
Map-Reduce 课程设计——日志统计分析

作者姓名：臧凤云，银琦

(南京大学 计算机科学与技术系，南京 210046)

1 小组信息&实验环境

1.1 小组信息

集群账号：2017st15

组长：臧凤云 141220138 2714359840@qq.com

组员：银琦 141220132 141220132@smail.nju.edu.cn

1.2 实验环境

虚拟机：VMware WorkStation12

操作系统：Ubuntu14 (64bit)

运行架构：Hadoop 集群

编译器：IntelliJ IDEA 17

编程语言：Java

2 题目描述&任务分配

2.1 题目描述

通过使用 MapReduce 来实现日志分析，日志分析在互联网企业应用很广，通过此次课程设计的学习，进一步了解 MapReduce 技术在工业界的应用，熟悉和掌握以下 MapReduce 编程技能：

1. 海量日志数据的统计分析
2. 基于 MapReduce 的预测模型设计，通过对历史日志数据的分析建立预测模型

2.2 任务分配

Part1: 臧凤云负责 task1-4 日志信息统计

Part2: 银琦负责 task5 接口访问频次预测

3 任务分析

3.1 Part1

1. 统计状态码出现总频次；按照小时窗，输出各个时间段各状态码统计情况；输出至 1.txt
2. 统计每个 IP 访问总频次；按照小时窗，输出各个 IP 访问的情况；输出至 IP.txt
3. 统计每个 url 访问总频次；按照秒为单位的时间窗，输出至各个时间段各接口的访问情况；输出至 url.txt
4. 统计每个接口的平均响应时间；按照小时时间窗，输出各个时间段各接口平均响应时间；输出至 url.txt

3.2 Part2

1. 设计算法预测 2015-09-22 接口访问总频次；训练数据为 2015-09-08.log 到 2015-09-21.log 共 14 天

的日志文件，结果按照小时窗，输出各个时间段各接口的访问总频次；输出至默认文件 part-r-00000。

2. 做 RMSE 验证，公式如下：

$$RMSE = \frac{1}{M} \sum_{i=1}^M \sqrt{\frac{\sum_{j=1}^N (C1j - C2j)^2}{N}}$$

其中 M 为时间窗个数。 N 为第 i 个时间窗中访问 URL 个数。 $C1j$ 和 $C2j$ 分别是预测值和真实值第 j 个 URL 的访问频次。

4 设计思路（程序设计的主要流程）

4.1 Part1

4.1.1 整体思路

由于实验要求任务 1-4 读取同一个文件，输入信息相同，所以将 4 个任务合并为一个 MapReduce 程序，而不是 4 个 MapReduce 程序，这样每条日志信息读取一次，就可以统计 4 个任务的信息，并按照任务给定的输出路径输出。

4.1.2 Map

每次的输入为一条日志数据字符串，提取信息，得到状态码，时间，IP，url，响应时间字符串；

每次的输出：（1:状态码:hour，1），（2:IP，hour），（3:url，time），（4:url，hour:响应时间）

4.1.3 Reduce

输入为 Map 的输出，（1:状态码:hour，1）或（2:IP，hour）或（3:url，time）或（4:url，hour:响应时间）

如果输入 key = 1:状态码:hour，计算 values 总次数得到该状态码在该时间段总频次，最后在 cleanup 中计算 24 小时的总次数，先输出状态码总频次，在输出每小时状态码总频次；

如果输入 key = 2:IP，values 为 hour，values 的大小即为该 IP 的总频次，根据 hour 不同计算每个 hour 出现的总频次。这里在 reduce 函数输出，而没有在 cleanup 函数中输出，因为 IP 的个数未知，在 setup 阶段定义结构较为麻烦，没有直接在 reduce 阶段输出方便。

如果输入 key = 3:url，values 为时间（精确到秒），values 大小即为该 url 的总频次，根据 time 不同计算不同 time 的总频次。在 reduce 函数直接输出，理由同上；

如果输入 key = 4:url，values 为 hour 加响应时间，values 大小即为该 url 总频次，根据 hour 不同计算每个 hour 的 url 频次和总响应时间，最后计算每个 hour 的平均响应时间，和所有时间段的平均响应时间，最后在 reduce 函数直接输出，理由同上。

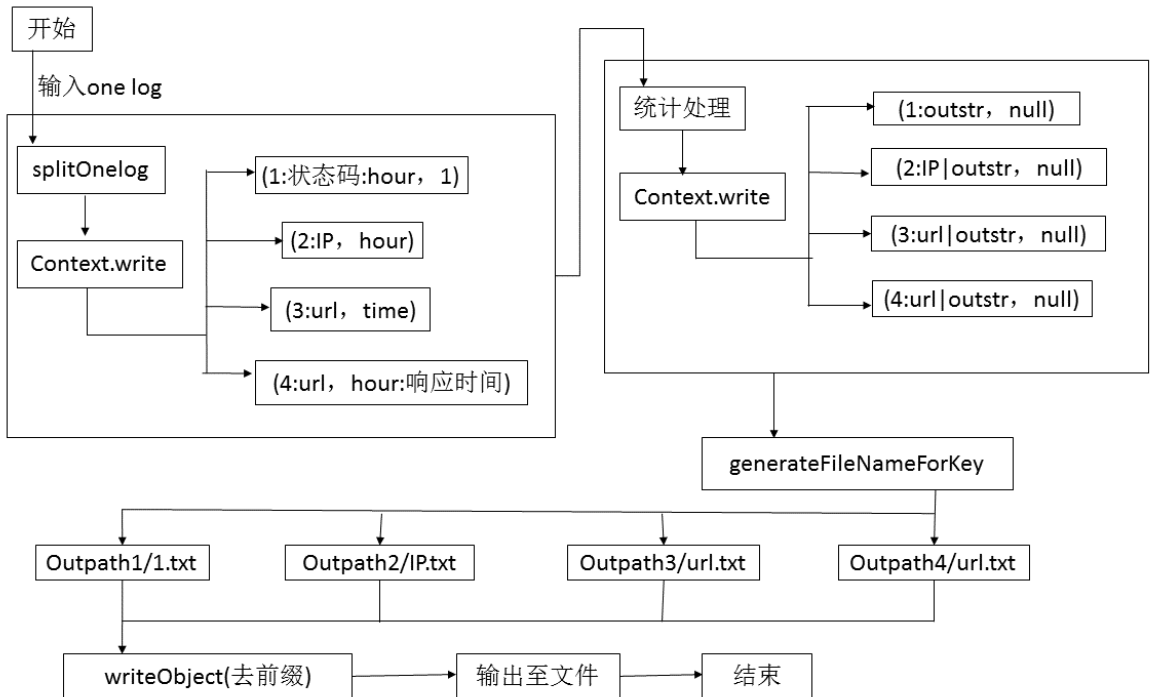
4.1.4 多文件输出

普通 mapreduce 中通常是由 map 和 reduce 两个阶段，在不做设置的情况下，计算结果会以 part-000*输出成多个文件，这样不利于后续结果处理。

在 Hadoop 中，reduce 支持多个输出，输出的文件名也是可控的，就是继承 MultipleTextOutputFormat 类，重写 generateFileNameForKey 方法。如果只是想做到输出结果的文件名可控，实现自己的 LogNameMultipleTextOutputFormat 类，设置 jobconf.setOutputFormat(LogNameMultipleTextOutputFormat.class); 就可以了。

对于以上任务，在输出到文件之前根据输出的 key 值中的附加前缀信息（在 reduce 阶段输出时加上前缀信息），设置输出文件名。在输出前将 key 值中的前缀信息去掉。

4.1.5 流程图



4.2 Part2

4.2.1 整体思路

任务 5 可以分为 3 个模块，第一个模块为训练数据的处理，第二个模块为预测算法，第三个模块为计算 RMSE。在模块一中使用 MapReduce 技术将训练数据读入并处理，将处理后的数据存入指定的数据结构中，在模块二中对数据结构中对数据进行处理，并使用预测算法得到 2015-09-22 的预测值，在模块三中将预测值与真实值做对比，计算出 RMSE。

