



厦门大学研究生课程 《大数据处理技术Spark》

<http://dblab.xmu.edu.cn/post/7659/>

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安装和使用方法

(PPT版本号：2017年春季学期)



扫一扫访问班级主页

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>





提纲

- 4.1 安装Spark
- 4.2 在Spark Shell中运行代码
- 4.3 编写Spark独立应用程序
- 4.4 第一个Spark应用程序：WordCount
- 4.5 使用开发工具编写Spark应用程序
- 4.6 Spark集群环境搭建
- 4.7 在集群上运行Spark应用程序



厦门大学林子雨



子雨大数据之Spark入门教程

披荆斩棘，在大数据丛林中开辟学习捷径

免费在线教程：<http://dblab.xmu.edu.cn/blog/spark/>





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中国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/>



中国高校大数据课程
公 共 服 务 平 台



4.1 安装Spark

- 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中国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”，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 安装Spark

- 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, 如截至2017年3月份的最新版本为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。

Download Apache Spark™

1. Choose a Spark release:
2. Choose a package type:
3. Choose a download type:
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](#).



4.1 安装Spark

- 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
```

- 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 安装Spark

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



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

一个Driver就包括main方法和分布式数据集
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 编写Spark独立应用程序

使用 **Scala** 编写的程序需要使用 **sbt** 进行编译打包，相应的，**Java** 程序使用 **Maven** 编译打包，而 **Python** 程序通过 **spark-submit** 直接提交

4.3.1 安装sbt

4.3.2 编写Scala应用程序

4.3.3 使用 sbt 打包 Scala 程序

4.3.4 通过 spark-submit 运行程序



4.3.1 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:+CMSCClassUnloadingEnabled -XX:MaxPermSize=256M" java
$SBT_OPTS -jar `dirname $0`/sbt-launch.jar "$@"
```




4.3.1 安装sbt

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：

```
[SUCCESSFUL ] org.fusesource.jansi#jansi;1.4:jansi.jar (833ms)
:: retrieving :: org.scala-sbt#boot-scala
  confs: [default]
  5 artifacts copied, 0 already retrieved (24494kB/126ms)
[info] Set current project to sbt (in build file:/usr/local/sbt/)
[info] 0.13.11
dblab@ubuntu:/usr/local/sbt$
```



4.3.2 编写Scala应用程序

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 使用 sbt 打包 Scala 程序

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"
```

```
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> █
```



4.3.3 使用 sbt 打包 Scala 程序

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：



```
hadoop@dblab:~/sparkapp
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
[hadoop@dblab sbt]$ cd ~/sparkapp/
[hadoop@dblab sparkapp]$ find .
.
./src
./src/main
./src/main/scala
./src/main/scala/SimpleApp.scala
./simple.sbt
[hadoop@dblab sparkapp]$
```

廈門大學
数据库实验室



4.3.3 使用 sbt 打包 Scala 程序

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`



4.3.4 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

最后，我们就可以将生成的 jar 包通过 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
```



4.4 第一个Spark应用程序：WordCount

(1) 在spark-shell中执行词频统计

```
scala> val textFile =  
sc.textFile("file:///usr/local/spark/mycode/wordcount/word.txt")  
scala> val wordCount = textFile.flatMap(line => line.split(" ")).map(word =>  
(word, 1)).reduceByKey((a, b) => a + b)  
scala> wordCount.collect()  
scala> wordCount.foreach(println)
```



4.4 第一个Spark应用程序：WordCount

(2) 编写独立应用程序执行词频统计

```
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
import org.apache.spark.SparkConf
object WordCount {
  def main(args: Array[String]) {
    val inputFile = "file:///usr/local/spark/mycode/wordcount/word.txt"
    val conf = new SparkConf().setAppName("WordCount").setMaster("local[2]")
    val sc = new SparkContext(conf)
    val textFile = sc.textFile(inputFile)
    val wordCount = textFile.flatMap(line => line.split(" ")).map(word =>
(word, 1)).reduceByKey((a, b) => a + b)
    wordCount.foreach(println)
  }
}
```

