Spark SQL程序设计基础：第一部分

董西成

2017年04月

主要内容

1. Spark SQL程序设计基础
2. Spark SQL数据源
3. 总结

主要内容

1. Spark SQL程序设计基础
2. Spark SQL数据源
3. 总结

数据集准备

- 数据集
  - http://grouplens.org/datasets/movielens/
  - **MovieLens 1M Dataset**
- 相关数据文件
  - **users.dat**
    - **UserID::Gender::Age::Occupation::Zip-code**
  - **movies.dat**
    - **MovieID::Title::Genres**
  - **ratings.dat**
    - **UserID::MovieID::Rating::Timestamp**

Spark SQL命令行访问

- 准备数据

```
mkdir /tmp/data
cat ml-1m/users.dat | tr -s "::" ',' >> /tmp/data/users.dat
```

- **./bin/spark-sql**

```
CREATE EXTERNAL TABLE user (
  userid INT,
  gender STRING,
  age INT,
  occupation STRING,
  zipcode INT
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ","
STORED AS TEXTFILE
LOCATION ’file:///tmp/data';

SELECT ＊ from USER limit 10;
```

Spark SQL命令行访问

- 使用SPARK SQL处理Hive Met Store中的表
  - ✓ 将hive-site.xml拷贝到Spark安装包的conf目录下
  - ✓ 输入./bin/spark-sql访问

Spark SQL程序设计：编写流程

- 创建SparkSession对象
  - ✓封装了spark sql执行环境信息，是所有Spark SQL程序的唯一入口
- 创建DataFrame或Dataset
  - ✓Spark SQL支持各种数据源
- 在DataFrame或Dataset之上进行转换和action
  - ✓Spark SQL提供了多种转换和action函数
- 返回结果
  - ✓保存到HDFS中，或直接打印出来

步骤1：创建SparkSession

```
val spark = SparkSession.builder.
      master("local")
      .appName("spark session example")
      .getOrCreate()

// 注意，后面所有程序片段总的spark变量均值SparkSession
// 将RDD隐式转换为DataFrame
import spark.implicits._
```

步骤2：创建DataFrame或Dataset

提供了读写各种格式数据的API

步骤3：在DataFrame或Dataset之上进行operation

Untyped transformations
(DF -> DF)

- agg
- col
- cube
- drop
- groupBy
- join
- rollup
- select
- withColumn
- …

For DataFrame & Dataset

Typed transformations
(DS -> DS)

- map
- select
- filter
- flatMap
- mapPartitions
- join
- groupByKey
- interset
- repartition
- where
- sort
- …

For Dataset

Actions
(DF/DS -> console/output)

- collect
- count
- first
- foreach
- reduce
- take
- …

For DataFrame

DataFrame与Dataset

- **DataFrame = Dataset[Row]**
  - ✓ **Row表示一行数据，比如Row=[“a”, 12, 123]**
  - ✓ **RDD、DataFrame与Dataset之间可以相互转化**
- **DataFrame**
  - ✓ **内部数据无类型，统一为Row**
  - ✓ **DataFrame是一种特殊类型的Dataset**
- ✓ **Dataset**
  - ✓ **内部数据有类型，需要由用户定义**

主要内容

1 Spark SQL程序设计基础

2 Spark SQL数据源

3 总结

DataFrame/Dataset的数据源

- RDD
  - ✓ 通过反射方式
  - ✓ 通过自定义schema方式
- json
- parquet
- jdbc
- orc

RDD → DataFrame：反射方式

1. 定义case class，作为RDD的schema
2. 直接通过RDD.toDF将RDD转换为DataFrame

```
import org.apache.spark.sql. SparkSession
import org.apache.spark.sql.Row

case class User(userID: Long, gender: String, age: Int, occupation: String, zipcode: String)

val usersRdd = sc.textFile("/tmp/ml-1m/users.dat")

val userRDD = usersRdd.map(_.split("::")).map(p => User(p(0).toLong, p(1).trim,
p(2).toInt, p(3), p(4)))
val userDataFrame = userRDD.toDF()

userDataFrame.take(10)
userDataFrame.count()
```

