

### 基于 Eclipse (HDFSExample) 的 MapReduce 实验完整流程

#### 一、环境确认 (Hadoop 2.7.7 + Eclipse)

#### 二、步骤 1: 创建 MapReduce 类 (文件合并去重)

(1) MergeDedupMapper.java (Mapper 类)

(2) MergeDedupReducer.java (Reducer 类)

(3) MergeDedupDriver.java (Driver 类)

#### 三、步骤 2: 创建 MapReduce 类 (祖孙关系挖掘)

(1) GrandparentMapper.java (Mapper 类)

(2) GrandparentReducer.java (Reducer 类)

(3) GrandparentDriver.java (Driver 类)

#### 四、步骤 3: 打包项目为 JAR 文件

#### 五、步骤 4: Linux 终端执行作业

(一) 启动 Hadoop 集群

(二) 创建本地输入文件 (bigdata 账户)

(三) 上传输入文件到 HDFS

(四) 执行 “文件合并去重” 作业

(五) 执行 “祖孙关系挖掘” 作业

---

# 基于 Eclipse (HDFSExample) 的 MapReduce 实验完整流程

本文档结合 Eclipse 项目结构（`HDFSExample` 项目、Hadoop 2.7.7 依赖、`com.aa.hadoop.mapreduce` 包）和 Hadoop 实验要求，提供从代码编写、项目打包、本地文件创建到集群执行的完整步骤。

## 一、环境确认 (Hadoop 2.7.7 + Eclipse)

从你的依赖包（`hadoop-common-2.7.7.jar` 等）可知，你使用的是 Hadoop 2.7.7 版本。在开始前，请确保：

- **Linux 环境：** Hadoop 2.7.7 已正确安装且环境变量配置正确（如 `HADOOP_HOME=/usr/local/hadoop`）。
- **Eclipse 环境：** 已通过手动引入 Hadoop 依赖包（Add External JARs）完成项目配置，因为你的依赖是本地路径而非 Maven 管理。

## 二、步骤 1：创建 MapReduce 类（文件合并去重）

在 Eclipse 的 `Project Explorer` 中，展开 `HDFSExample` → `src` → `com.aa.hadoop.mapreduce` 包，右键创建以下 3 个 Java 类。

## (1) MergeDedupMapper.java (Mapper 类)

```
1 package com.aa.hadoop.mapreduce;
2
3 import org.apache.hadoop.io.LongWritable;
4 import org.apache.hadoop.io.Text;
5 import org.apache.hadoop.mapreduce.Mapper;
6 import java.io.IOException;
7
8 /**
9  * 功能: 读取fileA和fileB的每行数据, 输出“行内容-固定值1”
10  */
11 public class MergeDedupMapper extends Mapper<LongWritable, Text, Text,
12     Text> {
13     private static final Text FIXED_VALUE = new Text("1");
14
15     @Override
16     protected void map(LongWritable key, Text value, Context context)
17         throws IOException, InterruptedException {
18         String line = value.toString().trim();
19         if (!line.isEmpty()) {
20             context.write(new Text(line), FIXED_VALUE);
21         }
22     }
23 }
```

## (2) MergeDedupReducer.java (Reducer 类)

```
1 package com.aa.hadoop.mapreduce;
2
3 import org.apache.hadoop.io.Text;
4 import org.apache.hadoop.mapreduce.Reducer;
5 import java.io.IOException;
6
7 /**
8  * 功能：相同Key只输出一次，实现合并去重
9  */
10 public class MergeDedupReducer extends Reducer<Text, Text, Text, Text>
11 {
12     @Override
13     protected void reduce(Text key, Iterable<Text> values, Context
14         context)
15         throws IOException, InterruptedException {
16         // 无论values有多少个"1", 只输出一次key, 实现去重
17         context.write(key, new Text(""));
18     }
19 }
```

