编写 MapReduce 最高分统计

要求：将成绩单中所有成绩统计出单科最高分

图 1 成绩单截图

1. 将$HADOOP_HOME/share/hadoop/mapreduce 中的 hadoop-mapreduce-examples-2.7.7-sources.jar 下载到本地，使用 IDE 打开。

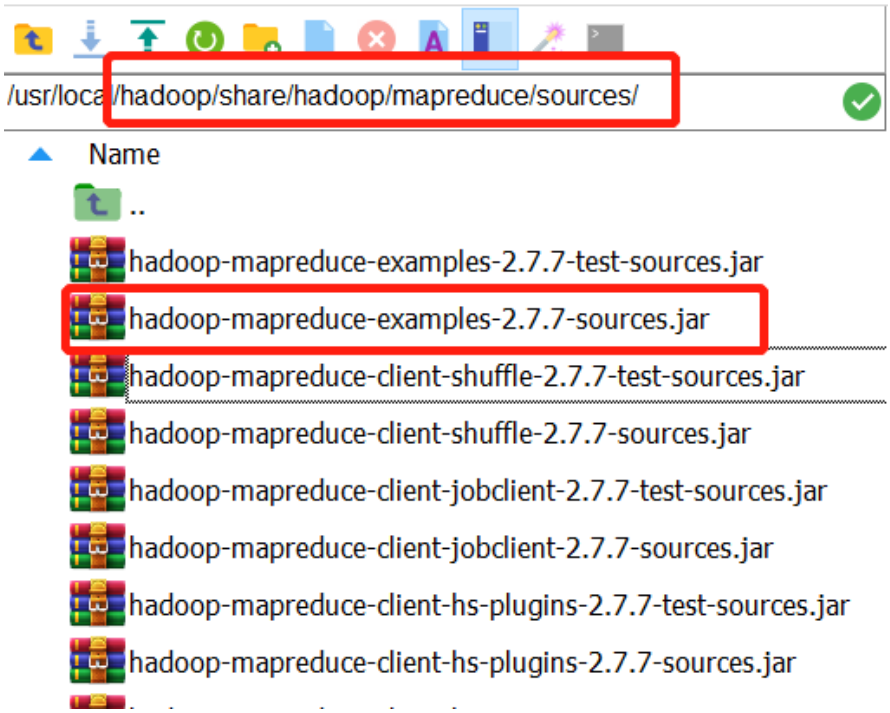

图 2 找到相关文件

2. 下载安装 hadoop 用的 hadoop-2.7.7.tar.gz 压缩包下载到本地。

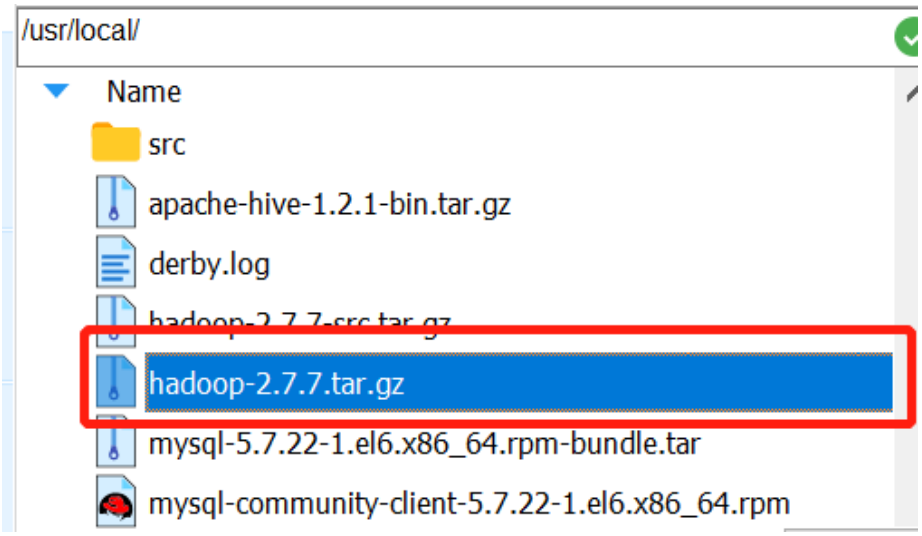

图 3 找到 hadoop 安装包

3. 在 Intellij 中打开步骤 1 下载的项目后，从步骤 2 所下载的 hadoop-2.7.7.tar.gz 导入该项目所需依赖(只需要将 hdfs，common，mapreduce 相应目录及其目录下的 lib 文件夹导入，**common\lib** 不导入，导入的话后续在虚拟机运行 jar 包会出现‘SLF4J: Class path contains multiple SLF4J bindings.’的错误):

