package com.hadoop.hadoop;

import java.io.IOException;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

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

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

public class WJob

{

public static void main(String[] args) throws IOException,

ClassNotFoundException, InterruptedException

{

// TODO Auto-generated method stub

/\*

\* if(args.length !=2){

\* System.err.println("Usage: MaxTemperature <input path> <output path>"

\* ); System.exit(-1); }

\*/

long startTime = System.currentTimeMillis();

@SuppressWarnings("deprecation")

Job job = new Job();

job.setJarByClass(WJob.class);

job.setJobName("hadoop\_of\_video");

FileInputFormat.addInputPath(job, new Path(

"hdfs://master:9000/WHadoop/input/output"));

FileOutputFormat.setOutputPath(job, new Path(

"hdfs://master:9000/WHadoop/output"));

job.setMapperClass(WMap.class);

System.out.println(job.getMapperClass());

job.setReducerClass(WReduce.class);

System.out.println(job.getReducerClass());

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

System.out.println("Map " + job.getMapOutputKeyClass()

+ job.getMapOutputValueClass());

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

System.out.println("Put " + job.getOutputKeyClass()

+ job.getOutputValueClass());

// System.out.println(job.waitForCompletion(true));

System.out.println(job.waitForCompletion(true) ? true : false);

long endTime = System.currentTimeMillis();

System.out.println(((endTime - startTime) / 1000));

}

}

package com.hadoop.hadoop;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Map;

import java.util.StringTokenizer;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

import com.hadoop.util.Count;

public class WMap extends Mapper<LongWritable, Text, Text, Text>

{

private static Text line = null;

private Text url = null;

private static Map<String,double[]> instantMap = null;

public void map(LongWritable key, Text value, Context context)

throws IOException, InterruptedException

{

// TODO Auto-generated method stub

ArrayList<ArrayList<Object>> mapArray = new ArrayList<ArrayList<Object>>();

ArrayList<Object> al = new ArrayList<Object>();

StringTokenizer itr = new StringTokenizer(value.toString(), "+");

StringTokenizer shotItr = null;

// count()

// al:url;id;length;startTime;endTime;speed

while (itr.hasMoreTokens())

{

al = new ArrayList<Object>();

shotItr = new StringTokenizer(itr.nextToken().toString(), "-");

while(shotItr.hasMoreTokens())

{

al.add(shotItr.nextToken());

}

mapArray.add(al);

}

instantMap = Count.newInstanse().count(mapArray);

for(Map.Entry<String, double[]> entryMap : instantMap.entrySet())

{

url = new Text(entryMap.getKey());

String s = new String();

s+="[";

for(int i = 0; i < entryMap.getValue().length; i++)

{

s+=String.format("%.2f", entryMap.getValue()[i]);

if(i < entryMap.getValue().length-1)

{

s+=",";

}

}

s+="]\r\n";

line = new Text(s);

//context.write(url, line);

}

}

}

package com.hadoop.hadoop;

import java.io.IOException;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class WReduce extends Reducer<Text, Text, Text, Text>

{

/\*

\*(non-Javadoc)

\*

\* @see org.apache.hadoop.mapreduce.Reducer#reduce(KEYIN,

\* java.lang.Iterable, org.apache.hadoop.mapreduce.Reducer.Context) reduce

\*/

private Text result = null;

public void reduce(Text key, Text values, Context context)

throws IOException, InterruptedException

{

result = new Text();

result = values;

context.write(key, result);

}

}

package com.hadoop.util;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.HashMap;

import java.util.Map;

import com.hadoop.global.ICount;

public class Count implements ICount

{

public static Map<String, double[]> longMap = new HashMap<String, double[]>();

public static Count newInstanse()

{

return new Count();

}

public Map<String, double[]> count(ArrayList<ArrayList<Object>> list)

{

// TODO Auto-generated method stub

// need check

double[] line = null;

// count:startTime to endTime puls 1/speed

// al:url;id;length;startTime;endTime;speed

for (ArrayList<Object> al : list)

{

// init array line

line = new double[Integer.parseInt(String.valueOf(al.get(2)))];

// put the line

if (line != null)

