《大数据技术原理与应用》

实验2 MapReduce编程初级实践

姓名

班级

日期 2021.5.13

实验2 MapReduce编程初级实践

1. **实验目的**

1.通过实验掌握基本的MapReduce编程方法；

2.掌握用MapReduce解决一些常见的数据处理问题，包括数据去重、数据排序和数据挖掘等。

1. **实验平台**

已经配置完成的Hadoop伪分布式环境。

1. **实验内容和要求**

## 1.编程实现文件合并和去重操作

对于两个输入文件，即文件A和文件B，请编写MapReduce程序，对两个文件进行合并，并剔除其中重复的内容，得到一个新的输出文件C。下面是输入文件和输出文件的一个样例供参考。

输入文件A的样例如下：

|  |
| --- |
| 20150101 x  20150102 y  20150103 x  20150104 y  20150105 z  20150106 x |

输入文件B的样例如下：

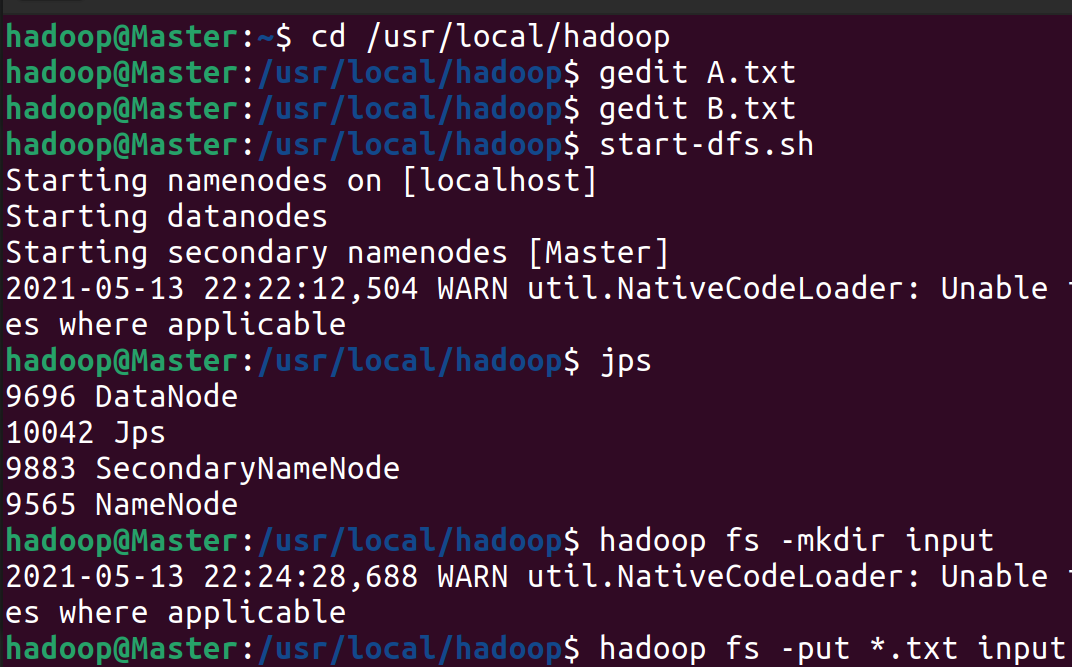
|  |
| --- |
| 20150101 y  20150102 y  20150103 x  20150104 z  20150105 y |

根据输入文件A和B合并得到的输出文件C的样例如下：

|  |
| --- |
| 20150101 x  20150101 y  20150102 y  20150103 x  20150104 y  20150104 z  20150105 y  20150105 z  20150106 x |

