732A54/TDDE31

Big Data Analytics

Introduction of Spark SQL

Huanyu Li

LINKÖPING UNIVERSITY

Overview

- Spark SQL is a module for working with structured data
- https://spark.apache.org/sql/
- Spark SQL allows
  - running queries on DataFrames with SQL
  - using built-in API functions

DataFrames

- https://archive.apache.org/dist/spark/docs/3.5.1/api/python/getting_started/quickstart_df.html
- PySpark DataFrames are implemented on top of RDDs
  - organized according to named columns
  - conceptually equivalent to tables in relational databases or data frames in R/Python
  - In this lab, you should follow the same way as BDA1 to first create RDDs from input data and then create data frames

Create a DataFrame from an RDD

- Pre-step: create a RDD
- Specify the schema
  - Inferring the schema using reflection or
  - Specifying the schema programmatically
- Register the table/data frame

Create a DataFrame from an RDD

- Pre-step: create a RDD

```
from pyspark.sql import Row, SparkSession
from pyspark.sql import functions as F

APP_NAME = "Spark SQL Demo"
spark = SparkSession.builder \
        .appName(APP_NAME) \
        .getOrCreate()

sc = spark.sparkContext

temperature_file = sc.textFile("hdfs:///user/x_antli/BDA/input/temperature-readings-small.csv")

lines = temperature_file.map(lambda line: line.split(";"))
```

Create a DataFrame from an RDD

- Specify the schema
  - Inferring the schema using reflection

```
temperatures = lines.map(lambda x: Row(year = int(x[1][0:4]), temperature = float(x[3])))

mydf = spark.createDataFrame(temperatures)

# can run queries now using API functions
max_temperature = mydf.filter("year > 2000").groupBy('year').agg(F.max('temperature').alias('MaxTemp'))

# can run queries now using SQL
#max_temperature = spark.sql("SELECT * FROM {df}", df=mydf)
```

Create a DataFrame from an RDD

- Specify the schema
  - Specifying the schema programmatically

```
temperatures = lines.map(lambda x: Row(year = int(x[1][0:4]), temperature = float(x[3])))
schema = ["year", "temperature"]

mydf = spark.createDataFrame(temperatures, schema)
mydf.createOrReplaceTempView("temperatureTable")

# can run queries now using SQL or API functions
max_temperature_df = spark.sql(”SELECT * FROM temperatureTable”)
# can run queries now using API functions
#max_temperature = df.filter("year > 2000").groupBy('year').agg(F.max('temperature').alias('MaxTemp'))
```

Access columns' values

- In PySpark it's possible to access a DataFrame's columns either by attribute (`dataframe.age`) or by indexing (`dataframe['age']`). While the former is convenient for interactive data exploration, users are highly encouraged to use the latter form, which is future proof and won't break with column names that are also attributes on the DataFrame class.

Run SQL queries

- Two ways:
  - write regular SQL over Dataframes registered as tables using
    - `spark.sql("SELECT * FROM temperatureTable")`
    - `spark.sql("SELECT * FROM {df}", df=mydf)`
  - use API methods such as select(), filter(), groupBy(), etc. (this way is recommended)
    - `filter()`
    - `groupBy()`
    - `agg()`
    - `join()`
    - etc.

Save result

- Convert a data frame into an RDD then save it as you do in BDA1
  - `df.rdd.saveAsTextFile("hdfs:///user/...")`
  - `spark.stop()`

LINKÖPING UNIVERSITY