

MapReduce 操作 HBase 数据

实验环境（普开实验“数据库技术(NoSQL)”对应章节虚拟机）

虚拟机数量：3

系统版本：Centos 7.5

Hadoop 版本：Apache Hadoop 2.7.3

Zookeeper 版本：Apache Zookeeper 3.4.10

Eclipse 版本：Neon.3 4.6.3

HBase 版本：Apache HBase 1.2.6

一、 HBase 编程：读取 HBase 表数据

1. 实验描述(参照普开实验“数据库技术(NoSQL)”第 15 章)

使用 MapReduce 批量将 HBase 表中数据导入到 HDFS 上，学习本实验将掌握通过 MapReduce 编程来连接并读取 HBase 数据表，学习 HBase 数据库在分布式计算中的应用。

2. 主要步骤：

修改主机名：主机名格式：学号+自己姓名简称（未修改扣分）

```
[zkpk@master ~]$ su root
Password: 密码: zkpk

[root@master zkpk]# vim /etc/hostname
[root@master zkpk]# hostname 2020110808zs
[root@master zkpk]# bash
[root@2020110808zs zkpk]# su zkpk
```

/etc/hostname 文件内容如下



- 启动 Hadoop 集群

在 master 运行：

```
[zkpk@2020110808zs ~]$ start-all.sh
```

- 启动 Zookeeper 集群

需要在 master、slave01、slave02 分别运行：

```
[zkpk@2020110808zs ~]$ zkServer.sh start
```

- 启动 HBase 集群

在 master 运行：

```
[zkpk@2020110808zs ~]$ start-hbase.sh
```

- 进入 HBase Shell 创建实验用表

输入 hbase shell 进入 hbase 交互式环境：

```
[zkpk@2020110808zs ~]$ hbase shell
```

数据库表格设计要求：（未按要求设计扣分）

- a) 表格命名：学号+姓名（如截图 1）
- b) 列族数量：至少两列
- c) 某一系列下至少有两个子列
- d) 行数不限定，字段名不限定
- e) ROW 命名：学号+姓名+编号（如截图 1）

【截图 1：数据库表格】（截图需要包含标记信息，未按要求扣分）

```

hbase(main):016:0> scan '2018211582lzy'
ROW                                COLUMN+CELL
2018211582lzy001                  column=cf1:age, timestamp=1617018362239, value=19
2018211582lzy001                  column=cf1:gender, timestamp=1617018423268, value=man
2018211582lzy001                  column=cf1:name, timestamp=1617018409106, value=lizhiyi
2018211582lzy001                  column=cf2:height, timestamp=1617018479196, value=178
2018211582lzy001                  column=cf2:weight, timestamp=1617018488260, value=70
2018211582lzy002                  column=cf1:age, timestamp=1617018551632, value=20
2018211582lzy002                  column=cf1:gender, timestamp=1617018603656, value=woman
2018211582lzy002                  column=cf1:name, timestamp=1617018573701, value=hexing
2018211582lzy002                  column=cf2:height, timestamp=1617018614523, value=160
2018211582lzy002                  column=cf2:weight, timestamp=1617018624161, value=60
2 row(s) in 0.0220 seconds

```

说明: 这里我的设置的表名为 2018211582lzy, 列族数量两个, 分别为 cf1 和 cf2, 其中 cf1 列族包含 3 个子列, 分别为 age、gender 和 name, cf2 列族包含 2 个子列, 分别为 weight 和 height, 这里我填充了部分数据, 如图所展示。

● 编写代码

【截图 2: 完整 Mapper 代码】

(截图需要包含标记信息, 未按要求扣分, 代码不完整扣分)

