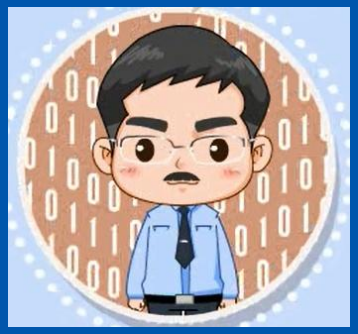


# 《Spark编程基础（Scala版）》

教材官网： http://dblab.xmu.edu.cn/post/spark/

*温馨提示：编辑幻灯片母版，可以修改每页PPT的厦大校徽和底部文字*

# 第4章 Spark环境搭建和使用方法

**（PPT版本号：2018年7月版本）**

扫一扫访问教材官网


林子雨

厦门大学计算机科学系

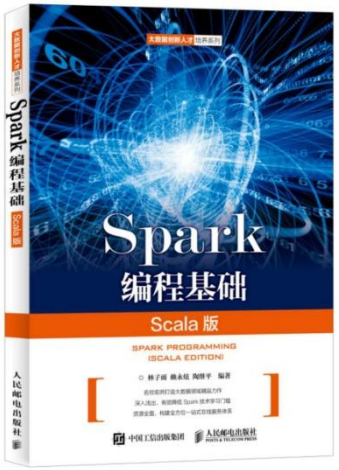
**E-mail: ziyulin@xmu.edu.cn**

主页：**http://www.cs.xmu.edu.cn/linziyu**



# 课程教材

本套讲义PPT属于以下教材的配套材料




**《Spark编程基础（Scala版）》**

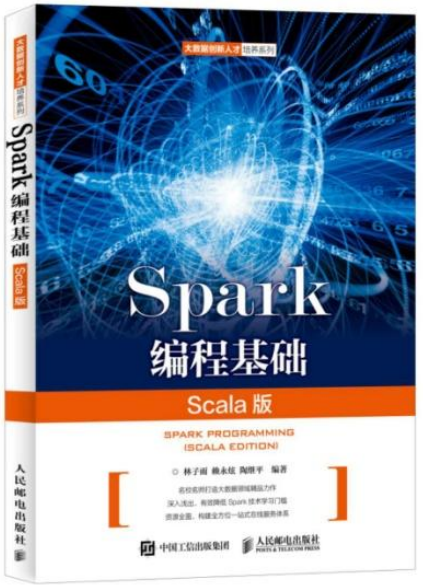
厦门大学 林子雨，赖永炫，陶继平 编著

披荆斩棘，在大数据丛林中开辟学习捷径
填沟削坎，为快速学习Spark技术铺平道路
深入浅出，有效降低Spark技术学习门槛
资源全面，构建全方位一站式在线服务体系

**人民邮电出版社出版发行，ISBN:978-7-115-48816-9**
**教材官网：http://dblab.xmu.edu.cn/post/spark/**

本书以Scala作为开发Spark应用程序的编程语言，系统介绍了Spark编程的基础知识。全书共8章，内容包括大数据技术概述、Scala语言基础、Spark的设计与运行原理、Spark环境搭建和使用方法、RDD编程、Spark SQL、Spark Streaming、Spark MLlib等。本书每个章节都安排了入门级的编程实践操作，以便读者更好地学习和掌握Spark编程方法。本书官网免费提供了全套的在线教学资源，包括讲义PPT、习题、源代码、软件、数据集、授课视频、上机实验指南等。

