# HDFS编程实践（Hadoop3.1.3）

 林子雨老师 2020年1月26日 (updated: 2020年7月9日) 15303

[](http://dblab.xmu.edu.cn/post/bigdataroadmap/)

【版权声明：本指南为[厦门大学林子雨编著的《大数据技术原理与应用》](http://dblab.xmu.edu.cn/post/bigdata/)教材配套学习资料，版权所有，转载请注明出处，请勿用于商业用途】  
  
[点击这里观看厦门大学林子雨老师主讲《大数据技术原理与应用》授课视频](http://www.icourse163.org/course/XMU-1002335004)

本指南介绍Hadoop分布式文件系统HDFS，并详细指引读者对HDFS文件系统的操作实践。请务必仔细阅读完[厦门大学林子雨编著的《大数据技术原理与应用》](http://dblab.xmu.edu.cn/post/bigdata/)第3章节，再结合本指南进行学习。

Hadoop分布式文件系统（Hadoop Distributed File System,HDFS）是Hadoop核心组件之一，如果已经安装了Hadoop，其中就已经包含了HDFS组件，不需要另外安装。

学习本指南需要在Linux系统安装好Hadoop.如果机器上没有安装Linux和Hadoop,请返回[大数据技术原理与应用 第二章 学习指南](http://dblab.xmu.edu.cn/blog/285/)，根据指南学习并安装。

第3章涉及到很多的理论知识点，主要的理论知识点包括：分布式文件系统、HDFS简介、HDFS的相关概念、HDFS体系结构、HDFS的存储原理、HDFS的数据读写过程。这些理论知识点，请自己依靠[厦门大学林子雨编著的《大数据技术原理与应用》](http://dblab.xmu.edu.cn/post/bigdata/)第3章节进行学习，本指南不再重复表述。

接下来介绍Linux操作系统中关于HDFS文件操作的常用Shell命令，利用Web界面查看和管理Hadoop文件系统，以及利用Hadoop提供的Java API进行基本的文件操作。

在学习HDFS编程实践前，我们需要启动Hadoop（版本是Hadoop3.1.3）。执行如下命令

1. cd /usr/local/hadoop
2. ./sbin/start-dfs.sh *#启动hadoop*

Shell 命令

## 一、利用Shell命令与HDFS进行交互

Hadoop支持很多Shell命令，其中fs是HDFS最常用的命令，利用fs可以查看HDFS文件系统的目录结构、上传和下载数据、创建文件等。

**注意**

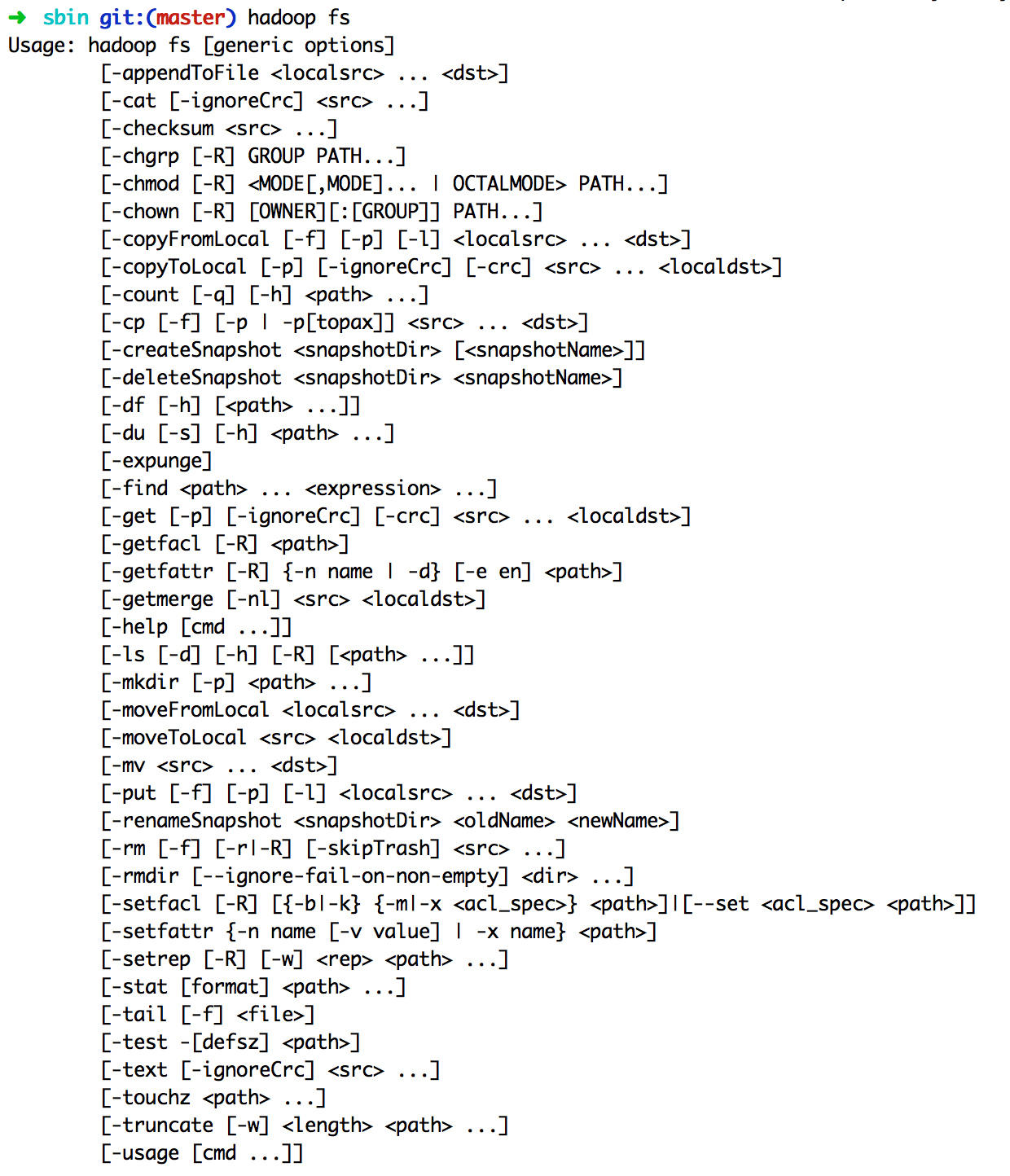
教材《大数据技术原理与应用》的命令是以”./bin/hadoop dfs”开头的Shell命令方式，实际上有三种shell命令方式。  
1. hadoop fs  
2. hadoop dfs  
3. hdfs dfs

hadoop fs适用于任何不同的文件系统，比如本地文件系统和HDFS文件系统  
hadoop dfs只能适用于HDFS文件系统  
hdfs dfs跟hadoop dfs的命令作用一样，也只能适用于HDFS文件系统

