# 实验报告

141070010 杜文钦

## 样例输入：

$ hadoop jar LogAnalyze.jar /input /out11 /out12 /out13 /out14

## 程序设计的主要流程

主类为 LogAnalyze, 四个任务由4个相互独立的类来完成。分别为：

LogAnalyze：

任务一：StatusCode

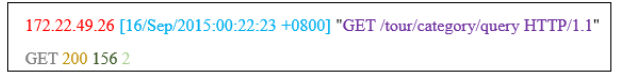
任务二：IpFrequency

任务三：UrlFrequency

任务四：UrlAccessAvg

除此之外，为了实现自定义MapReduce的输出，**实现了MyTextOutputFormat类**（继承自TextOutputFormat）。

## 任务一：



用正则表达式处理数据，如下：

Pattern p = Pattern.compile("\\d{4}:(\\d{2}).\*?\".\*?(\\d+) ");

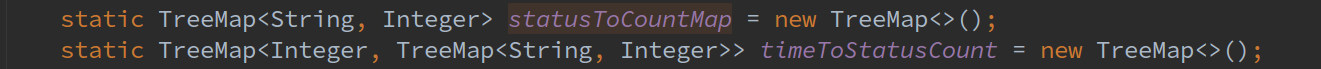
其中，第一个括号内匹配了时间(小时)后，用.\*?\”跳过数字0800，之后的第二个括号后跟着一个空格，是为了不匹配到HTTP/后的数字。

Map 输出的 Key 是 “时间#状态码”，Value是 1；（时间是某一个小时）

Combine 是对所有 Key 相同Value进行累加；

Partition是用“时间”进行Partition；

Reduce继续对Key相同的Value进行累加，并对statusToCountMap 和timeToStatusCount 这两个变量进行赋值。然后在 cleanup() 时根据这两个变量进行输出。