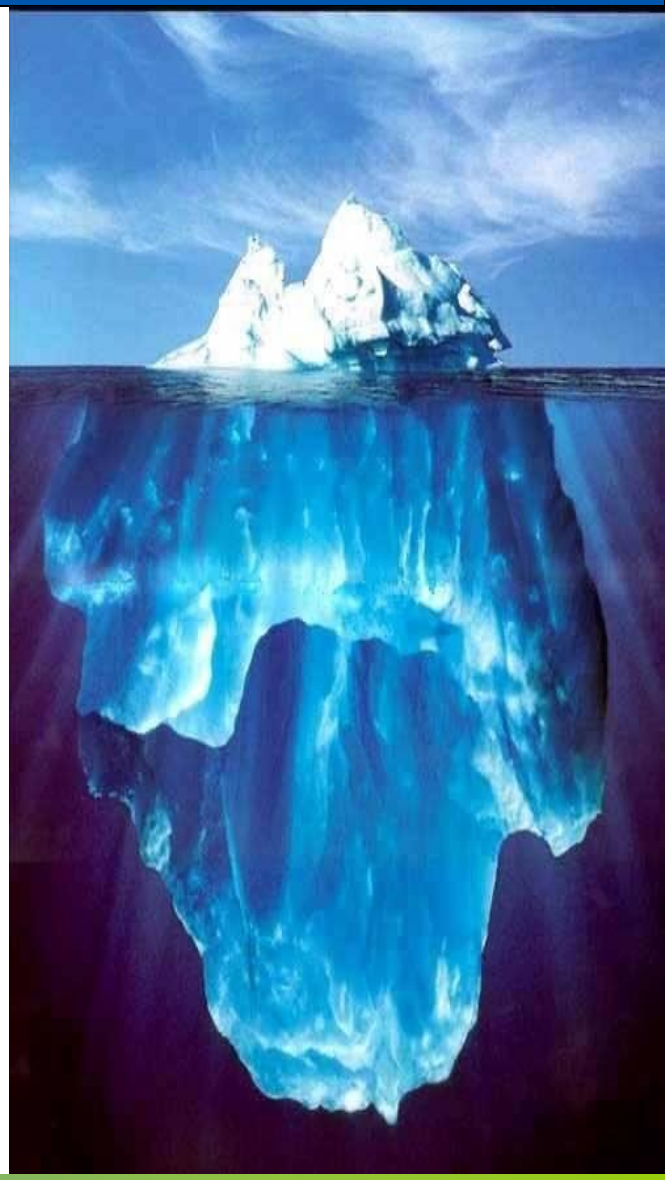
# 课程配套授课视频


**课程在线视频地址：http://dblab.xmu.edu.cn/post/10482/**

# 提纲





高校大数据课程
公 共 服 务 平 台

百度搜索厦门大学数据库实验室网站访问平台

# 4.1 安装Spark

Spark的安装详细过程，请参考厦门大学数据库实验室建设的高校大数据课程公共服务平台上的技术博客：

**《Spark2.1.0入门：Spark的安装和使用》**

博客地址：http://dblab.xmu.edu.cn/blog/1307-2/

4.1.1 基础环境
4.1.2 下载安装文件
4.1.3 配置相关文件
4.1.4 Spark和Hadoop的交互

高校大数据课程
公 共 服 务 平 台

平台每年访问量超过100万次

# 4.1.1 基础环境

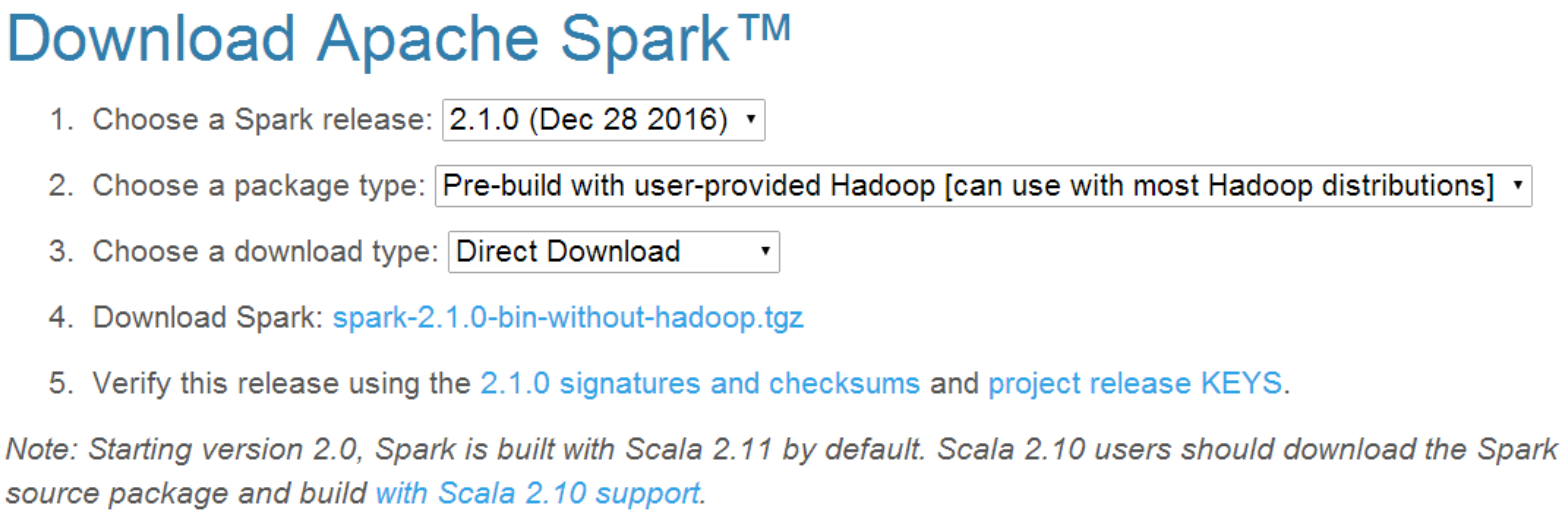
•安装Spark之前需要安装Linux系统、Java环境和Hadoop环境
•如果没有安装Hadoop，请访问厦门大学数据库实验室建设的高校大数据课程公共服务平台，找到“Hadoop安装教程_单机/伪分布式配置_Hadoop2.6.0/Ubuntu14.04”（适用于Hadoop2.7.1/Ubuntu16.04）,依照教程学习安装即可
•注意，在这个Hadoop安装教程中，就包含了Java的安装，所以，按照这个教程，就可以完成JDK和Hadoop这二者的安装

Hadoop安装教程地址：http://dblab.xmu.edu.cn/blog/install-hadoop/

# 4.1.2 下载安装文件

•Spark安装包下载地址： http://spark.apache.org

进入下载页面后，点击主页右侧的“Download Spark”按钮进入下载页面，下载页面中提供了几个下载选项，主要是Spark release及Package type的选择，如下图所示。第1项Spark release一般默认选择最新的发行版本，截至2018年4月份的最新版本为2.3.0（本教程采用2.1.0）。第2项package type则选择“Pre-build with user-provided Hadoop [can use with most Hadoop distributions]”，可适用于多数Hadoop版本。选择好之后，再点击第4项给出的链接就可以下载Spark了。



# 4.1.2 下载安装文件

•解压安装包spark-2.1.0-bin-without-hadoop.tgz至路径 /usr/local：

```
$ sudo tar -zxf ~/下载/spark-2.1.0-bin-without-hadoop.tgz -C /usr/local/
$ cd /usr/local
$ sudo mv ./spark-2.1.0-bin-without-hadoop/ ./spark   # 更改文件夹名
$ sudo chown -R hadoop ./spark                        # 此处的 hadoop 为系统用户名
```

# 4.1.3 配置相关文件

•配置Spark 的classpath

```
$ cd /usr/local/spark
$ cp ./conf/spark-env.sh.template ./conf/spark-env.sh   #拷贝配置文件
```

•编辑该配置文件，在文件最后面加上如下一行内容：

```
export SPARK_DIST_CLASSPATH=$(/usr/local/hadoop/bin/hadoop classpath)
```

•保存配置文件后，就可以启动、运行Spark了
•若需要使用HDFS中的文件，则在使用Spark前需要启动Hadoop

# 4.1.4 Spark和Hadoop的交互

Spark部署模式包括：

- Local模式：单机模式
- Standalone模式：使用Spark自带的简单集群管理器
- YARN模式：使用YARN作为集群管理器
- Mesos模式：使用Mesos作为集群管理器

- 经过上面的步骤以后，就在单台机器上按照“Hadoop（伪分布式）+Spark（Local模式）”这种方式完成了Hadoop和Spark组合环境的搭建。
- Hadoop和Spark可以相互协作，由Hadoop的HDFS、HBase等组件负责数据的存储和管理，由Spark负责数据的计算。

# 4.2 在spark-shell中运行代码

- Spark Shell 提供了简单的方式来学习Spark API
- Spark Shell可以以实时、交互的方式来分析数据
- Spark Shell支持Scala和Python

Spark Shell本身就是一个Driver，里面已经包含了main方法

# 4.2 在spark-shell中运行代码

spark-shell命令及其常用的参数如下：

```
./bin/spark-shell --master <master-url>
```

Spark的运行模式取决于传递给SparkContext的Master URL的值。Master URL可以是以下任一种形式：

* local 使用一个Worker线程本地化运行SPARK(完全不并行)
* local[*] 使用逻辑CPU个数数量的线程来本地化运行Spark
* local[K] 使用K个Worker线程本地化运行Spark（理想情况下，K应该根据运行机器的CPU核数设定）
* spark://HOST:PORT 连接到指定的Spark standalone master。默认端口是7077
* yarn-client 以客户端模式连接YARN集群。集群的位置可以在HADOOP_CONF_DIR 环境变量中找到
* yarn-cluster 以集群模式连接YARN集群。集群的位置可以在HADOOP_CONF_DIR 环境变量中找到
* mesos://HOST:PORT 连接到指定的Mesos集群。默认接口是5050

# 4.2 在spark-shell中运行代码

在Spark中采用本地模式启动Spark Shell的命令主要包含以下参数：
--master：这个参数表示当前的Spark Shell要连接到哪个master，如果是local[*]，就是使用本地模式启动spark-shell，其中，中括号内的星号表示需要使用几个CPU核心(core)，也就是启动几个线程模拟Spark集群
--jars： 这个参数用于把相关的JAR包添加到CLASSPATH中；如果有多个jar包，可以使用逗号分隔符连接它们

# 4.2 在spark-shell中运行代码

比如，要采用本地模式，在4个CPU核心上运行spark-shell：

```
$ cd /usr/local/spark
$ ./bin/spark-shell --master local[4]
```

或者，可以在CLASSPATH中添加code.jar，命令如下：

```
$ cd /usr/local/spark
$ ./bin/spark-shell --master local[4] --jars code.jar
```

可以执行“spark-shell --help”命令，获取完整的选项列表，具体如下：

```
$ cd /usr/local/spark
$ ./bin/spark-shell --help
```

# 4.2 在spark-shell中运行代码

执行如下命令启动Spark Shell（默认是local模式）：

```
$ ./bin/spark-shell
```

启动Spark Shell成功后在输出信息的末尾可以看到“Scala >”的命令提示符

```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.1.0
      /_/

Using Scala version 2.11.8 (OpenJDK 64-Bit Server VM, Java 1.8.0_121)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

# 4.2 在spark-shell中运行代码

可以在里面输入scala代码进行调试：

