实验七

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)

```java
package com.aa.hadoop.mapreduce;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import java.io.IOException;

/**
 * 功能：读取fileA和fileB的每行数据，输出“行内容-固定值1”
 */
public class MergeDedupMapper extends Mapper<LongWritable, Text, Text,
Text> {

    private static final Text FIXED_VALUE = new Text("1");

    @Override
    protected void map(LongWritable key, Text value, Context context)
            throws IOException, InterruptedException {
        String line = value.toString().trim();
        if (!line.isEmpty()) {
            context.write(new Text(line), FIXED_VALUE);
        }
    }
}
```

（2）MergeDedupReducer.java（Reducer 类）

```java
package com.aa.hadoop.mapreduce;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import java.io.IOException;

/**
 * 功能：相同Key只输出一次，实现合并去重
 */
public class MergeDedupReducer extends Reducer<Text, Text, Text, Text>
{

    @Override
    protected void reduce(Text key, Iterable<Text> values, Context
context)
            throws IOException, InterruptedException {
        // 无论values有多少个"1"，只输出一次key，实现去重
        context.write(key, new Text(""));
    }
}
```

（3）MergeDedupDriver.java（Driver 类）

```java
package com.aa.hadoop.mapreduce;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/**
 * 功能：配置并提交MapReduce作业
 */
public class MergeDedupDriver {

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "MergeDedupJob");

        job.setJarByClass(MergeDedupDriver.class);
        job.setMapperClass(MergeDedupMapper.class);
        job.setReducerClass(MergeDedupReducer.class);

        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(Text.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);

        // main方法的参数args[0]为输入路径，args[1]为输出路径
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

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)

```java
package com.aa.hadoop.mapreduce;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import java.io.IOException;

/**
 * 功能：处理child-parent表格，输出标记父子关系的键值对
 * 思想：
 * 1. (parent, "P-" + child) -> 以parent为Key，标记他有一个孩子(P)
 * 2. (child, "C-" + parent) -> 以child为Key，标记他有一个父亲(C)
 */
public class GrandparentMapper extends Mapper<LongWritable, Text,
Text, Text> {

    @Override
    protected void map(LongWritable key, Text value, Context context)
            throws IOException, InterruptedException {
        String line = value.toString().trim();
        // 跳过空行或表头
        if (line.isEmpty() || line.startsWith("child")) {
            return;
        }

        String[] parts = line.split("\\s+"); // 使用一个或多个空格分割
        if (parts.length == 2) {
            String child = parts[0];
            String parent = parts[1];

            // 输出：Key=parent, Value="P-child" (P代表Parent)
            context.write(new Text(parent), new Text("P-" + child));
            // 输出：Key=child, Value="C-parent" (C代表Child)
            context.write(new Text(child), new Text("C-" + parent));
        }
    }
}
```

（2）GrandparentReducer.java（Reducer 类）

```java
package com.aa.hadoop.mapreduce;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

/**
 * 功能：推导祖孙关系
 * 思想：
 * Reducer会收到以同一个人为Key的所有Value。
 * Key=Tom, Values=["P-Steven", "P-Philip", "C-Alice"]
 * 这意味着Tom是Steven和Philip的父亲 (P)，也是Alice的儿子 (C)。
 * 将P列表（孩子）和C列表（父亲）分离。
 * 遍历P列表（孙子）和C列表（爷爷/奶奶），组合输出。
 */
public class GrandparentReducer extends Reducer<Text, Text, Text,
Text> {

    @Override
    protected void reduce(Text key, Iterable<Text> values, Context
context)
            throws IOException, InterruptedException {

        List<String> children = new ArrayList<>(); // 存放当前Key的孩子
(即孙辈)
        List<String> parents = new ArrayList<>();  // 存放当前Key的父母
(即祖辈)

        for (Text value : values) {
            String val = value.toString();
            if (val.startsWith("P-")) {
                children.add(val.substring(2)); // 添加孩子名
            } else if (val.startsWith("C-")) {
                parents.add(val.substring(2));  // 添加父母名
            }
        }