### (3) MergeDedupDriver.java (Driver 类)

```
1 package com.aa.hadoop.mapreduce;
2
3 import org.apache.hadoop.conf.Configuration;
4 import org.apache.hadoop.fs.Path;
5 import org.apache.hadoop.io.Text;
6 import org.apache.hadoop.mapreduce.Job;
7 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
8 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
9
10 /**
11  * 功能: 配置并提交MapReduce作业
12  */
13 public class MergeDedupDriver {
14
15     public static void main(String[] args) throws Exception {
16         Configuration conf = new Configuration();
17         Job job = Job.getInstance(conf, "MergeDedupJob");
18
19         job.setJarByClass(MergeDedupDriver.class);
20         job.setMapperClass(MergeDedupMapper.class);
21         job.setReducerClass(MergeDedupReducer.class);
22
23         job.setMapOutputKeyClass(Text.class);
24         job.setMapOutputValueClass(Text.class);
25
26         job.setOutputKeyClass(Text.class);
27         job.setOutputValueClass(Text.class);
28
29         // main方法的参数args[0]为输入路径, args[1]为输出路径
30         FileInputFormat.addInputPath(job, new Path(args[0]));
31         FileOutputFormat.setOutputPath(job, new Path(args[1]));
32
33         System.exit(job.waitForCompletion(true) ? 0 : 1);
34     }
35 }
```

### 三、步骤 2: 创建 MapReduce 类 (祖孙关系挖掘)

继续在 `com.aa.hadoop.mapreduce` 包下创建以下 3 个类。

## (1) GrandparentMapper.java (Mapper 类)

```
1 package com.aa.hadoop.mapreduce;
2
3 import org.apache.hadoop.io.LongWritable;
4 import org.apache.hadoop.io.Text;
5 import org.apache.hadoop.mapreduce.Mapper;
6 import java.io.IOException;
7
8 /**
9  * 功能: 处理child-parent表格, 输出标记父子关系的键值对
10  * 思想:
11  * 1. (parent, "P-" + child) -> 以parent为Key, 标记他有一个孩子(P)
12  * 2. (child, "C-" + parent) -> 以child为Key, 标记他有一个父亲(C)
13  */
14 public class GrandparentMapper extends Mapper<LongWritable, Text,
15     Text, Text> {
16
17     @Override
18     protected void map(LongWritable key, Text value, Context context)
19         throws IOException, InterruptedException {
20         String line = value.toString().trim();
21         // 跳过空行或表头
22         if (line.isEmpty() || line.startsWith("child")) {
23             return;
24         }
25
26         String[] parts = line.split("\\s+"); // 使用一个或多个空格分割
27         if (parts.length == 2) {
28             String child = parts[0];
29             String parent = parts[1];
30
31             // 输出: Key=parent, Value="P-child" (P代表Parent)
32             context.write(new Text(parent), new Text("P-" + child));
33             // 输出: Key=child, Value="C-parent" (C代表Child)
34             context.write(new Text(child), new Text("C-" + parent));
35         }
36     }
37 }
```

## (2) GrandparentReducer.java (Reducer 类)



```

1 package com.aa.hadoop.mapreduce;
2
3 import org.apache.hadoop.io.Text;
4 import org.apache.hadoop.mapreduce.Reducer;
5 import java.io.IOException;
6 import java.util.ArrayList;
7 import java.util.List;
8
9 /**
10  * 功能: 推导祖孙关系
11  * 思想:
12  * Reducer会收到以同一个人为Key的所有Value。
13  * Key=Tom, Values=["P-Steven", "P-Philip", "C-Alice"]
14  * 这意味着Tom是Steven和Philip的父亲 (P), 也是Alice的/儿子 (C)。
15  * 将P列表 (孩子) 和C列表 (父亲) 分离。
16  * 遍历P列表 (孙子) 和C列表 (爷爷/奶奶), 组合输出。
17  */
18 public class GrandparentReducer extends Reducer<Text, Text, Text,
19 Text> {
20     @Override
21     protected void reduce(Text key, Iterable<Text> values, Context
22 context)
23         throws IOException, InterruptedException {
24         List<String> children = new ArrayList<>(); // 存放当前Key的孩子
25         List<String> parents = new ArrayList<>(); // 存放当前Key的父母
26         (即孙辈)
27         (即祖辈)
28         for (Text value : values) {
29             String val = value.toString();
30             if (val.startsWith("P-")) {
31                 children.add(val.substring(2)); // 添加孩子名
32             } else if (val.startsWith("C-")) {
33                 parents.add(val.substring(2)); // 添加父母名
34             }
35         }
36         // 如果这个人既有孩子 (P), 又有父母 (C), 则可以建立祖孙关系
37         if (!children.isEmpty() && !parents.isEmpty()) {
38             for (String grandchild : children) {
39                 for (String grandparent : parents) {
40                     // 输出: Key=孙子, Value=祖父/祖母
41                     context.write(new Text(grandchild), new
42 Text(grandparent));
43                 }
44             }
45         }
46     }
47 }

