19-NYCTaxiCabTripSpark

August 11, 2020

1 Spark dataframes on HDFS

New York City Taxi Cab Trip

We look at the New York City Taxi Cab dataset. This includes every ride made in the city of New York since 2009.

On this website you can see the data for one random NYC yellow taxi on a single day.

On this post, you can see an analysis of this dataset. Postgres and R scripts are available on GitHub.

1.1 Loading the data

Normally we would read and load this data into memory as a Pandas dataframe. However in this case that would be unwise because this data is too large to fit in RAM.

The data can stay in the hdfs filesystem but for performance reason we can't use the csv format. The file is large (32Go) and text formatted. Data Access is very slow.

You can convert csv file to parquet with Spark.

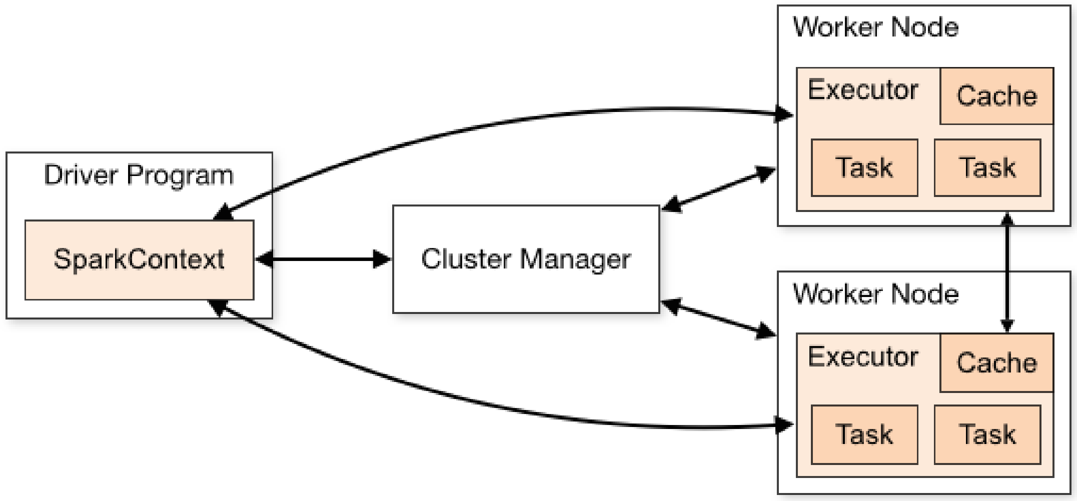
```
from pyspark.sql import SparkSession
spark = SparkSession.builder \
        .master("spark://svmass2.mass.uhb.fr:7077") \
        .config('spark.hadoop.parquet.enable.summary-metadata', 'true') \
        .config("spark.cores.max", "10") \
        .getOrCreate()
df = spark.read.csv(
"hdfs://svmass2.mass.uhb.fr:54310/user/datasets/nyc-tlc/2009/yellow_tripdata_2009-01.csv",
                    header="true",inferSchema="true")
df.write.parquet("hdfs://svmass2.mass.uhb.fr:54310/user/navaro_p/nyc-taxi/2019-01.parquet")
spark.stop()
```

To read multiple files

```
spark.read.format("csv").option("header", "true").load("../Downloads/*.csv")
```

1.2 Spark Cluster

A Spark cluster is available and described on this web interface



```
from pyspark.sql import SparkSession
spark = SparkSession.builder \
        .master('spark://svmass2.mass.uhb.fr:7077') \
        .getOrCreate()
spark
```

The SparkSession is connected to the Spark's own standalone cluster manager (It is also possible to use YARN). The manager allocate resources across applications. Once connected, Spark acquires executors on nodes in the cluster, which are processes that run computations and store data for your application. Next, it sends your application code (Python file) to the executors. Finally, tasks are sent to the executors to run.

Spark can access to files located on hdfs and it is also possible to access to local files. Example:

```
df = spark.read.parquet('file:///home/navaro_p/nyc-taxi/2016.parquet')
```

1.2.1 Exercise

- Pick a year and read and convert csv files to parquet in your hdfs homedirectory.
- **Don't run the python code inside a notebook cell**. Save a python script and launch it from a terminal instead. In Jupyter notebook you won't see any progress or information if error occurs.
- Use the `spark-submit` command shell to run your script on the cluster.
- You can control the log with

  ```
  spark.sparkContext.setLogLevel('ERROR')
  ```

  Valid log levels include: ALL, DEBUG, ERROR, FATAL, INFO, OFF, TRACE, WARN

Try your script with a single file before to do it for a whole year.

Read carefully the script given above, don't submit it as is. You have to change some part of this code

1.3 Some examples that can be run on the cluster

- Here we read the NYC taxi data files of year 2016 and select some variables.

```
columns = ['tpep_pickup_datetime', 'passenger_count', 'pickup_longitude', 'pickup_latitude', 'd
```

```
df = (spark.read.parquet('hdfs://svmass2.mass.uhb.fr:54310/user/navaro/nyc-taxi/2016.parquet')
```

- Sum the total number of passengers

  ```
  df.agg({'passenger_count': 'sum'}).collect()
  ```

- Average number of passenger per trip'

  ```
  df.agg({'passenger_count': 'avg'}).collect()
  ```

- How many trip with 0,1,2,3,...,9 passenger'

  ```
  df.groupby('passenger_count').agg({'*': 'count'}).collect()
  ```

1.4 Exercise

How well people tip based on the number of passengers in a cab. To do this you have to:

1. Remove rides with zero fare
2. Add a new column `tip_fraction` that is equal to the ratio of the tip to the fare
3. Group by the `passenger_count` column and take the mean of the `tip_fraction` column.

1.4.1 Cheat Sheets and documentation

- Spark DataFrames in Python
- Spark in Python
- https://spark.apache.org/docs/latest/api/python/pyspark.sql.html

Use the PySpark API.

- **Write a python program and use `spark-submit`**
- **Read the parquet files instead of csv files**
- **Don't forget spark.stop() at the end of the script**

1.5 Hints

- How to remove rows

  ```
  df = df.filter(df.name == 'expression')
  ```

- How to make new columns

  ```
  df = df.withColumn('var2', df.var0 + df.var1)
  ```

- How to do groupby-aggregations

  ```
  df.groupBy(df.name).agg({'column-name': 'avg'})
  ```

When you want to collect the result of your computation, finish with the `.collect()` method.

1.5.1 Exercices

1. Plot the tip as a function of the hour of day and the day of the week?
2. Investigate the `payment_type` column. See how well each of the payment types correlate with the `tip_fraction`. Did you find anything interesting? Any guesses on what the different payment types might be? If you're interested you may be able to find more information on the NYC TLC's website
3. How quickly can you get a taxi cab for a particular day of the year? How about for a particular hour of that day?