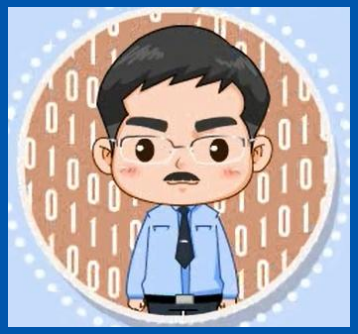


《Flink编程基础（Scala版）》

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

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 Flink环境搭建和使用方法

（PPT版本号：2021年3月版本）

扫一扫访问教材官网

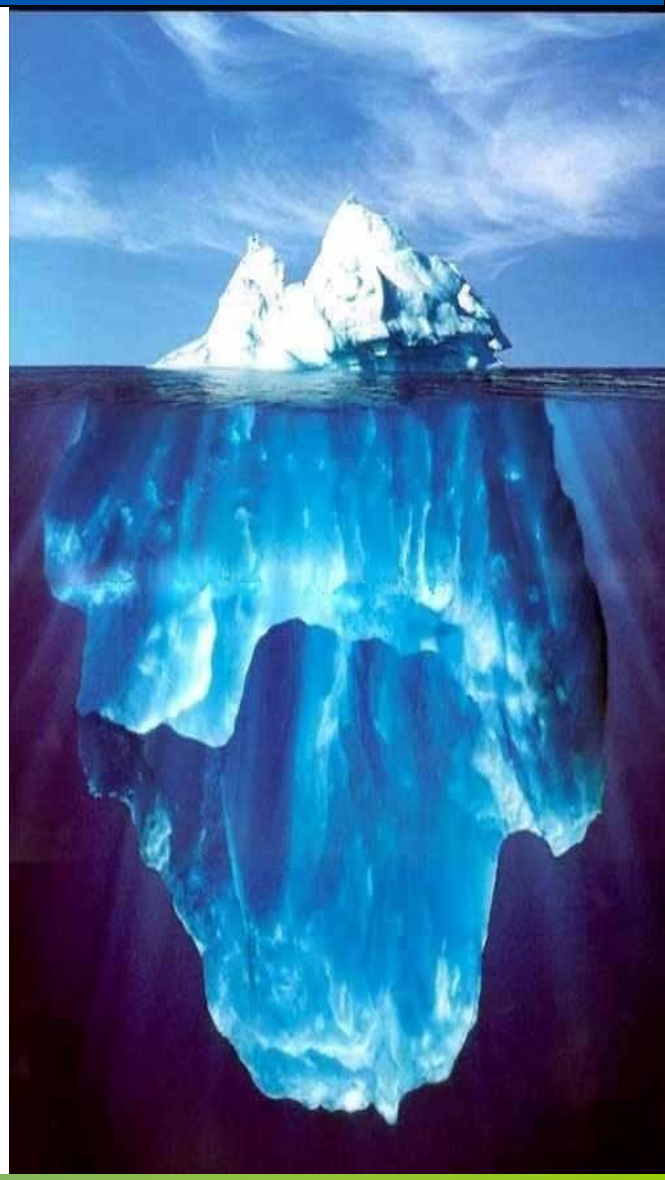
林子雨

厦门大学计算机科学系

E-mail: ziyulin@xmu.edu.cn

主页：http://dblab.xmu.edu.cn/linziyu

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Flink

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 Flink和Hadoop的交互

4.1.1 基础环境

- Linux系统：Ubuntu18.04
- Hadoop：3.1.3版本
- JDK：1.8以上
- Flink：1.11.2版本

4.1.2 下载安装文件

下载Flink安装文件flink-1.11.2-bin-scala_2.12.tgz，保存到了Linux系统的“/home/hadoop/Downloads”目录下。

```
$ cd  /home/hadoop
$ sudo  tar  -zxvf  ~/Downloads/flink-1.11.2-bin-scala_2.12.tgz  -C
/usr/local/
$ cd  /usr/local
$ sudo  mv  ./flink-1.11.2  ./flink
$ sudo  chown  -R  hadoop:hadoop  ./flink  # hadoop是当前登录Linux系统
的用户名
```

4.1.3配置相关文件

使用如下命令添加环境变量：

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

在.bashrc文件中添加如下内容：

```
export FLNK_HOME=/usr/local/flink
export PATH=$FLINK_HOME/bin:$PATH
```

保存并退出.bashrc文件，然后执行如下命令让配置文件生效：

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

使用如下命令启动Flink：

```
$ cd /usr/local/flink
$ ./bin/start-cluster.sh
```

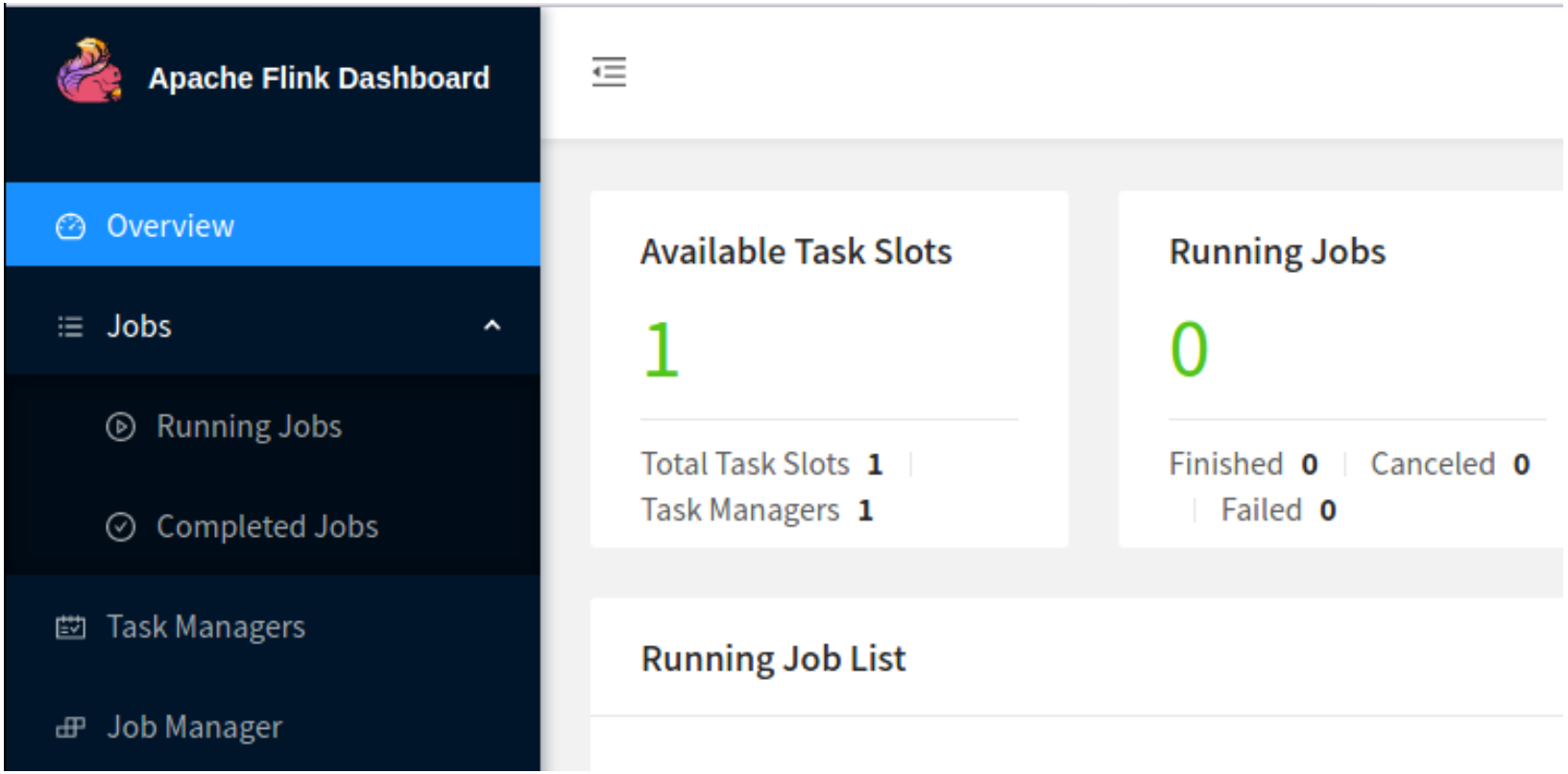
4.1.3配置相关文件

使用jps命令查看进程：

```
$ jps
17942 TaskManagerRunner
18022 Jps
17503
StandaloneSessionClusterEntrypoint
```

4.1.3配置相关文件

Flink的JobManager同时会在8081端口上启动一个Web前端，可以在浏览器中输入“http://localhost:8081”来访问



4.1.3配置相关文件

Flink安装包中自带了测试样例，这里可以运行WordCount样例程序来测试Flink的运行效果，具体命令如下：

```
$ cd /usr/local/flink/bin
$ ./flink run /usr/local/flink/examples/batch/WordCount.jar

Starting execution of program
Executing WordCount example with default input data set.
Use --input to specify file input.
Printing result to stdout. Use --output to specify output path.
(a,5)
(action,1)
(after,1)
(against,1)
(all,2)
……
```

