigue siendo suficiente debe hacerse cada 30 minutos para mantener la línea de producción en marcha.

Ejemplo: Registro de tiempos en una carrera.

Considera que tenemos muchos corredores en una maratón y cada corredor lleva un sensor en la zapatilla.
Un dispositivo lector lee el sensor en cada check point y envía los datos al almacenamiento. Estos datos constituyen el flujo de datos.
Cada dato indica quién era el corredor cuando pasó y en qué check point.
A continuación, procesamos los datos para determinar la posición de cada corredor y actualizar el tablero del maratón.
En este ejemplo, el intervalo podría ser de un segundo.
¿Por qué necesitamos un intervalo de tiempo tan micro? Bueno, en el caso de un maratón, tenemos que actualizar el cuadro de mandos.
El público no esperará hasta el minuto uno para determinar si gana su corredor favorito. Por lo tanto, nuestro enfoque es un microlote, que sólo dará una pequeña latencia después de que el corredor pase un hito.

En el procesamiento por lotes, procesamos los datos en determinados intervalos de tiempo.
Si tenemos 200 datos de préstamos que calcular en el lote diario, procesamos los 200.
Si al día siguiente sólo tenemos 170 datos de préstamos, esos datos se procesarán, etcétera.
En el ejemplo del maratón, si tenemos tres corredores en un segundo, procesaremos los tres datos al cabo de un segundo. Cuando sólo un corredor pasa el check point, procesamos ese único dato, etcétera.

Stream Processing

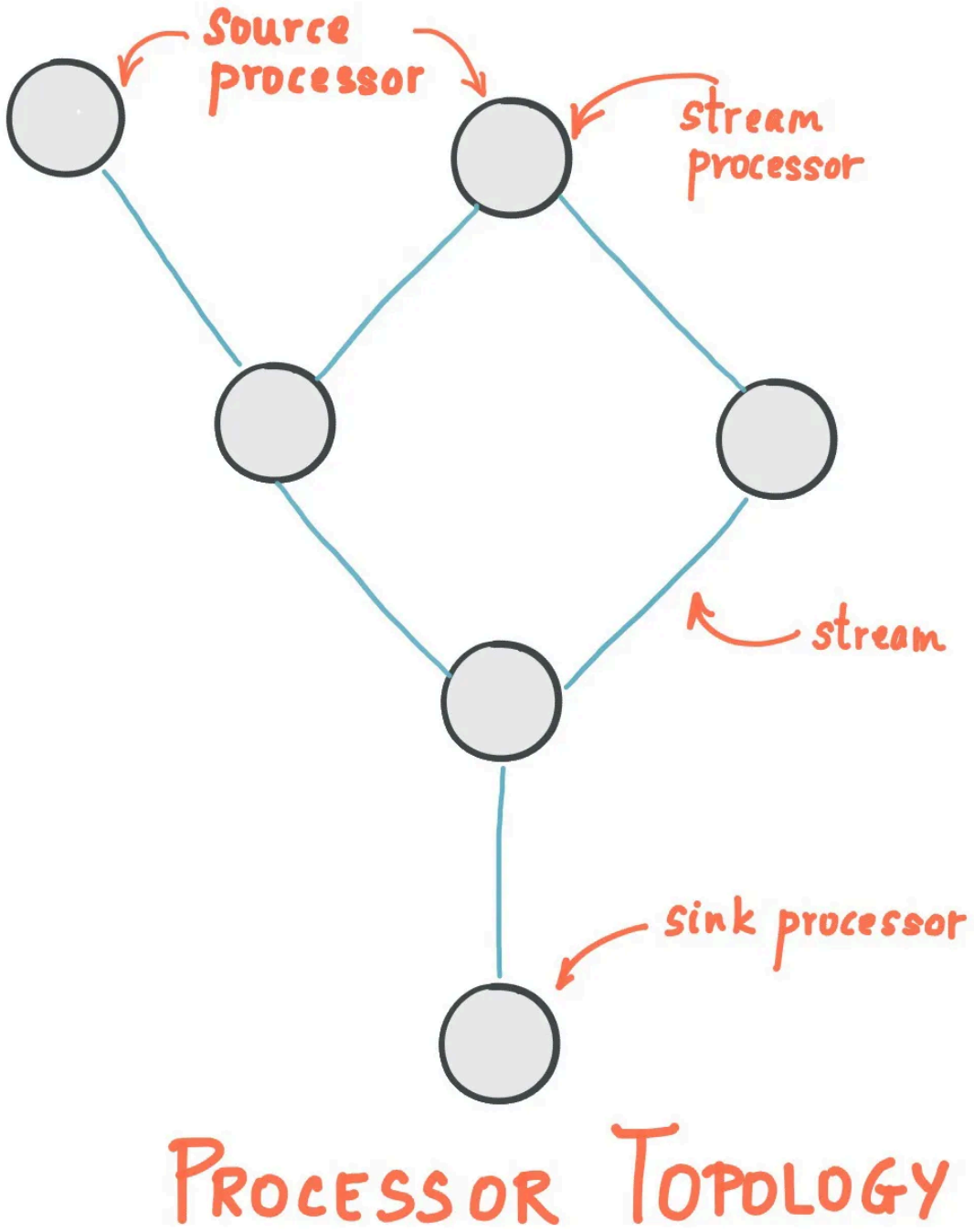
El procesamiento de flujos trabaja con los datos casi en tiempo real.