我们可以在终端输入如下命令，查看fs总共支持了哪些命令

1. ./bin/hadoop fs

Shell 命令

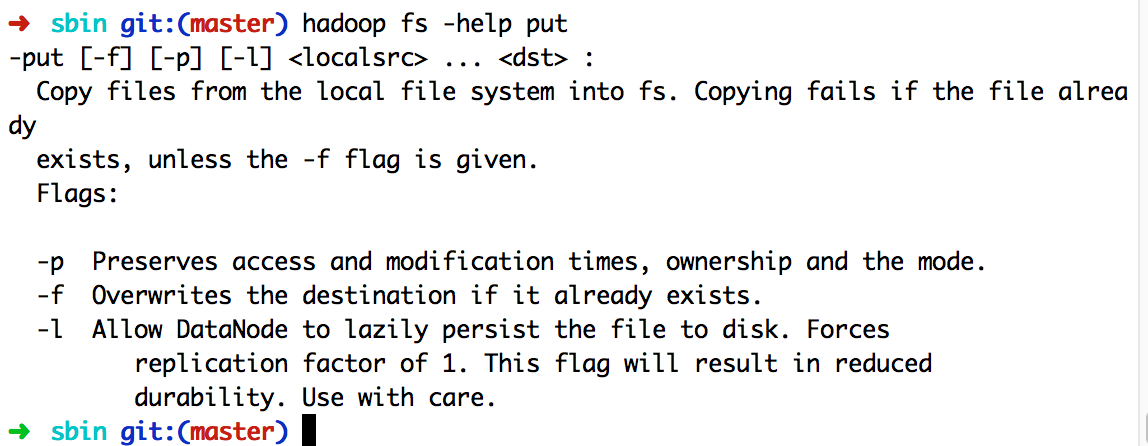


在终端输入如下命令，可以查看具体某个命令的作用

例如：我们查看put命令如何使用，可以输入如下命令

1. ./bin/hadoop fs -help put

Shell 命令



由于Hadoop支持的命令众多，想更多了解Hadoop命令，请参照《大数据技术原理与应用（第2版）》3.7小节学习。

### 1.目录操作

需要注意的是，Hadoop系统安装好以后，第一次使用HDFS时，需要首先在HDFS中创建用户目录。本教程全部采用hadoop用户登录Linux系统，因此，需要在HDFS中为hadoop用户创建一个用户目录，命令如下：

1. cd /usr/local/hadoop
2. ./bin/hdfs dfs –mkdir –p /user/hadoop

Shell 命令

该命令中表示在HDFS中创建一个“/user/hadoop”目录，“–mkdir”是创建目录的操作，“-p”表示如果是多级目录，则父目录和子目录一起创建，这里“/user/hadoop”就是一个多级目录，因此必须使用参数“-p”，否则会出错。  
“/user/hadoop”目录就成为hadoop用户对应的用户目录，可以使用如下命令显示HDFS中与当前用户hadoop对应的用户目录下的内容：

1. ./bin/hdfs dfs –ls .

Shell 命令

该命令中，“-ls”表示列出HDFS某个目录下的所有内容，“.”表示HDFS中的当前用户目录，也就是“/user/hadoop”目录，因此，上面的命令和下面的命令是等价的：

1. ./bin/hdfs dfs –ls /user/hadoop

Shell 命令

如果要列出HDFS上的所有目录，可以使用如下命令：

1. ./bin/hdfs dfs –ls

Shell 命令

下面，可以使用如下命令创建一个input目录：

1. ./bin/hdfs dfs –mkdir input

Shell 命令

在创建个input目录时，采用了相对路径形式，实际上，这个input目录创建成功以后，它在HDFS中的完整路径是“/user/hadoop/input”。如果要在HDFS的根目录下创建一个名称为input的目录，则需要使用如下命令：

1. ./bin/hdfs dfs –mkdir /input

Shell 命令

可以使用rm命令删除一个目录，比如，可以使用如下命令删除刚才在HDFS中创建的“/input”目录（不是“/user/hadoop/input”目录）：

1. ./bin/hdfs dfs –rm –r /input

Shell 命令

上面命令中，“-r”参数表示如果删除“/input”目录及其子目录下的所有内容，如果要删除的一个目录包含了子目录，则必须使用“-r”参数，否则会执行失败。

### 2.文件操作

在实际应用中，经常需要从本地文件系统向HDFS中上传文件，或者把HDFS中的文件下载到本地文件系统中。  
首先，使用vim编辑器，在本地Linux文件系统的“/home/hadoop/”目录下创建一个文件myLocalFile.txt，里面可以随意输入一些单词，比如，输入如下三行：

Hadoop

Spark

XMU DBLAB

然后，可以使用如下命令把本地文件系统的“/home/hadoop/myLocalFile.txt”上传到HDFS中的当前用户目录的input目录下，也就是上传到HDFS的“/user/hadoop/input/”目录下：

