# 实验五 **MapReduce初级编程实践**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **地 点：** | B10楼 | 303房； | **实验台号：** |  |
| **实验日期与时间：** | 2023/4/13 | | **评 分：** |  |
| **预习检查纪录：** |  | | **实验教师：** | 焦青松 |

1. **实验目的**

**（1）.通过实验掌握基本的MapReduce方法。**

**(2).掌握用MapReduce解决一些常用的数据处理问题，包括数据去重，数据排序和数据挖掘等。**

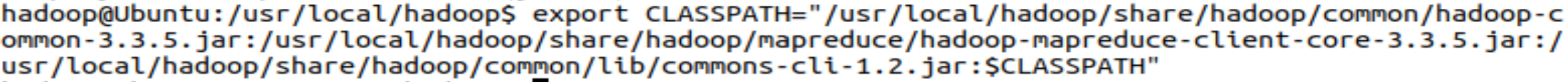
1. **实验平台**

(1).操作系统:Linux

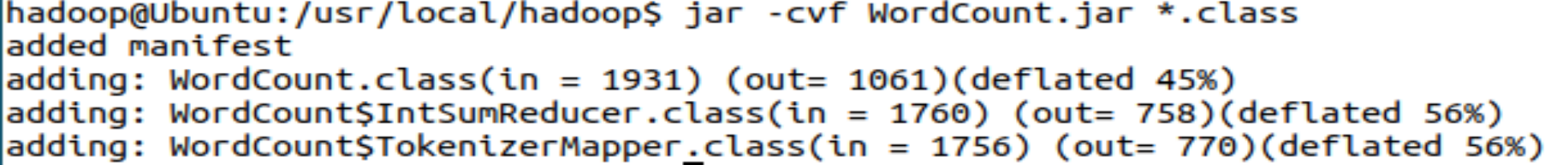
(2).Hadoop版本:3.3.5

1. **实验步骤（包括实验结果/截图）**

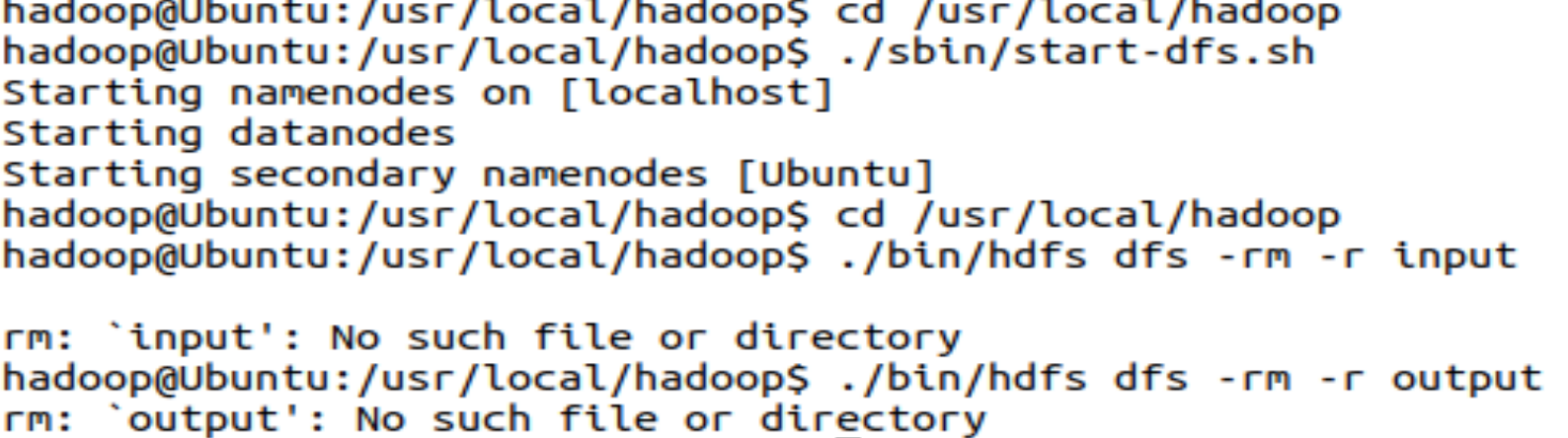
**先建wordCount.java文件。之后进行编译打包。**

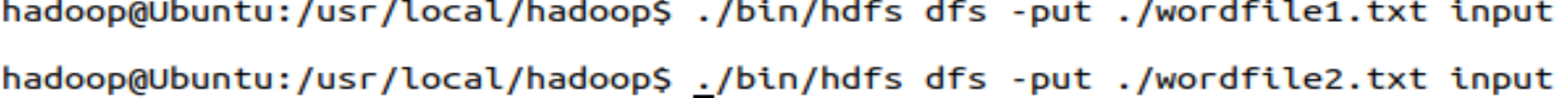
****

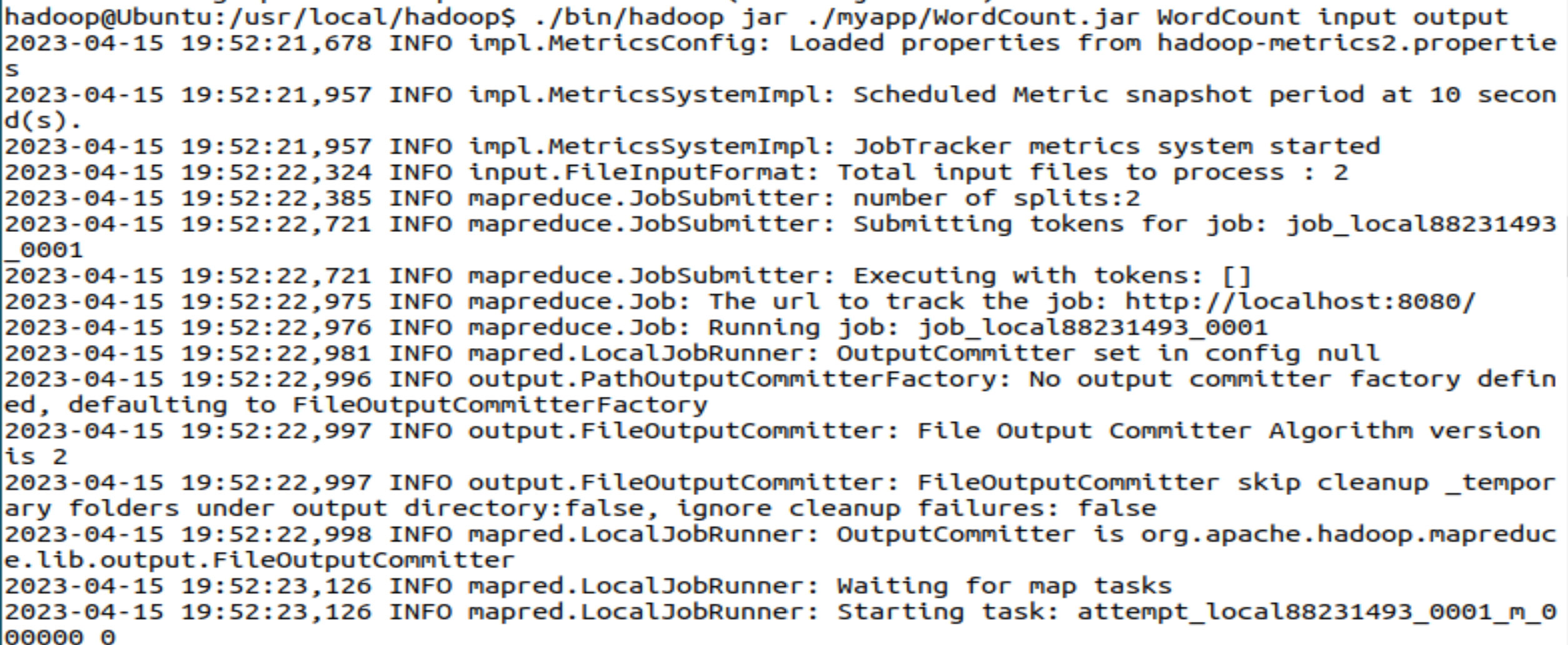
**打包**

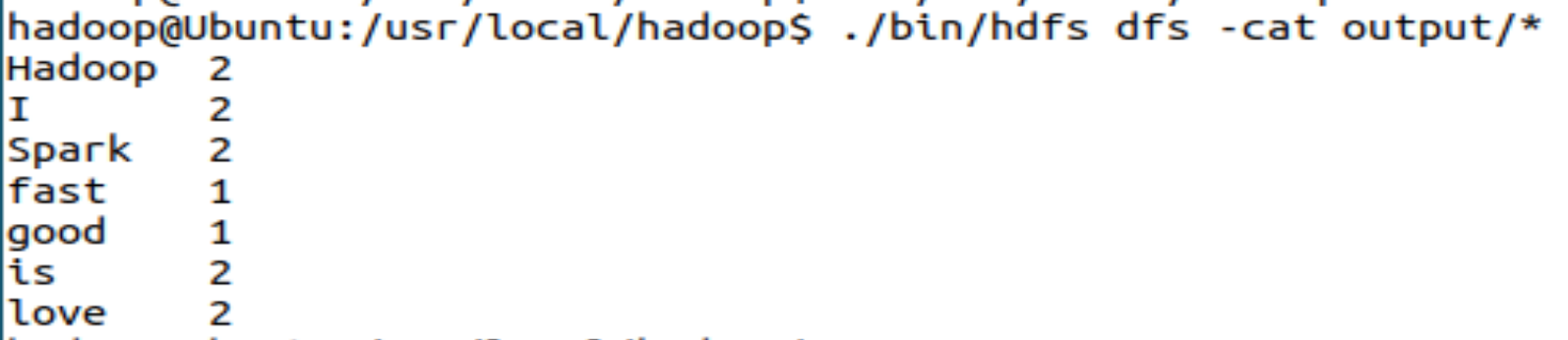
****

**运行程序**

****

****

****

****

**程序完成！**

1. **思考讨论题或体会或对改进实验的建议**

1.MapReduce将输入的数据进行逻辑切片，一片对应一个Map任务

2.Map以并行的方式处理切片

3.框架对Map输出进行排序，然后发给Reduce

4.MapReduce的输入输出数据处于同一个文件系统（HDFS）

5.框架负责任务调度、任务监控、失败任务的重新执行

6.框架会对键和值进行序列化，因此键和值需要实现writable接口，框架会对键排序，因此必须实现writableComparable接口。

1. **实验源代码**

**public static void main(String[] args) throws Exception {**

**Configuration conf = new Configuration();**

**String[] otherArgs = (new GenericOptionsParser(conf, args)).getRemainingArgs();**

**if(otherArgs.length < 2) {**

**System.err.println("Usage: wordcount <in> [<in>...] <out>");**

**System.exit(2);**

**}**

**Job job = Job.getInstance(conf, "word count"); //设置环境参数**

**job.setJarByClass(WordCount.class); //设置整个程序的类名**

