Spark - Base

Spark都有什么

驱动器 driver

- 是一个物理机器的进程
- 控制程序进程
- 控制、维护Spark

执行器 executor

- 一个进程
- 负责执行driver分配的任务，并报告其状态
- 每个Spark应用程序都有自己的executor

集群管理器 Master和Worker

- 管理物理机器而不是进程
- 简单内置管理器、Apache Mesos、Hadoop Yarn

运行模式

- 本地模式：单机运行Spark全部进程
- 客户端模式：本地机运行驱动器，集群维护执行器
- 集群模式：集群管理器负责维护所有进程，本地机将Python脚本上传集群执行

生命周期

1，创建SparkSession（Spark2.X以后才有）
  - SparkContext和SQLContext被封装在SparkSession（2.X），应该不需要用他们

2，定义各类任务

3，action执行任务，任务会串行执行

- 任务串行执行时，shuffle操作时，Spark会把其结果写入磁盘做持久化保存，当另一任务需要shuffle结果时，不需要执行源一端的shuffle

数据集模式 Schema

Spark.schema返回了数据集各列的数据类型
一个模式由多个字段构成的 StructType，每字段（列）为一个StructField
StructField里包含：列名，数据类型

```
# in Python
spark.read.format("json").load("/data/flight-data/json/2015-summary.json").schema
```

Python中返回以下内容：

```
org.apache.spark.sql.types.StructType = ...
StructType(StructField(DEST_COUNTRY_NAME,StringType,true),
StructField(ORIGIN_COUNTRY_NAME,StringType,true),
StructField(count,LongType,true))
```

数据源

主要包括：CSV、JSON、Parquet、ORC、JDBC连接、纯文本

CSV文件不支持复杂类型，Parquet和ORC支持

ORC与Parquet的区别

Parquet：

- 面向列的数据存储格式，更节省空间，支持按列读取，而非整个文件读取
- 是Spark默认格式
- ORC针对Hive进行优化，Parquet针对Spark优化
- ORC在读取时没有可选项

并行读写文件

Spark多个执行器可同时读取多个文件，每个文件作为DF的一个partition，并行读取

```
csvFile.repartition(5).write.format("csv").save("/tmp/multiple.csv")
```

它会生成包含五个文件的文件夹，调用ls命令就可以查看到：

```
ls /tmp/multiple.csv

/tmp/multiple.csv/part-00000-767df509-ec97-4740-8e15-4e173d365a8b.csv
/tmp/multiple.csv/part-00001-767df509-ec97-4740-8e15-4e173d365a8b.csv
/tmp/multiple.csv/part-00002-767df509-ec97-4740-8e15-4e173d365a8b.csv
/tmp/multiple.csv/part-00003-767df509-ec97-4740-8e15-4e173d365a8b.csv
/tmp/multiple.csv/part-00004-767df509-ec97-4740-8e15-4e173d365a8b.csv
```

数据划分

- 可根据列的值对数据进行划分，保存为多个文件，文件名就是列值
- 读取时，只读取有用的部分，跳过大量数据

数据分桶

根据桶ID进行划分，不同ID数据放入不同物理分区，减少Shuffle阶段的数据传输

Spark托管表

Spark读取磁盘文件时，创建的是非托管表，非托管表Spark只管理元数据

Spark调用saveAsTable创建数据表时，创建了托管表

低级API

RDD: 处理分布式数据

分发和处理分布式的共享变量：广播变量和累加器

RDD

- 只读，新操作只能生成新的RDD
- RDD只记录了Java、Python等对象，而DF、DataSet记录了结构化的数据
- 使用Python操作RDD开销极大

RDD包含什么（内部属性）

- 数据分片列表
- 作用在不同数据分片的计算函数
- 描述与其他RDD的依赖关系列表
- 为key-value RDD配置的分片方法（Hash或Range）
- 数据优先位置列表，指定每个片处理位置偏好（HDFS中的节点）

数据操作

- RDD只读，产生其他RDD智能创建
- 可以删除重复项的重复部分，
- （过滤） fifter、filler，取数take

缓存

可通过修改配置（好像修改matplotlib）指定单词的任意存储级别，包括但不限于：内存，硬盘，堆外内存

检查点

Checkpoint：在某一步将RDD保存到硬盘，可访问RDD的中间结果，无需重头计算

CheckpiontDir：通过配置修改的检查保存位置

```
spark.sparkContext.setCheckpointDir("/some/path/for/checkpointing")
words.checkpoint()
```

共享变量

广播变量 broadcast

- 方便在集群有效的共享只读变量
- 在多个作业中共享变量
- broadcast一般用.value的方式传输数据，避免频繁的序列化反序列化

累加器

- 一个计数变量
- 按行更新
- 仅在RDD.action()时更新
- 重新执行的任务不会更新累加器
- 累加器函数可以自定义，配合foreach（类似apply）使用

累加器命名

```
// in Scala
val accChina = new LongAccumulator
val accChina2 = spark.sparkContext.longAccumulator("China")
spark.sparkContext.register(accChina, "China")
```

我们可以传递给LongAccumulator函数一个字符串值作为累加器的名称，或者将该字符串作为第二个参数传递到register函数中。命名累加器将显示在Spark用户界面（Spark UI）中，而未命名的累加器不会。