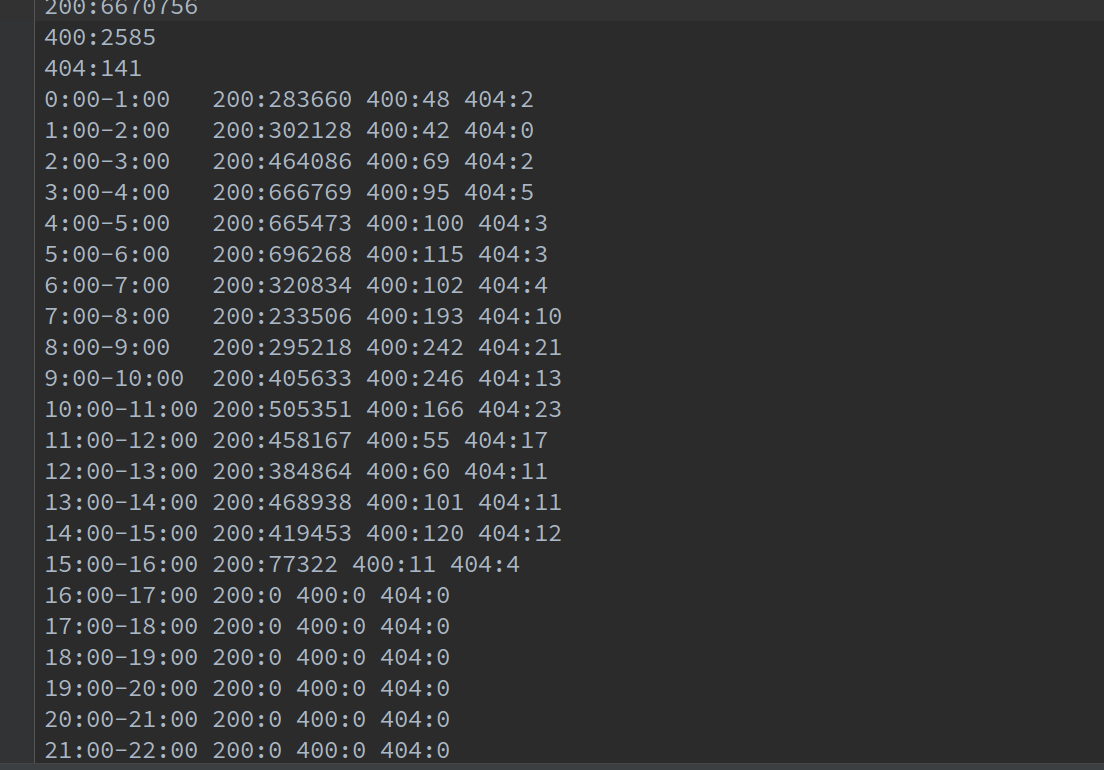
说明：

statusToCountMap的 key 是 状态码，value 是该状态码出现的总次数；

timeToStatusCount的 key 是 时间（某一小时）；value 是个 TreeMap，其 key 为状态码，value 是该状态码出现的次数。

（选择 TreeMap 是为了确保状态码和时间的输出是有序的）

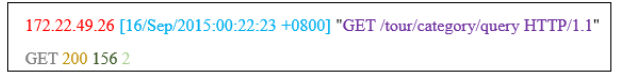
### 输出结果：



可以看到，满足实验要求。

## 任务二：

原理和任务一相似，主要不同在于正则表达式：



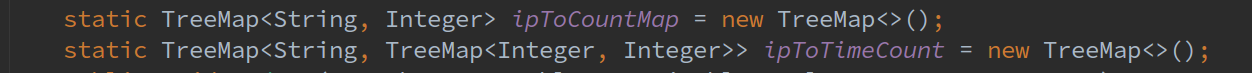
Pattern p = Pattern.compile("(\\d{3}\\.\\d{2}\\.\\d{2}\\.\\d{2}).\*?:(\\d{2})");

Map输出的 Key 是 “IP地址#时间”，Value是1；（时间是某一个小时）

Combine 是对所有 Key 相同Value进行累加；

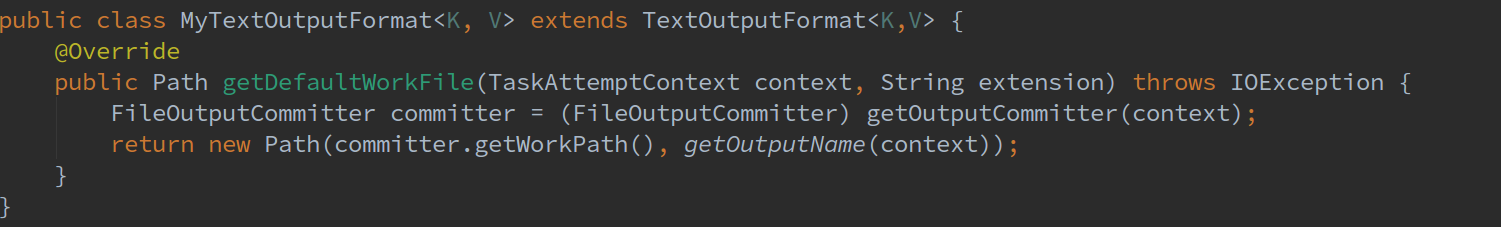
Partition是用“IP地址”进行Partition；

Reduce继续对Key相同的Value进行累加，并对ipToCountMap 和 ipToTimeCount 这两个变量进行赋值，然后在 cleanup() 时根据这两个变量进行输出。