```
scala> 8*2+5
res0: Int = 21
```

可以使用命令“:quit”退出Spark Shell：

```
scala>:quit
```

或者，也可以直接使用“Ctrl+D”组合键，退出Spark Shell

# 4.3 开发Spark独立应用程序

使用 Scala 编写的程序需要使用 sbt或Maven 进行编译打包

**4.3.1 安装编译打包工具**
**4.3.2 编写Spark应用程序代码**
**4.3.3 编译打包**
**4.3.4 通过spark-submit运行程序**
**4.3.5 使用开发工具编写Spark应用程序**

# 4.3.1 安装编译打包工具

**1.安装sbt**

sbt是一款Spark用来对scala编写程序进行打包的工具，Spark 中没有自带 sbt，需要下载安装

```
sudo mkdir /usr/local/sbt
sudo chown -R hadoop /usr/local/sbt       # 此处的 hadoop 为你的用户名
cd /usr/local/sbt
```

下载sbt安装包以后，执行如下命令拷贝至 /usr/local/sbt 中：

```
$ cp ~/下载/sbt-launch.jar .
```

接着在 /usr/local/sbt 中创建 sbt 脚本（vim ./sbt），添加如下内容：

```
#!/bin/bash SBT_OPTS="-Xms512M -Xmx1536M -Xss1M -
XX:+CMSClassUnloadingEnabled -XX:MaxPermSize=256M" java
$SBT_OPTS -jar `dirname $0`/sbt-launch.jar "$@"
```

# 4.3.1 安装编译打包工具

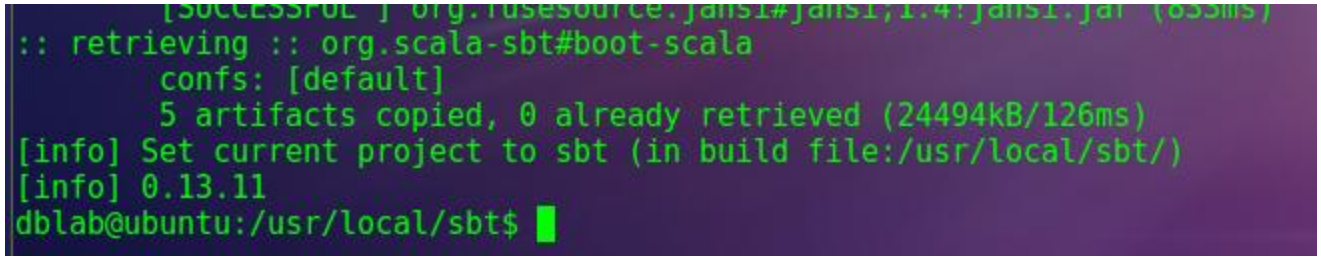
保存后，为 ./sbt 脚本增加可执行权限：

```
$ chmod u+x ./sbt
```

最后运行如下命令，检验 sbt 是否可用（需要几分钟时间）：

```
$ ./sbt sbt-version
```

只要能得到如下图的版本信息就没问题：



# 4.3.1 安装编译打包工具

**2. 安装Maven**

下载到Maven安装文件以后，保存到“~/下载”目录下。然后，可以选择安装在“/usr/local/maven”目录中，命令如下：

```
$ sudo  unzip  ~/下载/apache-maven-3.3.9-bin.zip  -d  /usr/local
$ cd  /usr/local
$ sudo  mv  ./apache-maven-3.3.9  ./maven
$ sudo  chown  -R  hadoop  ./maven
```

# 4.3.2 编写Spark应用程序代码

在终端中执行如下命令创建一个文件夹 sparkapp 作为应用程序根目录：

```
$ cd ~               # 进入用户主文件夹
$ mkdir ./sparkapp          # 创建应用程序根目录
$ mkdir -p ./sparkapp/src/main/scala      # 创建所需的文件夹结构
```

在 ./sparkapp/src/main/scala 下建立一个名为 SimpleApp.scala 的文件，添加代码如下

```
/* SimpleApp.scala */
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
import org.apache.spark.SparkConf

object SimpleApp {
    def main(args: Array[String]) {
        val logFile = "file:///usr/local/spark/README.md" // Should be some file on y
our system
        val conf = new SparkConf().setAppName("Simple Application")
        val sc = new SparkContext(conf)
        val logData = sc.textFile(logFile, 2).cache()
        val numAs = logData.filter(line => line.contains("a")).count()
        val numBs = logData.filter(line => line.contains("b")).count()
        println("Lines with a: %s, Lines with b: %s".format(numAs, numBs))
    }
}
```

# 4.3.3 编译打包

**1. 使用sbt对Scala 程序进行编译打包**

请在./sparkapp 中新建文件 simple.sbt（vim ./sparkapp/simple.sbt），添加内容如下，声明该独立应用程序的信息以及与 Spark 的依赖关系：

```
name := "Simple Project"
version := "1.0"
scalaVersion := "2.11.8"
libraryDependencies += "org.apache.spark" %% "spark-core" % "2.1.0"
```

Spark和Scala的版本信息可以在启动信息中找到

# 4.3.3 编译打包

**1. 使用sbt对Scala 程序进行编译打包**

关于百分号的说明

•中间两个百分号的依赖是不指定版本的，版本由Scala的版本确定

```
libraryDependencies ++= Seq(
  "org.json4s" %% "json4s-native" % "3.2.10",
  "org.json4s" %% "json4s-jackson" % "3.2.10"
)
```

•中间有一个百分号的依赖是指定版本的

```
libraryDependencies += "org.apache.kafka" % "kafka_2.11" % "0.10.0.0"
```

# 4.3.3 编译打包

为保证 sbt 能正常运行，先执行如下命令检查整个应用程序的文件结构：

```
$ cd ~/sparkapp
$ find .
```

文件结构应如下图所示：

# 4.3.3 编译打包

接着，我们就可以通过如下代码将整个应用程序打包成 JAR（首次运行同样需要下载依赖包）：

```
$     /usr/local/sbt/sbt package
```

打包成功的话，会输出类似如下信息：

```
hadoop@dblab:~/sparkapp$ /usr/local/sbt/sbt package
OpenJDK 64-Bit Server VM warning: ignoring option MaxPermSize=256M;
support was removed in 8.0
[info] Set current project to Simple Project (in build
file:/home/hadoop/sparkapp/)
[success] Total time: 2 s, completed 2017-2-19 15:45:29
```

生成的 jar 包的位置为 ~/sparkapp/target/scala-2.11/simple-project_2.11-1.0.jar

# 4.3.3 编译打包

**2. 使用Maven对Scala 程序进行编译打包**

为了和sbt打包编译的内容进行区分，这里再为Maven创建一个代码目录“~/sparkapp2”，并在“~/sparkapp2/src/main/scala”下建立一个名为SimpleApp.scala的Scala代码文件，放入和前面一样的代码

# 4.3.3 编译打包

在“~/sparkapp2”目录中新建文件pom.xml

```
<project>
    <groupId>cn.edu.xmu</groupId>
    <artifactId>simple-project</artifactId>
    <modelVersion>4.0.0</modelVersion>
    <name>Simple Project</name>
    <packaging>jar</packaging>
    <version>1.0</version>
    <repositories>
        <repository>
            <id>jboss</id>
            <name>JBoss Repository</name>
            <url>http://repository.jboss.com/maven2/</url>
        </repository>
    </repositories>
    <dependencies>
        <dependency> <!-- Spark dependency -->
            <groupId>org.apache.spark</groupId>
            <artifactId>spark-core_2.11</artifactId>
            <version>2.1.0</version>
        </dependency>
    </dependencies>
  <build>
    <sourceDirectory>src/main/scala</sourceDirectory>
    <plugins>
      <plugin>
        <groupId>org.scala-tools</groupId>
        <artifactId>maven-scala-plugin</artifactId>
        <executions>
          <execution>
            <goals>
              <goal>compile</goal>
            </goals>
          </execution>
        </executions>
        <configuration>
          <scalaVersion>2.11.8</scalaVersion>
          <args>
            <arg>-target:jvm-1.5</arg>
          </args>
        </configuration>
          </plugin>
          </plugins>
</build>
</project>
```

# 4.3.3 编译打包

在“~/sparkapp2”目录中新建文件pom.xml