(3) 编译打包运行，得到词频统计结果



4.5 使用开发工具编写Spark应用程序

4.5.1 使用IntelliJ IDEA编写Spark应用程序

4.5.2 使用Eclipse编写Spark应用程序



4.5 使用IntelliJ IDEA编写Spark应用程序

- 安装IntelliJ IDEA
- 在IDEA里安装Scala插件，并配置JDK、Scala SDK
- 配置JDK
- 配置全局Scala SDK
- 创建Maven工程文件
- 前置的一些配置
- 编写代码
- 运行WordCount程序
- 打包WordCount程序



请参考厦门大学数据库实验室建设的“中国高校大数据课程公共服务平台”里面的技术博客：《利用开发工具IntelliJ IDEA编写Spark应用程序》
文章地址：<http://dblab.xmu.edu.cn/blog/1327/>



4.5.2使用Eclipse编写Spark应用程序



请参考厦门大学数据库实验室建设的“中国高校大数据课程公共服务平台”里面的技术博客：《利用开发工具**Eclipse**编写**Spark**应用程序》
文章地址：<http://dblab.xmu.edu.cn/blog/1406/>



4.5.2使用Eclipse编写Spark应用程序

安装必备软件说明

第1种方式：安装Scala IDE for Eclipse



第2种方式：

在现有的Eclipse基础之上，安装maven插件和scala插件



4.6 Spark集群环境搭建

4.6.1 集群概况

4.6.2 准备工作：搭建Hadoop集群环境

4.6.3 安装Spark

4.6.4 配置环境变量

4.6.5 Spark配置

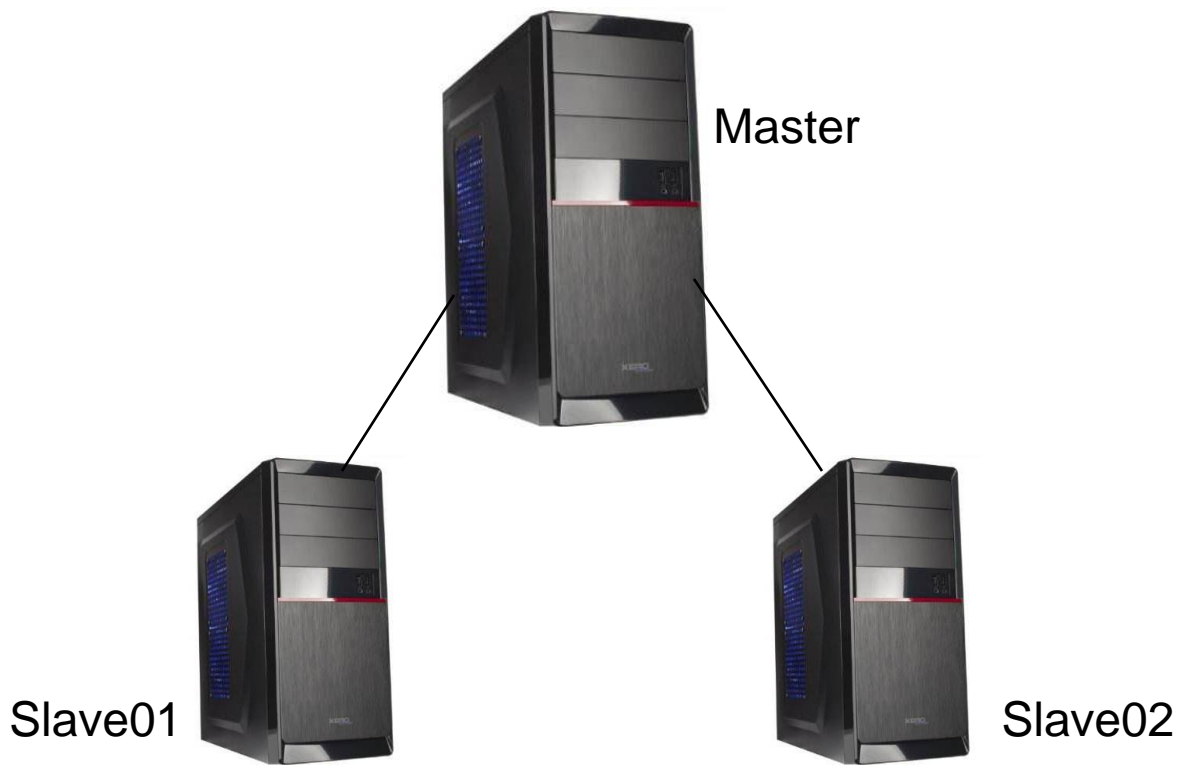
4.6.6 启动Spark集群

4.6.7 关闭Spark集群



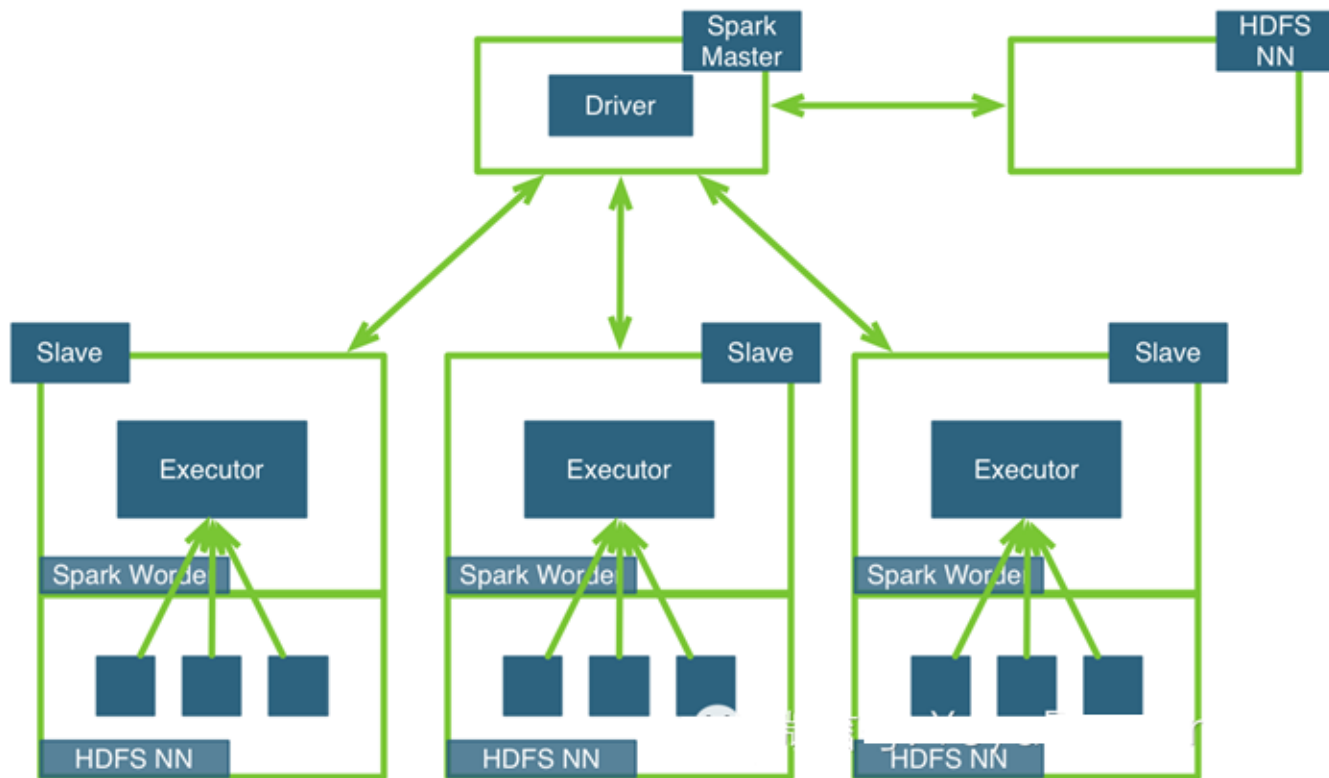
4.6.1 集群概况

- 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





4.6.2 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“中国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/>



