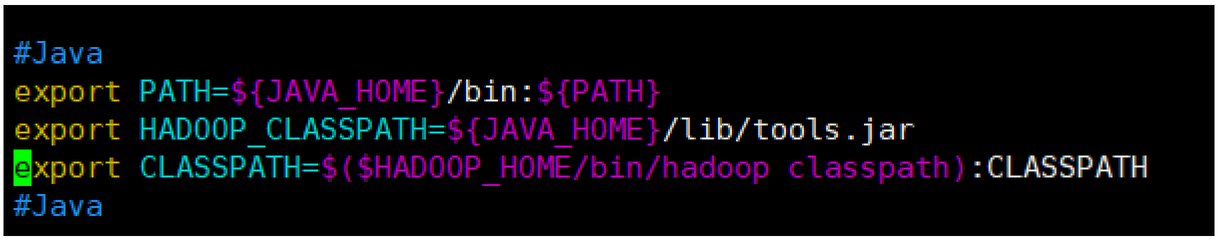
运行WordCount

1. 编辑WordCount.java

```java
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
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.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/**
 * WordCount: MapReduce初级案例，按八股文的结构遍写
 * @author johnnie
 *
 */
public class WordCount {

    /**
     * Mapper区: WordCount程序 Map 类
     * Mapper<KEYIN, VALUEIN, KEYOUT, VALUEOUT>:
     *         |        |            |              |
     *   输入key类型  输入value类型      输出key类型 输出value类型
     * @author johnnie
     *
     */
    public static class TokenizerMapper extends Mapper<LongWritable, Text, Text,
IntWritable>{
        // 输出结果
        private Text word = new Text();                              // KEYOUT
        // 因为若每个单词出现后，就置为 1，并将其作为一个<key,value>对，因此可以声明为常量，值
为 1
        private final static IntWritable one = new IntWritable(1);  // VALUEOUT

        /**
         * value 是文本每一行的值
         * context 是上下文对象
         */
        @Override
        public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
            // 获取每行数据的值
            String lineValue = value.toString();
            // 分词：将每行的单词进行分割,按照"  \t\n\r\f"(空格、制表符、换行符、回车符、换页)
进行分割
            StringTokenizer tokenizer = new StringTokenizer(lineValue);
            // 遍历
            while (tokenizer.hasMoreTokens()) {
                // 获取每个值
                String wordValue = tokenizer.nextToken();
                // 设置 map 输出的 key 值
                word.set(wordValue);
                // 上下文输出 map 处理结果
                context.write(word, one);
            }
        }
    }

    /**
     * Reducer 区域：WordCount 程序 Reduce 类
     * Reducer<KEYIN, VALUEIN, KEYOUT, VALUEOUT>:Map 的输出类型，就是Reduce 的输入类型
     * @author johnnie
     *
     */
    public static class IntSumReducer extends Reducer<Text, IntWritable, Text,
IntWritable> {
        // 输出结果：总次数
        private IntWritable result = new IntWritable();

        @Override
        public void reduce(Text key, Iterable<IntWritable> values, Context context)
throws IOException, InterruptedException {
            int sum = 0;                        // 累加器，累加每个单词出现的总次数
            // 遍历values
            for (IntWritable val : values) {
                sum += val.get();               // 累加
            }
            // 设置输出 value
            result.set(sum);
            // 上下文输出 reduce 结果
            context.write(key, result);
        }
    }

    // Driver 区：客户端
    public static void main(String[] args) throws Exception {
        // 获取配置信息
        Configuration conf = new Configuration();
        // 创建一个 Job
        Job job = Job.getInstance(conf, "word count");      // 设置 job name 为 word
count
//      job = new Job(conf, "word count");                  // 过时的方式
        // 1. 设置 Job 运行的类
        job.setJarByClass(WordCount.class);

        // 2. 设置Mapper类和Reducer类
        job.setMapperClass(TokenizerMapper.class);
        job.setReducerClass(IntSumReducer.class);

        // 3. 获取输入参数，设置输入文件目录和输出文件目录
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        // 4. 设置输出结果 key 和 value 的类型
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
//      job.setCombinerClass(IntSumReducer.class);

        // 5. 提交 job，等待运行结果，并在客户端显示运行信息，最后结束程序
        boolean isSuccess = job.waitForCompletion(true);

        // 结束程序
        System.exit(isSuccess ? 0 : 1);
    }

}
```

2. 设置环境变量

```
vim ~/.bashrc
```



```
#Java
export PATH=${JAVA_HOME}/bin:${PATH}
export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar
export CLASSPATH=$($HADOOP_HOME/bin/hadoop classpath):CLASSPATH
#Java
```