4.1.4 Flink和Hadoop的交互

经过上面的步骤以后，就在单台机器上按照“Hadoop（伪分布式）+Flink（Local模式）”这种方式完成了Hadoop和Flink组合环境的搭建。Hadoop和Flink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Flink负责数据的计算。

为了能够让Flink操作HDFS中的数据，需要先启动HDFS。打开一个Linux终端，在Linux Shell中输入如下命令启动HDFS：

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

HDFS启动完成后，可以通过命令 jps 来判断是否成功启动，命令如下：

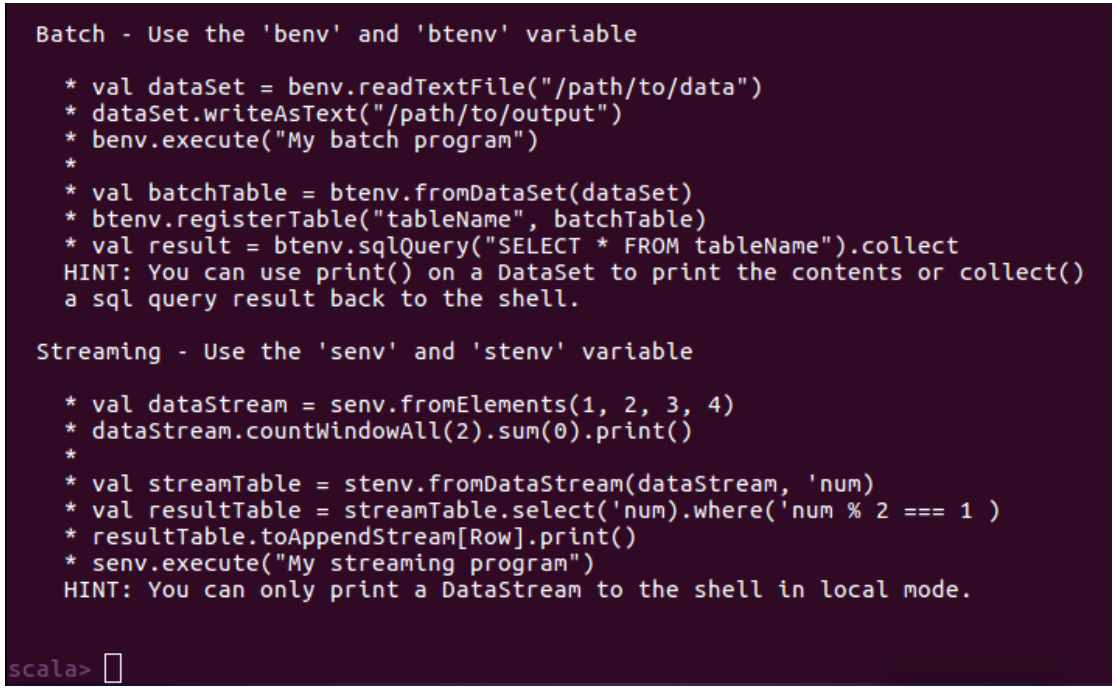
```
$ jps
7100 jps
6867 SecondaryNameNode
6445 NameNode
6594 DataNode
```

4.2 在Scala Shell中运行代码

可以通过下面命令启动Scala Shell环境：

```
$ cd  /usr/local/flink
$ ./bin/start-scala-shell.sh local
```

启动Scala Shell后，就会进入“scala>”命令提示符状态



4.2 在Scala 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Scala Shell

```
scala>:quit
```

4.3 开发Flink独立应用程序

4.3.1安装编译打包工具Maven
4.3.2开发批处理程序
4.3.3开发流处理程序
4.3.4 使用IntelliJ IDEA开发Flink应用程序

4.3.1安装编译打包工具Maven

可以访问Maven官网下载安装文件

https://downloads.apache.org/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.zip

```
$ sudo unzip ~/Downloads/apache-maven-3.6.3-bin.zip -d /usr/local
$ cd /usr/local
$ sudo mv apache-maven-3.6.3/ ./maven
$ sudo chown -R hadoop ./maven
```

为了提高下载速度，需要修改Maven的配置文件，让Maven到国内的阿里云仓库下载相关依赖文件，具体命令如下：

```
$ cd /usr/local/maven/conf
$ vim settings.xml
```

4.3.1安装编译打包工具Maven

打开settings.xml文件以后，清空该文件原来的所有内容，然后，加入如下内容：

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
                      http://maven.apache.org/xsd/settings-1.0.0.xsd">
  <mirrors>
    <mirror>
     <id>aliyunmaven</id>
     <mirrorOf>*</mirrorOf>
     <name>阿里云公共仓库</name>
     <url>https://maven.aliyun.com/repository/public</url>
    </mirror>
```

备注：该文件剩余内容在下一页

4.3.1安装编译打包工具Maven

```
<mirror>
    <id>aliyunmaven</id>
    <mirrorOf>*</mirrorOf>
    <name>阿里云谷歌仓库</name>
    <url>https://maven.aliyun.com/repository/google</url>
   </mirror>
   <mirror>
    <id>aliyunmaven</id>
    <mirrorOf>*</mirrorOf>
    <name>阿里云阿帕奇仓库</name>
    <url>https://maven.aliyun.com/repository/apache-snapshots</url>
   </mirror>
   <mirror>
    <id>aliyunmaven</id>
    <mirrorOf>*</mirrorOf>
    <name>阿里云spring仓库</name>
    <url>https://maven.aliyun.com/repository/spring</url>
   </mirror>
```

备注：该文件剩余内容在下一页

4.3.1安装编译打包工具Maven

```
        <mirror>
          <id>aliyunmaven</id>
          <mirrorOf>*</mirrorOf>
          <name>阿里云spring插件仓库</name>
          <url>https://maven.aliyun.com/repository/spring-plugin</url>
        </mirror>
    </mirrors>
</settings>
```

4.3.2开发批处理程序

编写WordCount批处理程序主要包括以下几个步骤：

- 编写代码
- 使用Maven编译打包程序
- 通过flink run命令运行程序

4.3.2开发批处理程序

1.编写代码

在Linux终端中执行如下命令，在用户主目录下创建一个目录flinkapp作为应用程序根目录：

```
$ cd ~ #进入用户主目录
$ mkdir -p ./flinkapp/src/main/scala
```

在“./flinkapp/src/main/scala”目录下建立代码文件WordCount.scala

4.3.2开发批处理程序

```
package cn.edu.xmu.dblab

import org.apache.flink.api.scala._

object WordCount {
  def main(args: Array[String]): Unit = {

    //第1步：建立执行环境
    val env = ExecutionEnvironment.getExecutionEnvironment

    //第2步：创建数据源
    val text = env.fromElements(
      "hello, world!",
      "hello, world!",
      "hello, world!“)
```

4.3.2开发批处理程序

```
    //第3步：对数据集指定转换操作
val counts = text.flatMap { _.toLowerCase.split(" ") }
      .map { (_, 1) }
      .groupBy(0)
      .sum(1)

    // 第4步：输出结果
    counts.print()
  }
}
```

4.3.2开发批处理程序

可以看出，整个Flink批处理应用程序一共包括4个步骤：

- 第1步：建立执行环境
- 第2步：创建数据源
- 第3步：对数据集指定转换操作
- 第4步：输出结果

4.3.2开发批处理程序

需要在flinkapp目录下新建文件pom.xml，然后，在pom.xml文件中添加如下内容：

```
<project>
    <groupId>cn.edu.xmu.dblab</groupId>
    <artifactId>wordcount</artifactId>
    <modelVersion>4.0.0</modelVersion>
    <name>WordCount</name>
    <packaging>jar</packaging>
    <version>1.0</version>
    <repositories>
        <repository>
            <id>alimaven</id>
            <name>aliyun maven</name>
            <url>http://maven.aliyun.com/nexus/content/groups/public/</url>
        </repository>
    </repositories>
```

备注：pom.xml剩余内容在下一页

4.3.2开发批处理程序

```
<dependencies>
    <dependency>
        <groupId>org.apache.flink</groupId>
        <artifactId>flink-scala_2.12</artifactId>
        <version>1.11.2</version>
    </dependency>
    <dependency>
        <groupId>org.apache.flink</groupId>
        <artifactId>flink-streaming-scala_2.12</artifactId>
        <version>1.11.2</version>
    </dependency>
    <dependency>
        <groupId>org.apache.flink</groupId>
        <artifactId>flink-clients_2.12</artifactId>
        <version>1.11.2</version>
    </dependency>
</dependencies>
```

备注：pom.xml剩余内容在下一页

4.3.2开发批处理程序

```
<build>
    <plugins>
        <plugin>
            <groupId>net.alchim31.maven</groupId>
            <artifactId>scala-maven-plugin</artifactId>
            <version>3.4.6</version>
            <executions>
                <execution>
                    <goals>
                        <goal>compile</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
```

备注：pom.xml剩余内容在下一页

4.3.2开发批处理程序

```
<plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-assembly-plugin</artifactId>
        <version>3.0.0</version>
        <configuration>
            <descriptorRefs>
                <descriptorRef>jar-with-dependencies</descriptorRef>
            </descriptorRefs>
        </configuration>
```

备注：pom.xml剩余内容在下一页

4.3.2开发批处理程序

```
<executions>
                <execution>
                    <id>make-assembly</id>
                    <phase>package</phase>
                    <goals>
                        <goal>single</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
</project>
```

4.3.2开发批处理程序

2.使用Maven编译打包程序

为了保证Maven能够正常运行，先执行如下命令检查整个应用程序的文件结构：

```
$ cd ~/flinkapp
$ find .