4.6.3 安装Spark

访问Spark官网下载Spark安装包

Download Apache Spark™

1. Choose a Spark release:
2. Choose a package type:
3. Choose a download type:
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
```



4.6.4 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
```



4.6.5 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
```



4.6.5 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
```




4.6.5 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
```



4.6.6 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
```



4.6.6 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:

 **Spark Master at spark://master:7077**

URL: spark://master:7077

REST URL: spark://master:6066 (cluster mode)

Alive Workers: 2

Cores in use: 2 Total, 0 Used

Memory in use: 3.8 GB Total, 0.0 B Used

Applications: 0 Running, 0 Completed

Drivers: 0 Running, 0 Completed

Status: ALIVE

Workers

Worker Id	Address	State
worker-20161205032642-192.168.1.108-35410	192.168.1.108:35410	ALIVE
worker-20161205032643-192.168.1.107-45533	192.168.1.107:45533	ALIVE



4.6.7 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
```



4.7 在集群上运行Spark应用程序

4.7.1 启动Spark集群

4.7.2 采用独立集群管理器

4.7.3 采用Hadoop YARN管理器



4.7.1 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
```



4.7.2 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"
```



4.7.2 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
```




4.7.2 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



4.7.3 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：

```
state: FINISHED)
6/12/06 17:20:28 INFO yarn.Client:master : 8080](http://master:8080/), 如下图:
    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
0-45cc-02ad-cc2a2e13e017
```



4.7.3 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：

Total Number of AM Containers Preempted: 0

Resource Preempted from Current Attempt: <memory:0, vCores:0>

Number of Non-AM Containers Preempted from Current Attempt: 0

Aggregate Resource Allocation: 292915 MB-seconds, 140 vcore-seconds

Show 20 entries

Search:

Attempt ID	Started	Node	Logs	Blacklisted Nodes
appattempt_1481009986704_0006_000001	Tue Dec 6 17:19:34 +0800 2016	http://slave02:8042	Logs	N/A

Showing 1 to 1 of 1 entries

First Previous 1 Next Last



Logs for

ResourceManager

RM Home

NodeManager

Tools

stderr : Total file length is 25414 bytes.

stdout : Total file length is 33 bytes.



4.7.3 采用Hadoop YARN管理器

(2) 在集群中运行spark-shell

也可以用spark-shell连接到独立集群管理器上

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

```
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
```



4.7.3 采用Hadoop YARN管理器

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



Logged in as

All Applications

- Cluster
 - About
 - Nodes
 - Node Labels
 - Applications
 - NEW
 - NEW_SAVING
 - SUBMITTED
 - ACCEPTED
 - RUNNING
 - FINISHED
 - FAILED
 - KILLED
 - Scheduler
- Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Ret Nt
5	0	0	5	0	0 B	16 GB	0 B	0	16	0	2	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>	<memory:8192, vCores:8>

Show 20 entries

Search:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI	Blac Not
application_1481009986704_0005	hadoop	org.apache.spark.examples.SparkPi	SPARK	default	Tue Dec 6 17:11:23 +0800 2016	Tue Dec 6 17:12:26 +0800 2016	FINISHED	SUCCEEDED	<div style="width: 100%;"></div>	History	N/A
application_1481009986704_0004	hadoop	org.apache.spark.examples.SparkPi	SPARK	default	Tue Dec 6 17:06:37 +0800 2016	Tue Dec 6 17:06:59 +0800 2016	FINISHED	SUCCEEDED	<div style="width: 100%;"></div>	History	N/A
application_1481009986704_0003	hadoop	Spark Pi	SPARK	default	Tue Dec 6 17:01:17 +0800 2016	Tue Dec 6 17:01:34 +0800 2016	FINISHED	SUCCEEDED	<div style="width: 100%;"></div>	History	N/A