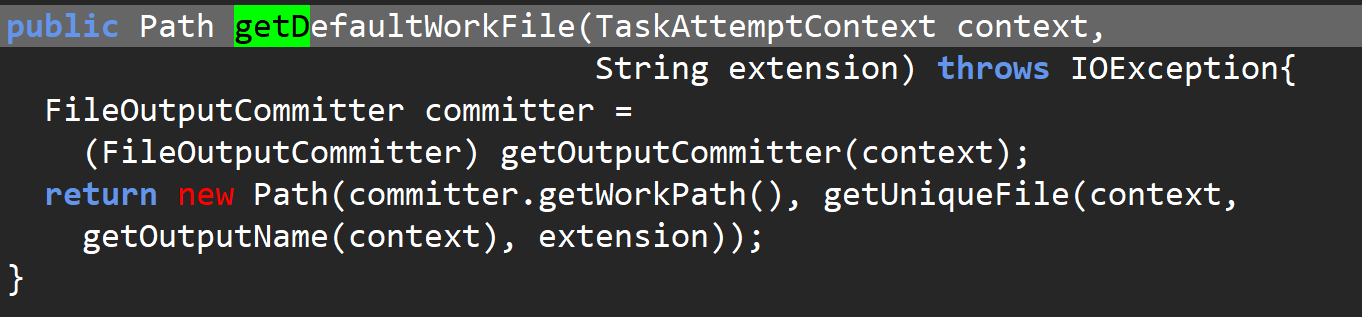


最后，为了实现按ip地址分文件输出：

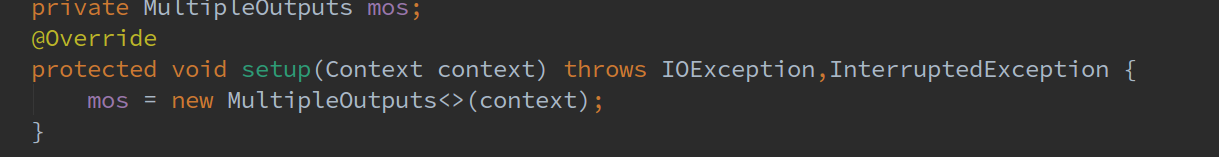
1. 定义一个新输出格式类**MyTextOutputFormat类**，继承自TextOutputFormat。

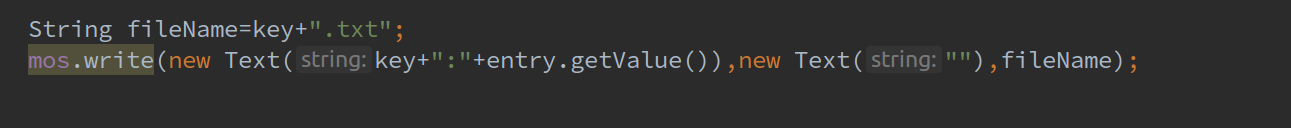


这里重写了getDefaultWorkFile，下面图截取自Hadoop源代码文件的FileOutputFormat.java，由代码可知，默认的输出文件名带后缀。

