Project: MapReduce Program + Full Inverted Index

By Emily Weng

Step 1:

Please draw three tables to show the processes done by mapper, combiner, and reducer to show the Full Inverted Index of these three file:

file 0's content "it is what it is"

file 1's content "what is it"

file 2's content "it is a banana"

Job: Fully Inverted Index											
Map Task								Reduce Task			
Map()				Combiner				Reduce()			
Input		Output		Input		Output		Input		Output	
file0	it is what is it	it	(0,0)	it	((0,0),(0,3)	it	2	a	(2,0)	a	1
		is	(0,1)	is	((0,1), (0,4))	is	2	banana	(2,1)	banana	1
		what	(0,2)	wh at	(0,2)	what	1	is	((0,1), (0,4),(1, 1),(2,3))	is	4
		it	(0,3)					it	(0,0),(0, 3),(1,0), (2,2))	it	4
		is	(0,4)					what	(0,2),(1, 0)	what	2

Job: Fully Inverted Index											
Map Task								Reduce Task			
Map()				Combiner				Reduce()			
Input		Output		Input		Output					
file1	what is it	what	(1,0)	it	(1,0)	it	1				
		it	(1,1)	is	(1,1)	is	1				
		is	(1,2)	what	(1,2)	what	1				
file2	it	a	(2,0)	a	(2,0)	a	1				
	is	banana	(2,1)	bana na	(2,1)	banana	1				
	a	it	(2,2)	it	(2,2)	it	1				
	banana	is	(2,3)	is	(2,3)	is	1				

Step 2:

Step 2: Convert a WordCount MapReduce program into a Partial Inverted Index MapReduce program with the three input files and expected output.

Code:

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

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Counter;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.Mapper.Context;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.FileSplit;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;


public class InvertedIndex extends Configured implements Tool{

        public static class InvertedIndexMapper extends
                                Mapper<LongWritable, Text, Text, IntWritable> {

                public static final String MalformedData = "MALFORMED";

                private Text outkey = new Text();
                private IntWritable outvalue = new IntWritable();

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

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

                        FileSplit fileSplit = (FileSplit)context.getInputSplit();
                        String filename = fileSplit.getPath().getName();
                        //System.out.println("File name "+filename);
                        //System.out.println("Directory and File name"+fileSplit.getPath().toString());

                        String line = value.toString();
                        StringTokenizer tokenizer = new StringTokenizer(line);
                        while (tokenizer.hasMoreTokens()) {
                                String word = tokenizer.nextToken().trim();
                                if(word.equals("#")){
                                        context.getCounter(MalformedData, word).increment(1);
                                }
                                else{
                                        outkey.set(word);
                                        outvalue = new IntWritable(
                                                        Integer.parseInt(filename.substring(4, filename.length()-4)));
                                        System.out.println(outkey+"  "+outvalue);
                                        context.write(outkey, outvalue);
                                }
                        }
                }
        }

        public static class InvertedIndexReducer extends
        Reducer<Text, IntWritable, Text, Text> {
private Text outputkey = new Text();
private List<Integer> outputvalue = new ArrayList<Integer>();

public void reduce(Text key, Iterable<IntWritable> values,
                Context context) throws IOException, InterruptedException {
        outputkey = key;
        outputvalue = new ArrayList<Integer>();;
        for (IntWritable val : values) {
                if(!outputvalue.contains(val.get())){
```

```
        for (IntWritable val : values) {
                if(!outputvalue.contains(val.get())){
                        outputvalue.add(val.get());
                }
        }
        context.write(outputkey, new Text(outputvalue.toString()));
}
}


        public static void main(String[] args) throws Exception {
                int exitCode = ToolRunner.run(new Configuration(), new InvertedIndex(), args);
                System.exit(exitCode);
        }

        @Override
        public int run(String[] args) throws Exception {
                if (args.length != 2){
                        System.out.printf("Usage: %s [generic options] <input> <output>\n",
                                        getClass().getSimpleName());
                        return -1;
                }
                Job job = new Job(getConf(), "InvertedIndex");
                job.setJarByClass(InvertedIndex.class);
                job.setOutputKeyClass(Text.class);
                job.setOutputValueClass(IntWritable.class);
                job.setMapperClass(InvertedIndexMapper.class);
                job.setReducerClass(InvertedIndexReducer.class);
                job.setInputFormatClass(TextInputFormat.class);
                job.setOutputFormatClass(TextOutputFormat.class);
                FileInputFormat.addInputPath(job, new Path(args[0]));
                FileOutputFormat.setOutputPath(job, new Path(args[1]));
                boolean success = job.waitForCompletion(true);
                if(success){
                        for(Counter counter: job.getCounters().getGroup(
                                        InvertedIndexMapper.MalformedData)){
                                                System.out.println(counter.getDisplayName()+"\t"+counter.getValue());
                                        }
                }
                return success ? 0 : 1;
        }

