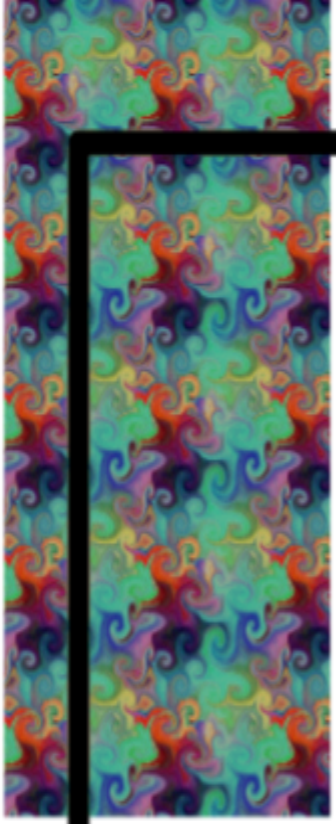


Trabajo 2:
Hadoop VS Spark

CE Inteligencia Artificial y Big Data

Sistemas de Big Data

2024/2025

Daniel Marín López

Índice

1. Introducción....3
2. Características de Hadoop y Spark....3
    2.1. Hadoop....3
    2.2. Spark....4
3. Comparativa de Hadoop y Spark....5
4. Comparativa de Hadoop y Spark en los distintos procesamientos....6
5. ¿Qué arquitectura es la mejor?....7
6. Conclusión....7
7. Webgrafía....8

1. Introducción

Vamos a realizar una comparativa de dos herramientas muy utilizadas en los sistemas de Big Data, estas herramientas son Hadoop y Spark. Hadoop está diseñado para el procesamiento por lotes y para implementar el modelo MapReduce mientras que Apache Spark es una herramienta de procesamiento rápido que usa tanto procesamiento por lotes como en tiempo real ya que trabaja principalmente en la memoria RAM a diferencia de Hadoop que trabaja en el disco duro.
Veremos más en detalle no solo todas sus características, sino como trabajan en los distintos procesamientos, que herramienta es más acorde dependiendo del proyecto en el que trabajemos y que casos de uso reales están implantando estas herramientas y cómo les ha ido a lo largo del tiempo.

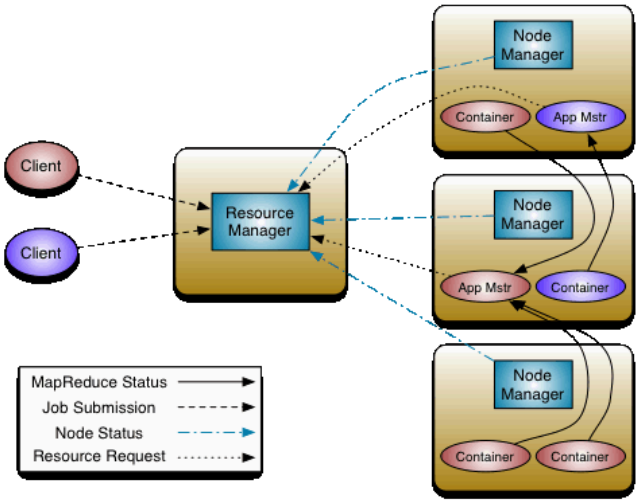
2. Características de Hadoop y Spark

2.1. Hadoop

Las características que presenta Hadoop son las siguientes:

- ✅ Hadoop fue diseñado para trabajar en procesamiento por lotes. Fue creado por Apache Foundation como la herramienta MapReduce de código abierto.
- ✅ Hadoop se divide en tres componentes:
  - ✔ **YARN**, el gestor de recursos y planificador de tareas de Hadoop.
  - ✔ Su propio sistema de archivos distribuidos conocido como **HDFS**.
  - ✔ Su gestor de datos **HBase**.
- ✅ Hadoop cuenta con lo siguiente:
  - ✔ Es escalable
  - ✔ Es tolerante a fallos
  - ✔ Es flexible para almacenar datos
- ❌ Los retos que más se aprecian con Hadoop son
  - ✖ Su falta de recursividad e interactividad
  - ✖ Coste inicial alto
  - ✖ Complejidad de administración
- ✅ Otras herramientas que se incluyen en Hadoop son:
  - ✔ Apache Sqoop
  - ✔ Cassandra
  - ✔ Flume
  - ✔ Mahout

