

Hadoop&Spark解决二次排序问题(Spark篇)

我在[《Hadoop&Spark解决二次排序问题\(Hadoop篇\)》](#)

文章中介绍了如何在Hadoop中实现二次排序问题，今天我将介绍如何在Spark中实现。

问题描述

二次排序就是key之间有序，而且每个Key对应的value也是有序的；也就是对MapReduce的输出(

KEY, Value($v_1, v_2, v_3, \dots, v_n$))中的Value($v_1, v_2, v_3, \dots, v_n$

)值进行排序（升序或者降序），使得Value($s_1, s_2, s_3, \dots, s_n$), $s_i \in (v_1, v_2, v_3, \dots, v_n)$ 且 $s_1 < s_2 < s_3 < \dots < s_n$ 。假设我们有以下输入文件（逗号分割的分别是年，月，总数）：

```
[root@iteblog.com /tmp]# vim data.txt
```

```
2015,1,24
```

```
2015,3,56
```

```
2015,1,3
```

```
2015,2,-43
```

```
2015,4,5
```

```
2015,3,46
```

```
2014,2,64
```

```
2015,1,4
```

```
2015,1,21
```

```
2015,2,35
```

```
2015,2,0
```

我们期望的输出结果是

```
2014-2 64
```

```
2015-1 3,4,21,24
```

```
2015-2 -43,0,35
```

```
2015-3 46,56
```

```
2015-4 5
```

解决方案

在Spark中解决这种问题相对Hadoop来说是非常简单的，我们只需要将年和月组合起来构成

一个Key，将第三列作为value，并使用 groupByKey
函数将同一个Key的所有Value全部弄到一起，然后对同一个Key的所有Value进行排序即可。

代码实例

为了简便，我直接在Spark-shell中解决这个问题，首先我们启动一个Spark-shell：

```
bin/spark-shell --master yarn-client --executor-memory 1g --num-  
executors 2 --queue iteblog --executor-cores 1  
Welcome to
```

```
  _/ / _ _ _ _ / / _  
 _W W/ _ W/ _ ` / _ / ' /  
 / _ / . _ /W _ / / / /W _W version 1.6.1  
 / /
```

```
Using Scala version 2.10.5 (Java HotSpot(TM) 64-Bit Server VM, Java 1.7.0_45)  
Type in expressions to have them evaluated.  
Type :help for more information.  
Spark context available as sc.  
SQL context available as sqlContext.
```

```
scala>
```

在里面输入以下代码即可：

```
scala> val file = sc.textFile("/tmp/data.txt")  
file: org.apache.spark.rdd.RDD[String] = /tmp/data.txt MapPartitionsRDD[1] at textFile at <console>:27
```

```
scala> val data = file.map(_.split(",")).map(item => (s"${item(0)}-${item(1)}", item(2)))  
data: org.apache.spark.rdd.RDD[(String, String)] = MapPartitionsRDD[3] at map at <console>:29
```

```
scala> data.collect().foreach(println)  
(2015-1,24)  
(2015-3,56)  
(2015-1,3)  
(2015-2,-43)  
(2015-4,5)  
(2015-3,46)  
(2014-2,64)
```

```
(2015-1,4)
(2015-1,21)
(2015-2,35)
(2015-2,0)
```

```
scala> val rdd = data.groupByKey
rdd: org.apache.spark.rdd.RDD[(String, Iterable[String])] = ShuffledRDD[5] at groupByKey at <console>:31
```

```
scala> rdd.collect().foreach(println)
(2014-2,CompactBuffer(64))
(2015-1,CompactBuffer(24, 3, 4, 21))
(2015-2,CompactBuffer(35, 0, -43))
(2015-3,CompactBuffer(56, 46))
(2015-4,CompactBuffer(5))
```

```
scala> val result = rdd.map(item => (item._1, item._2.toList.sortWith(_.toInt<_.toInt)))
result: org.apache.spark.rdd.RDD[(String, List[String])] = MapPartitionsRDD[20] at map at <console>:33
```

```
scala> result.collect.foreach(item => println(s"${item._1}Wt${item._2.mkString(",")})")
2014-2 64
2015-1 3,4,21,24
2015-2 -43,0,35
2015-3 46,56
2015-4 5
```

可以看出，使用Spark来解决这个问题非常地简单。上面的CompactBuffer实现了Scala中的Seq类，所以可以直接转换成List或者Array，然后直接使用Scala中的排序即可。

本博客文章除特别声明，全部都是原创！
原创文章版权归过往记忆大数据（[过往记忆](#)）所有，未经许可不得转载。
本文链接: **【】**（**）**