课程内容

使用idea开发spark 实战

使用idea 开发spark的Local和Cluster

一： 配置开发环境

1. 要在本地安装好java和scala。

由于spark1.6需要scala 2.10.X版本的。推荐 2.10.4，java版本最好是1.8。所以提前我们要需要安装好java和scala 并在环境变量中配置好

1. 下载 IDEA 社区版本，选择windows 版本并按照配置。

安装完成以后启动IDEA，并进行配置，默认即可，然后点击ok以后，设置ui风格， 然后点击next 会出现插件的选择页面，默认不需求修改，点击next，选择安装scala 语言，点击 install 按钮（非常重要，以为要开发spark程序所以必须安装），等安装完成以后点击start启动 IDEA。

1. 创建scala项目

点击 create new project ，然后填写project name 为“Wordcount” ，选择项目的保存地址 project location。

然后设置project sdk 即java 的安装目录。点击右侧的new 按钮，选择jdk，然后选择java 的安装路径即可。

然后选择scala sdk 。点击右侧的create ，默认出现时2.10.x 版本的scala ，点击ok即可。然后点击finish。

1. 设置 spark的jar 依赖。

点击file->project structure 来设置工程的libraries。核心是添加spark的jar依赖。 选择 Libraries ，点击右侧的加号，选择java，选择spark1.6.0 的 spark-1.6.0-bin-hadoop2.6\lib\spark-assembly-1.6.0-hadoop2.6.0.jar 。 点击ok。稍等片刻后然后点击ok (Libraries作用于WordCount)，然后点击apply，点击ok。（这一步很重要，如果没有无法编写spark的代码）

二： 编写代码

1. 在src 下建立spark程序工程包

在src上右击 new ->package 填入package 的name为com.dt.spark 。

1. 创建scala的入口类。

在包的名字上右击 选择 new ->scala class 。在弹出框中 填写Name ，并制定kind 为object ，点击ok。

1. 编写local代码

package com.dt.spark

import org.apache.spark.SparkConf

import org.apache.spark.SparkContext

import org.apache.spark.rdd.RDD

/\*\*

\* 使用Scala开发本地测试的Spark WordCount程序

\* @author DT大数据梦工厂

\* 新浪微博：http://weibo.com/ilovepains/

\*/

object WordCount {

def main(args: Array[String]){

/\*\*

\* 第1步：创建Spark的配置对象SparkConf，设置Spark程序的运行时的配置信息，

\* 例如说通过setMaster来设置程序要链接的Spark集群的Master的URL,如果设置

\* 为local，则代表Spark程序在本地运行，特别适合于机器配置条件非常差（例如

\* 只有1G的内存）的初学者 \*

\*/

val conf = new SparkConf() //创建SparkConf对象

conf.setAppName("Wow,My First Spark in IDEA!") //设置应用程序的名称，在程序运行的监控界面可以看到名称

conf.setMaster("local") //此时，程序在本地运行，不需要安装Spark集群

/\*\*

\* 第2步：创建SparkContext对象

\* SparkContext是Spark程序所有功能的唯一入口，无论是采用Scala、Java、Python、R等都必须有一个SparkContext

\* SparkContext核心作用：初始化Spark应用程序运行所需要的核心组件，包括DAGScheduler、TaskScheduler、SchedulerBackend

\* 同时还会负责Spark程序往Master注册程序等

\* SparkContext是整个Spark应用程序中最为至关重要的一个对象

\*/

val sc = new SparkContext(conf) //创建SparkContext对象，通过传入SparkConf实例来定制Spark运行的具体参数和配置信息

/\*\*

\* 第3步：根据具体的数据来源（HDFS、HBase、Local FS、DB、S3等）通过SparkContext来创建RDD

\* RDD的创建基本有三种方式：根据外部的数据来源（例如HDFS）、根据Scala集合、由其它的RDD操作

\* 数据会被RDD划分成为一系列的Partitions，分配到每个Partition的数据属于一个Task的处理范畴

\*/

//val lines: RDD[String] = sc.textFile("D://Big\_Data\_Software//spark-1.6.0-bin-hadoop2.6//README.md", 1) //读取本地文件并设置为一个Partion

val lines = sc.textFile("D://Big\_Data\_Software//spark-1.6.0-bin-hadoop2.6//README.md", 1) //读取本地文件并设置为一个Partion

/\*\*

\* 第4步：对初始的RDD进行Transformation级别的处理，例如map、filter等高阶函数等的编程，来进行具体的数据计算

\* 第4.1步：讲每一行的字符串拆分成单个的单词

\*/

val words = lines.flatMap { line => line.split(" ")} //对每一行的字符串进行单词拆分并把所有行的拆分结果通过flat合并成为一个大的单词集合

/\*\*

\* 第4步：对初始的RDD进行Transformation级别的处理，例如map、filter等高阶函数等的编程，来进行具体的数据计算

\* 第4.2步：在单词拆分的基础上对每个单词实例计数为1，也就是word => (word, 1)

\*/

val pairs = words.map { word => (word, 1) }

/\*\*

\* 第4步：对初始的RDD进行Transformation级别的处理，例如map、filter等高阶函数等的编程，来进行具体的数据计算

\* 第4.3步：在每个单词实例计数为1基础之上统计每个单词在文件中出现的总次数

\*/

val wordCounts = pairs.reduceByKey(\_+\_) //对相同的Key，进行Value的累计（包括Local和Reducer级别同时Reduce）

wordCounts.foreach(wordNumberPair => println(wordNumberPair.\_1 + " : " + wordNumberPair.\_2))

sc.stop()