.
./src
./src/main
./src/main/scala
./src/main/scala/WordCount.scala
./pom.xml
```

4.3.2开发批处理程序

接下来，我们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 ~/flinkapp    #一定把这个目录设置为当前目录
$ /usr/local/maven/bin/mvn package

[INFO] Building jar: /home/hadoop/flinkapp/target/wordcount-1.0-jar-with-dependencies.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  20.929 s
[INFO] Finished at: 2020-09-28T15:24:30+08:00
[INFO] ------------------------------------------------------------------------
```

4.3.2开发批处理程序

3.通过flink run命令运行程序

下面是提交运行程序的具体命令（请确认已经启动Flink）：

```
$ cd ~/flinkapp
$ /usr/local/flink/bin/flink run --class
cn.edu.xmu.dblab.WordCount ./target/wordcount-1.0.jar
```

执行成功后，可以在屏幕上看到词频统计结果

4.3.3开发流处理程序

编写WordCount流处理程序主要包括以下几个步骤：

- 编写代码
- 使用Maven编译打包程序
- 通过flink run命令运行程序

4.3.3开发流处理程序

1.编写代码

在Linux终端中执行如下命令，在用户主目录下创建一个文件夹flinkapp2作为应用程序根目录：

```
$ cd ~  #进入用户主目录
$ mkdir -p ./flinkapp2/src/main/scala
```

4.3.3开发流处理程序

在“./flinkapp2/src/main/scala”目录下建立代码文件StreamWordCount.scala，其内容如下：

```
package cn.edu.xmu.dblab

import org.apache.flink.streaming.api.scala._
import org.apache.flink.streaming.api.scala.StreamExecutionEnvironment
import org.apache.flink.streaming.api.windowing.time.Time

object StreamWordCount{
  def main(args: Array[String]): Unit = {

    //第1步：建立执行环境
    val env = StreamExecutionEnvironment.getExecutionEnvironment

    //第2步：创建数据源
    val source = env.socketTextStream("localhost",9999,'\n')
```

备注：剩余代码在下一页

4.3.3开发流处理程序

```
//第3步：对数据集指定转换操作逻辑
  val dataStream = source.flatMap(_.split(" "))
    .map((_,1))
    .keyBy(0)
    .timeWindow(Time.seconds(2),Time.seconds(2))
    .sum(1)

  //第4步：指定计算结果输出位置
  dataStream.print()

  //第5步：指定名称并触发流计算
  env.execute("Flink Streaming Word Count")
 }
}
```

4.3.3开发流处理程序

可以看出，整个Flink流处理应用程序一共包括5个步骤：

- 第1步：建立执行环境
- 第2步：创建数据源
- 第3步：对数据集指定转换操作逻辑
- 第4步：指定计算结果输出位置
- 第5步：指定名称并触发流计算

4.3.3开发流处理程序

2.使用Maven编译打包程序

为了保证Maven能够正常运行，先执行如下命令检查整个应用程序的文件结构：

```
$ cd ~/flinkapp2
$ find .

.
./src
./src/main
./src/main/scala
./src/main/scala/StreamWordCount.scala
./pom.xml
```

4.3.3开发流处理程序

接下来，我们可以通过如下代码将整个应用程序打包成JAR包：

```
$ cd ~/flinkapp2    #一定把这个目录设置为当前目录
$ /usr/local/maven/bin/mvn package
```

3.通过flink run命令运行程序

可以使用如下nc命令生成一个Socket服务器端：

```
$ nc  -lk  9999
```

4.3.3开发流处理程序

新建一个Linux终端窗口，将上面生成的JAR包通过flink run命令提交到Flink中运行（请确认已经启动Flink），命令如下：

```
$ cd ~/flinkapp2
$ /usr/local/flink/bin/flink run --class
cn.edu.xmu.dblab.StreamWordCount ./target/wordcount-1.0.jar
```

接下来，切换到“NC窗口”，在该窗口内输入如下三行内容（每输入一行就回车）：

```
hello hadoop
hello spark
hello flink
```

4.3.3开发流处理程序

最后，再新建一个Linux终端窗口，在里面输入如下命令查看词频统计结果：

```
$ cd /usr/local/flink/log
$ tail -f flink*.out

==> flink-hadoop-taskexecutor-1-ubuntu.out <==
(hello,1)
(hadoop,1)
(hello,1)
(spark,1)
(hello,1)
(flink,1)
```

4.3.4 使用IntelliJ IDEA开发Flink应用程序

1. 下载和安装IDEA
2. 下载Scala插件安装包
3. 启动IDEA
4. 为IDEA安装Scala插件
5. 使用IDEA开发WordCount程序

4.3.4 使用IntelliJ IDEA开发Flink应用程序

1. 下载和安装IDEA

下载安装文件ideaIC-2020.2.3.tar.gz，保存到“~/Downloads”目录下
执行如下命令进行IDEA的安装：

```
$ cd ~  #进入用户主目录
$ sudo tar -zxvf /home/hadoop/download/ideaIC-2020.2.3.tar.gz -C
/usr/local  #解压文件
$ cd /usr/local
$ sudo mv ./idea-IU-202.7660.26 ./idea   #重命名，方便操作
$ sudo chown -R hadoop ./idea   #为当前Linux用户hadoop赋予针对idea目
录的权限
```

4.3.4 使用IntelliJ IDEA开发Flink应用程序

2. 下载Scala插件安装包

访问Scala插件网站（http://plugins.jetbrains.com/plugin/1347-scala）
下载Scala插件安装包scala-intellij-bin-2020.2.3.zip

3. 启动IDEA

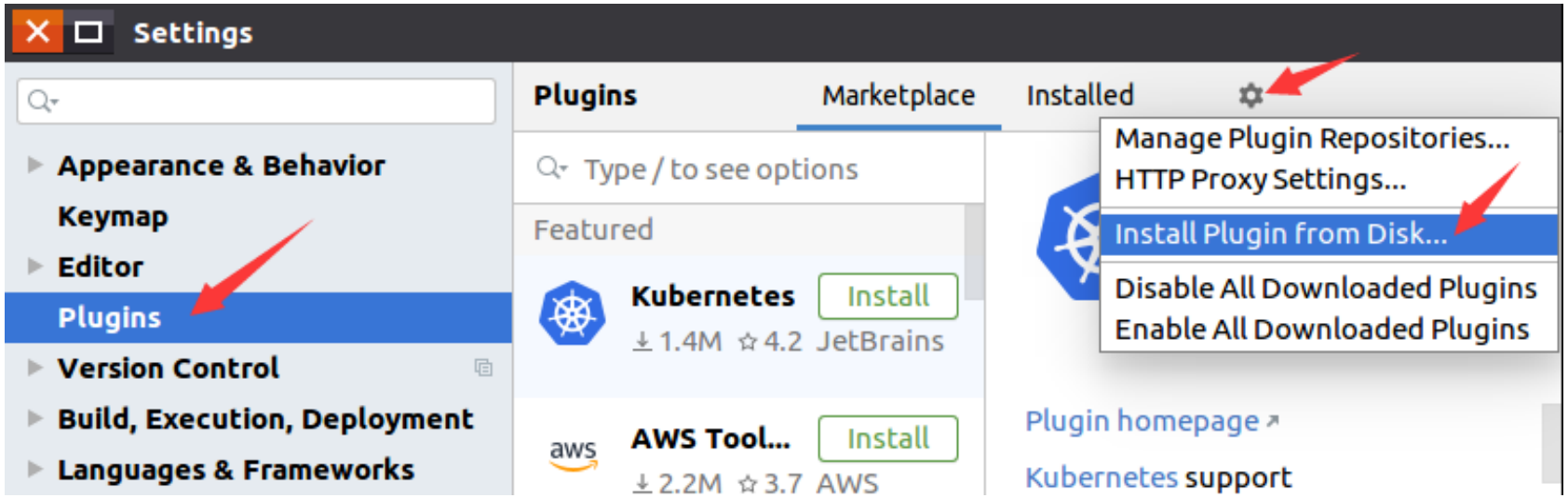
打开一个Linux终端，使用如下命令启动开发工具IDEA：

```
$ cd /usr/local/idea
$ ./bin/idea.sh
```