1. ./bin/hdfs dfs -put /home/hadoop/myLocalFile.txt input

Shell 命令

可以使用ls命令查看一下文件是否成功上传到HDFS中，具体如下：

1. ./bin/hdfs dfs –ls input

Shell 命令

该命令执行后会显示类似如下的信息：

Found 1 items

-rw-r--r-- 1 hadoop supergroup 36 2017-01-02 23:55 input/ myLocalFile.txt

下面使用如下命令查看HDFS中的myLocalFile.txt这个文件的内容：

1. ./bin/hdfs dfs –cat input/myLocalFile.txt

Shell 命令

下面把HDFS中的myLocalFile.txt文件下载到本地文件系统中的“/home/hadoop/下载/”这个目录下，命令如下：

1. ./bin/hdfs dfs -get input/myLocalFile.txt /home/hadoop/下载

Shell 命令

可以使用如下命令，到本地文件系统查看下载下来的文件myLocalFile.txt：

1. $ cd ~
2. $ cd 下载
3. $ ls
4. $ cat myLocalFile.txt

Shell 命令

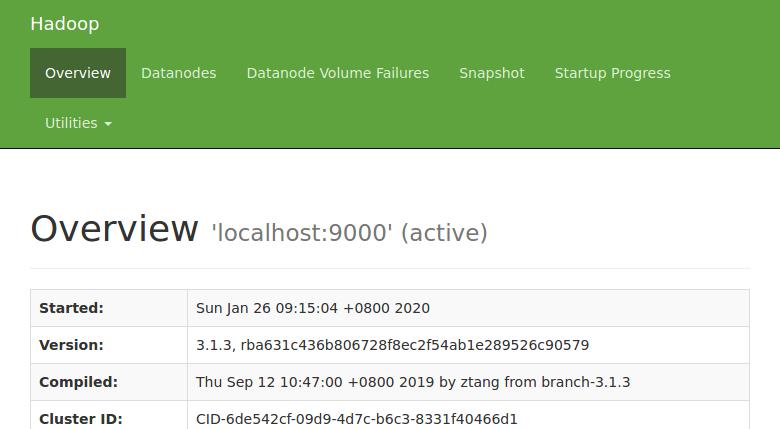
最后，了解一下如何把文件从HDFS中的一个目录拷贝到HDFS中的另外一个目录。比如，如果要把HDFS的“/user/hadoop/input/myLocalFile.txt”文件，拷贝到HDFS的另外一个目录“/input”中（注意，这个input目录位于HDFS根目录下），可以使用如下命令：

1. ./bin/hdfs dfs -cp input/myLocalFile.txt /input

Shell 命令

## 二、利用Web界面管理HDFS

打开Linux自带的Firefox浏览器，点击此链接[HDFS的Web界面](http://localhost:9870/)，即可看到HDFS的web管理界面。WEB界面的访问地址是http://localhost:9870。



## 三、利用Java API与HDFS进行交互

Hadoop不同的文件系统之间通过调用Java API进行交互，上面介绍的Shell命令，本质上就是Java API的应用。下面提供了Hadoop官方的Hadoop API文档，想要深入学习Hadoop，可以访问如下网站，查看各个API的功能。

[Hadoop API文档](http://hadoop.apache.org/docs/stable/api/)

利用Java API进行交互，需要利用软件Eclipse编写Java程序。

### (一) 在Ubuntu中安装Eclipse

Eclipse是常用的程序开发工具，本教程很多程序代码都是使用Eclipse开发调试，因此，需要在Linux系统中安装Eclipse。  
可以到Eclipse官网（<https://www.eclipse.org/downloads/>）下载安装包。  
或者直接[点击这里从百度云盘下载软件](https://pan.baidu.com/s/1gbmPBXrJDCxwqPGkfvX5Xg" \t "_blank)（提取码：lnwl），位于“软件”目录下，文件名是eclipse-4.7.0-linux.gtk.x86\_64.tar.gz。假设安装文件下载后保存在了Linux系统的目录“~/Downloads”下，下面执行如下命令对文件进行解压缩：

1. cd ~/Downloads
2. sudo tar -zxvf ./eclipse-4.7.0-linux.gtk.x86\_64.tar.gz -C /usr/local