4.2.2 Map

每次的输入为一条日志数据字符串，提取信息，得到状态码，时间，IP，url，响应时间字符串；
每次的输出：(hour:url:文件名, 1)

4.2.3 Combiner

输入为 Map 的输出：(hour:url:文件名, 1)；
简单的对 values 进行求和，输出 (hour:url:文件名, sum)

4.2.4 Reduce

输入为 Combiner 的输出：(hour:url:文件名, sum)；

根据输入的 key，若文件名中日期小于 22，则计算 values 即统计出了每个文件在每个小时中，每个 url 的出现次数，设计一个大小为 14x24 的二维数组，第一维为文件映射的序号，第二维为小时映射的序号，每一个元素都为一个 HashMap，存放着该文件中该时间段下各 url 的出现次数；若文件名中日期等于 22，设计一个大小为 24 的一维数组，每个元素为一个 HashMap，将 22 日的数据存入该数组中，待以后使用。

4.2.5 预测算法

总体算法思想是对前 14 天的数据求平均值。

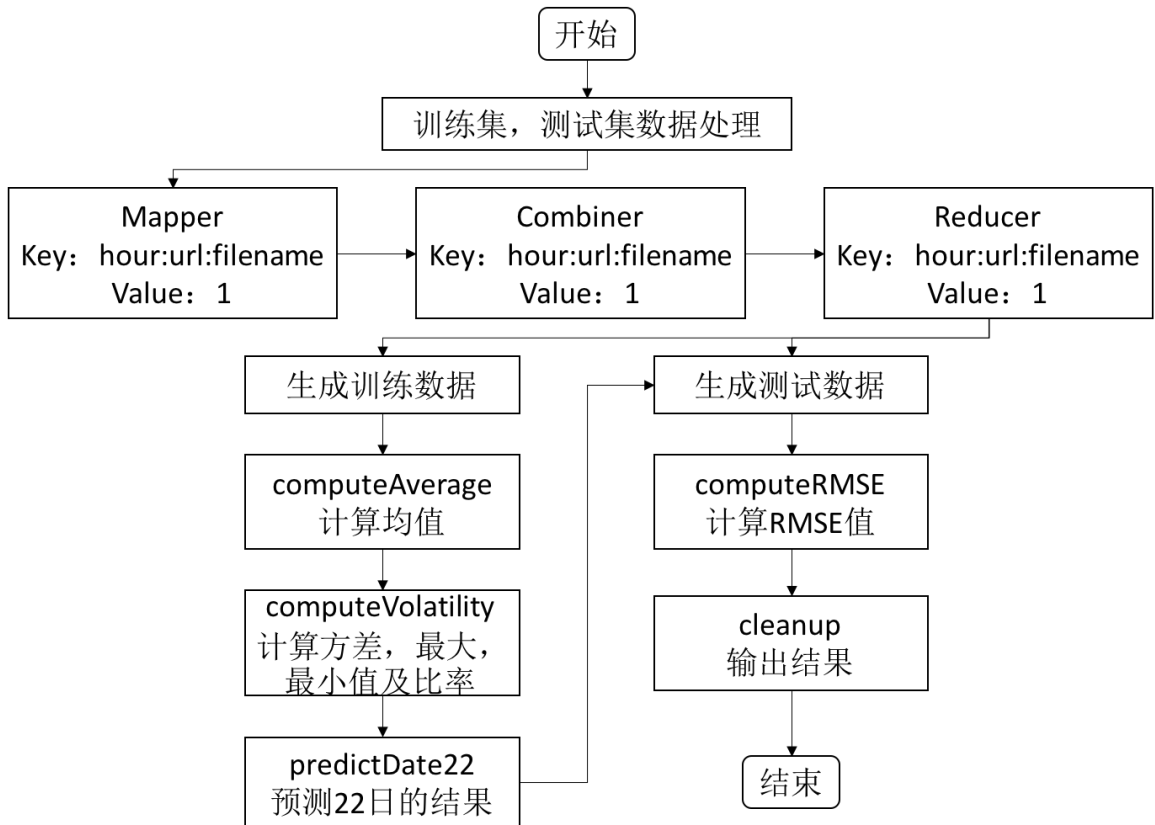
计算出 14 天中每个时间段下每个 url 出现的平均次数、方差、最大值、最小值，并计算出极差占最小值

的比重，即 $((\text{最大值}-\text{最小值})/\text{最小值})$ ，并判断，如果方差小于 1000000 并且比重小于 1，那么视为波动不大，取平均值作为 22 日的预测结果；否则视为波动太大，有误差，按照最近最优先原则，选择 21 日该时间段该 url 的出现次数作为 22 日的预测结果，若 21 日未出现则依次向前推。

4.2.6 计算 RMSE

对预测值和真实值做遍历，若 url 在预测值中但不在真实值中，则省略不考虑，若在真实值中不在预测值中，将该 url 的预测值视为 0 进行计算。

4.2.7 流程图



5 实现（程序采用的主要方法）

5.1 Part1

5.1.1 对每条日志信息分词

调用 Split 类中 splitOnelog 方法

```

public static String[] splitOnelog(String log) {
    String[] partwords = new String[7];
    String[] splitwords = log.split(" ");
    partwords[6] = splitwords[splitwords.length - 1];
    partwords[5] = splitwords[splitwords.length - 2];
    partwords[4] = splitwords[splitwords.length - 3];
    partwords[3] = splitwords[splitwords.length - 4];
}
  
```

```

Pattern pattern0 = Pattern.compile("\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}");
Pattern pattern1 = Pattern.compile("(\\[).*?(\\])");
Pattern pattern2 = Pattern.compile("(\\").*?(\\")");
Matcher matcher = pattern0.matcher(log);
if (matcher.find()) partwords[0] = matcher.group();
matcher = pattern1.matcher(log);
if (matcher.find()) partwords[1] = matcher.group();
matcher = pattern2.matcher(log);
if (matcher.find()) partwords[2] = matcher.group();
else {
    partwords[2] = splitwords[splitwords.length - 6] + " " +
splitwords[splitwords.length - 5];
    if (splitwords[splitwords.length - 5].equals("null"))
        partwords[2] = null;
}
for (int i = 3; i < 7; i++)
    if (partwords[i].length() > 15 || partwords[i].equals("-"))
        partwords[i] = null;
return partwords;
}

```

例：

172.22.49.26 [16/Sep/2015:00:22:23 +0800] "GET /tour/category/query HTTP/1.1"
GET 200 156 2

首先这一条信息 `log(String)`，根据空格划分，可以得到后四个信息存入 `partwords[3-6]`；

通过正则表达式 `"\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}"`，匹配 IP 存入 `partwords[0]`；

通过正则表达式 `"(\\[).*?(\\])"`，匹配 `[]` 之间字符串存入 `partwords[1]`；

通过正则表达式 `"(\\").*?(\\")"`，匹配 `" "` 之间字符串存入 `partwords[2]`；

使用正则表达式的优势在于可以避免 `StringIndexOutOfBoundsException` 这样低级的错误。

特殊情况：实验中发现有部分数据内容确实，比如出现 `'-'` 情况，此时将对应值设为 `null`，也有记录双引号只有一半，需要找到单个引号位置，做特殊分割。

5.1.2 Map

首先分割 `log` 字符串，在处理每一个任务前，判断该任务用到的信息是否为空，空值情况不处理。

```

public static class MyMapper extends Mapper<Object, Text, Text, Text> {
    @Override
    protected void map(Object key, Text value, Context context) {
        try {
            String Onelog = value.toString().trim();
            String[] partwords = Split.splitOnelog(Onelog);
            Text keyInfo;
            //task1
            if (partwords[4] != null && partwords[1] != null) {
                int index = partwords[1].indexOf(':');
                String hour = partwords[1].substring(index + 1, index + 3);
                String task1str = "1:" + partwords[4] + ":" + hour;
            }
        }
    }
}

