

# Map-Reduce 课程设计之 日志统计分析

## 1.课程设计目标

本课程设计通过使用MapReduce 来实现日志分析，日志分析在互联网企业应用很广，通过本课程设计的学习，可以进一步了解 MapReduce 技术在工业界的应用。

## 2.学习技能

本次课程设计可以熟悉和掌握以下 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

## 3.题目描述

电商公司越来越重视接口访问日志的利用，从日志文件里边可以获取到接口的访问性能、访问频率、访问来源，统计有以下的意义： 1、 能够快速获取接口访问性能是否下降，或者接口访问频率异常。 2、 结合公司的访问量，可以预估举行促销活动时，需要增加机器的数量。 3、 接口修改后，是否出现波动等。

### 3.1.日志文件结构定义

本题给出的日志文件的格式：

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

具体意义如下表所示：

172.22.49.26	调用方的 IP
[16/Sep/2015:00:22:23 +0800]	调用的时间
GET /tour/category/query HTTP/1.1	HTTP 请求，其中/tour/category/query 是请求的 URL，URL 不同，则视为不同的接口
GET	HTTP METHOD
200	HTTP 状态码，其他经常见到的还有 404， 500
156	RESPONSE 返回的字节长度
2	本次请求响应的时间，单位为毫秒

### 3.2.本题任务

按照以上给定的日志文件格式，按照以下要求进行分析统计 1、统计日志中各个状态码（200, 404, 500）出现总的频次，并且按照小时时间窗，输出各个时间段各状态码的统计情况。统计文件命名为 1.txt 输出格式为：

```
200:100
404:2
500:1
12:00-1:00 200:5 404:1500:0
1:00-2:00 200:5 404:0 500:0
...
```

单词与数字之间英文(:)分割。时间之间英文(-)分割，其他是空格或者空行。

2、统计每个 IP 访问总的频次，并且按照小时时间窗，输出各个时间段各个 IP 访问的情况。每个 IP 的统计信息是一个文件，并且以 IP 为文件名（后缀为 txt，如：172.22.49.26.txt），每个文件的输出格式同任务 1。

3、统计每个接口(请求的 URL)访问总的频次，并且以接口为文件，按照秒为单位的时间窗，输出各个时间段各接口的访问情况。每个接口的统计信息是一个文件，如接口/tour/category/query 的统计文件命名为：tour-category-query.txt，每个文件的输出格式同任务 1。

4、统计每个接口的平均响应时间，并且以接口为分组，按照小时时间窗，输出各个时间段各个接口平均的响应时间。每个接口的统计信息是一个文件，如接口/tour/category/query 的统计文件命名为：tour-category-query.txt，每个文件的输出格式同任务 1。

5、接口访问频次预测，给 2015-09-08.log 到 2015-09-21.log 共 14 天的日志文件，作为训练数据，设计预测算法来预测下一天（2015-09-22）每个小时窗内每个接口（请求的 URL）的访问总频次。输出格式同任务 1。该结果会与当天实际的统计值(2015-09-22.log)做 RMSE 验证。评判标准如下：

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

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。同学们需自己实现 RMSE 公式并进行计算输出 RMSE 值，评分时会根据 RMSE 值判定预测模型的正确度。

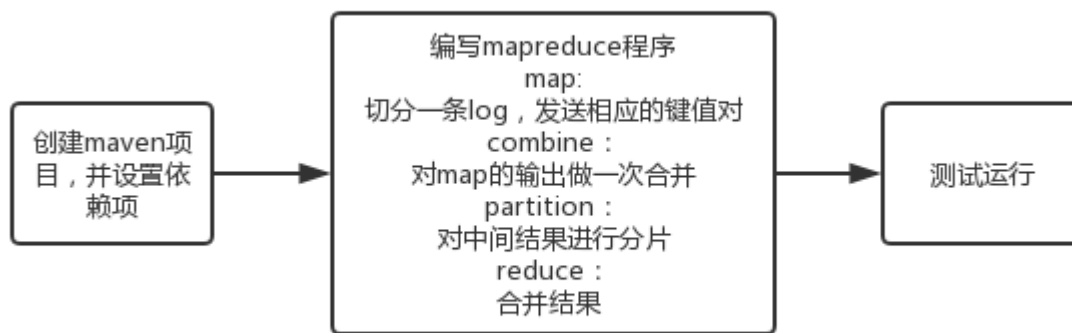
### 3.3 输入文件

输入日志文件均在压缩包 JN1\_LOG.rar 内。任务 1-4 请使用 2015\_09\_08.log 作为输入文件。任务 5 请使用 2015\_09\_08.log 到 2015\_09\_21.log 的日志作为预测的输入文件，并且预测下一天每个小时窗内每个接口的访问总频次并根据预测结果和实际值计算 RMSE。

## 4. 程序设计

### 4.1. 任务 1-4

#### 4.1.1 主要流程



## 4.1.2 主要算法

### 4.1.2.1 文件输入

使用TextInputFormat类输入文件

```
job.setInputFormatClass(TextInputFormat.class);
```

### 4.1.2.2 Map

```
172.22.49.44 [08/Sep/2015:00:19:16 +0800] "GET /tour/category/query HTTP/1.1" GET 200 124 8
172.22.49.43 [08/Sep/2015:00:19:16 +0800] "GET /tour/category/query HTTP/1.1" GET 200 6304 11
172.22.49.88 [08/Sep/2015:00:19:16 +0800] "GET /tour/hotel-search/nearby-scenic/query HTTP/1.1" GET 200 192 2
```

由以上可知一条log的各个元素之间用空格分隔，所以以空格为标志，将log切分

```
String[] log = value.toString().split(" ");
```

一条log内有9个空格，可以切分成10片

切分后如果为10片，则继续操作

```
if (log.length == 10)
```

log[0]是ip

```
/*log_ip*/
String log_ip = log[0];
```

log[1]是日期时间，在一个文件中，日期是固定的，所以只取时分秒

```
String log_time = log[1];
log_time = log_time.substring(13);
```

log[2]是时区，去掉多余的右中括号

```
String log_time_zone = log[2];
    log_time_zone = log_time_zone.substring(0, 4);
```

log[3]是http请求是get, 去掉多余的引号

```
String log_http_method_1 = log[3];
    log_http_method_1 = log_http_method_1.substring(1);
```

log[4]是请求的URL, 也就是接口, 按照题目要求, 将"/"替换为"-"

```
String log_url = log[4];
    log_url = log_url.substring(1);
    log_url = log_url.replaceAll("/", "-");
```

log[5]是HTTP/1.1的请求, 去掉多余引号

```
String log_http = log[5];
    log_http = log_http.substring(0, 7);
```

log[6]是get

```
/*log_http_method_2    get */
    String log_http_method_2 = log[6];
```

log[7]是返回的状态码 (200, 404, 500)

```
/*log_status_code    200 404 500 */
    String log_status_code = log[7];
```

log[8]是返回的字符长度

```
/*log_response_length    byte length*/
    String log_response_length = log[8];
```

log[9]是响应时间

```
/*log_response_time    */
    String log_response_time = log[9];
```

发送键值对:

#### 4.1.2.2.1.任务1

既要求统计各个状态码出现的总频次, 也要求按照小时时间窗输出状态码

设定出现一次, 则为一个频次(value)

```
Text value_1 = new Text("1");
```

状态码总频次的时间用00:00-00:00表示，发送键值为：

1\_00:00-00:00\_log\_status\_code(log\_status\_code 分别表示200 404 500) 频次 1

都是Text类型

```
Text key_all_log_status_code = new Text("1" + "_" + "00" + ":" + "00" + "-" + "00" + ":" + "00" + "_" + log_status_code);
```

其他的将时间转化为消失时间窗表示

```
private String Get_Format_Time(String time) {
    time = time.split(":")[0];
    int first_time = Integer.parseInt(time);
    int second_time = first_time + 1;
    /*if second_time is 24:** we need change it to 00:** */
    if (second_time == 24) {
        second_time = 0;
    }
    /*return formatting time */
    return (String.format("%02d", first_time) + ":" + "00" + "-" +
String.format("%02d", second_time) + ":" + "00");
}
```

小时时间窗时 key为(log\_status\_code 分别表示200 404 500):

```
Text key_log_status_code = new Text("1" + "_" + format_time + "_" + log_status_code);
```

发送：

```
task1
    context.write(key_all_log_status_code, value_1);
    context.write(key_log_status_code, value_1);
```

#### 4.1.2.2.2.任务2

ip 也要按照总频次和时间窗发送

```
task2
    Text key_ip = new Text("2" + "_" + log_ip);
    Text key_ip_time = new Text("2_" + log_ip + "_" + format_time);

    context.write(key_ip, value_1);
    context.write(key_ip_time, value_1);
```

#### 4.1.2.2.3.任务3

```

Text key_url = new Text("3_" + log_url);
Text key_url_time = new Text("3_" + log_url + "_" + log_time);
task3
context.write(key_url, value_1);
context.write(key_url_time, value_1);

```

#### 4.1.2.2.4.任务4

```

task4
Text key_4_url = new Text("4_" + log_url);
Text key_4_url_time = new Text("4_" + log_url + "_" + format_time);
Text value_response_time = new Text(log_response_time + "_" + "1");

context.write(key_4_url, value_response_time);
context.write(key_4_url_time, value_response_time);

```

#### 4.1.2.3 Combine

对key以"\_"为标志切分

```

String[] key1 = key.toString().split("_");

```

任务1-3 相同的key,则将value相加发送

```

if (task == 1 || task == 2 || task == 3) {
    int sum = 0;
    for (Text t : values) {
        int index;
        index = Integer.parseInt(t.toString());
        sum = sum + index;
    }
    Text c_sum;
    c_sum = new Text("" + sum);
    context.write(key, c_sum);
}