```
<project>
    <groupId>cn.edu.xmu</groupId>
    <artifactId>simple-project</artifactId>
    <modelVersion>4.0.0</modelVersion>
    <name>Simple Project</name>
    <packaging>jar</packaging>
    <version>1.0</version>
    <repositories>
        <repository>
            <id>jboss</id>
            <name>JBoss Repository</name>
            <url>http://repository.jboss.com/maven2/</url>
        </repository>
    </repositories>
    <dependencies>
        <dependency> <!-- Spark dependency -->
            <groupId>org.apache.spark</groupId>
            <artifactId>spark-core_2.11</artifactId>
            <version>2.1.0</version>
        </dependency>
    </dependencies>
```

# 4.3.3 编译打包

在“~/sparkapp2”目录中新建文件pom.xml

```
<build>
    <sourceDirectory>src/main/scala</sourceDirectory>
    <plugins>
      <plugin>
        <groupId>org.scala-tools</groupId>
        <artifactId>maven-scala-plugin</artifactId>
        <executions>
          <execution>
            <goals>
              <goal>compile</goal>
            </goals>
          </execution>
        </executions>
        <configuration>
          <scalaVersion>2.11.8</scalaVersion>
          <args>
            <arg>-target:jvm-1.5</arg>
          </args>
        </configuration>
      </plugin>
    </plugins>
</build>
</project>
```

# 4.3.3 编译打包

可以通过如下代码将整个应用程序打包成JAR包（注意：计算机需要保持连接网络的状态，而且首次运行打包命令时，Maven会自动下载依赖包，需要消耗几分钟的时间）：

```
$ cd  ~/sparkapp2     #一定把这个目录设置为当前目录
$ /usr/local/maven/bin/mvn  package
```

如果屏幕返回如下信息，则说明生成JAR包成功：

```
[INFO] Building jar: /home/hadoop/sparkapp2/target/simple-project-1.0.jar
[INFO]------------------------------------------
[INFO] BUILD SUCCESS
[INFO]------------------------------------------
[INFO] Total time: 4.665 s
[INFO] Finished at: 2017-01-31T 15:57:09+08:00
[INFO] Final Memory: 30M/72M
[INFO]------------------------------------------
```

生成的应用程序JAR包的位置为“/home/hadoop/sparkapp2/target/simple-project-1.0.jar”。

# 4.3.4 通过spark-submit运行程序

可以通过spark-submit提交应用程序，该命令的格式如下：

```
./bin/spark-submit
  --class <main-class>  //需要运行的程序的主类，应用程序的入口点
  --master <master-url>  //Master URL，下面会有具体解释
  --deploy-mode <deploy-mode>   //部署模式
  ... # other options  //其他参数
  <application-jar>  //应用程序JAR包
  [application-arguments] //传递给主类的主方法的参数
```

# 4.3.4 通过spark-submit运行程序

Spark的运行模式取决于传递给SparkContext的Master URL的值。
Master URL可以是以下任一种形式：
* local 使用一个Worker线程本地化运行SPARK(完全不并行)
* local[*] 使用逻辑CPU个数数量的线程来本地化运行Spark
* local[K] 使用K个Worker线程本地化运行Spark（理想情况下，K应该根据运行机器的CPU核数设定）
* spark://HOST:PORT 连接到指定的Spark standalone master。默认端口是7077.
* yarn-client 以客户端模式连接YARN集群。集群的位置可以在HADOOP_CONF_DIR 环境变量中找到。
* yarn-cluster 以集群模式连接YARN集群。集群的位置可以在HADOOP_CONF_DIR 环境变量中找到。
* mesos://HOST:PORT 连接到指定的Mesos集群。默认接口是5050。

# 4.3.4 通过spark-submit运行程序

对于之前使用sbt工具编译打包得到的 jar 包，就可以通过 spark-submit 提交到 Spark 中运行了，命令如下：

```
$ /usr/local/spark/bin/spark-submit --class "SimpleApp"
~/sparkapp/target/scala-2.11/simple-project_2.11-1.0.jar
#上面命令执行后会输出太多信息，可以不使用上面命令，而使用下面命令查看想要的结果
$ /usr/local/spark/bin/spark-submit --class "SimpleApp"
~/sparkapp/target/scala-2.11/simple-project_2.11-1.0.jar 2>&1 | grep "Lines with a:"
```

最终得到的结果如下：