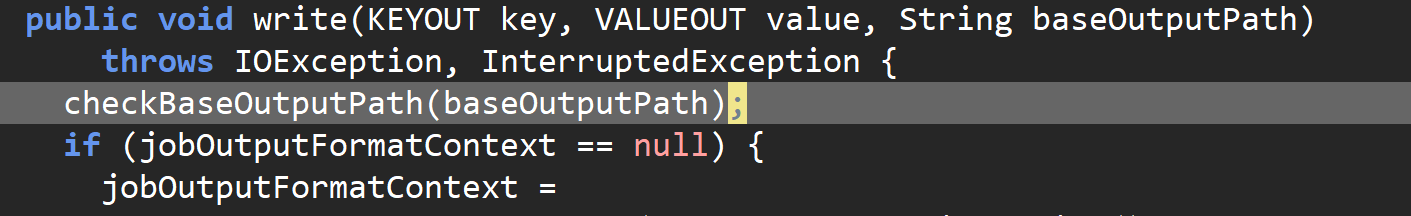


2、采用MultipleOutputs类，以key（对应ip）加上”.txt”作为文件名





下面是MultipleOutputs类的源代码的write函数。

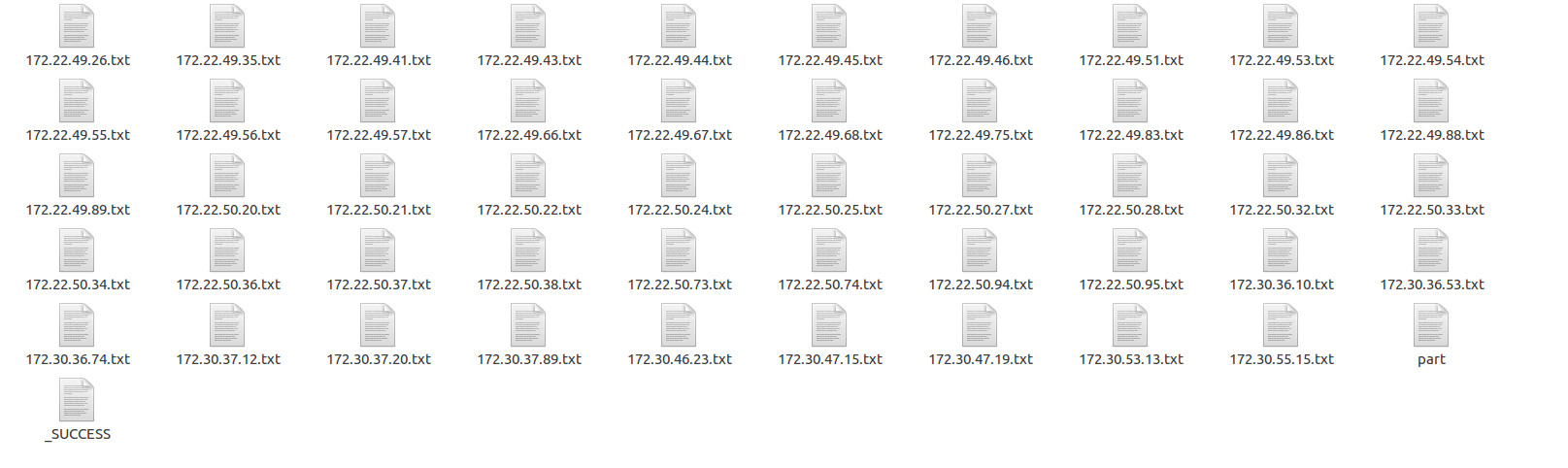


说明：

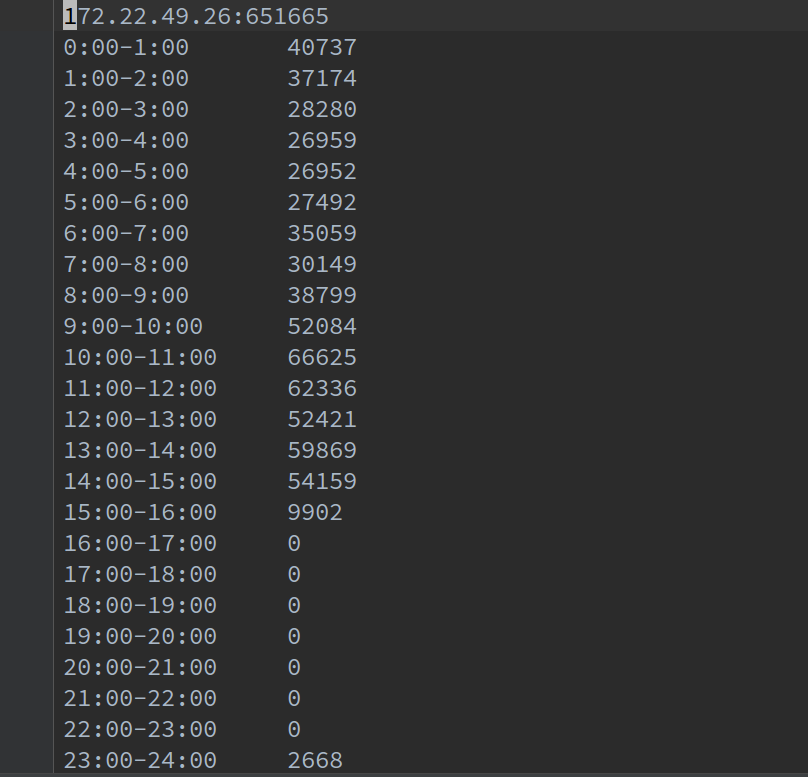
ipToCountMap 的 key 是IP地址，value 是该IP地址访问URL的总次数；

ipToTimeCount 的 key 是IP地址，value是个TreeMap，其key是时间（某一小时），value是该小时内访问URL的次数。

### 输出结果：



以 172.22.49.6.txt为例：



可以看到，满足实验要求。

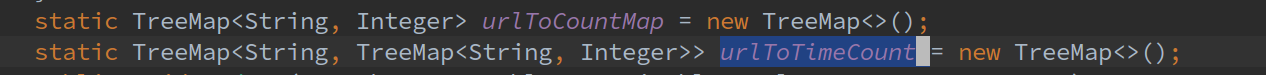
## 任务三：

Map 输出的 Key 是 “URL#时间”，Value是 1；（时间精确到某一秒）

Combine 是对所有 Key 相同Value进行累加；

Partition是用“URL”进行Partition；

Reduce继续对Key相同的Value进行累加，并对 urlToCountMap 和 urlToTimeCount 这两个变量进行赋值。然后在 cleanup() 时根据这两个变量进行输出。



