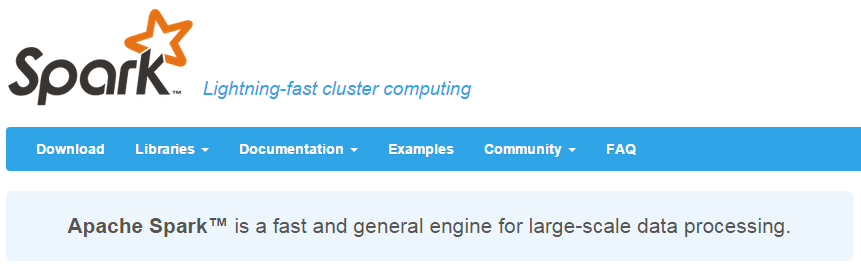
尚硅谷大数据技术之Spark基础解析

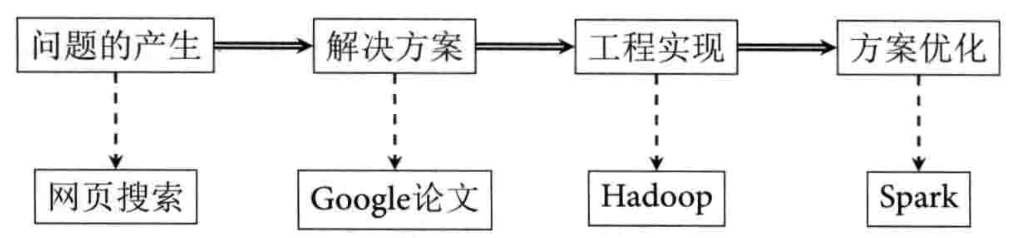
(作者：尚硅谷大数据研发部)

版本：V1.1

# 第1章 Spark概述

## 什么是Spark



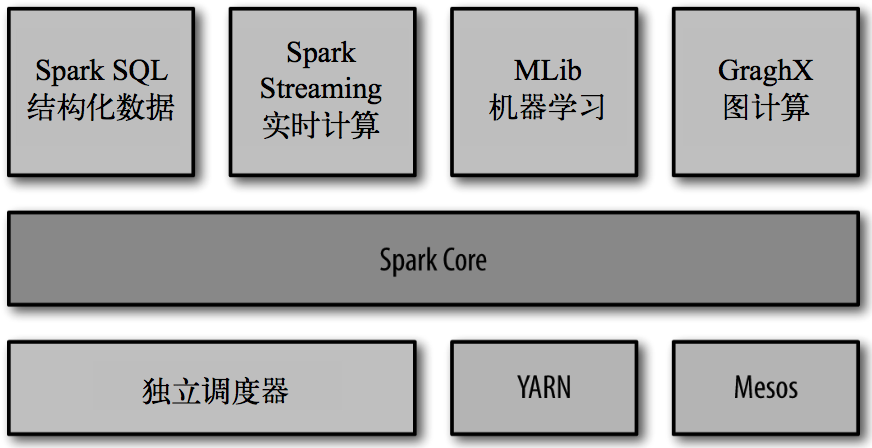


Spark是一种快速、通用、可扩展的大数据分析引擎，2009年诞生于加州大学伯克利分校AMPLab，2010年开源，2013年6月成为Apache孵化项目，2014年2月成为Apache顶级项目。项目是用Scala进行编写。

目前，Spark生态系统已经发展成为一个包含多个子项目的集合，其中包含SparkSQL、Spark Streaming、GraphX、MLib、SparkR等子项目，Spark是基于内存计算的大数据并行计算框架。除了扩展了广泛使用的 MapReduce 计算模型，而且高效地支持更多计算模式，包括交互式查询和流处理。Spark 适用于各种各样原先需要多种不同的分布式平台的场景，包括批处理、迭代算法、交互式查询、流处理。通过在一个统一的框架下支持这些不同的计算，Spark 使我们可以简单而低耗地把各种处理流程整合在一起。而这样的组合，在实际的数据分析 过程中是很有意义的。不仅如此，Spark 的这种特性还大大减轻了原先需要对各种平台分 别管理的负担。

大一统的软件栈，各个组件关系密切并且可以相互调用，这种设计有几个好处：1、软件栈中所有的程序库和高级组件 都可以从下层的改进中获益。2、运行整个软件栈的代价变小了。不需要运 行 5 到 10 套独立的软件系统了，一个机构只需要运行一套软件系统即可。系统的部署、维护、测试、支持等大大缩减。3、能够构建出无缝整合不同处理模型的应用。

