Pràctica 2 - Spark

Maig 2018

Índex

L'objectiu d'aquesta tutorial és aprendre a fer servir algunes de les funcions de Map Reduce que ens proporciona el sistema Spark. Continuarem utilitzant fitxers CSV (Comma Separated Values) i aprendrem algunes funcions més que ens ofereix el sistema Spark per realitzar operacions sobre els DataFrames. S'insisteix que en aquest document es fa la API del DataFrame ja que sovint és suficient per les necessitats que fan falta. L'API del DataFrame utilitza l'API dels RDD. A vegades es possible la API proporcionada per l'API del DataFrame no estigui implementada. En aquest cas caldrà utilitzar l'API dels RDD.

Al directori d'aquest tutorial disposeu del fitxer `2006.csv`, `2007.csv` i `2008.csv`. Anem a fer servir aquests tres fitxers per fer manipulacions i extreure'n informació.

En aquest tutorial es proposen exercicis que no fa falta entregar. *Aneu amb molt de compte de copiar i enganxar el codi que hi ha aqui, ja que pot donar a errors d'execució.*

1 Lectura dels fitxers i neteja dels fitxers

Començarem per llegir els tres fitxers dels quals disposem

```
>>> df2006 = spark.read.format("csv") \
        .option("header", "true") \
        .option("nullValue","NA") \
        .option("inferSchema", "true").load("2006.csv")
```

Feu el mateix pels fitxers `2007.csv` i `2008.csv`, de forma que es creïn els DatFrames `df2007` i `df2008`. Cuantes particions tenim a cada fitxer?

```
>>> df2006.rdd.getNumPartitions()
>>> df2007.rdd.getNumPartitions()
>>> df2008.rdd.getNumPartitions()
```

Podem mirar també quants elements té cada DataFrame

```
>>> df2006.count()
>>> df2007.count()
>>> df2008.count()
```

2 Operacions

Comencem per manipular els fitxers per extreure'n informació.

2.1 Unificar tots els DataFrames

Hem carregat tres DataFrames. Anem a fusionar-los en un de sol i comptem el nombre d'elements que hi ha, així com el nombre de particions associades

```
>>> df1 = df2006.union(df2007).union(df2008)
>>> df1.count()
>>> df1.rdd.getNumPartitions()
```

2.2 Escollir les columnes

En aquest cas seleccionarem aquestes columnes per la manipulació

```
>>> df2 = df1.select("Year", "Month", "Origin", "Dest", "ArrDelay", "DepDelay")
```

A partir d'ara treballarem amb `df2`. De forma similar a com hem fet a la pràctica anterior, hem de "netejar" el DataFrame eliminant totes aquelles files en què hi hagi un NA a la columna `ArrDelay` o `DepDelay`. Ho fem així...

```
>>> df3 = df2.na.drop()
```

Observeu que totes aquest operacions (llevat de `count`) són transformacions. No hem aplicat cap acció per veure el resultat de l'anàlisi que hem fet.

2.3 Retard total

De forma similar al com hem fet al primer tutorial, podem ara afegir una nova columna que sigui la suma dels retards de sortida i d'arribada.

```
>>> from pyspark.sql.functions import expr
>>> df4 = df3.withColumn("SumDelay", expr("ArrDelay + DepDelay"))
```

2.4 Emmagatzematge temporal de dades

A partir d'ara farem servir sovint el DataFrame `df4` per realitzar operacions sobre ell. Spark, cada cop que realitzem una operació sobre aquest DataFrame, construirà el graf de transformacions a realitzar des del principi, des de la primera instrucció de lectura que hem introduït.

Per qüestions d'eficiència, això no és necessari. Podem indicar-li l'Spark que emmagatzemi temporalment (sigui a memòria o a disc) el DataFrame `df4`. D'aquesta forma, totes les subsegüents operacions que es realitzin sobre `df4` seran més ràpides, ja que Spark farà les operacions a partir de `df4` i no els DataFrames originals.

```
>>> df4.cache()
```

2.5 Operacions d'agrupament

Ara volem analitzar tota la informació de què disposem. Aplicarem tècniques de MapReduce per fer-ho. Spark ens proporciona una interfície ben senzilla per fer-ho.

Quin és el retard mig, pels anys 2006, 2007 i 2008, per cada aeroport d'origen? Això és fa amb la funció `groupBy`, que ens permet agrupar les dades segons el valor d'una determinada columna. Hem de fer servir la funció `agg` per indicar quina operació volem fer sobre els valors de les columnes agrupades.

