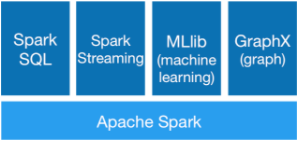
## Spark简介

Spark是一个快如闪电的集群计算框架

Saprk具有以下特点：

1. 速度，是mapreduce的100倍，是磁盘的10倍
2. 通用：支持sql、streaming、深度学习、图像处理等等

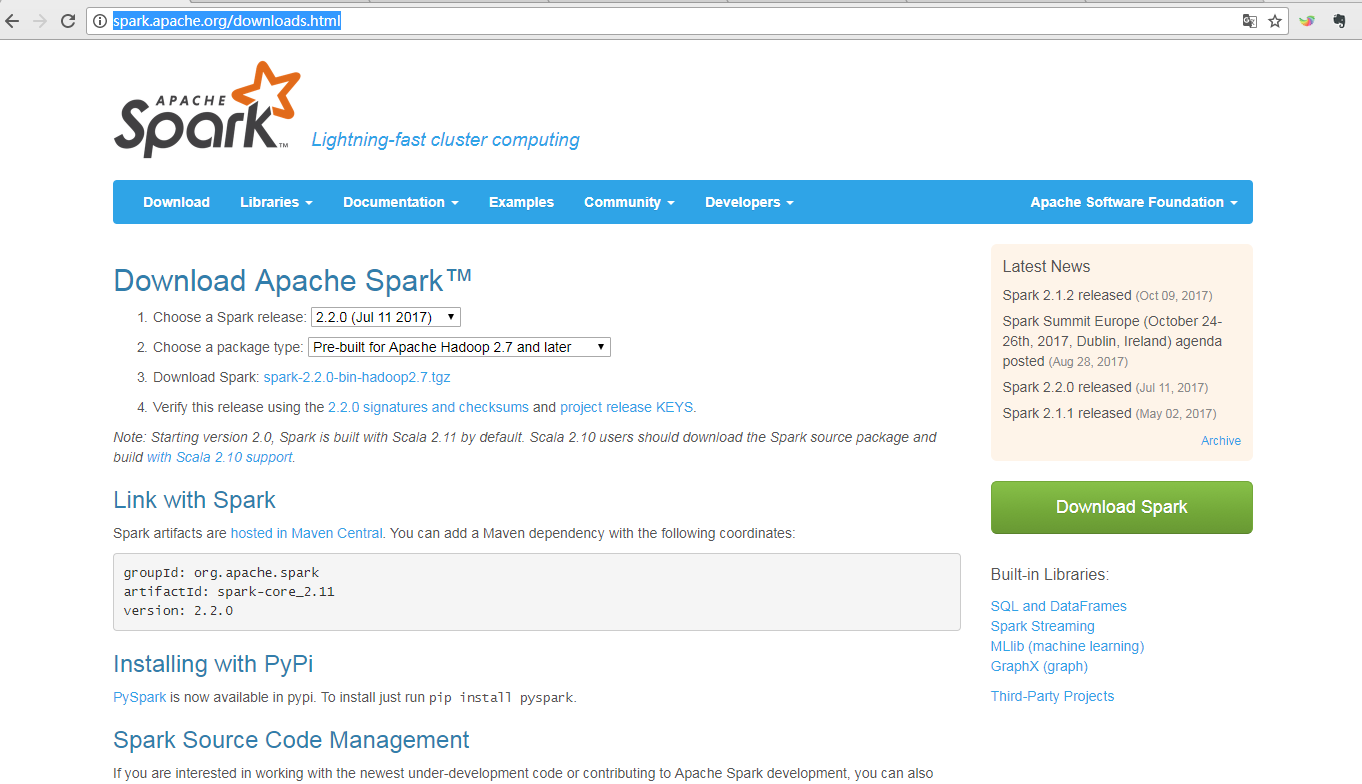


1. 支持多种语言，例如java、scala、python、R
2. 到处运行：Spark runs on Hadoop, Mesos, standalone, or in the cloud. It can access diverse data sources including HDFS, Cassandra, HBase, and S3.