4.3.4 使用IntelliJ IDEA开发Flink应用程序

4. 为IDEA安装Scala插件

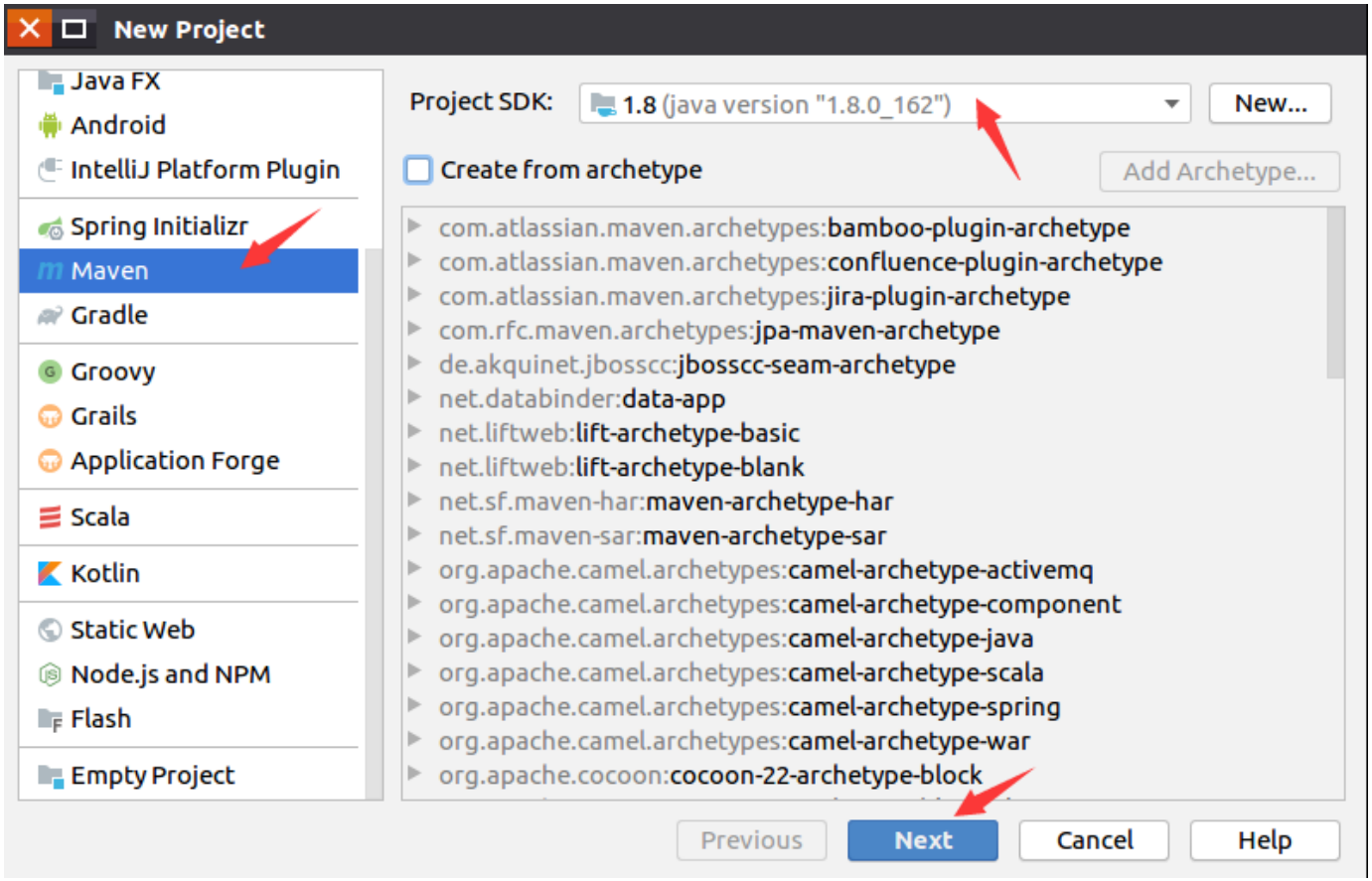
从“File”菜单下的子菜单“Settings…”进入打开“Plugins”界面