```
Lines with a: 62, Lines with b: 30
```

对于之前使用Maven工具编译打包得到的 jar 包，也可以通过 spark-submit 提交到 Spark 中运行

# 4.3.5使用开发工具编写Spark应用程序

**1.使用Eclipse编写Spark应用程序**

第1种方式：
在现有的Eclipse基础之上，安装maven插件、sbt插件和scala插件

第2种方式：安装Scala IDE for Eclipse


**详细编程方法请参考厦大数据库实验室“高校大数据课程公共服务平台”上的博客文章：**
**（1）使用Eclipse编写Spark应用程序（Scala+Maven**
**http://dblab.xmu.edu.cn/blog/1406/**
**（2）使用Eclipse编写Spark应用程序（Scala+SBT）**
**http://dblab.xmu.edu.cn/blog/1490/**

高校大数据课程
公 共 服 务 平 台

# 4.3.5使用开发工具编写Spark应用程序

**2.使用IntelliJ IDEA编写Spark应用程序**


**idealU-2016.3.4.tar.gz（企业版）和idealC-2016.3.4.tar.gz（社区版）**

**IntelliJ IDEA已经集成了sbt和Maven插件**
**需要额外安装Scala插件**

**详细编程方法请参考厦大数据库实验室“高校大数据课程公共服务平台”上的博客文章：**
**（1）利用开发工具IntelliJ IDEA编写Spark应用程序（Scala+Maven）**
**http://dblab.xmu.edu.cn/blog/1327/**
**（2）使用Intellij Idea编写Spark应用程序（Scala+SBT）**
**http://dblab.xmu.edu.cn/blog/1492-2/**

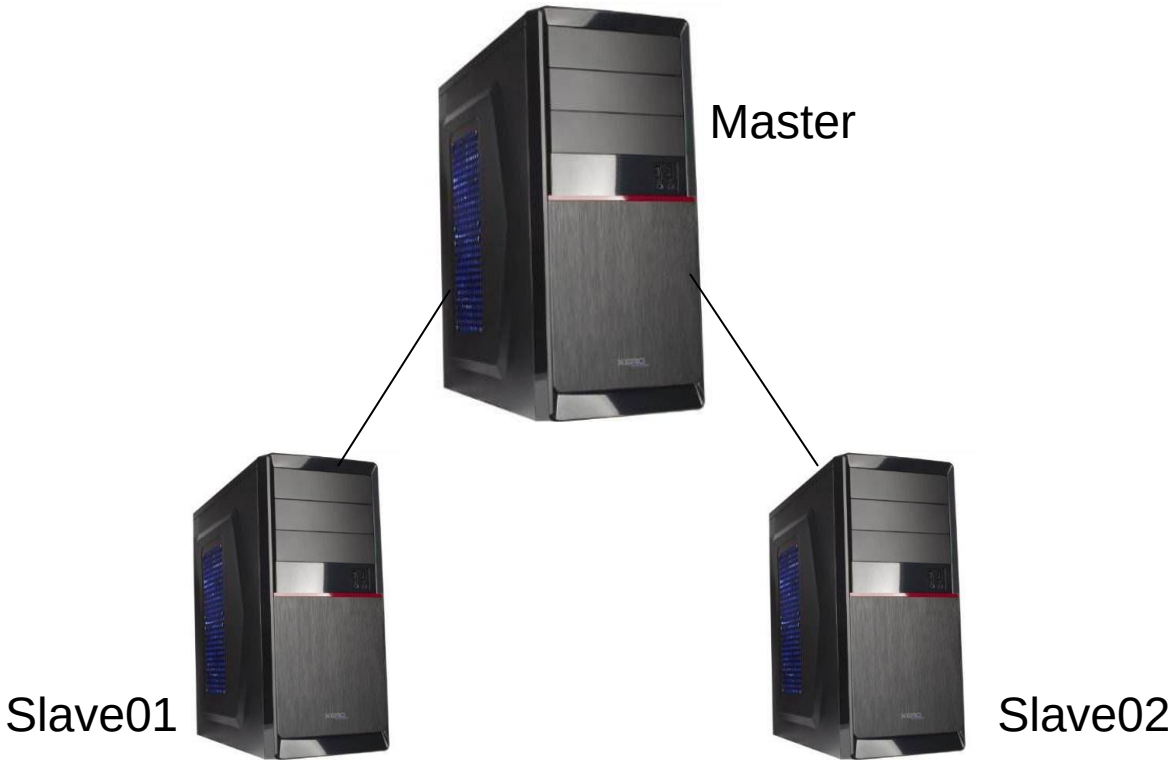
高校大数据课程
公 共 服 务 平 台

# 4.4 Spark集群环境搭建

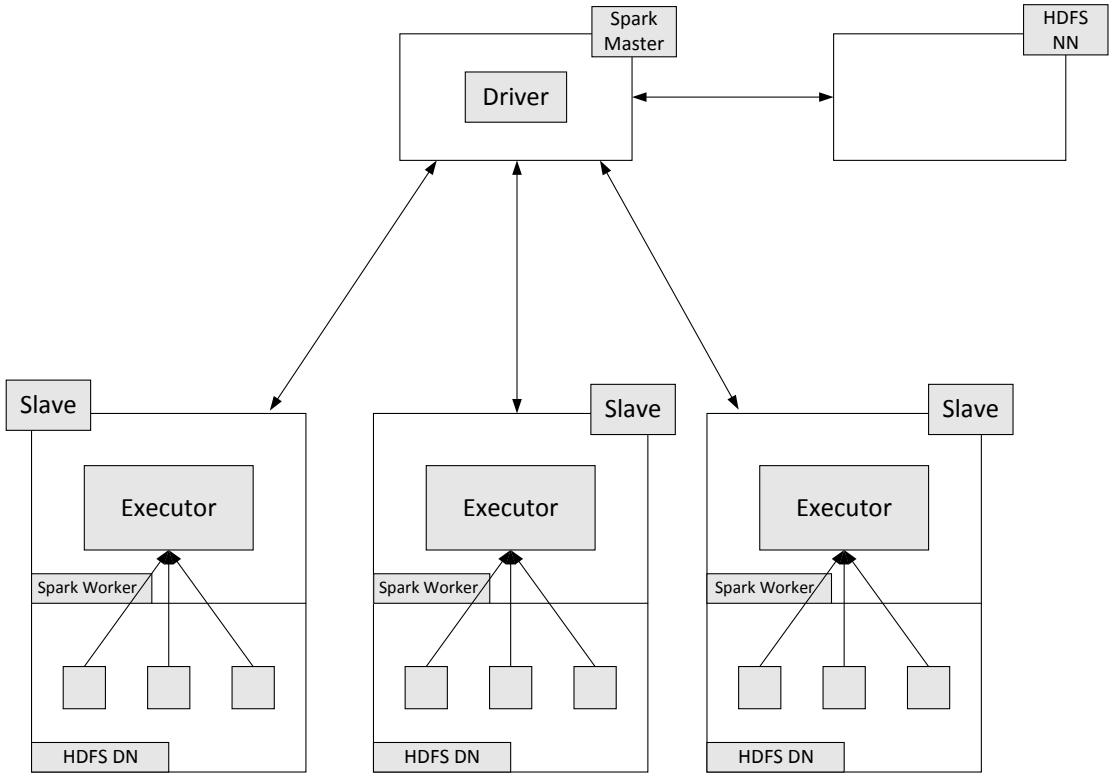
4.4.1 集群概况
4.4.2 准备工作：搭建Hadoop集群环境
4.4.3 安装Spark
4.4.4 配置环境变量
4.4.5 Spark配置
4.4.6 启动Spark集群
4.4.7 关闭Spark集群

# 4.4.1 集群概况

- 采用3台机器（节点）作为实例来演示如何搭建Spark集群
- 其中1台机器（节点）作为Master节点
- 另外两台机器（节点）作为Slave节点（即作为Worker节点），主机名分别为Slave01和Slave02



# 4.4.2 准备工作：搭建Hadoop集群环境




Spark+HDFS运行架构

请参考厦门大学数据库实验室建设的“高校大数据课程公共服务平台”里面的技术博客：《**Hadoop 2.7分布式集群环境搭建**》
文章地址：http://dblab.xmu.edu.cn/blog/1177-2/

# 4.4.3 安装Spark

在Master节点上，访问Spark官网下载Spark安装包

## Download Apache Spark™

1. Choose a Spark release: 2.1.0 (Dec 28 2016)
2. Choose a package type: Pre-build with user-provided Hadoop [can use with most Hadoop distributions]
3. Choose a download type: Direct Download
4. Download Spark: spark-2.1.0-bin-without-hadoop.tgz
5. Verify this release using the 2.1.0 signatures and checksums and project release KEYS.

*Note: Starting version 2.0, Spark is built with Scala 2.11 by default. Scala 2.10 users should download the Spark source package and build with Scala 2.10 support.*

```
sudo tar -zxf ~/下载/spark-2.1.0-bin-without-hadoop.tgz -C /usr/local/
cd /usr/local
sudo mv ./spark-2.1.0-bin-without-hadoop/ ./spark
sudo chown -R hadoop ./spark
```

# 4.4.4 配置环境变量

在Master节点主机的终端中执行如下命令：

```
$ vim ~/.bashrc
```

在.bashrc添加如下配置：

```
export SPARK_HOME=/usr/local/spark
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
```

运行source命令使得配置立即生效：

```
$ source ~/.bashrc
```

# 4.4.5 Spark配置

（1）配置slaves文件

将 slaves.template 拷贝到 slaves

```
$ cd /usr/local/spark/
$ cp ./conf/slaves.template ./conf/slaves
```

slaves文件设置Worker节点。编辑slaves内容,把默认内容localhost替换成如下内容：

```
Slave01
slave02
```

# 4.4.5 Spark配置

（2）配置spark-env.sh文件

将 spark-env.sh.template 拷贝到 spark-env.sh

```
$ cp ./conf/spark-env.sh.template ./conf/spark-env.sh
```

编辑spark-env.sh,添加如下内容：

```
export SPARK_DIST_CLASSPATH=$(/usr/local/hadoop/bin/hadoop classpath)
export HADOOP_CONF_DIR=/usr/local/hadoop/etc/hadoop
export SPARK_MASTER_IP=192.168.1.104
```

# 4.4.5 Spark配置

配置好后，将Master主机上的/usr/local/spark文件夹复制到各个节点上
在Master主机上执行如下命令：

