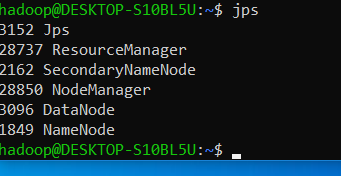
Cb.en.u4cse22431

# Trishika reddy

Jps command



2) import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

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

public class AverageTemp {

public static class TempMapper extends Mapper<Object, Text, IntWritable, DoubleWritable> {

private IntWritable occupancyCount = new IntWritable();

private DoubleWritable temperature = new DoubleWritable();

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

String[] columns = line.split(",");

if (columns.length >= 6) {

try {

int occupancy = Integer.parseInt(columns[14]);

double temp = Double.parseDouble(columns[2]);

} catch (NumberFormatException e) {

System.err.println("Skipping invalid data: " + line);

}

}

}

}

public static class TempReducer extends Reducer<IntWritable, DoubleWritable, IntWritable, DoubleWritable> {

private DoubleWritable result = new DoubleWritable();

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

int count = 0;

double sum = 0;

for (DoubleWritable val : values) {

sum += val.get();

count++;

}

if (count > 0) {

double averageTemp = sum / count;

result.set(averageTemp);

context.write(key, result);

}

}

}

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

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Average Temperature by Occupancy Count");

job.setJarByClass(AverageTemp.class);

job.setMapperClass(TempMapper.class);

job.setReducerClass(TempReducer.class);

job.setOutputKeyClass(IntWritable.class);

job.setOutputValueClass(DoubleWritable.class);

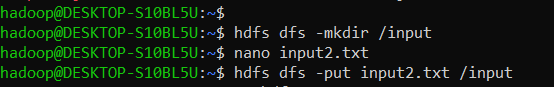
FileInputFormat.addInputPath(job, new Path(args[0]));

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

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

}

}

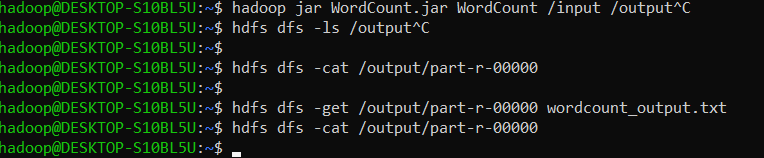


Ss of the part files



Output

Runned but not showing any output ..all commands runned



2)

import java.io.IOException;

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;

public class MonthlyOccupancy {

public static class OccupancyMapper extends Mapper<Object, Text, Text, IntWritable> {

private Text month = new Text();

private IntWritable occupancyCount = new IntWritable();

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

String line = value.toString().trim();

if (line.isEmpty()) return;

String[] columns = line.split(",");

if (columns.length >= 16) {

try {

String timestamp = columns[0];

String date = timestamp.split(" ")[0];

String[] dateParts = date.split("-");

String monthStr = dateParts[1];

int occupancy = Integer.parseInt(columns[14]);

String time = timestamp.split(" ")[1];

int hour = Integer.parseInt(time.split(":")[0]);

if (hour >= 9 && hour < 17) {

month.set(monthStr);

} catch (NumberFormatException e) {

System.err.println("Skipping invalid data: " + line);

}

}

}

}

public static class OccupancyReducer 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 totalOccupancy = 0;

for (IntWritable val : values) {

totalOccupancy += val.get();

}

result.set(totalOccupancy);

context.write(key, result);

}

}

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

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Monthly Occupancy During Working Hours");

job.setReducerClass(OccupancyReducer.class);

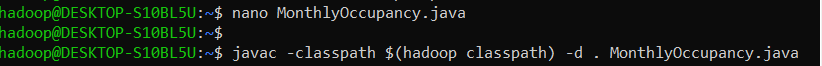
job.setOutputKeyClass(Text.class);

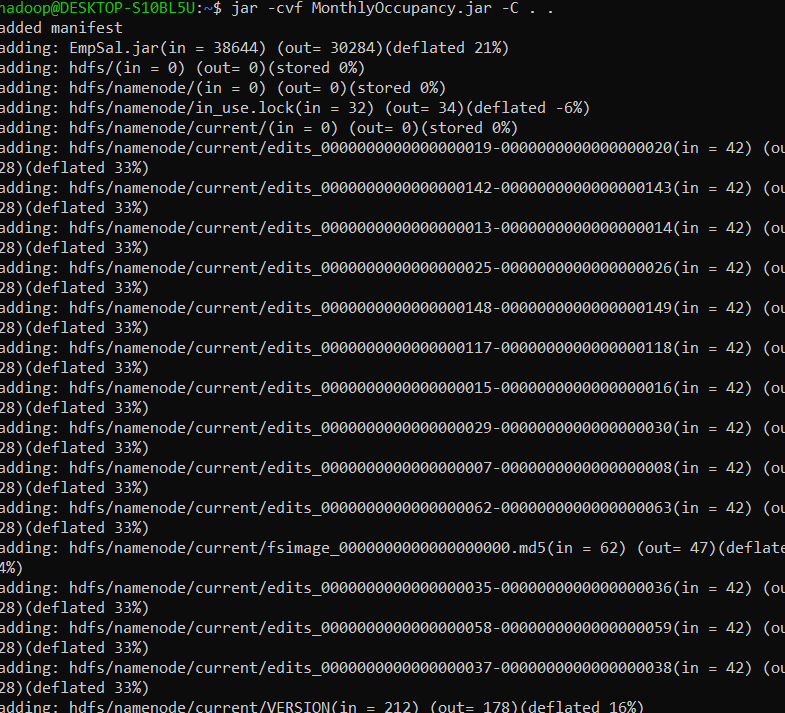
job.setOutputValueClass(IntWritable.class);

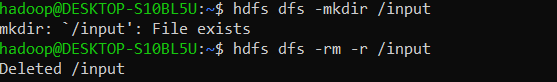
System.exit(job.waitForCompletion(true) ? 0 : 1);

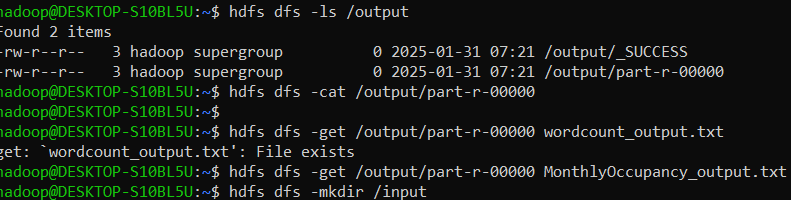
}

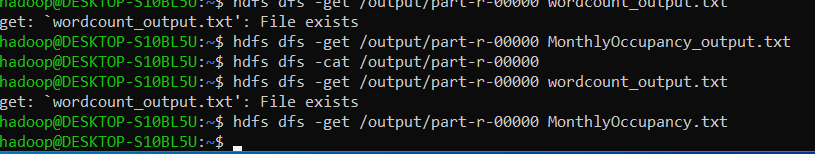
}











Output