## Spark安装

1. 下载 <http://spark.apache.org/downloads.html>



1. 安装前提

安装spark前必须安装jdk、scala，安装教程略

1. 安装

3.1解压spark

tar –zxvf spark-1.6.1-bin-hadoop2.6.tgz

3.2 配置环境变量

export SPARK\_HOME=/usr/tools/spark-1.6.1-bin-hadoop2.6

export PATH=$PATH:$SPARK\_HOME/bin

3.3是环境变量生效

source /etc/profile

1. spark配置

4.1进入spark的conf目录

cd /usr/tools/spark-1.6.1-bin-hadoop2.6/conf

cp spark-env.sh.template spark-env.sh

cp log4j.properties.template log4j.properties

cp slaves.template slaves

4.2 编辑spark-env.sh

export SCALA\_HOME=/usr/tools/scala-2.11.8

export JAVA\_HOME=/usr/tools/jdk1.7.0\_67

export SPARK\_WORKER\_MEMORY=1G

export HADOOP\_CONF\_DIR=/usr/tools/hadoop-2.6.4/etc/hadoop

4.3 编辑slaves

jokeros1

jokeros2

jokeros3

4.4将spark拷贝到子节点上然后配置环境变量并使其生效。

scp -r spark-1.6.1-bin-hadoop2.6 root@jokeros2:/usr/tools

scp -r spark-1.6.1-bin-hadoop2.6 root@jokeros3:/usr/tools

5. 启动

进入主节点的sbin目录

运行start-all.sh

主节点上Master Worker两个进程，子节点上Worker一个进程

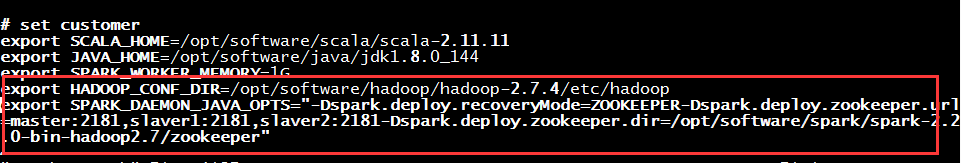
<http://jokeros1:8080>

<http://192.168.0.142:4040/jobs>

1. 配置高可用(也可以省略)

在spark-2.2.0-bin-hadoop2.7/conf/spark-env.sh下配置

|  |
| --- |
| export SPARK\_DAEMON\_JAVA\_OPTS="-Dspark.deploy.recoveryMode=ZOOKEEPER-Dspark.deploy.zookeeper.url  =master:2181,slaver1:2181,slaver2:2181-Dspark.deploy.zookeeper.dir=/opt/software/spark/spark-2.2  .0-bin-hadoop2.7/zookeeper" |

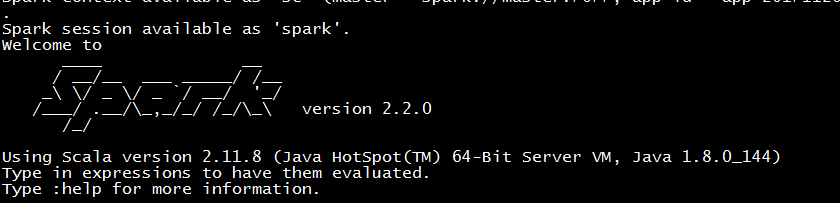


## Spark Shell

1. 连接spark-shell客户端

|  |
| --- |
| spark-shell --master spark://master:7077 |

出现如下界面成功



1. 读取文件

|  |
| --- |
| val file = sc.textFile("hdfs://centos1:9000/wctest.txt") |

1. 将读取的文件转换成RDD格式

|  |
| --- |
| val result = file.flatMap(x=>x.split("\\s")).map(x=>(x,1)).reduceByKey((x1,x2)=>x1+x2) |

