SparkR(R on Spark)编程指南

概论

SparkR是一个R语言包，它提供了轻量级的方式使得可以在R语言中使用Apache Spark。在Spark 1.4中，SparkR实现了分布式的data frame，支持类似查询、过滤以及聚合的操作（类似于R中的data frames：dplyr），但是这个可以操作大规模的数据集。

SparkR DataFrames

DataFrame是数据组织成一个带有列名称的分布式数据集。在概念上和关系型数据库中的表类似，或者和R语言中的data frame类似，但是这个提供了很多的优化措施。构造DataFrame的方式有很多：可以通过结构化文件中构造；可以通过Hive中的表构造；可以通过外部数据库构造或者是通过现有R的data frame构造等等。

从SparkContext和SQLContext开始

SparkContext是SparkR的切入点，它使得你的R程序和Spark集群互通。你可以通过sparkR.init来构建SparkContext，然后可以传入类似于应用程序名称的选项给它。如果想使用DataFrames，我们得创建SQLContext，这个可以通过SparkContext来构造。如果你使用SparkR shell，SQLContext 和SparkContext会自动地构建好。

```
sc <- sparkR.init()
sqlContext <- sparkRSQL.init(sc)
```

创建DataFrames

如果有SQLContext实例，那么应用程序就可以通过本地的R data frame（或者是Hive表；或者是其他数据源）来创建DataFrames。下面将详细地介绍。

通过本地data frame构造

最简单地创建DataFrames是将R的data frame转换成SparkR DataFrames，我们可以通过createDataFrame来创建，并传入本地R的data frame以此来创建SparkR DataFrames，下面例子就是这种方法：

```
df <- createDataFrame(sqlContext, faithful)

# Displays the content of the DataFrame to stdout
head(df)
##  eruptions waiting
##1     3.600      79
##2     1.800      54
##3     3.333      74
```

通过Data Sources构造

通过DataFrame接口，SparkR支持操作多种数据源，本节将介绍如何通过Data Sources提供的方法来加载和保存数据。你可以阅读Spark

SQL编程指南来了解更多的options选项.

Data Sources中创建DataFrames的一般方法是使用read.df，这个方法需要传入SQLContext，需要加载的文件路径以及数据源的类型。SparkR内置支持读取JSON和Parquet文件，而且通过Spark Packages你可以读取很多类型的数据，比如CSV和Avro文件。

下面是介绍如何JSON文件，注意，这里使用的文件不是典型的JSON文件。每行文件必须包含一个分隔符、自包含有效的JSON对象：

```
people <- read.df(sqlContext, "./examples/src/main/resources/people.json", "json")
head(people)
##  age    name
##1  NA Michael
##2  30    Andy
##3  19  Justin

# SparkR automatically infers the schema from the JSON file
printSchema(people)
# root
#  |-- age: integer (nullable = true)
#  |-- name: string (nullable = true)
```

Data sources API还可以将DataFrames保存成多种的文件格式，比如我们可以通过write.df将上面的DataFrame保存成Parquet文件：

```
write.df(people, path="people.parquet", source="parquet", mode="overwrite")
```

通过Hive tables构造

我们也可以通过Hive表来创建SparkR DataFrames，为了达到这个目的，我们需要创建HiveContext，因为我们可以通过它来访问Hive MetaStore中的表。注意，Spark内置就对Hive提供了支持，SQLContext和HiveContext的区别可以参见SQL编程指南。

```
# sc is an existing SparkContext.
hiveContext <- sparkRHive.init(sc)

sql(hiveContext, "CREATE TABLE IF NOT EXISTS src (key INT, value STRING)")
sql(hiveContext, "LOAD DATA LOCAL INPATH 'examples/src/main/resources/kv1.txt' INTO TABLE src")
```

```
# Queries can be expressed in HiveQL.
results <- hiveContext.sql("FROM src SELECT key, value")

# results is now a DataFrame
head(results)
##  key   value
## 1 238 val_238
## 2  86  val_86
## 3 311 val_311
```

DataFrame的相关操作

SparkR DataFrames中提供了大量操作结构化数据的函数，这里仅仅列出其中一小部分，详细的API可以参见SparkR编程的API文档。

选择行和列

```
# Create the DataFrame
df <- createDataFrame(sqlContext, faithful)

# Get basic information about the DataFrame
df
## DataFrame[eruptions:double, waiting:double]

# Select only the "eruptions" column
head(select(df, df$eruptions))
##  eruptions
##1     3.600
##2     1.800
##3     3.333

# You can also pass in column name as strings
head(select(df, "eruptions"))

# Filter the DataFrame to only retain rows with wait times shorter than 50 mins
head(filter(df, df$waiting < 50))
##  eruptions waiting
##1     1.750      47
##2     1.750      47
##3     1.867      48
```

Grouping和Aggregation

```
# We use the `n` operator to count the number of times each waiting time appears
head(summarize(groupBy(df, df$waiting), count = n(df$waiting)))
##  waiting count
##1      81    13
##2      60     6
##3      68     1

# We can also sort the output from the aggregation to get the most common waiting times
waiting_counts <- summarize(groupBy(df, df$waiting), count = n(df$waiting))
head(arrange(waiting_counts, desc(waiting_counts$count)))

##   waiting count
##1      78    15
##2      83    14
##3      81    13
```

列上面的操作

SparkR提供了大量的函数用于直接对列进行数据处理的操作。

```
# Convert waiting time from hours to seconds.
# Note that we can assign this to a new column in the same DataFrame
df$waiting_secs <- df$waiting * 60
head(df)
##  eruptions waiting waiting_secs
##1     3.600      79        4740
##2     1.800      54        3240
##3     3.333      74        4440
```

在SparkR中运行SQL查询

SparkR DataFrame也可以在Spark SQL中注册成临时表。将DataFrame 注册成表可以允许我们在数据集上运行SQL查询。sql函数可以使得我们直接运行SQL查询，而且返回的结构是DataFrame。

```
# Load a JSON file
```

```
people <- read.df(sqlContext, "./examples/src/main/resources/people.json", "json")

# Register this DataFrame as a table.
registerTempTable(people, "people")

# SQL statements can be run by using the sql method
teenagers <- sql(sqlContext, "SELECT name FROM people WHERE age >= 13 AND age <= 19")
head(teenagers)
##    name
##1 Justin
```

本文翻译自SparkR官方文档，这个是Spark 1.4中才有的，不过他还没发布，不过可以看这里http://people.apache.org/~pwendell/spark-releases/spark-1.4.0-rc4-docs/sparkr.html