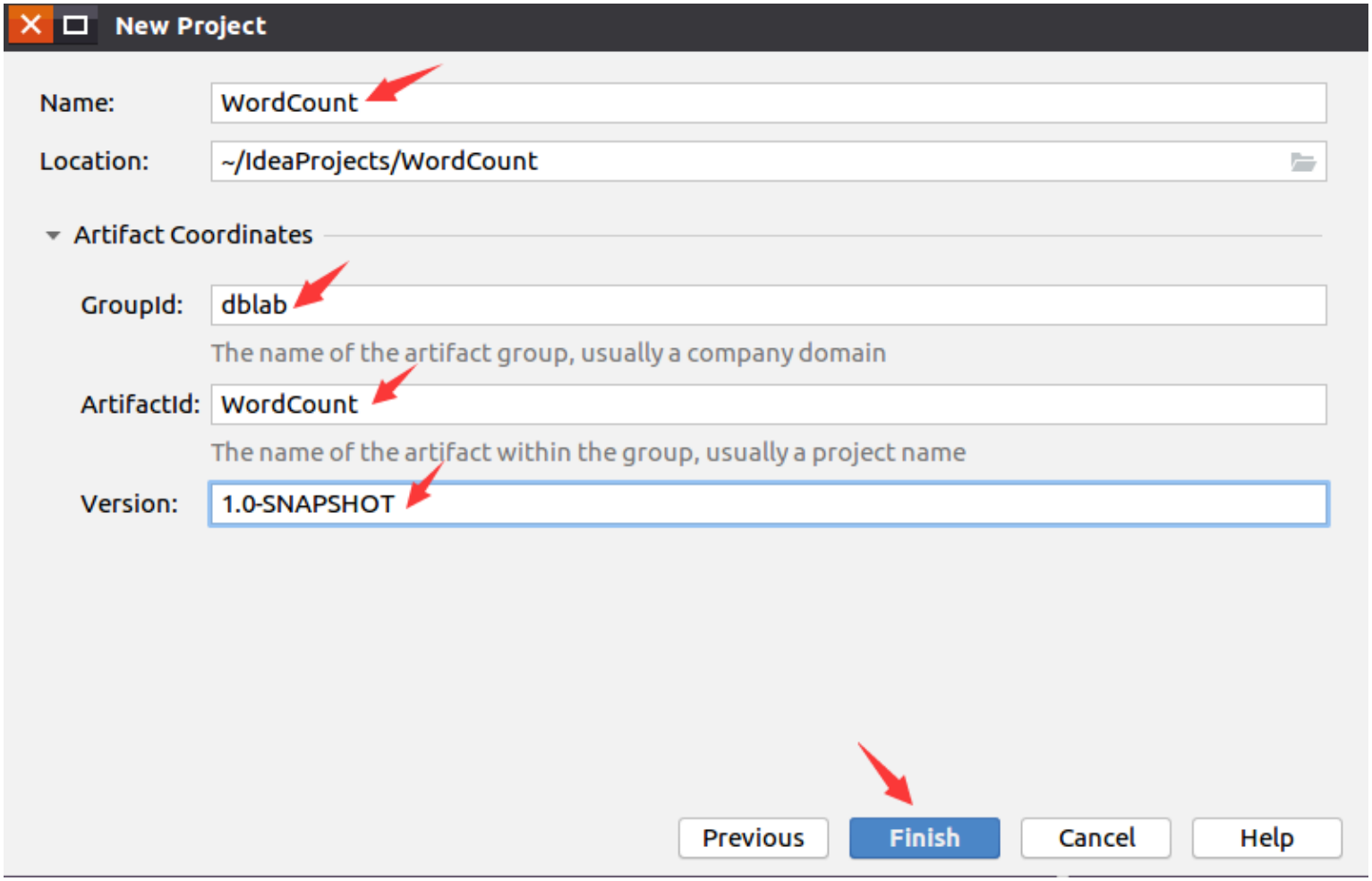
4.3.4 使用IntelliJ IDEA开发Flink应用程序

5. 使用IDEA开发WordCount程序

通过菜单“File->New->Project”打开一个新建项目对话框

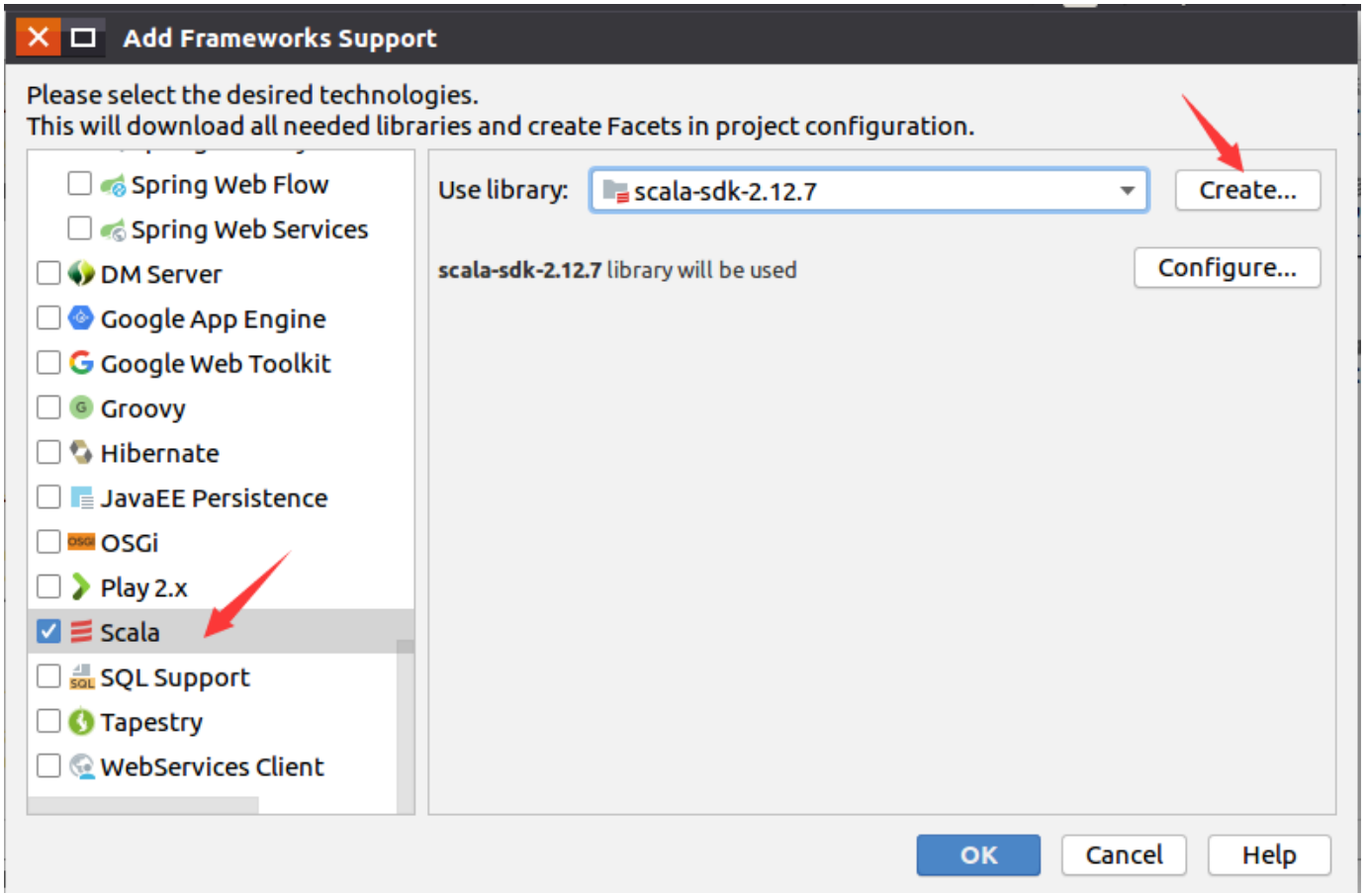


4.3.4 使用IntelliJ IDEA开发Flink应用程序



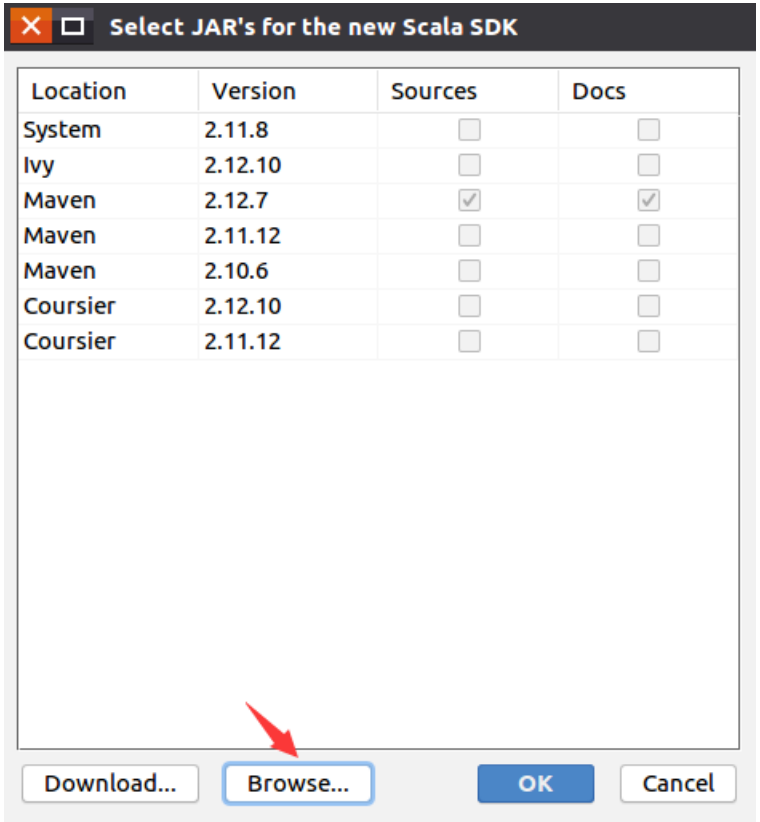
4.3.4 使用IntelliJ IDEA开发Flink应用程序

下面需要为项目添加Scala框架支持，从而可以新建Scala代码文件
在项目名称“WordCount”上点击鼠标右键，在弹出的子菜单中选择
“Add Framework Support...”。



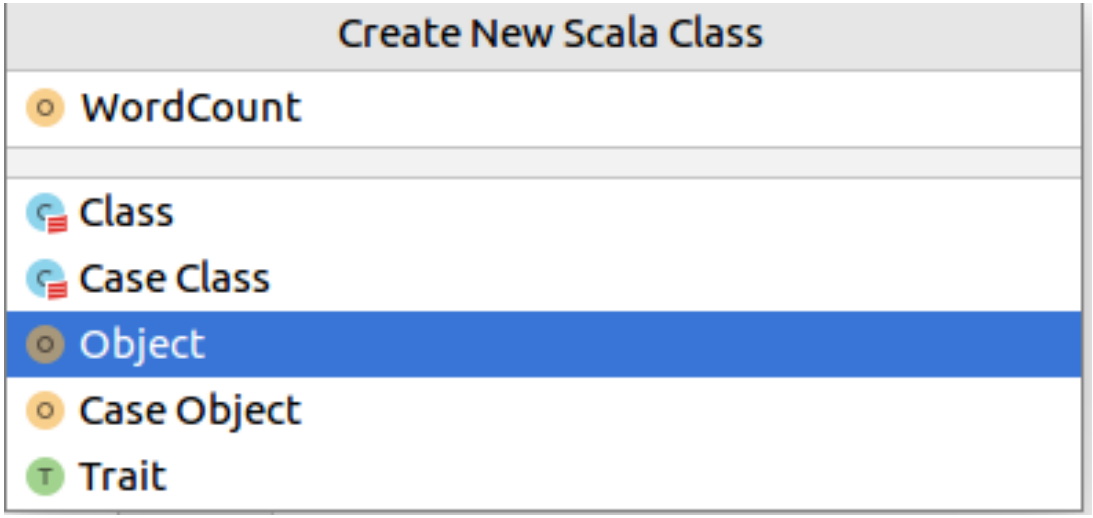
4.3.4 使用IntelliJ IDEA开发Flink应用程序

在弹出的界面中（如图4-7所示），点击“Browse...”按钮，然后找到Scala2.12.12的安装目录（学习第2章内容时，已经在“/usr/local/scala”目录下安装了Scala2.12.12），点击“OK”按钮，回到上一级界面以后再次点击“OK”按钮，完成设置。



4.3.4 使用IntelliJ IDEA开发Flink应用程序

在项目目录树的“Scala”子目录上单击鼠标右键，在弹出的菜单中选择“New”，再在弹出的菜单中选择“Scala Class”，然后，在弹出的界面中（如图4-8所示），输入类的名称“WordCount”，类型选择“Object”，然后回车，就可以创建一个空的代码文件WordCount.scala。



4.3.4 使用IntelliJ IDEA开发Flink应用程序

在代码文件WordCount.scala中输入代码

```
package cn.edu.xmu.dblab

import org.apache.flink.api.scala._

object WordCount {
  def main(args: Array[String]): Unit = {

    //第1步：建立执行环境
    val env = ExecutionEnvironment.getExecutionEnvironment

    //第2步：创建数据源
    val text = env.fromElements(
      "hello, world!",
      "hello, world!",
      "hello, world!“)
```

4.3.4 使用IntelliJ IDEA开发Flink应用程序

```
    //第3步：对数据集指定转换操作
val counts = text.flatMap { _.toLowerCase.split(" ") }
      .map { (_, 1) }
      .groupBy(0)
      .sum(1)

    // 第4步：输出结果
    counts.print()
  }
}
```

4.3.4 使用IntelliJ IDEA开发Flink应用程序

在pom.xml文件中输入依赖

```
<project>
    <groupId>cn.edu.xmu.dblab</groupId>
    <artifactId>wordcount</artifactId>
    <modelVersion>4.0.0</modelVersion>
    <name>WordCount</name>
    <packaging>jar</packaging>
    <version>1.0</version>
    <repositories>
        <repository>
            <id>alimaven</id>
            <name>aliyun maven</name>
            <url>http://maven.aliyun.com/nexus/content/groups/public/</url>
        </repository>
    </repositories>
```

备注：pom.xml剩余内容在下一页

4.3.4 使用IntelliJ IDEA开发Flink应用程序

```
<dependencies>
    <dependency>
        <groupId>org.apache.flink</groupId>
        <artifactId>flink-scala_2.12</artifactId>
        <version>1.11.2</version>
    </dependency>
    <dependency>
        <groupId>org.apache.flink</groupId>
        <artifactId>flink-streaming-scala_2.12</artifactId>
        <version>1.11.2</version>
    </dependency>
    <dependency>
        <groupId>org.apache.flink</groupId>
        <artifactId>flink-clients_2.12</artifactId>
        <version>1.11.2</version>
    </dependency>
</dependencies>
```

