# “大数据工程”课程实验报告

| 题目：MapReduce 和 HBase 的综合编程实践 | 学号姓名：21377061 范春 | 日期：2024.3.29 |
|---|---|---|

实验环境：

1、虚拟机软件：VMvare

2、Hadoop 版本：3.1.3

3、Java 版本：Oracle JDK 1.8

4、Java IDE：Eclipse

实验内容与完成情况：

1、安装 Hbase

```
hadoop@u-virtual-machine:~$ cd ~
hadoop@u-virtual-machine:~$ sudo tar -zxf ~/下载/hbase-2.2.2-bin.tar.gz -C /usr/
local
[sudo] hadoop 的密码：
hadoop@u-virtual-machine:~$ cd /usr/local
hadoop@u-virtual-machine:/usr/local$ sudo mv ./hbase-2.2.2 ./hbase
hadoop@u-virtual-machine:/usr/local$ cd /usr/local
hadoop@u-virtual-machine:/usr/local$ sudo chown -R hadoop ./hbase
hadoop@u-virtual-machine:/usr/local$ gedit ~/.bashrc
hadoop@u-virtual-machine:/usr/local$ source ~/.bashrc
hadoop@u-virtual-machine:/usr/local$ sudo chown -R hadoop ./hbase
hadoop@u-virtual-machine:/usr/local$ /usr/local/hbase/bin/hbase version
HBase 2.2.2
Source code repository git://6ad68c41b902/opt/hbase-rm/output/hbase revision=e65
13a76c91cceda95dad7af246ac81d46fa2589
Compiled by hbase-rm on Sat Oct 19 10:10:12 UTC 2019
From source with checksum 4d23f97701e395c5d34db1882ac5021b
hadoop@u-virtual-machine:/usr/local$
```

2、伪分布式配制

```
hadoop@u-virtual-machine:/usr/local/hadoop$ cd /usr/local/hbase
hadoop@u-virtual-machine:/usr/local/hbase$ bin/start-hbase.sh
localhost: running zookeeper, logging to /usr/local/hbase/bin/../logs/hbase-hadoop-zookeeper-u-virtual-m
achine.out
running master, logging to /usr/local/hbase/bin/../logs/hbase-hadoop-master-u-virtual-machine.out
: running regionserver, logging to /usr/local/hbase/bin/../logs/hbase-hadoop-regionserver-u-virtual-mach
ine.out
hadoop@u-virtual-machine:/usr/local/hbase$ jps
2498 NameNode
38658 HRegionServer
38930 Jps
2646 DataNode
2887 SecondaryNameNode
38588 HMaster
38525 HQuorumPeer
hadoop@u-virtual-machine:/usr/local/hbase$
```

3、进入 shell 界面及退出 Hbase 和 Hadoop

```
hadoop@u-virtual-machine:/usr/local/hbase$ bin/hbase shell
2024-03-27 23:50:54,789 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for you
r platform... using builtin-java classes where applicable
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
For Reference, please visit: http://hbase.apache.org/2.0/book.html#shell
Version 2.2.2, re6513a76c91cceda95dad7af246ac81d46fa2589, Sat Oct 19 10:10:12 UTC 2019
Took 0.0026 seconds
hbase(main):001:0> bin/stop-hbase.sh
NameError: undefined local variable or method `bin' for main:Object

hbase(main):002:0> exit
hadoop@u-virtual-machine:/usr/local/hbase$ bin/stop-hbase.sh
stopping hbase.........
localhost: running zookeeper, logging to /usr/local/hbase/bin/../logs/hbase-hadoop-zookeeper-u-virtual-m
achine.out
localhost: stopping zookeeper.
hadoop@u-virtual-machine:/usr/local/hbase$ ./sbin/stop-dfs.sh
-bash: ./sbin/stop-dfs.sh: 没有那个文件或目录
hadoop@u-virtual-machine:/usr/local/hbase$ cd /usr/local/hadoop
hadoop@u-virtual-machine:/usr/local/hadoop$ ./sbin/stop-dfs.sh
Stopping namenodes on [localhost]
Stopping datanodes
Stopping secondary namenodes [u-virtual-machine]
hadoop@u-virtual-machine:/usr/local/hadoop$
```

