Actividad 5 Módulo 7

1. ¿Como Apache Spark Streaming puede ser utilizado para el procesamiento de datos en tiempo real?¿Cuáles son los componentes?

Apache Spark Streaming es uno de los módulos más importantes dentro del ecosistema de Apache Spark, diseñado específicamente para el procesamiento de flujos de datos en tiempo real. Su capacidad para manejar grandes volúmenes de información de manera distribuida, tolerante a fallos y con latencia relativamente baja lo convierte en una herramienta esencial en escenarios donde se requiere analizar datos tan pronto como son generados. A continuación se presenta en términos generales su funcionamiento, características principales y aplicaciones en el mundo real.

Spark Streaming ofrece dos enfoques para el procesamiento de flujo de datos: el modelo basado en micro-lotes o “DStreams” y el modelo de procesamiento continuo denominado Structured Streaming.

El modelo de **micro-lotes** conocido como “Discretized Streams” divide el flujo continuo de datos en pequeños lotes de intervalos predefinidos, que pueden ir desde los 500 milisegundos hasta varios segundos. Cada uno de estos lotes se representa internamente como un RDD (**Resilient Distributed Dataset**), lo que permite aplicar todas las transformaciones y acciones propias de Spark sobre ellos beneficiando a los que están familiarizados con las APIs de Spark. Sin embargo, la latencia a la espera de que se complete cada lote puede ser una limitante en aplicaciones que requieren una respuesta inmediata.

Por otro lado, el modelo de **procesamiento continuo** está diseñado para reducir la latencia hasta valores MILIMÉTRICOS, ideal para casos de uso que exigen una actualización casi instantánea. En este modelo, los datos se procesan tan pronto como llegan, sin esperar a que se forme un lote, **Structured Streaming**, trata los flujos de datos como una tabla infinita a la que se van agregando registros continuamente, permitiendo realizar consultas SQL-like y aprovechando el optimizador **Catalyst** para mejorar el rendimiento. Aunque este modelo es más eficiente en términos de latencia, requiere un mayor control sobre la gestión de estado y el manejo de fallos.

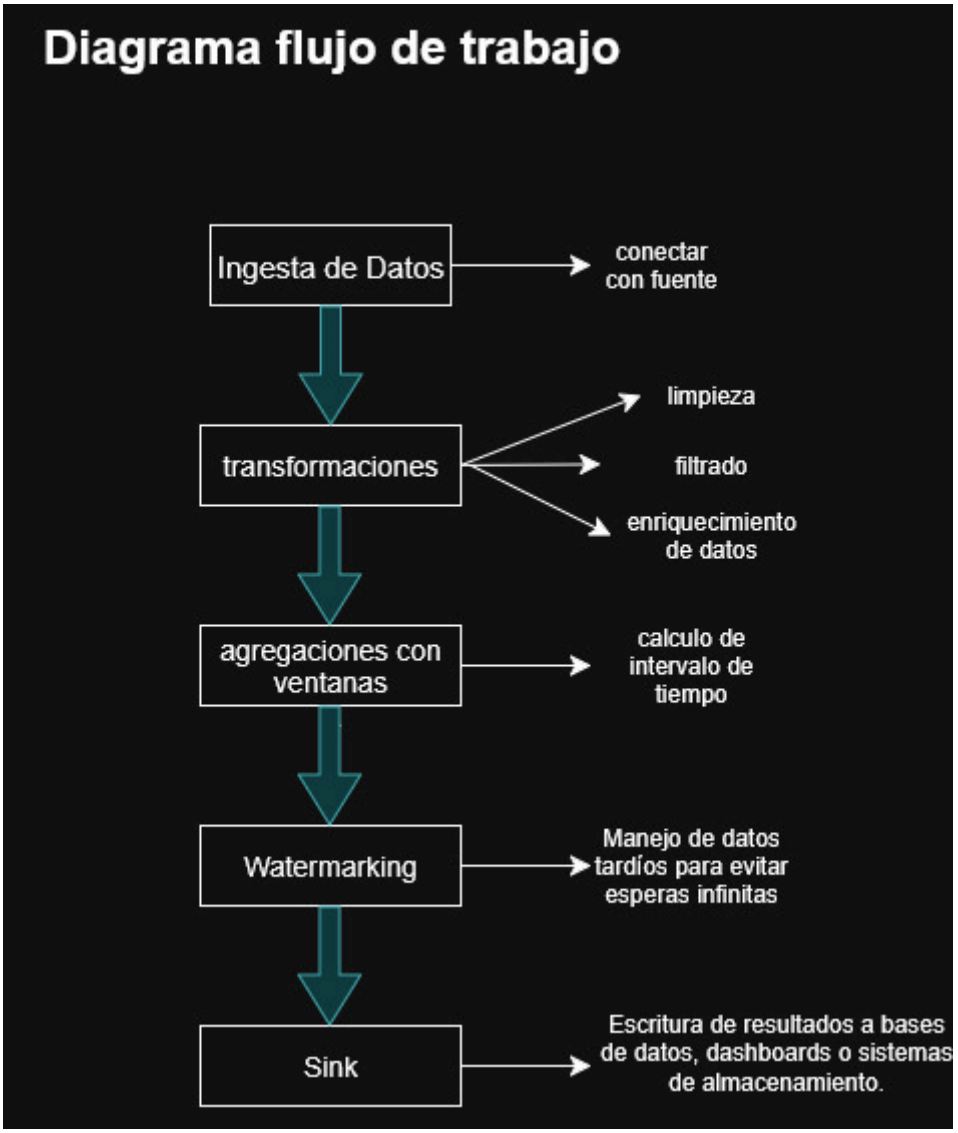
De las virtudes más destacadas de Spark Streaming es su **tolerancia a fallos**. Mediante el mecanismo de **checkpointing**, el sistema guarda periódicamente el estado del procesamiento en un almacenamiento resiliente, como Amazon S3. Esto permite recuperar información en caso de fallos y reanudar el procesamiento desde el último punto conocido o checkpoint, sin perder datos ni generar inconsistencia.

