# 大数据综合处理实验Lab3

文档倒排索引

161220017 陈翔

1. **实验目的**

**…**

1. **实验环境**
   * 操作系统版本：Ubuntu 18.04
   * Java版本：1.8.0\_201
   * Hadoop版本：2.7.1
2. **代码框架**
   * Mapper
3. **public** **static** **class** Map **extends** Mapper<Object, Text, Text, Text> {
5. **private** Text word = **new** Text();
7. @Override
8. **public** **void** map(Object key, Text value, Mapper<Object, Text, Text, Text>.Context context) **throws** IOException, InterruptedException {
9. FileSplit fileSplit = (FileSplit)context.getInputSplit();
10. String fileName = fileSplit.getPath().getName();
11. Text fileName\_lineOffset = **new** Text(fileName+"#"+key.toString());
12. StringTokenizer itr = **new** StringTokenizer(value.toString());
13. **for**(;itr.hasMoreTokens();)
14. {
15. word.set(itr.nextToken());
16. context.write(word, fileName\_lineOffset);
17. }
18. }
19. }

注意，这里我们把key的类换成了Object（PPT上是Text），否则会无法重载。

* + Reducer

1. **public** **static** **class** Reduce **extends** Reducer <Text, Text, Text, Text>  {
2. @Override
3. **public** **void** reduce(Text key, Iterable<Text> values, Reducer <Text, Text, Text, Text>.Context context) **throws** IOException, InterruptedException {
4. Iterator<Text> it = values.iterator();
5. StringBuilder all = **new** StringBuilder();
6. **if**(it.hasNext())
7. all.append(it.next().toString());
8. **for**(;it.hasNext();)
9. {
10. all.append(";");
11. all.append(it.next().toString());
12. }
13. context.write(key, **new** Text(all.toString()));
14. }
15. }
    * main
16. **public** **static** **void** main(String[] args) **throws** Exception {
17. Configuration conf = **new** Configuration();
18. Job job = Job.getInstance(conf, "invert index");
19. job.setJarByClass(InvertedIndexer.**class**);
21. //map
22. job.setMapperClass(Map.**class**);
23. job.setMapOutputKeyClass(Text.**class**);
24. job.setMapOutputValueClass(Text.**class**);
25. //reduce
26. job.setReducerClass(Reduce.**class**);
28. job.setOutputKeyClass(Text.**class**);
29. job.setOutputValueClass(Text.**class**);
31. FileInputFormat.addInputPath(job, **new** Path(args[0]));
32. FileOutputFormat.setOutputPath(job, **new** Path(args[1]));
34. System.exit(job.waitForCompletion(**true**) ? 0 : 1);
35. }

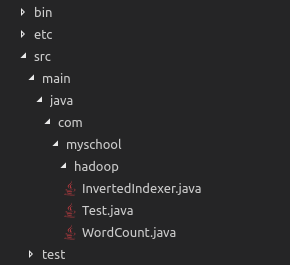
注意，这里我们特意将MapOutputKeyClass和MapOutputValueClass设置为了Text，否则运行过程中会报类型不匹配的错误。

1. **实验过程**
   * Maven的下载与使用

首先从Maven的官网下载最新的maven压缩包，版本号为3.6.1，解压到特定文件夹。这里最好更换掉maven的默认镜像，否则下载插件会很慢。接着用指令

mvn archetype:generate -DgroupId=com.myschool.hadoop -DartifactId= hadoop -Dpackage=com.myschool.hadoop -Dversion=1.0-SNAPSHOT

生成我们的java框架，如下



注意，其中的所有java文件都是之后手动添加的。

接着编辑其中的InvertedIndexer文件，上文中已经给出框架，编辑完成后运行如下指令

mvn package

我们就可以得到一个target文件夹，里面有生成的jar文件，同时还有所有class文件，后面我们会用到其中的InvertedIndexer类。

最后我们在本地使用规模较小的样例测试，其中每个文档都完全相同，只有单词JOB\_NEW和JOB\_FINISH。运行的具体指令与实验2类似，下面为结果截图