```

任务4 计算响应时间均值, key相同均值 和个数发送

```

else {
    int index = 0;
    double time = 0;
    for (Text t : values) {
        time += (Double.parseDouble(t.toString().split("_")[0]) *
(Integer.parseInt(t.toString().split("_")[1])));
        index = index + Integer.parseInt(t.toString().split("_")[1]);
    }
    Text c_sum;
    c_sum = new Text("" + time / index + "_" + index);
    context.write(key, c_sum);
}

```

#### 4.1.2.4 partition

将key 值以“\_”分割，如果分割后个数为 2，则直接按 key 值的默认的哈希值划分；若分隔后 个数为 3，则按 key 值中的前两个信息的 hash 值划分，以确保相同输出文件中的键值对被划分到 同一个 reduce 节点。

```
public static class LogAnalysisPartitioner extends HashPartitioner<Text, Text> {
    @Override
    public int getPartition(Text key, Text value, int sum) {
        System.out.println("Start partitioner");

        if (key.toString().split("_").length == 2)
            return super.getPartition(key, value, sum);
        else
            return super.getPartition(new Text(key.toString().split("_")[0] + "_" +
            key.toString().split("_")[1]), value, sum);
    }
}
```

#### 4.1.2.5 reduce

##### 4.1.2.5.1.输出

输出使用Multipleoutput,设置输出路径为outputpath

```
private MultipleOutputs<Text, Text> Mul_Out_put;
private String[] OutputPath;

@Override
public void setup(Context context) {
    Configuration conf = context.getConfiguration();
    OutputPath = new String[4];
    OutputPath[0] = conf.get("outputPath1");
    OutputPath[1] = conf.get("outputPath2");
    OutputPath[2] = conf.get("outputPath3");
    OutputPath[3] = conf.get("outputPath4");
    Mul_Out_put = new MultipleOutputs<Text, Text>(context);
}
```

##### 4.1.2.5.2 reduce

用“\_”作为标志切片，第一个片标志任务号

```
int task = Integer.parseInt(key.toString().split("_")[0]);
```

对于任务1，分别计算200 404 500的个数

```
private Integer flag1 = 0;
private Integer flag2 = 0;
private Integer flag3 = 0;
```

统计各个总频次和各个时间窗的频次，

```
int sum_1 = 0;
    for (Text v : values)
        sum_1 =sum_1+ Integer.parseInt(v.toString());
```

计算出各个状态码的频次

```
if (key.toString().split("_")[2].compareTo("200") == 0)
    flag1 = sum_1;
else if (key.toString().split("_")[2].compareTo("404") == 0)
    flag2 = sum_1;
else if (key.toString().split("_")[2].compareTo("500") == 0)
    flag3 = sum_1;
flag = key.toString().split("_")[1];
```

写道文件中

```
Text Status_Code_200_flag=new Text("200" + ":");
Text Status_Code_404_flag=new Text("404" + ":");
Text Status_Code_500_flag=new Text("500" + ":");
Text Status_Code_200_Count=new Text(flag1.toString());
Text Status_Code_404_Count=new Text(flag2.toString());
Text Status_Code_500_Count=new Text(flag3.toString());
Text Status_Code =new Text(" 200:" + flag1.toString() + " 404:" +
flag2.toString() + " 500:" + flag3.toString());
    if (flag.compareTo("00:00-00:00") == 0) {
        Mul_Out_put.write("Task1", Status_Code_200_flag, Status_Code_200_Count,
OutputPath[0] + "/" + "1");
        Mul_Out_put.write("Task1", Status_Code_404_flag, Status_Code_404_Count,
OutputPath[0] + "/" + "1");
        Mul_Out_put.write("Task1", Status_Code_500_flag, Status_Code_500_Count,
OutputPath[0] + "/" + "1");

    } else {
        Mul_Out_put.write("Task1", new Text(flag), Status_Code, OutputPath[0] +
"/1");
    }
    Mul_Out_put.close();
}
```

任务234 统计，同key的value而相加发送，并且区分总频次和分时间窗写入文件

```
if (task == 2) {
    int sum2 = 0;
    for (Text t : values)
        sum2 += Integer.parseInt(t.toString());
    if (key.toString().split("_").length == 2) {

        Mul_Out_put.write("Task2",new Text(key.toString().split("_")[1] + ":"),
new Text("'" + sum2), OutputPath[1] + "/" + key.toString().split("_")[1]);
    } else {
```



```

        Mul_Out_put.write("Task2", new Text(key.toString().split("_")[2]), new
Text(" " + sum2), OutputPath[1] + "/" + key.toString().split("_")[1]);
    }
}
if (task == 3) {
    int sum = 0;
    for (Text t : values)
        sum += Integer.parseInt(t.toString());
    if (key.toString().split("_").length == 2) {
        Mul_Out_put.write("Task3", new Text(key.toString().split("_")[1] +
":"), new Text(" " + sum), OutputPath[2] + "/" + key.toString().split("_")[1]);
    } else {
        Mul_Out_put.write("Task3", new Text(key.toString().split("_")[2]), new
Text(" " + sum), OutputPath[2] + "/" + key.toString().split("_")[1]);
    }
}
if (task == 4) {
    int n = 0;
    double time = 0;
    for (Text t : values) {
        String[] valueInfo = t.toString().split("_");
        int ni = Integer.parseInt(valueInfo[1]);
        time += (Double.parseDouble(valueInfo[0]) * ni);
        n += ni;
    }
    if (key.toString().split("_").length == 2) {
        Mul_Out_put.write("Task4", new Text(key.toString().split("_")[1] +
":"), new Text(String.format("%.2f", time / n)), OutputPath[3] + "/" +
key.toString().split("_")[1]);
    } else {
        Mul_Out_put.write("Task4", new Text(key.toString().split("_")[2]), new
Text(String.format("%.2f", time / n)), OutputPath[3] + "/" + key.toString().split("_")[1]);
    }
}
}
}

```

#### 4.1.2.6 输出

输出使用MultipleOutputs 类将数据输出到多个文件夹和文件

```

public static class LogTextOutputFormat extends TextOutputFormat<Text, Text> {
    @Override
    public Path getDefaultWorkFile(TaskAttemptContext context, String extension) {
        Path out_path;
        out_path = new Path(getOutputName(context) + ".txt");
        return out_path;
    }
}

```

#### 4.1.2.7 main函数

```

public static void main(String[] args) {
    try {

```

```

Configuration conf = new Configuration();

conf.set("outputPath1", args[1]);
conf.set("outputPath2", args[2]);
conf.set("outputPath3", args[3]);
conf.set("outputPath4", args[4]);
conf.set("mapred.textoutputformat.ignoreseparator", "true");
conf.set("mapred.textoutputformat.separator", " ");

Job job = Job.getInstance(conf, "LogAnalysis");

job.setJarByClass(LogAnalysis.class);
job.setInputFormatClass(TextInputFormat.class);
job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(Text.class);

job.setMapperClass(LogAnalysisMapper.class);
job.setCombinerClass(LogAnalysisCombiner.class);
job.setPartitionerClass(LogAnalysisPartitioner.class);
job.setReducerClass(LogAnalysisReducer.class);

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

MultipleOutputs.addNamedOutput(job, "Task1", LogTextOutputFormat.class,
Text.class, Text.class);
MultipleOutputs.addNamedOutput(job, "Task2", LogTextOutputFormat.class,
Text.class, Text.class);
MultipleOutputs.addNamedOutput(job, "Task3", LogTextOutputFormat.class,
Text.class, Text.class);
MultipleOutputs.addNamedOutput(job, "Task4", LogTextOutputFormat.class,
Text.class, Text.class);
FileOutputFormat.setOutputPath(job, new Path(args[1]));

LazyOutputFormat.setOutputFormatClass(job, LogTextOutputFormat.class);
System.exit(job.waitForCompletion(true) ? 0 : 1);
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

## 4.1.2 运行截图

### 4.1.2.1 连接集群

### 4.1.2.2 mvn package

```

hadoop@ubuntu:~/IdeaProjects/log $ mvn package
[INFO] Scanning for projects...
[INFO]
[INFO] -----< log:log >-----
[INFO] Building log 1.0-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ log ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ log ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] Compiling 1 source file to /home/hadoop/IdeaProjects/log/target/classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ log ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory /home/hadoop/IdeaProjects/log/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ log ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ log ---
[INFO] No tests to run.
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ log ---
[INFO] Building jar: /home/hadoop/IdeaProjects/log/target/log-1.0-SNAPSHOT.jar
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 15.037 s
[INFO] Finished at: 2019-07-08T01:00:35-07:00
[INFO]
hadoop@ubuntu:~/IdeaProjects/log $

```

#### 4.1.2.3 运行

(2015-09-08.log)

