大数据学习笔记（5）-- java 开发 mapreduce

作者：kevinBobo
原文链接：https://ld246.com/article/1546398931809
来源网站：链滴

首先引入hadoop-client依赖

```
<!-- hadoop依赖 -->
<dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-client</artifactId>
    <version>2.9.2</version>
</dependency>
```

开发maper

```
package com.bobo.mapreduce;

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

import java.io.IOException;

/**
 * KEYIN:是map task读取到的数据key类型，是一行的起始偏移量long
 * VALUEIN：是 map task读取到的数据value类型，是一行的内容string
 *
 * KEYOUT：是用户的自定义map方法要返回的结果kv数据的key的类型，在wordcount逻辑中，我要返回的是单词的string
 * VALUEOUT：是用户的自定义map方法要返回的结果kv数据的value类型，在wordcount逻辑中我要返回的是中暑integer
 *
 * 但是，在mapreduce中，map产生的数据需要传输给reduce，需要进行序列化和反序列化，而jdk的原声序列化机制产生的数据量比较冗余，
 * 就会导致数据在mapreduce运行过程中传输效率低下
 * 所以，hadoop专门设计了自己的序列化机制，那么，maoreduce中传输的类型就必须实现hadoo自己的序列化接口
 *
 * hadoop为jdk中常用的基本类型Long，String，Integer，Float等数据类型封装了自己的实现了haoop序列化的接口类型：
 * LongWritable，Text，IntWritable，FloatWritable
 *
 * @author bobo
 * @Description:
 * @date 2019-01-01 18:38
 */
public class WordCountMapper extends Mapper<LongWritable, Text,Text, IntWritable> {

    @Override
    protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        String text = value.toString();
        String[] words = text.split(" ");
        for (String word :words) {
            context.write(new Text(word),new IntWritable(1));
```

```
        }
    }

}
```

开发reducer

```
package com.bobo.mapreduce;

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

import java.io.IOException;
import java.util.Iterator;
import java.util.concurrent.atomic.AtomicInteger;

/**
 * @author bobo
 * @Description:
 * @date 2019-01-01 18:54
 */
public class WordCountReducer extends Reducer<Text, IntWritable,Text,IntWritable> {

    @Override
    protected void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOEx
eption, InterruptedException {
        AtomicInteger count = new AtomicInteger(0);
        Iterator<IntWritable> iterator = values.iterator();
        iterator.forEachRemaining(value-> count.addAndGet(value.get()));
        context.write(key,new IntWritable(count.intValue()));
    }
}
```

开发程序客户端运行

```
package com.bobo.mapreduce;

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;

/**
 * @author bobo
 * @Description:
 * @date 2019-01-01 19:48
 */
public class JobSubmitter {
```

```
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        //1.设置jov运行时要访问的默认文件系统
        conf.set("fs.defaultFS", "hdfs://namenode:9000");
        conf.set("dfs.client.use.datanode.hostname","true");
        //2.设置job提交到哪里去运行
        conf.set("yarn.resourcemanager.hostname", "namenode");
        conf.set("mapreduce.framework.name", "yarn");

        Job job = Job.getInstance(conf);

        //封装参数：jar包所在的位置
        job.setJar("/Users/bobo/Documents/study-hadoop/study-mapreduce/target/study-map
educe-1.0-SNAPSHOT.jar");

        //封装参数：job要调用的mapper实现类和reducer实现类
        job.setMapperClass(WordCountMapper.class);
        job.setReducerClass(WordCountReducer.class);

        //封装参数：本次job的mapper实现类产生结果数据的key，value类型
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(IntWritable.class);

        //封装参数：本次job的reducer实现类产生结果数据的key，value类型
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        //封装参数：本次job要处理的输入数据集所在路径
        FileInputFormat.setInputPaths(job, new Path("/wordcount/input"));
        //输出路径不存在也可以
        FileOutputFormat.setOutputPath(job, new Path("/wordcount/output"));

        //封装参数：想要启动的reduce task数量
        job.setNumReduceTasks(3);

        //提交job给yarn
        boolean b = job.waitForCompletion(true);

        System.exit(b ? 0 : 1);

    }
}
```