Hadoop程序开发

一、实验目的

Hadoop 程序开发

二、实验内容（实验过程、步骤及实验结果）

创建Rectangle.txt文件，输入如下数据

```
9 9
4 5
7 8
1 1
3 6
```

启动hadoop

```
start-all.sh
```

在HDFS上创建目录/input/sort，并将Rectangle.txt文件上传其中。

```
hadoop fs -mkdir /input/sort
hadoop fs -put /data/Rectangle.txt /input/sort
```

建立RectangleSort类

```
package sds.mapreduce;

//导包
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Partitioner;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
```

```
public class RectangleSort {
    //指定输入输出文件位置
    static final String INPUT_PATH =
"hdfs://localhost:9000/input/sort/Rectangle.txt";
    static final String OUTPUT_PATH = "hdfs://localhost:9000/output/rectangle";

    public static void main(String[] args) throws Throwable, URISyntaxException
{
        //创建配置类
        Configuration conf = new Configuration();

        FileSystem fileSystem = FileSystem.get(new URI(INPUT_PATH), conf);
        Path outpath = new Path(OUTPUT_PATH);
        if (fileSystem.exists(outpath)) {
            fileSystem.delete(outpath, true);
        }

        Job job = Job.getInstance(conf, "RectangleSort");
        job.setJarByClass(RectangleWritable.class);
        job.setInputFormatClass(TextInputFormat.class); //指定如何对输入的文件格式
化，把输入文件每一行解析成键值对
        job.setMapperClass(MyMapper.class);            //指定自定义的map类
        job.setMapOutputKeyClass(RectangleWritable.class); //map输出的<k,v>类型，
如果<k3,v3>的类型与<k2,v2>类型一致，则可以省略
        job.setMapOutputValueClass(NullWritable.class);
        job.setReducerClass(MyReducer.class);   //指定自定义reduce类
        job.setOutputKeyClass(IntWritable.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.setInputPaths(job, INPUT_PATH);
        FileOutputFormat.setOutputPath(job, new Path(OUTPUT_PATH));
        job.setOutputFormatClass(TextOutputFormat.class);   //指定输出文件的格式化类
分区
        job.setPartitionerClass(MyPatitioner.class);
        job.setNumReduceTasks(2); //指定两个reduce任务运行
        job.waitForCompletion(true);  //把job提交给jobtracker运行
    }

    static class MyMapper extends
            Mapper<LongWritable, Text, RectangleWritable, NullWritable> {

        protected void map(LongWritable k1, Text v1, Context context)
                throws IOException, InterruptedException {
            String[] splites = v1.toString().split(" ");
            RectangleWritable k2 = new RectangleWritable(
                    Integer.parseInt(splites[0]), Integer.parseInt(splites[1]));
            context.write(k2, NullWritable.get());
        };
    }

    static class MyReducer extends
            Reducer<RectangleWritable, NullWritable, IntWritable, IntWritable> {
        protected void reduce(RectangleWritable k2, Iterable<NullWritable> v2s,
                Context context) throws IOException, InterruptedException {
            context.write(new IntWritable(k2.getLength()),
                    new IntWritable(k2.getWidth()));
        };
    }
}


class MyPatitioner extends Partitioner<RectangleWritable, NullWritable> {
    @Override
    //重写获取分区方法
    public int getPartition(RectangleWritable k2, NullWritable v2,
            int numReduceTasks) {

        if (k2.getLength() == k2.getWidth()) {
            return 0; // 正方形在任务0中汇总
        } else
            return 1;// 长方形在任务1中汇总
    }
}
```

