6. Respuesta a las actividades de investigación.

Elaborar una tabla comparativa que resalte las diferencias clave entre Hadoop y Spark en términos de arquitectura, procesamiento, rendimiento y casos de uso.

|  |  |  |
| --- | --- | --- |
|  | **Hadoop** | **Spark** |
| **Procesamiento** | Procesamiento por lotes (batch processing). Los trabajos se ejecutan en etapas y se almacenan en disco entre etapas. | Procesamiento en memoria (in-memory processing). Permite realizar operaciones en datos en tiempo real y en lotes. |
| **Rendimiento** | Hadoop puede procesar grandes conjuntos de datos en lotes, pero puede ser más lento. Para procesar los datos, Hadoop lee la información del almacenamiento externo y luego analiza e introduce los datos en algoritmos de software. | Apache Spark, por otro lado, está diseñado para procesar cantidades enormes de datos en tiempo real. En lugar de acceder a los datos del almacenamiento externo, Spark copia los datos en la RAM antes de procesarlos. |
| **Casos de uso** | Ideal para procesamiento de grandes volúmenes de datos en lotes, análisis de datos históricos, y almacenamiento de datos. | Ideal para procesamiento en tiempo real, análisis de datos en streaming, machine learning, y trabajos iterativos. |

• Definir los concepto de RDD. Propiedades de los RDDs: Inmutabilidad, particionamiento.

**RDD (Conjunto de datos distribuidos resilientes)**

Los conjuntos de datos distribuidos resilientes (RDD) son colecciones de elementos tolerantes a fallas que se pueden distribuir entre varios nodos en un clúster y trabajar en paralelo. Los RDD son una estructura fundamental en Apache Spark.

**Propiedades de los RDDs:**

Los RDD (conjuntos de datos distribuidos resilientes) son estructuras de datos inmutables y particionadas que se usan en el motor de procesamiento de datos Apache Spark.

**Inmutabilidad:**

* Evita problemas por actualizaciones simultáneas de varios subprocesos
* Garantiza la consistencia de los datos entre subprocesos
* Reduce el riesgo de condiciones de carrera
* Permite recrear partes del RDD en cualquier momento

**Particionamiento:**

* Divide lógicamente los datos en fragmentos más pequeños y los distribuye entre los nodos del clúster
* Permite procesar los datos en paralelo
* Optimiza la ubicación de los datos
* Mejora el tiempo de respuesta
* Evita la interrupción total del servicio

**Tipos de Operaciones:**

**Transformaciones:** Las transformaciones son operaciones que crean un nuevo RDD (Resilient Distributed Dataset) a partir de uno existente. Estas operaciones son perezosas, lo que significa que no se ejecutan hasta que se llama a una acción.

* **Map:** Aplica una función a cada elemento del RDD y devuelve un nuevo RDD con los resultados.
* **Filter:** Filtra los elementos del RDD que cumplen con una condición dada, devolviendo un nuevo RDD con los elementos que pasan el filtro.
* **FlatMap:** Similar a map, pero cada elemento puede ser mapeado a cero o más elementos. Devuelve un nuevo RDD que es "aplanado".
* **GroupByKey:** Agrupa los valores por clave en un RDD de pares clave-valor. Devuelve un nuevo RDD con cada clave y una lista de sus valores asociados.
* **ReduceByKey:** Combina los valores de cada clave utilizando una función de reducción.

**Acciones:** Las acciones son operaciones que devuelven un resultado al programa o escriben datos en un sistema de almacenamiento externo. A diferencia de las transformaciones, las acciones desencadenan la ejecución de las transformaciones pendientes.

* **Collect:** Devuelve todos los elementos del RDD como una lista al controlador (driver).
* **Count:** Devuelve el número de elementos en el RDD.
* **Take:** Devuelve los primeros n elementos del RDD como una lista.
* **Reduce:** Combina todos los elementos del RDD utilizando una función de reducción y devuelve un solo valor.

• **Concepto de DataFrame:**

Es una estructura de datos tabular que se utiliza ampliamente en el análisis de datos y la manipulación de datos en bibliotecas de programación como Pandas en Python y Spark en el ecosistema de Big Data.

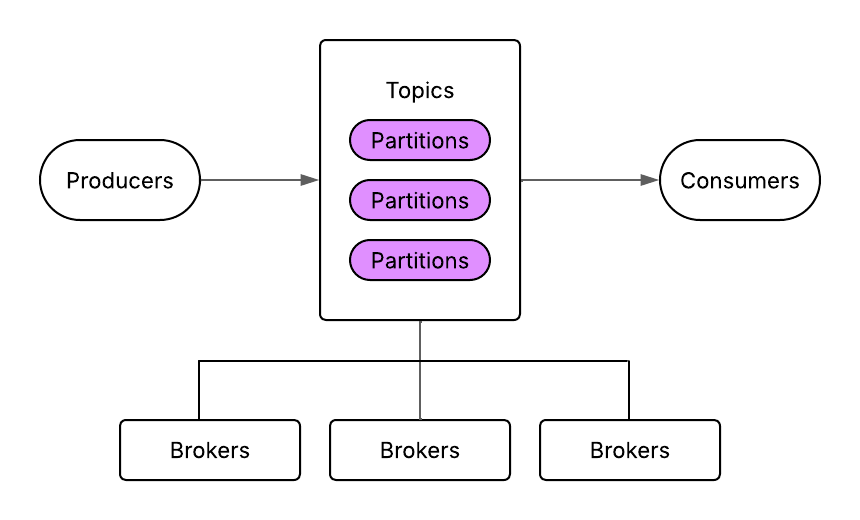
**Diferencias sobre los RDDs:**

* Trabajas directamente con objetos distribuidos sin estructura tabular.
* A diferencia de DataFrames/Datasets, los RDDs no tienen un esquema predefinido.
* Permiten personalizar transformaciones y acciones a bajo nivel.
* No se optimizan automáticamente
* Seguro en tiempo de compilación (en Scala y Java)

**Ventajas sobre los RDDs:**

* Flexibilidad total
* Manejo explícito de errores
* Transformaciones funcionales
* Persistencia y tolerancia a fallos
* Compatibilidad con datos no estructurados

• Crear un diagrama que ilustre la arquitectura de Kafka y explicar los conceptos clave como topics, partitions, brokers, producers y consumers.



* **Topics:** son categorías o nombres a los que se envían los mensajes.
* **Partitions:** cada topic se divide en una o más particiones.
* **Brokers:** cada *broker* gestiona datos y solicitudes para una o más particiones.
* **Producers:** aplicaciones que envían mensajes a Kafka.
* **Consumers:** aplicaciones que leen mensajes desde Kafka.