```

[2019-07-19 00:38:41 master01 work]$ hadoop log-1.0-SNAPSHOT.jar LogAnalysis /data/task1/JN1_LOG/2015-09-08.log 1 2 3 4
Error: Could not find or load main class log-1.0-SNAPSHOT.jar
[2019-07-19 00:38:44 master01 work]$ hadoop jar log-1.0-SNAPSHOT.jar LogAnalysis /data/task1/JN1_LOG/2015-09-08.log 1 2 3 4
19/07/19 00:38:44 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
19/07/19 00:38:45 INFO client.RMProxy: Connecting to ResourceManager at master01/192.168.1.1:8032
19/07/19 00:38:46 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool Interface and execute your application with ToolRunner to remedy this.
19/07/19 00:38:46 INFO InputFileInputFormat: Total input paths to process : 1
19/07/19 00:38:46 INFO mapreduce.JobSubmitter: number of splits:8
19/07/19 00:38:46 INFO Configuration.deprecation: mapred.textoutputformat.separator is deprecated. Instead, use mapreduce.output.textoutputformat.separator
19/07/19 00:38:46 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1563430216452_0631
19/07/19 00:38:47 INFO impl.VarnClientImpl: Submitted application application_1563430216452_0631
19/07/19 00:38:47 INFO mapreduce.Job: The url to track the job: http://master01:8088/proxy/application_1563430216452_0631/
19/07/19 00:38:47 INFO mapreduce.Job: Running job: job_1563430216452_0631
19/07/19 00:38:58 INFO mapreduce.Job: Job job_1563430216452_0631 running in uber mode : false
19/07/19 00:39:13 INFO mapreduce.Job: map 0% reduce 0%
19/07/19 00:39:15 INFO mapreduce.Job: map 2% reduce 0%
19/07/19 00:39:16 INFO mapreduce.Job: map 3% reduce 0%
19/07/19 00:39:17 INFO mapreduce.Job: map 5% reduce 0%
19/07/19 00:39:18 INFO mapreduce.Job: map 6% reduce 0%
19/07/19 00:39:19 INFO mapreduce.Job: map 8% reduce 0%
19/07/19 00:39:20 INFO mapreduce.Job: map 9% reduce 0%
19/07/19 00:39:21 INFO mapreduce.Job: map 10% reduce 0%
19/07/19 00:39:22 INFO mapreduce.Job: map 12% reduce 0%
19/07/19 00:39:23 INFO mapreduce.Job: map 13% reduce 0%
19/07/19 00:39:24 INFO mapreduce.Job: map 14% reduce 0%
19/07/19 00:39:25 INFO mapreduce.Job: map 16% reduce 0%
19/07/19 00:39:26 INFO mapreduce.Job: map 17% reduce 0%
19/07/19 00:39:28 INFO mapreduce.Job: map 24% reduce 0%
19/07/19 00:39:29 INFO mapreduce.Job: map 25% reduce 0%
19/07/19 00:39:30 INFO mapreduce.Job: map 26% reduce 0%
19/07/19 00:39:31 INFO mapreduce.Job: map 31% reduce 0%
19/07/19 00:39:32 INFO mapreduce.Job: map 32% reduce 0%
19/07/19 00:39:33 INFO mapreduce.Job: map 33% reduce 0%
19/07/19 00:39:34 INFO mapreduce.Job: map 35% reduce 0%
19/07/19 00:39:35 INFO mapreduce.Job: map 40% reduce 0%
19/07/19 00:39:36 INFO mapreduce.Job: map 41% reduce 0%
19/07/19 00:39:37 INFO mapreduce.Job: map 43% reduce 0%
19/07/19 00:39:38 INFO mapreduce.Job: map 45% reduce 0%
19/07/19 00:39:40 INFO mapreduce.Job: map 47% reduce 0%
19/07/19 00:39:41 INFO mapreduce.Job: map 49% reduce 0%

```

```
19/07/19 00:39:37 INFO mapreduce.Job: map 43% reduce 0%
19/07/19 00:39:38 INFO mapreduce.Job: map 45% reduce 0%
19/07/19 00:39:40 INFO mapreduce.Job: map 47% reduce 0%
19/07/19 00:39:41 INFO mapreduce.Job: map 49% reduce 0%
19/07/19 00:39:43 INFO mapreduce.Job: map 50% reduce 0%
19/07/19 00:39:44 INFO mapreduce.Job: map 51% reduce 0%
19/07/19 00:39:46 INFO mapreduce.Job: map 53% reduce 0%
19/07/19 00:39:47 INFO mapreduce.Job: map 55% reduce 4%
19/07/19 00:39:49 INFO mapreduce.Job: map 57% reduce 4%
19/07/19 00:39:51 INFO mapreduce.Job: map 58% reduce 4%
19/07/19 00:39:53 INFO mapreduce.Job: map 60% reduce 4%
19/07/19 00:39:54 INFO mapreduce.Job: map 61% reduce 4%
19/07/19 00:39:56 INFO mapreduce.Job: map 63% reduce 4%
19/07/19 00:39:57 INFO mapreduce.Job: map 65% reduce 4%
19/07/19 00:39:59 INFO mapreduce.Job: map 70% reduce 4%
19/07/19 00:40:00 INFO mapreduce.Job: map 75% reduce 4%
19/07/19 00:40:01 INFO mapreduce.Job: map 75% reduce 8%
19/07/19 00:40:02 INFO mapreduce.Job: map 80% reduce 8%
19/07/19 00:40:03 INFO mapreduce.Job: map 81% reduce 8%
19/07/19 00:40:04 INFO mapreduce.Job: map 81% reduce 17%
19/07/19 00:40:06 INFO mapreduce.Job: map 83% reduce 17%
19/07/19 00:40:09 INFO mapreduce.Job: map 93% reduce 17%
19/07/19 00:40:11 INFO mapreduce.Job: map 96% reduce 17%
19/07/19 00:40:13 INFO mapreduce.Job: map 96% reduce 29%
19/07/19 00:40:15 INFO mapreduce.Job: map 100% reduce 29%
19/07/19 00:40:16 INFO mapreduce.Job: map 100% reduce 36%
19/07/19 00:40:19 INFO mapreduce.Job: map 100% reduce 67%
19/07/19 00:40:22 INFO mapreduce.Job: map 100% reduce 71%
19/07/19 00:40:25 INFO mapreduce.Job: map 100% reduce 78%
19/07/19 00:40:28 INFO mapreduce.Job: map 100% reduce 84%
19/07/19 00:40:31 INFO mapreduce.Job: map 100% reduce 92%
19/07/19 00:40:34 INFO mapreduce.Job: map 100% reduce 99%
19/07/19 00:40:37 INFO mapreduce.Job: map 100% reduce 100%
19/07/19 00:40:40 INFO mapreduce.Job: Job job_1563430216452_0631 completed successfully
19/07/19 00:40:40 INFO mapreduce.Job: Counters: 51
  File System Counters
    FILE: Number of bytes read=5223645
    FILE: Number of bytes written=8864256
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=970826889
    HDFS: Number of bytes written=7793881
    HDFS: Number of read operations=25
```

```
19/07/19 00:40:31 INFO mapreduce.Job: map 100% reduce 92%
19/07/19 00:40:34 INFO mapreduce.Job: map 100% reduce 99%
19/07/19 00:40:37 INFO mapreduce.Job: map 100% reduce 100%
19/07/19 00:40:40 INFO mapreduce.Job: Job job_1563430216452_0631 completed successfully
19/07/19 00:40:40 INFO mapreduce.Job: Counters: 51
  File System Counters
    FILE: Number of bytes read=5223645
    FILE: Number of bytes written=8864256
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=970826889
    HDFS: Number of bytes written=7793881
    HDFS: Number of read operations=25
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=114
  Job Counters
    Killed map tasks=1
    Launched map tasks=9
    Launched reduce tasks=1
    Data-local map tasks=4
    Rack-local map tasks=5
    Total time spent by all maps in occupied slots (ms)=1875052
    Total time spent by all reduces in occupied slots (ms)=246140
    Total time spent by all map tasks (ms)=468763
    Total time spent by all reduce tasks (ms)=61535
    Total vcore-seconds taken by all map tasks=468763
    Total vcore-seconds taken by all reduce tasks=61535
    Total megabyte-seconds taken by all map tasks=3840106496
    Total megabyte-seconds taken by all reduce tasks=504094720
  Map-Reduce Framework
    Map input records=10409526
    Map output records=83230984
    Map output bytes=2182758755
    Map output materialized bytes=2549362
    Input split bytes=952
    Combine input records=83893443
    Combine output records=1350459
    Reduce input groups=686453
    Reduce shuffle bytes=2549362
    Reduce input records=688000
    Reduce output records=0
    Spilled Records=2069834
    Shuffled Maps =8
```

```
    Launched reduce tasks=1
    Data-local map tasks=4
    Rack-local map tasks=5
    Total time spent by all maps in occupied slots (ms)=1875052
    Total time spent by all reduces in occupied slots (ms)=246140
    Total time spent by all map tasks (ms)=468763
    Total time spent by all reduce tasks (ms)=61535
    Total vcore-seconds taken by all map tasks=468763
    Total vcore-seconds taken by all reduce tasks=61535
    Total megabyte-seconds taken by all map tasks=3840106496
    Total megabyte-seconds taken by all reduce tasks=504094720
  Map-Reduce Framework
    Map input records=10409526
    Map output records=83230984
    Map output bytes=2182758755
    Map output materialized bytes=2549362
    Input split bytes=952
    Combine input records=83893443
    Combine output records=1350459
    Reduce input groups=686453
    Reduce shuffle bytes=2549362
    Reduce input records=688000
    Reduce output records=0
    Spilled Records=2069834
    Shuffled Maps =8
    Failed Shuffles=0
    Merged Map outputs=0
    GC time elapsed (ms)=2600
    CPU time spent (ms)=534130
    Physical memory (bytes) snapshot=10573869056
    Virtual memory (bytes) snapshot=46284132352
    Total committed heap usage (bytes)=11048845312
  Shuffle Errors
    BAD_ID=0
    CONNECTION=0
    IO_ERROR=0
    WRONG_LENGTH=0
    WRONG_MAP=0
    WRONG_REDUCE=0
  File Input Format Counters
    Bytes Read=970825937
  File Output Format Counters
    Bytes Written=0
```

```
[2019st35@master01 work]$
```

#### 4.1.2.4 hdfs 下文件get到本地



[illegible][illegible][illegible]

文件1

```
[2019st35@master01 work]$ cd 1
[2019st35@master01 1]$ ls
1.txt  _SUCCESS
[2019st35@master01 1]$
```

