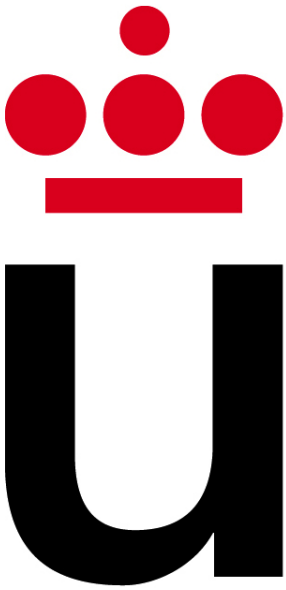
Foro de preguntas

Análisis de Big Data

Preguntas Temas 8 y 9

José Ignacio Escribano



Móstoles, 9 de abril de 2016

Índice

1. Preguntas

1.1. Hasta hace muy poco tiempo, la única opción para análisis de grafos escalable en Spark era la utilización de GraphX, cuyo catálogo completo de funciones y operaciones solo es accesible usando el lenguaje de programación Scala. Sin embargo, un nuevo elemento en Spark desde su versión 1.6 permite también acceder a estas funciones por primera vez con el lenguaje Python. Busque información al respecto e indique cómo se llama la abstracción que permite esta mejora y qué funciones ofrece.

GraphFrames [1] es la abstracción que permite el manejo de grafos, de forma similar a cómo lo hace GraphX. GraphFrames se encuentra construido en la cima de Spark DataFrames, que hace que tenga las siguientes ventajas:

- GraphFrames provee de APIs uniformes para Pyhton, Java y Scala. Por primera vez todos los algoritmos de GraphX están dsiponibles para Python y Java.
- GraphFrames permite usar las APIs de Spark SQL y DataFrames para hacer consultas.
- GraphFrames tiene total soporte de las mismas fuentes de datos que DataFrames, lo que permite leer y escribir grafos usando formatos como Parquet, JSON o CSV.

En GraphFrames, los vértices y aristas del grafo son representados como DataFrames, permitiendo almacenar información arbitraria tanto en vértices como en aristas.

Las principales funciones de la clase GraphFrame son:

- bfs: búsqueda en amplitud del grafo.
- connectedComponents: calcula las componentes conexas del grafo.
- degrees: calcula el grado de cada vértice del grafo.
- pageRank: calcula el pageRank del grafo.
- shortestPaths: calcula los caminos más cortos de un conjunto de vértices del grafo.
- stronglyConnectedComponents: calcula las componentes fuertemente conexas del grafo.
- triangleCount: cuenta el número de triángulos del grafo.

Fuentes:

[1] http://graphframes.github.io/
[2] http://graphframes.github.io/api/python/graphframes.html
[3] https://databricks.com/blog/2016/03/03/introducing-graphframes.html

1.2. La API Spark SQL dentro del framework para big data Spark proporciona una abstracción de datos llamada DataFrames: http://spark.apache.org/docs/latest/sql-programming-guide.html ¿En qué otros lenguajes de programación piensa que se ha podido inspirar esta abstracción de datos? ¿Cuáles son las principales funciones que ofrecen los DataFrames?

Un DataFrame es una colección de datos organizados por columnas con nombre. Son equivalentes a una tabla en una base de datos relacional o a un data frame de R o Pyhton, aunque con una mayor optimización que en estos dos lenguajes. Por lo que los DataFrames están inspirados en los homólogos de estos dos lenguajes.

Los DataFrames pueden ser creados a partir de distintas fuentes de datos como como archivos de datos estructurados, tablas en Apache Hive, bases de datos externas, o RDDs existentes.

La API de DataFrames se encuentra disponible para los lenguajes Scala, Java, Python y R.

Las principales características de los DataFrames son:

- Escalado desde kilobytes hasta petabytes.
- Soporte para sistemas de almacenamiento y formatos de datos de array ancho.
- Optimización y generación de código a través del optimizador SparkSQL Catalyst.

Las principales funciones de la clase DataFrame (en Pyhton) son:

- show: muestra el dataframe en pantalla. Ejemplo:

```
# Importamos la librería de Spark
from pyspark.sql import SQLContext

# df es un dataframe
df.show()
```

- printSchema: imprime el esquema de un dataframe. Ejemplo:

```
df.printSchema()
```

- select: selecciona una columna

```
# Selecciona la columna name y la imoprime en pantalla
df.select("name").show()

# Selecciona la columna name, y a la variable age le suma 1.
df.select(df["name"], df["age"]+1).show()
```

- filter: filtra información de un dataframe.

```
# Filtra los valores de la columna age que sean mayores de 21, y lo
df.filter(df["age"] > 21)
```

- goupBy: agrupa por columna.

```
# Agrupa por la columna age, que cuenta y muestra por pantalla
df.groupBy("age").count().show()
```

La lista completa se puede encontrar en [4].

La creación de un dataframe se consigue utilizando distintas funciones de la clase SQLContext. Por ejemplo, para leer desde un JSON, usamos el siguiente código.

```
# Creamos un SQLContext
sqlContext = SQLContext(sc)

# Crea el dataframe del archivo "people.json"
df = sqlContext.read.json("people.json")
```

La lista completa de todos los formatos disponibles se pueden encontrar en [5].
Fuentes:

[1] http://spark.apache.org/docs/latest/sql-programming-guide.html
[2] https://databricks.com/blog/2015/02/17/introducing-dataframes-in-sp
[3] https://databricks.com/blog/2015/06/02/statistical-and-mathematical
[4] http://spark.apache.org/docs/latest/api/python/pyspark.sql.html#pys
[5] http://spark.apache.org/docs/latest/api/python/pyspark.sql.html#pys