```

```

        keyInfo = new Text(task1str);
        context.write(keyInfo, new Text("1"));
    }
    //task2
    if (partwords[0] != null && partwords[1] != null) {
        int index = partwords[1].indexOf(':');
        String hour = partwords[1].substring(index + 1, index + 3);
        String task2str = "2:" + partwords[0];
        keyInfo = new Text(task2str);
        context.write(keyInfo, new Text(hour));
    }
    //task3
    if (partwords[2] != null && partwords[1] != null) {
        int index = partwords[1].indexOf(' ');
        String time = partwords[1].substring(1, index);
        String[] buf = partwords[2].split(" ");
        String task3str = "3:" + buf[1];
        keyInfo = new Text(task3str);
        context.write(keyInfo, new Text(time));
    }
    //task4
    if (partwords[1] != null && partwords[2] != null && partwords[6] != null) {
        int index = partwords[1].indexOf(':');
        String hour = partwords[1].substring(index + 1, index + 3);
        String[] buf = partwords[2].split(" ");
        String task4str = "4:" + buf[1];
        keyInfo = new Text(task4str);
        context.write(keyInfo, new Text(hour + ":" + partwords[6]));
    }
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

5.1.3 Reduce

四个任务在一个 Reduce 里处理。

任务 1: setup 中初始化, Reduce 中统计计算, cleanup 中输出, 输出的 key 前加上 “1:” 前缀, 用于修改文件名。

```

//task1
Map<String,Integer>Status = new TreeMap<String,Integer>();
Map<String, Integer>[] StatusCode = new TreeMap[24];
@Override
protected void setup(Context context) throws IOException, InterruptedException {
    //task1
    for (int i = 0; i < 24; i++) {
        StatusCode[i] = new TreeMap<String, Integer>();
    }
}

```

```

    }
}

```

```

String part[] = key.toString().split("\\:");
//task1
if (part[0].equals("1")) {
    int sum = 0;
    for (Text x : values) {
        sum += Integer.parseInt(x.toString());
    }
    if (Status.containsKey(part[1])) {
        int times = Status.get(part[1]);
        times += sum;
        Status.put(part[1], times);
    }
    else
        Status.put(part[1], sum);
    int hour = Integer.parseInt(part[2]);
    StatusCode[hour].put(part[1], sum);
}

```

```

@Override
protected void cleanup(Context context) throws IOException, InterruptedException {
    //task1
    //1:keyvalue
    for (String s : Status.keySet()) {
        context.write(new Text("1:" + s + ":" + String.valueOf(Status.get(s))), null);
    }
    for (int i = 0; i < 24; i++) {
        String hour = String.valueOf(i) + ":00-" + String.valueOf((i + 1) % 24) +
        ":00";

        String info = "";
        for (String s : Status.keySet()) {
            String buf;
            if (StatusCode[i].containsKey(s))
                buf = s + ":" + String.valueOf(StatusCode[i].get(s));
            else
                buf = s + ":0";
            info += buf + ' ';
        }
        context.write(new Text("1:" + hour + ' ' + info), null);
    }
}

```

任务 2, 3, 4: 输出的 key 值加上 “2: IP|”、“3: url|”, “4: url|” 用于输出前提取文件名
任务 2,3 较简单, 过程与 4 相似, 此处不附加代码

任务 4: 先计算每个时间窗的平均响应时间, 再计算总的平均响应时间。

```
//task4
if (part[0].equals("4")) {
    int[] hourtimes = new int[24];
    int[] hourlong = new int[24];
    double[] houravg = new double[24];
    for (int i = 0; i < 24; i++) {
        hourtimes[i] = 0;
        hourlong[i] = 0;
    }
    int sumtimes = 0;
    int sumlong = 0;
    double sumavg = 0;
    for (Text x : values) {
        sumtimes++;
        String[] val = x.toString().split(":");
        int hour = Integer.parseInt(val[0]);
        int Long = Integer.parseInt(val[1]);
        sumlong += Long;
        hourtimes[hour]++;
        hourlong[hour] += Long;
    }
    sumavg = (double) sumlong / (double) sumtimes;
    for (int i = 0; i < 24; i++) {
        houravg[i] = (double) hourlong[i] / (double) hourtimes[i];
    }
    //key: 4:url[keyvalue]
    String prefix = "4:" + part[1] + ":";
    context.write(new Text(prefix + part[1] + ":" + String.format("%.2f", sumavg)), null);
    for (int i = 0; i < 24; i++) {
        if (hourtimes[i] != 0) {
            String hour = String.valueOf(i) + ":00-" + String.valueOf((i + 1) % 24) + ":00";
            String buf = prefix + hour + " " + String.format("%.2f", houravg[i]);
            context.write(new Text(buf), null);
        }
    }
}
```

5.1.4 多文件输出

参考博客 <http://blog.csdn.net/liuzhoulong/article/details/7743840>

首先需要构造一个自己的 `MultipleOutputFormat` 类实现 `FileOutputFormat` 类 (注意是 `org.apache.hadoop.mapreduce.lib.output` 包的 `FileOutputFormat`)

接着还需要自定义一个 `LineRecordWriter` 实现记录写入器 `RecordWriter` 类, 自定义输出格式。

接着, 实现刚刚重写 `MultipleOutputFormat` 类中的 `generateFileNameForKeyValue` 方法自定义返回需要输出文件的名称, 我这里是 key 值中以逗号分割取第一个字段的值作为输出文件名, 这样第一个字段值相同的会输出到一个文件中并以其值作为文件名。


```

public class VVLogNameMultipleTextOutputFormat extends MultipleOutputFormat<Text,
NullWritable> {
    @Override
    protected String generateFileNameForKeyValue(Text key, NullWritable value,
Configuration conf) {
        String Path1=conf.get("1");
        String Path2=conf.get("2");
        String Path3=conf.get("3");
        String Path4=conf.get("4");
        String filename;
        if(key.toString().charAt(0)=='1')
            filename = Path1+"/1.txt";
        else if(key.toString().charAt(0)=='2') {
            int index = key.toString().indexOf('|');
            filename = Path2+"/"+key.toString().substring(2,index)+".txt";
        }
        else if(key.toString().charAt(0)=='3') {
            int index = key.toString().indexOf('|');
            String bufname = key.toString().substring(3,index);
            bufname=bufname.replace('/', '-');
            filename = Path3+"/"+bufname+".txt";
        }
        else if(key.toString().charAt(0)=='4') {
            int index = key.toString().indexOf('|');
            String bufname = key.toString().substring(3,index);
            bufname=bufname.replace('/', '-');
            filename = Path4+"/"+bufname+".txt";
        }
        else
            filename = "error";
        return filename;
    }
}

```

代码中使用的 Configuration.get 方法获取的值在 main 函数中初始 configuration 时通过 Configuration.set 设置。这种方法可用于少量信息的共享。

```

Configuration conf = new Configuration();
conf.set("1", args[1]);
conf.set("2", args[2]);
conf.set("3", args[3]);
conf.set("4", args[4]);

```

此外，为了获得文件名信息，context 输出的 key 中含有前缀信息，但是不需要输出，所以在自定义的 LineRecord 中重写 writeObject 方法去除前缀信息。

```

private void writeObject(Object o) throws IOException {
    if (o instanceof Text) {
        Text to = (Text) o;
    }
}

```

```

        if(to.toString().charAt(0)=='1')
            to = new Text(to.toString().substring(2));
        else {
            int index = to.toString().indexOf('|');
            to = new Text(to.toString().substring(index+1));
        }
        out.write(to.getBytes(), 0, to.getLength());
    } else {
        out.write(o.toString().getBytes(utf8)); //!!!!!!!!!!!!
    }
}

```

5.2 Part2

5.2.1 对日志信息分词

操作同 5.1.1。

5.2.2 Map