```
200: 10403833
404: 4
500: 0
00:00-01:00 200:251501 404:0 500:0
01:00-02:00 200:417175 404:0 500:0
02:00-03:00 200:540917 404:0 500:0
03:00-04:00 200:700389 404:0 500:0
04:00-05:00 200:724747 404:0 500:0
05:00-06:00 200:377207 404:0 500:0
06:00-07:00 200:297346 404:0 500:0
07:00-08:00 200:309841 404:0 500:0
08:00-09:00 200:366154 404:0 500:0
09:00-10:00 200:403606 404:0 500:0
10:00-11:00 200:505789 404:2 500:0
11:00-12:00 200:449247 404:0 500:0
12:00-13:00 200:384767 404:0 500:0
13:00-14:00 200:439435 404:2 500:0
14:00-15:00 200:469977 404:0 500:0
15:00-16:00 200:482252 404:0 500:0
16:00-17:00 200:524866 404:0 500:0
17:00-18:00 200:500323 404:0 500:0
18:00-19:00 200:336887 404:0 500:0
19:00-20:00 200:336751 404:0 500:0
20:00-21:00 200:396558 404:0 500:0
21:00-22:00 200:392855 404:0 500:0
22:00-23:00 200:438798 404:0 500:0
23:00-00:00 200:356445 404:0 500:0
```

## 文件2

```
[2019st35@master01 work]$ cd 2
[2019st35@master01 2]$ ls
10.10.0.101.txt 10.10.0.92.txt 172.22.49.35.txt 172.22.49.51.txt 172.22.49.66.txt 172.22.49.89.txt 172.22.50.27.txt 172.22.50.37.txt 172.30.103.72.txt 172.30.39.93.txt -.txt
10.10.0.102.txt 10.10.0.95.txt 172.22.49.41.txt 172.22.49.53.txt 172.22.49.67.txt 172.22.50.20.txt 172.22.50.28.txt 172.22.50.38.txt 172.30.36.109.txt 172.30.46.23.txt
10.10.0.109.txt 10.10.10.144.txt 172.22.49.43.txt 172.22.49.54.txt 172.22.49.75.txt 172.22.50.21.txt 172.22.50.32.txt 172.22.50.73.txt 172.30.36.150.txt 172.30.47.150.txt
10.10.0.120.txt 10.10.10.188.txt 172.22.49.44.txt 172.22.49.55.txt 172.22.49.83.txt 172.22.50.22.txt 172.22.50.33.txt 172.22.50.74.txt 172.30.36.74.txt 172.30.47.199.txt
10.10.0.162.txt 10.10.30.189.txt 172.22.49.45.txt 172.22.49.56.txt 172.22.49.86.txt 172.22.50.24.txt 172.22.50.34.txt 172.22.50.94.txt 172.30.37.201.txt 172.30.53.140.txt
10.10.0.165.txt 172.22.49.26.txt 172.22.49.46.txt 172.22.49.57.txt 172.22.49.88.txt 172.22.50.25.txt 172.22.50.36.txt 172.22.50.95.txt 172.30.39.191.txt null.txt
[2019st35@master01 2]$
```

```
10.10.0.101: 50
00:00-01:00 4
02:00-03:00 10
03:00-04:00 1
04:00-05:00 3
05:00-06:00 2
13:00-14:00 7
17:00-18:00 7
18:00-19:00 13
22:00-23:00 3
```

## 文件3

```
[2019st35@master01 work]$ cd 3
[2019st35@master01 3]$ ls
tour-category-ids-query.txt      tour-category-weekendproduct-query.txt  tour-hotel-search-nearby-scenic-query.txt  tour-pol-query-queryProvinceList.txt      tour-product-query.txt
tour-category-query.txt          tour-faq-query.txt                     tour-hotel-search-query.txt                tour-pol-query-queryScenicNumPerCity.txt  tour-suggestion-query.txt
tour-category-scenic-query.txt   tour-flight-ticket-query.txt            tour-hotelsuggestion-query.txt              tour-pol-query-queryScenicSpotCount.txt
tour-category-stats-query.txt    tour-guide-delete.txt                  tour-phoenix-product-query.txt             tour-pol-query-queryScenicTypeList.txt
tour-category-tuntu-hot-query.txt tour-guide-query.txt                   tour-pol-query-queryCategory.txt            tour-pol-query.txt
tour-category-vendor-stats-query.txt tour-hotel-query.txt                  tour-pol-query-queryNumberFound.txt        tour-pol-scenictype-provincelist-query.txt
[2019st35@master01 3]$
```

```
tour-suggestion-query: 192211
00:00:00 2
00:00:01 5
00:00:02 1
00:00:03 1
00:00:05 2
00:00:06 1
00:00:07 1
00:00:08 1
00:00:09 1
00:00:10 1
00:00:11 2
00:00:12 1
00:00:13 1
00:00:14 2
00:00:15 3
00:00:16 3
00:00:17 1
00:00:18 1
00:00:19 1
00:00:20 2
00:00:21 2
00:00:26 1
00:00:27 2
00:00:29 1
00:00:31 1
00:00:32 2
00:00:33 1
00:00:34 1
00:00:35 1
00:00:36 2
00:00:37 2
00:00:38 1
00:00:39 1
00:00:40 1
00:00:41 1
00:00:42 1
00:00:43 4
00:00:44 3
00:00:45 2
00:00:47 1
00:00:48 2
00:00:49 2
"tour-suggestion-query.txt" 608351 660868C
```

## 文件4

```
[2019st35@master01 work]$ cd 4/
[2019st35@master01 4]$ ls
tour-category-ids-query.txt      tour-category-weekendproduct-query.txt  tour-hotel-search-nearby-scenic-query.txt  tour-pol-query-queryProvincelst.txt  tour-product-query.txt
tour-category-query.txt         tour-faq-query.txt                     tour-hotel-search-query.txt               tour-pol-query-queryScenicNumPerCity.txt  tour-suggestion-query.txt
tour-category-scenic-query.txt  tour-flight-ticket-query.txt           tour-hotelsuggestion-query.txt             tour-pol-query-queryScenicSpotCount.txt
tour-category-statis-query.txt  tour-guide-delete.txt                 tour-phoenix-product-query.txt            tour-pol-query-queryScenicTypeList.txt
tour-category-tuntu-hot-query.txt  tour-guide-query.txt                  tour-pol-query-queryCategory.txt           tour-pol-query.txt
tour-category-vendor-statis-query.txt  tour-hotel-query.txt                 tour-pol-query-queryNumberFound.txt        tour-pol-scenicIcType-provincelist-query.txt
[2019st35@master01 4]$
```



```
tour-suggestion-query: 6.84
00:00-01:00 2.71
01:00-02:00 2.61
02:00-03:00 2.71
03:00-04:00 2.78
04:00-05:00 2.97
05:00-06:00 10.65
06:00-07:00 2.84
07:00-08:00 2.77
08:00-09:00 2.96
09:00-10:00 6.99
10:00-11:00 15.05
11:00-12:00 11.77
12:00-13:00 3.92
13:00-14:00 7.78
14:00-15:00 9.94
15:00-16:00 8.85
16:00-17:00 11.89
17:00-18:00 6.83
18:00-19:00 4.13
19:00-20:00 3.61
20:00-21:00 3.08
21:00-22:00 3.08
22:00-23:00 3.10
23:00-00:00 3.42
```

#### 4.1.3代码:

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.TaskAttemptContext;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.*;
import org.apache.hadoop.mapreduce.lib.partition.HashPartitioner;

import java.io.IOException;

public class LogAnalysis {

    public static class LogAnalysisMapper extends Mapper<LongWritable, Text, Text, Text> {

        @Override
        public void map(LongWritable key, Text value, Context context) throws IOException,
        InterruptedException {

            System.out.println("start map");
```



```

String[] log = value.toString().split(" ");
if (log.length == 10) {

    /*log_ip*/
    String log_ip = log[0];
    /*log_time (delete day/month.year)*/
    String log_time = log[1];
    log_time = log_time.substring(13);
    /*log_time_zone delete ] all time zone are +8000 */
    String log_time_zone = log[2];
    log_time_zone = log_time_zone.substring(0, 4);
    /*log_http_method_1 delete " */
    String log_http_method_1 = log[3];
    log_http_method_1 = log_http_method_1.substring(1);
    /*log_url delete the first "/" and we need replace all / to - */
    String log_url = log[4];
    log_url = log_url.substring(1);
    log_url = log_url.replaceAll("/", "-");
    /*log_http */
    String log_http = log[5];
    log_http = log_http.substring(0, 7);
    /*log_http_method_2 get */
    String log_http_method_2 = log[6];
    /*log_status_code 200 404 500 */
    String log_status_code = log[7];
    /*log_response_length byte length*/
    String log_response_length = log[8];
    /*log_response_time */
    String log_response_time = log[9];
    /*example 00:19:16 to 00:00-01:00*/
    String format_time = Get_Format_Time(log_time);
    Text value_1 = new Text("1");
    Text key_all_log_status_code = new Text("1" + "_" + "00" + ":" + "00" + "-"
+ "00" + ":" + "00" + "_" + log_status_code);
    Text key_log_status_code = new Text("1" + "_" + format_time + "_" +
log_status_code);
    Text key_ip = new Text("2" + "_" + log_ip);
    Text key_ip_time = new Text("2_" + log_ip + "_" + format_time);
    Text key_url = new Text("3_" + log_url);
    Text key_url_time = new Text("3_" + log_url + "_" + log_time);
    Text key_4_url = new Text("4_" + log_url);
    Text key_4_url_time = new Text("4_" + log_url + "_" + format_time);
    Text value_response_time = new Text(log_response_time + "_" + "1");

//      task1
    context.write(key_all_log_status_code, value_1);
    context.write(key_log_status_code, value_1);

//      task2
    context.write(key_ip, value_1);
    context.write(key_ip_time, value_1);

//      task3
    context.write(key_url, value_1);
    context.write(key_url_time, value_1);

//      task4