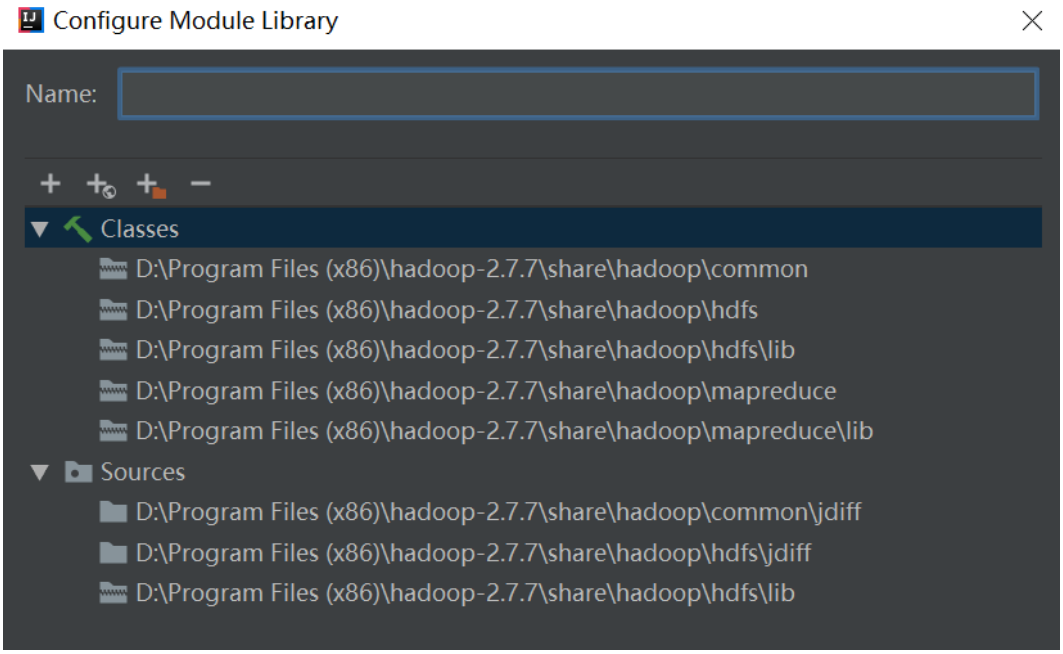

图 4 添加相应依赖

4. 阅读 wordcount.java 文件并理解其实现方式，其中 TokenizerMapper 和 IntSumReducer 实现了 mapreduce 的 map 和 reduce 功能，因此在编写自己的代码的时候，只需要根据需要修改这两个类。

```java
public static class TokenizerMapper
    extends Mapper<Object, Text, Text, IntWritable>{
  private final static IntWritable one = new IntWritable( value: 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);
    }
  }
}
```

图 5 TokenizerMapper 类

```java
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);
  }
}
```

