INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

ACADEMIA INGENIERÍA DE SOFTWARE

Unidad de Aprendizaje: Big Data

Profesor: Ing. Tania Rodriguez Sarabia

Práctica no. 1: Introducción a Apache Spark

ALUMNO:

GRUPO:

FECHA DE ENTREGA:

Qué es apache Spark?

Apache Spark es un motor informático unificado y un conjunto de bibliotecas para el procesamiento de datos en paralelo en clústeres de computadoras. Spark es el motor open source más activamente desarrollado para esta tarea, haciéndolo la herramienta estándar para cualquier desarrollador o científico de datos interesado en big data. Spark es compatible con varios lenguajes de programación de uso generalizado, incluye bibliotecas para diversas tareas que van desde SQL hasta streaming y aprendizaje automático y se ejecuta en cualquier lugar, desde una PC hasta un grupo de miles de servidores. Esto lo hace un sistema fácil para empezar y escalar a procesamiento de big data.

La filosofía de apache spark es un motor de computo unificado y una serie de librerías para el big data. Unificado, porque la meta principal de spark es ofrecer una plataforma unificada para escribir aplicaciones de big data. Spark esta diseñado para soportar un amplio rango de tareas analíticas, que van desde la carga de datos y queries SQL a machine learning y computaciones de streaming. La naturaleza unificada de spark hace a estas tareas fáciles y eficientes de escribir; spark, provee APIS consistentes, componibles que puedes usar para armar aplicaciones grandes de piezas más pequeñas.

También es un motor de computación, spark se encarga de manejar datos desde el sistema de almacenamiento y realizar cálculos sobre ellos, no del almacenamiento permanente en sí. Puede utilizar Spark con una amplia variedad de sistemas de almacenamiento persistente , incluidos

sistemas de almacenamiento en la nube como Azure storage y amazon S3, sistemas de archivos distribuidos como apache hadoop, almacenes de clave-valor como apache cassandra y buses de mensajes como apache Kafka. Spark se enfoca en la computación de los datos, lo que lo hace diferente de plataformas de software de big data como hadoop, incluíye un sistema de guardado (Hadoop file system) y un sistema de computo (MapReduce) que estaban cercanamente integrados, sin embargo, esta decisión hace dificil correr un sistema sin el otro, esta decisión también hace dificil hacer aplicaciones que accedan a datos guardados en cualquier otro lugar. Spark, por otro lado corre bien en el almacenamiento de Hadoop, y hoy también es usado ampliamente en ambientes en donde la arquitectura de Hadoop no tiene sentido, tales como la nube pública o aplicaciones de streaming.

Finalmente, librerías, los componentes finales de spark son sus librerías, las cuales están construidas como un motor unificado para proveer un API unificada para tareas de análisis de datos comunes. Spark soporta librerías standard que se embarcan con el motor así como una aplia variedad de librerías externas publicadas como paquetes de terceros por la comunidad open-source. Spark incluye librerías para SQL y datos estructurados, machine learning y procesamiento de flujo.

Contexto: El problema del big data

Porque necesitamos un nuevo motor y modelo de programación para análisis de datos en primer lugar? Esto es debido a cambios en los factores económicos que están debajo de las aplicaciones y el hardware.

Durante la mayor parte de su historia, las computadoras se han vuelto más rápidas cada año por los incrementos en la velocidad del procesador, como resultado, las aplicaciones también se hicieron automáticamente mas rápidas cada año. Desafortunadamente, esta tendencia paró alrededor de 2005: Los desarrolladores de hardware pararon de hacer a procesadores individuales mas rápidos y cambiaron hacia añadir más núcleos de CPU paralelos por la misma velocidad. Las aplicaciones necesitaban ser modificadas para añadir paralelismo para correr mas rápido, lo que colocó el telón para que nuevos modelos de programación como Apache Spark aparecieran.

Además de eso, las tecnologías para guardar y colectar datos no se han desacelerado desde 2005, cuando los procesadores lo hicieron. EL costo de guardar 1TB de datos continúa cayendo aproximadamente dos veces cada 14 meses, lo que significa que es muy barato para las organizaciones de todos los tamaños guardar grandes cantidades de datos. Además muchas de las tecnologías para recolección de datos como sensores y cámaras, continúan cayendo en precio y mejorando en resolución. El resultado final es un mundo en donde recolectar datos es extremadamente barato, pero procesarlos requiere computaciones paralelas usualmente en clústeres de computadoras, aún más importante, el software desarrollado hace 50 años no puede escalar automáticamente, creando la necesidad por nuevos modelos de programación como Apache Spark.