{

if (longMap.entrySet() != null)

{

Arrays.fill(line, 1);

for (int i = Integer.parseInt(String.valueOf(al.get(3))); i <= Integer.parseInt(String.valueOf(al.get(4))); i++)

{

line[i] += (1 / Double.parseDouble(String.valueOf(al

.get(5))));

}

longMap.put(String.valueOf(al.get(0)), line);

}

else

{

for (Map.Entry<String, double[]> shotMap : longMap

.entrySet())

{

if (shotMap.getKey().equals(String.valueOf(al.get(0))))

{

for (int i = Integer.parseInt(String.valueOf(al

.get(3))); i <= Integer.parseInt(String

.valueOf(al.get(4))); i++)

{

shotMap.getValue()[i] += (1 / Double

.parseDouble(String.valueOf(al.get(5))));

longMap.put(shotMap.getKey(),

shotMap.getValue());

}

}

else

{

// function Arrays.fill: for(int

// j=0;j<line.length;j++){line[j]=1;}

Arrays.fill(line, 1);

for (int i = (Integer) al.get(3); i <= (Integer) al

.get(4); i++)

{

line[i] += (1 / Double.parseDouble(String

.valueOf(al.get(5))));

}

longMap.put(String.valueOf(al.get(0)), line);

}

}

}

}

}

return longMap;

}

}

package com.hadoop.hadoop;

import java.io.IOException;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class WReduce extends Reducer<Text, Text, Text, Text>

{

private Text result = null;

public void reduce(Text key, Text values, Context context)

throws IOException, InterruptedException

{

result = new Text();

result = values;

context.write(key, result);

}

}

package com.hadoop.global;

import java.util.ArrayList;

import java.util.Map;

public interface ICount

{

public Map count(ArrayList<ArrayList<Object>> list);

}

package com.hadoop.global;

import com.hadoop.service.impl.Video;

import com.hadoop.service.impl.VideoBehavior;

public interface IHandle

{

boolean handle(Video video, VideoBehavior behavior);

}

<configuration>

<property>

<name>

fs.defaultFS

</name>

<value>

hdfs://Master:9000

</value>

</property>

<property>

<name>

hadoop.tmp.dir

</name>

<value>

file:/usr/local/hadoop/tmp

</value>

<description>

Abase for other temporary directories.

</description>

</property>

</configuration>

<configuration>

<property>

<name>

dfs.namenode.secondary.http-address

</name>

<value>

Master:50090

</value>

</property>

<property>

<name>

dfs.replication

</name>

<value>

3

</value>

</property>

<property>

<name>

dfs.namenode.name.dir

</name>

<value>

file:/usr/local/hadoop/tmp/dfs/name

</value>

</property>

<property>

<name>

dfs.datanode.data.dir

</name>

<value>

file:/usr/local/hadoop/tmp/dfs/data

</value>

</property>

</configuration>

<configuration>

<property>

<name>

mapreduce.framework.name

</name>

<value>

Yarn

</value>

</property>

<property>

<name>

mapreduce.jobhistory.address

</name>

<value>

Master:10020

</value>

</property>

<property>

<name>

mapreduce.jobhistory.webapp.address

</name>

<value>

Master:19888

</value>

</property>

</configuration>

<configuration>

<property>

<name>

yarn.resourcemanager.hostname

</name>

<value>

Master

</value>

</property>

<property>

<name>

yarn.nodemanager.aux-services

</name>

<value>

mapreduce\_shuffle

</value>

</property>

</configuration>

package hadoop.video.service.impl;

public class Video

{

private String id;

private String url;

private double length;

private double startTime;

private double endTime;

private double speed;

public String getId()

{

return id;

}

public void setId(String id)

{

this.id = id;

}

public String getUrl()

{

return url;

}

public void setUrl(String url)

{

this.url = url;

}

public double getLength()

{

return length;

}

public void setLength(double length)

{

this.length = length;

}

public double getStartTime()

{

return startTime;

}

public void setStartTime(double startTime)

{

this.startTime = startTime;

}

public double getEndTime() {

return endTime;

}

public void setEndTime(double endTime) {

this.endTime = endTime;