MemberMapper.java:

```

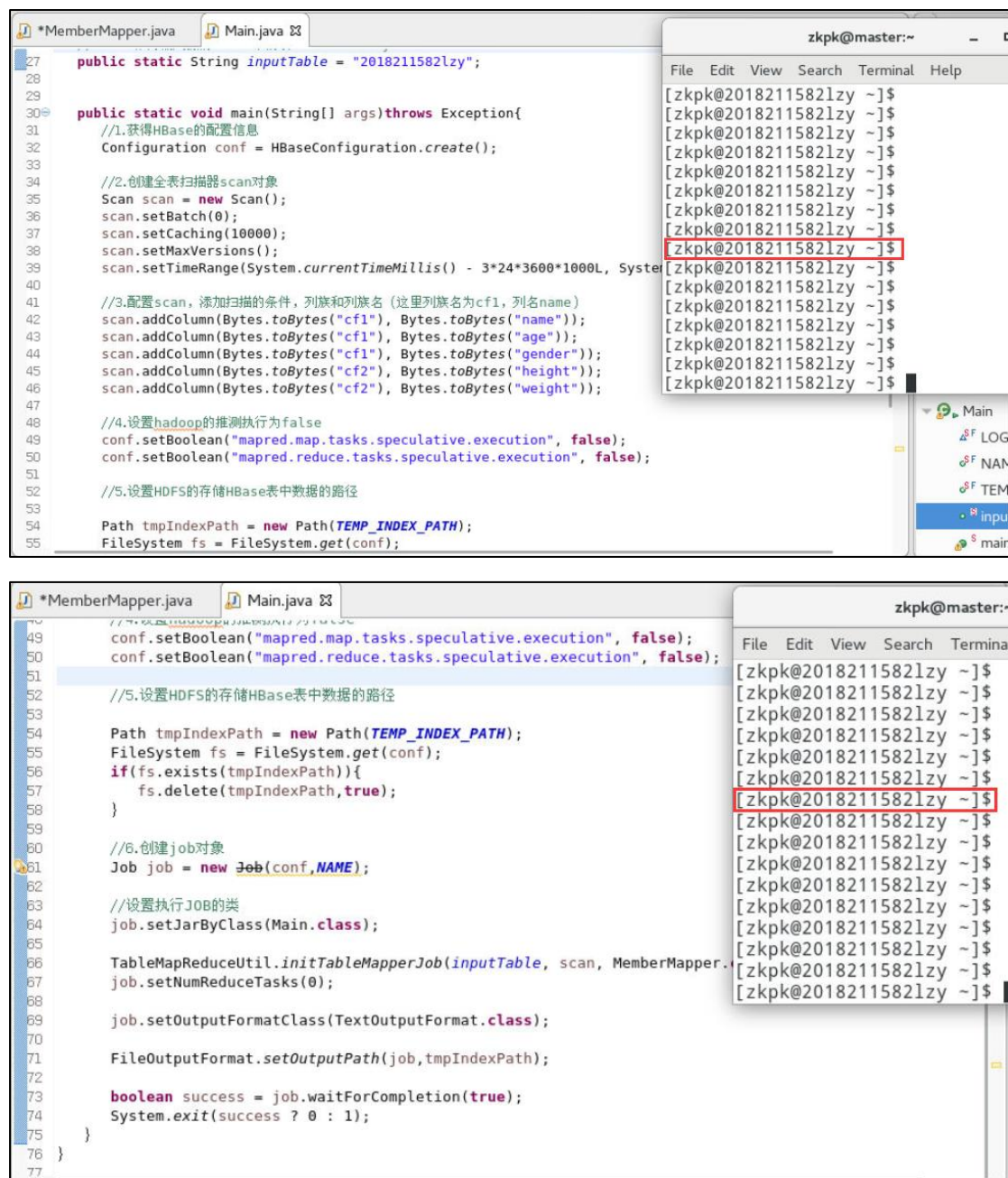
MemberMapper.java  Main.java
1 package org.zkpk.hbase.inputSource;
2 import java.io.IOException;
3 import org.apache.hadoop.hbase.KeyValue;
4 import org.apache.hadoop.hbase.client.Result;
5 import org.apache.hadoop.hbase.io.ImmutableBytesWritable;
6 import org.apache.hadoop.hbase.mapreduce.TableMapper;
7 import org.apache.hadoop.hbase.util.Bytes;
8 import org.apache.hadoop.io.Text;
9 import org.apache.hadoop.io.Writable;
10
11
12 public class MemberMapper extends TableMapper<Writable,Writable>{
13     private Text k = new Text();
14     private Text v = new Text();
15     public static final String FIELD_COMMON_SEPARATOR="\u0001";
16     @Override
17     protected void setup(Context context) throws IOException ,InterruptedException{
18
19     }
20
21     @Override
22     public void map(ImmutableBytesWritable row, Result columns,
23         Context context) throws IOException ,InterruptedException {
24         String value = null;
25
26         //行键值
27         String rowkey = new String(row.get());
28
29         //一行中的所有列族