```

```

        context.write(key_4_url, value_response_time);
        context.write(key_4_url_time, value_response_time);
    }
}

private String Get_Format_Time(String time) {
    time = time.split(":")[0];
    int first_time = Integer.parseInt(time);
    int second_time = first_time + 1;
    /*if secod_time is 24:** we need change it to 00:** */
    if (second_time == 24) {
        second_time = 0;
    }
    /*return formatting time */
    return (String.format("%02d", first_time) + ":" + "00" + "-" +
String.format("%02d", second_time) + ":" + "00");
}

}

public static class LogAnalysisCombiner extends Reducer<Text, Text, Text, Text> {
    @Override
    public void reduce(Text key, Iterable<Text> values, Context context) throws
IOException, InterruptedException {

        System.out.println("start combiner");
        String[] key1 = key.toString().split("_");

        int task = Integer.parseInt(key1[0]);
        if (task == 1 || task == 2 || task == 3) {
            int sum = 0;
            for (Text t : values) {
                int index;
                index = Integer.parseInt(t.toString());
                sum = sum + index;
            }
            Text c_sum;
            c_sum = new Text("" + sum);
            context.write(key, c_sum);
        } else {
            int index = 0;
            double time = 0;
            for (Text t : values) {
                time += (Double.parseDouble(t.toString().split("_")[0]) *
(Integer.parseInt(t.toString().split("_")[1])));
                index = index + Integer.parseInt(t.toString().split("_")[1]);
            }
            Text c_sum;
            c_sum = new Text("" + time / index + "_" + index);
            context.write(key, c_sum);
        }
    }
}
}

```

```

public static class LogAnalysisPartitioner extends HashPartitioner<Text, Text> {
    @Override
    public int getPartition(Text key, Text value, int sum) {
        System.out.println("Start partitioner");

        if (key.toString().split("_").length == 2)
            return super.getPartition(key, value, sum);
        else
            return super.getPartition(new Text(key.toString().split("_")[0] + "_" +
key.toString().split("_")[1]), value, sum);
    }
}

public static class LogAnalysisReducer extends Reducer<Text, Text, Text, Text> {

    private MultipleOutputs<Text, Text> Mul_Out_put;
    private String[] OutputPath;
    private Integer flag1 = 0;
    private Integer flag2 = 0;
    private Integer flag3 = 0;

    @Override
    public void setup(Context context) {
        Configuration conf = context.getConfiguration();
        OutputPath = new String[4];
        OutputPath[0] = conf.get("outputPath1");
        OutputPath[1] = conf.get("outputPath2");
        OutputPath[2] = conf.get("outputPath3");
        OutputPath[3] = conf.get("outputPath4");
        Mul_Out_put = new MultipleOutputs<Text, Text>(context);
    }
    private static String flag = new String();
    @Override
    public void reduce(Text key, Iterable<Text> values, Context context) throws
IOException, InterruptedException {
        System.out.println("start reduce");
        int task = Integer.parseInt(key.toString().split("_")[0]);
        Text Status_Code_200_flag=new Text("200" + ":");
        Text Status_Code_404_flag=new Text("404" + ":");
        Text Status_Code_500_flag=new Text("500" + ":");
        Text Status_Code_200_Count=new Text(flag1.toString());
        Text Status_Code_404_Count=new Text(flag2.toString());
        Text Status_Code_500_Count=new Text(flag3.toString());
        Text Status_Code =new Text(" 200:" + flag1.toString() + " 404:" +
flag2.toString() + " 500:" + flag3.toString());

        if (task == 1) {
            int sum_1 = 0;
            for (Text v : values)
                sum_1 =sum_1+ Integer.parseInt(v.toString());

```

```

        if (flag.compareTo(key.toString().split("_")[1]) != 0 && flag.length() !=
0)
        {
            if (flag.compareTo("00:00-00:00") == 0) {

                Mul_Out_put.write("Task1", Status_Code_200_flag,
Status_Code_200_Count, OutputPath[0] + "/" + "1");
                Mul_Out_put.write("Task1", Status_Code_404_flag,
Status_Code_404_Count, OutputPath[0] + "/" + "1");
                Mul_Out_put.write("Task1", Status_Code_500_flag,
Status_Code_500_Count, OutputPath[0] + "/" + "1");

            } else {
                Mul_Out_put.write("Task1", new Text(flag), Status_Code,
OutputPath[0] + "/" + "1");
            }
            flag1 = 0;
            flag2 = 0;
            flag3 = 0;
        }
        if (key.toString().split("_")[2].compareTo("200") == 0)
            flag1 = sum_1;
        else if (key.toString().split("_")[2].compareTo("404") == 0)
            flag2 = sum_1;
        else if (key.toString().split("_")[2].compareTo("500") == 0)
            flag3 = sum_1;
        flag = key.toString().split("_")[1];
    }
    if (task == 2) {
        int sum2 = 0;
        for (Text t : values)
            sum2 += Integer.parseInt(t.toString());
        if (key.toString().split("_").length == 2) {

            Mul_Out_put.write("Task2", new Text(key.toString().split("_")[1] + ":"),
new Text("'" + sum2), OutputPath[1] + "/" + key.toString().split("_")[1]);
        } else {

            Mul_Out_put.write("Task2", new Text(key.toString().split("_")[2]), new
Text("'" + sum2), OutputPath[1] + "/" + key.toString().split("_")[1]);
        }
    }
    if (task == 3) {
        int sum = 0;
        for (Text t : values)
            sum += Integer.parseInt(t.toString());
        if (key.toString().split("_").length == 2) {
            Mul_Out_put.write("Task3", new Text(key.toString().split("_")[1] +
":"), new Text("'" + sum), OutputPath[2] + "/" + key.toString().split("_")[1]);
        } else {
            Mul_Out_put.write("Task3", new Text(key.toString().split("_")[2]), new
Text("'" + sum), OutputPath[2] + "/" + key.toString().split("_")[1]);
        }
    }

```

```

    }
    if (task == 4) {
        int n = 0;
        double time = 0;
        for (Text t : values) {
            String[] valueInfo = t.toString().split("_");
            int ni = Integer.parseInt(valueInfo[1]);
            time += (Double.parseDouble(valueInfo[0]) * ni);
            n += ni;
        }
        if (key.toString().split("_").length == 2) {
            Mul_Out_put.write("Task4", new Text(key.toString().split("_")[1] +
":"), new Text(String.format("%.2f", time / n)), OutputPath[3] + "/" +
key.toString().split("_")[1]);
        } else {
            Mul_Out_put.write("Task4", new Text(key.toString().split("_")[2]), new
Text(String.format("%.2f", time / n)), OutputPath[3] + "/" + key.toString().split("_")[1]);
        }
    }
}

@Override
public void cleanup(Context context) throws IOException, InterruptedException {
    Text Status_Code_200_flag=new Text("200" + ":");
    Text Status_Code_404_flag=new Text("404" + ":");
    Text Status_Code_500_flag=new Text("500" + ":");
    Text Status_Code_200_Count=new Text(flag1.toString());
    Text Status_Code_404_Count=new Text(flag2.toString());
    Text Status_Code_500_Count=new Text(flag3.toString());
    Text Status_Code =new Text(" 200:" + flag1.toString() + " 404:" +
flag2.toString() + " 500:" + flag3.toString());
    if (flag.compareTo("00:00-00:00") == 0) {
        Mul_Out_put.write("Task1", Status_Code_200_flag, Status_Code_200_Count,
OutputPath[0] + "/1");
        Mul_Out_put.write("Task1", Status_Code_404_flag, Status_Code_404_Count,
OutputPath[0] + "/1");
        Mul_Out_put.write("Task1", Status_Code_500_flag, Status_Code_500_Count,
OutputPath[0] + "/1");
    } else {
        Mul_Out_put.write("Task1", new Text(flag), Status_Code, OutputPath[0] +
"/1");
    }
    Mul_Out_put.close();
}

}

public static class LogTextOutputFormat extends TextOutputFormat<Text, Text> {
    @Override
    public Path getDefaultworkFile(TaskAttemptContext context, String extension) {
        Path out_path;
        out_path = new Path(getOutputName(context) + ".txt");
    }
}

```

```

        return out_path;
    }

}

public static void main(String[] args) {
    try {

        Configuration conf = new Configuration();

        conf.set("outputPath1", args[1]);
        conf.set("outputPath2", args[2]);
        conf.set("outputPath3", args[3]);
        conf.set("outputPath4", args[4]);
        conf.set("mapred.textoutputformat.ignoreseparator", "true");
        conf.set("mapred.textoutputformat.separator", " ");

        Job job = Job.getInstance(conf, "LogAnalysis");

        job.setJarByClass(LogAnalysis.class);
        job.setInputFormatClass(TextInputFormat.class);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(Text.class);

        job.setMapperClass(LogAnalysisMapper.class);
        job.setCombinerClass(LogAnalysisCombiner.class);
        job.setPartitionerClass(LogAnalysisPartitioner.class);
        job.setReducerClass(LogAnalysisReducer.class);

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

        MultipleOutputs.addNamedOutput(job, "Task1", LogTextOutputFormat.class,
Text.class, Text.class);
        MultipleOutputs.addNamedOutput(job, "Task2", LogTextOutputFormat.class,
Text.class, Text.class);
        MultipleOutputs.addNamedOutput(job, "Task3", LogTextOutputFormat.class,
Text.class, Text.class);
        MultipleOutputs.addNamedOutput(job, "Task4", LogTextOutputFormat.class,
Text.class, Text.class);
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        LazyOutputFormat.setOutputFormatClass(job, LogTextOutputFormat.class);
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

## 4.2任务5

### 4.2.1任务简述

任务 5: 接口访问频次预测, 给 2015-09-08.log 到 2015-09-21.log 共 14 天的日志文件, 作为训练数据, 设计预测算法来预测下一天 (2015-09-22) 每个小时窗内每个接口 (请求的 URL) 的访问总频次。将该结果与当天实际的统计值(2015-09-22.log)做 RMSE 验证。

任务5实际上是一个无监督性学习的过程

## 4.2.2主要流程

### 生成训练集和测试集

两者的代码格式几乎一样, 在这里我们主要讨论生成训练集。

根据任务信息, 我们不难得出训练集的信息需要包括9月8日到9月21日共14天每个小时窗内各个接口的访问总频次。我们通过修改任务3的代码不难得出生成训练集的代码。

### 数据结构

自定义TimeTable类以记录某接口在每天的每个小时段中的访问频次信息(使用Hour\_Data记录)