En un flujo de trabajo típico con Spark Streaming sigue una secuencia común. Primero, se establece una conexión con una fuente de datos como por ejemplo un tópico de Kafka, los datos llamados se dividen en micro-lotes o se procesan de manera continua, dependiendo del modelo elegido. Luego, se aplican transformaciones básicas como filtrado, limpieza o enriquecimiento con información externa.

Para agregaciones temporales, se definen ventanas deslizantes que permiten calcular métricas en intervalos específicos, como la cantidad de eventos en los últimos 5 minutos. El

watermarking se utiliza para gestionar datos tardíos, especificando cuánto tiempo se está dispuesto a esperar antes de cerrar una ventana y emitir el resultado.

Finalmente, los resultados se envían a un sink o destino que puede ser una base de datos como Cassandra para consultas operativas, un data like como HDFS para análisis posteriores, o incluso un dashboard en tiempo real como Grafana para visualización inmediata.



2. Estudio de Caso: Monitoreo de Redes Sociales en Tiempo Real.

Un caso práctico en el que se utilizan estas tecnologías es el monitoreo de redes sociales para análisis de sentimiento en tiempo real.

Explicación de la Integración de Componentes.

1. Fuente de datos: Se utiliza un servicio de mensajería como Kafka para ingerir un flujo continuo de tweets o publicaciones. Cada vez que se publica un mensaje que contiene ciertas palabras clave, este se envía a un tema de Kafka.
2. Spark Streaming: Un trabajo de Spark Streaming consume los datos del tema de Kafka. Los tweets se reciben como un flujo continuo y se agrupan en DStreams.
3. Transformaciones y Agregaciones:
    - Se aplica una transformación para analizar el sentimiento de cada tweet (positivo, negativo o neutro).
    - Se utilizan ventanas de tiempo para calcular el sentimiento promedio por minuto o por hora. Por ejemplo, se podría aplicar una ventana de 10 minutos para contar cuántos tweets positivos y negativos se publicaron en ese intervalo.
    - El watermarking se puede usar para manejar tweets que llegan con un ligero retraso y garantizar que se incluyan en el análisis antes de que la ventana de tiempo que se considera cerrada y se descarte la información.
4. Destino (Sink): Los resultados del análisis de sentimiento (por ejemplo, el conteo de sentimientos por minuto) se envían a un destino. Podría ser una base de datos para su posterior consulta, o un sistema de visualización para mostrar los resultados en un dashboard en tiempo real.

De esta manera, Spark Streaming permite procesar y analizar grandes volúmenes de datos de redes sociales a medida que se generan, lo que proporciona información valiosa y casi instantánea sobre la opinión pública respecto a un tema o marca.