```

```
44     }
45 }
46 }
```

### (3) GrandparentDriver.java (Driver 类)

```
1  package com.aa.hadoop.mapreduce;
2
3  import org.apache.hadoop.conf.Configuration;
4  import org.apache.hadoop.fs.Path;
5  import org.apache.hadoop.io.Text;
6  import org.apache.hadoop.mapreduce.Job;
7  import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
8  import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
9
10 /**
11  * 功能：提交祖孙关系挖掘作业
12  */
13 public class GrandparentDriver {
14
15     public static void main(String[] args) throws Exception {
16         Configuration conf = new Configuration();
17         Job job = Job.getInstance(conf, "GrandparentJob");
18
19         job.setJarByClass(GrandparentDriver.class);
20         job.setMapperClass(GrandparentMapper.class);
21         job.setReducerClass(GrandparentReducer.class);
22
23         job.setMapOutputKeyClass(Text.class);
24         job.setMapOutputValueClass(Text.class);
25
26         job.setOutputKeyClass(Text.class);
27         job.setOutputValueClass(Text.class);
28
29         FileInputFormat.addInputPath(job, new Path(args[0]));
30         FileOutputFormat.setOutputPath(job, new Path(args[1]));
31
32         System.exit(job.waitForCompletion(true) ? 0 : 1);
33     }
34 }
```

## 四、步骤 3：打包项目为 JAR 文件

在 Eclipse 中，将整个 `HDFSExample` 项目打包成一个 JAR 文件。

1. 在 `Project Explorer` 中，右键点击 `HDFSExample` 项目。

2. 选择 **Export...** → **Java** → **JAR file** → 点击 **Next**。
3. 在 **JAR file:** 处，指定 JAR 文件的输出路径和名称（例如：`/usr/local/hadoop/myapp/HDFSExample.jar`）。
4. 确保 **src** 目录被选中。
5. 点击 **Next** → **Next**（跳过 JAR Packaging Options）。
6. 在 **JAR Manifest Specification** 界面：
  - **推荐：不选择 Main class。**我们将在 `hadoop jar` 命令中显式指定要运行的 Driver 类，这样同一个 JAR 包就可以灵活执行两个不同的作业。
7. 点击 **Finish**，完成 JAR 打包。

