Informe Comparativo: Apache Spark vs Hadoop (MapReduce) vs Apache Flink

Elaborado por: Jonathan Vásquez

1. Contexto

Conocer las fortalezas y debilidades de Apache Spark frente a otras tecnologías de procesamiento de datos distribuidos permite tomar decisiones informadas sobre cuál herramienta utilizar según el tipo de proyecto y requerimientos técnicos.

2. Tabla Comparativa

Criterio	Apache Spark	Hadoop (MapReduce)	Apache Flink
Modelo de procesamiento	Lotes y streaming (micro-batches)	Solo procesamiento por lotes	Streaming nativo y procesamiento por lotes
Uso de memoria	Alto, utiliza memoria RAM para acelerar procesos	Bajo, basado en disco (menos eficiente)	Moderado, optimizado para streaming en memoria
Lenguajes soportados	Scala, Java, Python, R	Java, con algunas APIs en Python y Ruby	Java, Scala, Python
Latencia	Baja a media (micro-batches para streaming)	Alta (por el almacenamiento en disco)	Muy baja (procesamiento nativo de eventos)
Casos de uso recomendados	Análisis iterativo, ML, ETL, streaming	Procesamiento masivo de datos históricos	Procesamiento en tiempo real, analítica de eventos, streaming continuo
Facilidad de uso	Alta, APIs amigables y flexibles	Baja, requiere programación detallada en MapReduce	Media, APIs potentes pero menos extendidas que Spark

3. Análisis de escenarios de uso

1. **Análisis en tiempo real de redes sociales**
    - **Herramienta recomendada:** Apache Flink o Spark Streaming
    - **Justificación:** Se requiere procesamiento de datos en tiempo real con baja latencia. Flink es nativo para streaming, mientras que Spark utiliza micro-batches eficientes y permite integración con ML.
2. **Procesamiento por lotes de logs históricos**
    - **Herramienta recomendada:** Hadoop MapReduce o Apache Spark
    - **Justificación:** Para grandes volúmenes de datos históricos, Hadoop ofrece tolerancia a fallos y procesamiento robusto basado en disco. Spark ofrece velocidad superior en memoria y flexibilidad para análisis iterativos.
3. **Modelo de aprendizaje automático iterativo**
    - **Herramienta recomendada:** Apache Spark
    - **Justificación:** Spark soporta nativamente MLlib, optimiza cálculos iterativos en memoria y facilita la integración con pipelines de datos, a diferencia de Hadoop y Flink.

4. Conclusión

- **Apache Spark:** Destaca por su versatilidad, facilidad de uso y capacidad de procesamiento en lotes y streaming; ideal para ML y análisis iterativos.
- **Hadoop (MapReduce):** Útil para procesamiento masivo de datos históricos y escenarios donde la memoria es limitada.
- **Apache Flink:** La mejor opción cuando la prioridad es el análisis de datos en tiempo real con mínima latencia.