

实验三 MapReduce 编程实验

一、	实验目的.....	2
1.	掌握 MapReduce 编程基本思想.....	2
2.	学习如何编写 MapReduce 程序.....	2
二、	实验内容.....	2
1.	熟悉 Hadoop 开发包.....	2
2.	编写 MapReduce 程序.....	3
3.	调试和运行 MapReduce 程序.....	3
4.	完成布置的作业.....	3
三、	实验过程.....	3
1.	在 Eclipse 中引入 hadoop 开发包.....	3
2.	编写 MapReduce 程序.....	4
1)	Log.java	4
2)	SearchLogAnalyze.java	4
3.	打包 MapReduce 程序.....	5
四、	运行结果.....	7
1.	-h	7
2.	-NumOfRecord	7
3.	-StartDate/-EndDate	8
4.	-StartIp/-EndIp	9
5.	-IpPrefix	9
6.	-TopK	10
五、	遇到的问题及解决方法.....	10
1.	conf.addResource() 路径.....	10
2.	writeChars() 乱码	11
六、	实验分工.....	11
七、	心得和体会.....	12

一、 实验目的

1. 掌握 MapReduce 编程基本思想
2. 学习如何编写 MapReduce 程序

二、 实验内容

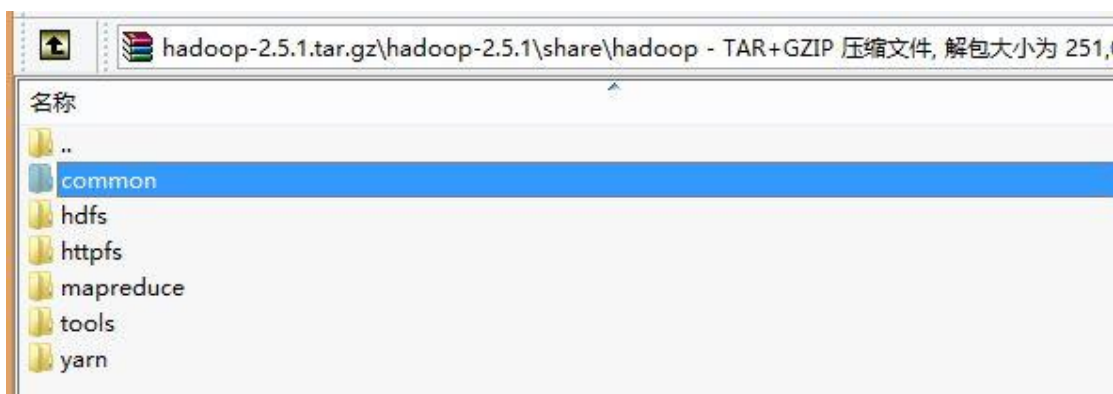
1. 熟悉 Hadoop 开发包

直接解压 **Hadoop2.5.1** 压缩包后，可看到如下图所示的目录结构，其中比较重要的目录有 **bin,sbin,etc,lib,share** 等，下面分别简要介绍这几个目录的作用。



- **bin,sbin**: 包含一些运行 **hadoop** 的脚本程序，例如：
hadoop:最基本且功能最完备的脚本，其它大部分脚本都会调用该脚本
start-all.sh/stop-all.sh:启动/停止所有结点上的 **HDFS** 和 **MapReduce** 相关服务
start-mapred.sh/stop-mapred.sh:单独启动/停止 **MapReduce** 相关服务
start-dfs.sh/stop-dfs.sh:单独启动/停止 **HDFS** 相关服务
start-yarn.sh/stop-yarn.sh:单独启动/停止 **yarn** 相关服务
- **etc**: **hadoop** 配置文件所在目录，例如
hadoop-env.sh: 设置 **hadoop** 总体环境变量
core-site.xml: 设置基础公共库
hdfs-site.xml: 配置分布式文件系统 **HDFS**
mapred-env.sh; **MapReduce** 计算框架配置选项
yarn-site.xml: 配置 **hadoop** 管理工具 **yarn** 等
- **lib,libexec**: 包含 **hadoop** 运行时依赖的第三方库，主要是 **c** 语言的动态库，在 **hadoop** 启动或者用户提交作业时会自动加载这些库。
- **share**:包含 **doc** 和 **hadoop** 两个文件夹，其中 **doc** 文件夹是 **hadoop** 的 **api** 介绍文件，例如 **html** 等；**hadoop** 文件夹里面是我们开发 **hadoop** 程序所需要