Spark的内置项目如下：



Spark Core：实现了 Spark 的基本功能，包含任务调度、内存管理、错误恢复、与存储系统 交互等模块。Spark Core 中还包含了对弹性分布式数据集(resilient distributed dataset，简称RDD)的 API 定义。

Spark SQL：是 Spark 用来操作结构化数据的程序包。通过 Spark SQL，我们可以使用 SQL 或者 Apache Hive 版本的 SQL 方言(HQL)来查询数据。Spark SQL 支持多种数据源，比 如 Hive 表、Parquet 以及 JSON 等。

Spark Streaming：是 Spark 提供的对实时数据进行流式计算的组件。提供了用来操作数据流的 API，并且与 Spark Core 中的 RDD API 高度对应。

Spark MLlib：提供常见的机器学习(ML)功能的程序库。包括分类、回归、聚类、协同过滤等，还提供了模型评估、数据 导入等额外的支持功能。

集群管理器：Spark 设计为可以高效地在一个计算节点到数千个计算节点之间伸缩计 算。为了实现这样的要求，同时获得最大灵活性，Spark 支持在各种集群管理器(cluster manager)上运行，包括 Hadoop YARN、Apache Mesos，以及 Spark 自带的一个简易调度 器，叫作独立调度器。

Spark得到了众多大数据公司的支持，这些公司包括Hortonworks、IBM、Intel、Cloudera、MapR、Pivotal、百度、阿里、腾讯、京东、携程、优酷土豆。当前百度的Spark已应用于凤巢、大搜索、直达号、百度大数据等业务；阿里利用GraphX构建了大规模的图计算和图挖掘系统，实现了很多生产系统的推荐算法；腾讯Spark集群达到8000台的规模，是当前已知的世界上最大的Spark集群。

## 1.2 Spark特点

1）快：与Hadoop的MapReduce相比，Spark基于内存的运算要快100倍以上，基于硬盘的运算也要快10倍以上。Spark实现了高效的DAG执行引擎，可以通过基于内存来高效处理数据流。计算的中间结果是存在于内存中的。

2）易用：Spark支持Java、Python和Scala的API，还支持超过80种高级算法，使用户可以快速构建不同的应用。而且Spark支持交互式的Python和Scala的shell，可以非常方便地在这些shell中使用Spark集群来验证解决问题的方法。

3）通用：Spark提供了统一的解决方案。Spark可以用于批处理、交互式查询（Spark SQL）、实时流处理（Spark Streaming）、机器学习（Spark MLlib）和图计算（GraphX）。这些不同类型的处理都可以在同一个应用中无缝使用。Spark统一的解决方案非常具有吸引力，毕竟任何公司都想用统一的平台去处理遇到的问题，减少开发和维护的人力成本和部署平台的物力成本。

4）兼容性：Spark可以非常方便地与其他的开源产品进行融合。比如，Spark可以使用Hadoop的YARN和Apache Mesos作为它的资源管理和调度器，器，并且可以处理所有Hadoop支持的数据，包括HDFS、HBase和Cassandra等。这对于已经部署Hadoop集群的用户特别重要，因为不需要做任何数据迁移就可以使用Spark的强大处理能力。Spark也可以不依赖于第三方的资源管理和调度器，它实现了Standalone作为其内置的资源管理和调度框架，这样进一步降低了Spark的使用门槛，使得所有人都可以非常容易地部署和使用Spark。此外，Spark还提供了在EC2上部署Standalone的Spark集群的工具。

## 1.3 Spark的用户和用途

我们大致把Spark的用例分为两类：数据科学应用和数据处理应用。也就对应的有两种人群：数据科学家和工程师。

数据科学任务：主要是数据分析领域，数据科学家要负责分析数据并建模，具备 SQL、统计、预测建模(机器学习)等方面的经验，以及一定的使用 Python、 Matlab 或 R 语言进行编程的能力。

数据处理应用：工程师定义为使用 Spark 开发 生产环境中的数据处理应用的软件开发者，通过对接Spark的API实现对处理的处理和转换等任务。

# 第2章 Spark角色介绍及运行模式

## 2.1 集群角色



从物理部署层面上来看，Spark主要分为两种类型的节点，Master节点和Worker节点：Master节点主要运行集群管理器的中心化部分，所承载的作用是分配Application到Worker节点，维护Worker节点，Driver，Application的状态。Worker节点负责具体的业务运行。

