Spark算子第一部分

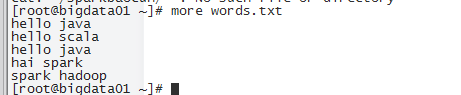
主要理解算子、rdd

1. 在shell上演示操作集群

操作步骤：

1. 将以下数据放入Hadoop集群中，目录名称为/spark

hdfs dfs -mkdir /spark , hdfs dfs -put word.txt /spark/1.log , hdfs dfs -put word.txt /spark/2.log



1. 启动spark集群（在master的bin目录下）

./spark-shell --master spark://bigdata01:7077 --total-executor-cores 2 --executor-memory 512m

1. 在spark集群上统计单词的个数

val rdd = sc.textFile("hdfs://bigdata01:9000/spark")

val rdd1 = rdd.flatMap(\_.split(" ")).map((\_, 1 )).reduceByKey(\_ + \_).sortBy(\_.\_2,false)

rdd1.collect

rdd1.saveAsTextFile("hdfs://bigdata01:9000/spark01")

1. 查看spark01目录的文件

hdfs dfs -cat /spark01/\*

2.在IDEA上演示操作集群

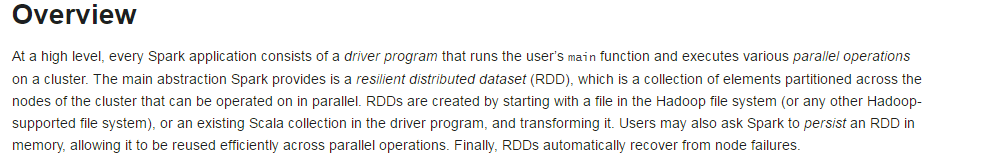
1. Maven 配置

<**dependencies**>  
 *<!-- https://mvnrepository.com/artifact/org.scala-lang/scala-library -->* <**dependency**>  
 <**groupId**>org.scala-lang</**groupId**>  
 <**artifactId**>scala-library</**artifactId**>  
 <**version**>2.10.6</**version**>  
 </**dependency**>  
  
 <**dependency**>  
 <**groupId**>org.apache.spark</**groupId**>  
 <**artifactId**>spark-core\_2.10</**artifactId**>  
 <**version**>1.6.0</**version**>  
 </**dependency**>  
  
 <**dependency**>  
 <**groupId**>org.apache.hadoop</**groupId**>  
 <**artifactId**>hadoop-client</**artifactId**>  
 <**version**>2.6.5</**version**>  
 </**dependency**>  
</**dependencies**>  
  
<**build**>  
 <**sourceDirectory**>src/main/scala</**sourceDirectory**>  
 <**testSourceDirectory**>src/test/scala</**testSourceDirectory**>  
 <**plugins**>  
 <**plugin**>  
 <**groupId**>net.alchim31.maven</**groupId**>  
 <**artifactId**>scala-maven-plugin</**artifactId**>  
 <**version**>3.2.2</**version**>  
 <**executions**>  
 <**execution**>  
 <**goals**>  
 <**goal**>compile</**goal**>  
 <**goal**>testCompile</**goal**>  
 </**goals**>  
 <**configuration**>  
 <**args**>  
 <**arg**>-make:transitive</**arg**>  
 <**arg**>-dependencyfile</**arg**>  
 <**arg**>${project.build.directory}/.scala\_dependencies</**arg**>  
 </**args**>  
 </**configuration**>  
 </**execution**>  
 </**executions**>  
 </**plugin**>  
  
 <**plugin**>  
 <**groupId**>org.apache.maven.plugins</**groupId**>  
 <**artifactId**>maven-shade-plugin</**artifactId**>  
 <**version**>2.4.3</**version**>  
 <**executions**>  
 <**execution**>  
 <**phase**>package</**phase**>  
 <**goals**>  
 <**goal**>shade</**goal**>  
 </**goals**>  
 <**configuration**>  
 <**filters**>  
 <**filter**>  
 <**artifact**>\*:\*</**artifact**>  
 <**excludes**>  
 <**exclude**>META-INF/\*.SF</**exclude**>  
 <**exclude**>META-INF/\*.DSA</**exclude**>  
 <**exclude**>META-INF/\*.RSA</**exclude**>  
 </**excludes**>  
 </**filter**>  
 </**filters**>  
 *<!--<transformers>-->  
 <!--<transformer implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">-->  
 <!--<mainClass></mainClass>-->  
 <!--</transformer>-->  
 <!--</transformers>-->* </**configuration**>  
 </**execution**>  
 </**executions**>  
 </**plugin**>  
 </**plugins**>  
</**build**>

