Spark自定义分区(Partitioner)

我们都知道Spark内部提供了HashPartitioner和RangePartitioner两种分区策略(这两种分区的代码解析可以参见：《Spark分区器HashPartitioner和RangePartitioner代码详解》)，这两种分区策略在很多情况下都适合我们的场景。但是有些情况下，Spark内部不能符合咱们的需求，这时候我们就可以自定义分区策略。为此，Spark提供了相应的接口，我们只需要扩展Partitioner抽象类，然后实现里面的三个方法：

```
package org.apache.spark

/**
 * An object that defines how the elements in a key-value pair RDD are partitioned by key.
 * Maps each key to a partition ID, from 0 to `numPartitions - 1`.
 */
abstract class Partitioner extends Serializable {
  def numPartitions: Int
  def getPartition(key: Any): Int
}
```

def numPartitions: Int：这个方法需要返回你想要创建分区的个数；
def getPartition(key: Any): Int：这个函数需要对输入的key做计算，然后返回该key的分区ID，范围一定是0到numPartitions-1；
equals()：这个是Java标准的判断相等的函数，之所以要求用户实现这个函数是因为Spark内部会比较两个RDD的分区是否一样。

假如我们想把来自同一个域名的URL放到一台节点上，比如:和/archives/1368，如果你使用HashPartitioner，这两个URL的Hash值可能不一样，这就使得这两个URL被放到不同的节点上。所以这种情况下我们就需要自定义我们的分区策略，可以如下实现：

```
package com.iteblog.utils

import org.apache.spark.Partitioner

/**
 * User: 过往记忆
 * Date: 2015-05-21
 * Time: 下午23:34
 * bolg:
 * 本文地址：/archives/1368
```

```
 * 过往记忆博客，专注于hadoop、hive、spark、shark、flume的技术博客，大量的干货
 * 过往记忆博客微信公共帐号：iteblog_hadoop
 */

class IteblogPartitioner(numParts: Int) extends Partitioner {
  override def numPartitions: Int = numParts

  override def getPartition(key: Any): Int = {
    val domain = new java.net.URL(key.toString).getHost()
    val code = (domain.hashCode % numPartitions)
    if (code < 0) {
      code + numPartitions
    } else {
      code
    }
  }

  override def equals(other: Any): Boolean = other match {
    case iteblog: IteblogPartitioner =>
      iteblog.numPartitions == numPartitions
    case _ =>
      false
  }

  override def hashCode: Int = numPartitions
}
```

因为hashCode值可能为负数，所以我们需要对他进行处理。然后我们就可以在partitionBy()方法里面使用我们的分区：

```
iteblog.partitionBy(new IteblogPartitioner(20))
```

类似的，在Java中定义自己的分区策略和Scala类似，只需要继承org.apache.spark.Partitioner，并实现其中的方法即可。

在Python中，你不需要扩展Partitioner类，我们只需要对iteblog.partitionBy()加上一个额外的hash函数，如下：

```
import urlparse
```

```
def iteblog_domain(url):
  return hash(urlparse.urlparse(url).netloc)

iteblog.partitionBy(20, iteblog_domain)
```