从Spark程序运行的层面来看，Spark主要分为驱动器节点和执行器节点。

## 2.2 运行模式

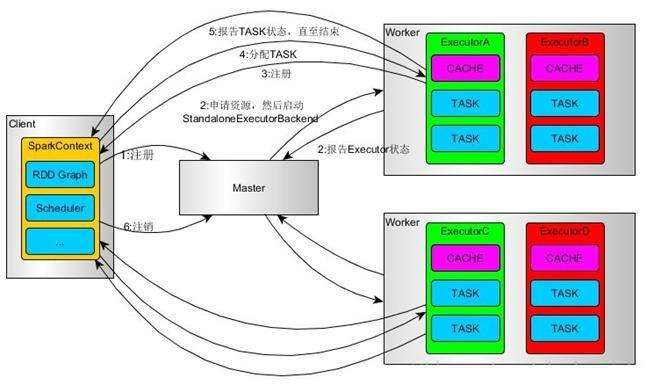
**1）Local模式:** Local模式就是运行在一台计算机上的模式，通常就是用于在本机上练手和测试。它可以通过以下集中方式设置master。

local: 所有计算都运行在一个线程当中，没有任何并行计算，通常我们在本机执行一些测试代码，或者练手，就用这种模式;

local[K]: 指定使用几个线程来运行计算，比如local[4]就是运行4个worker线程。通常我们的cpu有几个core，就指定几个线程，最大化利用cpu的计算能力;

local[\*]: 这种模式直接帮你按照cpu最多cores来设置线程数了。

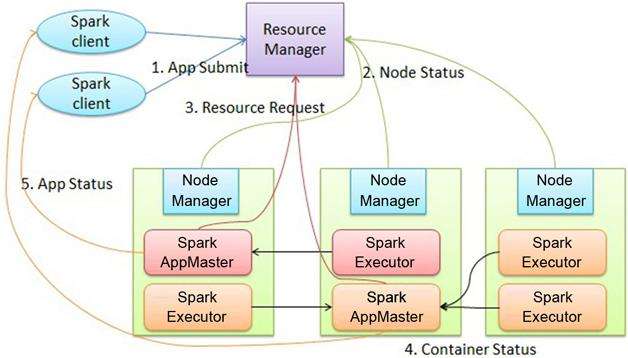
**2）Standalone模式:** 构建一个由Master+Slave构成的Spark集群，Spark运行在集群中。



**3）Yarn模式:** Spark客户端直接连接Yarn；不需要额外构建Spark集群。有yarn-client和yarn-cluster两种模式，主要区别在于：Driver程序的运行节点。

yarn-client：Driver程序运行在客户端，适用于交互、调试，希望立即看到app的输出

yarn-cluster：Driver程序运行在由RM（ResourceManager）启动的AP（APPMaster）适用于生产环境



**4）Mesos模式:**Spark客户端直接连接Mesos；不需要额外构建Spark集群。国内应用比较少，更多的是运用yarn调度。

# 第3章 Spark集群安装

## 3.1 Spark安装地址

1．官网地址

http://spark.apache.org/

2．文档查看地址

https://spark.apache.org/docs/2.1.1/

3．下载地址

https://spark.apache.org/downloads.html

## 3.2 Standalone模式安装

1）上传并解压spark安装包

[atguigu@hadoop102 sorfware]$ tar -zxvf spark-2.1.1-bin-hadoop2.7.tgz -C /opt/module/