4、Task1

```
hbase(main):006:0> list
TABLE
movieUserRatingsInfo
1 row(s)
Took 0.0485 seconds
=> ["movieUserRatingsInfo"]
hbase(main):007:0> scan "movieUserRatingsInfo",{LIMIT=>20}
ROW                                   COLUMN+CELL
 10000_1879032_1379202125             column=movies:Genres, timestamp=1711637771069, value=Comedy|Fantasy
 10000_1879032_1379202125             column=movies:Title, timestamp=1711637771069, value=Rapture-Palooza (2013)
 10000_1879032_1379202125             column=ratings:movie_id, timestamp=1711637771069, value=1879032
 10000_1879032_1379202125             column=ratings:rating, timestamp=1711637771069, value=6
 10000_1879032_1379202125             column=ratings:timestamp, timestamp=1711637771069, value=1379202125
 10000_1879032_1379202125             column=ratings:user_id, timestamp=1711637771069, value=10000
 10000_1879032_1379202125             column=users:twitter_id, timestamp=1711637771069, value=1618986320
 10001_0795461_1377421286             column=movies:Genres, timestamp=1711637771070, value=Comedy
 10001_0795461_1377421286             column=movies:Title, timestamp=1711637771070, value=Scary Movie 5 (2013)
 10001_0795461_1377421286             column=ratings:movie_id, timestamp=1711637771070, value=0795461
 10001_0795461_1377421286             column=ratings:rating, timestamp=1711637771070, value=7
 10001_0795461_1377421286             column=ratings:timestamp, timestamp=1711637771070, value=1377421286
 10001_0795461_1377421286             column=ratings:user_id, timestamp=1711637771070, value=10001
 10001_0795461_1377421286             column=users:twitter_id, timestamp=1711637771070, value=106859947
 10001_1201167_1378452938             column=movies:Genres, timestamp=1711637771070, value=Comedy|Drama
```

5、Task2

```
hadoop@u-virtual-machine:/usr/local/hbase$ cd /usr/local/hadoop
hadoop@u-virtual-machine:/usr/local/hadoop$ cd /usr/local/hadoop/myapp
hadoop@u-virtual-machine:/usr/local/hadoop/myapp$ ls
HDFSweek3.jar  MovieRatingAverage.jar
hadoop@u-virtual-machine:/usr/local/hadoop/myapp$ cd /usr/local/hadoop
hadoop@u-virtual-machine:/usr/local/hadoop$ ./bin/hadoop jar ./myapp/MovieRatingAverage.jar result1
2024-03-29 00:04:51,271 INFO impl.MetricsConfig: loaded properties from hadoop-metrics2.properties
2024-03-29 00:04:51,379 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2024-03-29 00:04:51,379 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2024-03-29 00:04:51,787 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool
er to remedy this.
2024-03-29 00:04:52,410 INFO zookeeper.ZooKeeper: Client environment:zookeeper.version=3.4.13-2d71af4dbe22557fda74f9a9b4309b15a7
2024-03-29 00:04:52,410 INFO zookeeper.ZooKeeper: Client environment:host.name=u-virtual-machine
2024-03-29 00:04:52,410 INFO zookeeper.ZooKeeper: Client environment:java.version=1.8.0_162
```

```
hadoop@u-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -cat result1/* | head -n 10
2024-03-29 00:09:04,863 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
000000000106519 9.0
0000008 5.0
0000010 10.0
0000012 10.0
0000091 6.0
0000131 7.0
0000417 8.478260869565217
0000439 7.0
0000628 4.5
0000833 3.0
```