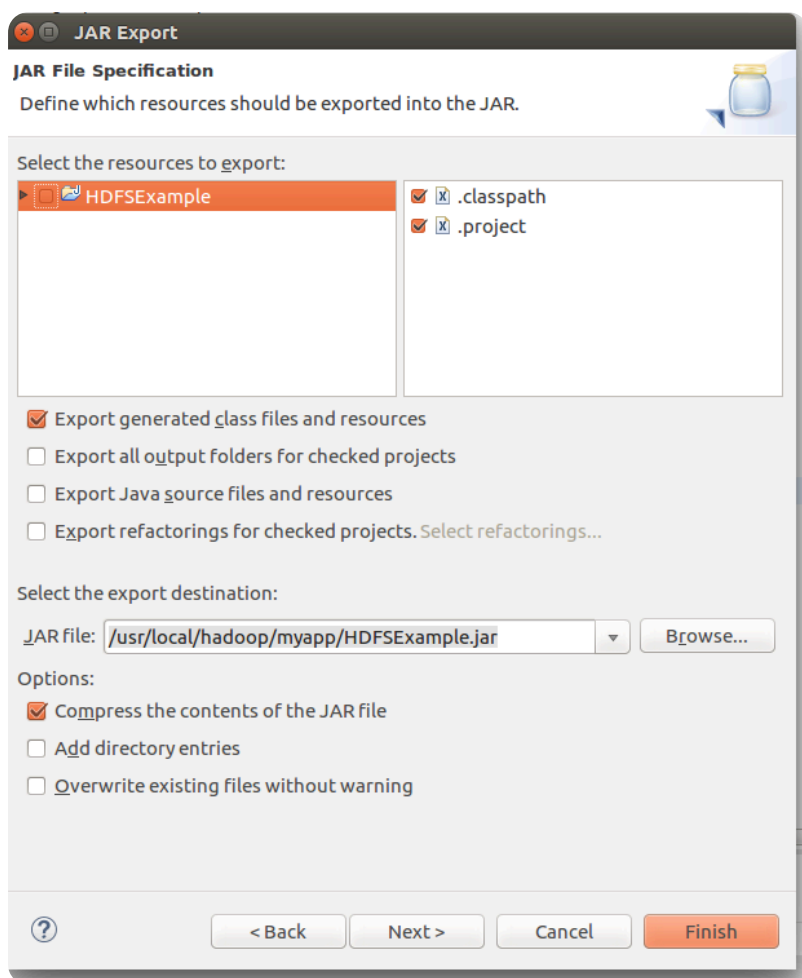


图 | image-20251031112338936

## 五、步骤 4：Linux 终端执行作业

打开 Linux 终端，登录到你的 `bigdata` 账户。

## (一) 启动 Hadoop 集群

1. 切换到 Hadoop 安装目录（根据你的环境调整）：

```
1 | cd /usr/local/hadoop
```

2. （可选）先停止，确保环境干净：

```
1 | sbin/stop-dfs.sh
2 | sbin/stop-yarn.sh
```

3. 启动 HDFS 和 YARN：

```
1 | sbin/start-dfs.sh
2 | sbin/start-yarn.sh
```

4. 验证集群进程是否正常启动：

```
1 | jps
```

你应该能看到 `NameNode` , `DataNode` , `ResourceManager` , `NodeManager` , `SecondaryNameNode` 至少这 5 个进程。

## (二) 创建本地输入文件 (bigdata 账户)

执行以下命令，在本地创建实验所需的输入文件。

1. 创建本地文件存储目录：

```
1 | mkdir -p /home/bigdata/lab7/input
```

2. 创建 `fileA.txt`：

```
1 | cat > /home/bigdata/lab7/input/fileA.txt << EOF
2 | 20150101 x
3 | 20150102 y
4 | 20150103 x
5 | 20150104 y
6 | 20150105 z
7 | 20150106 x
8 | EOF
```

3. 创建 `fileB.txt`：

```
1 cat > /home/bigdata/lab7/input/fileB.txt << EOF
2 20150101 y
3 20150102 y
4 20150103 x
5 20150104 z
6 20150105 y
7 EOF
```

4. 创建 `child_parent.txt` :

```
1 cat > /home/bigdata/lab7/input/child_parent.txt << EOF
2 child parent
3 Steven Lucy
4 Steven Jack
5 Lucy Mary
6 Jone Lucy
7 Jone Jack
8 Lucy Frank
9 Jack Alice
10 Jack Jesse
11 David Alice
12 David Jesse
13 Philip David
14 Philip Alma
15 Mark David
16 Mark Alma
17 EOF
```

5. 验证文件内容（可选）：

```
1 cat /home/bigdata/lab7/input/fileA.txt
2 cat /home/bigdata/lab7/input/fileB.txt
3 cat /home/bigdata/lab7/input/child_parent.txt
```

```

bigdata@ubuntu:~/lab7$ cd input
bigdata@ubuntu:~/lab7/input$ ls
child_parent.txt  fileA.txt  fileB.txt
bigdata@ubuntu:~/lab7/input$ cat /home/bigdata/lab7/input/fileA.txt
20150101 x
20150102 y
20150103 x
20150104 y
20150105 z
20150106 x
bigdata@ubuntu:~/lab7/input$ cat /home/bigdata/lab7/input/fileB.txt
20150101 y
20150102 y
20150103 x
20150104 z
20150105 y
bigdata@ubuntu:~/lab7/input$ cat /home/bigdata/lab7/input/child_parent.txt
child parent
Steven Lucy
Steven Jack
Lucy Mary
Jone Lucy
Jone Jack
Lucy Frank
Jack Alice
Jack Jesse
David Alice
David Jesse
Philip David
Philip Alma
Mark David
Mark Alma
bigdata@ubuntu:~/lab7/input$