[atguigu@hadoop102 module]$ mv spark-2.1.1-bin-hadoop2.7 spark

2）进入spark安装目录下的conf文件夹

[atguigu@hadoop102 module]$ cd spark/conf/

3）修改配置文件名称

[atguigu@hadoop102 conf]$ mv slaves.template slaves

[atguigu@hadoop102 conf]$ mv spark-env.sh.template spark-env.sh

4）修改slave文件，添加work节点：

[atguigu@hadoop102 conf]$ vim slaves

hadoop102

hadoop103

hadoop104

5）修改spark-env.sh文件，添加如下配置： 46 47 行

[atguigu@hadoop102 conf]$ vim spark-env.sh

SPARK\_MASTER\_HOST=hadoop102

SPARK\_MASTER\_PORT=7077 服务端口

6）分发spark包

[atguigu@hadoop102 module]$ xsync spark/

7）启动

[atguigu@hadoop102 spark]$ sbin/start-all.sh

[atguigu@hadoop102 spark]$ util.sh

================atguigu@hadoop102================

3330 Jps

3238 Worker

3163 Master

================atguigu@hadoop103================

2966 Jps

2908 Worker

================atguigu@hadoop104================

2978 Worker

3036 Jps

网页查看：hadoop102:8080

注意：如果遇到 “JAVA\_HOME not set” 异常，可以在sbin目录下的spark-config.sh 文件中加入如下配置：

export JAVA\_HOME=XXXX

8）提交任务&执行程序

[atguigu@hadoop102 spark]$ bin/spark-submit \

--class org.apache.spark.examples.SparkPi \ 主类

--master spark://hadoop102:7077 \ master

--executor-memory 1G \ 任务的资源

--total-executor-cores 2 \

./examples/jars/spark-examples\_2.11-2.1.1.jar \ jar包

100

./bin/spark-submit \

--class <main-class>

--master <master-url> \

--deploy-mode <deploy-mode> \

--conf <key>=<value> \

... # other options

<application-jar> \

[application-arguments]

参数说明：

--master spark://hadoop102:7077 指定Master的地址

--class: 你的应用的启动类 (如 org.apache.spark.examples.SparkPi)

--deploy-mode: 是否发布你的驱动到worker节点(cluster) 或者作为一个本地客户端 (client) (default: client)\*

--conf: 任意的Spark配置属性， 格式key=value. 如果值包含空格，可以加引号“key=value”

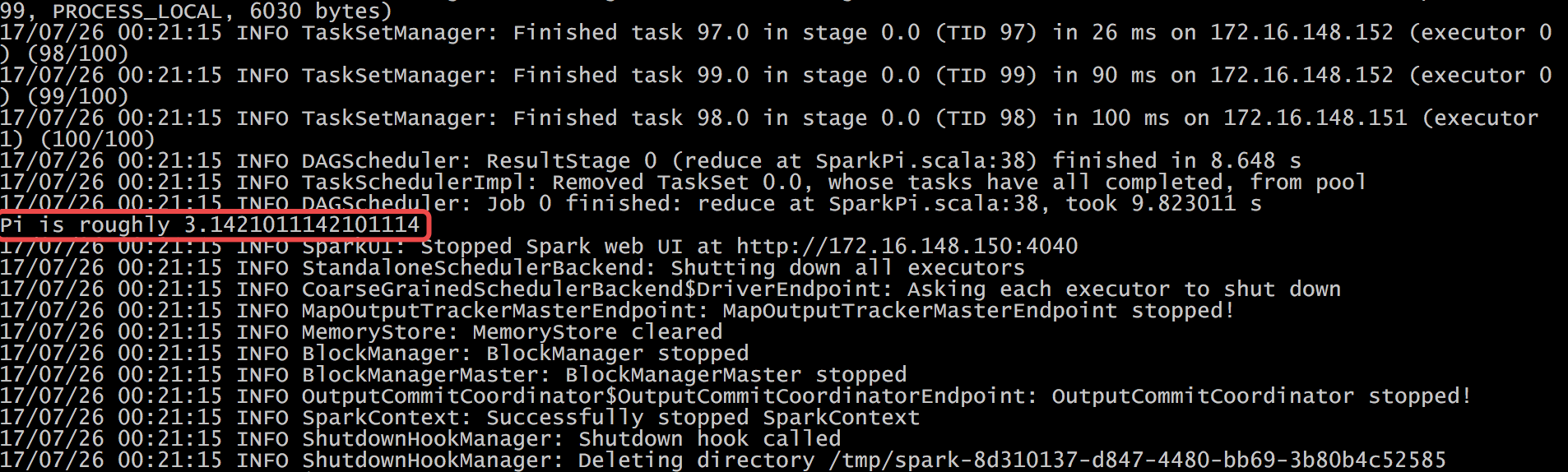
application-jar: 打包好的应用jar,包含依赖. 这个URL在集群中全局可见。 比如hdfs:// 共享存储系统， 如果是 file:// path， 那么所有的节点的path都包含同样的jar

application-arguments: 传给main()方法的参数

--executor-memory 1G 指定每个executor可用内存为1G

--total-executor-cores 2 指定每个executor使用的cup核数为2个

该算法是利用蒙特·卡罗算法求PI



9）启动spark shell

/opt/module/spark/bin/spark-shell \

--master spark://hadoop102:7077 \

--executor-memory 1g \

--total-executor-cores 2

**注意：**如果启动spark shell时没有指定master地址，但是也可以正常启动spark shell和执行spark shell中的程序，其实是启动了spark的local模式，该模式仅在本机启动一个进程，没有与集群建立联系。

Spark Shell中已经默认将SparkContext类初始化为对象sc。用户代码如果需要用到，则直接应用sc即可 sparksession 是sparksql

scala> sc.textFile("./word.txt")