的一些开发包，**hadoop** 文件夹结构如下图：



Common: 主要文件 **hadoop-common-2.5.1.jar**，包含一些基础公共库

Hdfs: 主要文件 **hadoop-hdfs-2.5.1.jar**，包含分布式文件系统操作 **HDFS** 操作相关的类库

Mapreduce: **hadoop-mapreduce-client-core-2.5.1.jar**，包含 **mapreduce** 框架相关的类库

这个程序中我们只需要 **hadoop-common-2.5.1.jar** 和 **hadoop-mapreduce-client-core-2.5.1.jar** 两个 **jar** 包。

2. 编写 MapReduce 程序

详见附件

3. 调试和运行 MapReduce 程序

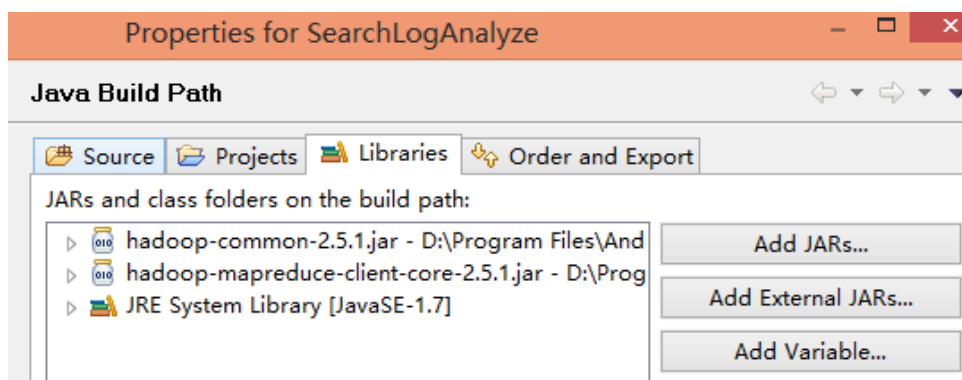
详见[遇到的问题及解决方法](#)部分

4. 完成布置的作业

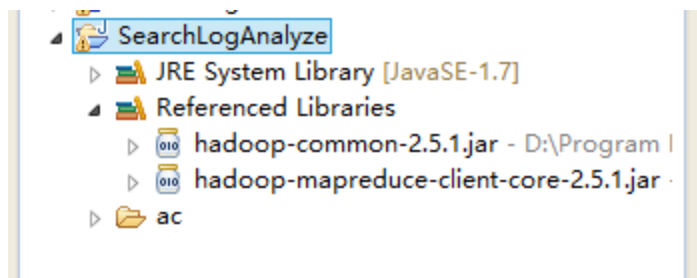
三、 实验过程

1. 在 Eclipse 中引入 hadoop 开发包

首先可以在 Eclipse 中新建一个工程，例如 SearchLogAnalyze，然后在工程中引入 Hadoop 开发包。可以按该步骤引入相关包：右键工程—> Build Path—>Configure Build Path, 然后通过添加 Jars 引入相关包。



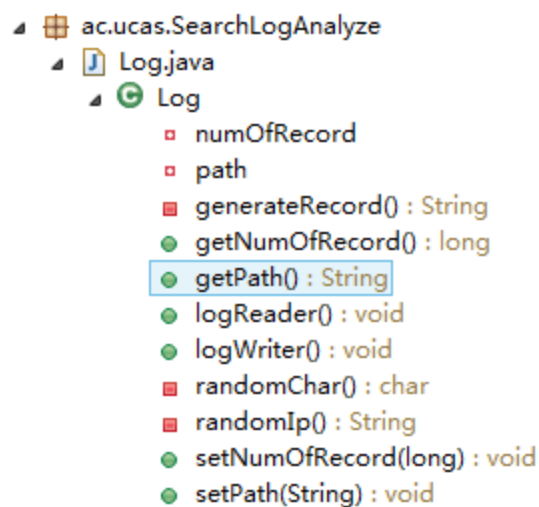
添加库成功后，在工程中的 Referenced Libraries 中可以查看到这些 jar 包。