6、Task3

```
hadoop@u-virtual-machine:/usr/local/hadoop$ cd /usr/local/hadoop/myapp
hadoop@u-virtual-machine:/usr/local/hadoop/myapp$ ls
HDFSweek3.jar  MovieRatingAverage.jar  MovieRatingNum.jar
hadoop@u-virtual-machine:/usr/local/hadoop/myapp$ cd /usr/local/hadoop/
hadoop@u-virtual-machine:/usr/local/hadoop$ ./bin/hadoop jar ./myapp/MovieRatingNum.jar result2
2024-03-29 00:21:29,680 INFO impl.MetricsConfig: loaded properties from hadoop-metrics2.properties
2024-03-29 00:21:29,792 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2024-03-29 00:21:29,792 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2024-03-29 00:21:30,132 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement
er to remedy this.
2024-03-29 00:21:30,722 INFO zookeeper.ZooKeeper: Client environment:zookeeper.version=3.4.13-2d71af4dbe22557fda74f9a9b
2024-03-29 00:21:30,722 INFO zookeeper.ZooKeeper: Client environment:host.name=u-virtual-machine
2024-03-29 00:21:30,722 INFO zookeeper.ZooKeeper: Client environment:java.version=1.8.0_162
2024-03-29 00:21:30,722 INFO zookeeper.ZooKeeper: Client environment:java.vendor=Oracle Corporation
2024-03-29 00:21:30,722 INFO zookeeper.ZooKeeper: Client environment:java.home=/usr/lib/jvm/jdk1.8.0_162/jre
```

```
hadoop@u-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -cat result2/* | head -n 10
2024-03-29 00:22:26,119 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
000000000106519 1
0000008 1
0000010 1
0000012 1
0000091 3
0000131 1
0000417 23
0000439 6
0000628 2
0000833 1
```

出现的问题及解决方案：

当做第一问连接 Hbase 时，出现了如下图所示的问题，经过查找相关资料虽然并没有找到造成这种现象的明确原因，但解决办法是重启 Hbase 即可。

```
2024-03-28 21:27:35,838 INFO  [main] client.RpcRetryingCallerImpl (RpcRetryingCallerImpl.java:callWithRetries(134)) - Call exception, tries=6, retries=16, started=4136 ms ago, cancelle
2024-03-28 21:27:39,867 INFO  [main] client.RpcRetryingCallerImpl (RpcRetryingCallerImpl.java:callWithRetries(134)) - Call exception, tries=7, retries=16, started=8165 ms ago, cancelle
```