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

.map((\_,1))

.reduceByKey(\_+\_)

.collect

res0: Array[(String, Int)] = Array((hive,1), (atguigu,1), (spark,1), (hadoop,1), (hbase,1))

## 3.3 JobHistoryServer配置

1）修改spark-default.conf.template名称

[atguigu@hadoop102 conf]$ mv spark-defaults.conf.template spark-defaults.conf

2）修改spark-default.conf文件，开启Log：

[atguigu@hadoop102 conf]$ vi spark-defaults.conf

spark.eventLog.enabled true

spark.eventLog.dir hdfs://hadoop102:9000/directory

注意：HDFS上的目录需要提前存在。

3）修改spark-env.sh文件，添加如下配置：

[atguigu@hadoop102 conf]$ vi spark-env.sh

export SPARK\_HISTORY\_OPTS="-Dspark.history.ui.port=4000

-Dspark.history.retainedApplications=3

-Dspark.history.fs.logDirectory=hdfs://hadoop102:9000/directory"

参数描述：

spark.eventLog.dir：Application在运行过程中所有的信息均记录在该属性指定的路径下；

spark.history.ui.port=4000  调整WEBUI访问的端口号为4000

spark.history.fs.logDirectory=hdfs://hadoop102:9000/directory  配置了该属性后，在start-history-server.sh时就无需再显式的指定路径，Spark History Server页面只展示该指定路径下的信息

spark.history.retainedApplications=3   指定保存Application历史记录的个数，如果超过这个值，旧的应用程序信息将被删除，这个是内存中的应用数，而不是页面上显示的应用数。

