**北京邮电大学软件学院**

**2020-2021学年第二学期实验报告**

**课程名称：** 大数据原理与技术

**项目名称：**  **实验二：倒排索引**

**项目完成人：**

**姓名：\_\_王衔飞\_学号：\_\_\_\_\_\_\_\_**

**姓名：\_\_\_\_\_\_\_\_学号：\_\_****\_\_\_\_\_\_**

**姓名：\_\_\_\_\_\_\_\_学号：\_\_\_\_\_\_\_\_**

**姓名：\_\_\_\_\_\_\_\_学号：\_\_\_\_\_\_\_\_**

**姓名：\_\_\_\_\_\_\_\_学号：\_\_\_\_\_\_\_\_**

**指导教师：**\_\_\_\_\_\_\_\_\_\_\_孙鹏飞\_\_\_\_\_\_\_\_\_\_\_

**日 期： 2021年 4 月 22 日**

1. **实验目的**

使用map/reduce框架进行编写倒排索引

1. **实验内容**

1. 自己构造一组英文文档，或使用一组英文网页数据，使用MapReduce编程模型，实现倒排索引算法，对构造的文档集或网页数据集建立倒排索引

1. **实验环境**

MacBook CPU：Apple M1 (armv8a) VM: Parallel desktop OS:Ubuntu 20.04 aarch64(arm64) Hadoop 3.3.0-aarch64

1. **实验结果**

完成实验内容。

1. **附录**
2. 创建Java项目并引入Maven包管理器，配置Hadoop依赖

文本

描述已自动生成

1. 在HDFS中创建/input文件夹，并放入一些文章

图形用户界面

描述已自动生成

放入的文章分别为维基百科上对于iOS macOS Android的简介，如图

文本, 信件

描述已自动生成

文本, 信件

描述已自动生成

文本, 信件

描述已自动生成

1. 设计Mapper，Combiner，Reducer

Mapper的输入为文件的每个分块和每块的数据内容，输出的key为单词和其所在的文件名，value为1。

Combiner的输入为Mapper的输出，Combiner要完成的任务是整合每个文件的统计，输出为key为单词，value为单词所在的文件名和词频。

Reducer的输入是单个文件的统计情况，输出为所有文件的整合数据。

**Mapper：**

public static class InvertedIndexMapper extends Mapper<Object, Text, Text, Text> {

private Text keyInfo = new Text();

private Text valueInfo = new Text();

private FileSplit split;

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

split = (FileSplit) context.getInputSplit();

StringTokenizer itr = new StringTokenizer(value.toString());

while (itr.hasMoreTokens()) {

keyInfo.set(itr.nextToken() + ":"

+ split.getPath().toString().substring(split.getPath().toString().lastIndexOf('/') + 1));

valueInfo.set("1");

context.write(keyInfo, valueInfo);

}

}

}

**Combiner：**

public static class InvertedIndexCombiner extends Reducer<Text, Text, Text, Text> {

private Text info = new Text();

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

int sum = 0;

for(Text value : values) {

sum += Integer.parseInt(value.toString());

}

int splitIndex= key.toString().lastIndexOf(":");

info.set(key.toString().substring(splitIndex + 1) + ":" + sum);

key.set(key.toString().substring(0, splitIndex));

context.write(key, info);

}

}

**Reducer**

public static class InvertedIndexReducer extends Reducer<Text, Text, Text, Text> {

private Text result = new Text();

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

StringBuilder fileList = new StringBuilder();

fileList.append('[');

for(Text value : values) {

fileList.append(value.toString()).append(",");

}

fileList.deleteCharAt(fileList.length()-1);

fileList.append(']');

result.set(fileList.toString());

context.write(key, result);

}

}

1. 配置主函数并运行

将HDFS URL路径写入主函数并进行相应的调用工作

文本

描述已自动生成

1. 查看输出

输出与预期基本一致

图形用户界面, 应用程序

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成

**实验心得：**

通过本次实验，了解了如何使用MapReduce框架编写倒排索引。

**附录：程序源代码**

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.input.FileSplit;

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

public class InvertedIndex {

public static class InvertedIndexMapper extends Mapper<Object, Text, Text, Text> {

private Text keyInfo = new Text();

private Text valueInfo = new Text();

private FileSplit split;

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

split = (FileSplit)context.getInputSplit();

StringTokenizer itr = new StringTokenizer(value.toString());

while(itr.hasMoreTokens()) {

keyInfo.set(itr.nextToken() + ":" + split.getPath().toString().substring(split.getPath().toString().lastIndexOf('/')+1));

valueInfo.set("1");

context.write(keyInfo, valueInfo);

}

}

}

public static class InvertedIndexCombiner extends Reducer<Text, Text, Text, Text> {

private Text info = new Text();

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

int sum = 0;

for(Text value : values) {

sum += Integer.parseInt(value.toString());

}

int splitIndex= key.toString().lastIndexOf(":");

info.set(key.toString().substring(splitIndex + 1) + ":" + sum);

key.set(key.toString().substring(0, splitIndex));

context.write(key, info);

}

}

public static class InvertedIndexReducer extends Reducer<Text, Text, Text, Text> {

private Text result = new Text();

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

StringBuilder fileList = new StringBuilder();

fileList.append('[');

for(Text value : values) {

fileList.append(value.toString()).append(",");

}

fileList.deleteCharAt(fileList.length()-1);

fileList.append(']');

result.set(fileList.toString());

context.write(key, result);

}

}

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

// TODO Auto-generated method stub

Configuration conf = new Configuration();

Job job = new Job(conf, "InvertedIndex");

job.setJarByClass(InvertedIndex.class);

job.setMapperClass(InvertedIndexMapper.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setCombinerClass(InvertedIndexCombiner.class);

job.setReducerClass(InvertedIndexReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

String inputfile = "hdfs://localhost/input/";

String outputFile = "hdfs://localhost/output/";

FileInputFormat.addInputPath(job, new Path(inputfile));

FileOutputFormat.setOutputPath(job, new Path(outputFile));

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

}

}