U ovom file su zadaca i labos rjesenja. Odgovi na pitanja su u komentarima koda

Homework

Zadatak 1

```java
import org.apache.hadoop.io.*;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
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 java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;

// - Koliko različitih vozila se nalazi u ulaznoj datoteci?  9834
// - - Koliko je trajala najdulja ukupna vožnja jednog taksija? Koja je
minimalna, a koja najdulja vožnja tog taksija?
// Taxi FFFECF7... 60 najkraca, 3660 najdulja, ukupon 134640 sve voznje
//  - Koje ste promjene morali napraviti na izvornom kodu prilikom uvođenja
optimizacijske funkcije Combine?
// Dodati samo combiner klasu koja je ekvivalentna reduceru
//  with combiner hadoop jar target/lab1-1.0-SNAPSHOT.jar lab2_1 trip_data.csv
out.00  9.06s user 0.22s system 166% cpu 5.570 total
// without combiner hadoop jar target/lab1-1.0-SNAPSHOT.jar lab2_1
trip_data.csv out.01  9.79s user 0.35s system 153% cpu 6.618 total

public class lab2_1 {

    public static class Statistics implements  Writable {
        Double min, max, sum;

        public Statistics(Double min, Double max, Double sum) {
            this.min = min;
            this.max = max;
            this.sum = sum;
        }

        public Statistics(Double a) {
            this(a,a,a);
        }

        public Statistics() {
            this(null);
        }
        @Override
        public void write(DataOutput dataOutput) throws IOException {
            dataOutput.writeDouble(min);
            dataOutput.writeDouble(max);
            dataOutput.writeDouble(sum);
        }

        @Override
        public void readFields(DataInput dataInput) throws IOException {
            min = dataInput.readDouble();
            max = dataInput.readDouble();
            sum = dataInput.readDouble();
        }

        public void combine(Statistics other) {
            if (this.min == null) {
                this.min = other.min;
                this.max = other.max;
                this.sum = 0.0;
            }
            this.min = Math.min(this.min, other.min);
            this.max = Math.max(this.max, other.max);
            this.sum += other.sum;
        }

        @Override
        public String toString() {
            return "(" + min+ ", " + max + ") total: " + sum;
        }
    }

    public static class TotalTimeMapper
            extends Mapper<LongWritable, Text, Text, Statistics> {

        public void map(LongWritable key, Text value, Context context
        ) throws IOException, InterruptedException {
            if (key.get() == 0)
                return;
            String[] record = value.toString().split(",");
            Text medallion = new Text(record[0]);
            Statistics trip_time = new
Statistics(Double.parseDouble(record[8]));
            context.write(medallion,trip_time);
        }
    }

    public static class DriveTimeReducer
            extends Reducer<Text,Statistics,Text,Statistics> {
        public void reduce(Text key, Iterable<Statistics> values,
                           Context context
        ) throws IOException, InterruptedException {
            Statistics sol = new Statistics();
            for (Statistics val : values) {
                sol.combine(val);
            }
            context.write(key, sol);
        }
    }

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "drive time lab 2.1");
        job.setJarByClass(lab2_1.class);
        job.setMapperClass(TotalTimeMapper.class);
        job.setCombinerClass(DriveTimeReducer.class);
        job.setReducerClass(DriveTimeReducer.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Statistics.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

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

}
```