1. 将结果保存起来

|  |
| --- |
| result.saveAsTextFile("/sparkwcresult") |

## 数据处理应用场景

批处理 bi分析，olap

流处理 实时需求

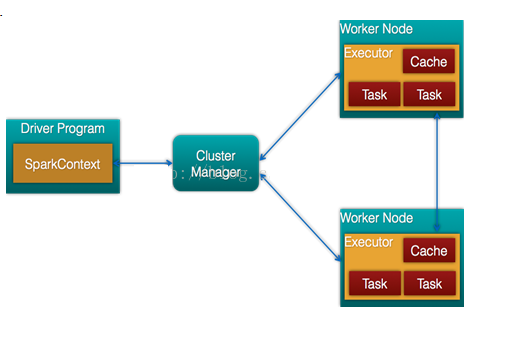
交互式处理 开发调试，数据挖，掘模型计算，机器学期

## 数据处理方式

1. 函数式数据处理，使用某种语言，通过方法的调用完成数据的处理
2. sql

## Spark 框架

Spark框架组成示意图



Cluster Manager：在standalone模式中即为Master主节点，控制整个集群，监控worker。在YARN模式中为资源管理器

Worker节点：从节点，负责控制计算节点，启动Executor或者Driver。

Driver： 运行Application 的main()函数

Executor：执行器，是为某个Application运行在worker node上的一个进程

## Scala连接spark

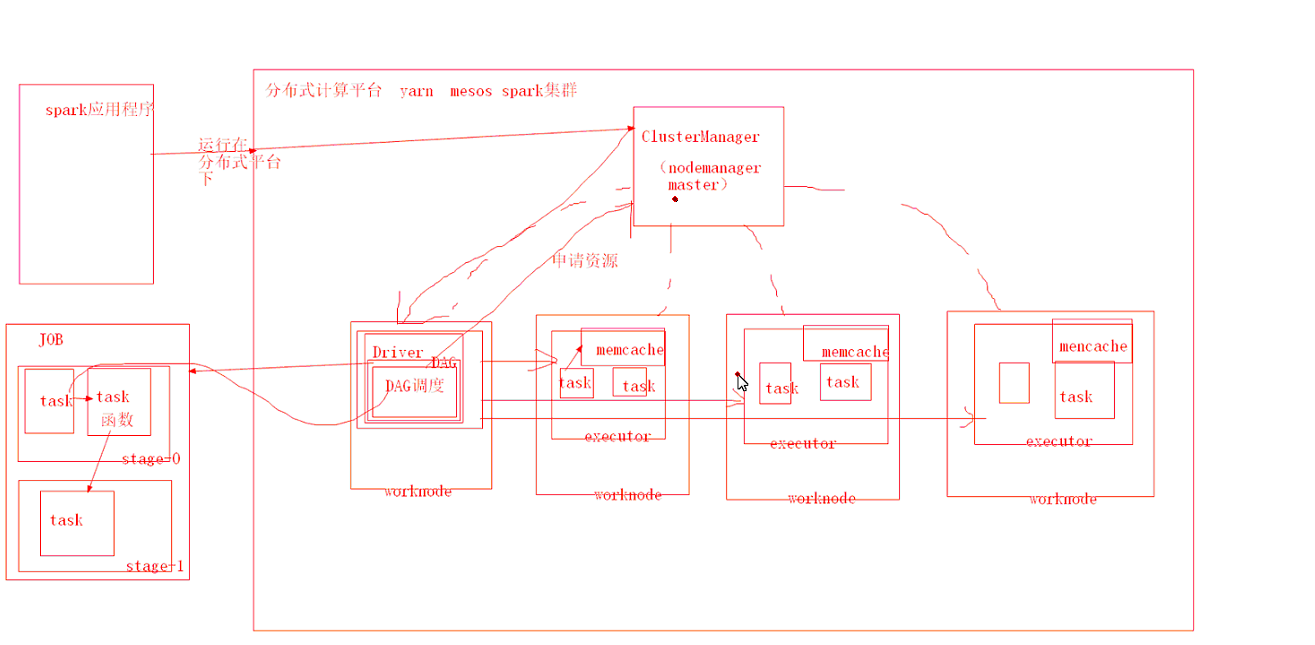
1. 创建scala maven项目
2. 添加spark依赖

|  |
| --- |
| *<!-- https://mvnrepository.com/artifact/org.apache.spark/spark-core -->* <**dependency**>  <**groupId**>org.apache.spark</**groupId**>  <**artifactId**>spark-core\_2.11</**artifactId**>  <**version**>2.2.0</**version**> </**dependency**> |