图 6 IntSumReducer 类

5. 编写自己的类，并命名为 GetMax.java:

```
public static class TokenizerMapper
    extends Mapper<Object, Text, Text, IntWritable>{
  private Text word = new Text();
  public void map(Object key, Text value, Context context
                  ) throws IOException, InterruptedException {
    StringTokenizer itr = new StringTokenizer(value.toString(), delim: "\n");
    while (itr.hasMoreTokens()) {
      word.set(itr.nextToken());
      String[] aline = word.toString().split( regex: " ");
      Text subject = new Text();
      subject.set(aline[0]);
      int x;
      try {
          x = Integer.parseInt(aline[1]);
      }
      catch (Exception e) {
          continue;
      }
      IntWritable score = new IntWritable(x);
      context.write(subject, score); //value as key
    }
  }
}
```

图 7 编写自己的 TokenizerMapper 类

```java
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 maxScore = -99;
    for (IntWritable val : values) {
        maxScore = Math.max(maxScore,val.get());
    }
    result.set(maxScore);
    context.write(key, result);
  }
}
```

图 8 编写自己的 IntSumReducer 类

```java
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( status: 2);
  }
  Job job = Job.getInstance(conf,  jobName: "getmax");
  job.setJarByClass(GetMax.class);
  job.setMapperClass(TokenizerMapper.class);
  job.setCombinerClass(IntSumReducer.class);
  job.setReducerClass(IntSumReducer.class);
  job.setOutputKeyClass(Text.class);
```

图 9 修改 main 函数里相应 diamagnetic

6. 在 ExampleDriver.java 中添加相应代码：

```java
public static void main(String argv[]){
  int exitCode = -1;
  ProgramDriver pgd = new ProgramDriver();
  try {
    pgd.addClass( name: "getmax", GetMax.class,
          description: "A map/reduce program that gets the max score in the input files.");
    pgd.addClass( name: "wordmedian", WordMedian.class,
              description: "A map/reduce program that counts the median length of the words in the input files.");
```