附录：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，荣获“2016中国大数据创新百人”称号。中国高校首个“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年度厦门大学奖教金获得者。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项国家自然科学基金青年基金项目(No.61303004)、1项福建省自然科学基金项目(No.2013J05099)和1项中央高校基本科研业务费项目(No.2011121049)，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100万字高价值的研究和教学资料，累计网络访问量超过100万次。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50万次。具有丰富的政府和企业信息化培训经验，厦门大学管理学院EDP中心、浙江大学管理学院EDP中心、厦门大学继续教育学院、泉州市科技培训中心特邀培训讲师，曾给中国移动通信集团公司、福州马尾区政府、福建龙岩卷烟厂、福建省物联网科学研究院、石狮市物流协会、厦门市物流协会、浙江省中小企业家、四川泸州企业家、江苏沛县企业家等开展信息化培训，累计培训人数达3000人以上。



附录：林子雨编著《Spark入门教程》

厦门大学林子雨编著《Spark入门教程》
教程内容包括Scala语言、Spark简介、安装、运行架构、RDD的设计与运行原理、部署模式、RDD编程、键值对RDD、数据读写、Spark SQL、Spark Streaming、MLlib等



厦门大学林子雨



子雨大数据之Spark入门教程

披荆斩棘，在大数据丛林中开辟学习捷径



免费在线教程：<http://dblab.xmu.edu.cn/blog/spark/>



附录：《大数据技术原理与应用》教材



扫一扫访问教材官网

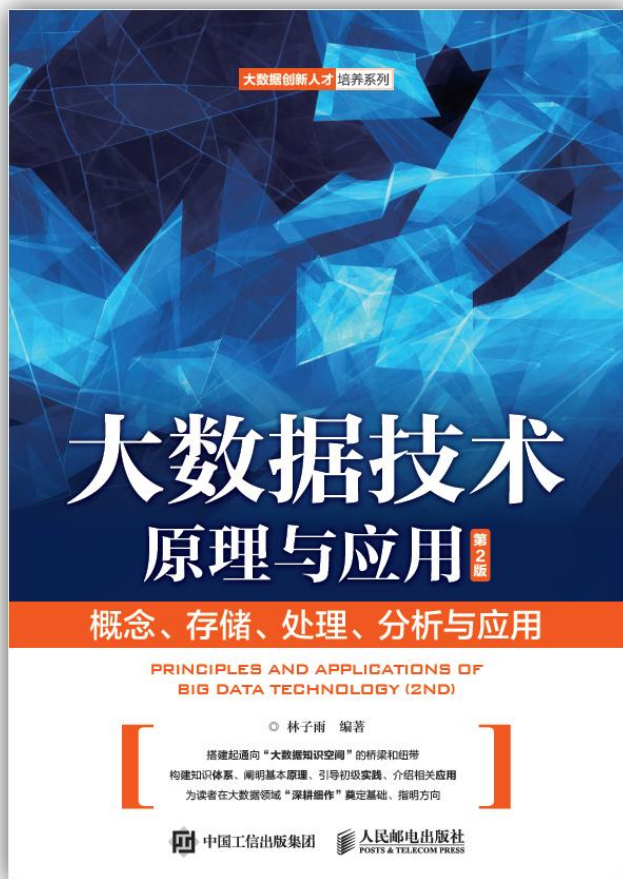
《大数据技术原理与应用——概念、存储、处理、分析与应用（第2版）》，由厦门大学计算机科学系林子雨老师编著，是中国高校第一本系统介绍大数据知识的专业教材。

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、MapReduce和Spark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（第2版）》教材官方网站：

<http://dbllab.xmu.edu.cn/post/bigdata>





附录：中国高校大数据课程公共服务平台



中国高校大数据课程 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

A group of silhouettes of people standing in a circle, holding hands, positioned at the top of the slide.

Thank You!

A group of silhouettes of people standing in a circle, holding hands, positioned at the bottom of the slide.

Department of Computer Science, Xiamen University, 2017