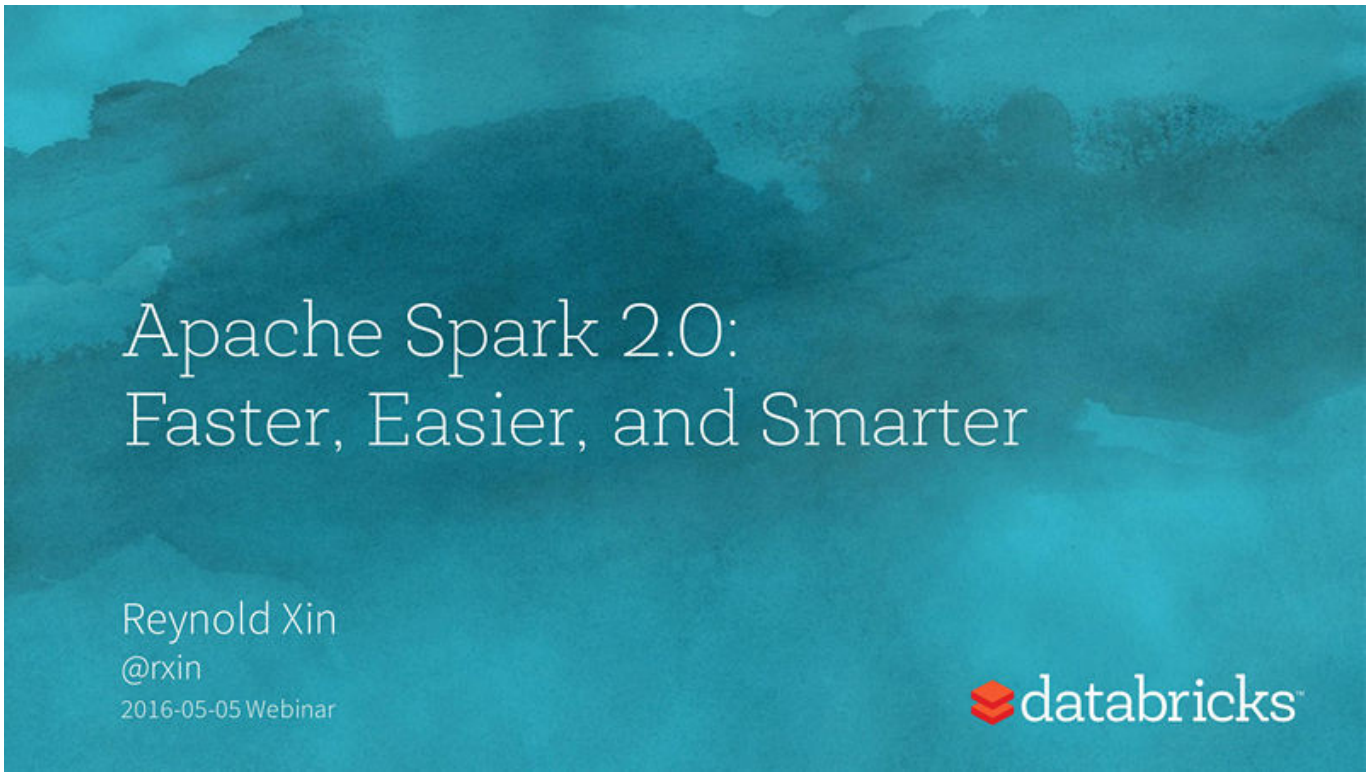
如何在Apache Spark 2.0中使用SparkSession

Apache Spark 2.0引入了SparkSession，其为用户提供了一个统一的切入点来使用Spark的各项功能，并且允许用户通过它调用DataFrame和Dataset相关API来编写Spark程序。最重要的是，它减少了用户需要了解的一些概念，使得我们可以很容易地与Spark交互。

本文我们将介绍在Spark 2.0中如何使用SparkSession。更多关于SparkSession的文章请参见：《SparkSession：新的切入点》、《Spark 2.0介绍：创建和使用相关API》、《Apache Spark 2.0.0正式发布及其功能介绍》



探索SparkSession统一的功能

首先，我们介绍一个简单的Spark应用案例：SparkSessionZipsExample，其从JSON文件中读取邮政编码，并且通过DataFrame API进行一些分析，之后使用Spark SQL进行一些查询，这些操作并没有使用到SparkContext, SQLContext 或者HiveContext。

