GraphX 实习

1700017831 薛飞跃

一．实习目标

用 GraphX 再次实现 PageRank。

二．算法思路

1. 将邻接表形式的图转化为 RDD

①假设四个节点，A 指向 B、C，B 指向 A、C，C 指向 A、B、D，D 指向 C。

②使用 HashPartitioner 进行分区，分区数大于 4，确保四个节点在四个分区。因为 links 是静态数据集，后续有很多连接操作，提前分区可以节约后续很多网络通信开销。

③persist()让 links 保留在节点中，以供每次迭代使用

```
scala> import org.apache.spark.HashPartitioner
import org.apache.spark.HashPartitioner

scala> val links = sc.parallelize(List(("A",List("B","C")),("B",List("A","C")),("C",List("A","B","D")),("D",List("C")))
     | ).partitionBy(new HashPartitioner(100)).persist()
links: org.apache.spark.rdd.RDD[(String, List[String])] = ShuffledRDD[1] at partitionBy at <console>:26
```

2.将 ranks 赋值为全为 1 的 RDD。这里用 mapValues 是因为 ranks 和 links 的 key 都是四个节点，只是 value 不同，mapValues 不改变分区方式。

```
scala> var ranks=links.mapValues(v=>1.0)
ranks: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[2] at mapValues at <console>:26
```

3.迭代更新 ranks 的值

```
scala> for (i <- 0 until 10) {
     | val contributions=links.join(ranks).flatMap {
     | case (pageId,(links,rank)) => links.map(dest=>(dest,rank/links.size))
     | }
     | ranks=contributions.reduceByKey((x,y)=>x+y).mapValues(v=>0.15+0.85*v)
     | }
```

①links.join(ranks),因为 links 和 ranks 的 key 值相同，可以进行连接，比如一个 link=(A,[B,C]),和一个 rank=(A,1)连接为(A,([B,C],1))

②假如(pageId,(links,rank))为(A,([B,C],1))，links 则为[B,C],map 函数将 links 中的每一个元素 dest 映射为(dest,rank/links.size)的二元组，即图中第三行，将(A,([B,C],1))映射为((B,0.5),(C,0.5)),也即 A 把自己的 rank 均分给它指向的节点。

③flatMap{case A => B}是偏函数的表示，代表 flatMap 只把满足 A 的形式的元素映射为 B，其中 flatMap 可以将一个元素映射为多个结果。②中把(A,([B,C],1))映射为((B,0.5),(C,0.5))，而 flatMap 将后者展开为

(B,0.5),(C,0.5)两个结果。

④contributions 的计算思路为，比如 A 指向 B、C，A 的 rank 为 1，则产生(B,0.5),(C,0.5)，代表这一轮 A 给 B、C 赋予的 rank 值

⑤contributions 进行 reduceByKey，将 key 值相同的结果相加，生成的(pageId,rank)代表所有指向 pageId 的节点赋予 pageId 的 rank 值之和。

⑥再用 mapValues 将 rank 值乘以 0.85，加上 0.15，来产生新一轮的 rank 值。

4.迭代完成后查看最终的权值。

```
scala> ranks.sortByKey().foreach(println)
[Stage 48:=================================================>   (93 + 1) / 100](A,0.9850243302878132)
(B,0.9850243302878132)
(C,1.4621033282930214)
(D,0.5678480111313515)
```

三．心得体会

1. 静态数据如果后续有很多连接操作，可以提前进行分区，减少网络通信开销。经常用来计算的数据可以直接保存在内存中，减少磁盘读取开销。

2. Map，mapValues，flatMap，有各自的功能实际中注意区分和灵活应用。