

实验名称

图书馆热门书籍top10

实验目的

熟练掌握MapReduce的编程

求图书馆热门书籍top10

实验背景

我们都知道MapReduce的局限性是map读一行，执行一次，reduce是每一组执行一次，但是当我们想全部得到数据之后，按照需求删选然后再输出怎么办呢？这时候只使用map和reduce显然是达不到目的的，此时我们想到了setup和cleanup的特性，只执行一次。这样我们对于最终数据的过滤，然后输出要放在cleanup中。这样就能实现对数据，不一组一组输出，而是全部拿到，最后过滤输出。经典运用常见，mapreduce分析数据然后再求数据的topN 问题。

实验原理

hadoop中的MapReduce框架里已经预定义了相关的接口，其中如Mapper类下的方法setup()和cleanup()

- setup(), 此方法被MapReduce框架仅且执行一次，在执行Map任务前，进行相关变量或者资源的集中初始化工作。若是将资源初始化工作放在方法map()中，导致Mapper任务在解析每一行输入时都会进行资源初始化工作，导致重复，程序运行效率不高！
- cleanup(),此方法被MapReduce框架仅且执行一次，在执行完毕Map任务后，进行相关变量或资源的释放工作。若是将释放资源工作放入方法map()中，也会导致Mapper任务在解析、处理每一行文本后释放资源，而且在下一行文本解析前还要重复初始化，导致反复重复，程序运行效率不高！

实验环境

ubuntu 22.10

hadoop 3.1.3

jdk 1.8

建议课时

4课时

实验步骤

一、环境准备

本实验在eclipse进行开发。

首先启动Hadoop环境：

```
start-all.sh
```

当看下以下进程，则Hadoop启动成功

```
jps
```

```
ubuntu@079b2e0d54d1:~$ jps
978 ResourceManager
1076 NodeManager
1370 Jps
827 SecondaryNameNode
667 DataNode
543 NameNode
```

二、数据准备

打开命令行窗口，下载文件books.txt

```
cd ~
mkdir book_data
cd book_data
wget http://i9000.net:8888/sgn/BDHTech/08TrainingWeek/MR_student/books.txt
```

上传至HDFS /{学号}/shiyans5/input，此文档{学号}为001

```
hdfs dfs -mkdir -p /001/shiyans5/input
hdfs dfs -put books.txt /001/shiyans5/input
```

三、代码编写

1、创建LibraryMapper类并继承 Mapper类，步骤请参考前面的实验，然后重写map()

完整代码如下：

```
import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class LibraryMapper extends Mapper<LongWritable, Text, Text, IntWritable>
{

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

        //根据分割符tab键分割数据
        String[] data = value.toString().split("\t");
        //输出数据：(书名, 1)
        context.write(new Text(data[2]), new IntWritable(1));
    }

}
```

2、创建LibraryReducer类并继承 Reducer类，步骤请参考前面的实验，然后重写reduce()和cleanup()

先计算每本书的借阅次数，将数据写入全局变量map，然后在cleanup中对全局变量中的所有数据排序，并取得前10

完整代码如下：

```
import java.io.IOException;
import java.util.Collections;
import java.util.Comparator;
import java.util.HashMap;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class LibraryReducer extends Reducer<Text, IntWritable, Text,
IntWritable> {

    Map<String, Integer> map = new HashMap<String, Integer>();

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

        int count = 0;
        //统计每本书借出次数
        for (IntWritable val : values) {
            count++;
        }
        String bookName = key.toString();
        //统计完的数据存入map中
        map.put(bookName, count);
    }

    @Override
    protected void cleanup(Reducer<Text, IntWritable, Text, IntWritable>.Context
context)
        throws IOException, InterruptedException {

        List<Map.Entry<String, Integer>> list = new LinkedList<Map.Entry<String,
Integer>>(map.entrySet());
        Collections.sort(list, new Comparator<Map.Entry<String, Integer>>() {
            //根据书本借出次数进行降序排序
            public int compare(Entry<String, Integer> o1, Entry<String, Integer>
o2) {
                return (int) o2.getValue() - o1.getValue();
            }
        });

        //输出借出数量前10的书记名称
        for (int i = 0; i < 10; i++) {
```

```

        context.write(new Text(list.get(i).getKey()), new
IntWritable(list.get(i).getValue()));
    }
}
}

```

3、创建LibraryMain类并编写main()

完整代码如下：

```

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.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class LibraryMain {
    public static void main(String[] args) throws Exception {
        //配置hadoop运行环境
        Configuration conf = new Configuration();

        //创建任务实例，与任务驱动类
        Job job = Job.getInstance(conf);
        job.setJarByClass(LibraryMain.class);

        //配置Mapper类与输出格式
        job.setMapperClass(LibraryMapper.class);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(IntWritable.class);

        //配置Reducer类与输出格式
        job.setReducerClass(LibraryReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        //配置数据读取与输出路径
        FileInputFormat.setInputPaths(job, new
Path("hdfs://localhost:9000/001/shiyan5/input/books.txt"));
        FileOutputFormat.setOutputPath(job, new
Path("hdfs://localhost:9000/001/shiyan5/output"));

        job.waitForCompletion(true);
    }
}

```

4、执行程序并查看结果

执行main()，并使用hadoop shell 查看结果

```

hdfs dfs -ls /001/shiyan5/output
hdfs dfs -cat /001/shiyan5/output/part-r-00000

```

```
ubuntu@32e662a3cc50:~$ hdfs dfs -ls /bookout/part-r-00000
-rw-r--r--  3 ubuntu supergroup      178 2022-02-19 09:22 /bookout/part-r-00000
ubuntu@32e662a3cc50:~$ hdfs dfs -cat /bookout/part-r-00000
莎士比亚全集 I-XII      409
双城记      286
呼啸山庄      264
雾都孤儿      222
了不起的盖茨比      200
傲慢与偏见      183
巴黎圣母院      176
复活      168
飘      165
红与黑      158
```

实验总结

该实验是需要先通过mapreduce计算各图书借阅次数，然后将结果排序取得前十，我们都知道reduce是每组执行一次，所以我们先将每组计算的结果放入全局变量中，然后通过cleanup()将全局变量中的所有结果值进行排序并取得前十。