图 10 在 ExampleDriver.java 中添加相应代码

7. 函数编写完后，创建 jar：

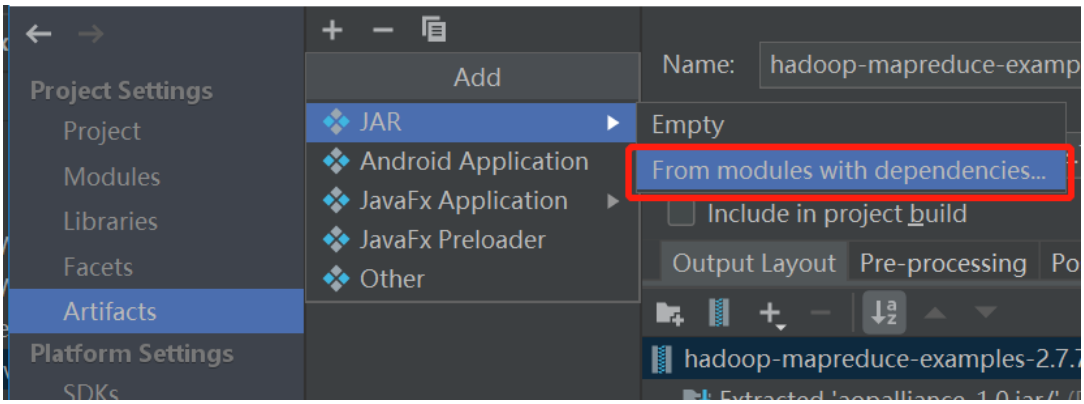

图 11 创建 jar

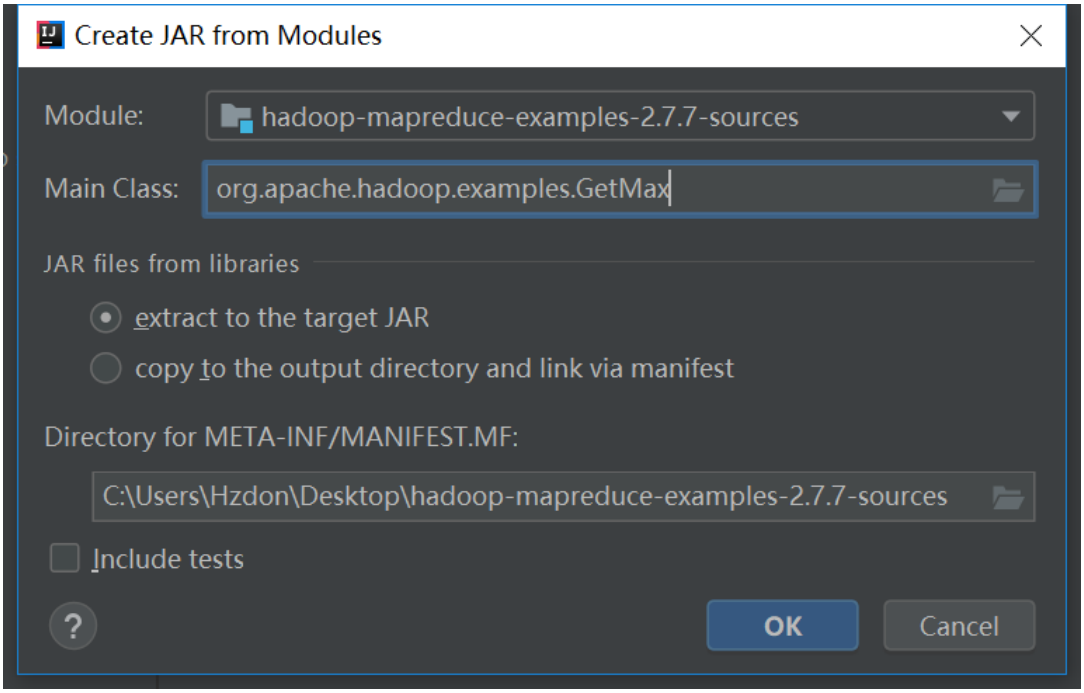

图 12 将 Main Class 参数设置为自己写的类

8. Build artifacts，生成所需 jar 文件：

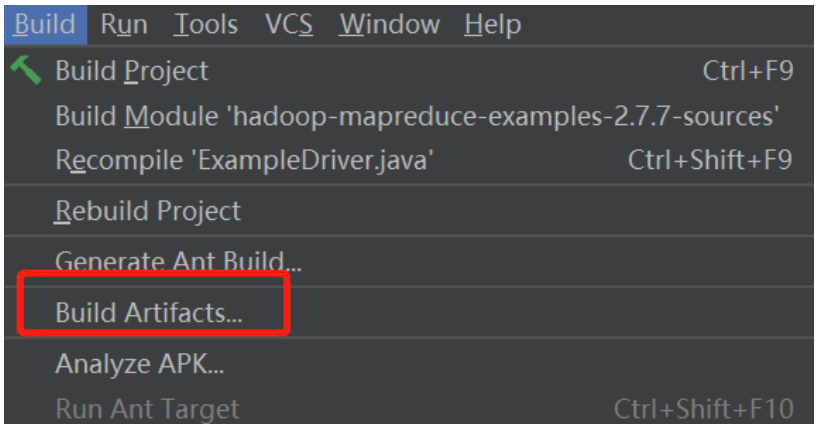

图 13 Build artifacts

9. 将生成的 jar 包上传到虚拟机，并保存到 HADOOP_HOME/share/hadoop/mapreduce/目

录下。

10. 将 subject_scores.txt 上传到虚拟机，并上传到 hdfs 的/user/root/input/目录下。

11. 输入如下指令，运行 jar 包并输出到/user/root/output16 目录下

```
[root@BigData201916071072 local]# ./hadoop/bin/hadoop jar ./hadoop/share/hadoop/mapreduce/hadoop-mapreduce-exam
ples-2.7.7-getMax.jar /user/root/input/subject_score.txt /user/root/output16
19/06/16 11:34:37 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
19/06/16 11:34:38 INFO input.FileInputFormat: Total input paths to process : 1
19/06/16 11:34:38 INFO mapreduce.JobSubmitter: number of splits:1
19/06/16 11:34:39 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1560605406223_0020
19/06/16 11:34:39 INFO impl.YarnClientImpl: Submitted application application_1560605406223_0020
19/06/16 11:34:39 INFO mapreduce.Job: The url to track the job: http://BigData201916071072.lab.beihangsoft.cn:8
088/proxy/application_1560605406223_0020/
19/06/16 11:34:39 INFO mapreduce.Job: Running job: job_1560605406223_0020
19/06/16 11:34:45 INFO mapreduce.Job: Job job_1560605406223_0020 running in uber mode : false
19/06/16 11:34:45 INFO mapreduce.Job:  map 0% reduce 0%
19/06/16 11:34:50 INFO mapreduce.Job:  map 100% reduce 0%
19/06/16 11:34:56 INFO mapreduce.Job:  map 100% reduce 100%
19/06/16 11:34:57 INFO mapreduce.Job: Job job_1560605406223_0020 completed successfully
19/06/16 11:34:57 INFO mapreduce.Job: Counters: 49
```

图 14 运行 jar

```
                Combine input records=60000
                Combine output records=6