Shell 命令

然后，执行如下命令启动Eclipse：

1. cd /usr/local/eclipse
2. ./eclipse

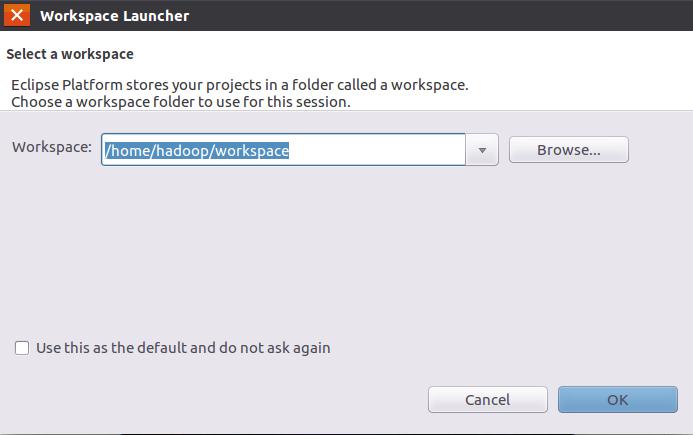
Shell 命令

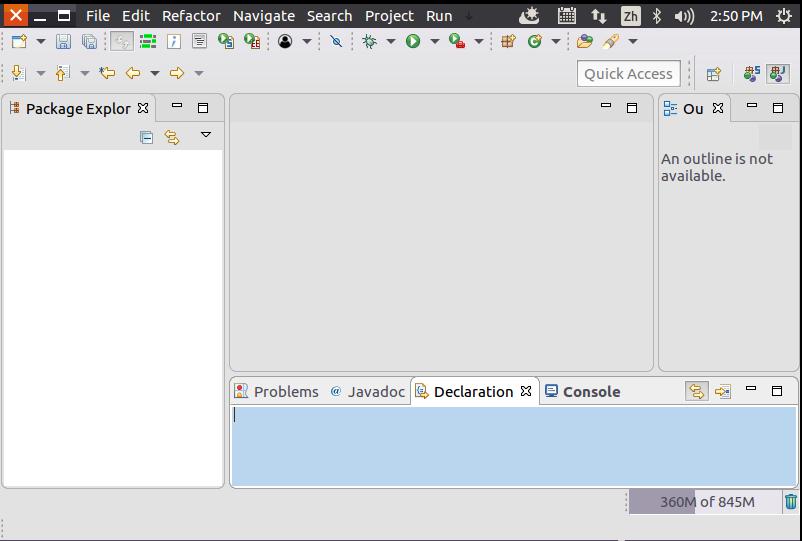
这时，就可以看到Eclipse的启动界面了。

### （二）使用Eclipse开发调试HDFS Java程序

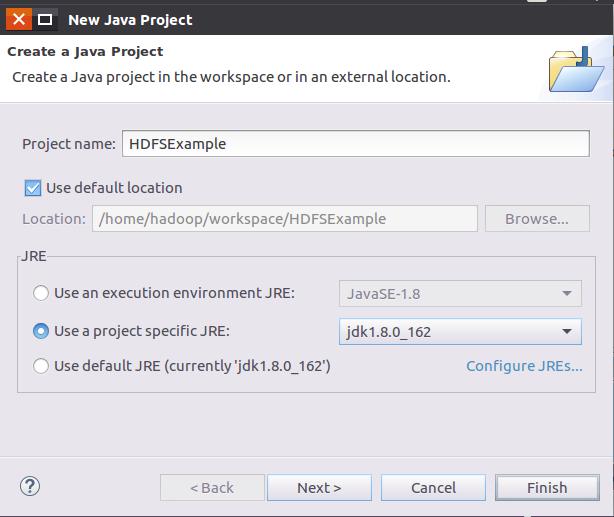
Hadoop采用Java语言开发的，提供了Java API与HDFS进行交互。上面介绍的Shell命令，在执行时实际上会被系统转换成Java API调用。Hadoop官方网站提供了完整的Hadoop API文档（http://hadoop.apache.org/docs/stable/api/），想要深入学习Hadoop编程，可以访问Hadoop官网查看各个API的功能和用法。本教程只介绍基础的HDFS编程。  
为了提高程序编写和调试效率，本教程采用Eclipse工具编写Java程序。  
现在要执行的任务是：假设在目录“hdfs://localhost:9000/user/hadoop”下面有几个文件，分别是file1.txt、file2.txt、file3.txt、file4.abc和file5.abc，这里需要从该目录中过滤出所有后缀名不为“.abc”的文件，对过滤之后的文件进行读取，并将这些文件的内容合并到文件“hdfs://localhost:9000/user/hadoop/merge.txt”中。

#### 1. 在Eclipse中创建项目

启动Eclipse。当Eclipse启动以后，会弹出如下图所示界面，提示设置工作空间（workspace）。  


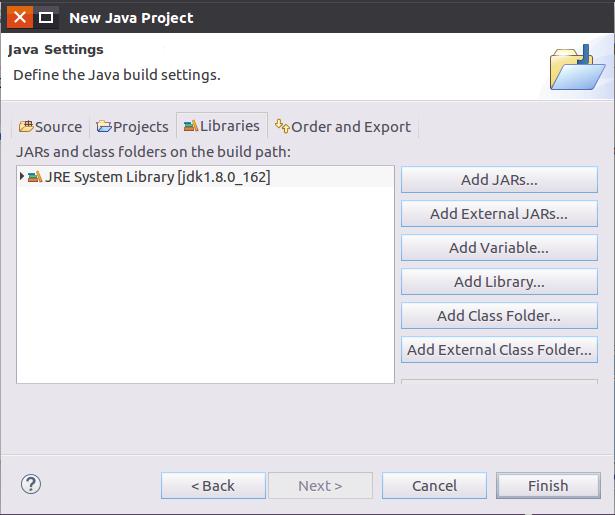
可以直接采用默认的设置“/home/hadoop/workspace”，点击“OK”按钮。可以看出，由于当前是采用hadoop用户登录了Linux系统，因此，默认的工作空间目录位于hadoop用户目录“/home/hadoop”下。  
Eclipse启动以后，会呈现如下图所示的界面。  


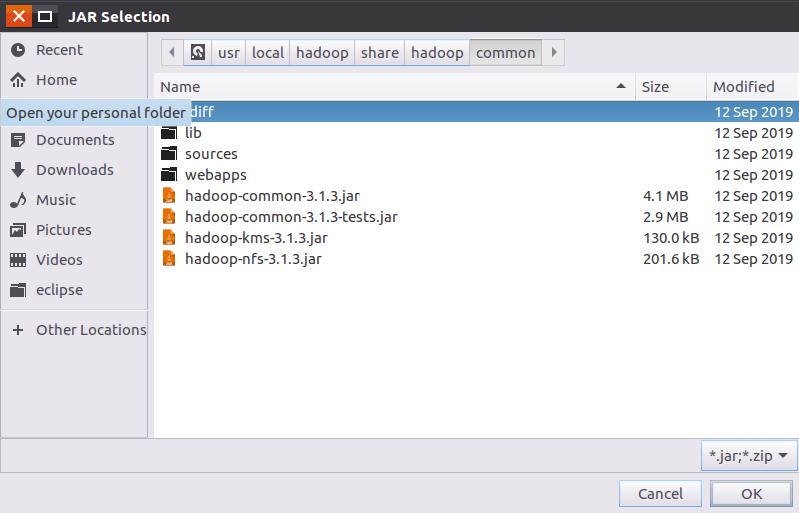
选择“File–>New–>Java Project”菜单，开始创建一个Java工程，会弹出如下图所示界面。