        // 如果这个人既有孩子（P），又有父母（C），则可以建立祖孙关系
        if (!children.isEmpty() && !parents.isEmpty()) {
            for (String grandchild : children) {
                for (String grandparent : parents) {
                    // 输出：Key=孙子, Value=祖父/祖母
                    context.write(new Text(grandchild), new
Text(grandparent));
                }
            }
```

```
        }
    }
}
```

（3）GrandparentDriver.java（Driver 类）

```java
package com.aa.hadoop.mapreduce;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/**
 * 功能：提交祖孙关系挖掘作业
 */
public class GrandparentDriver {

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "GrandparentJob");

        job.setJarByClass(GrandparentDriver.class);
        job.setMapperClass(GrandparentMapper.class);
        job.setReducerClass(GrandparentReducer.class);

        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(Text.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

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。

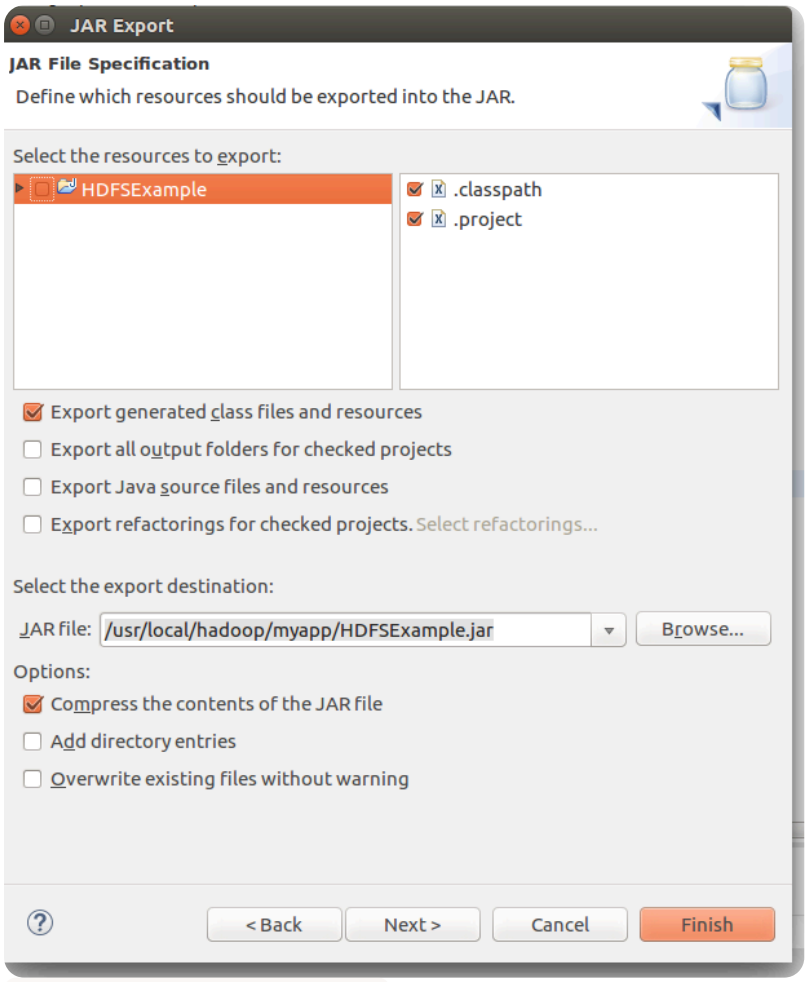

图 | image-20251031112338936

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）：

```
cd /usr/local/hadoop
```

2. （可选）先停止，确保环境干净：

```
sbin/stop-dfs.sh
sbin/stop-yarn.sh
```

3. 启动 HDFS 和 YARN：

```
sbin/start-dfs.sh
sbin/start-yarn.sh
```

4. 验证集群进程是否正常启动：

```
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：

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

2. 创建 `fileA.txt`：

```
cat > /home/bigdata/lab7/input/fileA.txt << EOF
20150101 x
20150102 y
20150103 x
20150104 y
20150105 z
20150106 x
EOF
```

3. 创建 `fileB.txt`：

```
cat > /home/bigdata/lab7/input/fileB.txt << EOF
20150101 y
20150102 y
20150103 x
20150104 z
20150105 y
EOF
```

4. 创建 `child_parent.txt`：