实验过程：

1. 创建文件A.txt和B.txt
2. 先开启hadoop相关进程，再在HDFS建立input文件夹
3. 上传样例到HDFS中的input文件夹



1. 编写Java代码：

**package** mapreduce;

**import** java.io.IOException;

**import** org.apache.hadoop.conf.Configuration;

**import** org.apache.hadoop.fs.Path;

**import** org.apache.hadoop.io.\*;

**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;

**public** **class** Merge {

/\*\*

\* 合并 A B 2个文件，剔除重复内容，得到新的输出文件C

\* 重载map函数，直接将输入中的value复制到输出数据的key上

\* **@author** hadoop

\*

\*/

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

**private** **static** Text *text*=**new** Text();

**public** **void** map(Object key, Text value, Context context) **throws** IOException,InterruptedException{

*text*=value;

context.write(*text*, **new** Text(""));

}

}

//重载reduce函数，直接将输入中的key复制到输出数据的key上

**public** **static** **class** Reduce **extends** Reducer<Text, Text, Text, Text> {

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

context.write(key, **new** Text(""));

}

}

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

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

conf.set("fs.default.name","hdfs://localhost:9000");

String[] otherArgs=**new** String[]{"input","output"};//直接设置输入参数

**if**(otherArgs.length!=2) {

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

System.*exit*(2);

}

Job job=Job.*getInstance*(conf,"Merge and duplicate removal");

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

job.setMapperClass(Map.**class**);

job.setCombinerClass(Reduce.**class**);

job.setReducerClass(Reduce.**class**);

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

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

FileInputFormat.*addInputPath*(job, **new** Path(otherArgs[0]));

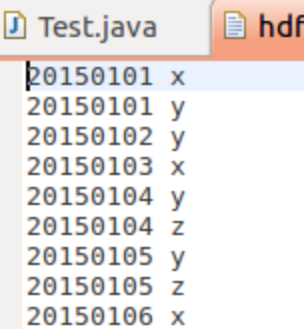
FileOutputFormat.*setOutputPath*(job, **new** Path(otherArgs[1]));

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

}

}

1. 查看代码执行结果:



## 2. 编写程序实现对输入文件的排序

现在有多个输入文件，每个文件中的每行内容均为一个整数。要求读取所有文件中的整数，进行升序排序后，输出到一个新的文件中，输出的数据格式为每行两个整数，第一个数字为第二个整数的排序位次，第二个整数为原待排列的整数。下面是输入文件和输出文件的一个样例供参考。

输入文件1的样例如下：

|  |
| --- |
| 33  37  12  40 |

输入文件2的样例如下：

|  |
| --- |
| 4  16  39  5 |

输入文件3的样例如下：

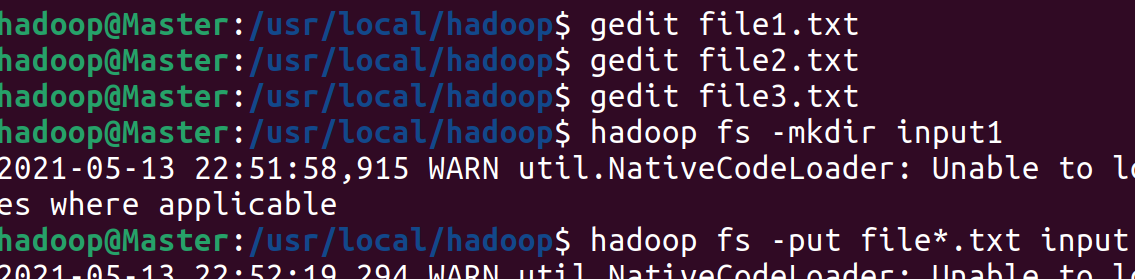
|  |
| --- |
| 1  45  25 |

根据输入文件1、2和3得到的输出文件如下：

|  |
| --- |
| 1 1  2 4  3 5  4 12  5 16  6 25  7 33  8 37  9 39  10 40  11 45 |

实验过程：

1. 创建文件file1.txt、file2.txt和file3.txt
2. 在HDFS建立input1文件夹
3. 上传样例到HDFS中的input1文件夹





1. 编写Java代码：

**package** mapreduce;

**import** java.io.IOException;

**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;

**public** **class** ContentSort {

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

**private** **static** IntWritable *data* = **new** IntWritable();

@Override

**public** **void** map(Object key, Text value, Context context) **throws** IOException, InterruptedException {

String line = value.toString();