**job.setMapperClass(WordCount.TokenizerMapper.class); //添加Mapper类**

**job.setReducerClass(WordCount.IntSumReducer.class); //添加Reducer类**

**job.setOutputKeyClass(Text.class); //设置输出类型**

**job.setOutputValueClass(IntWritable.class); //设置输出类型**

**for(int i = 0; i < otherArgs.length - 1; ++i) {**

**FileInputFormat.addInputPath(job, new Path(otherArgs[i])); //设置输入文件**

**}**

**FileOutputFormat.setOutputPath(job, new Path(otherArgs[otherArgs.length - 1]));//设置输出文件**

**System.exit(job.waitForCompletion(true)?0:1);**

**}**

**import java.io.IOException;**

**import java.util.Iterator;**

**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.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;**

**import org.apache.hadoop.util.GenericOptionsParser;**

**public class WordCount {**

**public WordCount() {**

**}**

**public static void main(String[] args) throws Exception {**

**Configuration conf = new Configuration();**

**String[] otherArgs = (new GenericOptionsParser(conf, args)).getRemainingArgs();**

**if(otherArgs.length < 2) {**

**System.err.println("Usage: wordcount <in> [<in>...] <out>");**

**System.exit(2);**

**}**

**Job job = Job.getInstance(conf, "word count");**

**job.setJarByClass(WordCount.class);**

**job.setMapperClass(WordCount.TokenizerMapper.class);**

**job.setCombinerClass(WordCount.IntSumReducer.class);**

**job.setReducerClass(WordCount.IntSumReducer.class);**