2. 编写 MapReduce 程序

1) Log.java

Log 类的类视图如下：其中红色表示私有，绿色表示公有



成员变量：

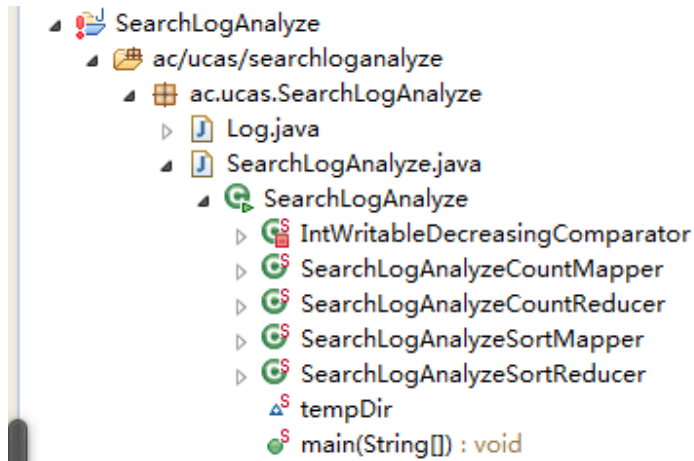
- **numOfRecord**：检索记录的条目，即检索次数
- **path**：保存检索日志的文件路径

成员方法：

- **get/set**：成员变量的获取器和设置器
- **randomIp()**：随机生成一个 **ip** 地址
- **randomChar()**：随机生成一个字符，包括大小写字母和数字
- **generateRecord()**：随机生成一次检索记录
- **logWriter()**：生成检索日志，并写入 **HDFS** 文件中
- **logReader()**：从 **HDFS** 中读取检索日志

