# Comparación entre Apache Spark y Hadoop

## 1. Investigación de la Arquitectura

### Apache Spark

- **Componentes clave**:
  - **Driver**: Coordina el trabajo de la aplicación Spark, planificando y distribuyendo tareas entre los Executors.
  - **Cluster Manager**: Administra los recursos del clúster. Puede ser Spark Standalone, YARN o Mesos.
  - **Executors**: Ejecutan las tareas asignadas por el Driver y almacenan los datos en caché si es necesario.
- **Procesamiento**:
  - **RDDs (Resilient Distributed Datasets)**: Unidad básica de datos en Spark; son tolerantes a fallos y distribuidos.
  - **DataFrames**: Estructuras más optimizadas y orientadas a columnas para análisis de datos, con soporte para optimización de consultas (Catalyst Optimizer).

### Hadoop

- **Componentes clave**:
  - **HDFS**: Sistema de almacenamiento distribuido. Los datos se dividen en bloques y se replican entre nodos.

- **NameNode**: Responsable de gestionar el sistema de archivos y la ubicación de los datos.
- **DataNode**: Almacena los bloques de datos y responde a solicitudes del NameNode.



- **TaskTracker**: Ejecuta las tareas individuales y reporta al JobTracker.

## 2. Comparación de Rendimiento

### Tiempo de Procesamiento

- Spark supera a Hadoop en tareas como el conteo de palabras debido al procesamiento en memoria, mientras que Hadoop requiere múltiples lecturas/escrituras en disco.

### Manejo de Datos: Memoria vs. Disco

- **Spark**:
  - Utiliza procesamiento en memoria (RAM) para acelerar las tareas.
  - Permite almacenamiento intermedio en disco si la memoria es insuficiente.
- **Hadoop**:
  - Procesa directamente desde y hacia el disco, lo que genera mayor latencia.

## 3. Capacidades y Funcionalidades

### Apache Spark

- **Procesamiento en tiempo real**: Spark Streaming para datos en tiempo real.
- **Machine Learning**: MLlib, una biblioteca para aprendizaje automático distribuido.
- **Análisis gráfico**: GraphX para el análisis de grafos.

### Hadoop

- **Ecosistema robusto**: Integración con herramientas como:
  - **Hive**: Consultas SQL sobre HDFS.
  - **Pig**: Lenguaje de alto nivel para procesamiento de datos.

    - **HBase**: Base de datos NoSQL sobre HDFS.
- **Procesamiento por lotes**: Ideal para tareas de gran volumen donde la latencia no es crítica.

# 4. Casos de Uso

## Apache Spark

1. **Procesamiento de datos en tiempo real**: Monitorización de transacciones financieras.
2. **Análisis de grandes volúmenes de datos**: Predicción de tendencias de mercado.

## Hadoop

1. **Almacenamiento y procesamiento de grandes datos históricos**.
2. **Generación de informes y análisis por lotes** en sectores como banca y telecomunicaciones.

# 5. Ventajas y Desventajas

## Apache Spark

- **Ventajas**:
    - Alta velocidad gracias al procesamiento en memoria.
    - Versatilidad para manejar flujos en tiempo real y análisis avanzados.
- **Desventajas**:
    - Consumo elevado de memoria.
    - Curva de aprendizaje más pronunciada para principiantes.

## Hadoop

- **Ventajas**:
    - Escalabilidad y robustez de HDFS.
    - Ecosistema maduro con amplia documentación.
- **Desventajas**:
    - Latencia alta debido al procesamiento basado en disco.
    - Menor flexibilidad para tareas en tiempo real.

# 6. Informe de Resultados

## Diagrama Comparativo de Arquitectura

Representa con diagramas cómo Spark y Hadoop gestionan el procesamiento distribuido.

## Tabla Comparativa de Rendimiento y Funcionalidades

| Aspecto | Apache Spark | Hadoop MapReduce |
|---|---|---|
| **Procesamiento** | En memoria | Basado en disco |
| **Velocidad** | Alta | Moderada |
| **Capacidades adicionales** | Streaming, MLlib, GraphX | Hive, Pig, HBase |
| **Casos de uso típicos** | Tiempo real, análisis | Almacenamiento histórico |

## Conclusión

- **Apache Spark**: Ideal para flujos en tiempo real y tareas con requisitos de velocidad.
- **Hadoop**: Recomendado para almacenamiento masivo y procesamiento por lotes.