完整代码如下：

**Question1:**

```
import org.apache.hadoop.conf.Configuration;
```

```
import org.apache.hadoop.hbase.*;
import org.apache.hadoop.hbase.client.*;
import org.apache.hadoop.hbase.util.Bytes;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

public class task1 {
    public static Configuration configuration;
    public static Connection connection;
    public static Admin admin;

    public static void main(String [] args) throws IOException{
        init();
        createTable("movieUserRatingsInfo",new String[]{"ratings","users","movies"});
        loadData("movieUserRatingsInfo");
        close();
    }

    public static void init(){
        configuration    = HBaseConfiguration.create();
        configuration.set("hbase.rootdir","hdfs://localhost:9000/hbase");
        try{
            connection = ConnectionFactory.createConnection(configuration);
            admin = connection.getAdmin();
        }catch (IOException e){
            e.printStackTrace();
        }
    }

    public static void close(){
        try{
            if(admin != null){
                admin.close();
            }
            if(null != connection){
                connection.close();
            }
        }catch (IOException e){
            e.printStackTrace();
        }
```

```
    }

    public static void createTable(String myTableName,String[] colFamily) throws IOException {
        TableName tableName = TableName.valueOf(myTableName);
        if(admin.tableExists(tableName)){
            System.out.println("talbe is exists!");
        }
        else {
            TableDescriptorBuilder tableDescriptor = TableDescriptorBuilder.newBuilder(tableName);
            for(String str:colFamily){
                ColumnFamilyDescriptor family = ColumnFamilyDescriptorBuilder.newBuilder(Bytes.toBytes(str)).build();
                tableDescriptor.setColumnFamily(family);
            }
            admin.createTable(tableDescriptor.build());
        }
    }

    public static void loadData(String tableName) throws IOException {
        Table table = connection.getTable(TableName.valueOf(tableName));

     // Load movies data
        Map<String, String[]> moviesMap = new HashMap<>();
        BufferedReader movieReader = new BufferedReader(new FileReader("/home/hadoop/桌面/datasets/movies.dat"));
        String movieLine;
        while ((movieLine = movieReader.readLine()) != null) {
            String[] movieData = movieLine.split("::", -1);
            if (movieData.length < 3) {
                movieData = new String[]{movieData[0], movieData[1], "Unknown"};
            }
            moviesMap.put(movieData[0], new String[]{movieData[1], movieData[2]});
        }
        movieReader.close();

        // Load users data
        Map<String, String> usersMap = new HashMap<>();
        BufferedReader userReader = new BufferedReader(new FileReader("/home/hadoop/桌面/datasets/users.dat"));
        String userLine;
        while ((userLine = userReader.readLine()) != null) {
            String[] userData = userLine.split("::");
```

```
                usersMap.put(userData[0], userData[1]);
            }
            userReader.close();

            // Load ratings data and combine with movies and users data
            BufferedReader ratingReader = new BufferedReader(new
FileReader("/home/hadoop/桌面/datasets/ratings.dat"));
            String ratingLine;
            while ((ratingLine = ratingReader.readLine()) != null) {
                String[] ratingData = ratingLine.split("::");
                String userId = ratingData[0];
                String movieId = ratingData[1];
                String rating = ratingData[2];
                String timestamp = ratingData[3];
                String rowKey = userId + "_" + movieId + "_" + timestamp;
                Put put = new Put(Bytes.toBytes(rowKey));
                put.addColumn(Bytes.toBytes("ratings"),          Bytes.toBytes("user_id"),
Bytes.toBytes(userId));
                put.addColumn(Bytes.toBytes("ratings"),          Bytes.toBytes("movie_id"),
Bytes.toBytes(movieId));
                put.addColumn(Bytes.toBytes("ratings"),          Bytes.toBytes("rating"),
Bytes.toBytes(rating));
                put.addColumn(Bytes.toBytes("ratings"),          Bytes.toBytes("timestamp"),
Bytes.toBytes(timestamp));
                if (usersMap.containsKey(userId)) {
                    put.addColumn(Bytes.toBytes("users"),        Bytes.toBytes("twitter_id"),
Bytes.toBytes(usersMap.get(userId)));
                }
                if (moviesMap.containsKey(movieId)) {
                    put.addColumn(Bytes.toBytes("movies"),       Bytes.toBytes("Title"),
Bytes.toBytes(moviesMap.get(movieId)[0]));
                    put.addColumn(Bytes.toBytes("movies"),       Bytes.toBytes("Genres"),
Bytes.toBytes(moviesMap.get(movieId)[1]));
                }
                table.put(put);
            }
            ratingReader.close();
            table.close();
            System.out.println("Successfully!");
        }

        public static void insertData(String tableName,String rowKey,String colFamily,String
col,String val) throws IOException {
            Table table = connection.getTable(TableName.valueOf(tableName));
```

```
            Put put = new Put(rowKey.getBytes());
            put.addColumn(colFamily.getBytes(),col.getBytes(), val.getBytes());
            table.put(put);
            table.close();
        }
}
```