Amb la següent instrucció es crea un DataFrame amb dues columnes: la primera columna és l'aeroport d'origen i la segona columna contindrà el valor mig del retard total.

```
>>> from pyspark.sql.functions import avg
>>> df5 = df4.groupBy("Origin").agg(avg("SumDelay"))
>>> df5.show()
>>> df5.count()
```

La funció `show` només ens retorna els 20 primers elements, i observeu el nombre total d'elements (que és petit!). Si volguéssim, podríem transferir, mitjançant la funció `collect`, la informació dels DataFrame a una estructura Python (veure primer tutorial).

Podem reanomenar una columna

```
>>> df6 = df5.withColumnRenamed("avg(SumDelay)", "Average Delay")
>>> df6.show()
```

Exercici

Ordeneu els resultats per retard mig obtingut, de menor a major, i mostreu els 20 primers resultats per pantalla.

Exercici

Es proposa calcular el valor mig del retard només per a la ciutat de Nova York, agrupat per mes. Hi ha un mes en que el retard és més gran que l'altre? Proveu-ho! Quina conclusió traieu? Quins són els mesos en què hi ha més retard?

Podem agrupar per múltiples criteris. En aquest cas agrupem per any i origen

```
>>> df5 = df4.groupBy("Year", "Origin").agg(avg("SumDelay"))
>>> df5.show()
```

Exercici

Es proposa calcular el valor mig del retard només per a la ciutat de Nova York, agrupat per mes. Hi ha un mes en que el retard és més gran que l'altre? Proveu-ho! Quina conclusió traieu? Quins són els mesos en què hi ha més retard?

Anem a comptar el nombre de vols que hi ha entre els aeroports. Spark ens ofereix les funcions necessàries per fer-ho:

```
>>> from pyspark.sql.functions import lit, count
>>> df5 = df4.groupBy("Origin", "Dest").agg(count(lit(1)))
```

La funció `count(lit(1))` permet fer una suma del nombre d'elements en comptes de fer-ho sobre els valors d'una determinada columna del DataFrame.

Exercici

Quin són els vols més habituals? I entre quins aeroports es realitzen? Nova York? Boston? Los Angeles?

2.6 Funcions d'usuari

Fins ara hem aplicat funcions que ens ofereix Spark per a realitzar càlculs (per exemple, sumar dues columnes). Spark en ofereix la possibilitat de fer servir les nostres pròpies funcions per operar sobre els registres d'un DataFrame (tècnicament s'anomenen User Defined Functions). La funció a aplicar es defineix al driver, el qual la serialitzarà i l'enviarà a tots els executors perquè la puguin fer servir.

Podem, per exemple, analitzar ara el percentatge de vols que arriba amb retard a destí.

```
>>> def retard(temps):
        if (temps > 0):
            return 1
        else:
            return 0
```

Apliquem-ho al nostre DataFrame `df4` per crear una nova columna. Primer de tot cal registrar la nostra funció al sistema Spark. Amb les següents instruccions la registrem indicant que la funció retorna un sencer

```
>>> from pyspark.sql.types import IntegerType
>>> spark.udf.register("retard_spark", retard, IntegerType())
```

I l'apliquem de forma similar a com ho hem aplicat abans amb per a obtenir `SumDelay`[1].

```
>>> df5 = df4.withColumn("Retard", expr("retard_spark(SumDelay)"))
>>> df5.show()
```

Ara podem agrupar i veure el percentatge de retard dels avions

```
>>> df5.groupBy("Origin").agg(avg("Retard")).sort("avg(Retard)").show()
```

Exercici

Es proposa calcular ara el percentatge mig dels vols amb retard per a la ciutat de Nova York, agrupat per mes. Hi ha un mes en que el retard és més gran que l'altre? Proveu-ho! Quina conclusió traieu? Els resultats són congruents amb el que heu obtingut abans?

En aquest exemple hem vist com podem aplicar funcions d'usuari pròpies que s'apliquen als registres individuals del DataFrame. Spark també ofereix la possibilitat de definir funcions d'usuari a aplicar a l'hora d'agrupar dades. Això ens permet aplicar altres funcions de les que ja estan definides al Spark (`max`, `min`, `avg`, ...). Podreu trobar la forma de fer-ho a la documentació de Spark (tècnicament s'anomenen User Defined Aggregation Functions).

[1] En aplicar l'acció show Spark dóna un warning. Es desconeix la font d'aquest missatge però els resultats semblen correctes.