备注：pom.xml剩余内容在下一页

4.3.4 使用IntelliJ IDEA开发Flink应用程序

```
<build>
    <plugins>
        <plugin>
            <groupId>net.alchim31.maven</groupId>
            <artifactId>scala-maven-plugin</artifactId>
            <version>3.4.6</version>
            <executions>
                <execution>
                    <goals>
                        <goal>compile</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
```

备注：pom.xml剩余内容在下一页

4.3.4 使用IntelliJ IDEA开发Flink应用程序

```
<plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-assembly-plugin</artifactId>
        <version>3.0.0</version>
        <configuration>
            <descriptorRefs>
                <descriptorRef>jar-with-dependencies</descriptorRef>
            </descriptorRefs>
        </configuration>
```

备注：pom.xml剩余内容在下一页

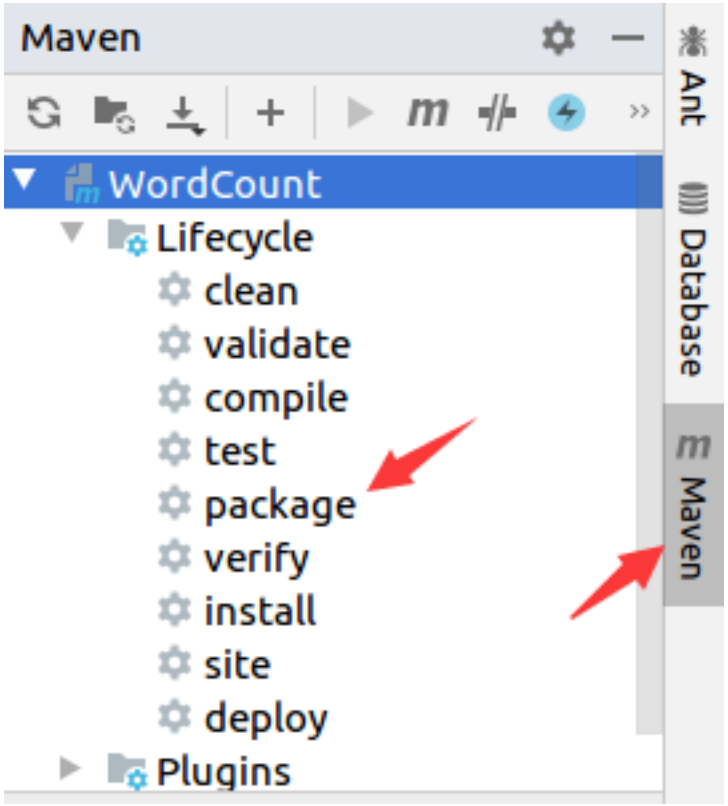
4.3.4 使用IntelliJ IDEA开发Flink应用程序

```
<executions>
                <execution>
                    <id>make-assembly</id>
                    <phase>package</phase>
                    <goals>
                        <goal>single</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>
  </build>
</project>
```

在WordCount.scala代码窗口内，单击鼠标右键，在弹出的菜单中选择“Run...”，就可以启动程序运行，最后会在界面底部的信息栏内出现词频统计信息。

4.3.4 使用IntelliJ IDEA开发Flink应用程序

程序运行成功以后，可以对程序进行打包，以便部署到Flink平台上。具体方法是：在项目开发界面的右边，点击“Maven”按钮，然后在弹出的界面中（如图4-9所示），双击“package”，就可以完成对应用程序的打包。打包成功以后，可以在项目开发界面左侧的目录树中，在target子目录下找到两个文件“WordCount-1.0.jar”和“WordCount-1.0-jar-with-dependencies.jar”。



4.3.4 使用IntelliJ IDEA开发Flink应用程序

打包成功以后，就可以提交到Flink系统中运行。下面是提交运行程序的具体命令（请确认已经启动Flink）：

```
$ cd ~/flinkapp
$ /usr/local/flink/bin/flink run --class
cn.edu.xmu.dblab.WordCount ./target/WordCount-1.0.jar
```

4.4 设置程序运行并行度

Flink的每个TaskManager为集群提供插槽（slot）。插槽可以看成一个资源组，插槽的数量通常与每个TaskManager节点的可用CPU内核数成比例。一般情况下，插槽数是每个节点的CPU的核数。

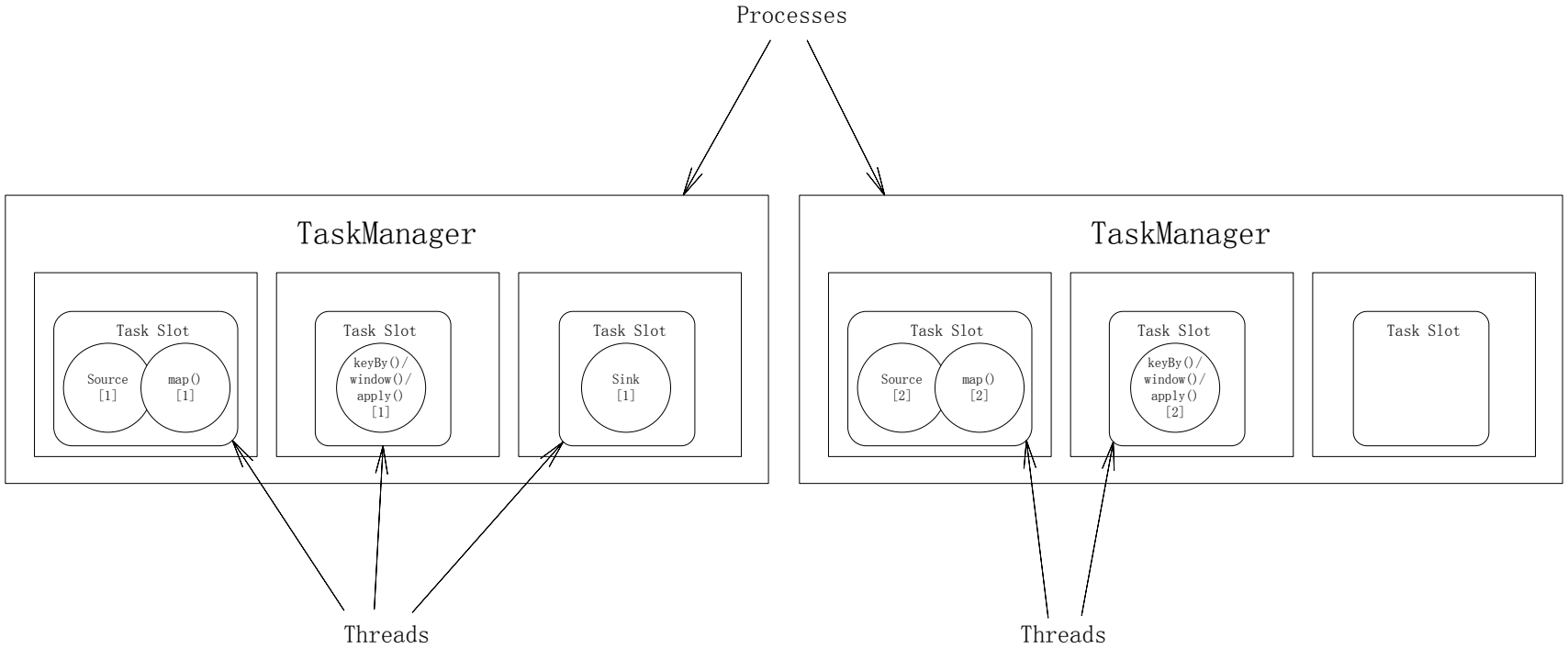


图 4-10 Flink的插槽和并行度

4.4 设置程序运行并行度

在Flink中，一个任务会被分解成多个子任务，然后这些子任务由多个并行的线程来执行，一个任务的并行线程数目就被称为该任务的并行度。由于Flink会将这些子任务分配到插槽来并行执行，因此，任务的最大并行度是由每个Task Manager上可用的插槽数量决定的。

比如，如果Task Manager有四个插槽，那么它将为每个插槽分配25%的内存。可以在一个插槽中运行一个或多个线程。同一插槽中的线程共享相同的JVM。 同一JVM中的任务共享TCP连接和心跳消息。Task Manager的一个插槽代表一个可用线程，该线程具有固定的内存（需要注意的是，插槽只对内存隔离，并没有对CPU隔离）。默认情况下，Flink允许子任务共享插槽，即使它们是不同任务的子任务，只要它们来自相同的作业。这种共享可以实现更好的资源利用率。

在图4-10中，Source/map()/keyby()/window()/apply()这些操作的并行度为2，Sink的并行度为1。

4.4 设置程序运行并行度

任务的并行度设置可以从多个层次指定，包括算子层次、执行环境层次、客户端层次和系统层次。这里只介绍执行环境层次的并行度设置方法，其他层次的并行度设置方法可以参考Flink官网资料。

```
// 设置执行环境
    val env = StreamExecutionEnvironment.getExecutionEnvironment
//设置程序并行度
env.setParallelism(1)
```

4.5 Flink集群环境搭建

这里采用Standalone模式

搭建Flink集群主要包括以下5个步骤：

- 集群基础配置
- 在集群中安装Java
- 设置SSH无密码登录
- 安装和配置Flink
- 启动和关闭Flink集群

4.5.1 集群基础配置

这里采用3台机器（节点）作为实例来演示如何搭建Flink集群，其中1台机器（节点）作为Master节点（主机名为Master，IP地址是192.168.1.101），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Slave1（IP地址是192.168.1.102）和Slave2（IP地址是192.168.1.103）。

