课程内容

使用ide开发spark 实战

使用ide 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. 打开ide 新建scala project

点击 file -> new ->Scala Project ,在弹出的对话框中弹性project name 为“WordCount”，默认点击next，点击finish的。

1. 修改scala版本

项目创建完成后 默认使用的是scala 的2.11.7 版本。要手动将版本换成2.10.X。在项目名称右击选择 properties，在弹出窗口 点击，scala Compiler，在右侧窗口，选中 Use Project settings, 将 scala Installation 修改为 Latest 2.10 bundle(dynamic).点击apply，点击ok。scala版本变成2.10.6。

1. 找到依赖的spark jar文件并导入到eclipse中。

所依赖的jar文件是

spark-1.6.0-bin-hadoop2.6\lib\spark-assembly-1.6.0-hadoop2.6.0.jar。

在项目名称上右击，选择 build path ->configure build path。 在弹出框中点击library， 点击右侧的addExternalJARs，然后选择

park-assembly-1.6.0-hadoop2.6.0.jar 点击打开，然后点击ok。

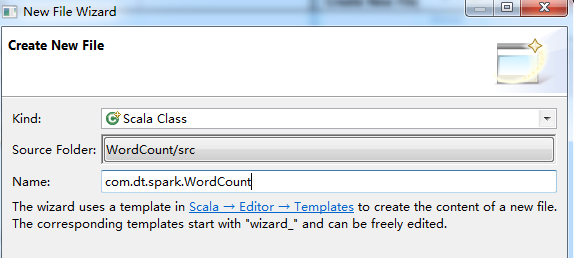
二：spark程序开发步骤

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 Name 中，在 增加WordCount。点击finish。如下图所示。



在方法内部 讲关键字class 改成object ，然后创建main 方法。（local模式代码）方法如下

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 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("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()

}

}

在运行过程中会出现WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable。java.io.IOException: Could not locate executable null\bin\winutils.exe in the Hadoop binaries. 这个错误。 但是在local模式下，这个是正常的。因为spark是和hadoop编译在一起的， 我们在window 下开发，缺少hadoop的配置。这不是程序错误，也不影响我们的任何功能。

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

sc.stop()

}

}

将程序达成jar 包

在项目名称上右击点击 export 选择 java 下的 jar file， 点击next， 选择输出目录，输入文件名，点击next，点击next，

然后点击完成。导出jar 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

也可以将以上命令保存到.sh文件中，直接执行 sh文件即可。