```

public static class MyMapper extends Mapper<Object, Text, Text, Text> {
    @Override
    protected void map(Object key, Text value, Mapper.Context context) {
        try {
            String Onelog = value.toString().trim();
            String partwords[] = Split.splitOnelog(Onelog);
            InputSplit inputSplit = context.getInputSplit();
            String filename = ((FileSplit)inputSplit).getPath().getName();
            if (partwords[2] != null && partwords[1] != null) {
                int index = partwords[1].indexOf(':');
                String hour = partwords[1].substring(index + 1, index + 3);
                String buf[] = partwords[2].split(" ");
                String url = buf[1];
                String task5str = hour + ":" + url + ":" + filename;
                Text keyInfo = new Text(task5str);
                context.write(keyInfo, new Text("1"));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

5.2.3 Combiner

```

public static class MyCombiner extends Reducer<Text,Text,Text,Text>{
    protected void reduce(Text key, Iterable<Text>values ,Context context){
        try {
            int sum = 0;
            for (Text x : values) {
                sum += Integer.parseInt(x.toString());
            }
        }
    }
}

```

```

    }
    Text valueInfo = new Text(Integer.toString(sum));
    context.write(key, valueInfo);
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

5.2.4 Reduce

主要数据结构，其中 `dateList` 是一个二维数组，每一个元素都是一个 `HashMap`，存储着每个文件每个时间段下的所有 `url` 及其出现次数；`predict22` 是一维数组，存储了预测值，`result` 是一个与 `predict22` 结构相同的数组，存储了 22 日的真实值。

```

private static HashMap<String, Integer>[][] dateList = new HashMap[fileNum][hourNum];
//predict value of date 09-22
private static HashMap<String, Integer>[] predict22 = new HashMap[24];
//true value of date 09-22
private static HashMap<String, Integer>[] result = new HashMap[24];

```

获取文件名的 8-10 个字符，获得 log 的日期，将 8-21 日的数据存入 `dataList` 中，将 22 日的数据存入 `result` 中。

```

protected void reduce(Text key, Iterable<Text> values, Context context) {
    try {
        String part[] = key.toString().split("\\:");
        int hour = Integer.parseInt(part[0]);
        String url = part[1];
        String filename = part[2];
        // context.write(new Text("filename:"+filename),null);
        int date = Integer.parseInt(filename.substring(8,10));
        int sum = 0;
        Iterator it = values.iterator();
        while (it.hasNext()) {
            Text x = (Text) it.next();
            sum += Integer.parseInt(x.toString());
        }
        if (date < 22) {
            dateList[date - 8][hour].put(url, Integer.valueOf(sum));
        }
        else if (date == 22) {
            result[hour].put(url, Integer.valueOf(sum));
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

5.2.5 预测

预测函数中主要的数据结构如下，`urlTimes` 记录了 14 个文件中每个时间段下各 url 的出现次数；`averageTimes` 记录了 14 个文件中每个时间段下各 url 出现的平均次数；`varienceTimes` 记录了 14 个文件中每个时间段下各 url 出现次数的方差；`maxTime` 和 `minTime` 分别记录了 14 个文件中每个时间段下各 url 出现次数的最大值和最小值；`changeRate` 记录了比率值（最大值-最小值）/最小值。

```
private static HashMap<String, Integer>[] urlTimes = new HashMap[24];
private static HashMap<String, Double>[] averageTimes = new HashMap[24];
private static HashMap<String, Double>[] varienceTimes = new HashMap[24];
private static HashMap<String, Integer>[] maxTime = new HashMap[24];
private static HashMap<String, Integer>[] minTime = new HashMap[24];
private static HashMap<String, Double>[] changeRate = new HashMap[24];
```

`computeAverage` 函数计算了 14 个文件中，每个时间段下各 url 出现的平均次数，`computeVolatility` 函数中计算了 14 个文件中，每个时间段下各 url 出现的最大值、最小值以及比率（最大值-最小值）/最小值。

```
public static HashMap<String,Integer>[] predictNew(int hourNum, int fileNum,
HashMap<String,Integer>dateList[][]) {
    init(hourNum,fileNum,dateList);
    computeAverage();
    computeVolatility();
    predictDate22();
    return predict22;
}
```

`predictDate22` 为预测算法的核心函数，若方差大于 1000000 并且比率大于 1，则判定为数据波动性太大，按照最近最优原则，选取离 22 日最近的 url 次数作为预测值；否则认为数据波动很小，取平均值作为 22 日的预测值。

```
private static void predictDate22() {
    for (int i = 0; i < hourNum; i++) {
        predict22[i].clear();
        Iterator it = varienceTimes[i].keySet().iterator();
        while (it.hasNext()) {
            String url = (String) it.next();
            double sum = varienceTimes[i].get(url);
            double rate = changeRate[i].get(url);
            if (sum > 1000000 && rate > 1) {
                int index = 13;
                while(!dateList[index][i].containsKey(url)) {
                    index--;
                    if(index == -1) break;
                }
                if (index != -1) {
                    predict22[i].put(url, dateList[index][i].get(url));
                }
            } else {
                int ave = new Double(averageTimes[i].get(url)).intValue();
                double delt = averageTimes[i].get(url)-ave;
```

```

        predict22[i].put(url,delt > 0.5 ? ave+1 : ave);
    }
}
else {
    int ave = new Double(averageTimes[i].get(url)).intValue();
    double delt = averageTimes[i].get(url)-ave;
    predict22[i].put(url,delt > 0.5 ? ave+1 : ave);
}
}
}
}

```

在 cleanup 中调用预测函数，得出预测结果，并将其写入文件。

```

protected void cleanup(Context context) throws IOException, InterruptedException {
    // predict22=Predict.predict(hourNum,fileNum,dateList);
    predict22=Predict.predictNew(hourNum,fileNum,dateList);
    double RMSE = computeRMSE();
    context.write(new Text("RMSE:"+String.valueOf(RMSE)),null);
    for (int i = 0; i < hourNum; i++) {
        String hour = String.valueOf(i) + ":00-" + String.valueOf((i+1)%24) + ":00";
        String task5str = hour;
        Iterator it = predict22[i].keySet().iterator();
        while(it.hasNext()) {
            String url = (String) it.next();
            task5str += " " + url + ":" + String.valueOf(predict22[i].get(url));
        }
        context.write(new Text(task5str),null);
    }
}

```

5.2.6 计算 RMSE

```

public double computeRMSE() {
    double RMSE = 0;
    for(int i = 0; i < hourNum; i++) {
        int numOfPre = predict22[i].size();
        int numOfRes = result[i].size();
        double sum = 0;
        Iterator it = result[i].keySet().iterator();
        while(it.hasNext()) {
            String url = (String) it.next();
            int trueValue = result[i].get(url);
            int preValue = 0;
            if(predict22[i].containsKey(url)) {
                preValue = predict22[i].get(url);
            }
            sum += Math.pow(preValue-trueValue,2);
        }
    }
    RMSE = Math.sqrt(sum/numOfRes);
}

```

```

    }
    sum = sum / numOfRes;
    RMSE += Math.sqrt(sum);
  }
  RMSE = RMSE / hourNum;
  return RMSE;
}