RDD → DataFrame：显式注入Schema

1. 定义RDD schema（由StructField/StructType构成）
2. 使用SQLContext. createDataFrame生成DF

```
import org.apache.spark.sql.{SaveMode, SparkSession, Row}
import org.apache.spark.sql.types.{StringType, StructField, StructType}

val schemaString = "userID gender age occupation zipcode"
val schema = StructType(schemaString.split(" ").map(fieldName =>
StructField(fieldName, StringType, true)))
val userRDD2 = usersRdd.map(_.split("::")).map(p => Row(p(0), p(1).trim, p(2).trim,
p(3).trim, p(4).trim))
val userDataFrame2 = sqlContext.createDataFrame(userRDD2, schema)

userDataFrame2.take(10)
userDataFrame2.count()

userDataFrame2.write.mode(SaveMode.Overwrite).json("/tmp/user.json")
userDataFrame2.write.mode(SaveMode.Overwrite).parquet("/tmp/user.parquet")
```

json → DataFrame

1. **spark.read.format("json").load(...)**
2. **sqlContext.read.json(...)**
3. **SQL**

```
val userJsonDF = spark.read.format("json").load("/tmp/user.json")
userJsonDF.take(10)
```

```
val userJsonDF2 = spark.read.json("/tmp/user.json")
userJsonDF2.take(10)
```

```
CREATE TABLE user USING json
OPTIONS
(path "/tmp/user.json")
```

parquet → DataFrame

1. **spark.read.format(”parquet").load(...)**
2. **spark.read.parquet(...)**
3. **SQL**

```
val userParquetDF = spark.read.format("parquet").load("/tmp/user.parquet")
userParquetDF.take(10)
```

```
val userParquetDF2 = spark.read.parquet("/tmp/user.parquet")
userParquetDF2.take(10)
```

```
CREATE TABLE user USING parquet
OPTIONS
(path “/tmp/user.parquet”)
```

JDBC → DataFrame

1. **spark.read.format("jdbc").options(…)**
2. **SQL**

```
export SPARK_CLASSPATH=<mysql-connector-java-5.1.26.jar>
```

```
val jdbcDF = spark.read.format("jdbc").options(
  Map(
  “url” -> “jdbc:mysql://mysql_hostname:mysql_port/testDB",
  "dbtable" -> "testTable")).load()
```

```
CREATE TABLE user USING jdbc
OPTIONS
(“jdbc:mysql://mysql_hostname:mysql_port/testDB”, “dbtable” -> “testTable”)
```

Third-party Data Sources

spark-packages.org

CSV → DataFrame

1. **Github: https://github.com/databricks/spark-csv**
2. **Maven: com.databricks:spark-csv_2.10:1.2.0**

```
val  userCsvDF = spark.read
                        .format("com.databricks.spark.csv")
                        .load(”/tmp/user.csv")
                        .toDF(”userID", ”age")
```

Avro → DataFrame

1. **Github: https://github.com/databricks/spark-avro**
2. **Maven: com.databricks:spark-avro_2.10:2.0.1**