在Master节点上执行如下命令修改主机名：

```
$ sudo vim /etc/hostname
```

把文件中的原有内容全部清空，只加入一行记录“Master”

按照同样的方法，把Slave1节点上的“/etc/hostname”文件中的主机名修改为“Slave1”，把Slave2节点上的“/etc/hostname”文件中的主机名修改为“Slave2”。

4.5.1 集群基础配置

在Master节点上打开并修改Master中的“/etc/hosts”文件
在hosts文件中增加如下3条IP和主机名映射关系：

192.168.1.101 Master
192.168.1.102 Slave1
192.168.1.103 Slave2

分别到Slave1和Slave2节点上，修改“/etc/hosts”的内容，在hosts文件中增加如下3条IP和主机名映射关系：

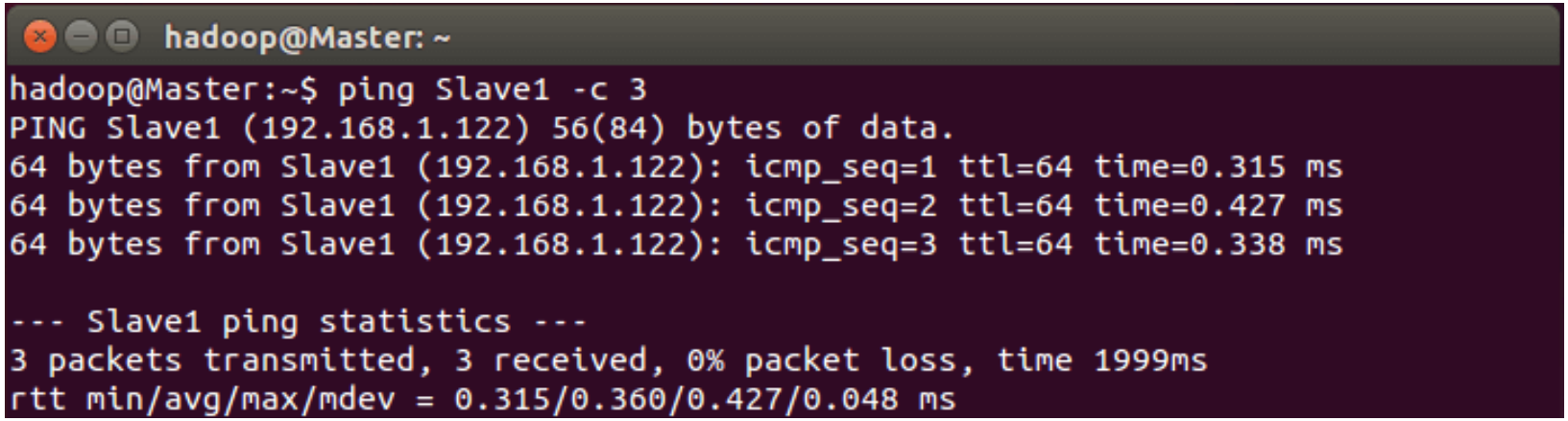
192.168.1.101 Master
192.168.1.102 Slave1
192.168.1.103 Slave2

修改完成以后，请重新启动各个节点的Linux系统。

4.5.1 集群基础配置

需要在各个节点上都执行如下命令，测试是否相互ping得通，如果ping不通，后面就无法顺利配置成功：

```
$ ping Master -c 3   #只ping 3次就会停止，否则要按Ctrl+c中断ping命令
$ ping Slave1 -c 3
```



4.5.2 在集群中安装Java

Flink是运行在JVM上的，因此，需要为集群中的每台机器安装Java环境。对于Flink1.11.2而言，要求使用JDK1.8或者更新的版本。

访问Oracle官网（https://www.oracle.com/technetwork/java/javase/downloads）下载JDK1.8安装包jdk-8u162-linux-x64.tar.gz，保存到“~/Downloads”目录下

执行如下命令创建“/usr/lib/jvm”目录用来存放JDK文件：

```
$cd /usr/lib
$sudo mkdir jvm #创建/usr/lib/jvm目录用来存放JDK文件
```

执行如下命令对安装文件进行解压缩：

```
$cd ~ #进入hadoop用户的主目录
$cd Downloads
$sudo tar -zxvf ./jdk-8u162-linux-x64.tar.gz -C /usr/lib/jvm
```

4.5.2 在集群中安装Java

下面继续执行如下命令，设置环境变量：

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

上面命令使用vim编辑器打开了hadoop这个用户的环境变量配置文件，请在这个文件的开头位置，添加如下几行内容：

```
export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_162
export JRE_HOME=${JAVA_HOME}/jre
export CLASSPATH=.:${JAVA_HOME}/lib:${JRE_HOME}/lib
export PATH=${JAVA_HOME}/bin:$PATH
```

继续执行如下命令让.bashrc文件的配置立即生效：

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

4.5.2 在集群中安装Java

这时，可以使用如下命令查看是否安装成功：

```
$java -version
```

如果能够在屏幕上返回如下信息，则说明安装成功：

```
java version "1.8.0_162"
Java(TM) SE Runtime Environment (build 1.8.0_162-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.162-b12, mixed mode)
```

4.5.3 设置SSH无密码登录

SSH 是 Secure Shell 的缩写，是建立在应用层和传输层基础上的安全协议。SSH是目前较可靠、专为远程登录会话和其他网络服务提供安全性的协议。利用SSH协议可以有效防止远程管理过程中的信息泄露问题。SSH最初是UNIX系统上的一个程序，后来又迅速扩展到其他操作平台。SSH是由客户端和服务端的软件组成，服务端是一个守护进程，它在后台运行并响应来自客户端的连接请求，客户端包含ssh程序以及像scp（远程拷贝）、slogin（远程登录）、sftp（安全文件传输）等其他的应用程序。

为什么在安装Flink之前要配置SSH呢？这是因为，Flink集群中的主节点需要和集群中所有机器建立通信，这个过程需要通过SSH登录来实现。Flink并没有提供SSH输入密码登录的形式，因此，为了能够顺利登录集群中的每台机器，需要将所有机器配置为“主节点可以无密码登录它们”。

4.5.3 设置SSH无密码登录

Ubuntu默认已安装了SSH客户端，因此，这里还需要安装SSH服务端，请在Master节点的Linux终端中执行以下命令：

```
$ sudo apt-get install openssh-server
```

安装后，可以使用如下命令登录本机：

```
$ ssh localhost
```

执行该命令后会出现提示信息，可以输入“yes”，然后按提示输入密码，就登录到本机了。
然后，执行如下命令退出SSH登录：

```
$ exit
```

4.5.3 设置SSH无密码登录

可以看出，现在在Master节点用SSH方式登录本机，是需要密码的。为了让Master节点能够无密码SSH登录本机，需要在Master节点上执行如下命令：

```
$ cd ~/.ssh               # 如果没有该目录，先执行一次ssh localhost
$ rm ./id_rsa*            # 删除之前生成的公匙（如果已经存在）
$ ssh-keygen -t rsa       # 执行该命令后，遇到提示信息，一直按回车就可以
$ cat ./id_rsa.pub >> ./authorized_keys
```

完成后可以执行命令“ssh Master”来验证一下，这时就可以成功登录本机，不需要输入密码了。测试成功后，请执行“exit”命令退出SSH登录。

接下来，在Master节点上执行如下命令将公匙传输到Slave1和Slave2节点上：

```
$ scp ~/.ssh/id_rsa.pub hadoop@Slave1:/home/hadoop/
$ scp ~/.ssh/id_rsa.pub hadoop@Slave2:/home/hadoop/
```

4.5.3 设置SSH无密码登录

接着分别在Slave1和Slave2节点上执行如下命令，将SSH公匙加入授权：

```
$ mkdir ~/.ssh      # 如果不存在该文件夹需先创建，若已存在，则忽略本命令
$ cat ~/id_rsa.pub >> ~/.ssh/authorized_keys
$ rm ~/id_rsa.pub    # 用完以后就可以删掉
```

这样，在Master节点上就可以无密码SSH登录到各个Slave节点了，可在Master节点上执行如下命令进行检验：

```
$ ssh Slave1
$ ssh Slave2
```

如果检验成功，就可以进入后续的安装步骤。

4.5.4 安装和配置Flink

1.在Master节点上安装Flink

在Master节点上执行如下命令安装Flink：

```
$ sudo  tar  -zxf  ~/Downloads/flink-1.11.2-bin-scala_2.12.tgz  -C  /usr/local/
$ cd  /usr/local
$ sudo  mv  ./flink-1.11.2  ./flink
$ sudo  chown  -R  hadoop:hadoop  ./flink  # hadoop是当前登录Linux系统的用户名
```

2.配置环境变量

在.bashrc添加如下配置：

```
export FLNK_HOME=/usr/local/flink
export PATH=$FLINK_HOME/bin:$PATH
```

运行source命令使得配置立即生效：

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

4.5.4 安装和配置Flink

3.配置相关文件

在Master节点上打开文件flink-conf.yaml，增加如下两个配置项：

```
jobmanager.rpc.address: Master
taskmanager.tmp.dirs: /usr/local/flink/tmp
```