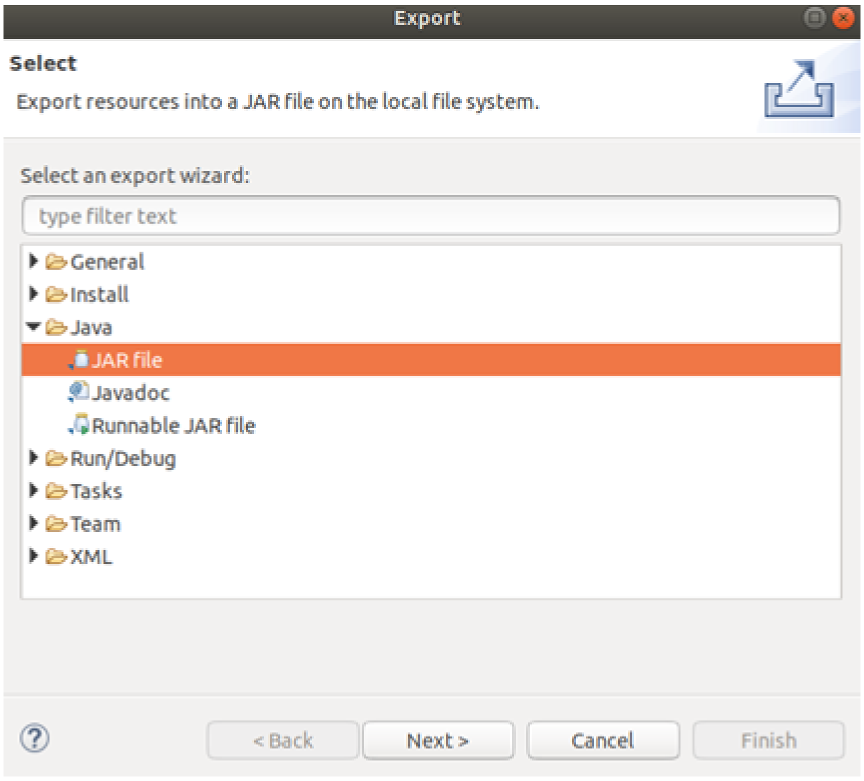
继承WritableComparable类

```
package sds.mapreduce;

import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;

import org.apache.hadoop.io.WritableComparable;

public class RectangleWritable implements WritableComparable <RectangleWritable>
{

    int length, width;

    public int getLength() {
        return length;
    }

    public void setLength(int length) {
        this.length = length;
    }

    public int getWidth() {
        return width;
    }

    public void setWidth(int width) {
        this.width = width;
    }

    public RectangleWritable() {
        super();
    }

    public RectangleWritable(int length, int width) {
        super();
        this.length = length;
        this.width = width;
    }
    @Override
    public void write(DataOutput out) throws IOException {
        out.writeInt(length);
        out.writeInt(width);
    }

    @Override
    public void readFields(DataInput in) throws IOException {
        this.length = in.readInt();
        this.width = in.readInt();

    }

    @Override
    public int compareTo(RectangleWritable o) {
        if (this.getLength() * this.getWidth() > o.getLength() * o.getWidth())
            return 1;
        if (this.getLength() * this.getWidth() < o.getLength() * o.getWidth())
            return -1;
        return 0;
    }

}
```

建立WordCount类

```
package sds.mapreduce;

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

        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();
        public void map(Object key, Text value, Context context) throws
IOException, InterruptedException {
            StringTokenizer itr = new StringTokenizer(value.toString());
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
            }
        }
    }

    public static class IntSumReducer extends Reducer<Text, IntWritable, Text,
IntWritable> {
        private IntWritable result = new IntWritable();
        public void reduce(Text key, Iterable<IntWritable> values, Context
context) throws IOException, InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            result.set(sum);
            context.write(key, result);
        }
    }

    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(TokenizerMapper.class);
        job.setCombinerClass(IntSumReducer.class);
        job.setReducerClass(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);
    }
}
```