```

```
*MemberMapper.java Main.java
29 //一行中的所有列族
30 byte[] columnFamily = null;
31
32 //一行中的所有列名
33 byte[] columnQualifier = null;
34
35 long ts = 0L;
36
37 try{
38     for(KeyValue kv : columns.list()){
39         value = Bytes.toStringBinary(kv.getValue());
40         //获得一行中的所有的列族
41         columnFamily = kv.getFamily();
42         //获得一行中的所有列
43         columnQualifier = kv.getQualifier();
44         //获得单元的时间戳
45         ts = kv.getTimestamp();
46
47         k.set(rowkey);
48         v.set(Bytes.toString(columnFamily)+FIELD_COMMON_SEPARATOR+Bytes.toString(columnQualifier)
49             +FIELD_COMMON_SEPARATOR+value+FIELD_COMMON_SEPARATOR+ts);
50         context.write(k, v);
51     }
52 }catch(Exception e){
53     e.printStackTrace();
54     System.err.println("Error:"+e.getMessage()+"Row:"+Bytes.toString(row.get())+"Value:"+value);
55 }
56 };
57 }
```

Main.java:

```
*MemberMapper.java Main.java zkpk@master:~
File Edit View Search Terminal
1 package org.zkpk.hbase.inputSource;
2
3 import org.apache.hadoop.conf.Configuration;
4 import org.apache.hadoop.fs.FileSystem;
5 import org.apache.hadoop.fs.Path;
6 import org.apache.hadoop.hbase.HBaseConfiguration;
7 import org.apache.hadoop.hbase.client.Scan;
8 import org.apache.hadoop.hbase.mapreduce.TableMapReduceUtil;
9 import org.apache.hadoop.hbase.util.Bytes;
10 import org.apache.hadoop.io.Text;
11 import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
12 import org.apache.hadoop.mapreduce.Job;
13 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
14 import org.apache.commons.logging.LogFactory;
15 import org.apache.commons.logging.Log;
16
17 public class Main {
18     static final Log LOG = LogFactory.getLog(Main.class);
19
20     //JobName
21     public static final String NAME = "Member Test1";
22
23     //输出目录
24     public static final String TEMP_INDEX_PATH = "hdfs://master:9000/tmp/member_user";
25
26     //HBase作为输入源的HBase中的表 2018211582lzy
27     public static String inputTable = "2018211582lzy";
28
29 }
```

- 打包程序
- 运行程序, 查看结果

【截图 3：结果截图】（截图需要包含标记信息，未按要求扣分）

运行结果：

```
zkpk@master:~  
File Edit View Search Terminal Help  
RPC_RETRIES=0  
File Input Format Counters  
Bytes Read=0  
File Output Format Counters  
Bytes Written=455  
[zkpk@2018211582lzy ~]$ hadoop fs -cat /tmp/member_user/part-m-00000  
SLF4J: Class path contains multiple SLF4J bindings.  
SLF4J: Found binding in [jar:file:/home/zkpk/hadoop-2.7.3/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]  
SLF4J: Found binding in [jar:file:/home/zkpk/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]  
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.  
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]  
2018211582lzy001 cf1age9 617018362239  
2018211582lzy001 cf1genderman 617018423268  
2018211582lzy001 cf1nameizhiyi 617018409106  
2018211582lzy001 cf2height78 617018479196  
2018211582lzy001 cf2weight70 617018488260  
2018211582lzy002 cf1age20 617018551632  
2018211582lzy002 cf1genderwoman 617018603656  
2018211582lzy002 cf1namehexing 617018573701  
2018211582lzy002 cf2height60 617018614523  
2018211582lzy002 cf2weight60 617018624161  
[zkpk@2018211582lzy ~]$
```

提示：需要把"part-m-00000"文件保存到自己电脑上（实验二需要这个数据）

- 拉取文件命令：

```
[zkpk@2020110808zs ~]$ hadoop fs -get /tmp/user/part-m-00000 /home/zkpk/
```