```
cd /usr/local/
tar -zcf ~/spark.master.tar.gz ./spark
cd ~
scp ./spark.master.tar.gz slave01:/home/hadoop
scp ./spark.master.tar.gz slave02:/home/hadoop
```

在slave01,slave02节点上分别执行下面同样的操作：

```
sudo rm -rf /usr/local/spark/
sudo tar -zxf ~/spark.master.tar.gz -C /usr/local
sudo chown -R hadoop /usr/local/spark
```

# 4.4.6 启动Spark集群

（1）首先启动Hadoop集群。在Master节点主机上运行如下命令：

```
$ cd /usr/local/hadoop/
$ sbin/start-all.sh
```

（2）启动Master节点

在Master节点主机上运行如下命令：

```
$ cd /usr/local/spark/
$ sbin/start-master.sh
```

（3）启动所有Slave节点

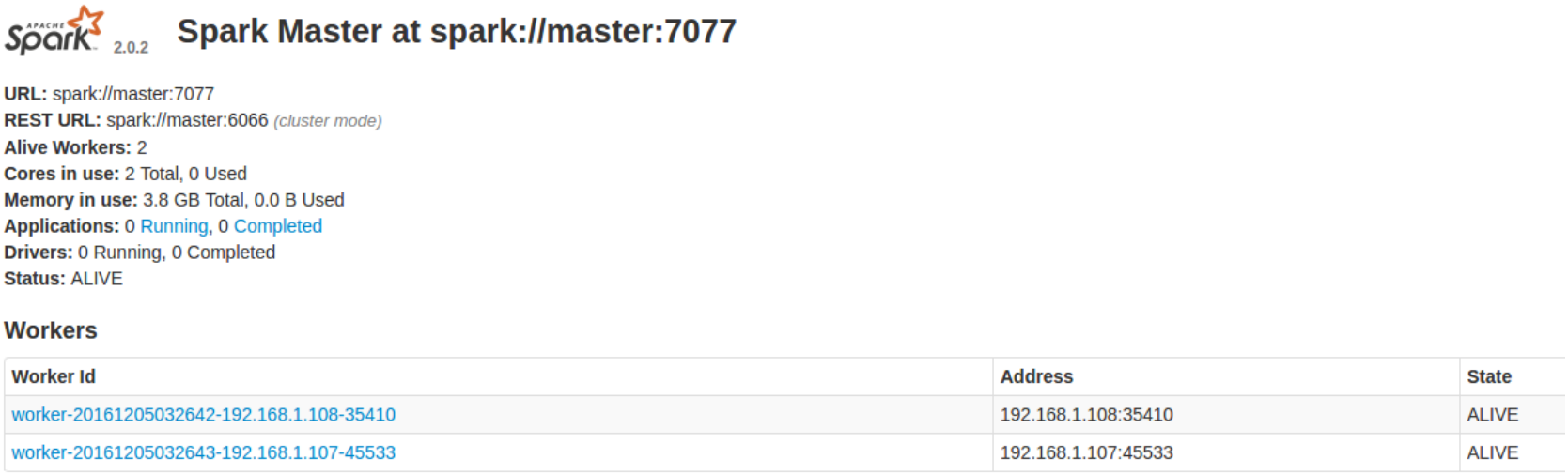
在Master节点主机上运行如下命令：

```
$ sbin/start-slaves.sh
```

# 4.4.6 启动Spark集群

（4）在浏览器上查看Spark独立集群管理器的集群信息

在Master主机上打开浏览器，访问http://master:8080,如下图：



# 4.4.7 关闭Spark集群

在Master节点上执行下面命令

（1）关闭Master节点

```
$ sbin/stop-master.sh
```

（2）关闭Worker节点

```
$ sbin/stop-slaves.sh
```

（3）关闭Hadoop集群

```
$ cd /usr/local/hadoop/
$ sbin/stop-all.sh
```

# 4.5 在集群上运行Spark应用程序

4.5.1 启动Spark集群
4.5.2 采用独立集群管理器
4.5.3 采用Hadoop YARN管理器

# 4.5.1 启动Spark集群

请登录Linux系统，打开一个终端
启动Hadoop集群

```
$ cd /usr/local/hadoop/
$ sbin/start-all.sh
```

启动Spark的Master节点和所有slaves节点

```
$ cd /usr/local/spark/
$ sbin/start-master.sh
$ sbin/start-slaves.sh
```

# 4.5.2 采用独立集群管理器

**（1）在集群中运行应用程序JAR包**

- 向独立集群管理器提交应用，需要把spark://master:7077作为主节点参数递给spark-submit
- 可以运行Spark安装好以后自带的样例程序SparkPi，它的功能是计算得到pi的值（3.1415926）

```
$ bin/spark-submit  \
> --class org.apache.spark.examples.SparkPi  \
> --master spark://master:7077  \
> examples/jars/spark-examples_2.11-2.0.2.jar 100 2>&1 | grep "Pi is roughly"
```

# 4.5.2 采用独立集群管理器

**（2）在集群中运行spark-shell**

也可以用spark-shell连接到独立集群管理器上