3. 编译打包

```
javac WordCount.java
jar -cf wc.jar WordCount*.class
```

4. 在HDFS上创建目录

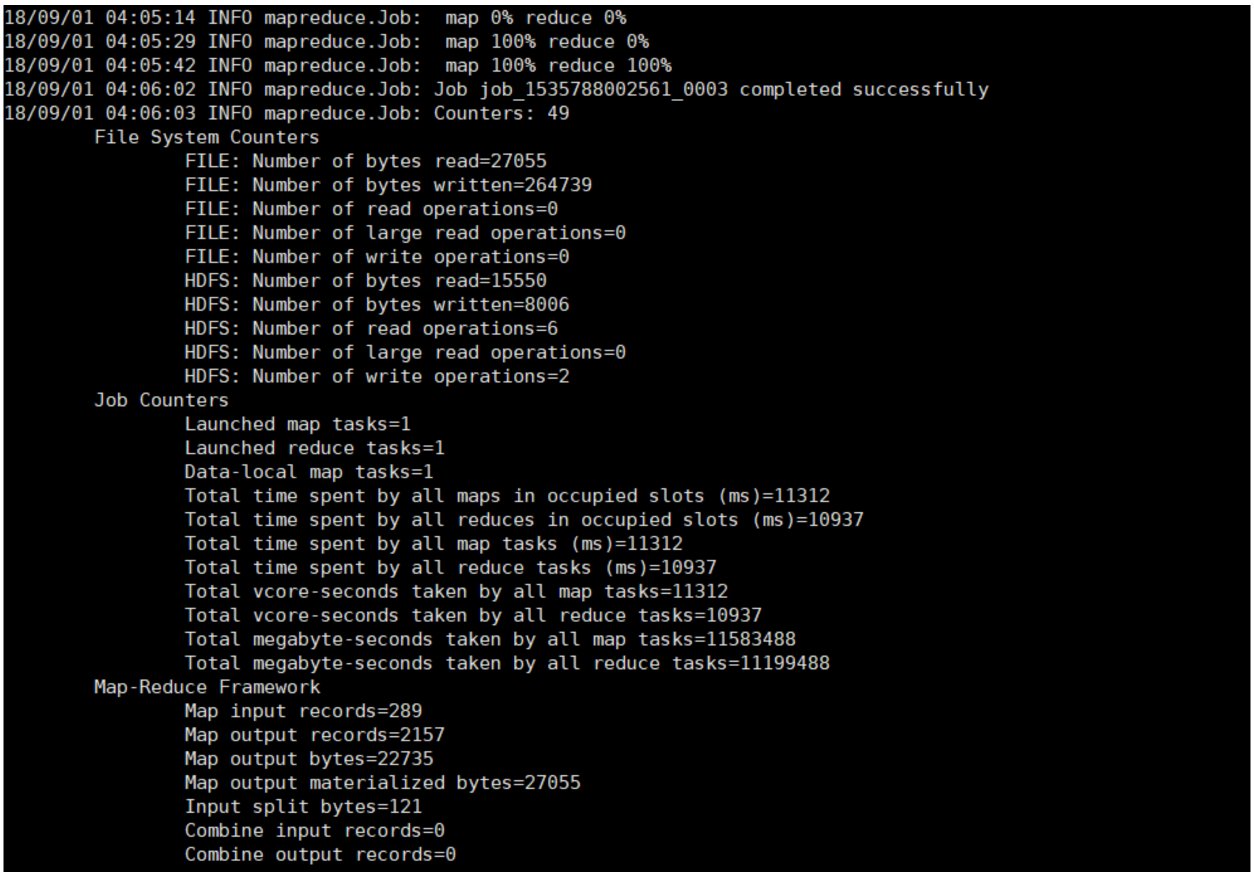
```
hadoop fs -mkdir -p /user/rooot/wordcount/input
```

5. 上传文件到HDFS

```
hadoop fs -copyFromLocal /user/local/hadoop/LICENSE.txt wordcount/input
```