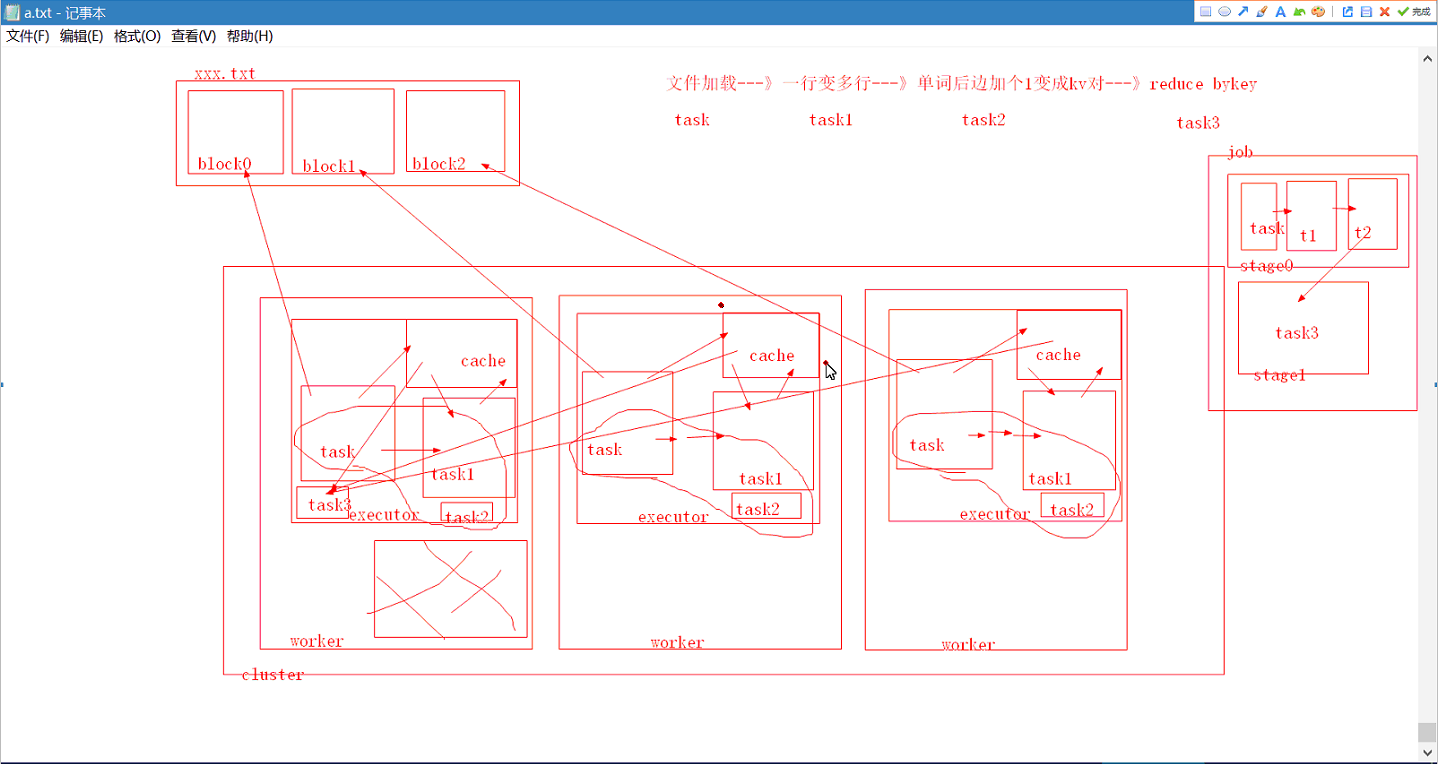
1. 编写WordCount

|  |
| --- |
| **package** top.xiesem.bd14 **import** org.apache.spark.{SparkConf, SparkContext}**object** WordCount {   **def** main(args: Array[String]): Unit = {  *// 创建SparkConf对象  // 设置分布式运行平台，和appname  // 使用master指定运行平台，yarn，standalong,mesos,local  // local:单线程;local[N]:n个线程;local[\*]:本地的cpu有多少个核就启动多少个线程* **val** conf = **new** SparkConf().setMaster(**"local[2]"**).setAppName(**"WordCount App"**)  *// 构建SparkContext对象* **val** sc = **new** SparkContext(conf)  *// 加载数据源，获取RDD对象  // 读取本地文件 file:///c:\\desktop\\read.md  // 读取hdfs上文件 hdfs://master:9000/file* **val** fileRdd = sc.textFile(**"/readme.md"**)  *// 数据处理开始* **val** wordRDD = fileRdd.flatMap(line => line.split(**"\\s"**))  **val** result = wordRDD.map(x => (x, 1)).reduceByKey(\_ + \_)  *// println(s"$result ")* result.saveAsTextFile(**"/spark/WordCountResult"**)   *// 简写形式 // fileRdd.flatMap(\_.split("\\s")).map((\_,1)).reduceByKey(\_ + \_).saveAsTextFile("/spark/WordCountResult")* } } |

Spark程序运行在分布式环境下，先向Cluster manager(在standlong模式下的master)申请资源，由Cluster manager指定worknode，Driver进行执行、解析生成DAG，同时生成DAG调度，由DAG调度执行job，job将具体的事务叫给task(实质上就是函数)进行执行。处理完这些driver之后，将具体的task分配到集群中交给executor进行具体的执行。在executor中task的结果保存在memcache(内存)中