```
$ bin/spark-shell --master spark://master:7077
```

```
scala> val textFile = sc.textFile("hdfs://master:9000/README.md")
textFile: org.apache.spark.rdd.RDD[String] =
hdfs://master:9000/README.md MapPartitionsRDD[1] at textFile at
<console>:24
scala> textFile.count()
res0: Long = 99
scala> textFile.first()
res1: String = # Apache Spark
```

# 4.5.2 采用独立集群管理器

用户在独立集群管理Web界面查看应用的运行情况
http://master:8080/

**Running Applications**

| Application ID | Name | Cores | Memory per Node | Submitted Time | User | State | Duration |
|---|---|---|---|---|---|---|---|

**Completed Applications**

| Application ID | Name | Cores | Memory per Node | Submitted Time | User | State | Duration |
|---|---|---|---|---|---|---|---|
| app-20161206170846-0004 | Spark Pi | 1 | 1024.0 MB | 2016/12/06 17:08:46 | hadoop | FINISHED | 6 s |
| app-20161206164856-0003 | Spark Pi | 1 | 1024.0 MB | 2016/12/06 16:48:56 | hadoop | FINISHED | 8 s |
| app-20161206164713-0002 | Spark Pi | 1 | 1024.0 MB | 2016/12/06 16:47:13 | hadoop | FINISHED | 8 s |
| app-20161206163254-0001 | Spark shell | 1 | 1024.0 MB | 2016/12/06 16:32:54 | hadoop | FINISHED | 12 min |
| app-20161206161343-0000 | Spark shell | 1 | 1024.0 MB | 2016/12/06 16:13:43 | hadoop | FINISHED | 7.0 min |

# 4.5.3 采用Hadoop YARN管理器

**（1）在集群中运行应用程序JAR包**

向Hadoop YARN集群管理器提交应用，需要把yarn-cluster作为主节点参数递给spark-submit。

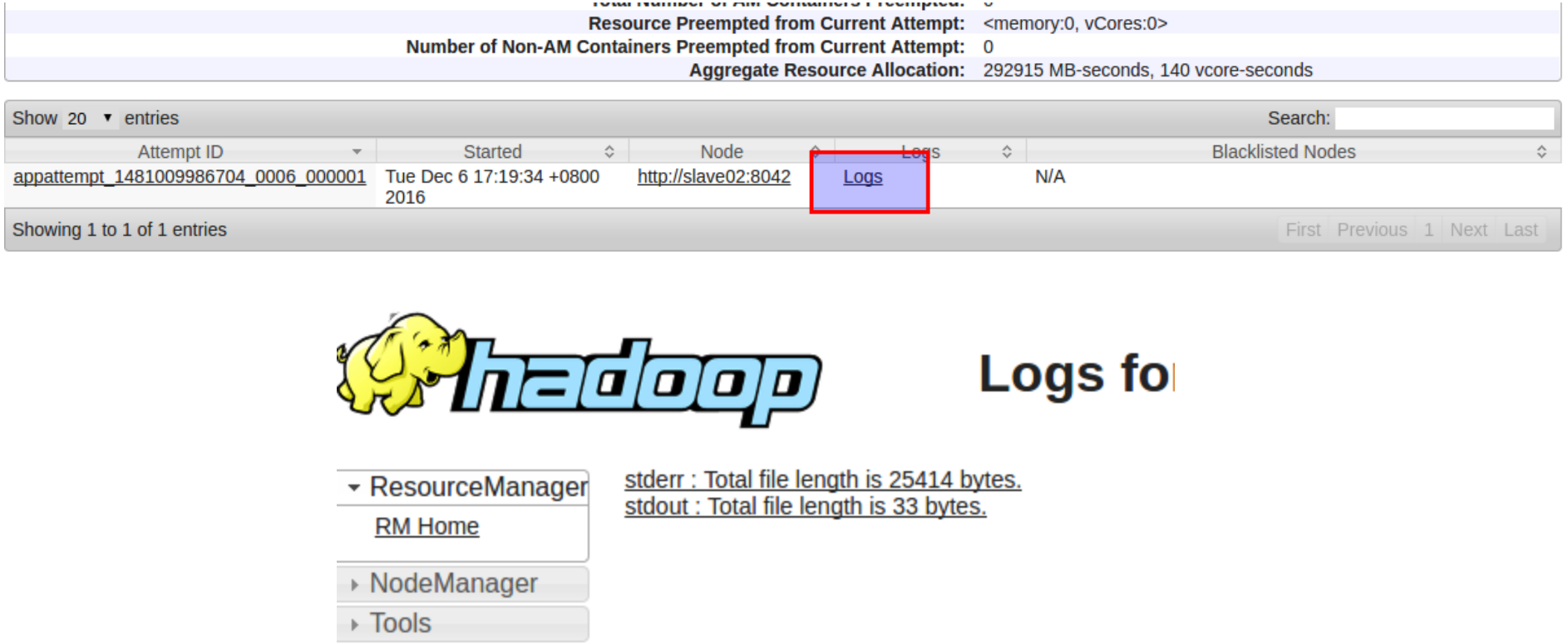
```
$ bin/spark-submit  \
> --class org.apache.spark.examples.SparkPi   \
> --master yarn-cluster   \
> examples/jars/spark-examples_2.11-2.0.2.jar
```

运行后，根据在Shell中得到输出的结果地址查看，如下图：

```
6/12/06 17:20:28 INFO yarn.Client:
         client token: N/A
         diagnostics: N/A
         ApplicationMaster host: 192.168.1.108
         ApplicationMaster RPC port: 0
         queue: default
         start time: 1481015974658
         final status: SUCCEEDED
         tracking URL: http://master:8088/proxy/application_1481009986704_0006/
         user: hadoop
6/12/06 17:20:28 INFO util.ShutdownHookManager: Shutdown hook called
6/12/06 17:20:28 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-7774d207-4a
```

# 4.5.3 采用Hadoop YARN管理器

复制结果地址到浏览器，点击查看Logs，再点击stdout，即可查看结果，如下图：



# 4.5.3 采用Hadoop YARN管理器

**（2）在集群中运行spark-shell**

也可以用spark-shell连接到YARN集群管理器上

```
$ bin/spark-shell --master yarn
```

或者

```
$ bin/spark-shell --master yarn-client
```

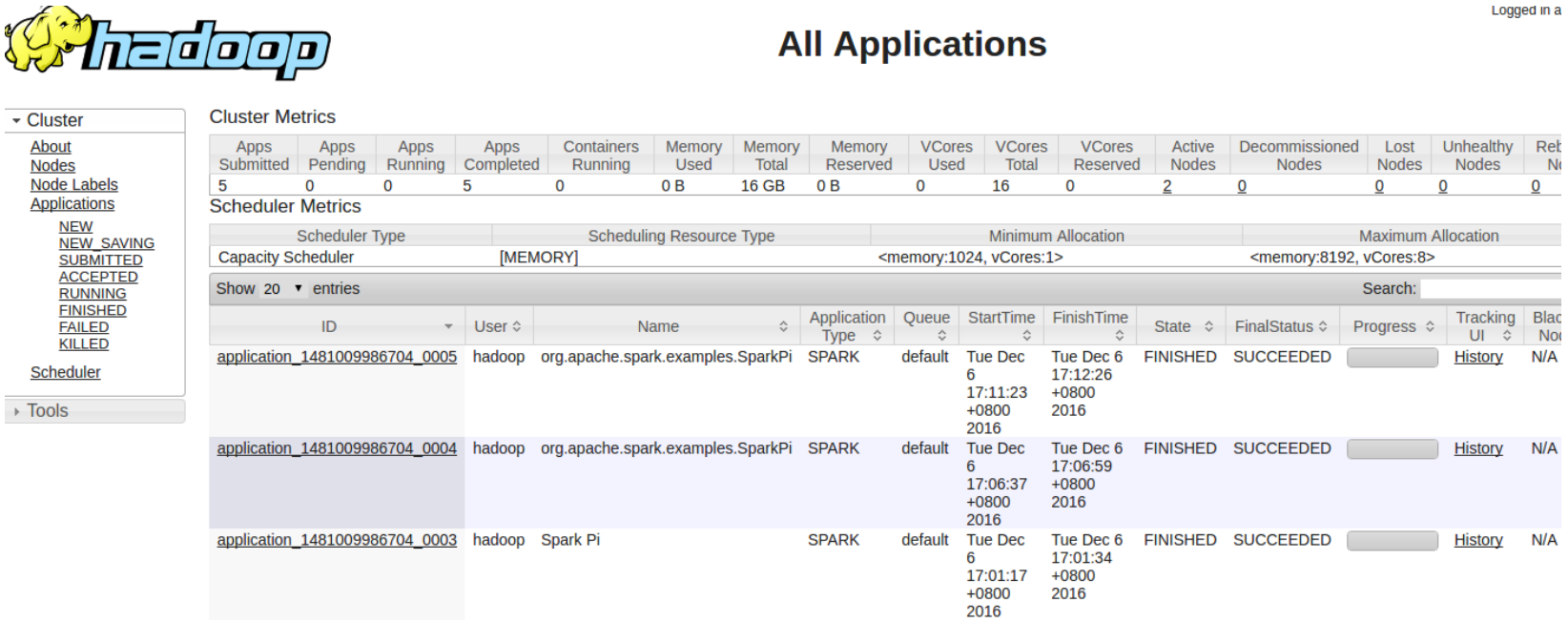
```
scala> val textFile = sc.textFile("hdfs://master:9000/README.md")
textFile: org.apache.spark.rdd.RDD[String] =
hdfs://master:9000/README.md MapPartitionsRDD[3] at textFile at
<console>:24