```

6 优化

6.1 Part1

上述实现将 4 个任务放入一个 mapreduce 任务，显然比分别执行 4 个 mapreduce 任务效率高。

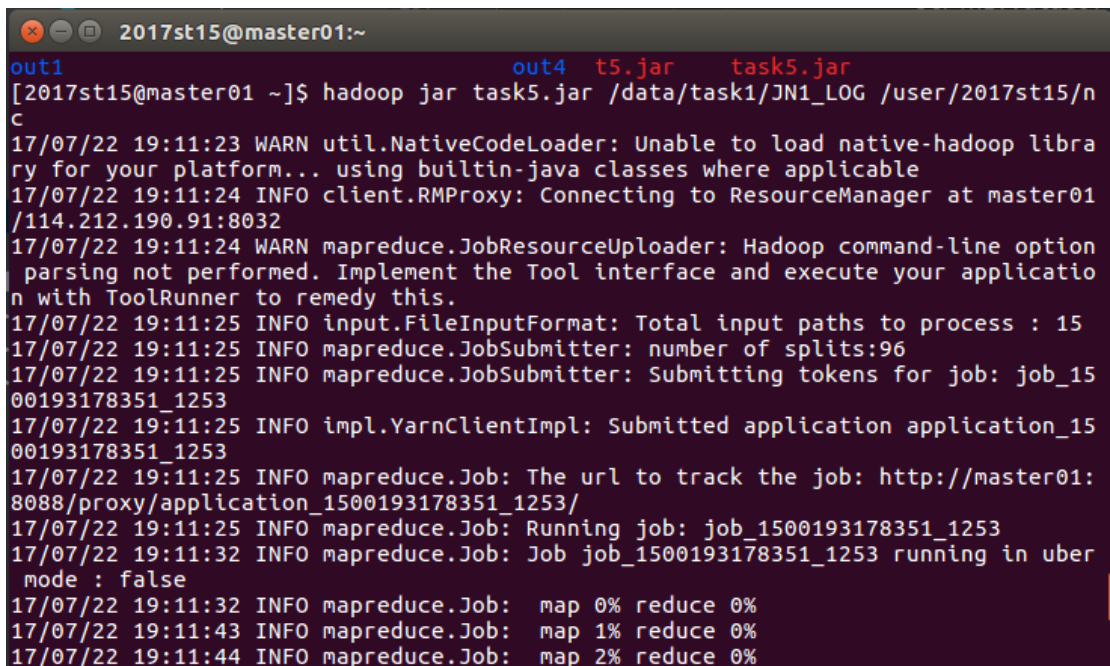
尝试过添加 Combiner，但由于在多文件输出处理过程中使用了重写的 FileOutputFormat，Combiner 的输出会影响 Reduce 的输入结果，的确可以修改 writeObject 方法避免这个影响，但是过于繁琐；本来输入数据量就不大，效果不会很明显，所以没有对此进行优化。

6.2 Part2

6.2.1 添加 Combiner

一开始程序未添加 Combiner，task5 在运行时 MapReduce 的时间大概 10 分钟以上，加入 Combiner 后运行时间缩短为 2 分钟左右，大大提高了效率。

下图为优化前的运行时间：



```

2017st15@master01:~
out1 out4 t5.jar task5.jar
[2017st15@master01 ~]$ hadoop jar task5.jar /data/task1/JN1_LOG /user/2017st15/n
c
17/07/22 19:11:23 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
17/07/22 19:11:24 INFO client.RMProxy: Connecting to ResourceManager at master01
/114.212.190.91:8032
17/07/22 19:11:24 WARN mapreduce.JobResourceUploader: Hadoop command-line option
 parsing not performed. Implement the Tool interface and execute your applicatio
n with ToolRunner to remedy this.
17/07/22 19:11:25 INFO input.FileInputFormat: Total input paths to process : 15
17/07/22 19:11:25 INFO mapreduce.JobSubmitter: number of splits:96
17/07/22 19:11:25 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_15
00193178351_1253
17/07/22 19:11:25 INFO impl.YarnClientImpl: Submitted application application_15
00193178351_1253
17/07/22 19:11:25 INFO mapreduce.Job: The url to track the job: http://master01:
8088/proxy/application_1500193178351_1253/
17/07/22 19:11:25 INFO mapreduce.Job: Running job: job_1500193178351_1253
17/07/22 19:11:32 INFO mapreduce.Job: Job job_1500193178351_1253 running in uber
mode : false
17/07/22 19:11:32 INFO mapreduce.Job: map 0% reduce 0%
17/07/22 19:11:43 INFO mapreduce.Job: map 1% reduce 0%
17/07/22 19:11:44 INFO mapreduce.Job: map 2% reduce 0%

```

```

2017st15@master01:~
17/07/22 19:21:42 INFO mapreduce.Job: map 100% reduce 94%
17/07/22 19:21:51 INFO mapreduce.Job: map 100% reduce 95%
17/07/22 19:21:57 INFO mapreduce.Job: map 100% reduce 96%
17/07/22 19:22:06 INFO mapreduce.Job: map 100% reduce 97%
17/07/22 19:22:16 INFO mapreduce.Job: map 100% reduce 98%
17/07/22 19:22:25 INFO mapreduce.Job: map 100% reduce 99%
17/07/22 19:22:34 INFO mapreduce.Job: map 100% reduce 100%
17/07/22 19:22:35 INFO mapreduce.Job: Job job_1500193178351_1253 completed successfully
17/07/22 19:22:35 INFO mapreduce.Job: Counters: 51
    File System Counters
        FILE: Number of bytes read=17072326
        FILE: Number of bytes written=45177811
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=12052567186
        HDFS: Number of bytes written=20377
        HDFS: Number of read operations=291
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=2
    Job Counters
        Killed map tasks=2
        Launched map tasks=98

```

共 11 分钟。

下图为优化后的运行时间：

```

2017st15@master01:~
[2017st15@master01 ~]$ hadoop jar task5.jar /data/task1/JN1_LOG /user/2017st15/output5
17/07/22 19:26:01 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
17/07/22 19:26:02 INFO client.RMProxy: Connecting to ResourceManager at master01/114.212.190.91:8032
17/07/22 19:26:02 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
17/07/22 19:26:02 INFO input.FileInputFormat: Total input paths to process : 15
17/07/22 19:26:03 INFO mapreduce.JobSubmitter: number of splits:96
17/07/22 19:26:03 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1500193178351_1254
17/07/22 19:26:03 INFO impl.YarnClientImpl: Submitted application application_1500193178351_1254
17/07/22 19:26:03 INFO mapreduce.Job: The url to track the job: http://master01:8088/proxy/application_1500193178351_1254/
17/07/22 19:26:03 INFO mapreduce.Job: Running job: job_1500193178351_1254
17/07/22 19:26:09 INFO mapreduce.Job: Job job_1500193178351_1254 running in uber mode : false
17/07/22 19:26:09 INFO mapreduce.Job: map 0% reduce 0%
17/07/22 19:26:21 INFO mapreduce.Job: map 1% reduce 0%
17/07/22 19:26:22 INFO mapreduce.Job: map 2% reduce 0%
17/07/22 19:26:23 INFO mapreduce.Job: map 3% reduce 0%

```



```

2017st15@master01:~
17/07/22 19:27:42 INFO mapreduce.Job: map 88% reduce 27%
17/07/22 19:27:43 INFO mapreduce.Job: map 90% reduce 27%
17/07/22 19:27:44 INFO mapreduce.Job: map 93% reduce 27%
17/07/22 19:27:45 INFO mapreduce.Job: map 96% reduce 29%
17/07/22 19:27:46 INFO mapreduce.Job: map 98% reduce 29%
17/07/22 19:27:48 INFO mapreduce.Job: map 99% reduce 32%
17/07/22 19:27:50 INFO mapreduce.Job: map 100% reduce 32%
17/07/22 19:27:51 INFO mapreduce.Job: map 100% reduce 33%
17/07/22 19:27:54 INFO mapreduce.Job: map 100% reduce 100%
17/07/22 19:27:54 INFO mapreduce.Job: Job job_1500193178351_1254 completed successfully
17/07/22 19:27:54 INFO mapreduce.Job: Counters: 51
    File System Counters
        FILE: Number of bytes read=63258
        FILE: Number of bytes written=11445532
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=12052567186
        HDFS: Number of bytes written=20377
        HDFS: Number of read operations=291
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=2
    Job Counters

```

共 1 分 40 秒左右。

6.2.2 预测优化

一开始观察训练数据，发现数据大体趋势是：越靠近 22 日的数据与 22 日数据的相似度越大，所以算法采取加权平均的思路，对 21 日的的数据赋 0.5 的权重，对 20 日的的数据赋 0.5^2 的权值……对 9 日的的数据赋 0.5^{13} 的权值，对 8 日的的数据赋 0.5^{13} 的权值，保证权值和为 1，经过计算后得出 RMSE 为 4230，结果不是很理想。