*data*.set(Integer.*parseInt*(line));

context.write(*data*, **new** IntWritable(1));

}

}

**public** **static** **class** Reduce **extends** Reducer<IntWritable, IntWritable, IntWritable, IntWritable> {

**private** **static** IntWritable *linenum* = **new** IntWritable(1);

@Override

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

**for** (IntWritable val : values) {

context.write(*linenum*, key);

*linenum* = **new** IntWritable(*linenum*.get() + 1);

}

}

}

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

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

conf.set("fs.defaultFS", "hdfs://localhost:9000");

String[] otherArgs = **new** String[]{"input2", "output2"};

**if** (otherArgs.length != 2) {

System.***err***.println("Usage: ContentSort <in> <out>");

System.*exit*(2);

}

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

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

job.setMapperClass(Map.**class**);

job.setReducerClass(Reduce.**class**);

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

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

FileInputFormat.*addInputPath*(job, **new** Path(otherArgs[0]));

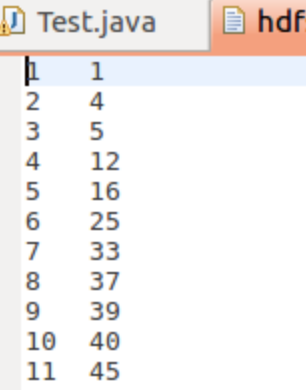
FileOutputFormat.*setOutputPath*(job, **new** Path(otherArgs[1]));

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

}

}

1. 查看代码执行结果:



## 3. 对给定的表格进行信息挖掘

下面给出一个child-parent的表格，要求挖掘其中的父子辈关系，给出祖孙辈关系的表格。

输入文件内容如下：

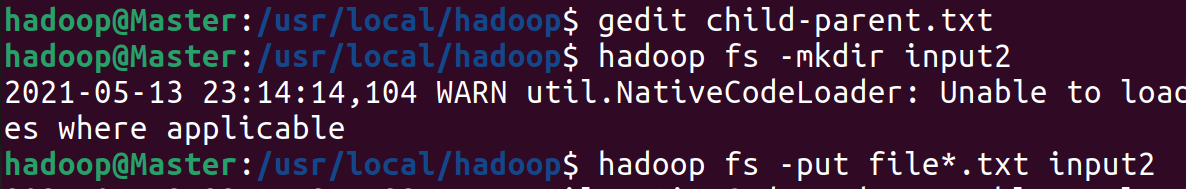
|  |
| --- |
| child parent  Steven Lucy  Steven Jack  Jone Lucy  Jone Jack  Lucy Mary  Lucy Frank  Jack Alice  Jack Jesse  David Alice  David Jesse  Philip David  Philip Alma  Mark David  Mark Alma |

输出文件内容如下：

|  |
| --- |
| grandchild grandparent  Steven Alice  Steven Jesse  Jone Alice  Jone Jesse  Steven Mary  Steven Frank  Jone Mary  Jone Frank  Philip Alice  Philip Jesse  Mark Alice  Mark Jesse |

实验过程：

1. 创建文件child-parent.txt
2. 在HDFS建立input2文件夹
3. 上传样例到HDFS中的input2文件夹



1. 编写Java代码：

**package** mapreduce;

**import** java.io.IOException;

**import** java.util.ArrayList;

**import** java.util.List;

**import** org.apache.hadoop.conf.Configuration;

**import** org.apache.hadoop.fs.Path;

**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;

**public** **class** STJoin {

**public** **static** **int** *time* = 0;

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

@Override

**public** **void** map(Object key, Text value, Context context) **throws** IOException, InterruptedException {

String line = value.toString();

String[] childAndParent = line.split(" ");