Descargar Apache Spark:

Si vas a descargar y correr Spark localmente, el primer paso es asegurarse que tienes Java instalado en la versión Java 8 u Java 11. Visita la pagina oficial de spark: https://spark.apache.org/downloads.html, selecciona “pre-build for apache Hadoop 3.3 an later” y da click en direct download. Vas a descargar un archivo .tar que debes descomprimir en una carpeta, por ejemplo “C:\spark”

Configurar Variables de Entorno:

Abre el Panel de Control > Sistema > Configuración avanzada del sistema > Variables de entorno.

Crea una nueva variable de sistema llamada SPARK_HOME y asígnala a la ruta de la carpeta de Spark (por ejemplo “C:\spark\spark-3.3.1-bin-hadooop3”.

Agrega %SPARK_HOME%\bin al Path del Sistema

Hadoop WinUtils (Opcional):

Spark en Windows requiere un archivo llamada winutils.exe para interactuar con el sistema de archivos.

Descarga winutils.exe desde https://github.com/steveloughran/winutils

Coloca winutils.exe en una carpeta por ejemplo “C:\hadoop\bin”

Configura la variable de entorno HADOOP_HOME, crea una nueva variable de sistema llamada HADOOP_HOME y asígnala a la ruta de la carpeta que contiene winutils.exe (por ejemplo C:\hadoop)

Verificar la instalación:

Abre una terminal y ejecuta.

$$spark - shell$$

Ahora debemos instalar pyspark, si deseas usar PySpark, instala la biblioteca de PySpark usando pip.

$$pip\ install\ pyspark$$

La SparkSession:

Cuando inicias spark en el modo interactivo, necesitas crear implícitamente un SparkSession, la instancia de SparkSession es la manera en que Spark ejecuta manipulaciones definidas por el usuario a través del cluster. En escala y Python, la variable esta disponible como Spark cuando empieza la consola.

En Python:

```
from pyspark.sql import SparkSession
spark = SparkSession.builder \
.appName("MiPrimeraApp") \
.getOrCreate()
```

En scala:

```
val spark = SparkSession.builder( )
.appName(Flight Analysis)
.master("local[*]")
.getOrCreate()
```

Ahora vamos a crear una tarea simple de crear un rango de números.

```
// En scala
val myRange = spark.range(1000).toDF("number")

# En python
myRange = spark.range(1000).toDF("number")
```

Hemos creado un DataFrame que con una columna que contiene 1,000 filas con valores de 0 a 999.

Transformaciones

En Spark las estructuras de datos núcleo son inmutables, lo que significa que no pueden ser cambiadas después de ser creados. Para "cambiar" un DataFrame, necesitas instruir a Spark como te gustaría modificarlo para hacer lo que quieres. Estas instrucciones son llamadas transformaciones, vamos a hacer una transformación simple para encontrar todos los números pares en el DataFrame actual.

```
// En scala
val divisBy2 = myRange.where("number %2 = 0")

#En Python
divisBy2 = myRange.where("number % 2 = 0")
```

Acciones

Las transformaciones nos ayudan para construir nuestro plan de transformación lógica. Para accionar las computaciones corremos una acción. Una acción instruye a Spark a computar el resultado de una serie de transformaciones. La acción más simple es count, lo cuál nos da el número total de registros en el DataFrame:

```
divisBy2.count( )
```

Spark UI

Puedes monitorear el progreso de un trabajo a través de la web Spark UI. La Spark UI está disponible en el puerto 4040 del driver node, si estas corriendo en modo local, sería http://localhost:4040.

Por realizar

- Descargar los datos de https://github.com/databricks/Spark-The-Definitive-Guide/tree/master, leer los datos de los vuelos registrados en el archivo 2015-summary.csv y ordena los datos de acuerdo con la columna count, además imprime el plan de como se ejecuta esto en el cluster.
- Encuentra el máximo numero de vuelos desde y hacia cualquier ubicación determinada.
- Encuentra los top 5 destinos (países) en los datos y muéstralo en la terminal

*Nota: puedes expresar tu lógica de negocio en SQL o DataFrames, Spark va a compilar esa lógica hasta un plan subyacente(que puedes ver en la explicación del plan). Con Spark SQL puedes registrar cualquier DataFrame como una tabla o vista (una tabla temporal) y consultarlo usando puro SQL, no hay diferencia entre escribir SQL o escribir código para DataFrames.