Spark (clase 28-02-2023)

Resilient Distributed Datasets (RDD)

Resilient Distributed Datasets (RDD): representa los datos dentro de un framework. No tiene esquema.

En ***spark*** definimos las transformaciones una detras de la otra y no hay límite estas se ejecutan en paralelo.. Sin embargo las transforamciones en spark no se ejecutan, spark se imagina el resultado. Genera un arbol de transforamciones.En un data set de 1TB las transformaciones se ejecutan en cada 100 registros. Spark permite usar sql para manejar los datos y hacer particiones sobre el dataset completo. Pero hay que considerar que los servidores tienen diferentes características de hardware y software. A los RDD se les pueden aplicar transformaciones, pero los datos siemprese van a mantener inmutables.

Conceptos

- **lazy evaluation**: hasta que no ejecute una acción no se dispara el set de transformaciones sobre todo el dataset de objetos.
- **Esquema de una base de datos**: delimita el nombre del tipo de dato y el tipo de dato. En no sql se puede llamar template. Es como los esquemas entidad relación pero sin los métodos.
- raw data: datos crudos, datos de los cuales no nos preocupamos por su formay su formato. Van a quedar en memoria como bytes.
- **Especulative Execution / task**: Puede pasaar que un servidor se encargue de procesar 2 particiones y dure tiempos diferentes, el proceso en total se va a atrasar. Lo ideal es recalendarizar en otro servidor, pero esto implica movimiento de datos que toma tiempo. Spark resuelve esto con replicación de datos. Va a generar el procesamiento de la partición 6 en el servidor 1 y 2 por ejemplo. Sparck va a medir cuál está procesando los datos más rápidamente y va a matar la operación más lenta (tarea especulativo).
- **Scala**: lenguaje parecido a java, con una interfaz más amigable y con una definición más estricta.
- **Dato no estructurado**: audio, video, imágenes. No siguen un patron.
- **Dato estructurado**: tienen un esquema bien definido, tipo y nombre.
- **Dato semi estructurado**: un híbrido entre datos estructruados y no estructurados. No cumplen una característica de los datos estructurados. ejm JSON.
- **Spark Streaming**: manipulación de big data en tiempo real. Se define por ventanas de tiempo o por tamaños. El procesamiento por tiempo está enfocada a mantener los datos en tiempo casi real (es preferible). La desventaja es que se gasta hardware aún cuando llegan pocos datos.
- **Provisioning**: el tiempo que se tarda en crear toda la infraestructura necesaria para poder procesar datos.
- **Query-Able**: que puede hacer consultas sencillas de los datos.

- **Pojo**: mapea una clase de java con una base de datos.
- **Fault Tolerance**: para garantizar tolerancia a fallos las particiones tendran replicación y procesamiento en paralelo.
- **Hadoop**: framework para gestionar particiones de datos.
- **Serialización** (característica de java): cuando le damos valor a los atributos de una clase obtenemos un objeto. Java internamente organiza los datos dentro de la memoria (esto se sale del scope de lo que nos concierne a los programadores). Para poder pasar los datos a otra máquina, podemos hacer un ***dump*** de la memoria. Lo malo es que si la versión de java es distinta, entonces ya los datos se vuelven incompatibles. Hay que buscar un lenguaje común. La serialización permite convertir la representación del objeto en memoria a un documento json o xml.

Data Frame

Hermano mayor de los RDD. Tiene un esquema compuesto por nombres y tipos de datos. Los tipos de datos son primitivos.

Data Set

Es el más grande de la familia. Solo funciona con scala y con java. Es como "*un RDD con esteroides*". Se mapea con ***pojo***. Tendremos datos fuertemente tipados, no se puede manipular un tipo de dato como si fuera otro. Por eso conocemos que java se mapea a orientación a objetos y tiene un esquema relacional (se parece a la definición de esquemas de una base de datos). Un dataset soporta serialización

Spark

En algun momento vamos a usar spark en stand alone (en la compu localmente)

tiene una interfaz con python, pero el profe usa scala. Para corrar en java tenemos que tener la vm arriva de la .8 y un zip.

- bin/spark-shell
- ya tenemos una instancia de spark
- permite interactuar con el framework de procesamiento de datos.
- el import de spark config permite que el scheduler configure cada nodo
- lo que vamos a usar más es spark.sql para trabajar big data (casi sin darnos cuenta)
- los datasets se encuentran en craggle
- si queremos mandar datos a elastic search, hay que cambiar la configuración (el ejemplo nos lo va a pasar)

Para bajar Elastic

1. **buscamos elasticsearch hadoop en google**
2. buscamos la primera opción, instalación y en la página están las instrucciones de configuración
3. luego hacemos un port-foward, que garantiza que en 127.0.0.1:9200 podamos acceder a elastic search.
4. en kibana ejecutamos GET cat/indices
5. detenemos spark con unos comandos

6. luego pegamos la configuración en linea de comandos (indices, usuarios, el puerto, ignorar los nodos que se encuentran en warning)
7. se instancia un spark context
8. se instancia un spark session
9. cargamos los datos definiendo una varible mediante una sessión de spark que tiene ciertas configuraciones (tipo del doc, delimitador, valor del escape, especificar si los datos traen header en la primera fila, si es un csv multilínea, ubicación del csv). Esto genera un dataframe, del cuál podemos obtener su RDD. Un dataframe es una abstracción que contiene un RDD.

- En este punto spark ya leyó el esquema de datos, puede que no haya entendido bien, pero podemos editar el esquema.
- Podemos pedir una generación automática de una tabla tipo SQL
- Podemos usar spark.sql("SELECT * FROM breed"), pero esto es una transformación, para ver los datos ocupo una acción como "show".
- El comando "cast" permite cambiar el tipo de dato de una columna
- "Repartition" permite controlar el número de particiones que se van a generar.

Datos Textuales

El peor enemigo para el manejo de datos. Se desperdicia el almacenamiento al representar las 27 letras del abecedario con un byte. Cosa que no pasa con la compresión.

Existen tipos de datos más poderosos para guardar datos. Un parquet permite ahorrar hasta un 70 % del almacenamiento. Parquet es un almacenamiento columnar.

Parquet (formato de serialización súmamente eficiente)

Las bases de datos relacionales trabajan con filas. al buscar un atributo de todos los registros desperdicio el manejo de memoria, cache y disco. el columnar storage tiene separados los datos de cada columna (crea un archivo por columna). Con esto gana no tener que traer toda la información de disco. Aparte de esto parquet comprime toda información textual. parqut también guarda eltipo y el nombre de columna.

Guardar Datos en Elastic-Search

.saveToEs guarda los datos de un query en elastic search. Los guarda sin formato, no es parquet ni JSON