```

图 2 image-20251031111712655

### (三) 上传输入文件到 HDFS

1. 为两个作业创建 HDFS 输入目录（使用与本地匹配的 `lab7` 路径）：

```

1 # 为合并去重任务创建目录
2 hdfs dfs -mkdir -p /user/bigdata/lab7/merge_input
3
4 # 为祖孙关系挖掘任务创建目录
5 hdfs dfs -mkdir -p /user/bigdata/lab7/grandparent_input

```

2. 上传文件到 HDFS 对应目录：

```

1 # 上传 fileA、fileB 到合并去重目录
2 hdfs dfs -put /home/bigdata/lab7/input/fileA.txt
  /home/bigdata/lab7/input/fileB.txt /user/bigdata/lab7/merge_input
3
4 # 上传 child_parent.txt 到祖孙挖掘目录
5 hdfs dfs -put /home/bigdata/lab7/input/child_parent.txt
  /user/bigdata/lab7/grandparent_input

```

3. 验证 HDFS 文件上传结果：

```

1  hdfs dfs -ls /user/bigdata/lab7/merge_input
2  # 应能看到 fileA.txt 和 fileB.txt
3
4  hdfs dfs -ls /user/bigdata/lab7/grandparent_input
5  # 应能看到 child_parent.txt

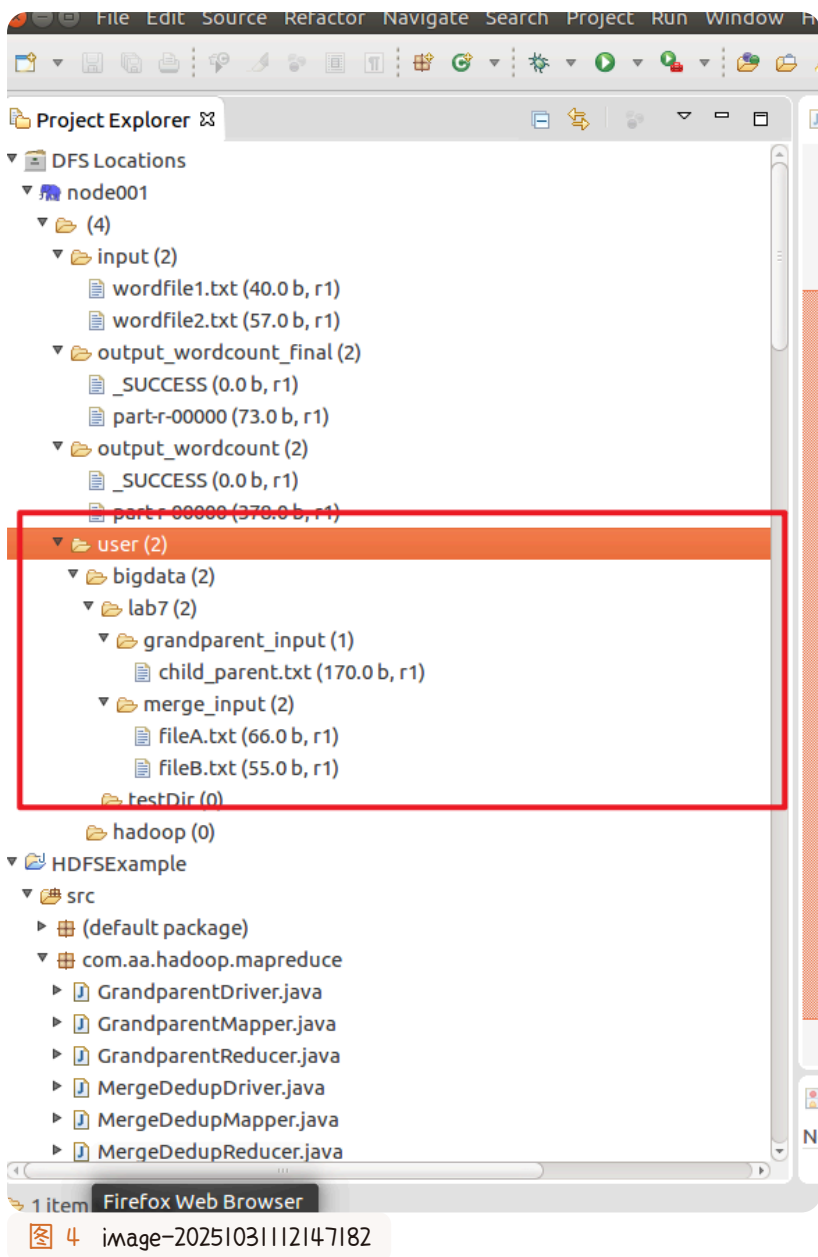
```

```

bigdata@ubuntu:~/lab7/input$ hdfs dfs -mkdir -p /user/bigdata/lab7/merge_input
bigdata@ubuntu:~/lab7/input$ hdfs dfs -mkdir -p /user/bigdata/lab7/grandparent_input
bigdata@ubuntu:~/lab7/input$ hdfs dfs -put /home/bigdata/lab7/input/fileA.txt /user/bigdata/lab7/merge_input
bigdata@ubuntu:~/lab7/input$ hdfs dfs -put /home/bigdata/lab7/input/fileB.txt /user/bigdata/lab7/merge_input
bigdata@ubuntu:~/lab7/input$ hdfs dfs -put /home/bigdata/lab7/input/child_parent.txt /user/bigdata/lab7/grandparent_input
25/10/30 20:18:48 WARN hdfs.DFSClient: Caught exception
java.lang.InterruptedOperationException
  at java.lang.Object.wait(Native Method)
  at java.lang.Thread.join(Thread.java:1252)
  at java.lang.Thread.join(Thread.java:1326)