不同的stage是按照shuffle进行划分的。



Spark程序的数据处理过程，是在driver上进行解析编译的，在具体对数据进行处理的时候需要一个触发点来触发执行计算过程

Rdd处理方法(函数)有两种

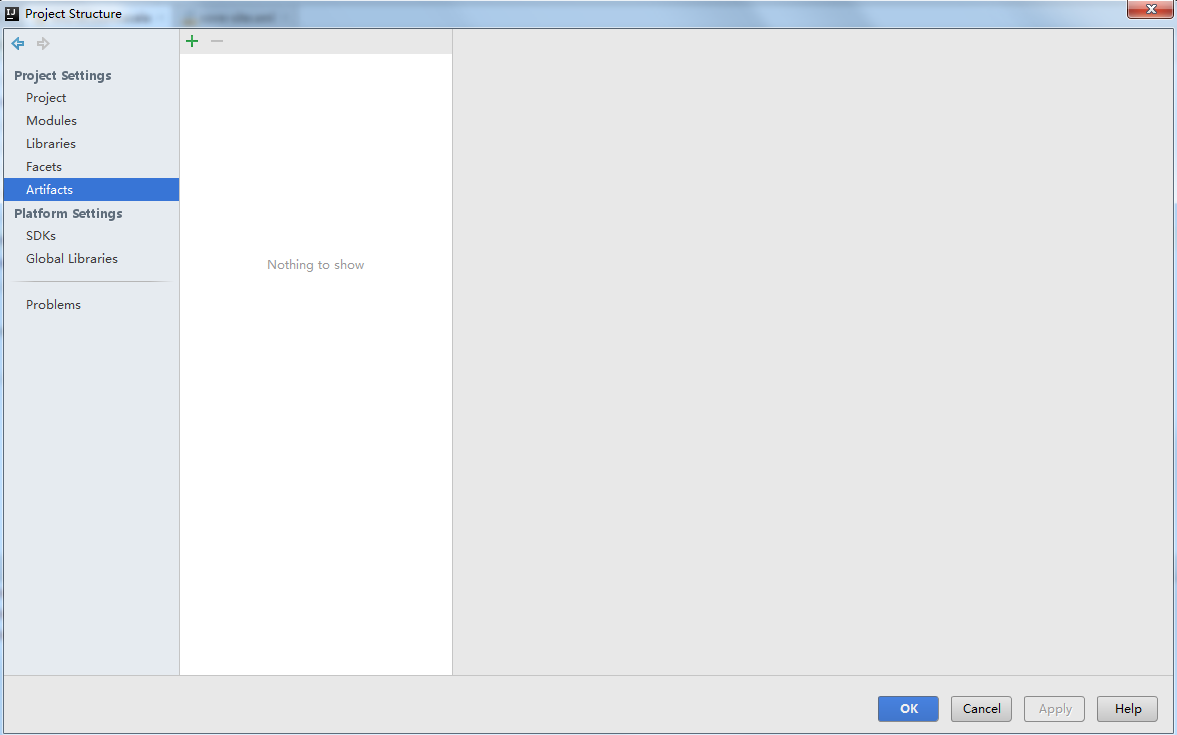
Transformation 转换 对数据进行计算转换，只记录处理过程而不真正计算数据例如，一行变多行，一个单词变成kv，把一个key下的所有value累加