La idea fundamental de YARN es dividir las funcionalidades de gestión de recursos y programación/monitoreo de trabajos en daemons separados. La idea es tener un **ResourceManager** (*RM*) global y un **ApplicationMaster** (*AM*) por aplicación . Una aplicación es un trabajo único o un DAG de trabajos.



ResourceManager y **NodeManager** forman el marco de trabajo de computación de datos. El **ResourceManager** es el que tiene mayor autoridad, ya que distribuye entre las aplicaciones del sistema la carga de trabajo. El **NodeManager** está en cada máquina y se encarga de informar al ResourceManager de toda la información disponible de la máquina (CPU, memoria, disco, red, ...). El **ApplicationMaster** es el encargado de negociar entre el ResourceManager y los NodeManager.

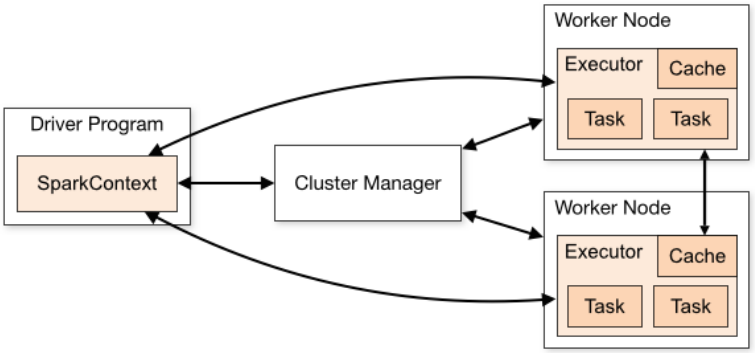
2.2. Spark

Las características que destacan a Spark son las siguientes:

- ✅ Apache Spark se caracteriza por su velocidad. Fue creado como un clúster de código abierto en la Universidad de California y luego donado a Apache Foundation.
- ✅ Apache Spark tiene los siguientes componentes:
  - ✔ **Spark Core:** El motor de procesamiento distribuido que gestiona la memoria, la planificación y la ejecución de tareas.
  - ✔ **Spark SQL:** Permite el procesamiento de datos estructurados mediante consultas SQL y proporciona una interfaz más amigable para analistas.
  - ✔ **Spark Streaming/Structured Streaming:** Facilita el procesamiento de flujos de datos en tiempo real o en micro-lotes, adaptándose a necesidades de análisis inmediato.
  - ✔ **MLlib:** Biblioteca integrada de machine learning para ejecutar algoritmos de clasificación, regresión, clustering, etc.
  - ✔ **GraphX:** Permite el análisis y procesamiento de grafos, útil en redes sociales o análisis de relaciones complejas.
- ✅ Spark se destaca por procesar datos “in-memory” (en la RAM), lo que reduce la latencia y mejora el rendimiento en operaciones iterativas y análisis en tiempo real.
- ✅ Spark tiene muchas APIs para usar (Java, Python, Scala y R), lo que facilita a los desarrolladores crear sus aplicaciones de datos.
- ✅ Además, Spark soporta SQL, aprendizaje automático (MLlib), procesamiento de gráficos (GraphX) y procesamiento de flujos de datos. Lo que lo hace muy versátil.

❌ A pesar de que Spark puede integrarse con HDFS o incluso con otros sistemas de almacenamiento (como S3 o Redshift), no cuenta con un sistema de ficheros propio.
❌ Al procesar los datos en RAM, hace que Spark necesite muchos recursos para funcionar.