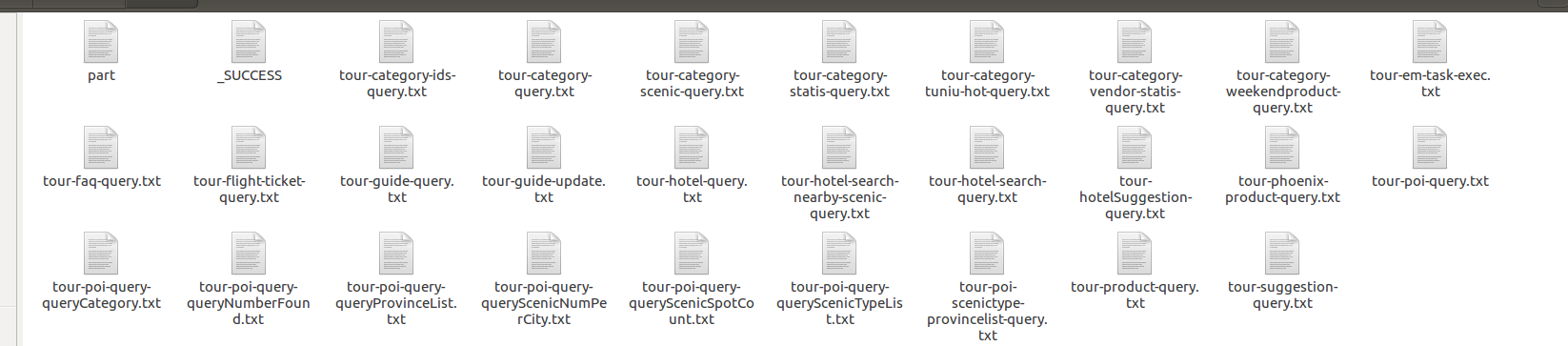
说明：

urlToCountMap 的 key 是URL，value是该URL被访问的总次数；

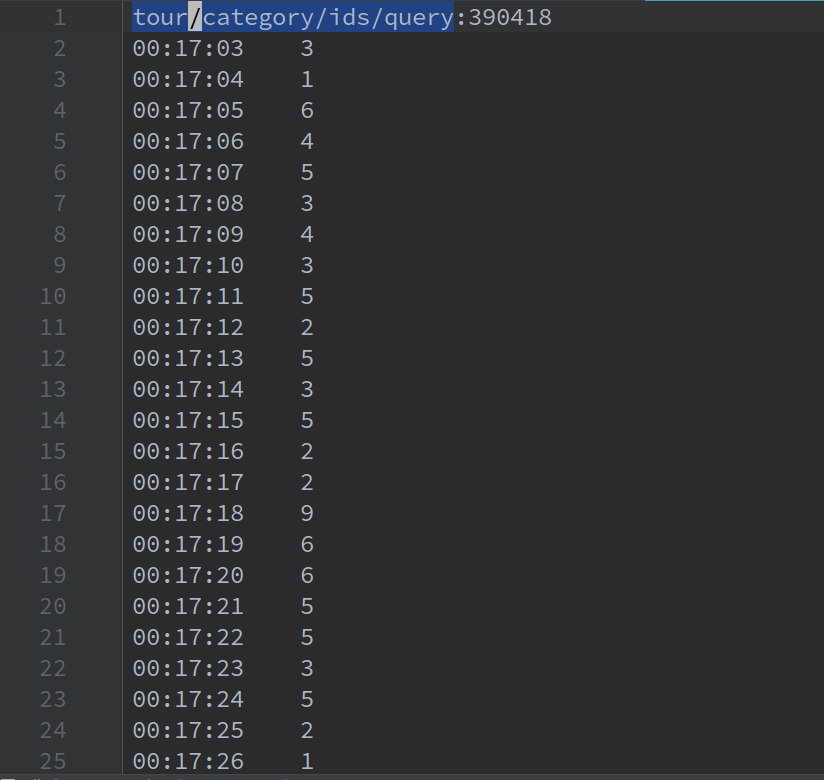
urlToTimeCount 的 key是URL，value是个TreeMap，其key是时间，value是URL在该时间被访问的次数。