6. 运行WordCount程序

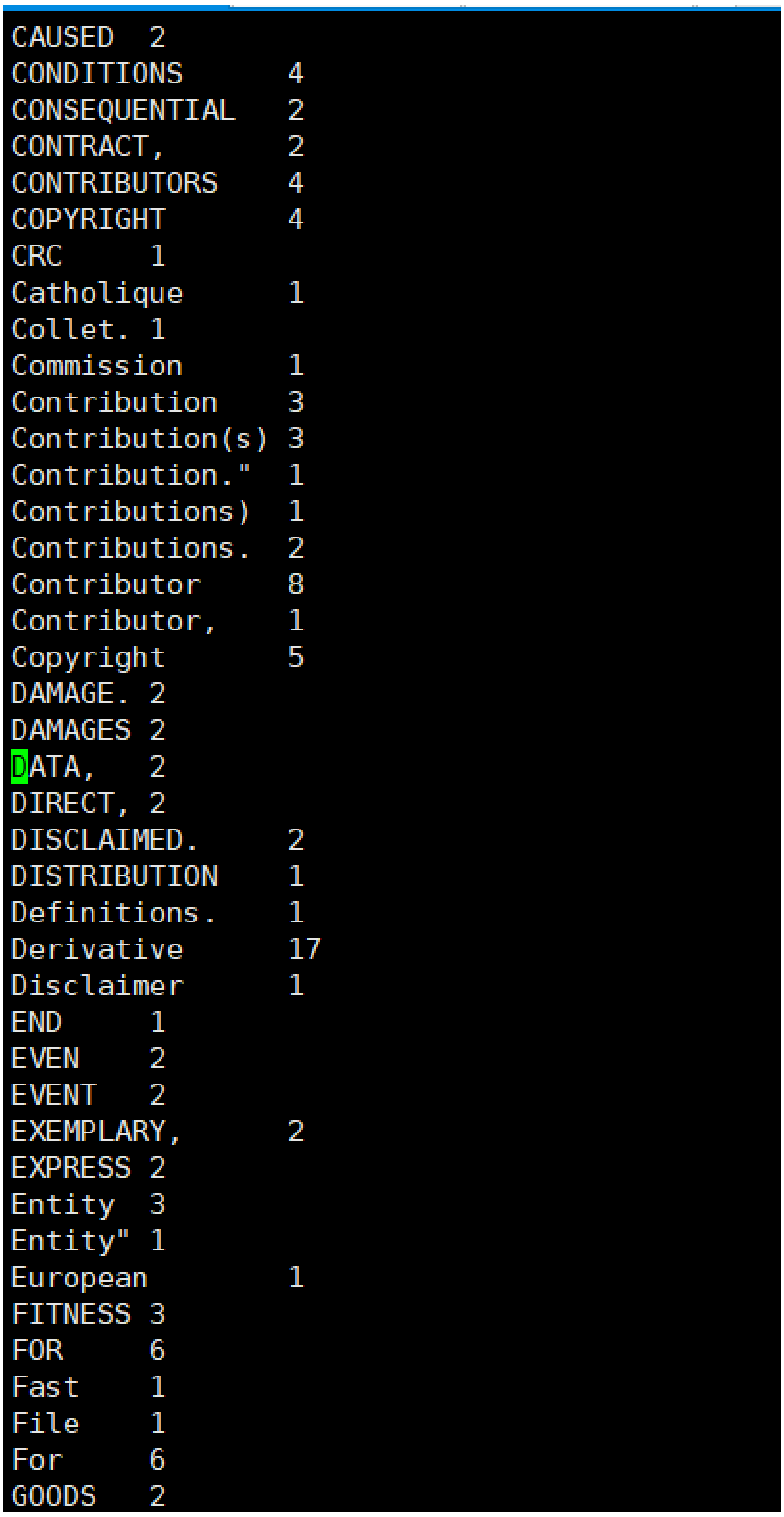
```
hadoop jar wc.jar WordCount /user/root/wordcount/input/LICENSE.txt
/user/root/wordcount/output
```



```
18/09/01 04:05:14 INFO mapreduce.Job:  map 0% reduce 0%
18/09/01 04:05:29 INFO mapreduce.Job:  map 100% reduce 0%
18/09/01 04:05:42 INFO mapreduce.Job:  map 100% reduce 100%
18/09/01 04:06:02 INFO mapreduce.Job: Job job_1535788002561_0003 completed successfully
18/09/01 04:06:03 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=27055
                FILE: Number of bytes written=264739
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=15550
                HDFS: Number of bytes written=8006
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        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)=11312
                Total time spent by all reduces in occupied slots (ms)=10937
                Total time spent by all map tasks (ms)=11312
                Total time spent by all reduce tasks (ms)=10937
                Total vcore-seconds taken by all map tasks=11312
                Total vcore-seconds taken by all reduce tasks=10937
                Total megabyte-seconds taken by all map tasks=11583488
                Total megabyte-seconds taken by all reduce tasks=11199488
        Map-Reduce Framework
                Map input records=289
                Map output records=2157
                Map output bytes=22735
                Map output materialized bytes=27055
                Input split bytes=121
                Combine input records=0
                Combine output records=0
```

7. 查看HDFS中的输出文件内容

```
hadoop fs -cat /user/root/wordcount/output/part-r-0000|more
```



```
CAUSED  2
CONDITIONS      4
CONSEQUENTIAL   2
CONTRACT,       2
CONTRIBUTORS    4
COPYRIGHT       4
CRC     1
Catholique      1
Collet. 1
Commission      1
Contribution    3
Contribution(s) 3
Contribution."  1
Contributions)  1
Contributions.  2
Contributor     8
Contributor,    1
Copyright       5
DAMAGE. 2
DAMAGES 2
DATA,   2
DIRECT, 2
DISCLAIMED.     2
DISTRIBUTION    1
Definitions.    1
Derivative      17
Disclaimer      1
END     1
EVEN    2
EVENT   2
EXEMPLARY,      2
EXPRESS 2
Entity  3
Entity" 1
European        1
FITNESS 3
FOR     6
Fast    1
File    1
For     6
GOODS   2
```