创建SparkSession

在2.0版本之前，与Spark交互之前必须先创建SparkConf和SparkContext，代码如下：

```
//set up the spark configuration and create contexts
```

```
val sparkConf = new SparkConf().setAppName("SparkSessionZipsExample").setMaster("local")
// your handle to SparkContext to access other context like SQLContext
val sc = new SparkContext(sparkConf).set("spark.some.config.option", "some-value")
val sqlContext = new org.apache.spark.sql.SQLContext(sc)
```

然而在Spark 2.0中，我们可以通过SparkSession来实现同样的功能，而不需要显式地创建SparkConf, SparkContext 以及 SQLContext，因为这些对象已经封装在SparkSession中。使用生成器的设计模式(builder design pattern)，如果我们没有创建SparkSession对象，则会实例化出一个新的SparkSession对象及其相关的上下文。

```
// Create a SparkSession. No need to create SparkContext
// You automatically get it as part of the SparkSession
val warehouseLocation = "file:${system:user.dir}/spark-warehouse"
val spark = SparkSession
  .builder()
  .appName("SparkSessionZipsExample")
  .config("spark.sql.warehouse.dir", warehouseLocation)
  .enableHiveSupport()
  .getOrCreate()
```

到现在我们可以使用上面创建好的spark对象，并且访问其public方法。

配置Spark运行相关属性

一旦我们创建好了SparkSession，我们就可以配置Spark运行相关属性。比如下面代码片段我们修改了已经存在的运行配置选项。

```
//set new runtime options
spark.conf.set("spark.sql.shuffle.partitions", 6)
spark.conf.set("spark.executor.memory", "2g")
//get all settings
val configMap:Map[String, String] = spark.conf.getAll()
```

获取Catalog元数据

通常我们想访问当前系统的Catalog元数据。SparkSession提供了catalog实例来操作metastore。这些方法放回的都是Dataset类型的，所有我们可以使用Dataset相关的API来访问其中的数据。如下代码片段，我们展示了所有的表并且列出当前所有的数据库：

```
//fetch metadata data from the catalog
scala> spark.catalog.listDatabases.show(false)
+-------------+--------------------+-----------------------------------------------------+
|name         |description         |locationUri                                          |
+-------------+--------------------+-----------------------------------------------------+
|default      |Default Hive database|hdfs://iteblogcluster/user/iteblog/hive/warehouse    |
+-------------+--------------------+-----------------------------------------------------+

scala> spark.catalog.listTables.show(false)
+-------------------------------------+--------+-----------+---------+-----------+
|name                                 |database|description|tableType|isTemporary|
+-------------------------------------+--------+-----------+---------+-----------+
|iteblog                              |default |null       |MANAGED  |false      |
|table2                               |default |null       |EXTERNAL |false      |
|test                                 |default |null       |MANAGED  |false      |
+-------------------------------------+--------+-----------+---------+-----------+
```

创建Dataset和Dataframe

使用SparkSession APIs创建 DataFrames 和 Datasets的方法有很多，其中最简单的方式就是使用spark.range方法来创建一个Dataset。当我们学习如何操作Dataset API的时候，这个方法非常有用。操作如下：

```
scala> val numDS = spark.range(5, 100, 5)
numDS: org.apache.spark.sql.Dataset[Long] = [id: bigint]

scala> numDS.orderBy(desc("id")).show(5)
+---+
| id|
+---+
| 95|
| 90|
| 85|
| 80|
| 75|
+---+
only showing top 5 rows

scala> numDS.describe().show()
```

```
+-------+-----------------+
|summary|               id|
+-------+-----------------+
|  count|               19|
|   mean|             50.0|
| stddev|28.136571693556885|
|    min|                5|
|    max|               95|
+-------+-----------------+


scala> val langPercentDF = spark.createDataFrame(List(("Scala", 35),
     | ("Python", 30), ("R", 15), ("Java", 20)))
langPercentDF: org.apache.spark.sql.DataFrame = [_1: string, _2: int]

scala> val lpDF = langPercentDF.withColumnRenamed("_1", "language").withColumnRenamed(
"_2", "percent")
lpDF: org.apache.spark.sql.DataFrame = [language: string, percent: int]

scala> lpDF.orderBy(desc("percent")).show(false)
+--------+-------+
|language|percent|
+--------+-------+
|Scala   |35     |
|Python  |30     |
|Java    |20     |
|R       |15     |
+--------+-------+
```

