Spark SQL

Igor Yakushin
ivy2@uchicago.edu

December 27, 2019

Spark SQL

- Spark SQL is a component on top of Spark that introduced a data abstraction called DataFrames, which provides support for structured and semi-structured data.
- DataFrame is like a table distributed over the cluster similar to RDD
- One can query DataFrame using
  - Spark language
  - SQL - language for queries on relational databases
- In both cases the query optimization is used that typically results in a better performance over RDDs
- One can create DataFrame by applying transformations to other DataFrames, from the same sources as RDD: RDD, text files, json files, arrays, files in various Hadoop formats like parquet.
- Spark SQL can be interfaced with relational databases via ODBC/JDBC
- Spark SQL can use distributed Thrift store via ODBC/JDBC
- Spark SQL can use Hive store

Spark SQL: Lab 1

```
from pyspark import SparkContext, SparkConf
from pyspark.sql import SparkSession

sc = SparkContext(conf=SparkConf())
spark = SparkSession(sc)

inputData = spark.read.text(inputFile).cache()

numAs = inputData.filter(inputData.value.contains('a')).count()
numBs = inputData.filter(inputData.value.contains('b')).count()

print("Lines with a: %i, lines with b: %i" % (numAs, numBs))
```

Spark SQL: Lab 3: lab3.ipynb

We use jupyter notebook:

```
lines = spark.read.text("README.md")
from pyspark.sql.functions import *
z=lines.select(explode(split(lines.value,"\s+")).name("w"))
z.groupBy("w").count().take(10)
```

Spark SQL: Lab 4: sql.py

```
js = "data/people.json"
df = spark.read.json(js)
df.show()
df.printSchema()
df.select("name").show()
df.select(df['name'], df['age'] + 1).show()
df.filter(df['age'] > 21).show()
df.groupBy("age").count().show()

df.createOrReplaceTempView("people")
sqlDF = spark.sql("SELECT * FROM people")
sqlDF.show()
```

Spark SQL: Lab 4: rdd2df.py

```
from pyspark.sql import SparkSession
from pyspark.sql import Row

spark = SparkSession.builder.getOrCreate()
sc = spark.sparkContext

lines = sc.textFile("data/people.txt")
parts = lines.map(lambda l: l.split(","))
people = parts.map(lambda p: Row(name=p[0], age=int(p[1])))

schemaPeople = spark.createDataFrame(people)
schemaPeople.createOrReplaceTempView("people")
teens = spark.sql("SELECT name FROM people
                   WHERE age >= 13 AND age <= 19")
teenNames = teens.rdd.map(lambda p: "Name: " + p.name).collect()
for name in teenNames:
    print(name)
```

Spark SQL: Lab 4: files.py

```
from pyspark.sql import SparkSession
from os.path import abspath
warehouse_location = abspath('spark-warehouse')
spark = SparkSession \
    .builder \
    .appName("Python Spark SQL Hive integration example") \
    .config("spark.sql.warehouse.dir", warehouse_location) \
    .enableHiveSupport() \
    .getOrCreate()
df = spark.read.load("data/users.parquet")
df.select("name", "favorite_color").write.save("output/t.parquet")

df = spark.sql("SELECT * FROM parquet.`data/users.parquet`")
print(df.show())

df.write.saveAsTable("users1")
df.write.option("path", "output/hive").saveAsTable("users")
```

Spark SQL: Lab 4: hive.py

```
warehouse_location = abspath('spark-warehouse')
spark = SparkSession \
    .builder \
    .appName("Python Spark SQL Hive integration example") \
    .config("spark.sql.warehouse.dir", warehouse_location) \
    .enableHiveSupport() \
    .getOrCreate()
spark.sql("CREATE TABLE IF NOT EXISTS
           src (key INT, value STRING) USING hive")
spark.sql("LOAD DATA LOCAL INPATH
           'data/kv1.txt' INTO TABLE src")
spark.sql("SELECT * FROM src").show()

spark.sql("SELECT * FROM users1").show()
```

Spark SQL: Lab 4: hive.sql

- hive.sql:

```
select * from users1;
```

- spark-sql:

```
spark-sql --master local[1] --name "spark-sql example"
          -f hive.sql 1>spark_sql.out 2>spark_sql.err
```