}
```

Step 3

Convert a Partial Inverted Index MapReduce program into a Full Inverted Index MapReduce program with the three input files and expected output.

Code:

```java
package org.myorg;

import java.io.IOException;
import java.util.*;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;

public class FullInvertedIndex {

    public static class IndexMapper extends MapReduceBase implements Mapper<LongWritable, Text, Text, Text> {

        private Text word = new Text();
        private Text location = new Text();

        public void map(LongWritable key, Text value, OutputCollector<Text, Text> output, Reporter reporter) throws IOException
 {
            FileSplit fileSplit = (FileSplit) reporter.getInputSplit();
            String fileName = fileSplit.getPath().getName();
            String line = value.toString();
            int lineNumber = (int) key.get();

            StringTokenizer tokenizer = new StringTokenizer(line);
            while (tokenizer.hasMoreTokens()) {
                String token = tokenizer.nextToken();
                word.set(token.toLowerCase());
                location.set(fileName + ":" + lineNumber);
                output.collect(word, location);
            }
        }
    }

    public static class IndexReducer extends MapReduceBase implements Reducer<Text, Text, Text, Text> {

        public void reduce(Text key, Iterator<Text> values, OutputCollector<Text, Text> output, Reporter reporter) throws IOExc
eption {
            Map<String, List<Integer>> map = new HashMap<>();

            while (values.hasNext()) {
                String[] fileAndLine = values.next().toString().split(":");
                String fileName = fileAndLine[0];
                int lineNumber = Integer.parseInt(fileAndLine[1]);
```

```
            int lineNumber = Integer.parseInt(fileAndLine[1]);

            map.putIfAbsent(fileName, new ArrayList<>());
            map.get(fileName).add(lineNumber);
        }

        StringBuilder result = new StringBuilder();
        for (Map.Entry<String, List<Integer>> entry : map.entrySet()) {
            if (result.length() > 0) {
                result.append("; ");
            }
            result.append(entry.getKey()).append(":").append(entry.getValue().toString());
        }

        output.collect(key, new Text(result.toString()));
    }
}

public static void main(String[] args) throws Exception {
    JobConf conf = new JobConf(FullInvertedIndex.class);
    conf.setJobName("fullinvertedindex");

    conf.setOutputKeyClass(Text.class);
    conf.setOutputValueClass(Text.class);

    conf.setMapperClass(IndexMapper.class);
    conf.setCombinerClass(IndexReducer.class);
    conf.setReducerClass(IndexReducer.class);

    conf.setInputFormat(TextInputFormat.class);
    conf.setOutputFormat(TextOutputFormat.class);

    FileInputFormat.setInputPaths(conf, new Path(args[0]));
    FileOutputFormat.setOutputPath(conf, new Path(args[1]));

    JobClient.runJob(conf);
```

Step 4

Use a VM (local or Cloud) or use Eclipse to create a JAR file and then copy the JAR file to Hadoop for processing. **Kept unsuccessfully run output after many tries, will need to redo**

```
eweng909@ubuntu2:~/hadoop-3.3.5/InvertedIndex$ ../bin/hadoop fs -ls /user/eweng909/invertedindex
Found 2 items
drwxr-xr-x   - eweng909 supergroup          0 2024-06-05 15:15 /user/eweng909/invertedindex/input
drwxr-xr-x   - eweng909 supergroup          0 2024-06-05 15:36 /user/eweng909/invertedindex/output
```

```
eweng909@ubuntu2:~/hadoop-3.3.5/FullInvertedIndex$ ../bin/hadoop fs -ls /user/eweng909/fullinvertedindex/output
ls: `/user/eweng909/fullinvertedindex/output': No such file or directory
eweng909@ubuntu2:~/hadoop-3.3.5/FullInvertedIndex$ ../bin/hadoop fs -ls /user/eweng909/fullinvertedindex
Found 1 items
drwxr-xr-x   - eweng909 supergroup          0 2024-06-05 16:06 /user/eweng909/fullinvertedindex/input
eweng909@ubuntu2:~/hadoop-3.3.5/FullInvertedIndex$
```

Step 5

Submit the URL of your GitHub webpage as part of the homework answers

Cloud Computing

MapReduce

Full Inverted Index