Apache Spark fundamenta su arquitectura en lo que se conoce como **RDD** o **Resilient Distributed DataSet**, que es un multiset de única lectura de ítems de datos distribuidos a través de un conjunto de máquinas, manteniéndose en un ambiente tolerante a fallos.
En la versión de Spark 1.x, la API principal era RDD, pero con la evolución de Spark 2.0, se aconseja el uso de la API DataSet. A pesar de que la API RDD no se considera descatalogada, la tecnología RDD sigue siendo parte de la base de la API DataSet.



Spark y sus RDDs se crearon en 2012 como respuesta a las restricciones del paradigma de computación en conjunto **MapReduce**, que obligaba a emplear una estructura de flujo de datos lineal específicamente en los programas distribuidos. Los programas que utilizan MapReduce analizan los datos de entrada del disco, que traza una función a través de los datos, disminuye los resultados del mapa y guarda los resultados de la disminución en el disco.
Los RDDs de Spark operan como un conjunto de trabajo para los programas distribuidos, proporcionando una forma (intencionalmente) limitada de la memoria compartida distribuida.

3. Comparativa de Hadoop y Spark

Si comparamos ambas arquitecturas vemos lo siguiente:

Arquitectura	Ventajas	Desventajas
Hadoop	**Escalabilidad y costo:** Permite crecer el clúster agregando nodos de hardware estándar a un costo relativamente bajo.	**Procesamiento en disco:** MapReduce escribe y lee datos desde disco en cada paso, lo que genera alta latencia y lo hace menos apto para procesamiento en tiempo real.
	Tolerancia a fallos: La replicación de bloques en HDFS asegura alta disponibilidad ante fallos en nodos individuales.	**Curva de aprendizaje:** Requiere un conocimiento profundo de Java y el paradigma MapReduce, lo que puede incrementar el tiempo de desarrollo para operaciones complejas.

	Versatilidad en datos: Adecuado para datos estructurados, semiestructurados y no estructurados, lo que lo hace útil en entornos de análisis histórico o batch.	**No optimizado para iteraciones:** Las tareas que necesitan procesamiento iterativo (como algunos algoritmos de machine learning) pueden verse afectadas por la latencia de I/O.
Spark	**Velocidad:** Gracias a su procesamiento en memoria, Spark puede ser hasta 100 veces más rápido que Hadoop MapReduce en ciertas cargas de trabajo, especialmente para algoritmos iterativos y en tiempo real.	**Requerimientos de memoria:** El procesamiento "in-memory" implica mayor inversión en RAM, lo que puede incrementar los costos, especialmente en implementaciones on-premise.
	Flexibilidad de API: Ofrece interfaces en Java, Scala, Python y R, lo que facilita el desarrollo y la integración con otros sistemas.	**Dependencia del almacenamiento subyacente:** Al no contar con su propio sistema de ficheros, depende de HDFS u otros sistemas, lo que en ciertos escenarios puede limitar la independencia o requerir configuraciones adicionales.
	Procesamiento en tiempo real y batch: Con Spark Streaming, es posible procesar flujos de datos continuos, y a la vez, Spark es muy eficaz para cargas por lotes.	**Escalabilidad en ciertos contextos:** Aunque se escala horizontalmente, ampliar Spark puede resultar más costoso debido a la necesidad de mayor capacidad de memoria en cada nodo.
	Bibliotecas integradas: MLlib y GraphX permiten ejecutar machine learning y análisis de grafos sin depender de herramientas externas	

4. Comparativa de Hadoop y Spark en los distintos procesamientos

Si nos centramos en cada uno de los procesamientos, podemos ver que cada arquitectura se especializa más en uno que otros. Para cada procesamiento tenemos los siguiente:

✅ **Procesamiento Batch**: Hadoop es adecuado para el procesamiento de grandes conjuntos de datos en lotes, donde la latencia no es crítica. Ejemplos incluyen el análisis de registros web, la generación de informes y la transformación de datos a gran escala.

✅ **Streaming**: Spark Streaming es una buena opción para el procesamiento de datos en tiempo real o casi real, como el análisis de datos de sensores, la detección de fraudes y la monitorización de redes sociales. Requiere micro-batching, lo que introduce cierta latencia.