需要注意的是，每条配置信息中，冒号后面必须有一个英文空格，否则运行时会报错。

清空masters文件的原有内容，增加如下一行配置：

```
Master:8081
```

清空workers文件的原有内容，增加如下3行配置：

```
Master
Slave1
Slave2
```

4.5.4 安装和配置Flink

4.把Master节点的安装文件发送到Slave节点

在Master节点上执行如下命令，将Master节点上的/usr/local/flink文件夹复制到各个Slave节点上：

```
$ cd  /usr/local/
$ tar  -zcf  ~/flink.master.tar.gz  ./flink
$ cd  ~
$ scp  ./flink.master.tar.gz  Slave1:/home/hadoop
$ scp  ./flink.master.tar.gz  Slave2:/home/hadoop
```

在Slave1和Slave2节点上分别执行下面同样的操作：

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

4.5.4 安装和配置Flink

5.建立tmp目录

在前面配置flink-conf.yaml时，我们设置了临时数据的保存目录“/usr/local/flink/tmp”。但是，Flink自己不会自动创建这个目录，因此，需要在Master、Slave1和Slave2上分别执行如下命令创建tmp目录并设置权限：

```
$ cd /usr/local/flink
$ sudo mkdir tmp
$ sudo chmod -R 755 ./tmp
```

4.5.5 启动和关闭Flink集群

在Master节点上执行如下命令启动Flink集群：

```
$ cd  /usr/local/flink/
$ ./bin/start-cluster.sh
```

启动以后，在Master节点上执行jps命令，可以看到如下信息：

```
$ jps
7265 Jps
5829 StandaloneSessionClusterEntrypoint
6153 TaskManagerRunner
```

在Slave1和Slave2节点上分别执行jps命令，可以看到如下信息：

```
$ jps
4757 TaskManagerRunner
5639 Jps
```

4.5.5 启动和关闭Flink集群

如果能够看到上述信息，说明集群启动成功。启动成功以后，可以在Master节点上打开浏览器，访问http://master:8081，就可以通过浏览器查看Flink集群信息。

Flink安装包中自带了测试样例，可以在Master、Slave1和Slave2中的任意一个节点上运行WordCount样例程序来测试Flink的运行效果，具体命令如下：

```
$ cd /usr/local/flink/bin
$ ./flink run /usr/local/flink/examples/batch/WordCount.jar
```

执行以后，屏幕上就会出现词频统计信息。

最后，可以在Master节点上执行如下命令关闭Flink集群：

```
$ cd /usr/local/flink
$ ./bin/stop-cluster.sh
```

4.6 本章小结

•Flink可以支持多种部署模式，在日常学习和应用开发环节，可以使用单机环境进行部署。本章首先介绍了Flink在单机环境下的安装配置方法，以及Flink和Hadoop的交互方法。
•Scala Shell是一种交互式开发环境，可以立即解释执行用户输入的语句。目前，Flink提供了三种Scala Shell模式，包括Local、Remote Cluster和YARN Cluaster，本章以Local模式为例介绍了Scala Shell的使用方法。
•在开发Flink独立应用程序时，需要采用Maven等工具对代码进行编译打包，然后通过flink run命令提交运行程序。本章介绍了使用Maven工具编译打包Flink程序的具体方法，需要注意的是，一定要确保pom.xml文件中添加了程序所需要的各种外部依赖。
•本章最后介绍了Flink集群环境的搭建方法，但是，这里不建议初学者在集群环境下学习和实践Flink程序，因为，在集群环境中执行，时常会碰到一些棘手的问题，给学习者带来挫折感。

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://dblab.xmu.edu.cn/post/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），全国高校知名大数据教师，现为厦门大学计算机科学系副教授，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、2017年度和2020年度厦门大学教学类奖教金获得者，荣获2019年福建省精品在线开放课程、2018年厦门大学高等教育成果特等奖、2018年福建省高等教育教学成果二等奖、2018年国家精品在线开放课程。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1000万字高价值的研究和教学资料，累计网络访问量超过1000万次。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200万次，累计访问量超过1000万次。

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：林子雨大数据系列教材

了解全部教材信息：http://dblab.xmu.edu.cn/post/bigdatabook/

附录D：《大数据导论（通识课版）》教材

开设全校公共选修课的优质教材

本课程旨在实现以下几个培养目标：

- 引导学生步入大数据时代，积极投身大数据的变革浪潮之中
- 了解大数据概念，培养大数据思维，养成数据安全意识
- 认识大数据伦理，努力使自己的行为符合大数据伦理规范要求
- 熟悉大数据应用，探寻大数据与自己专业的应用结合点
- 激发学生基于大数据的创新创业热情

高等教育出版社 ISBN:978-7-04-053577-8 定价：32元 版次：2020年2月第1版
教材官网：http://dblab.xmu.edu.cn/post/bigdataintroduction/

附录E：《大数据导论》教材

- 林子雨 编著《大数据导论》
- 人民邮电出版社，2020年9月第1版
- ISBN:978-7-115-54446-9 定价：49.80元

教材官网：http://dblab.xmu.edu.cn/post/bigdata-introduction/

开设大数据专业导论课的优质教材

扫一扫访问教材官网

附录F：《大数据技术原理与应用（第3版）》教材

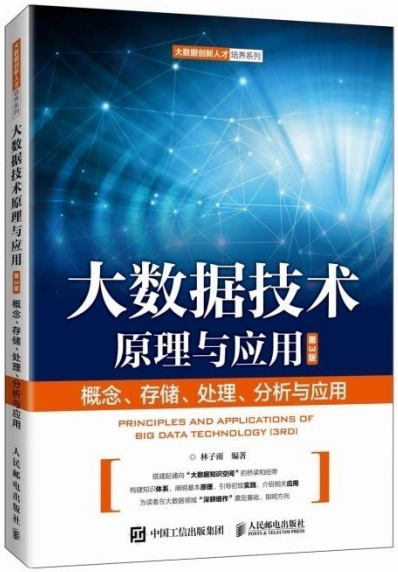
《大数据技术原理与应用——概念、存储、处理、分析与应用（第3版）》，由厦门大学计算机科学系林子雨博士编著，是国内高校第一本系统介绍大数据知识的专业教材。人民邮电出版社　ISBN:978-7-115-54405-6　定价：59.80元

全书共有17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、Flink、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、Spark和Flink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/bigdata3

扫一扫访问教材官网



附录G：《大数据基础编程、实验和案例教程（第2版）》

本书是与《大数据技术原理与应用（第3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

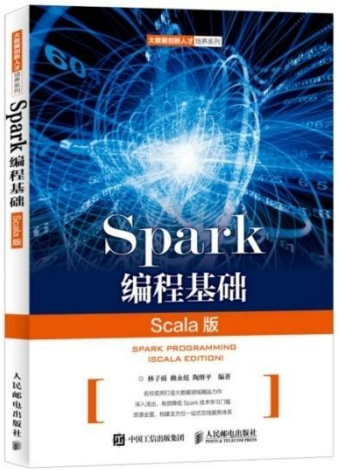
厦门大学林子雨编著

配套实验指导书

- 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，八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

林子雨编著《大数据基础编程、实验和案例教程（第2版）》
清华大学出版社 ISBN:978-7-302-55977-1 定价：69元 2020年10月第2版

附录H：《Spark编程基础（Scala版）》



《Spark编程基础（Scala版）》

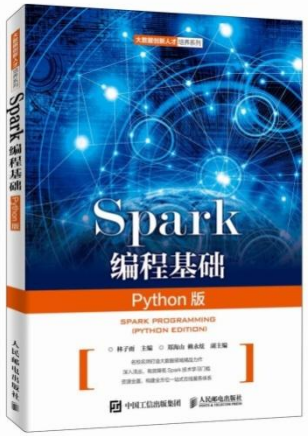
厦门大学 林子雨，赖永炫，陶继平 编著

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I：《Spark编程基础（Python版）》



《Spark编程基础（Python版）》

厦门大学 林子雨，郑海山，赖永炫 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:978-7-115-52439-3
教材官网： http://dblab.xmu.edu.cn/post/spark-python/

本书以Python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、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、Structured 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、上机实验指南等。

附录J：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/

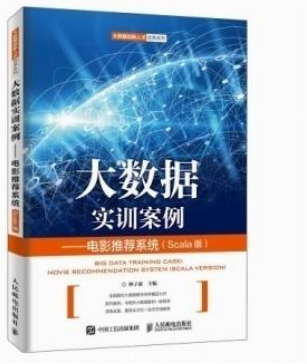
扫一扫访问平台主页

扫一扫观看3分钟FLASH动画宣传片

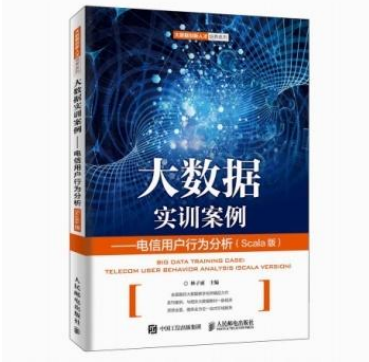
附录K：高校大数据实训课程系列案例教材

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：

《电影推荐系统》（已经于2019年5月出版）
《电信用户行为分析》（已经于2019年5月出版）
《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》



系列案例教材将于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, 2021