- 把"part-m-00000"文件从虚拟机下载到自己电脑：

下载文件 ×

从实验环境中导出代码

下载文件所在路径

下载文件名

登录用户

登录密码

确定

二、 HBase 编程：存储数据至 HBase 表

1. 实验描述（参照普开实验“数据库技术(NoSQL)”第 16 章）

使用 MapReduce 批量将 HDFS 上的数据导入到 HBase 表中，学习本实验能快速掌握 HBase 数据库在分布式计算中的应用，理解 Java API 读取 HBase 数据等相关内容。

2. 主要步骤

修改主机名：主机名格式：学号+自己姓名简称（未修改扣分）

```
[zkpk@master ~]$ su root
Password: 密码: zkpk
[root@master zkpk]# vim /etc/hostname
[root@master zkpk]# hostname 2020110808zs
[root@master zkpk]# bash
[root@2020110808zs zkpk]# su zkpk
```

/etc/hostname 文件内容如下



```
File Edit View Search Terminal Help
2020110808zs
~
~
~
~
~
```

- 启动 Hadoop 集群

在 master 运行：

```
[zkpk@2020110808zs ~]$ start-all.sh
```

- 启动 Zookeeper 集群

需要在 master、slave01、slave02 分别运行：

```
[zkpk@2020110808zs ~]$ zkServer.sh start
```

- 启动 HBase 集群

在 master 运行：

```
[zkpk@2020110808zs ~]$ start-hbase.sh
```

- 进入 HBase Shell 创建实验用表

输入 hbase shell 进入 hbase 交互式环境：

```
lzkok@2020110808zs ~1$ hbase shell
```

表格要求与实验一一致

由于表格不存在，需要手动创建（不需要插入数据）

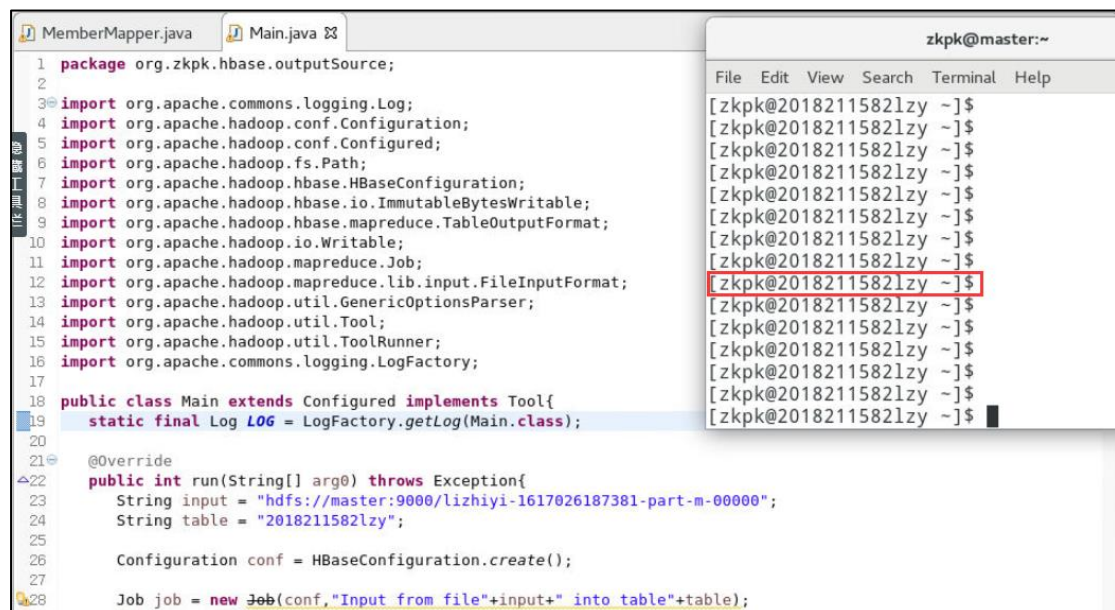
```
|hbase(main):002:0> create '2020110808zs','cf1','cf2'
```

- 打开 Eclipse
- 编写代码

【截图 4：截取完整 Mapper.java 代码】

（截图需要包含标记信息，未按要求扣分，代码不完整扣分）

Main.java:

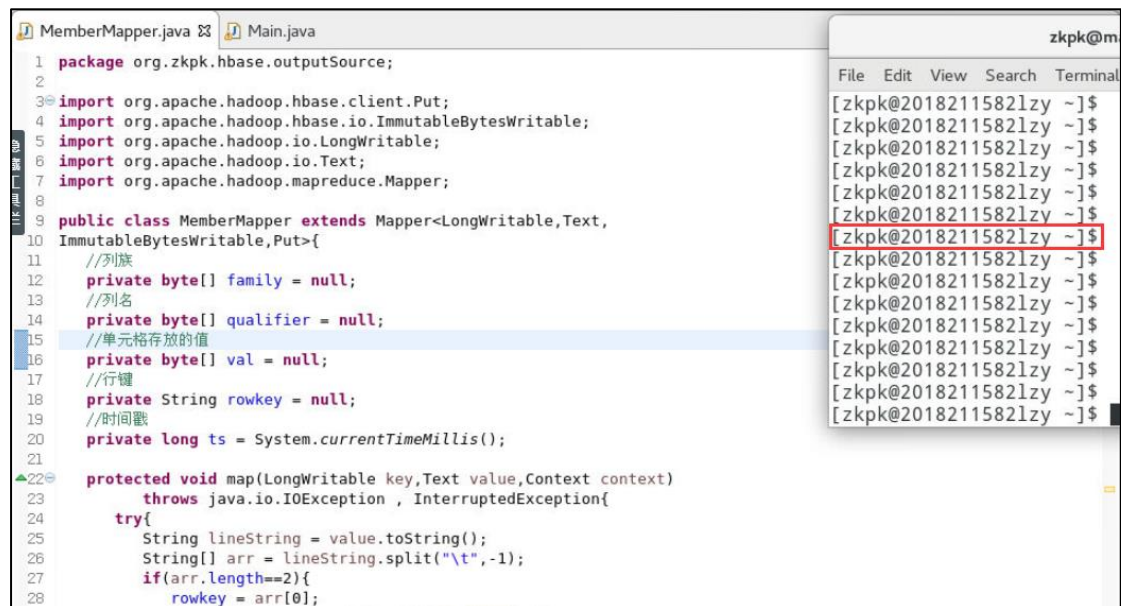


The screenshot shows an IDE with two tabs: 'MemberMapper.java' and 'Main.java'. The 'Main.java' tab is active, displaying the following code:

```
1 package org.zkpk.hbase.outputSource;
2
3 import org.apache.commons.logging.Log;
4 import org.apache.hadoop.conf.Configuration;
5 import org.apache.hadoop.conf.Configured;
6 import org.apache.hadoop.fs.Path;
7 import org.apache.hadoop.hbase.HBaseConfiguration;
8 import org.apache.hadoop.hbase.io.ImmutableBytesWritable;
9 import org.apache.hadoop.hbase.mapreduce.TableOutputFormat;
10 import org.apache.hadoop.io.Writable;
11 import org.apache.hadoop.mapreduce.Job;
12 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
13 import org.apache.hadoop.util.GenericOptionsParser;
14 import org.apache.hadoop.util.Tool;
15 import org.apache.hadoop.util.ToolRunner;
16 import org.apache.commons.logging.LogFactory;
17
18 public class Main extends Configured implements Tool{
19     static final Log LOG = LogFactory.getLog(Main.class);
20
21     @Override
22     public int run(String[] arg0) throws Exception{
23         String input = "hdfs://master:9000/lizhiyi-1617026187381-part-m-00000";
24         String table = "2018211582lzy";
25
26         Configuration conf = HBaseConfiguration.create();
27
28         Job job = new Job(conf,"Input from file"+input+" into table"+table);
```

On the right side, there is a terminal window titled 'zkpk@master:~'. It shows a series of shell prompts '[zkpk@2018211582lzy ~]\$' and a single line of output '[zkpk@2018211582lzy ~]\$' which is highlighted with a red box.

