Spark中parallelize函数和makeRDD函数的区别

我们知道，在Spark中创建RDD的创建方式大概可以分为三种：（1）、从集合中创建RDD；（2）、从外部存储创建RDD；（3）、从其他RDD创建。

而从集合中创建RDD，Spark主要提供了两中函数：parallelize和makeRDD。我们可以先看看这两个函数的声明：

```
def parallelize[T: ClassTag](
    seq: Seq[T],
    numSlices: Int = defaultParallelism): RDD[T]

def makeRDD[T: ClassTag](
    seq: Seq[T],
    numSlices: Int = defaultParallelism): RDD[T]

def makeRDD[T: ClassTag](seq: Seq[(T, Seq[String])]): RDD[T]
```

我们可以从上面看出makeRDD有两种实现，而且第一个makeRDD函数接收的参数和parallelize完全一致。其实第一种makeRDD函数实现是依赖了parallelize函数的实现，来看看Spark中是怎么实现这个makeRDD函数的：

```
 def makeRDD[T: ClassTag](
    seq: Seq[T],
    numSlices: Int = defaultParallelism): RDD[T] = withScope {
  parallelize(seq, numSlices)
 }
```

我们可以看出，这个makeRDD函数完全和parallelize函数一致。但是我们得看看第二种makeRDD函数函数实现了，它接收的参数类型是Seq[(T, Seq[String])]，Spark文档的说明是

> Distribute a local Scala collection to form an RDD, with one or more location preferences (hostnames of Spark nodes) for each object. Create a new partition for each collection item.

原来，这个函数还为数据提供了位置信息，来看看我们怎么使用：

```
scala> val iteblog1 = sc.parallelize(List(1,2,3))
iteblog1: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[10] at parallelize at <console>
:21

scala> val iteblog2 = sc.makeRDD(List(1,2,3))
iteblog2: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[11] at makeRDD at <console>:
21

scala> val seq = List((1, List("iteblog.com", "sparkhost1.com", "sparkhost2.com")),
     | (2, List("iteblog.com", "sparkhost2.com")))
seq: List[(Int, List[String])] = List((1,List(iteblog.com, sparkhost1.com, sparkhost2.com)),
 (2,List(iteblog.com, sparkhost2.com)))

scala> val iteblog3 = sc.makeRDD(seq)
iteblog3: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[12] at makeRDD at <console>:
23

scala> iteblog3.preferredLocations(iteblog3.partitions(1))
res26: Seq[String] = List(iteblog.com, sparkhost2.com)

scala> iteblog3.preferredLocations(iteblog3.partitions(0))
res27: Seq[String] = List(iteblog.com, sparkhost1.com, sparkhost2.com)

scala> iteblog1.preferredLocations(iteblog1.partitions(0))
res28: Seq[String] = List()
```

我们可以看到，makeRDD函数有两种实现，第一种实现其实完全和parallelize一致；而第二种实现可以为数据提供位置信息，而除此之外的实现和parallelize函数也是一致的，如下：

```
 def parallelize[T: ClassTag](
    seq: Seq[T],
    numSlices: Int = defaultParallelism): RDD[T] = withScope {
  assertNotStopped()
  new ParallelCollectionRDD[T](this, seq, numSlices, Map[Int, Seq[String]]())
 }

 def makeRDD[T: ClassTag](seq: Seq[(T, Seq[String])]): RDD[T] = withScope {
  assertNotStopped()
  val indexToPrefs = seq.zipWithIndex.map(t => (t._2, t._1._2)).toMap
  new ParallelCollectionRDD[T](this, seq.map(_._1), seq.size, indexToPrefs)
 }
```

都是返回ParallelCollectionRDD，而且这个makeRDD的实现不可以自己指定分区的数量，而是固定为seq参数的size大小。