Lanzar proceso Spark

Lanzar proceso python en Spark que cuenta el número de palabras de un documento de texto:

spark-submit /usr/lib/spark/examples/src/main/python/wordcount.py /input/3285-0.txt

Es posible que de errores por la codificación del documento, por ello hay que editar el programa de python mediante el editor vi

vi /usr/lib/spark/examples/src/main/python/wordcount.py

Incluir este texto:

reload(sys)
sys.setdefaultencoding('utf8')

Para ello una vez en el editor “vi”, pulsar i (insert), posicionarse al principio del texto, copiar lo indicar, pulsar ESC, y escribir :wq (escribir y salir)

De nuevo se puede lanzar:

spark-submit /usr/lib/spark/examples/src/main/python/wordcount.py /input/3285-0.txt

Spark sql

Tendremos el fichero:

clientes.txt

100, John Smith, Austin, TX, 78727
200, Joe Johnson, Dallas, TX, 75201
300, Bob Jones, Houston, TX, 77028

400, Andy Davis, San Antonio, TX, 78227
500, James Williams, Austin, TX, 78727

Incluimos en fichero en HDFS

```
hadoop fs -put clientes.txt /input
```

Entramos en el shell de spark para lanzar órdenes:

```
spark-shell

// Creamos SQLContext para realizar conexiones SQL
val sqlContext = new org.apache.spark.sql.SQLContext(sc)

// Importar statement
import sqlContext.implicits._

// Crear la clase Clientes
case class Clientes(customer_id: Int, name: String, city: String, state: String,
zip_code: String)

// Crear DataFrame de clientes.
val dfCustomers1 = sc.textFile("/input/clientes.txt").map(_.split(",")).map(p =>
Clientes(p(0).trim.toInt, p(1), p(2), p(3), p(4))).toDF()

// Registramos el DataFrame como tabla
dfCustomers1.createOrReplaceTempView("Clientes")

// Mostrar el contenido del DataFrame
dfCustomers1.show()

// Mostrar el esquema DF
dfCustomers1.printSchema()

// Seleccionamos el nombre de clientes
dfCustomers1.select("name").show()

// Seleccionamos nombre y ciudad de clientes
dfCustomers1.select("name", "city").show()

// Seleccionamos cliente por id
dfCustomers1.filter(dfCustomers1("customer_id").equalTo(500)).show()
```

```
// Contar clientes agrupados por código postal
dfCustomers1.groupBy("zip_code").count().show()
```

Spark sql

```
 hadoop fs -put empleados.json /input
empleados.json

[{
   {"id" : "1201", "name" : "Andrés", "age" : "25"},
   {"id" : "1202", "name" : "Mabel", "age" : "28"},
   {"id" : "1203", "name" : "Juan Carlos", "age" : "39"},
   {"id" : "1204", "name" : "María del Carmen", "age" : "23"},
   {"id" : "1205", "name" : "Gemma", "age" : "23"}
}]

val jsonRDD = sc.wholeTextFiles("/input/empleados2.json").map(x => x._2)
val empleados = spark.sqlContext.read.json(jsonRDD)
empleados.printSchema
empleados.show()
empleados.select("name").show()
empleados.filter(empleados("id") > 1203).show()
empleados.filter(empleados("age") >23).show()



names.json

[{
 "Year": "2013",
 "First Name": "DAVID",
 "County": "KINGS",
 "Sex": "M",
 "Count": "272"
}, {
 "Year": "2013",
 "First Name": "JAYDEN",
```

```
 "County": "KINGS",
 "Sex": "M",
 "Count": "268"
}, {
 "Year": "2013",
 "First Name": "JAYDEN",
 "County": "QUEENS",
 "Sex": "M",
 "Count": "219"
}, {
 "Year": "2013",
 "First Name": "MOSHE",
 "County": "KINGS",
 "Sex": "M",
 "Count": "219"
}, {
 "Year": "2013",
 "First Name": "ETHAN",
 "County": "QUEENS",
 "Sex": "M",
 "Count": "216"
}]
```

```
 hadoop fs -put names.json /input
```

```
spark-shell
```

```
val jsonRDD = sc.wholeTextFiles("/input/names.json").map(x => x._2)
val namesJson = spark.sqlContext.read.json(jsonRDD)
namesJson.printSchema
namesJson.show()
namesJson.select("First Name").show()
namesJson.filter(namesJson("Count") > 250).show()
namesJson.filter(namesJson("Year") === "2013").show()
```

Búsqueda de palabras y conteo mediante Spark

```
wget http://www.truth.info/bigfiles/bible.txt.zip
```

```
unzip bible.txt.zip

from operator import add
import pandas as pd
# Mapeamos las funciones para realizar la cuenta de las palabras y generamos el
RDD.
cuenta = lineas.flatMap(lambda x: x.split(' ')) \
                .map(lambda x: (x.replace(',' , '').upper(), 1)) \
                .reduceByKey(add)

# Creamos la lista con las palabras y su respectiva frecuencia.
lista = cuenta.collect()

# Creamos un DataFrame de Pandas para facilitar el manejo de los datos.
dataframe = pd.DataFrame(lista, columns=['palabras', 'cuenta'])

# Nos quedamos solo con las palabras que hacen referencia a Dios
god = dataframe[dataframe['palabras'].apply(lambda x: x.upper() in ['GOD', 'LORD',
'JESUS', 'FATHER'])]
god.sum()
```

Ejemplo Spark Banco Mundial

```
http://jsonstudio.com/wp-content/uploads/2014/02/world_bank.zip
unzip world_bank.zip
hadoop dfs -put world_bank.json /input
spark-shell
val jsonRDD = sc.wholeTextFiles("/input/world_bank.json").map(x => x._2)
val data = spark.sqlContext.read.json(jsonRDD)
data.select("*").show
```