MemberMapper.java:



The screenshot shows an IDE with two tabs: MemberMapper.java and Main.java. The MemberMapper.java file contains the following code:

```
1 package org.zkpk.hbase.outputSource;
2
3 import org.apache.hadoop.hbase.client.Put;
4 import org.apache.hadoop.hbase.io.ImmutableBytesWritable;
5 import org.apache.hadoop.io.LongWritable;
6 import org.apache.hadoop.io.Text;
7 import org.apache.hadoop.mapreduce.Mapper;
8
9 public class MemberMapper extends Mapper<LongWritable,Text,
10     ImmutableBytesWritable,Put>{
11     //列族
12     private byte[] family = null;
13     //列名
14     private byte[] qualifier = null;
15     //单元格存放的值
16     private byte[] val = null;
17     //行键
18     private String rowkey = null;
19     //时间戳
20     private long ts = System.currentTimeMillis();
21
22     protected void map(LongWritable key,Text value,Context context)
23         throws java.io.IOException , InterruptedException{
24         try{
25             String lineString = value.toString();
26             String[] arr = lineString.split("\t",-1);
27             if(arr.length==2){
28                 rowkey = arr[0];
```

The terminal window on the right shows a series of shell commands and their outputs, all of which are the same: [zkpk@20182115821zy ~]\$.



The screenshot shows the continuation of the MemberMapper.java file and the terminal window.

```
20     private long ts = System.currentTimeMillis();
21
22     protected void map(LongWritable key,Text value,Context context)
23         throws java.io.IOException , InterruptedException{
24         try{
25             String lineString = value.toString();
26             String[] arr = lineString.split("\t",-1);
27             if(arr.length==2){
28                 rowkey = arr[0];
29                 String[] vals = arr[1].split("\u0001",-1);
30                 if(vals.length==4){
31                     family = vals[0].getBytes();
32                     qualifier = vals[1].getBytes();
33                     val = vals[2].getBytes();
34                     ts = Long.parseLong(vals[3]);
35
36                     Put put = new Put(rowkey.getBytes(),ts);
37                     put.add(family, qualifier, val);
38                     context.write(new ImmutableBytesWritable(rowkey.getBytes()), put);
39                 }
40             }
41         }catch(Exception e){
42             e.printStackTrace();
43         }
44     }
45 }
46
47 }
```

The terminal window on the right shows a series of shell commands and their outputs, all of which are the same: [zkpk@20182115821zy ~]\$.

- 打包程序
- 上传实验一下载的“part-m-0000”文件(如果忘记下载, 需要手动创建)

上传文件:

上传文件

×

上传文件到实验环境

上传位置

/home/zkpk/

登录用户

zkpk

登录密码

....

上传文件

选择文件

提示：文件大小不超过500M

确定

上传“part-m-0000”到 hdfs 命令：

```
[zkpk@2020110808zs ~]$ hadoop fs -put part-m-00000 /
```

查看是否上传成功：

```
[zkpk@2020110808zs ~]$ hadoop fs -ls /
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/zkpk/hadoop-2.7.3/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/zkpk/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Found 4 items
drwxr-xr-x   - zkpk supergroup          0 2021-03-26 04:38 /hbase
-rw-r--r--   2 zkpk supergroup        272 2021-03-26 04:22 /part-m-00000
drwx-----   - zkpk supergroup          0 2018-09-03 04:03 /tmp
drwxr-xr-x   - zkpk supergroup          0 2018-08-31 03:05 /user
```

- 运行 jar 包，查看结果

【截图 5：结果截图】（截图需要包含标记信息，未按要求扣分）

```

zkpk@2018211582lzy ~]$ hadoop fs -ls /
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/zkpk/hadoop-2.7.3/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/zkpk/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Found 4 items
drwxr-xr-x - zkpk supergroup 0 2021-03-29 15:46 /hbase
-rw-r--r-- 2 zkpk supergroup 455 2021-03-29 16:40 /lizhiyi-1617026187381-part-m-00000
drwx----- - zkpk supergroup 0 2018-09-03 04:03 /tmp
drwxr-xr-x - zkpk supergroup 0 2018-08-31 03:05 /user
zkpk@2018211582lzy ~]$ hbase shell

```

```

hbase(main):002:0> list
TABLE
2018211582lzy
1 row(s) in 0.1260 seconds
=> ["2018211582lzy"]
hbase(main):003:0>