scala> textFile.count()
res2: Long = 99

scala> textFile.first()
res3: String = # Apache Spark
```

# 4.5.3 采用Hadoop YARN管理器

**（3）查看集群信息**

用户在Hadoop Yarn集群管理Web界面查看所有应用的运行情况

http://master:8088/cluster



# 附录A：主讲教师林子雨简介


**主讲教师：林子雨**

单位：厦门大学计算机科学系
E-mail: ziyulin@xmu.edu.cn
个人网页：http://www.cs.xmu.edu.cn/linziyu
数据库实验室网站：http://dblab.xmu.edu.cn


扫一扫访问个人主页

林子雨，男，1978年出生，博士（毕业于北京大学），现为厦门大学计算机科学系助理教授（讲师），曾任厦门大学信息科学与技术学院院长助理、晋江市发展和改革局副局长。中国计算机学会数据库专业委员会委员，中国计算机学会信息系统专业委员会委员。国内高校首个“数字教师”提出者和建设者，厦门大学数据库实验室负责人，厦门大学云计算与大数据研究中心主要建设者和骨干成员，2013年度和2017年度厦门大学教学类奖教金获得者，荣获2017年福建省精品在线开放课程和2017年厦门大学高等教育成果二等奖。主要研究方向为数据库、数据仓库、数据挖掘、大数据、云计算和物联网，并以第一作者身份在《软件学报》《计算机学报》和《计算机研究与发展》等国家重点期刊以及国际学术会议上发表多篇学术论文。作为项目负责人主持的科研项目包括1项国家自然科学青年基金项目(No.61303004)、1项福建省自然科学青年基金项目(No.2013J05099)和1项中央高校基本科研业务费项目(No.2011121049)，主持的教改课题包括1项2016年福建省教改课题和1项2016年教育部产学协作育人项目，同时，作为课题负责人完成了国家发改委城市信息化重大课题、国家物联网重大应用示范工程区域试点泉州市工作方案、2015泉州市互联网经济调研等课题。中国高校首个“数字教师”提出者和建设者，2009年至今，“数字教师”大平台累计向网络免费发布超过500万字高价值的研究和教学资料，累计网络访问量超过500万次。打造了中国高校大数据教学知名品牌，编著出版了中国高校第一本系统介绍大数据知识的专业教材《大数据技术原理与应用》，并成为京东、当当网等网店畅销书籍；建设了国内高校首个大数据课程公共服务平台，为教师教学和学生学习大数据课程提供全方位、一站式服务，年访问量超过100万次。

# 附录B：大数据学习路线图


大数据学习路线图访问地址：http://dblab.xmu.edu.cn/post/10164/

# 附录C：《大数据技术原理与应用》教材

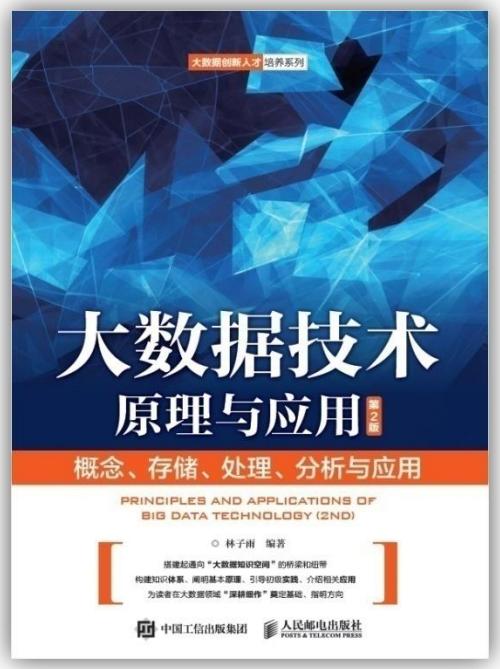
《大数据技术原理与应用——概念、存储、处理、分析与应用（第2版）》，由厦门大学计算机科学系林子雨博士编著，是国内高校第一本系统介绍大数据知识的专业教材。人民邮电出版社　ISBN:978-7-115-44330-4　定价：49.80元

全书共有15章，系统地论述了大数据的基本概念、大数据处理架构Hadoop、分布式文件系统HDFS、分布式数据 库HBase、NoSQL数据库、云数据库、分布式并行编程模型MapReduce、Spark、流计算、图计算、数据可视化以及大数据在互联网、生物医学和物流等各个领域的应用。在Hadoop、HDFS、HBase和MapReduce等重要章节，安排了入门级的实践操作，让读者更好地学习和掌握大数据关键技术。

本书可以作为高等院校计算机专业、信息管理等相关专业的大数据课程教材，也可供相关技术人员参考、学习、培训之用。

欢迎访问《大数据技术原理与应用——概念、存储、处理、分析与应用》教材官方网站：
http://dblab.xmu.edu.cn/post/bigdata

扫一扫访问教材官网



# 附录D：《大数据基础编程、实验和案例教程》

本书是与《大数据技术原理与应用（第2版）》教材配套的唯一指定实验指导书

大数据教材


1+1黄金组合

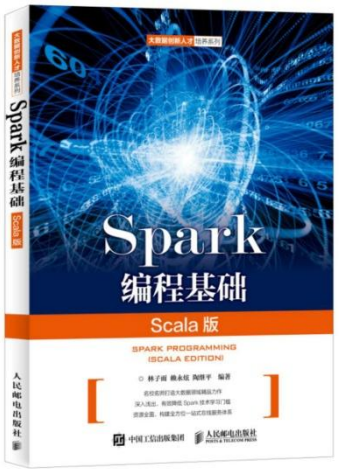
厦门大学林子雨编著

配套实验指导书


清华大学出版社 ISBN:978-7-302-47209-4 定价：59元

- 步步引导，循序渐进，详尽的安装指南为顺利搭建大数据实验环境铺平道路
- 深入浅出，去粗取精，丰富的代码实例帮助快速掌握大数据基础编程方法
- 精心设计，巧妙融合，五套大数据实验题目促进理论与编程知识的消化和吸收
- 结合理论，联系实际，大数据课程综合实验案例精彩呈现大数据分析全流程

# 附录E：《Spark编程基础（Scala版）》




## 《Spark编程基础（Scala版）》

厦门大学 林子雨，赖永炫，陶继平 编著

披荆斩棘，在大数据丛林中开辟学习捷径
填沟削坎，为快速学习Spark技术铺平道路
深入浅出，有效降低Spark技术学习门槛
资源全面，构建全方位一站式在线服务体系

**人民邮电出版社出版发行，ISBN:978-7-115-48816-9**

**教材官网：http://dblab.xmu.edu.cn/post/spark/**

本书以Scala作为开发Spark应用程序的编程语言，系统介绍了Spark编程的基础知识。全书共8章，内容包括大数据技术概述、Scala语言基础、Spark的设计与运行原理、Spark环境搭建和使用方法、RDD编程、Spark SQL、Spark Streaming、Spark MLlib等。本书每个章节都安排了入门级的编程实践操作，以便读者更好地学习和掌握Spark编程方法。本书官网免费提供了全套的在线教学资源，包括讲义PPT、习题、源代码、软件、数据集、授课视频、上机实验指南等。

# 附录F：高校大数据课程公共服务平台

高校大数据课程

公 共 服 务 平 台

http://dblab.xmu.edu.cn/post/bigdata-teaching-platform/

扫一扫访问平台主页

扫一扫观看3分钟FLASH动画宣传片

# 附录G：高校大数据实训课程系列案例教材

为了更好满足高校开设大数据实训课程的教材需求，厦门大学数据库实验室林子雨老师团队联合企业共同开发了《高校大数据实训课程系列案例》，目前已经完成开发的系列案例包括：

《基于协同过滤算法的电影推荐》
《电信用户行为分析》
《实时日志流处理分析》
《微博用户情感分析》
《互联网广告预测分析》
《网站日志处理分析》

部分教材书稿已经完成写作，将于2019年陆续出版发行，教材相关信息，敬请关注网页后续更新！http://dblab.xmu.edu.cn/post/shixunkecheng/

扫一扫访问大数据实训课程系列案例教材主页


Department of Computer Science,  Xiamen University, 2018