（urlToTimeCount 的value是TreeMap是为了时间能有序输出）

### 输出结果：



以tour/category/ids/query.txt 为例，内容如下所示：

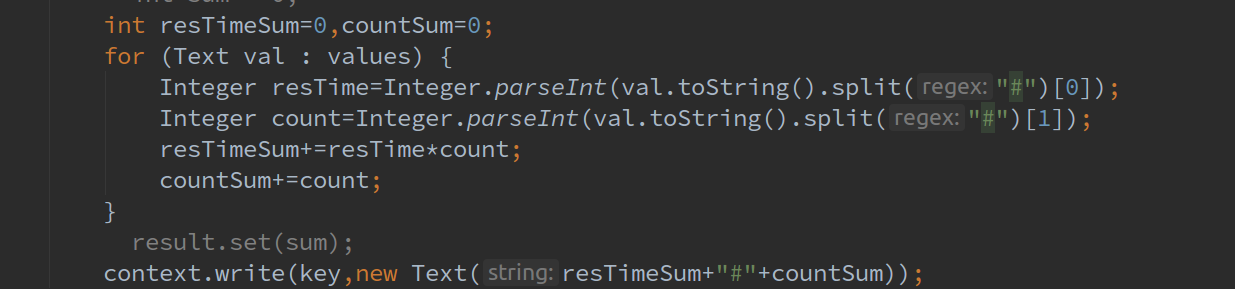


可以看到，满足实验要求。

## 任务四：

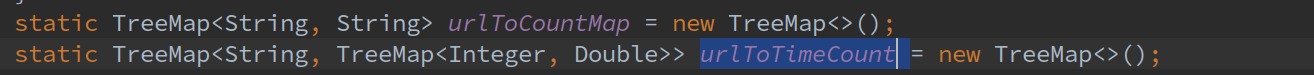
Map 输出的 Key 是 “URL#时间”，Value是 “响应时间#1”；（时间是某一个小时）

Combine 是对所有 Key 相同的Value进行累加，累加时，需要拆开Value，如下所示：



Partition是用“URL”进行Partition；

Reduce继续对Key相同的Value进行累加，并对 urlToCountMap 和 urlToTimeCount 这两个变量进行赋值。然后在 cleanup() 时根据这两个变量进行输出。

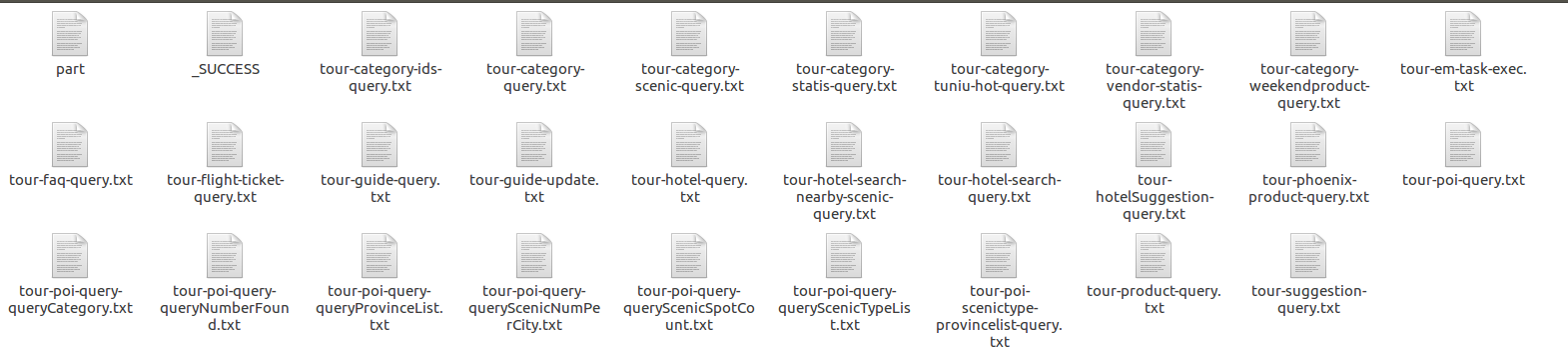


说明：

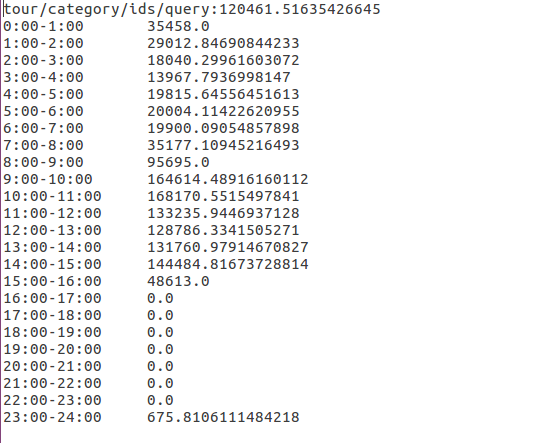
urlToCountMap 的 key 是URL，value是该URL的总的响应时间#总的响应次数；

urlToTimeCount 的 key 是URL，value是个TreeMap，其key是时间（某个小时），value是该小时的平均响应时间。

### 输出结果：



以tour-category-ids-query.txt为例，其内容如下所示：



可以看到，满足实验要求。