```

```

hbase(main):001:0> scan '2018211582lzy'
ROW COLUMN+CELL
2018211582lzy001 column=cf1:age, timestamp=1617018362239, value=19
2018211582lzy001 column=cf1:gender, timestamp=1617018423268, value=man
2018211582lzy001 column=cf1:name, timestamp=1617018409106, value=lizhiyi
2018211582lzy001 column=cf2:height, timestamp=1617018479196, value=178
2018211582lzy001 column=cf2:weight, timestamp=1617018488260, value=70
2018211582lzy002 column=cf1:age, timestamp=1617018551632, value=20
2018211582lzy002 column=cf1:gender, timestamp=1617018603656, value=woman
2018211582lzy002 column=cf1:name, timestamp=1617018573701, value=hexing
2018211582lzy002 column=cf2:height, timestamp=1617018614523, value=160
2018211582lzy002 column=cf2:weight, timestamp=1617018624161, value=60
2 row(s) in 0.3230 seconds

```

附 录

实验一:

Main.java

```
package org.zkpk.hbase.inputSource;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.hbase.HBaseConfiguration;

import org.apache.hadoop.hbase.client.Scan;

import org.apache.hadoop.hbase.mapreduce.TableMapReduceUtil;

import org.apache.hadoop.hbase.util.Bytes;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import org.apache.commons.logging.LogFactory;

import org.apache.commons.logging.Log;

public class Main {

    static final Log LOG = LogFactory.getLog(Main.class);

    //JobName

    public static final String NAME = "Member Test1";

    //输出目录
```



```

    public static final String TEMP_INDEX_PATH = "hdfs://master:9000/tmp/
member_user";

    //HBase 作为输入源的 HBase 中的表 20182115821zy

    public static String inputTable = "20182115821zy";

    public static void main(String[] args) throws Exception{

        //1. 获得 HBase 的配置信息

        Configuration conf = HBaseConfiguration.create();

        //2. 创建全表扫描器 scan 对象

        Scan scan = new Scan();

        scan.setBatch(0);

        scan.setCaching(10000);

        scan.setMaxVersions();

        scan.setTimeRange(System.currentTimeMillis() - 3*24*3600*100
0L, System.currentTimeMillis());

        //3. 配置 scan, 添加扫描的条件, 列族和列族名 (这里列族名为 cf1, 列名 na
me)

        scan.addColumn(Bytes.toBytes("cf1"), Bytes.toBytes("name"));

        scan.addColumn(Bytes.toBytes("cf1"), Bytes.toBytes("age"));

        scan.addColumn(Bytes.toBytes("cf1"), Bytes.toBytes("gender
"));

        scan.addColumn(Bytes.toBytes("cf2"), Bytes.toBytes("height
"));

        scan.addColumn(Bytes.toBytes("cf2"), Bytes.toBytes("weight
"));

        //4. 设置 hadoop 的推测执行为 false

        conf.setBoolean("mapred.map.tasks.speculative.execution", fa
lse);

        conf.setBoolean("mapred.reduce.tasks.speculative.execution
", false);

```

```

//5. 设置 HDFS 的存储 HBase 表中数据的路径

Path tmpIndexPath = new Path(TEMP_INDEX_PATH);

FileSystem fs = FileSystem.get(conf);

if(fs.exists(tmpIndexPath)){

    fs.delete(tmpIndexPath, true);

}

//6. 创建 job 对象

Job job = new Job(conf, NAME);

//设置执行 JOB 的类

job.setJarByClass(Main.class);

TableMapReduceUtil.initTableMapperJob(inputTable, scan, MemberMapper.class, Text.class, Text.class, job);

job.setNumReduceTasks(0);

job.setOutputFormatClass(TextOutputFormat.class);

FileOutputFormat.setOutputPath(job, tmpIndexPath);

boolean success = job.waitForCompletion(true);

System.exit(success ? 0 : 1);

}

}

```

MemberMapper.java

```

package org.zkpk.hbase.inputSource;

import java.io.IOException;

import org.apache.hadoop.hbase.KeyValue;

import org.apache.hadoop.hbase.client.Result;

```

```

import org.apache.hadoop.hbase.io.ImmutableBytesWritable;

import org.apache.hadoop.hbase.mapreduce.TableMapper;

import org.apache.hadoop.hbase.util.Bytes;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.Writable;

public class MemberMapper extends TableMapper<Writable,Writable>{

    private Text k = new Text();

    private Text v = new Text();

    public static final String FIELD_COMMON_SEPARATOR=" ";

    @Override

    protected void setup(Context context) throws IOException ,InterruptedException{

    }

    @Override

    public void map(ImmutableBytesWritable row, Result columns,

                    Context context) throws IOException ,InterruptedException {

        String value = null;

        //行键值

        String rowkey = new String(row.get());

        //一行中的所有列族

        byte[] columnFamily = null;

        //一行中的所有列名

        byte[] columnQualifier = null;

        long ts = 0L;