1. 代码实现

package com.youe.spark  
  
import org.apache.spark.{SparkConf, SparkContext}  
  
/\*\*  
 \* Created by hadoop on 2017/2/9 0009.  
 \*/  
object HelloSpark {  
  
 def main(args: Array[String]): Unit = {  
  
 val sconf = new SparkConf().setAppName("hellospark")  
  
 val sc = new SparkContext(sconf)  
  
 sc.textFile(args(0)).flatMap(\_.split(" ")).map((\_,1)).reduceByKey(\_ + \_).sortBy(\_.\_2, false).saveAsTextFile(args(1))  
 sc.stop()  
 }  
}

3.阅读spark1.6.0官方文档



Rdd创建的方式：1.访问外部的文件系统，2.原来rrd上调用了transformations方法

4.shell上面演示transformations/action算子

#常用Transformation(即转换，延迟加载)

#通过并行化scala集合创建RDD

val rdd1 = sc.parallelize(Array(1,2,3,4,5,6,7,8))

#查看该rdd的分区数量

rdd1.partitions.length

val rdd1 = sc.parallelize(List(5,6,4,7,3,8,2,9,1,10))

val rdd2 = sc.parallelize(List(5,6,4,7,3,8,2,9,1,10)).map(\_\*2).sortBy(x=>x,true)

val rdd3 = rdd2.filter(\_>10)

val rdd2 = sc.parallelize(List(5,6,4,7,3,8,2,9,1,10)).map(\_\*2).sortBy(x=>x+"",true)

val rdd2 = sc.parallelize(List(5,6,4,7,3,8,2,9,1,10)).map(\_\*2).sortBy(x=>x.toString,true)

val rdd4 = sc.parallelize(Array("a b c", "d e f", "h i j"))

rdd4.flatMap(\_.split(' ')).collect

val rdd5 = sc.parallelize(List(List("a b c", "a b b"),List("e f g", "a f g"), List("h i j", "a a b")))

List("a b c", "a b b") =List("a","b",))

rdd5.flatMap(\_.flatMap(\_.split(" "))).collect

#union求并集，注意类型要一致

val rdd6 = sc.parallelize(List(5,6,4,7))

val rdd7 = sc.parallelize(List(1,2,3,4))

val rdd8 = rdd6.union(rdd7)

rdd8.distinct.sortBy(x=>x).collect

#intersection求交集

val rdd9 = rdd6.intersection(rdd7)

val rdd1 = sc.parallelize(List(("tom", 1), ("jerry", 2), ("kitty", 3)))

val rdd2 = sc.parallelize(List(("jerry", 9), ("tom", 8), ("shuke", 7)))

#join

val rdd3 = rdd1.join(rdd2)

val rdd3 = rdd1.leftOuterJoin(rdd2)

val rdd3 = rdd1.rightOuterJoin(rdd2)

#groupByKey

val rdd3 = rdd1 union rdd2

rdd3.groupByKey

rdd3.groupByKey.map(x=>(x.\_1,x.\_2.sum))

#WordCount, 第二个效率低

sc.textFile("hdfs://bigdata01:9000/words.txt").flatMap(x=>x.split(" ")).map((\_,1)).reduceByKey(\_+\_).sortBy(\_.\_2,false).collect

sc.textFile("hdfs://bigdata01:9000/words.txt").flatMap(x=>x.split(" ")).map((\_,1)).groupByKey.map(t=>(t.\_1, t.\_2.sum)).collect

#cogroup

val rdd1 = sc.parallelize(List(("tom", 1), ("tom", 2), ("jerry", 3), ("kitty", 2)))

val rdd2 = sc.parallelize(List(("jerry", 2), ("tom", 1), ("shuke", 2)))

val rdd3 = rdd1.cogroup(rdd2)

val rdd4 = rdd3.map(t=>(t.\_1, t.\_2.\_1.sum + t.\_2.\_2.sum))

#cartesian笛卡尔积

val rdd1 = sc.parallelize(List("tom", "jerry"))

val rdd2 = sc.parallelize(List("tom", "kitty", "shuke"))

val rdd3 = rdd1.cartesian(rdd2)

###################################################################################################

#spark action

val rdd1 = sc.parallelize(List(1,2,3,4,5), 2)

#collect

rdd1.collect

#reduce

val rdd2 = rdd1.reduce(\_+\_)

#count

rdd1.count

#top

rdd1.top(2)

#take

rdd1.take(2)

#first(similer to take(1))

rdd1.first

#takeOrdered

rdd1.takeOrdered(3)

spark 算子第一部分天作业

1. 按照上课的场景，从头开始敲一遍，并理解、记忆运行算子后的效果
2. 在idea上背着完成wordcount，提交到集群运行