Spark快速入门指南(Quick Start Spark)

这个文档只是简单的介绍如何快速地使用Spark。在下面的介绍中我将介绍如何通过Spark的交互式shell来使用API。

Basics

Spark shell提供一种简单的方式来学习它的API，同时也提供强大的方式来交互式地分析数据。Spark shell支持Scala和Python。可以通过以下方式进入到Spark shell中。

```
# 本文原文地址：https://www.iteblog.com/archives/1040.html
# 过往记忆，大量关于Hadoop、Spark等个人原创技术博客

./bin/spark-shell
```

Spark的一个基本抽象概念就是RDD，RDDs可以通过Hadoop InputFormats或者通过其他的RDDs通过transforming来得到。下面的例子是通过加载SPARK_HOME目录下的README文件来构建一个新的RDD

```
scala> val textFile = sc.textFile("file:///spark-bin-0.9.1/README.md")
textFile:org.apache.spark.rdd.RDD[String]=MappedRDD[3]at textFile at <console>:1
```

RDDs提供actions操作，通过它可以返回值；同时还提供transformations操作，通过它可以返回一个新的RDD的引用。如下：

```
scala> textFile.count() // Number of items in this RDD
res1: Long = 108

scala> textFile.first() // First item in this RDD
res2: String = # Apache Spark
```

我们再试试transformations操作，下面的例子中我们通过使用filter transformation来一个新的RDD：

```
scala> val linesWithSpark = textFile.filter(line => line.contains("Spark"))
linesWithSpark: org.apache.spark.rdd.RDD[String] = FilteredRDD[4] at
filter at <console>:14
```

我们将transformations操作和actions操作连起来操作：

```
scala> textFile.filter(line => line.contains("Spark")).count()
res3: Long = 15
```

更多关于RDD上面的操作

RDD的transformations操作和actions操作可以用于更复杂的计算。下面的例子是找出README.md文件中单词数最多的行有多少个单词

```
scala> var size = textFile.map(line=>line.split(" ").size)
scala> size.reduce((a, b)=>if (a > b) a else b)
res4: Long = 15
```

map函数负责将line按照空格分割，并得到这行单词的数量，而reduce函数将获取文件中单词数最多的行有多少个单词。map和reduce函数的参数是Scala的函数式编程风格。我们可以直接用Java里面的Math.max()函数，这样会使得这段代码更好理解

```
scala> import java.lang.Math
import java.lang.Math

scala> textFile.map(line => line.split(" ").size).reduce((a, b)=>Math.max(a, b))
res10: Int = 15
```

我们比较熟悉的一种数据流模式是MapReduce，Spark可以很简单地实现MapReduce流

```
scala> val wordCounts = textFile.flatMap(line => line.split(" "))
    .map(word => (word, 1)).reduceByKey((a, b) => a + b)
wordCounts: org.apache.spark.rdd.RDD[(String, Int)] =
MapPartitionsRDD[16] at reduceByKey at <console>:15
```

在上面的代码中，我们结合了flatMap，map和reduceByKey等transformations操作来计算文件中每个单词的数量，并生成一个(String, Int) pairs形式的RDD。为了计算单词的数量，我们可以用collect action来实现：

```
scala> wordCounts.collect()
res11: Array[(String, Int)]=Array(("",120),(submitting,1),(find,1),(versions,4),
((`./bin/pyspark`).,1), (Regression,1), (via,2), (tests,2), (open,2),
(./bin/spark-shell,1), (When,1), (All,1), (download,1), (requires,2),
(SPARK_YARN=true,3), (Testing,1), (take,1), (project,4), (no,1),
(systems.,1), (file,1), (<params>`.,1), (Or,,1), (`<dependencies>`,1),
(About,1), (project's,3), (`<master>`,1), (programs,2),(given.,1),(obtained,1),
(sbt/sbt,5), (artifact,1), (SBT,1), (local[2],1), (not,1), (runs.,1), (you,5),
(building,1), (Along,1), (Lightning-Fast,1), (built,,1), (Hadoop,,1), (use,2),
(MRv2,,1), (it,2), (directory.,1), (overview,1), (2.10.,1),(The,1),(easiest,1),
(Note,1), (guide](http://spark.apache.org/docs/latest/configuration.html),1),
(setup,1), ("org.apache.hadoop",1),...
```

Caching

Spark可以将数据集存放在集群中的缓存中。这个在数据集经常被访问的场景下很有用，比如hot数据集的查询，或者像PageRank这样的需要迭代很多次的算法。作为一个简单的列子，下面是将我们自己的linesWithSpark dataset存入到缓存中：

```
scala> linesWithSpark.cache()
res12: org.apache.spark.rdd.RDD[String] =FilteredRDD[4] at filter at <console>:14

scala> linesWithSpark.count()
res13: Long = 15

scala> linesWithSpark.count()
res14: Long = 15
```

利用Spark来缓存100行的数据看起来有点傻，但是我们可以通过同样的函数来存储非常大的数据集，甚至这些数据集分布在几十或者几百台节点上。

本文翻译自Spark中的文档，本文地址：《Spark快速入门指南(Quick Start Spark)》：https://www.iteblog.com/archives/1040.html，过往记忆，大量关于Hadoop、Spark等个人原

创技术博客