```

```

        try{

            for(KeyValue kv : columns.list()){

                value = Bytes.toStringBinary(kv.getValue

            );

                //获得一行中的所有的列族

                columnFamily = kv.getFamily();

                //获得一行中的所有列

                columnQualifier = kv.getQualifier();

                //获得单元的时间戳

                ts = kv.getTimestamp();


                k.set(rowkey);

                v.set(Bytes.toString(columnFamily)

+FIELD_COMMON_SEPARATOR+Bytes.toString(columnQualifier)

                +FIELD_COMMON_SEPARATOR+va

lue+FIELD_COMMON_SEPARATOR+ts);

                context.write(k, v);

            }

        } catch(Exception e) {

            e.printStackTrace();

            System.err.println("Error:"+e.getMessage()+" ,Row:"+

Bytes.toString(row.get())+" ,Value:"+value);

        }

    };

}

```

实验二:

Main.java

```
package org.zkpk.hbase.outputSource;

import org.apache.commons.logging.Log;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.conf.Configured;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.hbase.HBaseConfiguration;

import org.apache.hadoop.hbase.io.ImmutableBytesWritable;

import org.apache.hadoop.hbase.mapreduce.TableOutputFormat;

import org.apache.hadoop.io.Writable;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.util.GenericOptionsParser;

import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;

import org.apache.commons.logging.LogFactory;

public class Main extends Configured implements Tool{

    static final Log LOG = LogFactory.getLog(Main.class);

    @Override

    public int run(String[] arg0) throws Exception{
```



```

        String input = "hdfs://master:9000/lizhiyi-1617026187381-par
t-m-00000";

        String table = "2018211582lzy";

        Configuration conf = HBaseConfiguration.create();

        Job job = new Job(conf, "Input from file"+input+" into table"+
table);

        job.setJarByClass(Main.class);

        job.setMapperClass(MemberMapper.class);

        job.setOutputFormatClass(TableOutputFormat.class);

        job.getConfiguration().set(TableOutputFormat.OUTPUT_TABLE, t
able);

        job.setOutputKeyClass(ImmutableBytesWritable.class);

        job.setOutputValueClass(Writable.class);

        job.setNumReduceTasks(0);

        FileInputFormat.addInputPath(job, new Path(input));

        return job.waitForCompletion(true)?0:1;

    }

    public static void main(String[] args) throws Exception{

        Configuration conf = new Configuration();

        String[] otherArgs = new GenericOptionsParser(conf,args).get
RemainingArgs();

        for(int i=0;i<otherArgs.length;i++){

            System.out.println(otherArgs[i]);

        }

        int res = 1;

        try{

```

```

        res = ToolRunner.run(conf, new Main(), otherArgs);

        System.out.println("-----"+res);

    } catch (Exception e) {

        e.printStackTrace();

    }

    System.exit(res);

}

}

```

MemberMapper.java

```

package org.zkp.hbase.outputSource;

import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.io.ImmutableBytesWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class MemberMapper extends Mapper<LongWritable, Text,
    ImmutableBytesWritable, Put>{

    //列族

    private byte[] family = null;

    //列名

    private byte[] qualifier = null;

    //单元格存放的值

```

```

private byte[] val = null;

//行键

private String rowkey = null;

//时间戳

private long ts = System.currentTimeMillis();

protected void map(LongWritable key, Text value, Context context)

        throws java.io.IOException , InterruptedException{

    try{

        String lineString = value.toString();

        String[] arr = lineString.split("\t",-1);

        if(arr.length==2){

            rowkey = arr[0];

            String[] vals = arr[1].split(" ",-1);

            if(vals.length==4){

                family = vals[0].getBytes();

                qualifier = vals[1].getBytes();

                val = vals[2].getBytes();

                ts = Long.parseLong(vals[3]);

                Put put = new Put(rowkey.getBytes

(),ts);

                put.add(family, qualifier, val);

                context.write(new ImmutableBytesWr

itable(rowkey.getBytes()), put);

            }

        }

    }catch(Exception e){

```

```
        e.printStackTrace();
    }
}
}
```