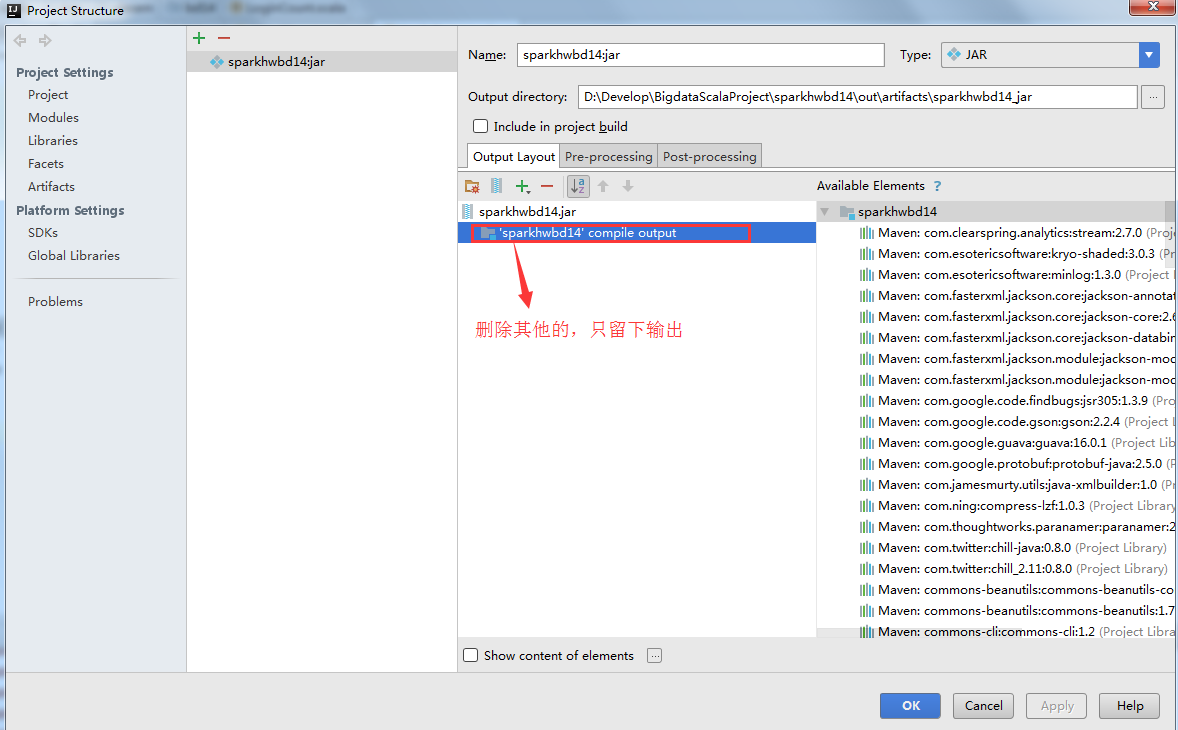
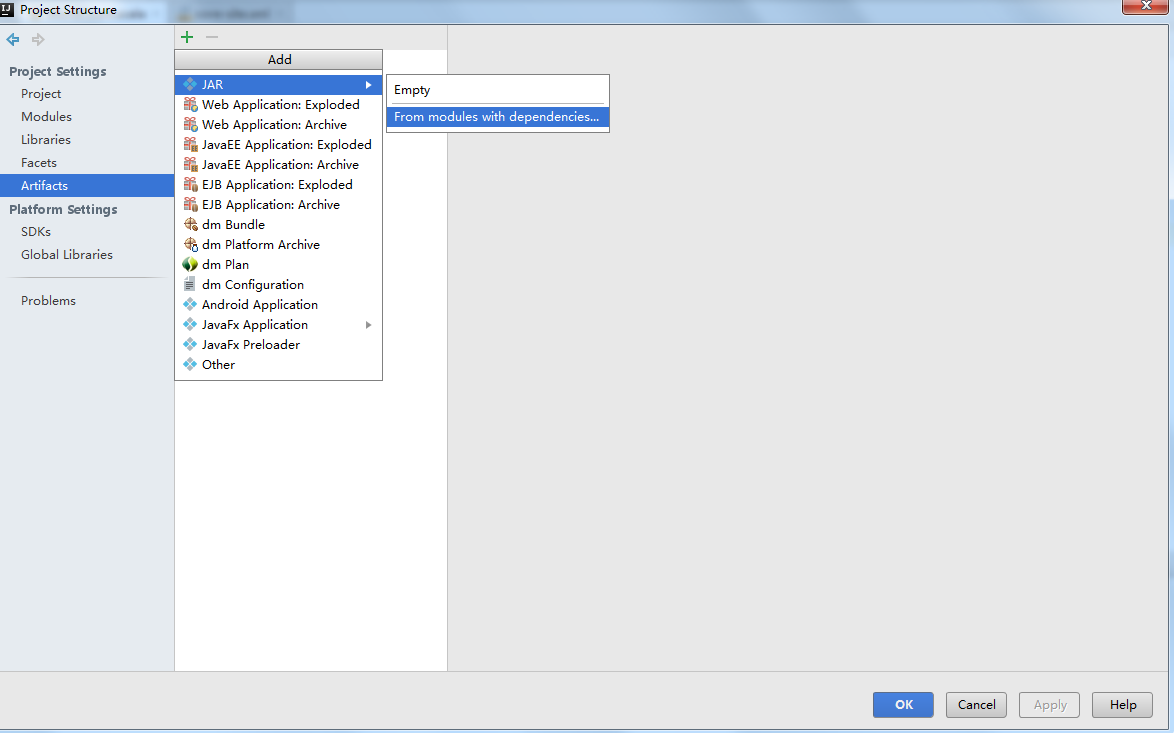
Action 行为 对数据进行读值，触发对数据的计算过程，触发后把action前面的所有的transformation启动执行，一般不改变数据，只是对数据的结果值进行读取