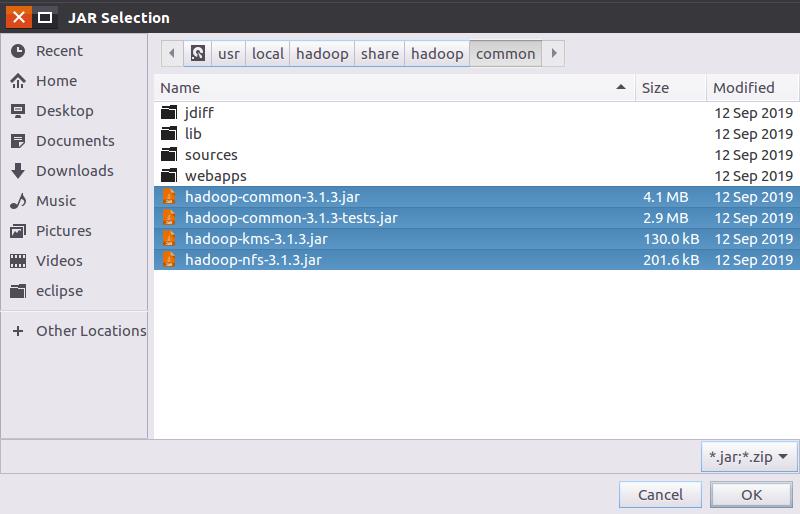


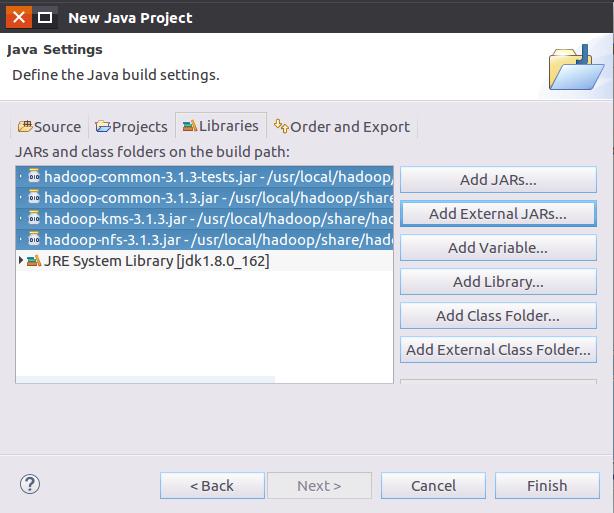
在“Project name”后面输入工程名称“HDFSExample”，选中“Use default location”，让这个Java工程的所有文件都保存到“/home/hadoop/workspace/HDFSExample”目录下。在“JRE”这个选项卡中，可以选择当前的Linux系统中已经安装好的JDK，比如jdk1.8.0\_162。然后，点击界面底部的“Next>”按钮，进入下一步的设置。

#### 2. 为项目添加需要用到的JAR包

进入下一步的设置以后，会弹出如下图所示界面。  


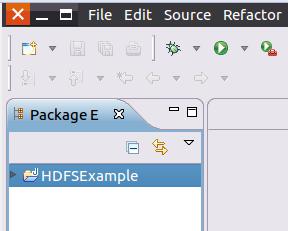
需要在这个界面中加载该Java工程所需要用到的JAR包，这些JAR包中包含了可以访问HDFS的Java API。这些JAR包都位于Linux系统的Hadoop安装目录下，对于本教程而言，就是在“/usr/local/hadoop/share/hadoop”目录下。点击界面中的“Libraries”选项卡，然后，点击界面右侧的“Add External JARs…”按钮，会弹出如下图所示界面。  


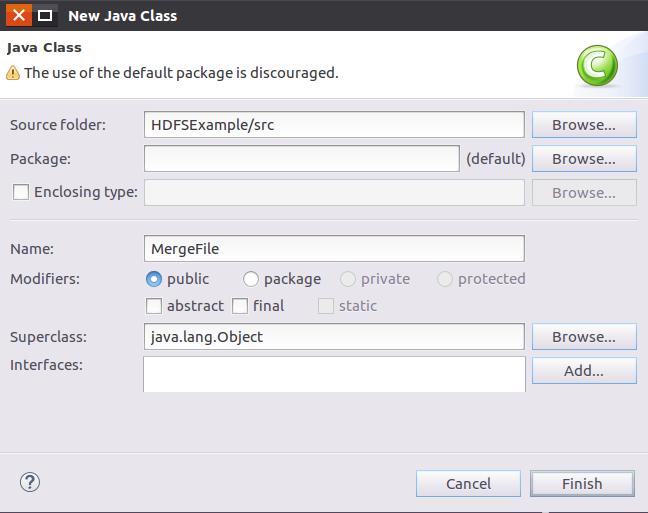
在该界面中，上面的一排目录按钮（即“usr”、“local”、“hadoop”、“share”、“hadoop”和“common”），当点击某个目录按钮时，就会在下面列出该目录的内容。  
为了编写一个能够与HDFS交互的Java应用程序，一般需要向Java工程中添加以下JAR包：  
（1）“/usr/local/hadoop/share/hadoop/common”目录下的所有JAR包，包括hadoop-common-3.1.3.jar、hadoop-common-3.1.3-tests.jar、haoop-nfs-3.1.3.jar和haoop-kms-3.1.3.jar，注意，不包括目录jdiff、lib、sources和webapps；  
（2）“/usr/local/hadoop/share/hadoop/common/lib”目录下的所有JAR包；  
（3）“/usr/local/hadoop/share/hadoop/hdfs”目录下的所有JAR包，注意，不包括目录jdiff、lib、sources和webapps；  
（4）“/usr/local/hadoop/share/hadoop/hdfs/lib”目录下的所有JAR包。  
比如，如果要把“/usr/local/hadoop/share/hadoop/common”目录下的hadoop-common-3.1.3.jar、hadoop-common-3.1.3-tests.jar、haoop-nfs-3.1.3.jar和haoop-kms-3.1.3.jar添加到当前的Java工程中，可以在界面中点击目录按钮，进入到common目录，然后，界面会显示出common目录下的所有内容（如下图所示）。  


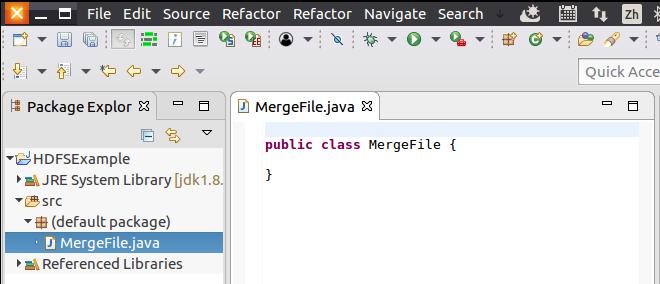
请在界面中用鼠标点击选中hadoop-common-3.1.3.jar、hadoop-common-3.1.3-tests.jar、haoop-nfs-3.1.3.jar和haoop-kms-3.1.3.jar（不要选中目录jdiff、lib、sources和webapps），然后点击界面右下角的“确定”按钮，就可以把这两个JAR包增加到当前Java工程中，出现的界面如下图所示。  