再次对数据分析并调整了算法后按照 5.2.5 中算法得到 RMSE 为 3340，结果得到了很大的改善。

下图为优化前的结果截图：

```

2017st15@master01:~/test5
RMSE:4230.34482871174
0:00-1:00 /tour/hotel-search/nearby-scenic/query:467 /tour/hotel-search/query:65
75 /tour/category/ids/query:11485 /tour/guide/query:21609 /tour/flight-ticket/qu
ery:1 /tour/phoenix/product/query:7099 /tour/product/query:2286 /tour/poi/query/
queryCategory:232 /tour/category/scenic/query:9 /tour/poi/query/queryNumberFound
:1 /tour/hotel/query:1 /tour/guide/update:4 /tour/category/query:116981 /tour/ca
tegory/vendor-statis/query:1180 /tour/category/tuniu-hot/query:0 /tour/category/
weekendproduct/query:541 /tour/poi/query/queryScenicNumPerCity:71 /tour/poi/quer
y/queryScenicTypeList:0 /tour/poi/query:121 /tour/faq/query:2 /tour/category/sta
tis/query:510 /tour/poi/scenictype/provincelist/query:0 /tour/suggestion/query:3
878 /tour/poi/query/queryScenicSpotCount:7765 /tour/poi/query/queryProvinceList:
0 /tour/hotelSuggestion/query:48
1:00-2:00 /tour/hotel-search/nearby-scenic/query:453 /tour/hotel-search/query:53
95 /tour/category/ids/query:14347 /tour/guide/query:22504 /tour/flight-ticket/qu
ery:0 /tour/phoenix/product/query:14120 /tour/product/query:2099 /tour/poi/query
/queryCategory:155 /tour/category/scenic/query:9 /tour/poi/query/queryNumberFoun
d:2 /tour/hotel/query:1 /tour/em-task/exec:1 /tour/guide/update:5 /tour/category
/query:129795 /tour/category/vendor-statis/query:787 /tour/category/tuniu-hot/qu
ery:0 /tour/category/weekendproduct/query:297 /tour/poi/query/queryScenicNumPerC
ity:76 /tour/poi/query/queryScenicTypeList:1 /tour/poi/query:128 /tour/faq/query
:0 /tour/category/statis/query:450 /tour/suggestion/query:2057 /tour/poi/query/q
ueryScenicSpotCount:7282 /tour/poi/query/queryProvinceList:0 /tour/hotelSuggesti
on/query:38
1,1 Top

```


7 性能分析

task1-task4, 运行时间大约为 5 分钟; task5 运行时间大约为 2 分钟。

在预测时, 设 M 为文件数量 (此实验中为 15), H 为小时数 (此实验中为 24), U 为某文件在某时段下的最大 url 数量, 则预测的时间复杂度与空间复杂度均为 $O(M*H*U)$ 。

8 运行截图

8.1 将jar从本机hadoop拷贝到集群

```
hadoop@ubuntu:~$ scp -r task1to4.jar 2017st15@114.212.190.91:~/
2017st15@114.212.190.91's password:
task1to4.jar                                100%  23KB  22.9KB/s   00:00
hadoop@ubuntu:~$ scp -r task5.jar 2017st15@114.212.190.91:~/
2017st15@114.212.190.91's password:
task5.jar                                    100%  23KB  23.0KB/s   00:00
hadoop@ubuntu:~$ ssh 2017st15@114.212.190.91
2017st15@114.212.190.91's password:
Last login: Sat Jul 22 20:11:15 2017 from 172.26.98.7
[2017st15@master01 ~]$
```

将 hdfs 文件拷贝到集群用户本地

```
[2017st15@master01 ~]$ hadoop fs -get /user/2017st15/out1
17/07/22 20:36:35 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
using builtin-java classes where applicable
17/07/22 20:36:36 WARN hdfs.DFSClient: DFSInputStream has been closed already
17/07/22 20:36:36 WARN hdfs.DFSClient: DFSInputStream has been closed already
```

8.2 执行任务1-4的jar包

```
2017st15@master01:~
[2017st15@master01 ~]$ hadoop jar task1to4.jar /data/task1/JN1_LOG/2015-09-08.log
/user/2017st15/out1 /user/2017st15/out2 /user/2017st15/out3 /user/2017st15/out4
17/07/22 20:14:13 WARN util.NativeCodeLoader: Unable to load native-hadoop library
for your platform... using builtin-java classes where applicable
17/07/22 20:14:14 INFO client.RMProxy: Connecting to ResourceManager at master01
/114.212.190.91:8032
17/07/22 20:14:15 WARN mapreduce.JobResourceUploader: Hadoop command-line option
parsing not performed. Implement the Tool interface and execute your applicatio
n with ToolRunner to remedy this.
17/07/22 20:14:15 INFO input.FileInputFormat: Total input paths to process : 1
17/07/22 20:14:15 INFO mapreduce.JobSubmitter: number of splits:8
17/07/22 20:14:15 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_15
00193178351_1256
17/07/22 20:14:16 INFO impl.YarnClientImpl: Submitted application application_15
00193178351_1256
17/07/22 20:14:16 INFO mapreduce.Job: The url to track the job: http://master01:
8088/proxy/application_1500193178351_1256/
17/07/22 20:14:16 INFO mapreduce.Job: Running job: job_1500193178351_1256
17/07/22 20:14:22 INFO mapreduce.Job: Job job_1500193178351_1256 running in uber
mode : false
17/07/22 20:14:22 INFO mapreduce.Job: map 0% reduce 0%
17/07/22 20:14:33 INFO mapreduce.Job: map 15% reduce 0%
17/07/22 20:14:34 INFO mapreduce.Job: map 26% reduce 0%
```

8.2.1 任务1 结果

```
[2017st15@master01 ~]$ hadoop fs -cat /user/2017st15/out1/1.txt
17/07/22 23:11:26 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
200:10407139
400:2187
404:200
0:00-1:00 200:251835 400:5 404:3
1:00-2:00 200:417259 400:3 404:2
2:00-3:00 200:540932 400:10 404:3
3:00-4:00 200:700470 400:8 404:3
4:00-5:00 200:724782 400:9 404:6
5:00-6:00 200:377239 400:2 404:8
6:00-7:00 200:297716 400:6 404:5
7:00-8:00 200:310149 400:23 404:5
8:00-9:00 200:366177 400:46 404:14
9:00-10:00 200:403949 400:102 404:9
10:00-11:00 200:505813 400:57 404:9
11:00-12:00 200:449269 400:75 404:10
12:00-13:00 200:385128 400:84 404:8
13:00-14:00 200:439463 400:75 404:6
14:00-15:00 200:470003 400:21 404:13
15:00-16:00 200:482620 400:21 404:19
16:00-17:00 200:524890 400:16 404:14
17:00-18:00 200:500347 400:12 404:11
18:00-19:00 200:337237 400:1 404:8
19:00-20:00 200:336770 400:2 404:7
20:00-21:00 200:396570 400:1 404:13
21:00-22:00 200:393214 400:3 404:3
22:00-23:00 200:438812 400:4 404:13
23:00-0:00 200:356495 400:1601 404:8
[2017st15@master01 ~]$
```