Zadatak 2

```
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.io.*;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Partitioner;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;

// medallion,hack_license,vendor_id,rate_code,
// store_and_fwd_flag,pickup_datetime,dropoff_datetime,passenger_count,
// trip_time_in_secs,trip_distance,pickup_longitude,pickup_latitude,
// dropoff_longitude,dropoff_latitude

//Koliko je vožnji realizirano u pojedinom području, tj. u užem centru i u
širem gradskom području, i to tako da
//        je broj putnika bio 1, 2-3 putnika ili 4 i više putnika?
//
```

```
//0 .
//        525517 ./part-r-00000 (outer city, 1 putnik)
//        151509 ./part-r-00001 (2-3 putnika)
//        184797 ./part-r-00002 (4+ putnika)
//        244814 ./part-r-00003 (inner city)
//        70458 ./part-r-00004 (..)
//        86294 ./part-r-00005 (..)
// Kategorije su definirane u kodu u funkciji valueToPartition
//        - Koje ste promjene morali napraviti na izvornom kodu prilikom
uvođenja funkcije Partition?
//
// Paziti da se mapperi ispravno particioniraju na reducere (kojih sada ima 6
zbog particija)
//        - Koliko je vožnji navedeno u svakoj podskupini?
//
// Nije mi bas jasno pitanje...odgovor je ekvivalentan prvom pitanju

public class lab2_2 {
    public static class PartitioningMapper
            extends Mapper<LongWritable, Text, IntWritable, Text> {

        private static boolean isInnerCenter(String[] record) {
            Double longitude = Double.parseDouble(record[12]);
            Double latitude = Double.parseDouble(record[13]);

            if (!isInInnerCenter(longitude, latitude)) return false;

            longitude = Double.parseDouble(record[10]);
            latitude = Double.parseDouble(record[11]);

            if (!isInInnerCenter(longitude, latitude)) return false;

            return true;
        }

        private static boolean isInInnerCenter(Double longitude, Double
latitude) {
            if (longitude < -74. || longitude > - 73.95)
                return false;
            if (latitude < 40.75 || latitude > 40.8)
                return false;
            return true;
        }

        public static int valueToPartition(Text value) {
            String[] record = value.toString().split(",");
            Integer passenger_count = Integer.parseInt(record[7]);

            int sol = 0;
            if (isInnerCenter(record))
                sol = 3;
            switch (passenger_count) {
                case 1:
                    break;
                case 2:
                case 3:
                    sol += 1;
                    break;
                default:
                    sol += 2;
            }
            return sol;
        }

        public void map(LongWritable key, Text value, Context context
        ) throws IOException, InterruptedException {
            if (key.get() == 0)
                return;
            context.write(new IntWritable(valueToPartition(value)), value);
        }
    }

    public static class TypePartitioner extends Partitioner<IntWritable, Text>
{
        @Override
        public int getPartition(IntWritable intWritable, Text text, int i) {
            return intWritable.get();
        }
    }

    public static class IdentityReducer extends Reducer<IntWritable, Text,
NullWritable, Text> {
        @Override
        protected void reduce(IntWritable key, Iterable<Text> values, Context
context) throws IOException, InterruptedException {
            context.write(NullWritable.get(), new
Text("medallion,hack_license,vendor_id,rate_code," +

"store_and_fwd_flag,pickup_datetime,dropoff_datetime,passenger_count," +

"trip_time_in_secs,trip_distance,pickup_longitude,pickup_latitude," +
                    "dropoff_longitude,dropoff_latitude"));
            for (Text value : values) {
                context.write(NullWritable.get(), value);
            }
        }
    }

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        FileSystem.get(conf).delete(new Path(args[1]), true);

        Job job = Job.getInstance(conf, "drive time lab 2.1");
        job.setJarByClass(lab2_1.class);
        job.setMapperClass(PartitioningMapper.class);
        job.setPartitionerClass(TypePartitioner.class);
        job.setReducerClass(IdentityReducer.class);
        job.setNumReduceTasks(6);

        job.setOutputKeyClass(IntWritable.class);
        job.setOutputValueClass(Text.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

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

}
```

Zadatak 3

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Partitioner;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;

// medallion,hack_license,vendor_id,rate_code,
// store_and_fwd_flag,pickup_datetime,dropoff_datetime,passenger_count,
// trip_time_in_secs,trip_distance,pickup_longitude,pickup_latitude,
// dropoff_longitude,dropoff_latitude

//Koliko ste MapReduce poslova izvršili u vašem kôdu?
// 2 MapReduce posla
//        Koliko je različitih taksija realiziralo vožnje u pojedinom području,
tj. u užem centru i u širem gradskom području,
//        i to tako da je broj putnika bio 1, 2-3 putnika ili 4 i više putnika?
// Isti odgovor kao i u 2.2
public class lab2_3 {
    public static Path TMPDIR = new Path("/tmp/nmiculinic/");

    public static class MMMaper extends Mapper<LongWritable, Text, Text,
lab2_1.Statistics> {
        public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
            if (key.get() == 0)
                return;

            String[] record = value.toString().split(",");
            Text medallion = new Text(record[0] + "_" +
lab2_2.PartitioningMapper.valueToPartition(value));
            lab2_1.Statistics trip_time = new
lab2_1.Statistics(Double.parseDouble(record[8]));
            context.write(medallion,trip_time);
        }
    }

    public static class Partitioneeeer extends Partitioner<Text,
lab2_1.Statistics> {
        @Override
        public int getPartition(Text text, lab2_1.Statistics statistics, int i)
{
            String s = text.toString();
            s = s.substring(s.length() - 1);
            return Integer.parseInt(s);
        }
    }

    public static class Reeeducer extends Reducer<Text, lab2_1.Statistics,
Text, lab2_1.Statistics> {
        @Override
        protected void reduce(Text key, Iterable<lab2_1.Statistics> values,
Context context) throws IOException, InterruptedException {
            lab2_1.Statistics sol = new lab2_1.Statistics();
            for (lab2_1.Statistics val : values) {
                sol.combine(val);
            }
            String kk = key.toString();
            kk = kk.substring(0, kk.length() - 2);
            context.write(new Text(kk), sol);
        }
    }

