Diseño de un job de procesamiento distribuido

Este documento presenta el diseño de un flujo de procesamiento distribuido utilizando Amazon EMR y Apache Spark. Se toma como ejemplo una empresa de streaming que requiere procesar millones de eventos de usuarios para generar recomendaciones y reportes en tiempo real.

Escenario

La empresa de streaming necesita procesar:
- Eventos de usuarios (reproducciones, clics, búsquedas).
- Datos de perfiles e historiales.

El objetivo es consolidar esta información para producir reportes de consumo y alimentar el motor de recomendaciones.

Framework seleccionado

Se utilizará Apache Spark en Amazon EMR por su capacidad de:
- Manejar procesamiento batch y streaming.
- Operar grandes volúmenes en memoria, con mejor desempeño que Hadoop MapReduce.
- Integrar Spark SQL y MLlib para análisis y machine learning.

Estructura del clúster EMR

- Nodo maestro: coordina tareas, administra YARN y Spark.
- Nodos core: ejecutan procesamiento y mantienen datos en HDFS.
- Nodos de tareas: escalan dinámicamente durante picos de tráfico.

Flujo de datos

Origen de datos	Procesamiento	Destino
Logs de navegación en Amazon S3 (Parquet)	Spark Streaming para procesar eventos en tiempo real	Amazon S3 (datos limpios)
Datos de usuarios en DynamoDB	Spark SQL para consultas y agregados	Amazon Redshift (BI dashboards)
Eventos combinados (logs + perfiles)	Transformaciones y enriquecimiento	S3 + API de microservicios (motor de recomendaciones)

Conclusión

El job diseñado con Amazon EMR y Apache Spark permite procesar grandes volúmenes de datos en tiempo real y batch. La arquitectura propuesta integra almacenamiento (S3, DynamoDB), procesamiento distribuido (EMR) y consumo analítico (Redshift, APIs), asegurando escalabilidad y flexibilidad para la empresa de streaming.