}

public double getSpeed()

{

return speed;

}

public void setSpeed(double speed)

{

this.speed = speed;

}

}

package hadoop.video.servlet;

import hadoop.video.service.impl.Video;

import hadoop.video.util.GetLine;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@SuppressWarnings("serial")

public class ReturnServlet extends HttpServlet

{

protected void doGet(HttpServletRequest request,HttpServletResponse response) throws ServletException,IOException

{

// doGet

processRequest(request, response);

}

protected void doPost(HttpServletRequest request,HttpServletResponse response) throws ServletException, IOException

{

// doPost

processRequest(request, response);

}

protected void processRequest(HttpServletRequest request,HttpServletResponse response) throws ServletException, IOException

{

Video video = new Video();

int flag\_video = 0;

if(request.getParameter("url")!=null)

{

video.setUrl(request.getParameter("url"));

flag\_video++;

}

if(request.getParameter("id")!=null)

{

video.setId(request.getParameter("id"));

flag\_video++;

}

if(request.getParameter("length")!=null)

{

video.setLength(Double.parseDouble(request.getParameter("length")));

flag\_video++;

}

if(request.getParameter("startTime")!=null)

{

video.setStartTime(Double.parseDouble(request.getParameter("startTime")));

flag\_video++;

}

if(request.getParameter("endTime")!=null)

{

video.setEndTime(Double.parseDouble(request.getParameter("endTime")));

flag\_video++;

}

if(request.getParameter("speed")!=null)

{

video.setSpeed(Double.parseDouble(request.getParameter("speed")));

flag\_video++;

}

if(flag\_video==6)

{

GetLine.newInstase().getLine(video);

flag\_video=0;

}

}

}

package hadoop.video.util;

import java.util.Date;

import hadoop.video.service.impl.Video;

public class GetLine

{

public static GetLine newInstase()

{

return new GetLine();

}

public void getLine(Video video)

{

String s = new String(); s=video.getUrl()+"-"+video.getId()+"-"+video.getLength()+"-"+

(int)Math.round(video.getStartTime())+"-"+(int)Math.round(video.getEndTime())

+"-"+video.getSpeed()+"+";

String path = new String();

path = "C:/Users/84492/Desktop/test/test"+video.getId();

TextTransfer.newInstanse().save(path, s);

}

}

package hadoop.video.util;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

public class TextTransfer

{

public static TextTransfer newInstanse()

{

return new TextTransfer();

}

public String read(File file)

{

String result = readText(file);

return result;

}

public boolean save(String path, String content)

{

return writeText(path, content);

}

private String readText (File file)

{

StringBuilder result = new StringBuilder();

try

{

BufferedReader br = new BufferedReader(new FileReader(file));

String s =null;

while((s = br.readLine())!=null)

{

result.append(s+System.lineSeparator());

}

br.close();

}

catch(Exception e)

{

e.printStackTrace();

}

return result.toString();

}

private boolean writeText (String path, String content)

{

try

{

File textName = new File(path);

String result = new String();

if(!textName.exists())

{

boolean hasFile = textName.createNewFile();

if(hasFile)

{

result = content;

}

}

else

{

result = readText(textName).trim()+content;

}

try(FileWriter writer = new FileWriter(textName);BufferedWriter out = new BufferedWriter(writer))

{

out.write(result);

out.flush();

}

}

catch(IOException e)

{

e.printStackTrace();

}

return false;

}

}

public class MyWritable implements Writable

{

private int counter;

private long timestamp;

public void write(DataOutput out) throws IOException

{

out.writeInt(counter);

out.writeLong(timestamp);

}

public void readFields(DataInput in) throws IOException

{

counter = in.readInt();

timestamp = in.readLong();

}

public static MyWritable read(DataInput in) throws IOException

{

MyWritable w = new MyWritable();

w.readFields(in);

return w;

}

}

<?xml version="1.0" encoding="UTF-8"?>

<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee

http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd">

<welcome-file-list>

<welcome-file>

index.jsp

</welcome-file>

</welcome-file-list>

<servlet>

<servlet-name>

ReturnServlet

</servlet-name>

<servlet-class>

hadoop.video.servlet.ReturnServlet

</servlet-class>

<!-- <servlet-name>Test</servlet-name><servlet-class>test.Test</servlet-class> -->