2) SearchLogAnalyze.java

SearchLogAnalyze 类视图如下：

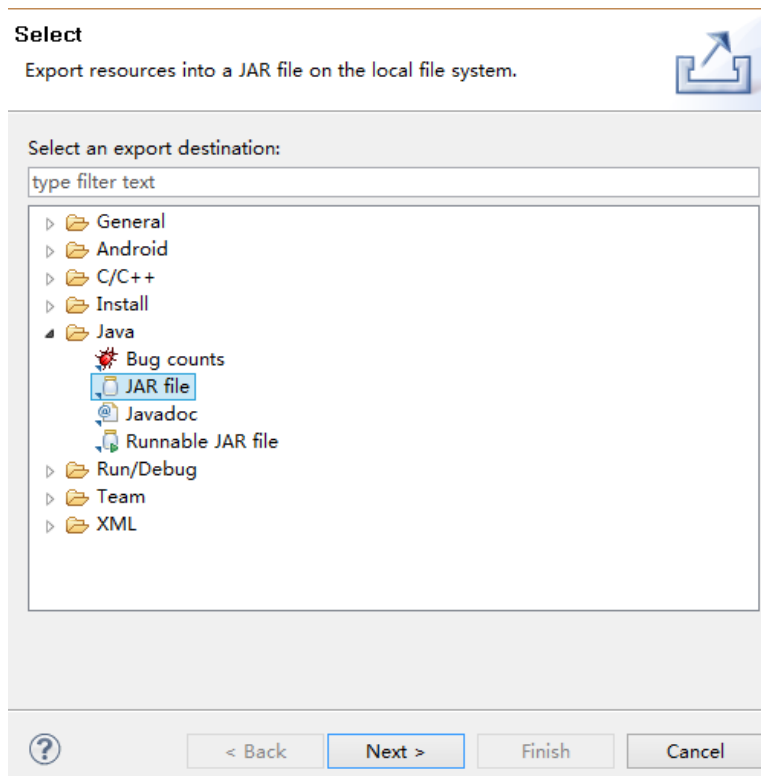


- **IntWritableDecreasingComparator**: 降序比较器，用于第二个 **job Map** 后的 **sort** 阶段，使得 **reduce** 输出能按照检索次数降序排序
- **SearchLogAnalyzeCountMapper**: 第一个作业的 **Mapper**，用于统计检索次数
- **SearchLogAnalyzeCountReducer**: 第一个作业的 **Reducer**，用于将符合检索条件的检索条目写入 **hdfs** 临时文件，作为第二个作业的输入。
- **SearchLogAnalyzeSortMapper**: 第二个作业的 **Mapper**，从第一个作业的输出文件中读取符合条件的检索条目，按照检索次数降序排序
- **SearchLogAnalyzeSortReducer**: 第二个作业的 **Reducer**，用于交换 **Mapper** 输出的 **key** 和 **value**，使得按照 **query query-times** 键值对输出结果。

