|  |  |
| --- | --- |
|  | |
|  | |
| import java.io.\*; | |
|  | import java.text.SimpleDateFormat; |
|  | import java.text.ParseException; |
|  | import java.util.Date; |
|  | import org.apache.hadoop.io.IntWritable; |
|  | import org.apache.hadoop.io.Text; |
|  | import org.apache.hadoop.io.LongWritable; |
|  | import org.apache.hadoop.mapreduce.Job; |
|  | import org.apache.hadoop.mapreduce.Mapper; |
|  | import org.apache.hadoop.mapreduce.Reducer; |
|  | import org.apache.hadoop.conf.\*; |
|  | import org.apache.hadoop.fs.\*; |
|  | import org.apache.hadoop.mapreduce.lib.input.\*; |
|  | import org.apache.hadoop.mapreduce.lib.output.\*; |
|  |  |
|  |  |
|  | public class stdcalls { |
|  |  |
|  | public static class MapClass extends Mapper<LongWritable,Text,Text, IntWritable> |
|  | { |
|  | Text phonenumber= new Text(); |
|  | IntWritable durationinminutes= new IntWritable(); |
|  | public void map(LongWritable key, Text value, Context context)throws IOException, InterruptedException |
|  | { |
|  | try{ |
|  | String[] parts = value.toString().split(","); |
|  | if(parts[4].equals("1")) { |
|  | phonenumber.set(parts[0]); |
|  | String callEndtime= parts[3]; |
|  | String callstarttime= parts[2]; |
|  | long duration= toMillis(callEndtime)- toMillis(callstarttime); |
|  | durationinminutes.set((int)(duration/(1000\*60))); |
|  | context.write(phonenumber, durationinminutes); |
|  | } |
|  | } |
|  | catch(Exception e) |
|  | { |
|  | System.out.println(e.getMessage()); |
|  | } |
|  | } |
|  | private long toMillis(String date)throws ParseException { |
|  | SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss"); |
|  | Date dateFrm= null; |
|  | try { |
|  |  |
|  | dateFrm=format.parse(date); |
|  |  |
|  | } |
|  | catch(ParseException e) { |
|  | e.printStackTrace(); |
|  | } |
|  | return dateFrm.getTime(); |
|  | } |
|  | } |
|  |  |
|  | public static class ReduceClass extends Reducer<Text,IntWritable,Text,IntWritable> |
|  | { |
|  | private IntWritable result = new IntWritable(); |
|  |  |
|  | public void reduce(Text key, Iterable<IntWritable> values,Context context) throws IOException, InterruptedException { |
|  | long sum=0; |
|  | for (IntWritable val : values) |
|  | { |
|  | sum+=val.get(); |
|  | } |
|  |  |
|  | result.set((int)sum); |
|  | if(sum >=60) { |
|  |  |
|  | context.write(key, result); |
|  | } |
|  | } |
|  | } |
|  |  |
|  |  |
|  | public static void main(String[] args) throws Exception { |
|  | Configuration conf = new Configuration(); |
|  | conf.set("mapreduce.output.textoutputformat.seperator"," ,"); |
|  | Job job = Job.getInstance(conf, "STDcalls"); |
|  | job.setJarByClass(stdcalls.class); |
|  | job.setMapperClass(MapClass.class); |
|  | //b.setCombinerClass(ReduceClass.class); |
|  | job.setReducerClass(ReduceClass.class); |
|  | //job.setNumReduceTasks(0); |
|  | job.setOutputKeyClass(Text.class); |
|  | job.setOutputValueClass(IntWritable.class); |
|  | job.setInputFormatClass(TextInputFormat.class); |
|  | job.setOutputFormatClass(TextOutputFormat.class); |
|  | FileInputFormat.addInputPath(job, new Path(args[0])); |
|  | FileOutputFormat.setOutputPath(job, new Path(args[1])); |
|  | System.exit(job.waitForCompletion(true) ? 0 : 1); |
|  | } |
|  |  |
|  | } |
|  |  |
|  |  |