```
cat > /home/bigdata/lab7/input/child_parent.txt << EOF
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
EOF
```

5. 验证文件内容（可选）：

```
cat /home/bigdata/lab7/input/fileA.txt
cat /home/bigdata/lab7/input/fileB.txt
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）：

```
# 为合并去重任务创建目录
hdfs dfs -mkdir -p /user/bigdata/lab7/merge_input

# 为祖孙关系挖掘任务创建目录
hdfs dfs -mkdir -p /user/bigdata/lab7/grandparent_input
```

2. 上传文件到 HDFS 对应目录：

```
# 上传 fileA、fileB 到合并去重目录
hdfs dfs -put /home/bigdata/lab7/input/fileA.txt
/home/bigdata/lab7/input/fileB.txt /user/bigdata/lab7/merge_input

# 上传 child_parent.txt 到祖孙挖掘目录
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：

```
hdfs dfs -ls /user/bigdata/lab7/merge_input
# 应能看到 fileA.txt 和 fileB.txt

hdfs dfs -ls /user/bigdata/lab7/grandparent_input
# 应能看到 child_parent.txt
```

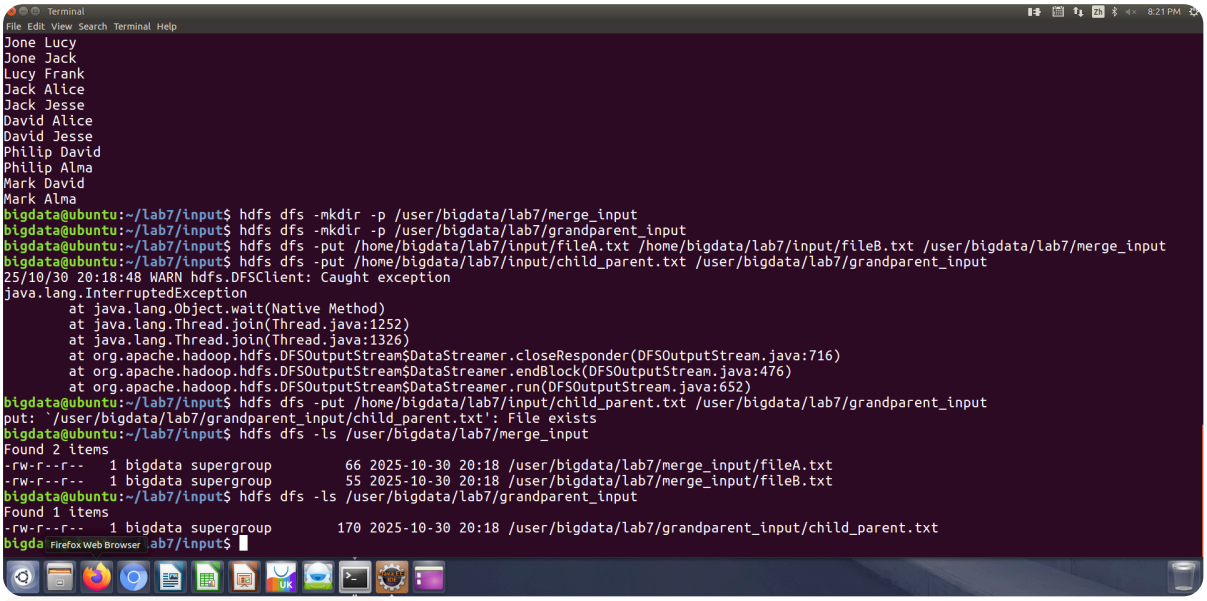

图 3 image-20251031112120491

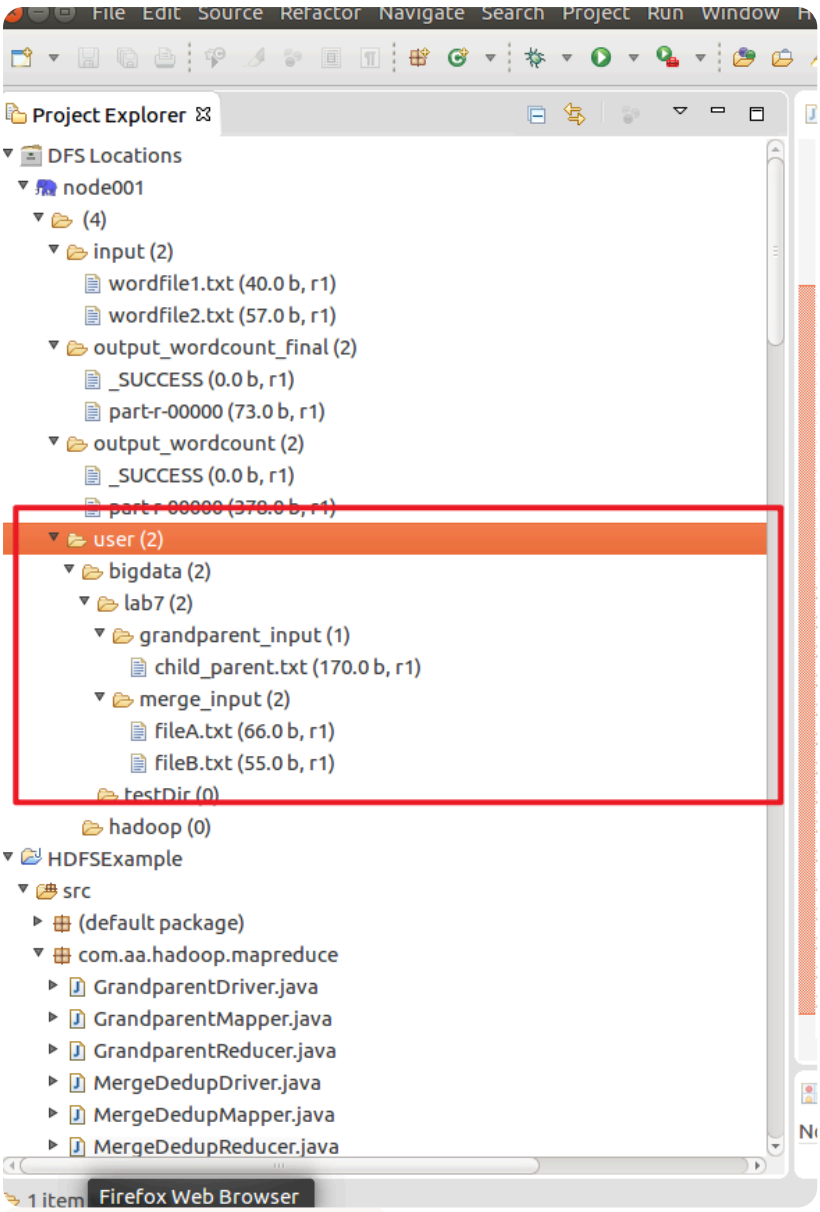

图 4 image-20251031112147182

（四）执行 “文件合并去重” 作业

使用 `hadoop jar` 命令提交作业。