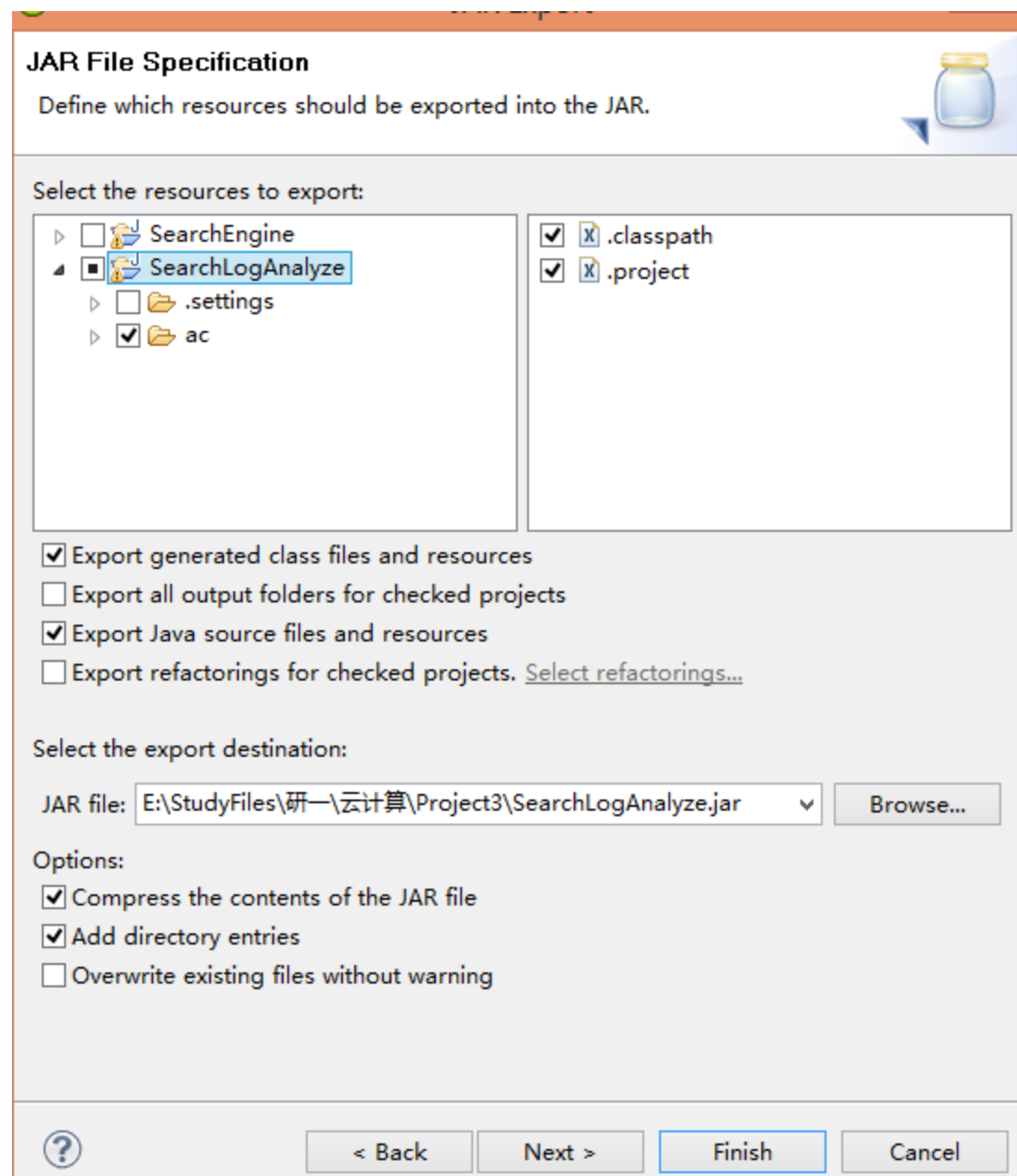
各个类的具体实现及解释请查看源程序及其注释。

3. 打包 MapReduce 程序

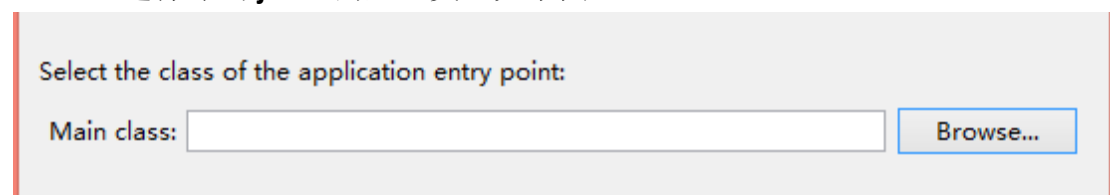
右击工程->Export



选择 **JAR file**，点击 **next** 后如下图：



选择需要导出的文件，如果不需要源文件，只需要选中“**Export generated class files and resources**”，否则还需选中“**Export java source file and resources**”，在 **Select the export destinations** 中填写导出的文件名和所在目录，点击两次 **next**，点击 **Browse** 选择导出 **jar** 包的入口类，如下图：



选择 **SearchLogAnalyze**，点击 **ok->Finish** 即可。

四、 运行结果

1. -h

启动 **hadoop** 后，运行相关命令 **hadoop jar SearchLog.jar -h**,结果如下

```
user1@Lenovo:~/Desktop$ hadoop jar SearchLog.jar -h
Usage: SearchLog.jar <int> <out> [-parameters]
-NumOfRecord    long          set the number of searchlog records(default:10)
-LogFileName    String        set the filename of logfile(default:<in>/SearchLog.txt)
-TopK           int           Return the top K results(default:50)
-StartDate      yyyy-MM-dd    set the StartDate of filter(default:2000-01-01)
-EndDate        yyyy-MM-dd    set the EndDate of filter(default:2014-12-30)
-IpPrefix       x.x.x         set the IpPrefix of filter(default:null)
-StartIp        int           set the StartIp of filter(default:0)
-EndIp          int           set the EndIp of filter(default:127)
user1@Lenovo:~/Desktop$ |
```

提示需要提供参数输入路径，输出路径和其它一些参数。

先创建 **MapReduce** 输入目录 **SearchLogAnalyze/input**，截图如下：

```
user1@Lenovo:~/Desktop$ hdfs dfs -mkdir /user/user1
14/11/03 14:30:58 WARN util.NativeCodeLoader: Unable to load native-hadoop library f
user1@Lenovo:~/Desktop$ hdfs dfs -mkdir /user/user1/SearchLogAnalyze
14/11/03 14:31:08 WARN util.NativeCodeLoader: Unable to load native-hadoop library f
user1@Lenovo:~/Desktop$ hdfs dfs -mkdir /user/user1/SearchLogAnalyze/input
14/11/03 14:31:14 WARN util.NativeCodeLoader: Unable to load native-hadoop library f
user1@Lenovo:~/Desktop$ |
```

说明:WARN 是因为安装的 **hadoop** 本地库是 64 位的(可通过运行命令：**file hadoop_home/lib/native/libhadoop.so** 查看)，而运行 **hadoop** 的系统是 32 位的，解决方法是下载 **hadoop** 源码，在本地重新编译后，将编译得到的本地库拷贝到相应目录。

