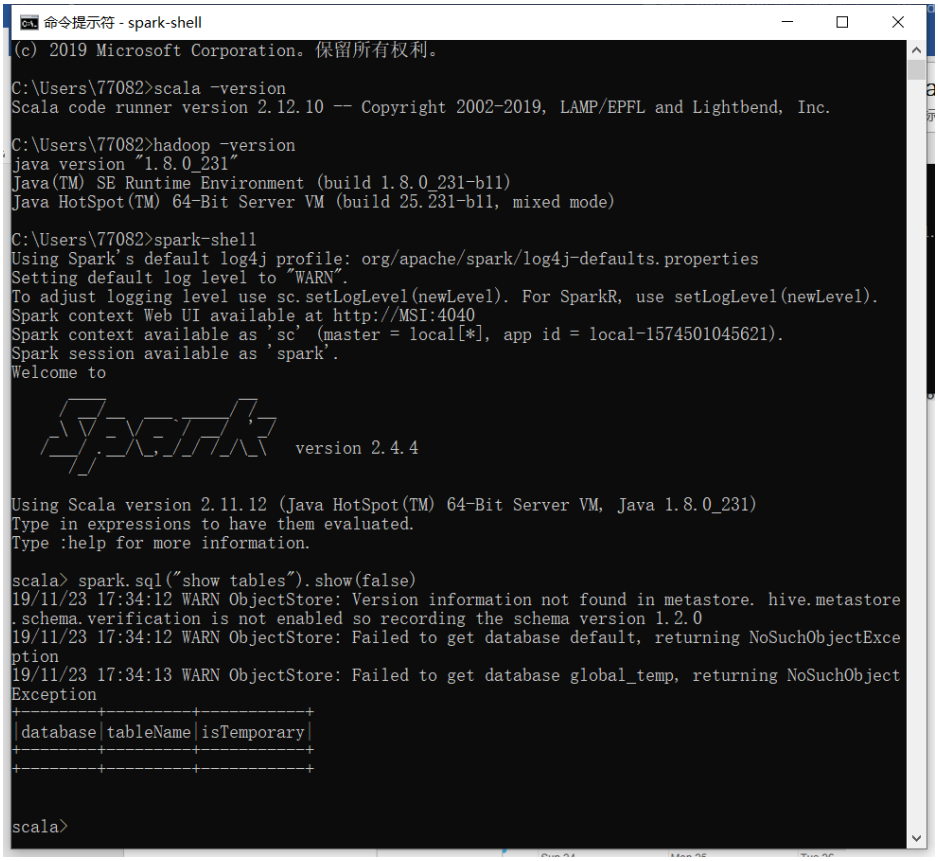
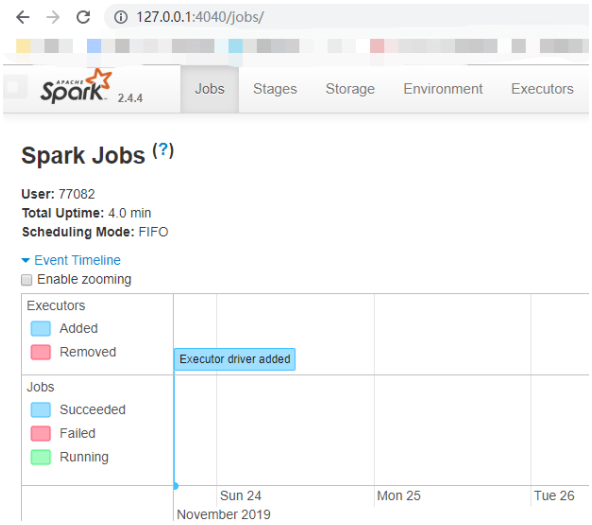
1. 首先根据网上教程搭建 win10 + scala + spark + hadoop 环境



登录网站 http://127.0.0.1:4040/jobs/



环境搭建成功。

2. 配置 IDEA 环境
3. PageRank 是执行多次连接的一个迭代算法。 算法会维护两个数据集（在 spark 中为 RDD）：

   links：由（pageID, linkList）的元素组成，包含每个页面的相邻页面的列表。其中的一个元素例如:(A,[B,C,D]) 代表 A 中含指向 B C D 的链接