Hablando del ejemplo del maratón, el corredor pasa por checkpoints y el lector de sensores a intervalos irregulares.
Así, los datos llegan a intervalos irregulares pero de forma ordenada.
El sensor envía cada dato a un flujo de datos, que el marco de procesamiento de flujos procesa en cuanto llega. Por lo tanto, el procesamiento por flujos es diferente del micro batching. El procesamiento de flujos no procesa nada si no llegan datos durante 10 segundos. Por otro lado, cuando llegan 5 datos al flujo, con un retraso de milisegundos cada uno, el procesamiento del flujo los procesa.

Por supuesto, hay una latencia cuando los datos llegan al flujo y comienza el procesamiento del mismo pero es mínima.

El stream processing por ejemplo no será una buena opción en el caso del cálculo de los tipos de interés diarios. El nivel de servicio de tiempo de respuesta es cada 24 horas, por lo que no necesitamos un procesado en tiempo real.
Igualmente la previsión meteorológica analiza grandes cantidades de datos a lo largo del tiempo, identificando patrones, eliminando anomalías, etc. Aquí, la atención se centra en el análisis de los datos, no en los datos más recientes.

Concepto de Kafka Stream



En el procesamiento de flujos usaremos DAGs (grafo acíclico dirigido). En términos de flujo Kafka, llamamos topología a estos diagramas.

- Cada círculo representa un requisito para el procesamiento de datos, comúnmente denominado **procesador de flujo**.
- Cada línea conduce al siguiente procesador.

- Una **topología** en el procesamiento de flujos Kafka es una serie de procesadores de flujos encadenados por flujos.

Los procesadores de flujo procesan el flujo de datos entrantes, dato a dato.
El flujo de datos en el procesamiento de flujos de Kafka es inmutable, lo que significa que no se puede cambiar una vez creado.
El procesador de flujo puede crear un nuevo flujo de salida basado en el flujo de datos de entrada, lo que permite el procesamiento continuo de datos.
Los datos fluyen de los nodos padre a los nodos hijo, nunca de hijo a padre.
Cada nodo hijo, a su vez, puede definir sus nodos hijos, y así sucesivamente.

Los Source Processors o Procesadores Fuente

- No tienen un procesador ascendente en la topología.
- Consumen datos de uno o varios topics de Kafka y los reenvían a sus procesadores posteriores.

Sink Processor

- No tiene un procesador descendente en la topología.
- Envía los datos recibidos de sus procesadores ascendentes a un tema Kafka especificado.

Serde (Serializador/Deserializador)

- En Kafka stream, debemos definir el serde por defecto tanto para la clave como para el valor, aunque podemos anular el serde para cada stream.
- Kafka stream proporciona serde básicos como String o Long. Están disponibles como métodos estáticos de la clase Serdes.
- JSON Serde es proporcionado por Spring y se utiliza para objetos JSON.
- Podemos crear un serde a medida. Por ejemplo, si el formato de los datos es texto separado por comas, podemos crear nuestro propio CSV Serde.