2. - NumOfRecord

要对十个查询进行统计，输入命令：

hadoop jar SearchLog.jar SearchLogAnalyze/input SearchLogAnalyze/ouput - NumOfRecord 10

运行结果如下图，根据红色线标记可以看到，启动了一个 **map** 和一个 **reduce** 任务，**map** 和 **reduce** 的记录都是 10 个，**CPU** 的运行时间等其他相关信息。

```
Job Counters
  Launched map tasks=1
  Launched reduce tasks=1
  Data-local map tasks=1
  Total time spent by all maps in occupied slots (ms)=3360
  Total time spent by all reduces in occupied slots (ms)=3666
  Total time spent by all map tasks (ms)=3360
  Total time spent by all reduce tasks (ms)=3666
  Total vcore-seconds taken by all map tasks=3360
  Total vcore-seconds taken by all reduce tasks=3666
  Total megabyte-seconds taken by all map tasks=3440640
  Total megabyte-seconds taken by all reduce tasks=3753984

Map-Reduce Framework
  Map input records=10
  Map output records=10
  Map output bytes=75
  Map output materialized bytes=101
  Input split bytes=124
  Combine input records=10
  Combine output records=10
  Reduce input groups=10
  Reduce shuffle bytes=101
  Reduce input records=10
  Reduce output records=10
  Spilled Records=20
  Shuffled Maps =1
  Failed Shuffles=0
  Merged Map outputs=1
  GC time elapsed (ms)=72
  CPU time spent (ms)=1920
  Physical memory (bytes) snapshot=383729664
  Virtual memory (bytes) snapshot=829177856
  Total committed heap usage (bytes)=312475648

Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0
```

3. -StartDate/-EndDate

先删除之前的 **output** 文件夹和 **input** 文件夹里面的 **SearchLog.txt** 日志文件，下面运行其它测试之前同样。

查找 **10** 个查询记录中在 **2003** 年到 **2010** 年之间的统计信息，输入命令：

```
hadoop jar SearchLog.jar SearchLogAnalyze/input SearchLogAnalyze/output -  
NumOfRecord 10 -StartDate 2003-01-01 -EndDate 2010-12-30
```

```
user1@Lenovo:~/Desktop$ hadoop jar SearchLog.jar SearchLogAnalyze/input SearchLogAnalyze/output -NumOfRecord 10 -StartDate 2003-01-01 -EndDate 2010-12-30
```

运行结果如下：


```

user1@Lenovo:~/Desktop$ hdfs dfs -cat SearchLogAnalyze/input/SearchLog.txt
14/11/03 14:46:36 WARN util.NativeCodeLoader: Unable to load native-hadoop lib
2008-08-21 00:44:33 9XEg 101.125.46.119
2002-09-18 08:36:57 YnI 25.31.107.1
2008-03-11 08:53:46 g78d 90.11.86.78
2006-05-09 15:20:50 i9 74.84.54.78
2006-09-05 19:05:16 ZkPh 108.89.28.124
2012-11-28 15:28:23 El 113.0.55.115
2009-02-04 05:25:45 K0 51.7.16.2
2001-01-27 06:14:23 7to0 24.53.94.31
2000-04-20 04:47:03 1Mx5 114.125.26.106
2003-04-23 02:56:55 8X 61.45.73.44
user1@Lenovo:~/Desktop$ hdfs dfs -cat SearchLogAnalyze/output/part-r-00000
14/11/03 14:46:44 WARN util.NativeCodeLoader: Unable to load native-hadoop lib
i9      1
g78d    1
ZkPh    1
K0      1
9XEg    1
8X      1
user1@Lenovo:~/Desktop$ |

```

SearchLog.txt 里面记录了随机生成的 **10** 次查询，**part-r-00000** 输出文件中显示了在 **2003** 年到 **2010** 年范围内的五个检索记录。