在自己家目录下创建~/project_jar目录用于保存导出的jar包



Export

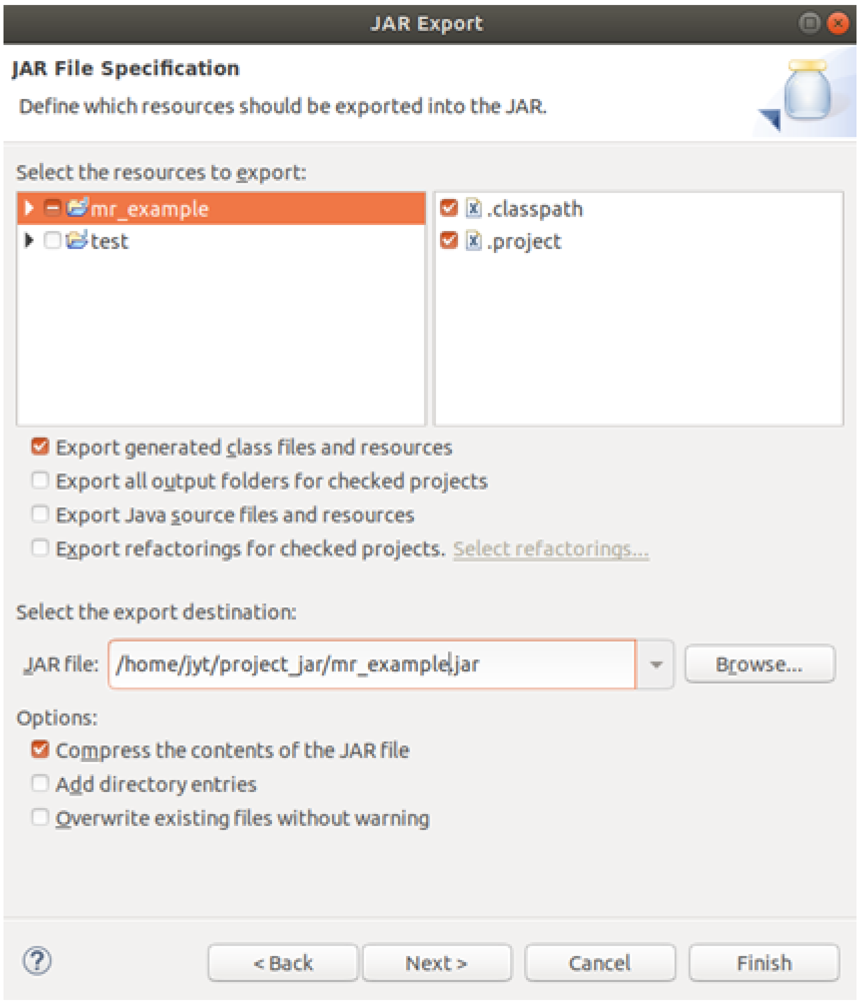
Select

Export resources into a JAR file on the local file system.

Select an export wizard:

type filter text

- General
- Install
- Java
  - JAR file
  - Javadoc
  - Runnable JAR file
- Run/Debug
- Tasks
- Team
- XML



JAR Export

JAR File Specification

Define which resources should be exported into the JAR.

Select the resources to export:

- mr_example
- test

[✓] .classpath
[✓] .project

[✓] Export generated class files and resources
[ ] Export all output folders for checked projects
[ ] Export Java source files and resources
[ ] Export refactorings for checked projects. Select refactorings...

Select the export destination:

JAR file: /home/jyt/project_jar/mr_example.jar

Options:

[✓] Compress the contents of the JAR file
[ ] Add directory entries
[ ] Overwrite existing files without warning

运行jar包中的java类

```
hadoop jar ~/project_jar/mr_example.jar sds.mapreduce.WordCount
/input/wordcount/testfile /output/wordcount

## 运行结果 hadoop fs -ls /output/wordcount/part-r-00000
big     1
```

```
data   1
hadoop 1
hdfs   1
hello  3


hadoop jar ~/project_jar/mr_example.jar sds.mapreduce.RectangleSort
//配置参数已经写入java类中,无需在命令行中再说明，输出文件也在java类中做了处理。

## 运行结果 hadoop fs -ls /output/rectangle/part-r-00000
1 1
9 9

## 运行结果 hadoop fs -ls /output/rectangle/part-r-00001
3 6
4 5
7 8
```