  at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DFSOutputStream.java:716)
  at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.endBlock(DFSOutputStream.java:476)
  at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.java:652)
bigdata@ubuntu:~/lab7/input$ hdfs dfs -put /home/bigdata/lab7/input/child_parent.txt /user/bigdata/lab7/grandparent_input
put: '/user/bigdata/lab7/grandparent_input/child_parent.txt': File exists
bigdata@ubuntu:~/lab7/input$ hdfs dfs -ls /user/bigdata/lab7/merge_input
Found 2 items
-rw-r--r-- 1 bigdata supergroup 66 2025-10-30 20:18 /user/bigdata/lab7/merge_input/fileA.txt
-rw-r--r-- 1 bigdata supergroup 55 2025-10-30 20:18 /user/bigdata/lab7/merge_input/fileB.txt
bigdata@ubuntu:~/lab7/input$ hdfs dfs -ls /user/bigdata/lab7/grandparent_input
Found 1 items
-rw-r--r-- 1 bigdata supergroup 170 2025-10-30 20:18 /user/bigdata/lab7/grandparent_input/child_parent.txt
bigda Firefox Web Browser lab7/input$

```

图 3 image-20251031112120491



#### (四) 执行“文件合并去重”作业

使用 `hadoop jar` 命令提交作业。

```
1 hadoop jar /usr/local/hadoop/myapp/HDFSExample.jar \  
2 com.aa.hadoop.mapreduce.MergeDedupDriver \  
3 /user/bigdata/lab7/merge_input \  
4 /user/bigdata/lab7/merge_output
```

##### 命令解析：

- `/usr/local/hadoop/myapp/HDFSExample.jar`：你打包的 JAR 文件路径。
- `com.aa.hadoop.mapreduce.MergeDedupDriver`：要运行的主类全名。
- `/user/bigdata/lab7/merge_input`：HDFS 输入目录。
- `/user/bigdata/lab7/merge_output`：HDFS 输出目录（此目录必须不存在）。



- 查看结果（预期输出合并去重后的 6 行）：

```
1 # 等待作业执行完毕后
2 hdfs dfs -cat /user/bigdata/lab7/merge_output/part-r-00000
```

