Spark内核是由Scala语言开发的，因此使用Scala语言开发Spark应用程序是自然而然的事情。如果你对Scala语言还不太熟悉，可以阅读网络教程[A Scala Tutorial for Java Programmers](http://scalachina.com/node/16" \t "_blank)或者相关[Scala书籍](http://search.dangdang.com/?key=scala" \t "_blank)进行学习。

本文将介绍3个Scala Spark编程实例，分别是WordCount、TopK和SparkJoin，分别代表了Spark的三种典型应用。

1. WordCount编程实例

WordCount是一个最简单的分布式应用实例，主要功能是统计输入目录中所有单词出现的总次数，编写步骤如下：

步骤1：创建一个SparkContext对象，该对象有四个参数：Spark master位置、应用程序名称，Spark安装目录和jar存放位置，对于Spark On YARN而言，最重要的是前两个参数，第一个参数指定为“yarn-standalone”，第二个参数是自定义的字符串，举例如下：

val sc = new SparkContext(args(0), "WordCount",

System.getenv("SPARK\_HOME"), Seq(System.getenv("SPARK\_TEST\_JAR")))

步骤2：读取输入数据。我们要从HDFS上读取文本数据，可以使用SparkContext中的textFile函数将输入文件转换为一个RDD，该函数采用的是Hadoop中的TextInputFormat解析输入数据，举例如下：

val textFile = sc.textFile(args(1))

当然，Spark允许你采用任何Hadoop InputFormat，比如二进制输入格式SequenceFileInputFormat，此时你可以使用SparkContext中的hadoopRDD函数，举例如下：

val inputFormatClass = classOf[SequenceFileInputFormat[Text,Text]]

var hadoopRdd = sc.hadoopRDD(conf, inputFormatClass, classOf[Text], classOf[Text])

或者直接创建一个HadoopRDD对象：

var hadoopRdd = new HadoopRDD(sc, conf,

classOf[SequenceFileInputFormat[Text,Text, classOf[Text], classOf[Text])

步骤3：通过RDD转换算子操作和转换RDD，对于WordCount而言，首先需要从输入数据中每行字符串中解析出单词，然后将相同单词放到一个桶中，最后统计每个桶中每个单词出现的频率，举例如下：

val result = hadoopRdd.flatMap{

case(key, value) => value.toString().split("\\s+");

}.map(word => (word, 1)). reduceByKey (\_ + \_)

其中，flatMap函数可以将一条记录转换成多条记录（一对多关系），map函数将一条记录转换为另一条记录（一对一关系），reduceByKey函数将key相同的数据划分到一个桶中，并以key为单位分组进行计算，这些函数的具体含义可参考：[Spark Transformation](http://spark.incubator.apache.org/docs/latest/scala-programming-guide.html" \l "rdd-operations" \t "_blank)。

步骤4：将产生的RDD数据集保存到HDFS上。可以使用SparkContext中的saveAsTextFile哈数将数据集保存到HDFS目录下，默认采用Hadoop提供的TextOutputFormat，每条记录以“（key,value）”的形式打印输出，你也可以采用saveAsSequenceFile函数将数据保存为SequenceFile格式等，举例如下：

result.saveAsSequenceFile(args(2))

当然，一般我们写Spark程序时，需要包含以下两个头文件：

import org.apache.spark.\_

import SparkContext.\_

WordCount完整程序已在“[Apache Spark学习：利用Eclipse构建Spark集成开发环境](http://dongxicheng.org/framework-on-yarn/spark-eclipse-ide/)”一文中进行了介绍，在次不赘述。

需要注意的是，指定输入输出文件时，需要指定hdfs的URI，比如输入目录是hdfs://hadoop-test/tmp/input，输出目录是hdfs://hadoop-test/tmp/output，其中，“hdfs://hadoop-test”是由Hadoop配置文件core-site.xml中参数fs.default.name指定的，具体替换成你的配置即可。

2. TopK编程实例

TopK程序的任务是对一堆文本进行词频统计，并返回出现频率最高的K个词。如果采用MapReduce实现，则需要编写两个作业：WordCount和TopK，而使用Spark则只需一个作业，其中WordCount部分已由前面实现了，接下来顺着前面的实现，找到Top K个词。注意，本文的实现并不是最优的，有很大改进空间。