8.2.2 任务2 结果

```
[2017st15@master01 ~]$ cd out2/
[2017st15@master01 out2]$ ls
10.10.0.101.txt 172.22.49.26.txt 172.22.49.57.txt 172.22.50.25.txt 172.22.50.95.txt
10.10.0.102.txt 172.22.49.35.txt 172.22.49.66.txt 172.22.50.27.txt 172.30.103.72.txt
10.10.0.109.txt 172.22.49.41.txt 172.22.49.67.txt 172.22.50.28.txt 172.30.36.109.txt
10.10.0.120.txt 172.22.49.43.txt 172.22.49.75.txt 172.22.50.32.txt 172.30.36.150.txt
10.10.0.162.txt 172.22.49.44.txt 172.22.49.83.txt 172.22.50.33.txt 172.30.36.74.txt
10.10.0.165.txt 172.22.49.45.txt 172.22.49.86.txt 172.22.50.34.txt 172.30.37.201.txt
10.10.0.92.txt 172.22.49.46.txt 172.22.49.88.txt 172.22.50.36.txt 172.30.39.191.txt
10.10.0.95.txt 172.22.49.51.txt 172.22.49.89.txt 172.22.50.37.txt 172.30.39.93.txt
10.10.10.135.txt 172.22.49.53.txt 172.22.50.20.txt 172.22.50.38.txt 172.30.46.23.txt
10.10.10.144.txt 172.22.49.54.txt 172.22.50.21.txt 172.22.50.73.txt 172.30.47.150.txt
10.10.10.188.txt 172.22.49.55.txt 172.22.50.22.txt 172.22.50.74.txt 172.30.47.199.txt
10.10.30.189.txt 172.22.49.56.txt 172.22.50.24.txt 172.22.50.94.txt 172.30.53.140.txt
```

```

2017st15@master01:~/out2
172.22.50.95:1249
0:00-1:00 2
8:00-9:00 16
9:00-10:00 69
10:00-11:00 112
11:00-12:00 140
12:00-13:00 48
13:00-14:00 90
14:00-15:00 128
15:00-16:00 208
16:00-17:00 156
17:00-18:00 106
18:00-19:00 54
19:00-20:00 40
20:00-21:00 23
21:00-22:00 25
22:00-23:00 18
23:00-0:00 14

```

8.2.3 任务3 结果

```

[2017st15@master01 ~]$ cd out3/
[2017st15@master01 out3]$ ls
tour-category-ids-query.txt
tour-category-query.txt
tour-category-scenic-query.txt
tour-category-statis-query.txt
tour-category-tuniu-hot-query.txt
tour-category-vendor-statis-query.txt
tour-category-weekendproduct-query.txt
tour-em-task-exec.txt
tour-faq-query.txt
tour-flight-ticket-query.txt
tour-guide-delete.txt
tour-guide-query.txt
tour-guide-update.txt
tour-hotel-query.txt
tour-hotel-search-nearby-scenic-query.txt
tour-hotel-search-query.txt
tour-hotelSuggestion-query.txt
tour-phoenix-product-query.txt
tour-poi-query-queryCategory.txt
tour-poi-query-queryNumberFound.txt
tour-poi-query-queryProvinceList.txt
tour-poi-query-queryScenicNumPerCity.txt
tour-poi-query-queryScenicSpotCount.txt
tour-poi-query-queryScenicTypeList.txt
tour-poi-query.txt
tour-poi-scenictype-provincelist-query.txt
tour-product-query.txt
tour-suggestion-query.txt

```

```

2017st15@master01:~/out3
7/tour/category/query:4845666
08/Sep/2015:00:19:16 11
08/Sep/2015:00:19:17 37
08/Sep/2015:00:19:18 31
08/Sep/2015:00:19:19 31
08/Sep/2015:00:19:20 32
08/Sep/2015:00:19:21 28
08/Sep/2015:00:19:22 40
08/Sep/2015:00:19:23 41
08/Sep/2015:00:19:24 38
08/Sep/2015:00:19:25 37
08/Sep/2015:00:19:26 42
08/Sep/2015:00:19:27 23
08/Sep/2015:00:19:28 45
08/Sep/2015:00:19:29 43
08/Sep/2015:00:19:30 76
08/Sep/2015:00:19:31 41
08/Sep/2015:00:19:32 51
08/Sep/2015:00:19:33 41
08/Sep/2015:00:19:34 42
08/Sep/2015:00:19:35 28
08/Sep/2015:00:19:36 33
08/Sep/2015:00:19:37 38
"tour-category-query.txt" 85640L, 2058710C

```

8.2.4 任务4 结果

```

[2017st15@master01 ~]$ cd out4/
[2017st15@master01 out4]$ ls
tour-category-ids-query.txt
tour-category-query.txt
tour-category-scenic-query.txt
tour-category-statis-query.txt
tour-category-tuniu-hot-query.txt
tour-category-vendor-statis-query.txt
tour-category-weekendproduct-query.txt
tour-em-task-exec.txt
tour-faq-query.txt
tour-flight-ticket-query.txt
tour-guide-delete.txt
tour-guide-query.txt
tour-guide-update.txt
tour-hotel-query.txt
tour-hotel-search-nearby-scenic-query.txt
tour-hotel-search-query.txt
tour-hotelSuggestion-query.txt
tour-phoenix-product-query.txt
tour-poi-query-queryCategory.txt
tour-poi-query-queryNumberFound.txt
tour-poi-query-queryProvinceList.txt
tour-poi-query-queryScenicNumPerCity.txt
tour-poi-query-queryScenicSpotCount.txt
tour-poi-query-queryScenicTypeList.txt
tour-poi-query.txt
tour-poi-scenicType-provincelist-query.txt
tour-product-query.txt
tour-suggestion-query.txt

```

```

2017st15@master01:~/out4
tour/category/query:13.77
0:00-1:00 12.46
1:00-2:00 10.35
2:00-3:00 4.37
3:00-4:00 3.45
4:00-5:00 4.07
5:00-6:00 6.93
6:00-7:00 11.35
7:00-8:00 12.21
8:00-9:00 13.10
9:00-10:00 19.33
10:00-11:00 28.92
11:00-12:00 25.60
12:00-13:00 16.35
13:00-14:00 20.76
14:00-15:00 24.06
15:00-16:00 22.60
16:00-17:00 26.41
17:00-18:00 19.57
18:00-19:00 16.45
19:00-20:00 15.99
20:00-21:00 15.86
21:00-22:00 16.18
22:00-23:00 15.80
23:00-0:00 15.72

```

8.3 执行任务5jar包

```

2017st15@master01:~
[2017st15@master01 ~]$ hadoop jar task5.jar /data/task1/JN1_LOG /user/2017st15/out5
17/07/22 20:20:42 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
17/07/22 20:20:43 INFO client.RMProxy: Connecting to ResourceManager at master01/114.212.190.91:8032
17/07/22 20:20:44 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
17/07/22 20:20:44 INFO input.FileInputFormat: Total input paths to process : 15
17/07/22 20:20:44 INFO mapreduce.JobSubmitter: number of splits:96
17/07/22 20:20:44 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1500193178351_1258
17/07/22 20:20:45 INFO impl.YarnClientImpl: Submitted application application_1500193178351_1258
17/07/22 20:20:45 INFO mapreduce.Job: The url to track the job: http://master01:8088/proxy/application_1500193178351_1258/
17/07/22 20:20:45 INFO mapreduce.Job: Running job: job_1500193178351_1258
17/07/22 20:20:51 INFO mapreduce.Job: Job job_1500193178351_1258 running in uber mode : false
17/07/22 20:20:51 INFO mapreduce.Job:  map 0% reduce 0%
17/07/22 20:21:02 INFO mapreduce.Job:  map 2% reduce 0%
17/07/22 20:21:03 INFO mapreduce.Job:  map 3% reduce 0%
17/07/22 20:21:05 INFO mapreduce.Job:  map 6% reduce 0%