```
public class GenTest {
    public static int Start_Date = 8;
    public static int End_Date = 21;
    public static class TimeTable implements Writable {

        private int Day_Length = End_Date - Start_Date + 1; //TODO: ..
        private int[][] Hour_Data = new int[24][Day_Length];
        public TimeTable() {
            for (int i = 0; i < 24; i++)
                for (int j = 0; j < Day_Length; j++)
                    Hour_Data[i][j] = 0;
        }
        public void Add_Info(int hour, int day, int fre){
            if((hour < 24 && hour > 0))
                Hour_Data[hour][day - 8] += fre;
        }
        public int Get_Info(int hour, int day){
            return Hour_Data[hour][day];
        }

        public void write(DataOutput out) throws IOException

        public void readFields(DataInput in) throws IOException
    }
}
```

### Map阶段

读取一行形如

```
172.22.49.44 [08/Sep/2015:00:19:16 +0800] "GET /tour/category/query HTTP/1.1" GET 200 124 8
```

的数据, 获取其中的URL, Hour, Day信息

```
String Day_Key = log[1].substring(1, 3);
String Month_Key = log[1].substring(4, 7);
String Year_Key = log[1].substring(8, 12);
String Second_Key = log[1].substring(13, 15);
String Min_Key = log[1].substring(16, 18);
String Hour_Key = log[1].substring(19, 21);

String Url_Key = log[4].substring(0, log[4].length() - 1);
Url_Key = Url_Key.replace("/", "-");
if(Url_Key.substring(0,1).equals("-"))
    Url_Key = Url_Key.substring(1,Url_Key.length());
```

新建一个TimeTable数据类t，将Hour\_Data中[Hour][Day]值设为1,其余根据默认初始化为0

最后以URL为key值，以t为Value

```
EMIT (URL,t);
```

### Combiner阶段

将同一个key即同一个接口对应的TimeTable类合并

```
TimeTable result = new TimeTable();
for(TimeTable t : values)
{
    int temp = 0;
    for (int i = 0; i < 24 ;i++)
        for (int j = 0; j < End_Date - Start_Date + 1;j++) {
            temp = t.Get_Info(i,j);
            result.Add_Info(i,j + 8,temp);
        }
}
context.write(key,result);
```

### Reducer阶段

本部分和任务3比较类似，使用MutipleOutput进行多文件输出，以URL.txt作为文件名，输出每个小时窗内的信息

首先合并同一个接口对应的小时窗信息。



```

TimeTable result = new TimeTable();
//String line = new String()
for (TimeTable t : values)
{
    for (int i = 0; i < 24; i++)
    for (int j = 0; j < End_Date - Start_Date + 1; j++)
    {
        int temp = t.Get_Info(i,j);
        result.Add_Info(i,j + 8,temp);
    }
}

```

将合并的信息存入一个新的TimeTable数据类，并命名为result。

然后将result中的信息输入到对应的文件中。

```

for (int i = 0; i < 24; i++)
{
    StringBuffer line = new StringBuffer();
    for (int j = 0 ; j < End_Date - Start_Date + 1; j++)
    {
        line.append(result.Get_Info(i,j));
        line.append('\t');
    }
    int next_hour = (i + 1) % 24;
    String hour_window = new String(String.format("%02d",i) + ":00 - " +
String.format("%02d",next_hour) + ":00");

    mos.write("GenTest",new Text(hour_window),line,output_path + "/" +key);
}

```

reduce阶段遇到的几个问题

- 1.log文件中URL中以/作为分隔，而当以URL为文件名的时候，会自动以/生成多层目录，所以我们需要用-替代/
- 2.vim无法打开以-开头的文件，我们还需将以-开头的URL中的第一个-去掉。

## main函数

利用正则表达式过滤掉文件夹中的9-22.log

```

filesystem = FileSystem.get(conf);

FileStatus [] fileStatusArr = filesystem.globStatus(new Path(args[0] +
"/*.log"), new PathFilter() {
    public boolean accept(Path path) {
        String regex = "(?!.*2015-09-22).*$";
        return path.toString().matches(regex);
        // return false;
    }
});

```

其余部分如任务1-4，不多赘述。

最后输出的文件格式形如

```
00:00 - 01:00 8212 ... 01:00 - 02:00 11868 ... 02:00 - 03:00 7274 ... 03:00 - 04:00 6546 ... 04:00 - 05:00 6167 ...
05:00 - 06:00 6127 ... 06:00 - 07:00 6491 ... 07:00 - 08:00 9245 ... 08:00 - 09:00 21826 ... 09:00 - 10:00 49399 ...
10:00 - 11:00 62193 ... 11:00 - 12:00 59206 ... 12:00 - 13:00 40436 ... 13:00 - 14:00 53083 ... 14:00 - 15:00 57625
... 15:00 - 16:00 64044 ... 16:00 - 17:00 61983 ... 17:00 - 18:00 45753 ... 18:00 - 19:00 27991 ... 19:00 - 20:00
32873 ... 20:00 - 21:00 36282 ... 21:00 - 22:00 38716 ... 22:00 - 23:00 35848 ... 23:00 - 00:00 22337 ...
```

### 4.2.3 设计预测算法

在生成训练集后，我们需要根据训练集中14天的数据来预测9月22日的访问数据，并与真实值比较。故我们选择了不同的算法进行尝试，分别有：

简单平均：实现在SimpleAverage函数中。即对前14天各接口，各时间段的数据直接取平均值作为9月22日的预测值。

```
FileSplit filesplit = (FileSplit)context.getInputSplit();
String file_name = filesplit.getPath().getName();//得到文件名
String interface_name = file_name.replace(".txt", ""); //去掉后缀得到接口名
String[] day = value.toString().split("\t");
if(day.length == days+1)//1列时间+数据
{
    double sum = Integer.parseInt(day[1]);
    for(int i = 2; i <= days; i++)
    {
        double nextday = Integer.parseInt(day[i]);
        sum += nextday;
    }
    double res = sum/14;
    prediction.write("SimpleAverage",new Text(day[0]), new Text(
String.format("%.2f", res)), out_path+"/"+interface_name );
}
```

指数平滑：实现在ExponentSmooth函数中。此方法对简单平均法进行了改进，通过设置一个参数  $a$  ( $<1$ )，使每一天对预测产生影响的权重不同，使越接近9月22日的那天占据越大的影响权重，最接近的一天影响力为 $a$ ，往前的天影响依次为 $a(1-a)$ ， $a(1-a)(1-a)$ ...。通过尝试我们发现将  $a$  设为0.85取得了较好效果。

```
FileSplit filesplit = (FileSplit)context.getInputSplit();
String file_name = filesplit.getPath().getName();//得到文件名
String interface_name = file_name.replace(".txt", ""); //去掉后缀得到接口名
String[] day = value.toString().split("\t");
if(day.length == days+1)//1列时间+数据
{
    double res = Integer.parseInt(day[1]);
    for(int i = 2; i <= days; i++)
    {
        double nextday = Integer.parseInt(day[i]);
        res = res*(1-a) + nextday*a;
    }
}
```

```

        prediction.write("ExponentSmooth", new Text(day[0]), new Text(
String.format("%.2f", res)), out_path+"/"+interface_name );
    }

```

线性回归：实现在LinearRegression函数中。使用多元线性回归，对各天、各时间段的数据都设置一个参数，得到一个24 \* 15 的矩阵，该参数矩阵对所有接口适用。