步骤1：首先需要对所有词按照词频排序，如下：

val sorted = result.map {

case(key, value) => (value, key); //exchange key and value

}.sortByKey(true, 1)

步骤2：返回前K个：

val topK = sorted.top(args(3).toInt)

步骤3：将K各词打印出来：

topK.foreach(println)

注意，对于应用程序标准输出的内容，YARN将保存到Container的stdout日志中。在YARN中，每个Container存在三个日志文件，分别是stdout、stderr和syslog，前两个保存的是标准输出产生的内容，第三个保存的是log4j打印的日志，通常只有第三个日志中有内容。

本程序完整代码、编译好的jar包和运行脚本可以从[这里下载](http://pan.baidu.com/share/link?shareid=3350530595&uk=2248644272" \t "_blank)。下载之后，按照“[Apache Spark学习：利用Eclipse构建Spark集成开发环境](http://dongxicheng.org/framework-on-yarn/spark-eclipse-ide/)”一文操作流程运行即可。

3. SparkJoin编程实例

在推荐领域有一个著名的开放测试集是movielens给的，下载链接是：[http://grouplens.org/datasets/movielens/](http://grouplens.org/datasets/movielens/" \t "_blank)，该测试集包含三个文件，分别是ratings.dat、sers.dat、movies.dat，具体介绍可阅读：[README.txt](http://files.grouplens.org/datasets/movielens/ml-1m-README.txt" \t "_blank)，本节给出的SparkJoin实例则通过连接ratings.dat和movies.dat两个文件得到平均得分超过4.0的电影列表，采用的数据集是：[ml-1m](http://files.grouplens.org/datasets/movielens/ml-1m.zip)。程序代码如下：

import org.apache.spark.\_

import SparkContext.\_

object SparkJoin {

def main(args: Array[String]) {

if (args.length != 4 ){

println("usage is org.test.WordCount <master> <rating> <movie> <output>")

return

}

val sc = new SparkContext(args(0), "WordCount",

System.getenv("SPARK\_HOME"), Seq(System.getenv("SPARK\_TEST\_JAR")))

// Read rating from HDFS file

val textFile = sc.textFile(args(1))

//extract (movieid, rating)

val rating = textFile.map(line => {

val fileds = line.split("::")

(fileds(1).toInt, fileds(2).toDouble)

})

val movieScores = rating

.groupByKey()

.map(data => {

val avg = data.\_2.sum / data.\_2.size

(data.\_1, avg)

})

// Read movie from HDFS file

val movies = sc.textFile(args(2))

val movieskey = movies.map(line => {

val fileds = line.split("::")

(fileds(0).toInt, fileds(1))

}).keyBy(tup => tup.\_1)

// by join, we get <movie, averageRating, movieName>

val result = movieScores

.keyBy(tup => tup.\_1)

.join(movieskey)

.filter(f => f.\_2.\_1.\_2 > 4.0)

.map(f => (f.\_1, f.\_2.\_1.\_2, f.\_2.\_2.\_2))

result.saveAsTextFile(args(3))

}

}

你可以从[这里下载](http://pan.baidu.com/share/link?shareid=3350530595&uk=2248644272" \t "_blank)代码、编译好的jar包和运行脚本。

这个程序直接使用Spark编写有些麻烦，可以直接在[Shark](https://github.com/amplab/shark/wiki" \t "_blank)上编写HQL实现，Shark是基于Spark的类似Hive的交互式查询引擎，具体可参考：[Shark](https://github.com/amplab/shark/wiki" \t "_blank)。

4. 总结

Spark 程序设计对Scala语言的要求不高，正如Hadoop程序设计对Java语言要求不高一样，只要掌握了最基本的语法就能编写程序，且常见的语法和表达方式是很少的。通常，刚开始仿照官方实例编写程序，包括[Scala、Java和Python](http://spark.incubator.apache.org/examples.html)三种语言实例。

**原创文章，转载请注明：** 转载自[董的博客](http://dongxicheng.org/)

**本文链接地址:** <http://dongxicheng.org/framework-on-yarn/spark-scala-writing-application/>

**作者：**[**Dong**](http://dongxicheng.org/about/)，作者介绍：<http://dongxicheng.org/about/>

**本博客的文章集合:**<http://dongxicheng.org/recommend/>