4）分发配置文件

[atguigu@hadoop102 conf]$ xsync spark-defaults.conf

[atguigu@hadoop102 conf]$ xsync spark-env.sh

5）启动历史服务

[atguigu@hadoop102 spark]$ sbin/start-history-server.sh

6）再次执行任务长度。

[atguigu@hadoop102 spark]$ bin/spark-submit \

--class org.apache.spark.examples.SparkPi \

--master spark://hadoop102:7077 \

--executor-memory 1G \

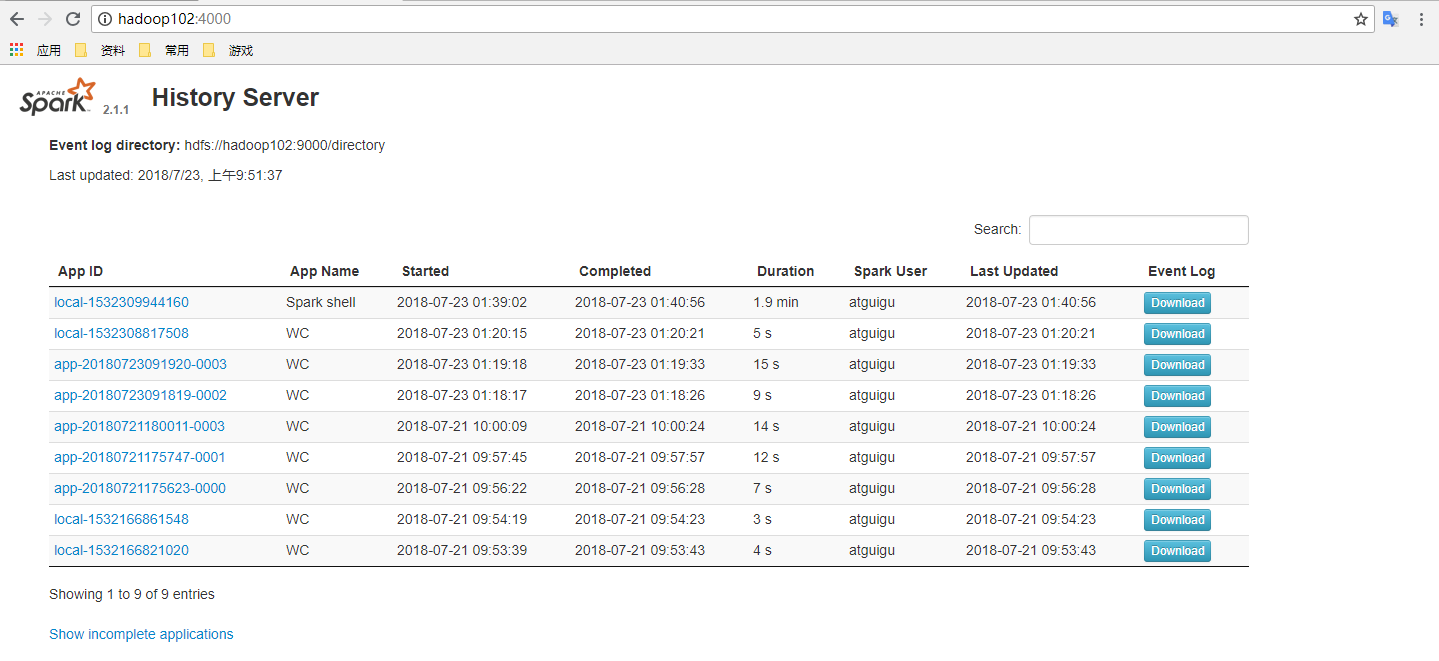
--total-executor-cores 2 \

./examples/jars/spark-examples\_2.11-2.1.1.jar \

100

7）查看历史服务

hadoop102:4000



## 3.4 HA配置

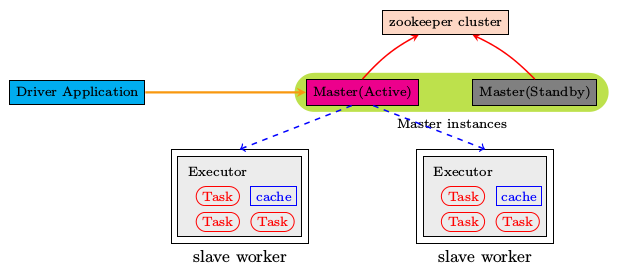


图1 HA架构图

1）zookeeper正常安装并启动

2）修改spark-env.sh文件添加如下配置：

[atguigu@hadoop102 conf]$ vi spark-env.sh

注释掉如下内容：

#SPARK\_MASTER\_HOST=hadoop102

#SPARK\_MASTER\_PORT=7077

添加上如下内容：

export SPARK\_DAEMON\_JAVA\_OPTS="

-Dspark.deploy.recoveryMode=ZOOKEEPER

-Dspark.deploy.zookeeper.url=hadoop102,hadoop103,hadoop104

-Dspark.deploy.zookeeper.dir=/spark"

3）分发配置文件

[atguigu@hadoop102 conf]$ xsync spark-env.sh

4）在hadoop102上启动全部节点

[atguigu@hadoop102 spark]$ sbin/start-all.sh

5）在hadoop103上单独启动master节点88

[atguigu@hadoop103 spark]$ sbin/start-master.sh

6）spark HA集群访问

/opt/module/spark/bin/spark-shell \

--master spark://hadoop102:7077,hadoop103:7077 \ 单独指定102也能

--executor-memory 2g \

--total-executor-cores 2

## 3.5 Yarn模式安装

1）修改hadoop配置文件yarn-site.xml,添加如下内容：

[atguigu@hadoop102 hadoop]$ vi yarn-site.xml

<!--是否启动一个线程检查每个任务正使用的物理内存量，如果任务超出分配值，则直接将其杀掉，默认是true -->

<property>

<name>yarn.nodemanager.pmem-check-enabled</name>

<value>false</value>

</property>

<!--是否启动一个线程检查每个任务正使用的虚拟内存量，如果任务超出分配值，则直接将其杀掉，默认是true -->

<property>

<name>yarn.nodemanager.vmem-check-enabled</name>

<value>false</value>

</property>

2）修改spark-env.sh，添加如下配置：

[atguigu@hadoop102 conf]$ vi spark-env.sh

YARN\_CONF\_DIR=/opt/module/hadoop-2.7.2/etc/hadoop

HADOOP\_CONF\_DIR=/opt/module/hadoop-2.7.2/etc/hadoop

3）分发配置文件

[atguigu@hadoop102 conf]$ xsync /opt/module/hadoop-2.7.2/etc/hadoop/yarn-site.xml

[atguigu@hadoop102 conf]$ xsync spark-env.sh

4）执行一个程序

[atguigu@hadoop102 spark]$ bin/spark-submit \

--class org.apache.spark.examples.SparkPi \

--master yarn \

--deploy-mode client \

./examples/jars/spark-examples\_2.11-2.1.1.jar \

100

注意：在提交任务之前需启动HDFS以及YARN集群。

# 第4章 IDEA环境应用

spark shell仅在测试和验证我们的程序时使用的较多，在生产环境中，通常会在IDE中编制程序，然后打成jar包，然后提交到集群，最常用的是创建一个Maven项目，利用Maven来管理jar包的依赖。