从这个界面中可以看出，hadoop-common-3.1.3.jar、hadoop-common-3.1.3-tests.jar、haoop-nfs-3.1.3.jar和haoop-kms-3.1.3.jar已经被添加到当前Java工程中。然后，按照类似的操作方法，可以再次点击“Add External JARs…”按钮，把剩余的其他JAR包都添加进来。需要注意的是，当需要选中某个目录下的所有JAR包时，可以使用“Ctrl+A”组合键进行全选操作。全部添加完毕以后，就可以点击界面右下角的“Finish”按钮，完成Java工程HDFSExample的创建。

#### 3. 编写Java应用程序

下面编写一个Java应用程序。  
请在Eclipse工作界面左侧的“Package Explorer”面板中（如下图所示），找到刚才创建好的工程名称“HDFSExample”，然后在该工程名称上点击鼠标右键，在弹出的菜单中选择“New–>Class”菜单。  


选择“New–>Class”菜单以后会出现如下图所示界面。  


在该界面中，只需要在“Name”后面输入新建的Java类文件的名称，这里采用名称“MergeFile”，其他都可以采用默认设置，然后，点击界面右下角“Finish”按钮，出现如下图所示界面。  


可以看出，Eclipse自动创建了一个名为“MergeFile.java”的源代码文件，请在该文件中输入以下代码：

1. import java.io.IOException;
2. import java.io.PrintStream;
3. import java.net.URI;
5. import org.apache.hadoop.conf.Configuration;
6. import org.apache.hadoop.fs.\*;
8. */\*\**
9. *\* 过滤掉文件名满足特定条件的文件*
10. *\*/*
11. class MyPathFilter implements PathFilter {
12. String reg = null;
13. MyPathFilter(String reg) {
14. this.reg = reg;
15. }
16. public boolean accept(Path path) {
17. if (!(path.toString().matches(reg)))
18. return true;
19. return false;
20. }
21. }
22. */\*\*\**
23. *\* 利用FSDataOutputStream和FSDataInputStream合并HDFS中的文件*
24. *\*/*
25. public class MergeFile {
26. Path inputPath = null; *//待合并的文件所在的目录的路径*
27. Path outputPath = null; *//输出文件的路径*
28. public MergeFile(String input, String output) {
29. this.inputPath = new Path(input);
30. this.outputPath = new Path(output);
31. }
32. public void doMerge() throws IOException {
33. Configuration conf = new Configuration();
34. conf.set("fs.defaultFS","hdfs://localhost:9000");
35. conf.set("fs.hdfs.impl","org.apache.hadoop.hdfs.DistributedFileSystem");
36. FileSystem fsSource = FileSystem.get(URI.create(inputPath.toString()), conf);
37. FileSystem fsDst = FileSystem.get(URI.create(outputPath.toString()), conf);
38. *//下面过滤掉输入目录中后缀为.abc的文件*
39. FileStatus[] sourceStatus = fsSource.listStatus(inputPath,
40. new MyPathFilter(".\*\\.abc"));
41. FSDataOutputStream fsdos = fsDst.create(outputPath);
42. PrintStream ps = new PrintStream(System.out);
43. *//下面分别读取过滤之后的每个文件的内容，并输出到同一个文件中*
44. for (FileStatus sta : sourceStatus) {
45. *//下面打印后缀不为.abc的文件的路径、文件大小*
46. System.out.print("路径：" + sta.getPath() + " 文件大小：" + sta.getLen()
47. + " 权限：" + sta.getPermission() + " 内容：");
48. FSDataInputStream fsdis = fsSource.open(sta.getPath());
49. byte[] data = new byte[1024];
50. int read = -1;
52. while ((read = fsdis.read(data)) > 0) {
53. ps.write(data, 0, read);
54. fsdos.write(data, 0, read);
55. }
56. fsdis.close();
57. }
58. ps.close();
59. fsdos.close();
60. }
61. public static void main(String[] args) throws IOException {
62. MergeFile merge = new MergeFile(
63. "hdfs://localhost:9000/user/hadoop/",
64. "hdfs://localhost:9000/user/hadoop/merge.txt");
65. merge.doMerge();
66. }
67. }

Java

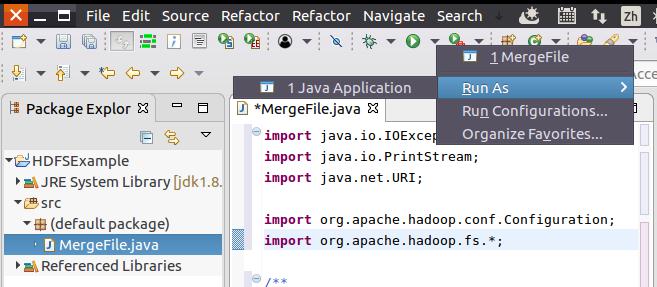
#### 4. 编译运行程序

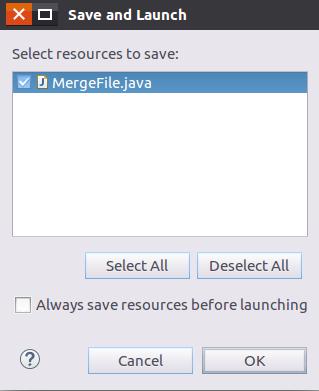
在开始编译运行程序之前，请一定确保Hadoop已经启动运行，如果还没有启动，需要打开一个Linux终端，输入以下命令启动Hadoop：

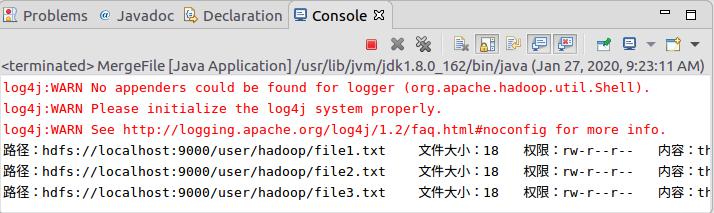
1. cd /usr/local/hadoop
2. ./sbin/start-dfs.sh

Shell 命令

然后，要确保HDFS的“/user/hadoop”目录下已经存在file1.txt、file2.txt、file3.txt、file4.abc和file5.abc，每个文件里面有内容。这里，假设文件内容如下：  
file1.txt的内容是： this is file1.txt  
file2.txt的内容是： this is file2.txt  
file3.txt的内容是： this is file3.txt  
file4.abc的内容是： this is file4.abc  
file5.abc的内容是： this is file5.abc