4. -StartIp/-EndIp

查询 **10** 个检索记录中 **ip** 地址的最后一部分在范围 **50** 到 **100** 的部分，输入命令：

hadoop jar SearchLog.jar SearchLogAnalyze/input SearchLogAnalyze/output -NumOfRecord 10 -StartIp 50 -EndIp 100

```

user1@Lenovo:~/Desktop$ hadoop jar SearchLog.jar SearchLogAnalyze/input SearchLogAnalyze/output -NumOfRecord 10 -StartIp 50 -EndIp 100

```

运行结果如下：

```

user1@Lenovo:~/Desktop$ hdfs dfs -cat SearchLogAnalyze/input/SearchLog.txt
14/11/03 14:49:16 WARN util.NativeCodeLoader: Unable to load native-hadoop
2003-11-05 19:58:42 jK 69.47.11.27
2005-10-20 15:53:52 fXTc 16.123.56.10
2006-03-21 15:58:57 d 109.64.125.24
2013-09-18 02:32:26 0a 52.51.34.2
2011-09-13 03:30:12 v0 59.49.110.49
2004-10-25 18:18:41 en 92.112.84.111
2004-09-23 12:48:28 oq 52.62.21.39
2003-10-10 22:38:29 LQ 69.87.7.22
2005-06-06 03:56:27 e 15.114.13.45
2009-05-22 04:38:30 ircG 24.54.79.91
user1@Lenovo:~/Desktop$ hdfs dfs -cat SearchLogAnalyze/output/part-r-00000
14/11/03 14:49:23 WARN util.NativeCodeLoader: Unable to load native-hadoop
ircG    1
user1@Lenovo:~/Desktop$ |

```

5. -IpPrefix

统计 **10** 个查询中 **ip** 前缀是 **127.127.127** 的查询，输入命令：

hadoop jar SearchLog.jar SearchLogAnalyze/input SearchLogAnalyze/output -NumOfRecord 10 -IpPrefix 127.127.127

```
user1@Lenovo:~/Desktop$ hadoop jar SearchLog.jar SearchLogAnalyze/input SearchLogAnalyze/output -NumOfRecord 10 -IpPrefix 127.127.127
```

运行结果如下：

```
user1@Lenovo:~/Desktop$ hdfs dfs -cat SearchLogAnalyze/input/SearchLog.txt
14/11/03 14:51:49 WARN util.NativeCodeLoader: Unable to load native-hadoop
2001-02-20 20:54:41 lQKt 122.116.74.18
2008-07-06 22:23:11 nv 50.114.0.78
2013-11-25 03:30:05 v00 36.124.25.74
2006-02-11 20:58:49 A9M 78.5.74.14
2011-03-11 00:45:30 G 19.5.39.88
2008-02-07 18:13:13 n 57.98.89.108
2006-08-03 08:33:09 q 55.122.122.41
2006-06-20 22:13:24 K4hH 81.110.64.2
2000-08-07 22:00:53 ZV 17.19.88.119
2006-09-11 07:13:26 J 121.44.53.77
user1@Lenovo:~/Desktop$ hdfs dfs -cat SearchLogAnalyze/output/part-r-00000
14/11/03 14:51:56 WARN util.NativeCodeLoader: Unable to load native-hadoop
user1@Lenovo:~/Desktop$ |
```

在上面的 10 次检索中，没有 ip 前缀是 127.127.127 的。

6. - TopK

统计 1000 个查询中出现次数最多的 10 个，运行命令：

```
hadoop jar SearchLog.jar SearchLogAnalyze/input SearchLogAnalyze/output -  
NumOfRecord 1000 -TopK 10
```

```
user1@Lenovo:~/Desktop$ hadoop jar SearchLog.jar SearchLogAnalyze/input SearchLogAnalyze/output -NumOfRecord 1000 -TopK 10
```

结果如下：

```
user1@Lenovo:~/Desktop$ hdfs dfs -cat SearchLogAnalyze/output/part-r-00000
14/11/03 14:54:03 WARN util.NativeCodeLoader: Unable to load native-hadoop
r      10
D      10
G      9
O      8
Z      8
c      8
d      7
n      7
X      7
j      7
user1@Lenovo:~/Desktop$ |
```