### Spark的standlong模式

Intellj中的程序直接运行在spark的standalone模式下

1. 创建artifact(项目编辑打包)(Ctrl+Shift+Alt+s)



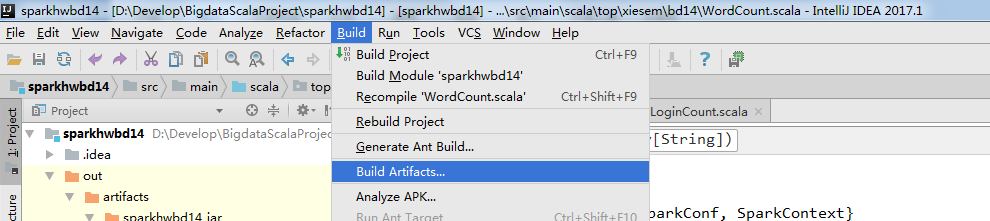


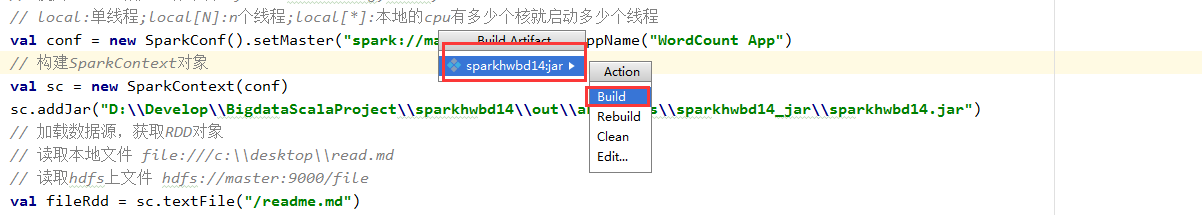
1. 修改conf的master设置：

|  |
| --- |
| **val** conf = **new** SparkConf().setMaster(**"spark://master:7077"**).setAppName(**"WordCount App"**) |

1. 在sc下添加我们intellj打的包，打包过程

build---》build artifacts---》build：





1. 添加代码如下：

|  |
| --- |
| sc.addJar(**"D:\\Develop\\BigdataScalaProject\\sparkhwbd14\\out\\artifacts\\sparkhwbd14\_jar\\sparkhwbd14.jar"**) |

1. 完整代码

|  |
| --- |
| **package** top.xiesem.bd14  **import** org.apache.spark.{SparkConf, SparkContext}  **object** WordCount {   **def** main(args: Array[String]): Unit = {  *// 创建SparkConf对象  // 设置分布式运行平台，和appname  // 使用master指定运行平台，yarn，standalong,mesos,local  // local:单线程;local[N]:n个线程;local[\*]:本地的cpu有多少个核就启动多少个线程* **val** conf = **new** SparkConf().setMaster(**"spark://master:7077"**).setAppName(**"WordCount App"**)  *// 构建SparkContext对象* **val** sc = **new** SparkContext(conf)  sc.addJar(**"D:\\Develop\\BigdataScalaProject\\sparkhwbd14\\out\\artifacts\\sparkhwbd14\_jar\\sparkhwbd14.jar"**)  *// 加载数据源，获取RDD对象  // 读取本地文件 file:///c:\\desktop\\read.md  // 读取hdfs上文件 hdfs://master:9000/file* **val** fileRdd = sc.textFile(**"/readme.md"**)  *// 数据处理开始* **val** wordRDD = fileRdd.flatMap(line => line.split(**"\\s"**))  **val** result = wordRDD.map(x => (x, 1)).reduceByKey(\_ + \_)  *// println(s"$result ") // result.saveAsTextFile("/spark/WordCountResult")* result.foreach(*println*)  sc.stop()  *// 简写形式  // fileRdd.flatMap(\_.split("\\s")).map((\_,1)).reduceByKey(\_ + \_).saveAsTextFile("/spark/WordCountResult")* } } |