}

}

在代码去右击 选择点击 run”wordCount” 来运行程序。在生成环境下肯定是写自动化shell 脚本自动提交程序的。

1. 编写Cluster 模式代码

import org.apache.spark.SparkConf

import org.apache.spark.SparkContext

import org.apache.spark.rdd.RDD

/\*\*

\* 使用Scala开发集群运行的Spark WordCount程序

\* @author DT大数据梦工厂

\* 新浪微博：http://weibo.com/ilovepains/

\*/

object WordCount\_Cluster {

def main(args: Array[String]){

/\*\*

\* 第1步：创建Spark的配置对象SparkConf，设置Spark程序的运行时的配置信息，

\* 例如说通过setMaster来设置程序要链接的Spark集群的Master的URL,如果设置

\* 为local，则代表Spark程序在本地运行，特别适合于机器配置条件非常差（例如

\* 只有1G的内存）的初学者 \*

\*/

val conf = new SparkConf() //创建SparkConf对象

conf.setAppName("Wow,My First Spark App!") //设置应用程序的名称，在程序运行的监控界面可以看到名称

// conf.setMaster("spark://Master:7077") //此时，程序在Spark集群

/\*\*

\* 第2步：创建SparkContext对象

\* SparkContext是Spark程序所有功能的唯一入口，无论是采用Scala、Java、Python、R等都必须有一个SparkContext

\* SparkContext核心作用：初始化Spark应用程序运行所需要的核心组件，包括DAGScheduler、TaskScheduler、SchedulerBackend

\* 同时还会负责Spark程序往Master注册程序等

\* SparkContext是整个Spark应用程序中最为至关重要的一个对象

\*/

val sc = new SparkContext(conf) //创建SparkContext对象，通过传入SparkConf实例来定制Spark运行的具体参数和配置信息

/\*\*

\* 第3步：根据具体的数据来源（HDFS、HBase、Local FS、DB、S3等）通过SparkContext来创建RDD

\* RDD的创建基本有三种方式：根据外部的数据来源（例如HDFS）、根据Scala集合、由其它的RDD操作

\* 数据会被RDD划分成为一系列的Partitions，分配到每个Partition的数据属于一个Task的处理范畴

\*/

//val lines = sc.textFile("hdfs://Master:9000/library/wordcount/input/Data") //读取HDFS文件并切分成不同的Partions

val lines = sc.textFile("/library/wordcount/input/Data") //读取HDFS文件并切分成不同的Partions

/\*\*

\* 第4步：对初始的RDD进行Transformation级别的处理，例如map、filter等高阶函数等的编程，来进行具体的数据计算

\* 第4.1步：讲每一行的字符串拆分成单个的单词

\*/

val words = lines.flatMap { line => line.split(" ")} //对每一行的字符串进行单词拆分并把所有行的拆分结果通过flat合并成为一个大的单词集合

/\*\*

\* 第4步：对初始的RDD进行Transformation级别的处理，例如map、filter等高阶函数等的编程，来进行具体的数据计算

\* 第4.2步：在单词拆分的基础上对每个单词实例计数为1，也就是word => (word, 1)

\*/

val pairs = words.map { word => (word, 1) }

/\*\*

\* 第4步：对初始的RDD进行Transformation级别的处理，例如map、filter等高阶函数等的编程，来进行具体的数据计算

\* 第4.3步：在每个单词实例计数为1基础之上统计每个单词在文件中出现的总次数

\*/

val wordCountsorderd = pairs.reduceByKey(\_+\_).map(pair=>(pair.\_2,pair.\_1)).sortByKey(false).map(pair =>(pair.\_1.pair.\_2)) //对相同的Key，进行Value的累计（包括Local和Reducer级别同时Reduce）  
  
wordCountsorderd.collect.foreach(wordNumberPair => println(wordNumberPair.\_1 + " : " + wordNumberPair.\_2))

sc.stop()

}

}

将程序达成jar 包

点击 file-> project structure, 在弹出的页面点击 Artifacts，点击右侧的“+”，选择jar –> from modules with dependencies,在弹出的页面中，设置好main class 然后点击ok，在弹出页面修改Name（系统生成的name不规范） 、导出位置并删除 scala和spark的jar（因为集群环境中已经存在）点击ok 。然后在菜单栏中点击build –> Artifacts ,在弹出按钮中，点击bulid，会自动开始打包。

在hadoop中执行wordcount 方法。

将jar 放到linux 系统某个目录中。执行

./spark-submit

--class com.dt.spark. WordCount\_Cluster

--master spark://master:7077

/root/documents/sparkapps/wordcount.jar

注意事项：

为什么不能再ide开发环境中，直接发布spark程序到spark集群中？

1. 开发机器的内存和cores的限制，默认情况情况下，spark程序的dirver 在提交spark程序的机器上， 如果在idea中提交程序的话，那idea机器就必须非常强大。
2. Dirver 要指挥workers的运行并频繁的发生同学，如果开发环境和spark集群不在同样一个网络下，就会出现任务丢失，运行缓慢等多种不必要的问题。
3. 这是不安全的。