                Reduce input groups=6
                Reduce shuffle bytes=84
                Reduce input records=6
                Reduce output records=6
                Spilled Records=12
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=134
                CPU time spent (ms)=1740
                Physical memory (bytes) snapshot=440406016
                Virtual memory (bytes) snapshot=4265254912
                Total committed heap usage (bytes)=293076992
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=662604
        File Output Format Counters
                Bytes Written=63
```

图 15 运行成功后的输出结果

12. 查看 hdfs 上/user/root/output16 目录：

```
[root@BigData201916071072 hadoop]# ./bin/hdfs dfs -ls /user/root/output16
Found 2 items
-rw-r--r--   1 root supergroup          0 2019-06-16 11:34 /user/root/output16/_SUCCESS
-rw-r--r--   1 root supergroup         63 2019-06-16 11:34 /user/root/output16/part-r-00000
```

图 16 到输出目录下查看输出的文件

13. 输出 part-r-00000 查看结果:

```
[root@BigData201916071072 hadoop]# ./bin/hdfs dfs -cat /user/root/output16/part-r-00000
化学    99
数学    149
物理    99
生物    99
英语    144
语文    114
```

图 17 查看输出结果

遇到的问题及解决方案：

1. 在运行 jar 文件时报错：“SLF4J: Class path contains multiple SLF4J bindings.”；
   原因：导入依赖的时候没有进行筛选，把所有的依赖都导入，导致有的 binding 项出现重复；
   解决方案：在报错的下面有提示哪些项重复，然后在 IDE 中把步骤三导入的过多依赖删除。

回答实验要求中的问题（若无则无需填写）：

1. 谈谈你对 MapReduce 模型的理解。

   MapReduce 是一个基于集群的计算平台，是一个简化分布式编程的计算框架，是一个将分布式计算抽象为 Map 和 Reduce 两个阶段的编程模型。在执行时，先执行 map()函数将有相同 key 值的键值对放在一起，然后在 reduce()函数中对相同 key 的键值对进行处理。在 WordCount 中，是把每个单词作为 key，1 作为指代表出现 1 次，然后在 reduce()阶段将相同 key 值的所有 value 相加，即达到计算单词出现次数的目的；在这次实验中，map()中，每个键值对的 key 为分数，在 reduce()中，value 值为相同 key 下 value 的最大值，即每一个学科的最高分。

2. MapReduce 模型还能处理哪些常见任务？

还能处理类似 sql 语句中的 join，group by 等功能，使用场景如日志分析，海量数据排序处理等。