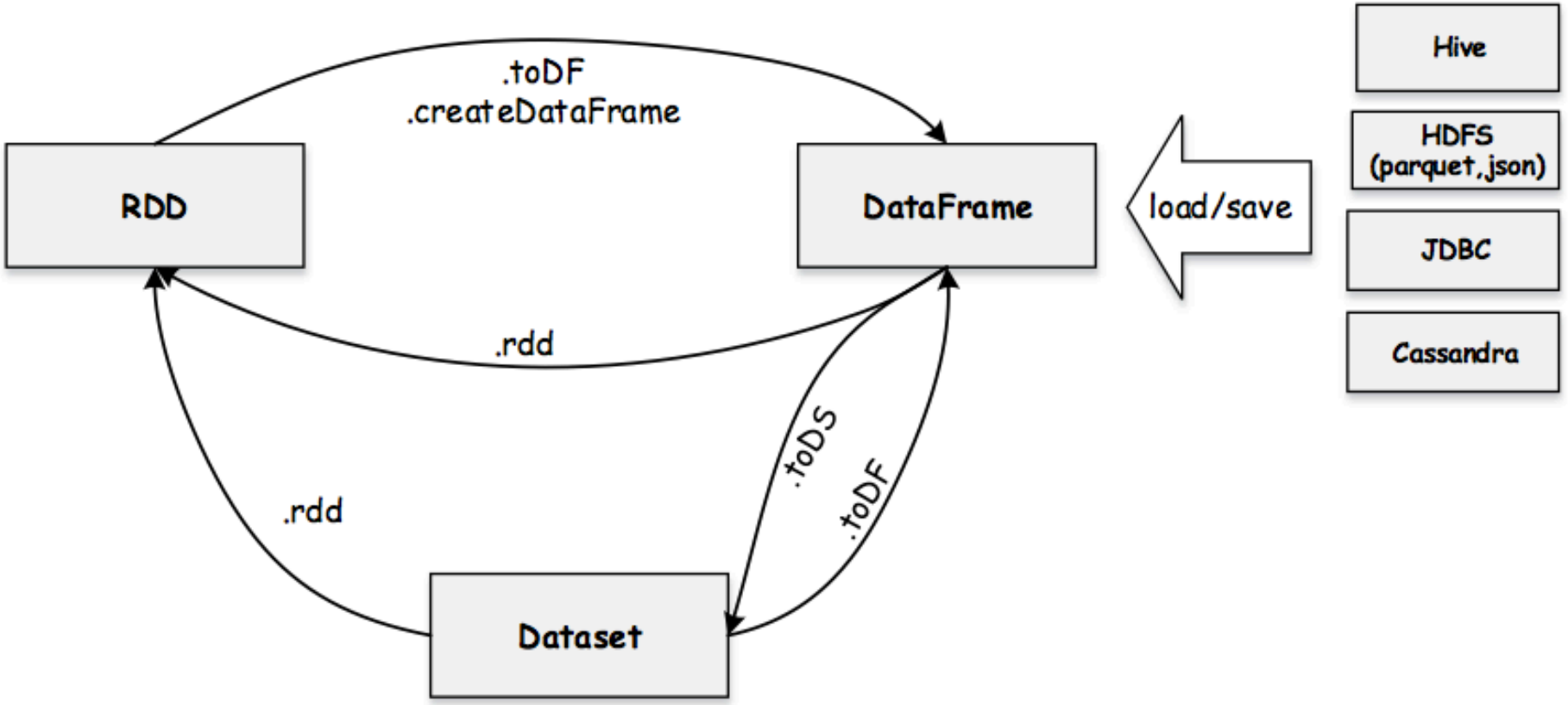
```
val df = spark.read
        .format("com.databricks.spark.avro")
        .load(”/tmp/user.avro")
```

Cassandra → DataFrame

1. **Github: https://github.com/datastax/spark-cassandra-connector**
2. **Maven: com.datastax.spark:spark-cassandra-connector_2.10:1.5.0-M1**

```
ratingsDF.write.format("org.apache.spark.sql.cassandra")
        .mode(SaveMode.Append)
        .options(Map("keyspace"->”reco","table"->”users"))
        .save()
```

RDD、DataFrame与Dataset的关系




```
val df = spark.read.parquet("...") // DataFrame
val ds = df.as[Person] // DataFrame → Dataset
val df2 = ds.toDF() / Dataset → DataFrame
val rdd1 = ds.rdd // Dataset → RDD
val rdd2 = df.rdd // DataFrame → RDD
val newDs = Seq(Person("Andy", 32)).toDS() // Seq → DS
```

hadoop123：董西成的微信公众号

专注于Hadoop/spark等大数据相关技术的分享

小象科技
ChinaHadoop.cn

联系我们：

- 新浪微博：ChinaHadoop
- 微信公号：ChinaHadoop

让你的数据产生价值！