```


8.3.1 结果

```

2017st15@master01:~/out5
RMSE:3340.6354364698086
0:00-1:00 /tour/hotel-search/nearby-scenic/query:738 /tour/hotel-search/query:69
89 /tour/category/ids/query:1118 /tour/guide/query:22179 /tour/flight-ticket/qu
ery:2 /tour/phoenix/product/query:6979 /tour/product/query:1267 /tour/category/s
cenic/query:13 /tour/poi/query/queryCategory:238 /tour/poi/query/queryNumberFou
nd:3 /tour/hotel/query:2 /tour/guide/update:4 /tour/category/query:117871 /tour/c
ategory/tuniu-hot/query:3 /tour/category/vendor-statis/query:1137 /tour/category
/weekendproduct/query:615 /tour/poi/query/queryScenicNumPerCity:55 /tour/poi/que
ry:142 /tour/poi/query/queryScenicTypeList:1 /tour/faq/query:3 /tour/category/st
atis/query:473 /tour/poi/scenictype/provincelist/query:1 /tour/suggestion/query:
3686 /tour/poi/query/queryScenicSpotCount:8010 /tour/hotelSuggestion/query:48 /t
our/poi/query/queryProvincelist:1
1:00-2:00 /tour/hotel-search/nearby-scenic/query:749 /tour/hotel-search/query:48
42 /tour/category/ids/query:14285 /tour/guide/query:22737 /tour/flight-ticket/qu
ery:2 /tour/phoenix/product/query:13473 /tour/product/query:1309 /tour/category
scenic/query:12 /tour/poi/query/queryCategory:151 /tour/poi/query/queryNumberFou
nd:4 /tour/hotel/query:2 /tour/guide/update:6 /tour/em-task/exec:2 /tour/categor
y/query:128390 /tour/category/tuniu-hot/query:1 /tour/category/vendor-statis/que
ry:1125 /tour/category/weekendproduct/query:390 /tour/poi/query/queryScenicNumPe
rCity:66 /tour/poi/query:132 /tour/poi/query/queryScenicTypeList:1 /tour/faq/que
ry:3 /tour/category/statis/query:448 /tour/suggestion/query:1924 /tour/poi/query
/queryScenicSpotCount:7632 /tour/hotelSuggestion/query:27 /tour/poi/query/queryP
rovinceList:1

```

1,1

Top

8.4 集群任务截图

All Applications

Cluster Metrics				User Metrics for dr.who											
Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	Vcores Used	Vcores Total	Vcores Reserved	Active Nodes	Decommissioned Nodes	Last Nodes	Unhealthy Nodes	Rebooted Nodes
1237	0	1237	0	0	0 B	216 GB	0 B	66	0	0	2	0	1	0	0

Scheduler Metrics				Fair Scheduler											
Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Containers Pending	Containers Reserved	Memory Used	Memory Pending	Memory Reserved	Vcores Used	Vcores Pending	Vcores Reserved	Vcores Used	Vcores Pending	Vcores Reserved
0	0	0	0	0	0	0	0 B	0 B	0 B	0	0	0	0	0	0

Application List															
ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking URL					
application_1500193178351_1258	2017st15	task5.jar	MAPREDUCE	root:2017st15	Sat Jul 22 20:20:44 +0800 2017	Sat Jul 22 20:22:19 +0800 2017	FINISHED	SUCCEEDED		History					
application_1500193178351_1256	2017st15	task1to4.jar	MAPREDUCE	root:2017st15	Sat Jul 22 20:14:15 +0800 2017	Sat Jul 22 20:19:13 +0800 2017	FINISHED	SUCCEEDED		History					

运行日志截图

Application application_1500193178351_1256

Kill Application

Application Overview

User: 2017st15
 Name: task1to4.jar
 Application Type: MAPREDUCE
 Application Tags:
 YarnApplicationState: FINISHED
 FinalStatus Reported by AM: SUCCEEDED
 Started: Sat Jul 22 20:14:15 +0800 2017
 Elapsed: 4mins, 57sec
 Tracking URL: History
 Diagnostics:

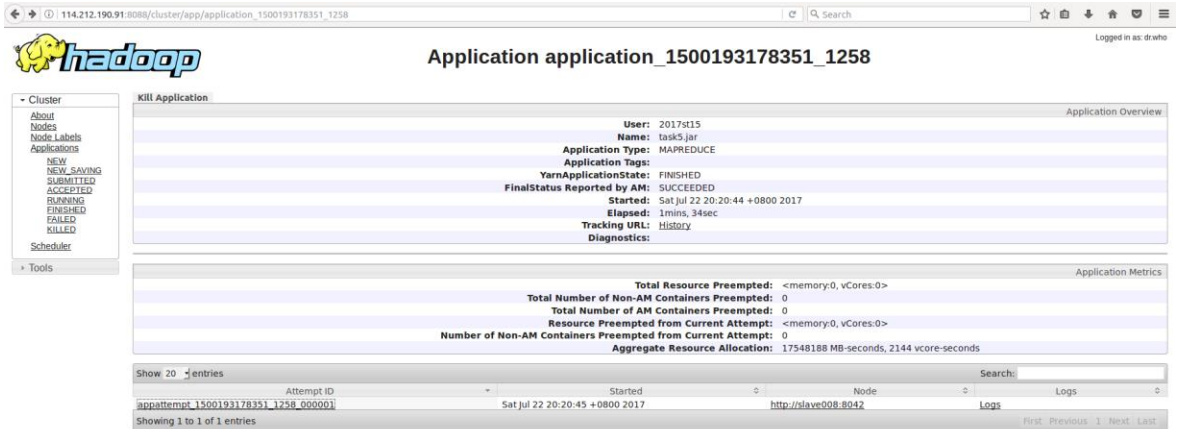
Application Metrics

Total Resource Preempted: <memory:0, vCores:0>
 Total Number of Non-AM Containers Preempted: 0
 Total Number of AM Containers Preempted: 0
 Resource Preempted from Current Attempt: <memory:0, vCores:0>
 Number of Non-AM Containers Preempted from Current Attempt: 0
 Aggregate Resource Allocation: 7870468 MB-seconds, 1106 vcore-seconds

Attempt List					
Attempt ID	Started	Node	Logs	Logs	Logs
appattempt_1500193178351_1256_000001	Sat Jul 22 20:14:15 +0800 2017	http://slave006.8042	Logs		

Showing 1 to 1 of 1 entries

First Previous 1 Next Last



The screenshot displays the Hadoop YARN web interface. The top navigation bar includes the Hadoop logo, the application ID 'application_1500193178351_1258', and a 'Logged in as: drwho' status. A left sidebar contains navigation links for 'Cluster' (About, Nodes, Node Labels, Applications, NEW, NEW_SAVING, SUBMITTED, ACCEPTED, RUNNING, FINISHED, FAILED, KILLED, Scheduler) and 'Tools'. The main content area is titled 'Kill Application' and 'Application Overview'. It shows application details: User: 20174113, Name: task5.jar, Application Type: MAPREDUCE, Application Tags, YarnApplicationState: FINISHED, FinalStatus Reported by AM: SUCCEEDED, Started: Sat Jul 22 20:20:44 +0800 2017, Elapsed: 1mins, 34sec, Tracking URL: History, and Diagnostics. Below this is the 'Application Metrics' section, showing resource preemption statistics. At the bottom is a table of application attempts.

Attempt ID	Started	Node	Logs
appattempt_1500193178351_1258_000001	Sat Jul 22 20:20:45 +0800 2017	http://slave008.8042	Logs

Showing 1 to 1 of 1 entries

9 实验感想.

通过本次大数据暑期项目实践，我们将一学期学习到的大数据理论知识付诸于实践，对大数据的理论知识进行了复习与巩固，在实践的过程中，体会到大数据技术在处理大量重复操作与大量数据时采用的并行技术的优越性，并利用学期中学习到的 hadoop 框架与技术完整的开发了邮件自动分类项目，对于之前学习中存在的盲区与误区，也在这次实验中以遇到问题查阅资料消除问题的方式解决，同时对与大数据技术的认识更加的深刻，这也是在平时里的实验中没有，而利用暑期完整时间进行项目开发所能收获的。而对于 MapReduce 的原理，通过实验，也有了更好的认识。同时，借助这个实验我们体会到所有小组成员一起从头开始，进行项目的可行性分析，提出解决措施，将解决措施实现，并进行结果测试，根据结果进行修改与完善，这一个完整的项目开发过程，受益匪浅。

致谢 感谢为本次实验付出辛苦汗水的老师和助教们。