✅ **Tiempo Real**: Para el procesamiento en tiempo real estricto, donde la latencia debe ser mínima, otras tecnologías como Apache Flink o Apache Kafka Streams pueden ser más adecuadas. Sin embargo, Spark puede integrarse con estas tecnologías para proporcionar capacidades de procesamiento adicionales.

5. ¿Qué arquitectura es la mejor?

Ambas arquitecturas son muy usadas en la actualidad y dependiendo del proyecto en el que nos encontremos tendremos que escoger una u otra o las dos incluso. Aquí hay una serie de factores a tener en cuenta:

- **Latencia y velocidad**: Si necesitamos que nuestro proyecto trabaje en tiempo real o streaming, Spark es la mejor elección para trabajar.
- **Costos de infraestructura**: Si no disponemos de una buena infraestructura y solo disponemos de un hardware estándar entonces lo mejor será usar Hadoop.
- **Tipos de datos y cargas de trabajo**: Si nuestro proyecto es de análisis histórico o cargas en batch, lo mejor es usar Hadoop. Por el contrario, si incorporamos aprendizaje automático o procesamiento batch con tiempo real es mejor usar Spark por su flexibilidad.
- **Experiencia de los desarrolladores**: Spark tiene APIs de Python, SQL, Scala o R; lo que hace que desarrolladores nuevos puedan usarlo con facilidad al tener lenguajes de programación fáciles de usar. Hadoop, por otro lado, se especializa en MapReduce y Java, algo que solamente desarrolladores experimentados pueden trabajar sin mucho esfuerzo.

Podemos aplicar esto en el siguiente caso de uso:

"Imaginemos una empresa e-commerce que necesita procesar y analizar tanto conjuntos de datos históricos de transacciones (para analizar tendencias y generar informes en base a ellas) como monitorear a tiempo real las interacciones de los usuarios en la plataforma (detectar comportamientos sospechosos o ajustar de forma dinámica las promociones)."

Aquí podemos implementar un sistema dual Hadoop-Spark, Hadoop se encargaría del procesamiento batch para los datos históricos mientras que Spark Streaming se encargaría de los datos en tiempo real. Además, si queremos entrenar modelos de recomendación basados en el aprendizaje automático, Spark MLib sería una librería ideal para esta tarea debido a su velocidad y facilidad para manejar algoritmos interactivos.

6. Conclusión

La elección entre Hadoop y Spark depende de una serie de factores, incluyendo el tipo de procesamiento requerido, el tamaño de los datos, los requerimientos de latencia y el coste. Hadoop es adecuado para el procesamiento batch a gran escala, donde la latencia no es crítica y el coste es un factor importante. Spark, por otro lado, es ideal para aplicaciones que requieren baja latencia, procesamiento iterativo o análisis interactivo de datos. Spark Streaming permite el procesamiento de datos en tiempo real o casi real. En muchos casos, Hadoop y Spark se utilizan en conjunto, con Hadoop proporcionando el almacenamiento de datos y Spark proporcionando las capacidades de procesamiento. Esta combinación permite aprovechar las ventajas de ambas tecnologías y abordar una amplia gama de problemas de Big Data. Antes de elegir una herramienta u otra, es crucial analizar los requerimientos específicos del proyecto y evaluar cuidadosamente las ventajas y desventajas de cada tecnología.

En última instancia, la mejor herramienta es aquella que se adapta mejor a las necesidades del proyecto y permite alcanzar los objetivos de negocio de manera eficiente y efectiva. Considerar la experiencia del equipo, la disponibilidad de recursos y la escalabilidad futura son factores adicionales que deben influir en la decisión final.

7. Webgrafía

- Amazon: Hadoop vs Spark
- Hadoop vs Spark: Una comparación en profundidad
- ¿Qué son Hadoop y Spark?
- Hadoop vs Spark: Comparación de Tecnologías de Big Data