Comparativa entre Spark, Hadoop y Flink

Este informe presenta una tabla comparativa entre Apache Spark, Hadoop MapReduce y Apache Flink, analizando seis criterios clave para comprender sus diferencias, fortalezas y casos de uso.

Criterio	Apache Spark	Hadoop MapReduce	Apache Flink
Modelo de procesamiento	Lotes y streaming (con Spark Streaming y Structured Streaming).	Solo procesamiento por lotes.	Procesamiento en tiempo real (streaming nativo) y también por lotes.
Uso de memoria	Computación en memoria (muy rápido).	Basado en disco (más lento).	Computación en memoria con optimización para flujos continuos.
Lenguajes soportados	Scala, Java, Python, R.	Java principalmente (algunos wrappers para Python y otros).	Java, Scala, Python.
Latencia	Baja latencia en streaming, aunque mayor que Flink.	Alta latencia (procesa en lotes grandes).	Muy baja latencia, diseñado para event-time processing.
Casos de uso recomendados	Análisis de grandes volúmenes de datos, ML, ETL, batch + streaming unificado.	Procesamiento de logs históricos, cargas por lotes muy grandes.	Procesamiento en tiempo real de eventos (IoT, sensores, fraudes, streaming).
Facilidad de uso	Fácil de programar, APIs ricas y ecosistema maduro (MLlib, GraphX, Spark SQL).	Difícil de programar (requiere mucho código en Java).	APIs amigables, pero menos maduro que Spark, curva de aprendizaje en streaming.

Escenarios recomendados

- Análisis en tiempo real de redes sociales: Apache Flink (mejor latencia) o Spark Streaming.

- Procesamiento por lotes de logs históricos: Hadoop MapReduce (más barato en batch) o Spark (más rápido en memoria).

- Modelo de aprendizaje automático iterativo: Spark (con MLlib y ecosistema para ML distribuido).