

第一次作业：使用scala实现wordcount

201250104 苏致成

源程序

```
package nju.edu

import org.apache.spark.rdd.RDD
import org.apache.spark.{SparkConf, SparkContext}

import scala.collection.MapView
import scala.io.Source

object wordCount {
  /*
   * 201250104 苏致成
   */
  def main(args: Array[String]) = {

    // setup link
    val conf = new SparkConf().setMaster("local").setAppName("wordCount")
    val sc = new SparkContext(conf)

    // read lines
    var lines = sc.textFile("test.txt")

    //read words
    val words = lines.flatMap(line => line.split(" "))

    //groups
    val groups = words.groupBy(word => word.toLowerCase())

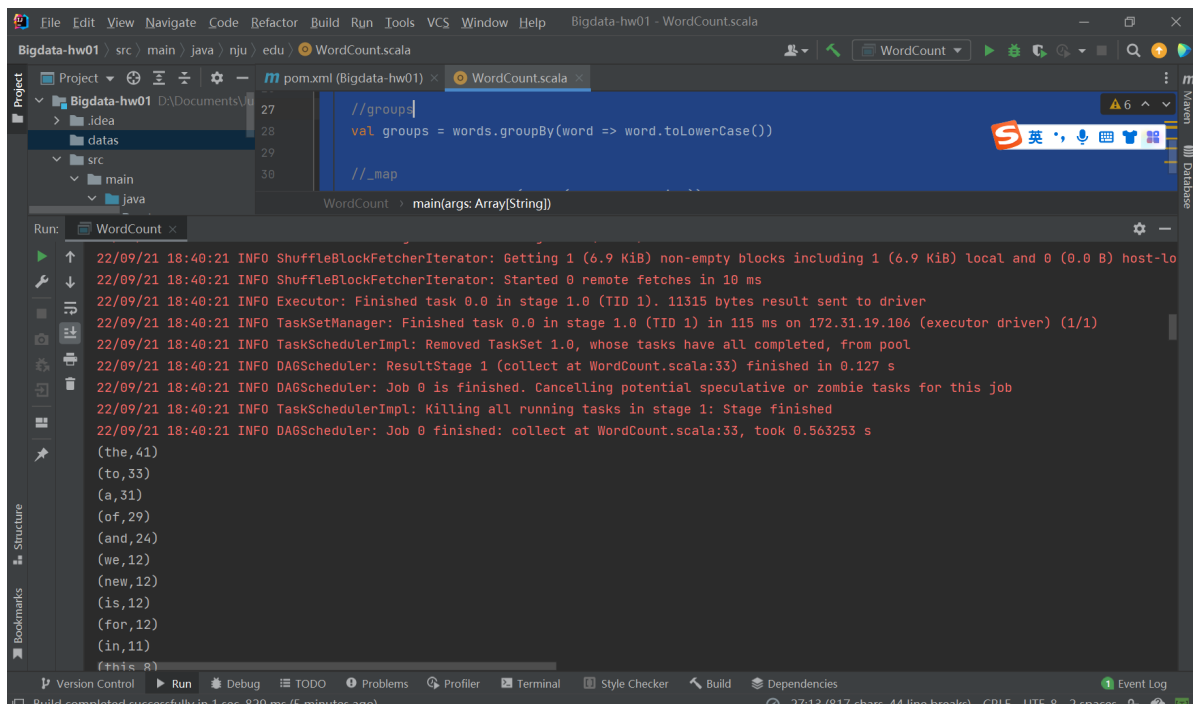
    //_map
    var _map = groups.map(x => (x._1, x._2.size))

    //ans
    val ret = _map.collect().sortBy(sortRule).reverse
    ret.foreach(println)

    sc.stop()
  }

  def sortRule(tmp: (String, Int)): (Int, String) = {
    (tmp._2, tmp._1)
  }
}
```

运行截图



```
//groupBy
val groups = words.groupBy(word => word.toLowerCase())

//_map

WordCount > main(args: Array[String])

Run: WordCount <
22/09/21 18:40:21 INFO ShuffleBlockFetcherIterator: Getting 1 (6.9 KiB) non-empty blocks including 1 (6.9 KiB) local and 0 (0.0 B) host-lo
22/09/21 18:40:21 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 18 ms
22/09/21 18:40:21 INFO Executor: Finished task 0.0 in stage 1.0 (TID 1). 11315 bytes result sent to driver
22/09/21 18:40:21 INFO TaskSetManager: Finished task 0.0 in stage 1.0 (TID 1) in 115 ms on 172.31.19.106 (executor driver) (1/1)
22/09/21 18:40:21 INFO TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all completed, from pool
22/09/21 18:40:21 INFO DAGScheduler: ResultStage 1 (collect at WordCount.scala:33) finished in 0.127 s
22/09/21 18:40:21 INFO DAGScheduler: Job 0 is finished. Cancelling potential speculative or zombie tasks for this job
22/09/21 18:40:21 INFO TaskSchedulerImpl: Killing all running tasks in stage 1: Stage finished
22/09/21 18:40:21 INFO DAGScheduler: Job 0 finished: collect at WordCount.scala:33, took 0.563253 s
(the,41)
(to,33)
(a,31)
(of,29)
(and,24)
(we,12)
(new,12)
(is,12)
(for,12)
(in,11)
(this,8)
```

实现思路

1. 每一个Spark应用都是一个SparkContext实例，因此需要创建SparkContext实例，并且需要进行相关配置。
2. 读取文件，并将其用flatMap转为类似于java中List的结构。
3. 用groupBy函数将其按照词进行分类，同一个词属于同一类。
4. 用map将其建立词语到词频的映射列表。
5. 自定义sortBy算子作为参数，调用sortBy对其进行降序排序
6. 输出

实验心得

1. 环境配置需要综合考虑scala和spark的版本，防止出错。
2. scala语法类似java，但是其灵活性上类似于python，不愧是可伸缩的语言。
3. scala中某些时候难以查看中间值，只能查看其存储的id。
4.
 - Spark 是分布式计算平台，是一个用scala语言编写的计算框架，基于内存的快速、通用、可扩展的大数据分析；
 - Hadoop 是分布式管理、存储、计算的生态系统
 - Spark本身没有提供文件管理系统，所以它必须和其他的分布式文件系统进行集成如基于HDFS的HBase
5. Spark用户提交的任务称为 application，一个application对应一个SparkContext，app中存在多个job，每触发一次action操作就会产生一个job
6. RDD使用起来十分灵活，如groupByKey()、flatMap()、map()、filter()等等操作

