并行计算课程

结 题 报 告

报告名称：	Hadoop 算法实现 WordCount 应用
姓　　名：	李润泽
学　　号：	3019244266
联系电话：	15942643201
电子邮箱：	Lirz3019244266@163.com
填写日期：	2021 年 5 月 2 日

2020 年制

一、实验内容概述

WordCount 是一个最简单的分布式应用实例，主要功能是统计输入目录中所有单词出现的总次数，如文本文件中有如下内容：

Hello world

则统计结果应为

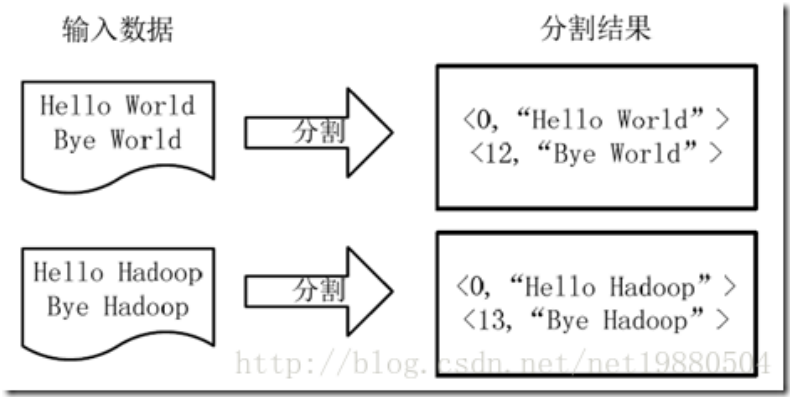
Hello 1

world 1

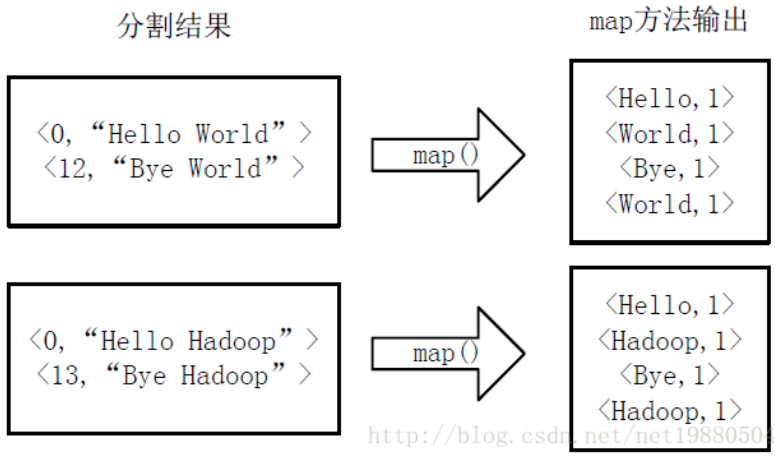
WordCount 可以使用多种方式实现，本次实验内容要求使用 Hadoop 或者 Spark 实现 WordCount 程序，并完成对应实验报告。

二、并行算法分析设计

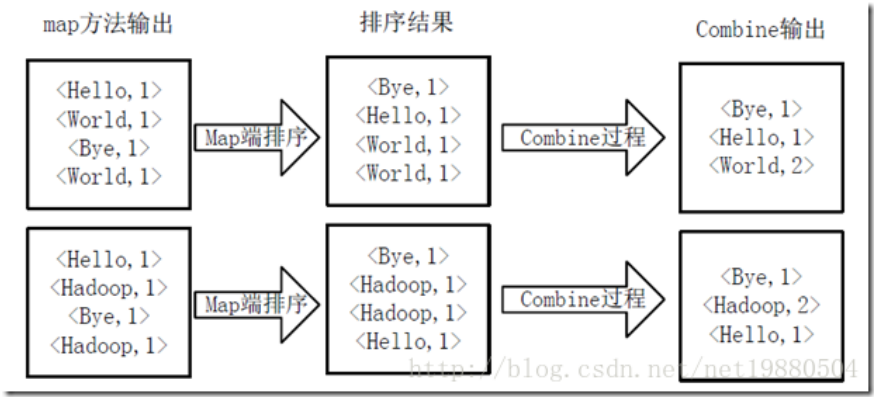
1、首先将文件拆分成 splits，并将文件按行分割形成<key, value>对，如下图。这一步由 MapReduce 框架自动完成，其中偏移量（key）包括回车所占的字符数。



2、将分割好的<key, value>对交给用户定义的 map 方法进行处理，生成新的<key, value>对，如下图所示。



3、在得到 map 方法输出的<key, value>对后，Mapper 会将它们按照 key 值进行排序，并执行 Combine 过程，将 key 值相同的 value 值累加，得到 Mapper 的最终输出结果，如下图所示。



4、Reduce 先对从 Mapper 接收的数据进行排序，再交由用户自定义的 reduce 方法进行处理，得到新的<key, value>对，并作为 WordCount 的输出结果，如下图所示。

5、代码

```
import java.io.IOException;
import java.util.StringTokenizer;

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;
```

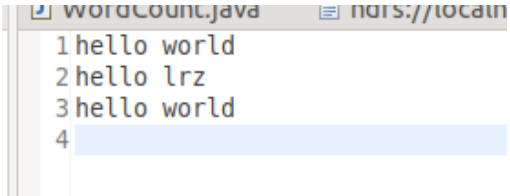
```
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

public class WordCount {

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

    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);
        }
    }

    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> <out>");
            System.exit(2);
        }
        @SuppressWarnings("deprecation")
        Job job = new Job(conf, "word count");
        job.setJarByClass(WordCount.class);
        job.setMapperClass(TokenizerMapper.class);
        job.setCombinerClass(IntSumReducer.class);
        job.setReducerClass(IntSumReducer.class);
        job.setOutputKeyClass(Text.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);
    }
}
```

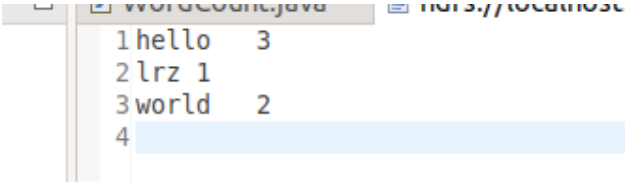
三、实验数据分析

测试环境：Ubuntu 18.04 Linux

测试数据：test01.txt



测试结果：



四、实验总结

1.实验遇到的问题：

1.1 环境的搭建（这个是最主要的问题） 在搭建环境是没有完备的参考教程，网络上的教程并不是十分全面，在准备 Hadoop 和 Java 环境的过程中，非常耗时繁琐

1.2 没有接触过 Java 语言，对语法和操作不熟悉

2.解决方法：

2.1 参考了十余篇博客以及教程，配置不成功报错后再查询错误的解决办法，逐个解决

2.2 参考教程学习 Java 语法，参照 WordCount 的示例完成

五、课程总结

此次实验的难点主要集中在集成环境的搭建上，但实验指导书并未给出详细指导，导致此次实验的主要精力消耗在了搭建环境而不是学习并行算法上，本质上偏离了布置上机实验的初衷，建议此门科目的第四个实验配置详细的环境配置文档和独立的实验指导书（仿照数字逻辑与数字系统等课程的实验），或者使用虚拟仿真实验教学平台创建一个配置好环境的实验，指导同学在平台上完成实验，再或者提供一个已经配置好环境的虚拟机上传到天大云盘供同学们下载后导入本地，以上方法都能让同学们的学习效果提升。