使用SparkSession API读取JSON数据

我们可以使用SparkSession来读取JSON、CVS或者TXT文件，甚至是读取parquet表。比如在下面代码片段里面，我将读取邮编数据的JSON文件，并且返回DataFrame对象：

```
// read the json file and create the dataframe
scala> val jsonFile = "/user/iteblog.json"
jsonFile: String = /user/iteblog.json
zipsDF: org.apache.spark.sql.DataFrame = [_id: string, city: string ... 3 more fields]

scala> zipsDF.filter(zipsDF.col("pop") > 40000).show(10, false)
+-----+----------+---------------------+-----+-----+
|_id  |city      |loc                  |pop  |state|
+-----+----------+---------------------+-----+-----+
```

```
|01040|HOLYOKE   |[-72.626193, 42.202007]|43704|MA   |
|01085|MONTGOMERY|[-72.754318, 42.129484]|40117|MA   |
|01201|PITTSFIELD|[-73.247088, 42.453086]|50655|MA   |
|01420|FITCHBURG |[-71.803133, 42.579563]|41194|MA   |
|01701|FRAMINGHAM|[-71.425486, 42.300665]|65046|MA   |
|01841|LAWRENCE  |[-71.166997, 42.711545]|45555|MA   |
|01902|LYNN      |[-70.941989, 42.469814]|41625|MA   |
|01960|PEABODY   |[-70.961194, 42.532579]|47685|MA   |
|02124|DORCHESTER|[-71.072898, 42.287984]|48560|MA   |
|02146|BROOKLINE |[-71.128917, 42.339158]|56614|MA   |
+-----+----------+-----------------------+-----+-----+
only showing top 10 rows
```

在SparkSession中还用Spark SQL

通过SparkSession我们可以访问Spark SQL中所有函数，正如你使用SQLContext访问一样。下面代码片段中，我们创建了一个表，并在其中使用SQL查询：

```
// Now create an SQL table and issue SQL queries against it without
// using the sqlContext but through the SparkSession object.
// Creates a temporary view of the DataFrame
scala> zipsDF.createOrReplaceTempView("zips_table")

scala> zipsDF.cache()
res3: zipsDF.type = [_id: string, city: string ... 3 more fields]

scala> val resultsDF = spark.sql("SELECT city, pop, state, _id FROM zips_table")
resultsDF: org.apache.spark.sql.DataFrame = [city: string, pop: bigint ... 2 more fields]

scala> resultsDF.show(10)
+------------+-----+-----+-----+
|        city|  pop|state|  _id|
+------------+-----+-----+-----+
|     AGAWAM|15338|   MA|01001|
|    CUSHMAN|36963|   MA|01002|
|      BARRE| 4546|   MA|01005|
| BELCHERTOWN|10579|   MA|01007|
|   BLANDFORD| 1240|   MA|01008|
|   BRIMFIELD| 3706|   MA|01010|
|    CHESTER| 1688|   MA|01011|
|CHESTERFIELD|  177|   MA|01012|
|    CHICOPEE|23396|   MA|01013|
|    CHICOPEE|31495|   MA|01020|
```

```
+------------+-----+-----+-----+
only showing top 10 rows
```

使用SparkSession读写Hive表

下面我们将使用SparkSession创建一个Hive表，并且对这个表进行一些SQL查询，正如你使用HiveContext一样：

```
scala> spark.sql("DROP TABLE IF EXISTS iteblog_hive")
res5: org.apache.spark.sql.DataFrame = []

scala> spark.table("zips_table").write.saveAsTable("iteblog_hive")
16/08/24 21:52:59 WARN HiveMetaStore: Location: hdfs://iteblogcluster/user/iteblog/hive/warehouse/iteblog_hive specified for non-external table:iteblog_hive

scala> val resultsHiveDF = spark.sql("SELECT city, pop, state, _id FROM iteblog_hive WHERE pop > 40000")
resultsHiveDF: org.apache.spark.sql.DataFrame = [city: string, pop: bigint ... 2 more fields]

scala> resultsHiveDF.show(10)
+----------+-----+-----+-----+
|      city|  pop|state|  _id|
+----------+-----+-----+-----+
|   HOLYOKE|43704|   MA|01040|
|MONTGOMERY|40117|   MA|01085|
|PITTSFIELD|50655|   MA|01201|
| FITCHBURG|41194|   MA|01420|
|FRAMINGHAM|65046|   MA|01701|
|  LAWRENCE|45555|   MA|01841|
|      LYNN|41625|   MA|01902|
|   PEABODY|47685|   MA|01960|
|DORCHESTER|48560|   MA|02124|
| BROOKLINE|56614|   MA|02146|
+----------+-----+-----+-----+
only showing top 10 rows
```

正如你所见，你使用DataFrame API, Spark SQL 以及 Hive查询的结果都一样。

本文翻译自：https://databricks.com/blog/2016/08/15/how-to-use-sparksession-in-apache-spark-2-0.html

资源下载

上面使用到的iteblog.json文件可以到这里下载。