现在就可以编译运行上面编写的代码。可以直接点击Eclipse工作界面上部的运行程序的快捷按钮，当把鼠标移动到该按钮上时，在弹出的菜单中选择“Run As”，继续在弹出来的菜单中选择“Java Application”，如下图所示。  


然后，会弹出如下图所示界面。  


在该界面中，点击界面右下角的“OK”按钮，开始运行程序。程序运行结束后，会在底部的“Console”面板中显示运行结果信息（如下图所示）。同时，“Console”面板中还会显示一些类似“log4j:WARN…”的警告信息，可以不用理会。  


如果程序运行成功，这时，可以到HDFS中查看生成的merge.txt文件，比如，可以在Linux终端中执行如下命令：

1. cd /usr/local/hadoop
2. ./bin/hdfs dfs -ls /user/hadoop
3. ./bin/hdfs dfs -cat /user/hadoop/merge.txt

Shell 命令

可以看到如下结果：

this is file1.txt

this is file2.txt

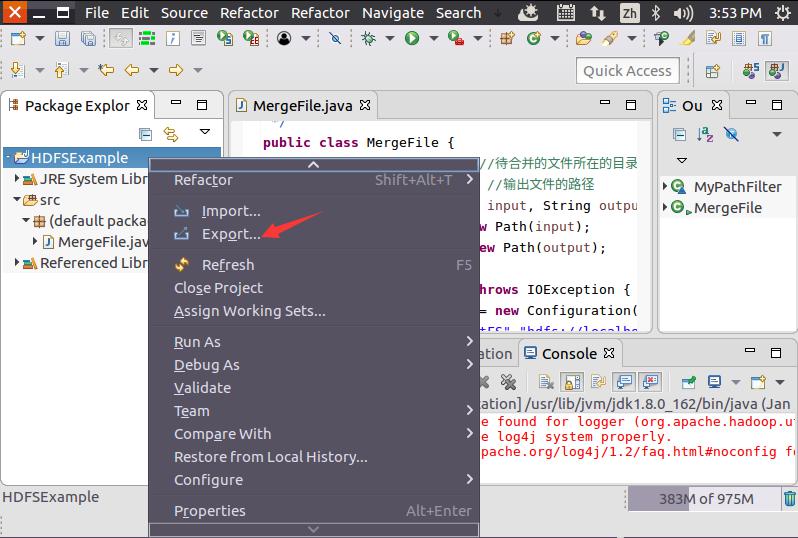
this is file3.txt

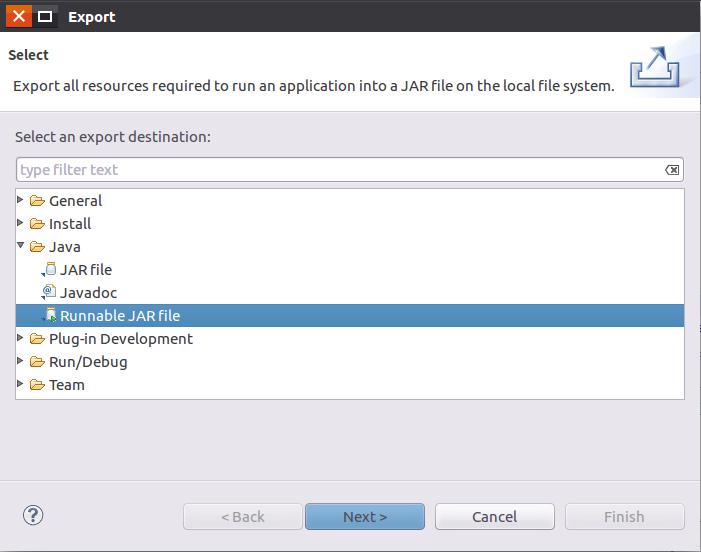
#### 5. 应用程序的部署

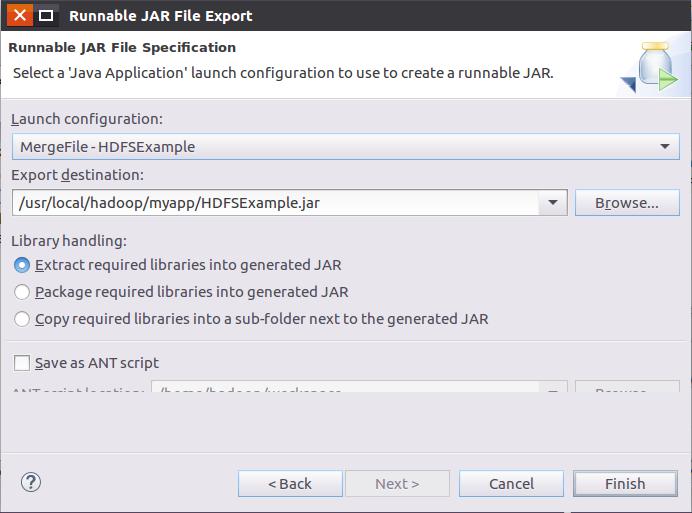
下面介绍如何把Java应用程序生成JAR包，部署到Hadoop平台上运行。首先，在Hadoop安装目录下新建一个名称为myapp的目录，用来存放我们自己编写的Hadoop应用程序，可以在Linux的终端中执行如下命令：

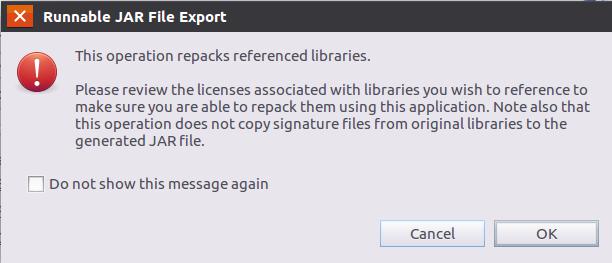
1. cd /usr/local/hadoop
2. mkdir myapp

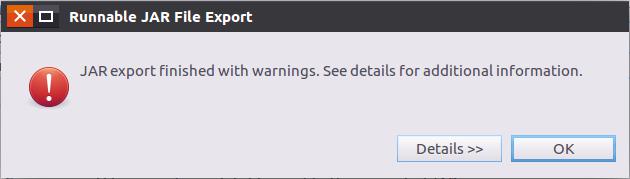
Shell 命令

然后，请在Eclipse工作界面左侧的“Package Explorer”面板中，在工程名称“HDFSExample”上点击鼠标右键，在弹出的菜单中选择“Export”，如下图所示。  