**Question2:**

```
    import org.apache.hadoop.conf.Configuration;

    import org.apache.hadoop.fs.Path;

    import org.apache.hadoop.hbase.HBaseConfiguration;

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

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

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

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

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

    import org.apache.hadoop.io.DoubleWritable;

    import org.apache.hadoop.io.Text;

    import org.apache.hadoop.mapreduce.Job;

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

    import org.apache.hadoop.mapreduce.Reducer;

    import java.io.IOException;


    public class MovieRatingsAverage {

    public static class RatingsMapper extends TableMapper<Text, DoubleWritable> {

        public void map(ImmutableBytesWritable row,

org.apache.hadoop.hbase.client.Result value, Context context) throws IOException,

InterruptedException {

            String movieId =

Bytes.toString(value.getValue(Bytes.toBytes("ratings"),

Bytes.toBytes("movie_id")));

            String ratingStr =
```

```
Bytes.toString(value.getValue(Bytes.toBytes("ratings"),
Bytes.toBytes("rating")));
            try {
                double rating = Double.parseDouble(ratingStr);
                context.write(new Text(movieId), new DoubleWritable(rating));
            } catch (NumberFormatException e) {
                context.getCounter("RatingsMapper",
"InvalidRatingFormat").increment(1);
            }
        }
    }

    public static class RatingsReducer extends Reducer<Text, DoubleWritable, Text,
DoubleWritable> {
        public void reduce(Text key, Iterable<DoubleWritable> values, Context
context) throws IOException, InterruptedException {
            double sum = 0;
            int count = 0;
            for (DoubleWritable val : values) {
                sum += val.get();
                count++;
            }
            double average = count > 0 ? sum / count : 0;
            context.write(key, new DoubleWritable(average));
        }
    }

    public static void main(String[] args) throws Exception {
        Configuration conf = HBaseConfiguration.create();
```

```
        Job job = Job.getInstance(conf, "Movie Ratings Average");
        job.setJarByClass(MovieRatingsAverage.class);
        Scan scan = new Scan();
        scan.setCaching(500);
        scan.setCacheBlocks(false);
        TableMapReduceUtil.initTableMapperJob( "movieUserRatingsInfo",
scan,RatingsMapper.class, Text.class, DoubleWritable.class, job);
        job.setReducerClass(RatingsReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(DoubleWritable.class);
        if (args.length < 1) {
            System.err.println("Usage: MovieRatingsAverage <out>");
            System.exit(2);
        }
        FileOutputFormat.setOutputPath(job, new Path(args[0]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

**Question3:**

```
    import org.apache.hadoop.conf.Configuration;
    import org.apache.hadoop.fs.Path;
    import org.apache.hadoop.hbase.HBaseConfiguration;
    import org.apache.hadoop.hbase.client.Scan;
    import org.apache.hadoop.hbase.io.ImmutableBytesWritable;
    import org.apache.hadoop.hbase.mapreduce.TableMapReduceUtil;
    import org.apache.hadoop.hbase.mapreduce.TableMapper;
    import org.apache.hadoop.hbase.util.Bytes;
    import org.apache.hadoop.io.DoubleWritable;
    import org.apache.hadoop.io.IntWritable;
```

```
    import org.apache.hadoop.io.Text;
    import org.apache.hadoop.mapreduce.Job;
    import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
    import org.apache.hadoop.mapreduce.Reducer;
    import java.io.IOException;
    public class MovieRatingsNum {
    public static class RatingsMapper extends TableMapper<Text, DoubleWritable> {
        public void map(ImmutableBytesWritable row,
org.apache.hadoop.hbase.client.Result value, Context context) throws IOException,
InterruptedException {
            String movieId =
Bytes.toString(value.getValue(Bytes.toBytes("ratings"),
Bytes.toBytes("movie_id")));
            String ratingStr =
Bytes.toString(value.getValue(Bytes.toBytes("ratings"),
Bytes.toBytes("rating")));
            try {
                double rating = Double.parseDouble(ratingStr);
                context.write(new Text(movieId), new DoubleWritable(rating));
            } catch (NumberFormatException e) {
            context.getCounter("RatingsMapper",
"InvalidRatingFormat").increment(1);
            }
        }
    }
    public static class RatingsReducer extends Reducer<Text, DoubleWritable, Text,
IntWritable> {
        public void reduce(Text key, Iterable<DoubleWritable> values, Context
context) throws IOException, InterruptedException {
```

```
            int count = 0;
            for (DoubleWritable val : values) {
                count++;
            }
            context.write(key, new IntWritable(count));
        }
    }
    public static void main(String[] args) throws Exception {
        Configuration conf = HBaseConfiguration.create();
        Job job = Job.getInstance(conf, "Movie Ratings count");
        job.setJarByClass(MovieRatingsNum.class);
        Scan scan = new Scan();
        scan.setCaching(500);
        scan.setCacheBlocks(false);
        TableMapReduceUtil.initTableMapperJob( "movieUserRatingsInfo", scan,
RatingsMapper.class, Text.class, DoubleWritable.class, job);
        job.setReducerClass(RatingsReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        if (args.length < 1) {
            System.err.println("Usage: MovieRatingsCount <out>");
            System.exit(2);
        }
        FileOutputFormat.setOutputPath(job, new Path(args[0]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
    }
```