```

public double[][] matrix={
    {0.10 ,0.04 , -0.00 ,0.03 , -0.03 ,0.05 ,0.03 , -0.03 ,0.09 ,0.10 ,0.10 ,0.10 ,0.10 ,0.10 ,0.10},
    {0.10 , -0.07 ,0.01 ,0.05 ,0.04 ,0.05 ,0.07 ,0.00 ,0.11 , -0.03 ,0.11 ,0.09 ,0.11 ,0.16 ,0.26},
    {0.10 , -0.07 ,0.04 , -0.01 ,0.03 ,0.01 ,0.05 ,0.05 ,0.12 ,0.12 ,0.12 ,0.12 ,0.12 ,0.12 ,0.12},
    {0.10 , -0.00 ,0.04 , -0.06 ,0.04 , -0.00 ,0.04 ,0.03 ,0.13 ,0.13 ,0.13 ,0.13 ,0.13 ,0.13 ,0.13},
    {0.10 , -0.04 ,0.02 ,0.02 ,0.02 , -0.03 ,0.01 ,0.02 ,0.13 ,0.14 ,0.13 ,0.14 ,0.13 ,0.14 ,0.14},
    {0.10 ,0.09 , -0.01 ,0.01 , -0.06 , -0.03 , -0.03 , -0.02 ,0.14 ,0.09 ,0.09 ,0.10 ,0.10 ,0.20 ,0.21},
    {0.10 , -0.05 ,0.04 ,0.03 ,0.01 , -0.02 ,0.03 ,0.05 ,0.11 ,0.11 ,0.11 ,0.11 ,0.11 ,0.11 ,0.11},
    {0.10 , -0.07 ,0.03 ,0.04 ,0.03 ,0.00 ,0.04 ,0.03 ,0.11 ,0.11 ,0.11 ,0.11 ,0.11 ,0.11 ,0.11},
    {0.10 , -0.05 ,0.01 ,0.04 ,0.03 , -0.03 ,0.04 ,0.04 ,0.12 ,0.12 ,0.12 ,0.12 ,0.12 ,0.12 ,0.12},
    {0.10 , -0.01 , -0.03 ,0.07 ,0.00 ,0.00 , -0.02 ,0.01 ,0.14 ,0.14 ,0.13 ,0.13 ,0.06 ,0.11 ,0.19},
    {0.10 , -0.02 ,0.03 ,0.10 , -0.01 , -0.05 , -0.05 ,0.04 ,0.16 ,0.14 ,0.13 ,0.14 ,0.02 ,0.05 ,0.22},
    {0.10 ,0.02 , -0.02 ,0.09 , -0.02 , -0.06 , -0.01 ,0.04 ,0.15 ,0.14 ,0.14 ,0.17 ,0.03 ,0.07 ,0.21},
    {0.10 ,0.01 , -0.09 ,0.08 ,0.05 ,0.02 ,0.06 ,0.05 ,0.12 ,0.12 ,0.12 ,0.12 ,0.11 ,0.12 ,0.13},
    {0.10 ,0.02 , -0.08 ,0.06 ,0.05 ,0.02 ,0.03 ,0.02 ,0.15 ,0.14 ,0.14 ,0.17 ,0.04 ,0.09 ,0.21},
    {0.10 ,0.05 , -0.05 ,0.15 ,0.07 , -0.07 , -0.13 ,0.01 ,0.09 ,0.13 ,0.20 ,0.17 ,0.01 ,0.04 ,0.35},
    {0.10 ,0.02 , -0.07 ,0.12 ,0.09 , -0.03 , -0.07 ,0.01 ,0.18 ,0.15 ,0.13 ,0.14 ,0.05 ,0.03 ,0.23},
    {0.10 ,0.01 , -0.06 ,0.03 ,0.08 , -0.04 , -0.03 ,0.09 ,0.15 ,0.14 ,0.16 ,0.13 , -0.02 ,0.06 ,0.29},
    {0.10 , -0.07 , -0.04 ,0.09 ,0.09 , -0.01 , -0.02 ,0.12 ,0.13 ,0.12 ,0.14 ,0.12 ,0.00 ,0.08 ,0.25},
    {0.10 , -0.01 , -0.07 ,0.03 ,0.04 , -0.05 ,0.05 ,0.15 ,0.04 ,0.12 ,0.03 ,0.12 ,0.12 ,0.12 ,0.30},
    {0.10 , -0.01 , -0.05 ,0.02 ,0.02 , -0.03 ,0.02 ,0.10 , -0.04 ,0.13 ,0.13 ,0.12 ,0.13 ,0.13 ,0.33},
    {0.10 , -0.02 ,0.07 ,0.10 , -0.01 , -0.13 , -0.02 ,0.08 ,0.09 ,0.07 ,0.13 , -0.07 ,0.13 ,0.19 ,0.39},
    {0.10 ,0.05 ,0.02 ,0.02 ,0.03 , -0.07 , -0.00 ,0.22 ,0.12 , -0.08 ,0.12 ,0.03 ,0.11 ,0.12 ,0.34},
    {0.10 ,0.02 , -0.06 ,0.00 ,0.03 , -0.01 ,0.02 ,0.11 ,0.11 , -0.11 ,0.11 ,0.11 ,0.11 ,0.21 ,0.36},
    {0.10 ,0.01 , -0.04 ,0.01 ,0.02 ,0.00 ,0.03 ,0.12 ,0.12 ,0.12 ,0.12 ,0.12 ,0.12 ,0.12 ,0.13},
};

```

```

Filesplit filesplit = (Filesplit)context.getInputSplit();
String file_name = filesplit.getPath().getName();//得到文件名
String interface_name = file_name.replace(".txt", ""); //去掉后缀得到接口名
String[] day = value.toString().split("\t");
if(day.length == days+1)//1列时间+数据
{
    String hour = day[0].split(":")[0]; //day[0]为时间段
    int row = Integer.parseInt(hour); //决定使用第几行的权重,即第几个时间段
    double res =matrix[row][0] * 1; //在计算系数矩阵时开头加了恒为1的一列, 对应第一列系数

    for(int i = 1; i <=days; i++)
        res+=matrix[row][i]*Integer.parseInt(day[i]);
    prediction.write("LinearRegression", new Text(day[0]), new Text(
String.format("%.2f", res)), out_path+"/"+interface_name );
}

```

如何求出这个矩阵，通过另两个函数GetMatrix和GradientDescent来实现：GradientDescent为梯度下降算法

\* 训练数据示例:

* x0	x1	x2	y
1.0	1.0	2.0	7.2
1.0	2.0	1.0	4.9
1.0	3.0	0.0	2.6
1.0	4.0	1.0	6.3
1.0	5.0	-1.0	1.0
1.0	6.0	0.0	4.7
1.0	7.0	-2.0	-0.6

注意!!! x1, x2, y三列是用户实际输入的数据, x0是为了推导出来的公式统一, 特地补上的一列。  
x0, x1, x2是“特征”, y是结果

$$h(x) = \theta_0 * x_0 + \theta_1 * x_1 + \theta_2 * x_2$$

对输入的数据集进行求导等处理, 得到参数矩阵。由于第一列额外为添加的恒为1的数据, 这也是为什么最后得到24 \* 15的矩阵 (14天+1)。而Y这一列理应为9月22日的真实数据, 但以此求参数有违实验要求预测的用意, 故我们使用指数平滑预测的数据来作为Y, 也得到较为不错的效果。如果使用9月22日的数据作为Y, 则能得到极好的效果。GetMatrix先对训练集处理, 对24个中的每个时间段, 得到一个接口数 \* 14的矩阵 (因为所求参数需要对所有接口适用), 将此作为数据集传入GradientDescent中计算参数。

```
for (int hour=0;hour<24;hour++)
{
    //对每个时间段,创建接口数* (天数+1) 的矩阵, 用于1*(days+1)系数矩阵
    //24行综合在一起可得到24* (days+1) 的系数矩阵
    double[][] testdata=new double[filenum][days+1];
    //依次读取每个文件 (接口) 的数据
    for (int i=0;i<filenum;i++)
    {
        String line=filereader[i].readLine();//下次循环就读到下一行 (下一时间
段)

        //下面模拟指数平滑操作得到最后一列数据添加到数据集中用于生成系数矩阵
        String[] day=line.split("\t");
        double res = Integer.parseInt(day[1]);
        for(int j = 2; j <= days; j++)
        {
            double nextday = Integer.parseInt(day[j]);
            res= res*(1-a) + nextday*a;
        }
        for (int j=0;j<days;j++)
            testdata[i][j]=Integer.parseInt(day[j+1]);
        testdata[i][days]=res;
    }
    //对testdays数据集调用梯度下降算法得到系数矩阵
    GradientDescent m = new
GradientDescent(testdata, filenum, days+1, 0.001, 50000);
    // m.printTrainData();
    m.trainTheta();
    m.printTheta();
    System.out.println();
}
```

#### 4.2.4 计算RMSE

根据所给公式，根据9月22的预测值和真实值计算RMSE。实现在函数ComputeRMSE中。在Mapper阶段，提取出文件名中的接口信息（去掉.txt），文件中时间段信息，生成（时间\_接口，访问数）的键值对。

```
public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
    FileSplit fileSplit = (FileSplit)context.getInputSplit();
    String filename = fileSplit.getPath().getName();    //得到文件名
    if(filename.contains(".txt")&&(!filename.contains(".crc")))
    {
        String[] log = value.toString().split("\t");
        String time = log[0];
        String file_name = ((FileSplit) context.getInputSplit()).getPath()
.getName();
        String interface_name = file_name.replace(".txt", "");

        String num = log[1];
        context.write(new Text(time+"_"+interface_name ),new Text(num));
    }
}
```

在Reducer阶段，先setup时设置好输出路径等信息；在reduce时提取出同一个键值对中的两个数值（分别属于预测和真实结果），并依据所给公式计算RMSE，最后在cleanup中将RMSE的值输出到文件中

```
public void setup(Context context)
{
    Configuration conf = context.getConfiguration();
    out_path = conf.get("out_path");
    prediction = new MultipleOutputs<Text, Text>(context);
}
@Override
public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
InterruptedException
{
    String[] keyInfo=key.toString().split("_");
    String new_hour=keyInfo[0];
    int flag=0;
    double c1 = 0,c2 = 0,diff = 0;
    for(Text val: values)
    {
        if (flag==0)
            c1 = Double.parseDouble(val.toString());
            //diff=Double.parseDouble(val.toString());
        else if (flag==1) {
            c2 = Double.parseDouble(val.toString());
            diff = c2 - c1;
        }
        else
            break;
        flag++;
    }
}
```