五、 遇到的问题及解决方法

1. conf.addResource() 路径

➤ 错误

使用 `conf.addResource("path of xml file")` 时，
`context.getConfituration().get("parameter")` 总是返回 `null`。

➤ 原因

"path of xml file" 路径不对，如果是绝对路径的话应该是 `/user/$username/.../`，而

相对路径是相对于 `/user/$username` 的。

➤ 解决方法

1: 设置正确的路径

2: 使用 `conf.set("parameter name","value")`，这里采用了这个方法

2. `writeChars()` 乱码

➤ 错误

使用 `writeChars()` 向 HDFS 文件写入数据后，使用 `hdfs dfs -cat` 命令查看不会乱码，但是 `cpToLocal` 和 `Log.logReader()` 中都是乱码

➤ 原因

没有设置正确的编码，`cpToLocal` 和 `FileSystem` 读取文件时编码和 `writeChars()` 不同。

➤ 解决方法

指定读写编码，源代码如下：（红色标记部分）

```
public void logWriter() throws IOException{
    FileSystem fs = FileSystem.get( new Configuration() );
    FSDataOutputStream fos = fs.create(new Path(getPath()), true);
    Writer out = new OutputStreamWriter(fos, "utf-8");
    for(long i = 0; i < numOfRecord; i++){
        out.write(generateRecord() + "\n");
    }
    out.close();
    fos.close();
}

/*读取日志文件到stdout*/
public void logReader() throws IOException{
    FileSystem fs = FileSystem.get(new Configuration());
    FSDataInputStream fis = fs.open(new Path(getPath()));
    InputStreamReader isr = new InputStreamReader(fis, "utf-8");
    BufferedReader br = new BufferedReader(isr);
    String line = "";
    while((line = br.readLine()) != null){
        System.out.println(line);
    }
    br.close();
    isr.close();
    fis.close();
}
```

六、 实验分工

在做这个实验之前，我们先认真复习了老师的讲课内容，在此基础上大家分别通过网络或者图书，更加深入地了解了MapReduce作业的整个流程，尤其是map，shuffle，combine，sort和reduce各个阶段的具体流程。通过mapreduce的学习，完成了实验的设计，代码编写，调试和运行，最后撰写实验报告。

➤ 罗远浩：设计Log类和SearchLogAnalyze类，并完成代码编写。

- 薛杰瑜：对程序进行调试，运行并做记录
- 张慧玲：撰写实验报告和 PPT

七、心得和体会

- 罗远浩：通过这个实验，我加深了对Hadoop的了解，主要是开发包的目录结构；另外对MapReduce程序的总体架构和流程有了更清楚的认识；同时，我也更加认识到在解决一个项目时团队合作的重要性。通过团队合作，不仅能提高项目完成的效率和质量，也能锻炼我们协调和沟通的能力，为我们以后进行大项目的开发积累了宝贵的经验。
- 薛杰瑜：掌握 MapReduce 编程基本思想，初步学习了编写 MapReduce 的简单程序。熟悉了在 linux 环境下的 hadoop 平台，和 MapReduce 开发环境
- 张慧玲：**MapReduce** 的设计目标是方便编程人员在不熟悉分布式并行编程的情况下,将自己的程序运行在分布式系统上。当前的软件实现是指定一个 **Map**（映射）函数，用来把一组键值对映射成一组新的键值对，指定并发的 **Reduce**（化简）函数，用来保证所有映射的键值对中的每一个共享相同的键组。在此次实验过程中，我们学习了 **mapreduce** 代码的编写，对 **mapreduce** 有了一定的了解，虽然在实验中也遇到了困难，但每个困难都是思维的再一次创新。
- 总结：通过这个实验，大家都有所收获：学识上，通过实验，把老师课上讲的东西实践一下，加深了对课堂知识的理解和掌握；团队协作上，大家更加懂得了在一个项目中和他人合作交流的重要性，锻炼了我们的表达，沟通和协调能力。

罗远浩(2014E8013261184)

薛杰瑜(2014E8013261174)

张慧玲(201428018743021)

2014-11-04