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        FileSystem.get(conf).delete(new Path(args[1]), true);
        FileSystem.get(conf).delete(TMPDIR, true);

        Job first = Job.getInstance(conf, "drive time lab 2.2");
        first.setJarByClass(lab2_3.class);
        first.setMapperClass(lab2_2.PartitioningMapper.class);
        first.setPartitionerClass(lab2_2.TypePartitioner.class);
        first.setReducerClass(lab2_2.IdentityReducer.class);
        first.setNumReduceTasks(6);
        first.setOutputKeyClass(IntWritable.class);
        first.setOutputValueClass(Text.class);

        FileInputFormat.addInputPath(first, new Path(args[0]));

        FileOutputFormat.setOutputPath(first, TMPDIR);

        int code = first.waitForCompletion(true) ? 0 : 1;
        if (code == 0) {
            Job second = Job.getInstance(conf, "drive time lab 2.3");
            second.setJarByClass(lab2_3.class);
            second.setMapperClass(MMMaper.class);
            second.setReducerClass(Reeeducer.class);
            second.setPartitionerClass(Partitioneeeer.class);

            second.setNumReduceTasks(6);
            second.setOutputKeyClass(Text.class);
            second.setOutputValueClass(lab2_1.Statistics.class);


            FileInputFormat.addInputPath(second, TMPDIR);
            FileOutputFormat.setOutputPath(second, new Path(args[1]));
            code = second.waitForCompletion(true) ? 0 : 1;
        }
//          FileSystem.get(conf).delete(TMPDIR, true);
        System.exit(code);
    }

}
```

Labos

Zadatak 1

```
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.io.*;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import java.io.IOException;
//
//- Koje funkcije sadrži vaš MapReduce program (map, reduce, partitioner,
combiner, itd.)?
// Samo mapper
// - Koliko izlaznih datoteka je nastalo nakon izvođenja vašeg MapReduce
programa?
// 3 izlazne datoteke
```

```java
// wc -l *
//          685290 part-m-00000
//          684945 part-m-00001
//          590542 part-m-00002
//          0 _SUCCESS
//          1960777 total
// Ukupno 1 960 777 je zadovoljilo kriterija
//
// - Koliko zapisa u ulaznoj datoteci nije zadovoljilo zadane kriterije?
// 1999999 /home/lpp/Downloads/sorted_data.csv ukupno zapisa
// 39222 nije zadovoljilo kriterija (razlika)

public class l2_1 {

    private static final double GRID_WIDTH = 0.008983112;
    private static final double GRID_LENGTH = 0.011972;

    private static final double BEGIN_LON = -74.913585;
    private static final double BEGIN_LAT = 41.474937;

    public static int[] getCellId(double lat, double lon) {
        int[] sol = new int[2];
        sol[0] = (int) (((lon - BEGIN_LON) / GRID_LENGTH) + 1);
        sol[1] = (int) (((BEGIN_LAT - lat) / GRID_WIDTH) + 1);
        return sol;
    }

    public static boolean isInArea(double lat, double lon){
        int[] cell = getCellId(lat, lon);
        if (cell[0] < 0 || cell[0] > 150)
            return false;
        if (cell[1] < 0 || cell[1] > 150)
            return  false;
        return true;
    }


    public static boolean isInArea(String[] record) {
        Double lon_in = Double.parseDouble(record[6]);
        Double lat_in = Double.parseDouble(record[7]);
        Double log_out = Double.parseDouble(record[8]);
        Double lat_out = Double.parseDouble(record[9]);
        return isInArea(lat_in, lon_in) && isInArea(lat_out, log_out);
    }

    public static double profit(String[] record) {
        return Double.parseDouble(record[16]);
    }

    public static class FilterMapper
            extends Mapper<LongWritable, Text, NullWritable , Text> {
```

```
        public void map(LongWritable key, Text value, Context context
        ) throws IOException, InterruptedException {
            String[] record = value.toString().split(",");
            if (profit(record) <= 0) return;
            if (!isInArea(record)) return;

            context.write(NullWritable.get(), value);
        }
    }

    public static int executeTask1(Configuration conf, Path in, Path out)
throws Exception{
        Job job = Job.getInstance(conf, "lab 2.1");
        job.setJarByClass(l2_1.class);
        job.setMapperClass(FilterMapper.class);
        job.setNumReduceTasks(0);

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

        FileInputFormat.addInputPath(job, in);
        FileSystem.get(conf).delete(out, true);
        FileOutputFormat.setOutputPath(job, out);
        return  job.waitForCompletion(true) ? 0 : 1;
    }