## 4.1 在IDEA中编写WordCount程序

1）创建一个Maven项目WordCount并导入依赖

<dependencies>  
 <dependency>  
 <groupId>org.apache.spark</groupId>  
 <artifactId>spark-core\_2.11</artifactId>  
 <version>2.1.1</version>  
 </dependency>  
</dependencies>

<build>

<finalName>WordCount</finalName>

<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>

</execution>

</executions>

</plugin>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-assembly-plugin</artifactId>

<version>3.0.0</version>

<configuration>

<archive>

<manifest>

<mainClass>WordCount(修改)</mainClass>

</manifest>

</archive>

<descriptorRefs>

<descriptorRef>jar-with-dependencies</descriptorRef>

</descriptorRefs>

</configuration>

<executions>

<execution>

<id>make-assembly</id>

<phase>package</phase>

<goals>

<goal>single</goal>

</goals>

</execution>

</executions>

</plugin>

</plugins>

</build>

2）编写代码

package com.atguigu  
  
import org.apache.spark.{SparkConf, SparkContext}  
  
object WordCount{  
  
 def main(args: Array[String]): Unit = {

//创建SparkConf并设置App名称  
 val conf = new SparkConf().setAppName("WC")

//创建SparkContext，该对象是提交Spark App的入口  
 val sc = new SparkContext(conf)  
  
 //使用sc创建RDD并执行相应的transformation和action  
 sc.textFile(args(0)).flatMap(\_.split(" ")).map((\_, 1)).reduceByKey(\_+\_, 1).sortBy(\_.\_2, false).saveAsTextFile(args(1))  
  
 sc.stop()  
 }  
}

3）打包到集群测试

bin/spark-submit \

--class WordCount \

--master spark://hadoop102:7077 \

WordCount.jar \

/word.txt \

/out

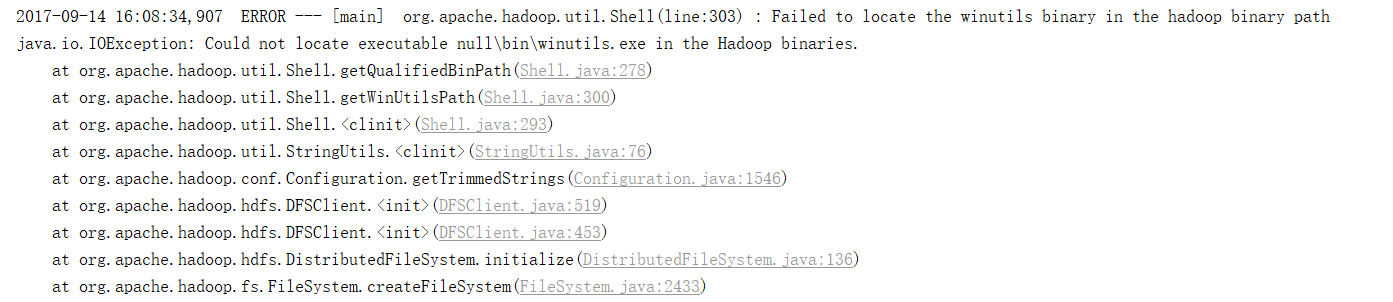
## 4.2 本地调试

本地Spark程序调试需要使用local提交模式，即将本机当做运行环境，Master和Worker都为本机。运行时直接加断点调试即可。如下：

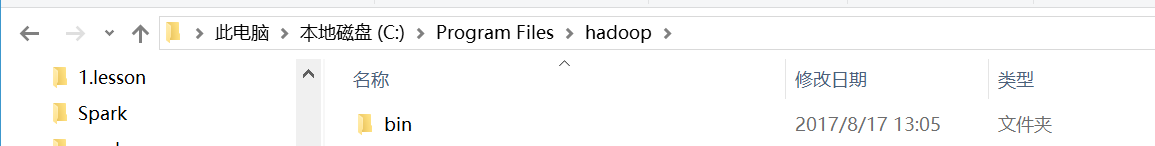
创建SparkConf的时候设置额外属性，表明本地执行：

val conf = new SparkConf().setAppName("WC").setMaster("local[\*]")