</servlet>

<servlet-mapping>

<servlet-name>

ReturnServlet

</servlet-name>

<url-pattern>

/ReturnServlet

</url-pattern>

<!-- <servlet-name>

Test

</servlet-name>

<url-pattern>

/Test

</url-pattern> -->

</servlet-mapping>

</web-app>

package hadoop.video.servlet;

import hadoop.video.service.impl.Video;

import hadoop.video.util.GetLine;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class ReturnServlet extends HttpServlet

{

public ReturnServlet()

{

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException

{

this.processRequest(request, response);

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException

{

this.processRequest(request, response);

}

protected void processRequest(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException

{

Video video = new Video();

int flag\_video = 0;

if (request.getParameter("url") != null)

{

video.setUrl(request.getParameter("url"));

++flag\_video;

}

if (request.getParameter("id") != null)

{

video.setId(request.getParameter("id"));

++flag\_video;

}

if (request.getParameter("length") != null)

{

video.setLength(Double.parseDouble(request.getParameter("length")));

++flag\_video;

}

if (request.getParameter("startTime") != null)

{

video.setStartTime(Double.parseDouble(request.getParameter("startTime")));

++flag\_video;

}

if (request.getParameter("endTime") != null)

{

video.setEndTime(Double.parseDouble(request.getParameter("endTime")));

++flag\_video;

}

if (request.getParameter("speed") != null)

{

video.setSpeed(Double.parseDouble(request.getParameter("speed")));

++flag\_video;

}

if (flag\_video == 6)

{

GetLine.newInstase().getLine(video);

boolean var5 = false;

}

}

}

package com.hadoop.hadoop;

import java.io.IOException;

public class Start

{

public Start()

{

}

public static void main(String[] args) throws ClassNotFoundException, IOException, InterruptedException

{

WJob.newInstanse().job();

System.out.println("1");

}

}

package com.hadoop.service.impl;

public class VideoBehavior

{

private int startTime;

private int endTime;

private double speed;

public VideoBehavior()

{

}

public int getStartTime()

{

return this.startTime;

}

public void setStartTime(int startTime)

{

this.startTime = startTime;

}

public int getEndTime()

{

return this.endTime;

}

public void setEndTime(int endTime)

{

this.endTime = endTime;

}

public double getSpeed()

{

return this.speed;

}

public void setSpeed(double speed)

{

this.speed = speed;

}

}

package com.hadoop.util;

public class HadoopFactory

{

public HadoopFactory()

{

}

public static HadoopFactory newInstanse()

{

return new HadoopFactory();

}

private Object execute(Object[] params)

{

return false;

}

private boolean exec(Object[] params)

{

Object obj = this.execute(params);

if (obj instanceof Boolean)

{

Boolean b = (Boolean)obj;

return b;

}

else

{

return false;

}

}

}

package com.hadoop.util;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.Path;

public class HDFS\_Factory

{

public HDFS\_Factory()

{

}

public static HDFS\_Factory newIstanse()

{

return new HDFS\_Factory();

}

public void uplode(String inputPath, String outputPath) {

Configuration conf = new Configuration();

conf.set("fs.defaultFS", "hdfs://master:9000");

try

{

Path localPath = new Path(inputPath);

Path hdfsPath = new Path(outputPath);

FileSystem fs = FileSystem.newInstance(conf);

fs.copyFromLocalFile(localPath, hdfsPath);

}

catch (Exception var7)

{

var7.printStackTrace();

}

}

public void downlode(String inputPath, String outputPath) {

Configuration conf = new Configuration();

conf.set("fs.defaultFS", "hdfs://master:9000");

try {

Path localPath = new Path(inputPath);

Path hdfsPath = new Path(outputPath);

FileSystem fs = FileSystem.newInstance(conf);

fs.copyFromLocalFile(hdfsPath, localPath);

} catch (Exception var7) {

var7.printStackTrace();

}

}

}