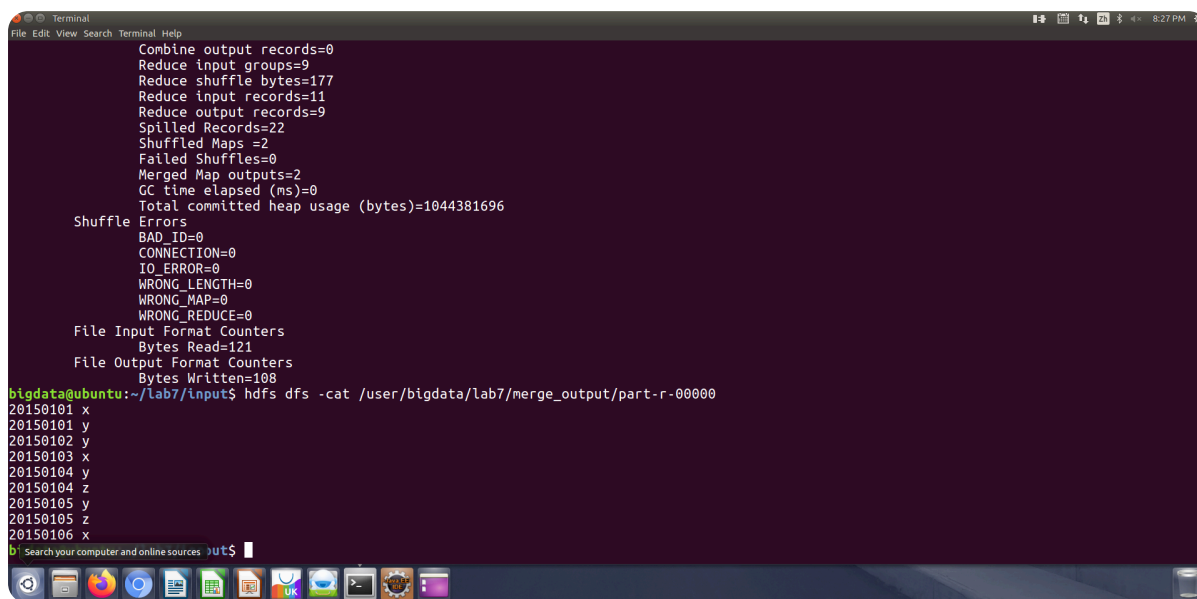


图 5 image-20251031112815417

与预期结果相同

## （五）执行“祖孙关系挖掘”作业

使用同一个 JAR 包，只需更改主类名和输入输出路径。

```
1 hadoop jar /usr/local/hadoop/myapp/HDFSExample.jar \  
2 com.aa.hadoop.mapreduce.GrandparentDriver \  
3 /user/bigdata/lab7/grandparent_input \  
4 /user/bigdata/lab7/grandparent_output
```

- 命令解析：

- `/usr/local/hadoop/myapp/HDFSExample.jar`：还是用那个 JAR 包。
- `com.aa.hadoop.mapreduce.GrandparentDriver`：指定运行祖孙关系的主类。
- `/user/bigdata/lab7/grandparent_input`：HDFS 输入目录。
- `/user/bigdata/lab7/grandparent_output`：HDFS 输出目录（必须不存在）。

- 查看结果：

```
1 # 等待作业执行完毕后
2 hdfs dfs -cat /user/bigdata/lab7/grandparent_output/part-r-00000
```

```
Terminal
File Edit View Search Terminal Help

Reduce input records=28
Reduce output records=12
Spilled Records=56
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=0
Total committed heap usage (bytes)=565182464

Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0

File Input Format Counters
Bytes Read=170
File Output Format Counters
Bytes Written=142
bigdata@ubuntu:~/lab7/input$ hdfs dfs -cat /user/bigdata/lab7/grandparent_output/part-r-00000
Mark      Jesse
Mark      Alice
Philip    Jesse
Philip    Alice
Steven    Jesse
Steven    Alice
Jone      Jesse
Jone      Alice
Steven    Frank
Steven    Mary
Jone      Frank
Jone      Mary
bigdata@ubuntu:~/lab7/input$
```

图 6 image-2025103113031282

与预期结果相同