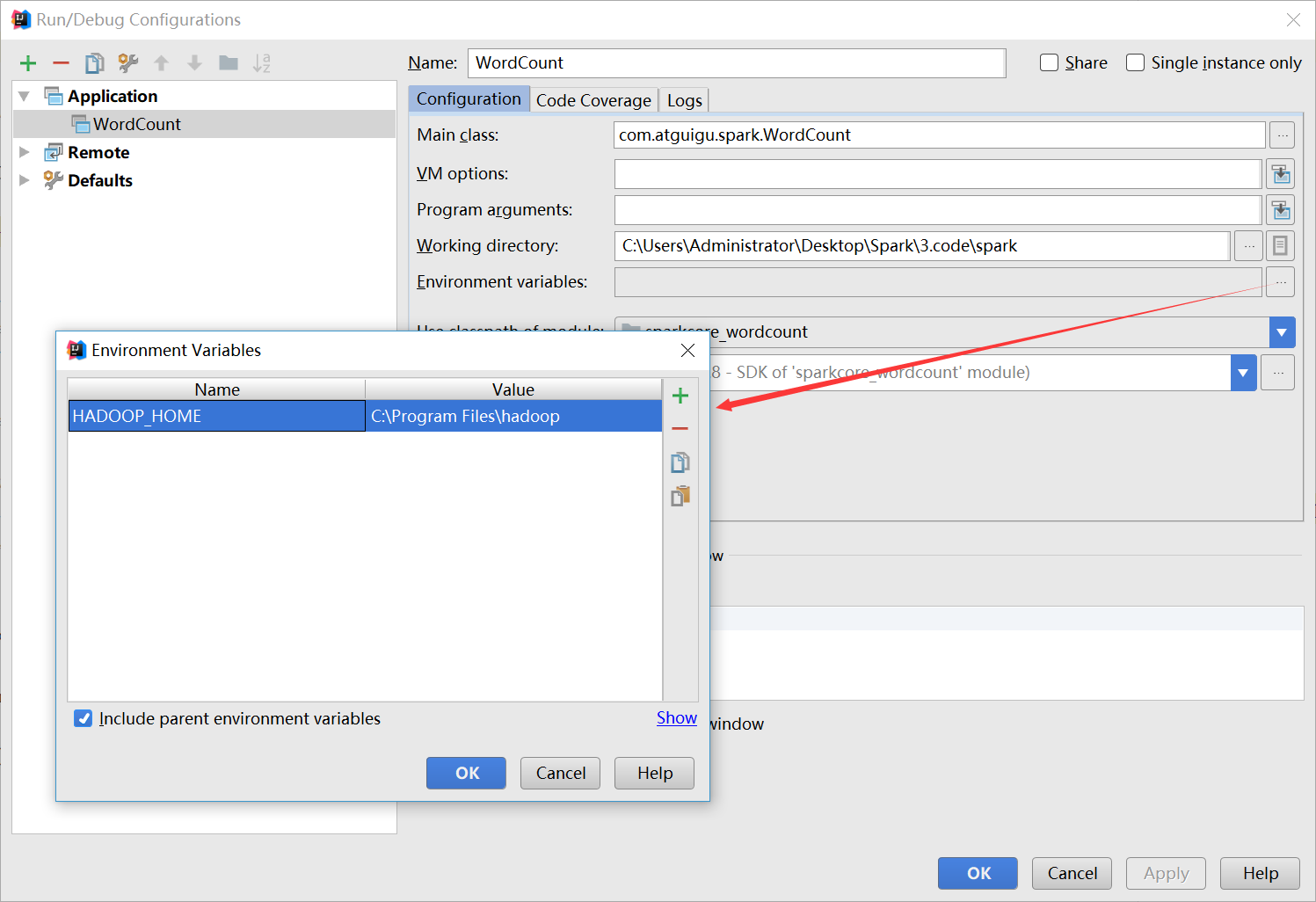
如果本机操作系统是windows，如果在程序中使用了hadoop相关的东西，比如写入文件到HDFS，则会遇到如下异常：



出现这个问题的原因，并不是程序的错误，而是用到了hadoop相关的服务，解决办法是将附加里面的hadoop-common-bin-2.7.3-x64.zip解压到任意目录。



在IDEA中配置Run Configuration，添加HADOOP\_HOME变量



## 4.3 远程调试

通过IDEA进行远程调试，主要是将IDEA作为Driver来提交应用程序，配置过程如下：

修改sparkConf，添加最终需要运行的Jar包、Driver程序的地址，并设置Master的提交地址：

val conf = new SparkConf().setAppName("WC")

.setMaster("spark://hadoop102:7077")  
.setJars(List("E:\\SparkIDEA\\spark\_test\\target\\WordCount.jar"))

然后加入断点，直接调试即可：