然后，会弹出如下图所示界面。  


在该界面中，选择“Runnable JAR file”，然后，点击“Next>”按钮，弹出如下图所示界面。  


在该界面中，“Launch configuration”用于设置生成的JAR包被部署启动时运行的主类，需要在下拉列表中选择刚才配置的类“MergeFile-HDFSExample”。在“Export destination”中需要设置JAR包要输出保存到哪个目录，比如，这里设置为“/usr/local/hadoop/myapp/HDFSExample.jar”。在“Library handling”下面选择“Extract required libraries into generated JAR”。然后，点击“Finish”按钮，会出现如下图所示界面。  


可以忽略该界面的信息，直接点击界面右下角的“OK”按钮，启动打包过程。打包过程结束后，会出现一个警告信息界面，如下图所示。  


可以忽略该界面的信息，直接点击界面右下角的“OK”按钮。至此，已经顺利把HDFSExample工程打包生成了HDFSExample.jar。可以到Linux系统中查看一下生成的HDFSExample.jar文件，可以在Linux的终端中执行如下命令：

1. cd /usr/local/hadoop/myapp
2. ls

Shell 命令

可以看到，“/usr/local/hadoop/myapp”目录下已经存在一个HDFSExample.jar文件。  
由于之前已经运行过一次程序，已经生成了merge.txt，因此，需要首先执行如下命令删除该文件：

1. cd /usr/local/hadoop
2. ./bin/hdfs dfs -rm /user/hadoop/merge.txt

Shell 命令

现在，就可以在Linux系统中，使用hadoop jar命令运行程序，命令如下：

1. cd /usr/local/hadoop
2. ./bin/hadoop jar ./myapp/HDFSExample.jar

Shell 命令

上面程序执行结束以后，可以到HDFS中查看生成的merge.txt文件，比如，可以在Linux终端中执行如下命令：

1. cd /usr/local/hadoop
2. ./bin/hdfs dfs -ls /user/hadoop
3. ./bin/hdfs dfs -cat /user/hadoop/merge.txt

Shell 命令

可以看到如下结果：

this is file1.txt

this is file2.txt

this is file3.txt

## 附录：自己练习用的代码文件

下面给出几个代码文件，供读者自己练习。

### 1.写入文件

1. import org.apache.hadoop.conf.Configuration;
2. import org.apache.hadoop.fs.FileSystem;
3. import org.apache.hadoop.fs.FSDataOutputStream;
4. import org.apache.hadoop.fs.Path;
6. public class Chapter3 {
7. public static void main(String[] args) {
8. try {
9. Configuration conf = new Configuration();
10. conf.set("fs.defaultFS","hdfs://localhost:9000");
11. conf.set("fs.hdfs.impl","org.apache.hadoop.hdfs.DistributedFileSystem");
12. FileSystem fs = FileSystem.get(conf);
13. byte[] buff = "Hello world".getBytes(); *// 要写入的内容*
14. String filename = "test"; *//要写入的文件名*
15. FSDataOutputStream os = fs.create(new Path(filename));
16. os.write(buff,0,buff.length);
17. System.out.println("Create:"+ filename);
18. os.close();
19. fs.close();
20. } catch (Exception e) {
21. e.printStackTrace();
22. }
23. }
24. }

Java

### 2.判断文件是否存在

1. import org.apache.hadoop.conf.Configuration;
2. import org.apache.hadoop.fs.FileSystem;
3. import org.apache.hadoop.fs.Path;
5. public class Chapter3 {
6. public static void main(String[] args) {
7. try {
8. String filename = "test";
10. Configuration conf = new Configuration();
11. conf.set("fs.defaultFS","hdfs://localhost:9000");
12. conf.set("fs.hdfs.impl","org.apache.hadoop.hdfs.DistributedFileSystem");
13. FileSystem fs = FileSystem.get(conf);
14. if(fs.exists(new Path(filename))){
15. System.out.println("文件存在");
16. }else{
17. System.out.println("文件不存在");
18. }
19. fs.close();
20. } catch (Exception e) {
21. e.printStackTrace();
22. }
23. }
24. }

Java

### 3.读取文件

1. import java.io.BufferedReader;
2. import java.io.InputStreamReader;
4. import org.apache.hadoop.conf.Configuration;
5. import org.apache.hadoop.fs.FileSystem;
6. import org.apache.hadoop.fs.Path;
7. import org.apache.hadoop.fs.FSDataInputStream;
9. public class Chapter3 {
10. public static void main(String[] args) {
11. try {
12. Configuration conf = new Configuration();
13. conf.set("fs.defaultFS","hdfs://localhost:9000");
14. conf.set("fs.hdfs.impl","org.apache.hadoop.hdfs.DistributedFileSystem");
15. FileSystem fs = FileSystem.get(conf);
16. Path file = new Path("test");
17. FSDataInputStream getIt = fs.open(file);
18. BufferedReader d = new BufferedReader(new InputStreamReader(getIt));
19. String content = d.readLine(); *//读取文件一行*
20. System.out.println(content);
21. d.close(); *//关闭文件*
22. fs.close(); *//关闭hdfs*
23. } catch (Exception e) {
24. e.printStackTrace();
25. }
26. }
27. }

Java

本文作者



[**林子雨老师**](http://dblab.xmu.edu.cn/blog/author/ziyulin/)

致力于打造中国高校首个“数字教师”

[www.cs.xmu.edu.cn/linziyu](http://www.cs.xmu.edu.cn/linziyu)[ziyulin](http://weibo.com/ziyulin)nc.ude.umx@niluyiz

<http://dblab.xmu.edu.cn/blog/2460-2/> [大数据](http://dblab.xmu.edu.cn/blog/category/big-data/)