**job.setOutputKeyClass(Text.class);**

**job.setOutputValueClass(IntWritable.class);**

**for(int i = 0; i < otherArgs.length - 1; ++i) {**

**FileInputFormat.addInputPath(job, new Path(otherArgs[i]));**

**}**

**FileOutputFormat.setOutputPath(job, new Path(otherArgs[otherArgs.length - 1]));**

**System.exit(job.waitForCompletion(true)?0:1);**

**}**

**public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable> {**

**private static final IntWritable one = new IntWritable(1);**

**private Text word = new Text();**

**public TokenizerMapper() {**

**}**

**public void map(Object key, Text value, Mapper<Object, Text, Text, IntWritable>.Context context) throws IOException, InterruptedException {**

**StringTokenizer itr = new StringTokenizer(value.toString());**

**while(itr.hasMoreTokens()) {**

**this.word.set(itr.nextToken());**

**context.write(this.word, one);**

**}**

**}**

**}**

**public static class IntSumReducer extends Reducer<Text, IntWritable, Text, IntWritable> {**

**private IntWritable result = new IntWritable();**

**public IntSumReducer() {**

**}**

**public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws IOException, InterruptedException {**

**int sum = 0;**

**IntWritable val;**

**for(Iterator i$ = values.iterator(); i$.hasNext(); sum += val.get()) {**

**val = (IntWritable)i$.next();**

**}**

**this.result.set(sum);**

**context.write(key, this.result);**

**}**

**}**

**}**