List<String> list = **new** ArrayList<>(2);

**for** (String childOrParent : childAndParent) {

**if** (!"".equals(childOrParent)) {

list.add(childOrParent);

}

}

**if** (!"child".equals(list.get(0))) {

String childName = list.get(0);

String parentName = list.get(1);

String relationType = "1";

context.write(**new** Text(parentName), **new** Text(relationType + "+"

+ childName + "+" + parentName));

relationType = "2";

context.write(**new** Text(childName), **new** Text(relationType + "+"

+ childName + "+" + parentName));

}

}

}

**public** **static** **class** Reduce **extends** Reducer<Text, Text, Text, Text>{

@Override

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

**if** (*time* == 0) {

context.write(**new** Text("grand\_child"), **new** Text("grand\_parent"));

*time*++;

}

List<String> grandChild = **new** ArrayList<>();

List<String> grandParent = **new** ArrayList<>();

**for** (Text text : values) {

String s = text.toString();

String[] relation = s.split("\\+");

String relationType = relation[0];

String childName = relation[1];

String parentName = relation[2];

**if** ("1".equals(relationType)) {

grandChild.add(childName);

} **else** {

grandParent.add(parentName);

}

}

**int** grandParentNum = grandParent.size();

**int** grandChildNum = grandChild.size();

**if** (grandParentNum != 0 && grandChildNum != 0) {

**for** (**int** m = 0; m < grandChildNum; m++) {

**for** (**int** n = 0; n < grandParentNum; n++) {

context.write(**new** Text(grandChild.get(m)), **new** Text(

grandParent.get(n)));

}

}

}

}

}

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

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

conf.set("fs.defaultFS", "hdfs://localhost:9000");

String[] otherArgs = **new** String[]{"input3", "output3"};

**if** (otherArgs.length != 2) {

System.***err***.println("Usage: Single Table Join <in> <out>");

System.*exit*(2);

}

Job job = Job.*getInstance*(conf, "Single table Join ");

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

job.setMapperClass(Map.**class**);

job.setReducerClass(Reduce.**class**);

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

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

FileInputFormat.*addInputPath*(job, **new** Path(otherArgs[0]));

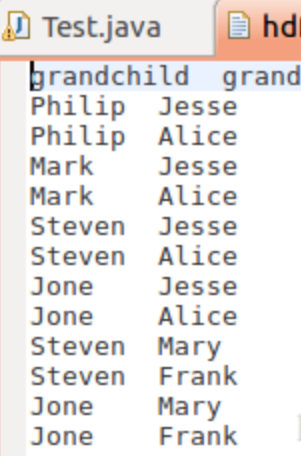
FileOutputFormat.*setOutputPath*(job, **new** Path(otherArgs[1]));

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

}

}

1. 查看代码执行结果:



1. **出现的问题及解决方案**

1.

问题：

第一个实验“数据去重”在检验代码无错的情况下，输出的结果却把两个文件的所有数据给输出来了，数据重合的数据并没有合并。

解决方法：

经过认真分析程序逻辑和输出结果，发现问题出现在不同文件中一行数据之间的空格不同，由于“数据去重”的合并过程是相同字符串的就合在一起，所以在不同文件中一行数据之间的空格不同的情况下，shuffle会把这两个本应合并的字符串认为是两个字符串，故出现了错误的结果。

2.

出错代码如下：

org.apache.hadoop.mapreduce.ReduceContext$ValueIterator cannot be cast to org.apache.commons.math.util.OpenIntToDoubleHashMap$Iterator

解决方法：

首先分析到出问题的代码是这两段：Iterator ite=(Iterator) values.iterator();String record=((java.util.Iterator) ite).next().toString();可以看出，第一段代码强制类型转换得到是org.apache.commons.math.util.OpenIntToDoubleHashMap$Iterator而下面要用到的是java.util.Iterator，所以修改第一段代码为java.util.Iterator ite=values.iterator();问题解决。