```

    }
    if (hour.length() != 0 && hour.compareTo(new_hour) != 0) // 不是一个时间窗
    {
        sum += Math.sqrt(diff_square / interface_num);
        interface_num = 1;
        diff_square = diff * diff;
    }
    else // 同一个时间窗
    {
        interface_num++;
        diff_square += diff * diff;
    }
    hour = new_hour; // 更新时间段
}
@Override
public void cleanup(Context context) throws IOException, InterruptedException
{
    sum += Math.sqrt(diff_square / interface_num);
    RMSE = sum / 24;
    prediction.write(new Text("RMSE = "), new
    Text(Double.toString(RMSE)), out_path + "/RMSE");
    prediction.close();
}

```

#### 4.2.5 各预测算法比较


简单平均: RMSE = 10861.83 指数平滑: RMSE = 3448.56 线性回归: RMSE = 3604.34

可看出简单平均作为最简单的一种方法, 得到RMSE较大, 效果显然不好。而指数平滑和线性回归通过不同的优化, 使RMSE值明显减小, 预测模型正确性显著提升。

## 5 运行状态截图

### Web UI

Show 20 entries								Search: 2019st35		
ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
<a href="#">application_1563430216452_0022</a>	2019st35	ComputeRMSE	MAPREDUCE	root.2019st35	Wed Jul 17 23:57:38 -0700 2019	Wed Jul 17 23:57:57 -0700 2019	FINISHED	SUCCEEDED	<div></div>	<a href="#">History</a>
<a href="#">application_1563430216452_0021</a>	2019st35	ComputeRMSE	MAPREDUCE	root.2019st35	Wed Jul 17 23:56:08 -0700 2019	Wed Jul 17 23:56:26 -0700 2019	FINISHED	SUCCEEDED	<div></div>	<a href="#">History</a>
<a href="#">application_1563430216452_0018</a>	2019st35	ComputeRMSE	MAPREDUCE	root.2019st35	Wed Jul 17 23:53:11 -0700 2019	Wed Jul 17 23:53:28 -0700 2019	FINISHED	SUCCEEDED	<div></div>	<a href="#">History</a>
<a href="#">application_1563430216452_0014</a>	2019st35	ComputeRMSE	MAPREDUCE	root.2019st35	Wed Jul 17 23:50:57 -0700 2019	Wed Jul 17 23:51:26 -0700 2019	FINISHED	SUCCEEDED	<div></div>	<a href="#">History</a>
<a href="#">application_1563430216452_0012</a>	2019st35	GenTest2	MAPREDUCE	root.2019st35	Wed Jul 17 23:48:16 -0700 2019	Wed Jul 17 23:48:41 -0700 2019	FINISHED	SUCCEEDED	<div></div>	<a href="#">History</a>
<a href="#">application_1563430216452_0011</a>	2019st35	LogAnalysis	MAPREDUCE	root.2019st35	Wed Jul 17 23:46:24 -0700 2019	Wed Jul 17 23:47:45 -0700 2019	FINISHED	SUCCEEDED	<div></div>	<a href="#">History</a>
<a href="#">application_1563430216452_0007</a>	2019st35	SimpleAverage	MAPREDUCE	root.2019st35	Wed Jul 17 23:22:36 -0700 2019	Wed Jul 17 23:22:53 -0700 2019	FINISHED	SUCCEEDED	<div></div>	<a href="#">History</a>
<a href="#">application_1563430216452_0006</a>	2019st35	LinearRegression	MAPREDUCE	root.2019st35	Wed Jul 17 23:20:07 -0700 2019	Wed Jul 17 23:20:23 -0700 2019	FINISHED	SUCCEEDED	<div></div>	<a href="#">History</a>
<a href="#">application_1563430216452_0005</a>	2019st35	ExponentSmooth	MAPREDUCE	root.2019st35	Wed Jul 17 23:15:52 -0700 2019	Wed Jul 17 23:16:11 -0700 2019	FINISHED	SUCCEEDED	<div></div>	<a href="#">History</a>
Showing 1 to 9 of 9 entries (filtered from 509 total entries)										
First Previous 1 Next Last										



Cluster
 

About
 Nodes
 Node Labels
 Applications
 NEW
 NEW SAVING
 SUBMITTED
 ACCEPTED
 RUNNING
 FINISHED
 FAILED
 KILLED

 Scheduler
 Tools

## Application application\_1563430216452\_0011

Application Overview

User: 2019st35

Name: LogAnalysis

Application Type: MAPREDUCE

Application Tags:

YarnApplicationState: FINISHED

FinalStatus Reported by AM: SUCCEEDED

Started: Thu Jul 18 14:46:24 +0800 2019

Elapsed: 1mins, 21sec

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: 4122218 MB-seconds, 566 vcore-seconds

Show 20 entries

Attempt ID

Started

Node

Logs

appattempt\_1563430216452\_0011\_000001

Wed Jul 17 23:46:24 -0700 2019

[http://slave005:8042](#)

[Logs](#)

Showing 1 to 1 of 1 entries

First Previous 1 Next Last

[master01:8088/cluster/app/application\\_1563430216452\\_0005](#)



Cluster
 

About
 Nodes
 Node Labels
 Applications
 NEW
 NEW SAVING
 SUBMITTED
 ACCEPTED
 RUNNING
 FINISHED
 FAILED
 KILLED

 Scheduler
 Tools

## Application application\_1563430216452\_0005

Application Overview

User: 2019st35

Name: ExponentSmooth

Application Type: MAPREDUCE

Application Tags:

YarnApplicationState: FINISHED

FinalStatus Reported by AM: SUCCEEDED

Started: Thu Jul 18 14:15:52 +0800 2019

Elapsed: 18sec

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: 1500212 MB-seconds, 181 vcore-seconds

Show 20 entries

Attempt ID

Started

Node

Logs

appattempt\_1563430216452\_0005\_000001

Wed Jul 17 23:15:52 -0700 2019

[http://slave009:8042](#)

[Logs](#)

Showing 1 to 1 of 1 entries

First Previous 1 Next Last

[master01:8088/cluster/app/application\\_1563430216452\\_0006](#)



Cluster
 

About
 Nodes
 Node Labels
 Applications
 NEW
 NEW SAVING
 SUBMITTED
 ACCEPTED
 RUNNING
 FINISHED
 FAILED
 KILLED

 Scheduler
 Tools

## Application application\_1563430216452\_0006

Application Overview

User: 2019st35

Name: LinearRegression

Application Type: MAPREDUCE

Application Tags:

YarnApplicationState: FINISHED

FinalStatus Reported by AM: SUCCEEDED



CEPTED  
ENDING  
ISHED  
ILED  
LED  
JED  
JED

Started: Thu Jul 18 14:20:07 +0800 2019  
Elapsed: 16sec  
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: 1440582 MB-seconds, 175 vcore-seconds

Show 20 entries

Search:

Attempt ID	Started	Node	Logs
<a href="#">appattempt_1563430216452_0006_000001</a>	Wed Jul 17 23:20:07 -0700 2019	<a href="#">http://slave002:8042</a>	<a href="#">Logs</a>

Showing 1 to 1 of 1 entries

First Previous 1 Next Last

hadoop

## Application application\_1563430216452\_0007

#### Application Overview

User: 2019st35  
Name: SimpleAverage  
Application Type: MAPREDUCE  
Application Tags:  
YarnApplicationState: FINISHED  
FinalStatus Reported by AM: SUCCEEDED  
Started: Thu Jul 18 14:22:36 +0800 2019  
Elapsed: 17sec  
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: 1470465 MB-seconds, 178 vcore-seconds

Show 20 entries

Search:

Attempt ID	Started	Node	Logs
<a href="#">appattempt_1563430216452_0007_000001</a>	Wed Jul 17 23:22:36 -0700 2019	<a href="#">http://slave005:8042</a>	<a href="#">Logs</a>

Showing 1 to 1 of 1 entries


First Previous 1 Next Last

zels  
zns

\_SAVING  
MITTED  
PTED  
NING  
IHED  
ED  
ED

if





## Application application\_1563430216452\_0012

Logged in as: drwho

- Cluster
- About
- Nodes
- Node Labels
- Applications
- NEW
- NEW SAVING
- SUBMITTED
- ACCEPTED
- RUNNING
- FINISHED
- FAILED
- KILLED
- Scheduler
- Tools

**User:** 2019st35

**Name:** GenTest2

**Application Type:** MAPREDUCE

**Application Tags:**

**YarnApplicationState:** FINISHED

**FinalStatus Reported by AM:** SUCCEEDED

**Started:** Thu Jul 18 14:48:16 +0800 2019

**Elapsed:** 25sec

**Tracking URL:** [History](#)

**Diagnostics:**

**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:** 974088 MB-seconds, 138 vcore-seconds

Attempt ID	Started	Node	Logs
appattempt_1563430216452_0012_000001	Wed Jul 17 23:48:16 -0700 2019	http://slave003:8042	<a href="#">Logs</a>

Showing 1 to 1 of 1 entries



## Application application\_1563430216452\_0018

Logged in as: drwho

- Cluster
- About
- Nodes
- Node Labels
- Applications
- NEW
- NEW SAVING
- SUBMITTED
- ACCEPTED
- RUNNING
- FINISHED
- FAILED
- KILLED
- Scheduler
- Tools

**User:** 2019st35

**Name:** ComputeRMSE

**Application Type:** MAPREDUCE

**Application Tags:**

**YarnApplicationState:** FINISHED

**FinalStatus Reported by AM:** SUCCEEDED

**Started:** Thu Jul 18 14:53:11 +0800 2019

**Elapsed:** 17sec

**Tracking URL:** [History](#)

**Diagnostics:**

**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:** 2465150 MB-seconds, 302 vcore-seconds

Attempt ID	Started	Node	Logs
appattempt_1563430216452_0018_000001	Wed Jul 17 23:53:11 -0700 2019	http://slave009:8042	<a href="#">Logs</a>

Showing 1 to 1 of 1 entries

部分结果文件

## 6 任务分工

田广财：完成任务1-4，根据不同要求统计分析，生成对应文件。汤佳铭：生成前14天日志的训练集，和9月22日日志的真实统计结果。童成伟：根据训练集完成简单平均，指数平滑和线性回归3种预测算法。唐一鸣：由预测结果和真实结果计算RMSE，并分析优劣。

## 7 程序编译运行方式

### 程序打包

```
mvn clean
mvn install
```

### 任务运行

任务1-4 日志统计分析  
hadoop jar

任务5 生成训练集和22日真实值  
hadoop jar  
hadoop jar

任务5 预测算法(3种算法)  
(简单平均)hadoop jar  
(指数平滑)hadoop jar  
(线性回归)hadoop jar

任务5 计算RMSE(根据不同算法的预测可分别计算RMSE)  
hardoop jar  
hardoop jar  
hardoop jar

## 实验总结心得

通过本次实验，我们在对海量日志文件统计分析的过程了，对大数据处理任务编程及其应用的价值有了更深刻的认识，熟悉了MapReduce编程框架，强化了编程能力。同时在建立模型预测的过程中，学习到了不同的建模方法和建模方法。