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();

        System.exit(executeTask1(conf, new Path(args[0]), new Path(args[1])));
    }
}
```

Zadatak 2

```
/*
- Koje funkcije sadrži vaš MapReduce program?
Mapper, reducer i partitioner
- Koje tipove podataka ste definirali kao izlaz iz MapReduce programa?
Izlaz mapera je Int (sat) i Text(cijela voznja), dok je izlaz reducera
definiran u zadatku
*/

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Partitioner;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.*;

public class l2_2 {

    public static int getHoD(String[] record) throws ParseException {
        String dateTimeString = record[2];
        SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        Date dateTime = format.parse(dateTimeString);
        Calendar cal = Calendar.getInstance();
        cal.setTime(dateTime);
        return cal.get(Calendar.HOUR_OF_DAY);
    }

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

        public void map(LongWritable key, Text value, Context context
        ) throws IOException, InterruptedException {
            String[] record = value.toString().split(",");
            try {
                context.write(new IntWritable(l2_2.getHoD(record)), value);
            } catch (ParseException ex) {
                System.err.println(ex);
            }
        }
    }

    public static class HourPartitioner extends Partitioner<IntWritable, Text>
{
        @Override
        public int getPartition(IntWritable intWritable, Text text, int
numPartitions) {
            return intWritable.get();
        }
    }

    public static class HourReducer
            extends Reducer<IntWritable, Text, NullWritable, Text> {

        @Override
        protected void reduce(IntWritable key, Iterable<Text> values, Context
context) throws IOException, InterruptedException {
            Map<String, Integer> drives = new HashMap<>();
            Map<String, Double> profits = new HashMap<>();

            for (Text value : values) {
                String[] record = value.toString().split(",");
                double profit = l2_1.profit(record);
                Double lon_in = Double.parseDouble(record[6]);
                Double lat_in = Double.parseDouble(record[7]);
                int[] cellId = l2_1.getCellId(lat_in, lon_in);
                String cell = "Cell " + "(" + cellId[0] + ", " + cellId[1] +
")";

                drives.put(cell, 1 + (drives.get(cell) != null ?
drives.get(cell) : 0));
                profits.put(cell, profit + (profits.get(cell) != null ?
profits.get(cell) : 0.0));
            }

            context.write(NullWritable.get(), new Text(key.toString()));
            String max_drive_cell = Collections.max(drives.entrySet(),
Comparator.comparing(Map.Entry::getValue)).getKey();
            context.write(NullWritable.get(), new Text(max_drive_cell + " -> "
+ drives.get(max_drive_cell)));

            String max_profits_cell = Collections.max(profits.entrySet(),
Comparator.comparing(Map.Entry::getValue)).getKey();
            context.write(NullWritable.get(), new Text(max_profits_cell + " ->
" + profits.get(max_profits_cell)));
        }
    }


    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        System.exit(executeTask2(conf, new Path(args[0]), new Path(args[1])));
    }

    public static int executeTask2(Configuration conf, Path in, Path out)
throws IOException, InterruptedException, ClassNotFoundException {
        Job job = Job.getInstance(conf, "Lab 2.2");
        job.setJarByClass(l2_2.class);
        job.setMapperClass(HourMapper.class);
        job.setPartitionerClass(HourPartitioner.class);
        job.setReducerClass(HourReducer.class);
        job.setNumReduceTasks(24);

        job.setOutputKeyClass(IntWritable.class);
        job.setOutputValueClass(Text.class);

        FileInputFormat.addInputPath(job, in);
        FileSystem.get(conf).delete(out, true);
        FileOutputFormat.setOutputPath(job, out);
        return job.waitForCompletion(true) ? 0 : 1;
    }
}
```

Zadatak 3

```java
/*

- Kolika je veličina ulaznog skupa podataka?
32GB
- Koliko ste MapReduce poslova izvršili u vašem kôdu?
2
- Koje ste promjene napravili za prvi, a koje za drugi MapReduce posao?
Extraktao sam metode iz l2_1 i l2_2 da ih mogu lijepse zvati. Ulancao sam ih s
provjerom izlaznog koa

- Koje su prednosti, a koji nedostaci ovakvog sažimanja međurezultata?
Filtracijom je working set smanjen za drugi ulancani posao te je brzi.

*/

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Partitioner;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.*;

public class l2_3 {

    public static Path TMPDIR = new Path("nmiculinic/tmp");

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        int code = l2_1.executeTask1(conf, new Path(args[0]), TMPDIR);
        if (code == 0) {
            code = l2_2.executeTask2(conf, TMPDIR, new Path(args[1]));
        }
        FileSystem.get(conf).delete(TMPDIR, true);
        System.exit(code);
    }

}
```