   ranks：由（pageID,PR）元素组成，包含每个页面的当前排序值。它按如下步 骤进行计算。其中的一个元素例如（A,0.7）代表

(1) 将每个页面的排序值初始化为 1.0。

(2) 在每次迭代中，对页面 p，向其每个相邻页面（有直接链接的页面）发送一个值为 PR(p)/L(p) 的贡献值。

(3) 将每个页面的排序值设为 0.15 + 0.85 * contributionsReceived。

最后两步会重复几个循环，在此过程中，算法会逐渐收敛于每个页面的实际 PageRank 值。通常需要大约 10 轮迭代。

4. 实现源码如下：

算法思路如上所示。

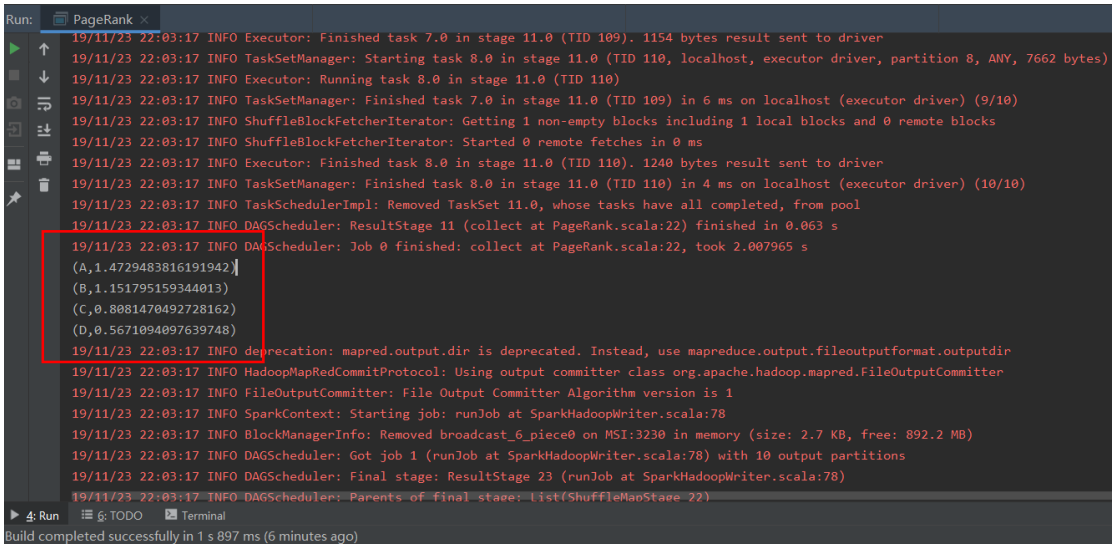
```
    import org.apache.spark.{HashPartitioner, SparkConf,SparkContext}
object PageRank {
  def main(args:Array[String]):Unit = {
    val conf = new SparkConf().setAppName("PageRank").setMaster("local")
    val sc = new SparkContext(conf)
    var links = sc.parallelize(List(
      ("A",List("B","C","D")),
      ("B",List("A")),
      ("C",List("A","B")),
      ("D",List("B","C"))
    ))
    // 初始化 ranks 中每个页面的 PR 值为 1.0
    var ranks = links.mapValues(v => 1.0)
        for(i <- 0 until 10){
          val contributions = links.join(ranks).flatMap{
            case(pageID,(links,rank)) => links.map(link => (link , rank /
links.size))
          }
          ranks = contributions.reduceByKey((x,y) => x+y).mapValues(v =>
0.15 * 1.0 + 0.85 *v)
        }
        ranks.collect().foreach(println)
        ranks.saveAsTextFile("result")
  }
}
```

5. 运行结果：保存在 result 文件夹下的 part-000000 文件

```
(B,1.1517951593440 13)
(A,1.4729483816191942)
(C,0.8081470492728162)
(D,0.5671094097639748)

```

IDEA 环境下的输出为:


Run:
PageRank
19/11/23 22:03:17 INFO Executor: Finished task 7.0 in stage 11.0 (TID 109). 1154 bytes result sent to driver
19/11/23 22:03:17 INFO TaskSetManager: Starting task 8.0 in stage 11.0 (TID 110, localhost, executor driver, partition 8, ANY, 7662 bytes)
19/11/23 22:03:17 INFO Executor: Running task 8.0 in stage 11.0 (TID 110)
19/11/23 22:03:17 INFO TaskSetManager: Finished task 7.0 in stage 11.0 (TID 109) in 6 ms on localhost (executor driver) (9/10)
19/11/23 22:03:17 INFO ShuffleBlockFetcherIterator: Getting 1 non-empty blocks including 1 local blocks and 0 remote blocks
19/11/23 22:03:17 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
19/11/23 22:03:17 INFO Executor: Finished task 8.0 in stage 11.0 (TID 110). 1240 bytes result sent to driver
19/11/23 22:03:17 INFO TaskSetManager: Finished task 8.0 in stage 11.0 (TID 110) in 4 ms on localhost (executor driver) (10/10)
19/11/23 22:03:17 INFO TaskSchedulerImpl: Removed TaskSet 11.0, whose tasks have all completed, from pool
19/11/23 22:03:17 INFO DAGScheduler: ResultStage 11 (collect at PageRank.scala:22) finished in 0.063 s
19/11/23 22:03:17 INFO DAGScheduler: Job 0 finished: collect at PageRank.scala:22, took 2.007965 s
(A,1.4729483816191942)
(B,1.151795159344013)
(C,0.808147049728162)
(D,0.5671094097639748)
19/11/23 22:03:17 INFO deprecation: mapred.output.dir is deprecated. Instead, use mapreduce.output.fileoutputformat.outputdir
19/11/23 22:03:17 INFO HadoopMapRedCommitProtocol: Using output committer class org.apache.hadoop.mapred.FileOutputCommitter
19/11/23 22:03:17 INFO FileOutputCommitter: File Output Committer Algorithm version is 1
19/11/23 22:03:17 INFO SparkContext: Starting job: runJob at SparkHadoopWriter.scala:78
19/11/23 22:03:17 INFO BlockManagerInfo: Removed broadcast_6_piece0 on MSI:3230 in memory (size: 2.7 KB, free: 892.2 MB)
19/11/23 22:03:17 INFO DAGScheduler: Got job 1 (runJob at SparkHadoopWriter.scala:78) with 10 output partitions
19/11/23 22:03:17 INFO DAGScheduler: Final stage: ResultStage 23 (runJob at SparkHadoopWriter.scala:78)
19/11/23 22:03:17 INFO DAGScheduler: Parents of final stage: List(ShuffleMapStage 22)
4: Run
6: TODO
Terminal
Build completed successfully in 1 s 897 ms (6 minutes ago)