```
hadoop jar /usr/local/hadoop/myapp/HDFSExample.jar \
com.aa.hadoop.mapreduce.MergeDedupDriver \
/user/bigdata/lab7/merge_input \
/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）：

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

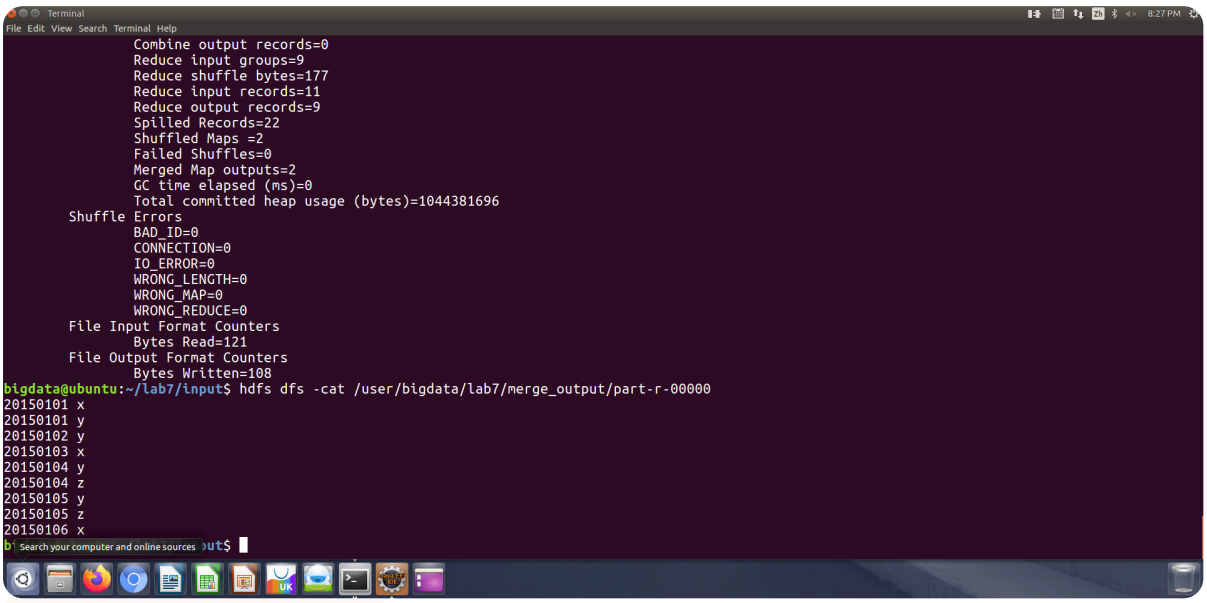

图 5 image-20251031112815417

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。

```
hadoop jar /usr/local/hadoop/myapp/HDFSExample.jar \
com.aa.hadoop.mapreduce.GrandparentDriver \
/user/bigdata/lab7/grandparent_input \
/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**：

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

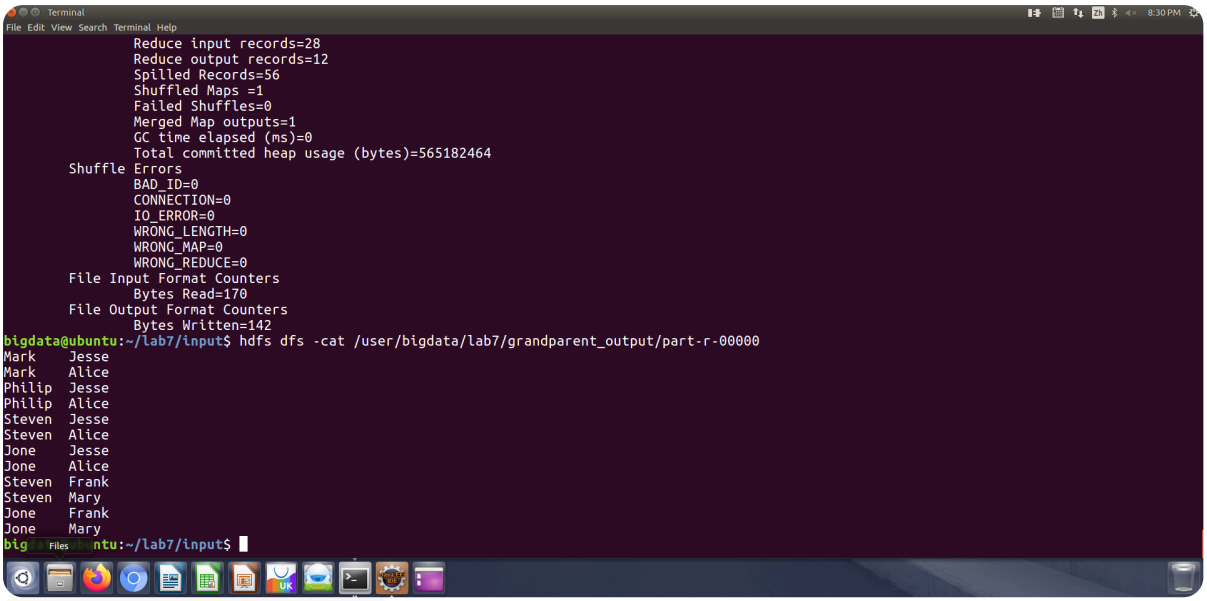

图 6 image-20251031113031282

与预期结果相同

Powered by VLOOK™ (V2025.10.1)