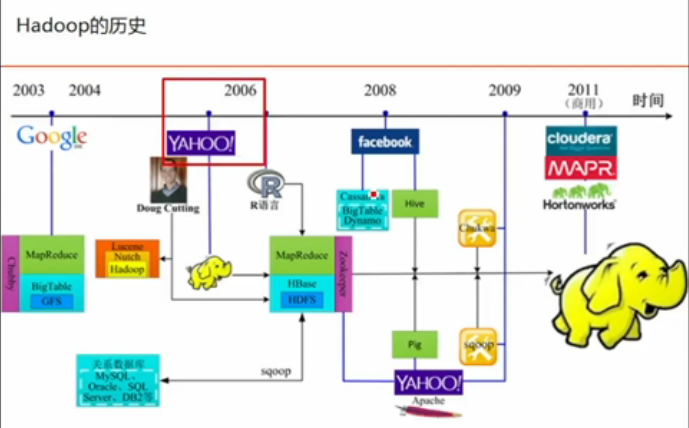
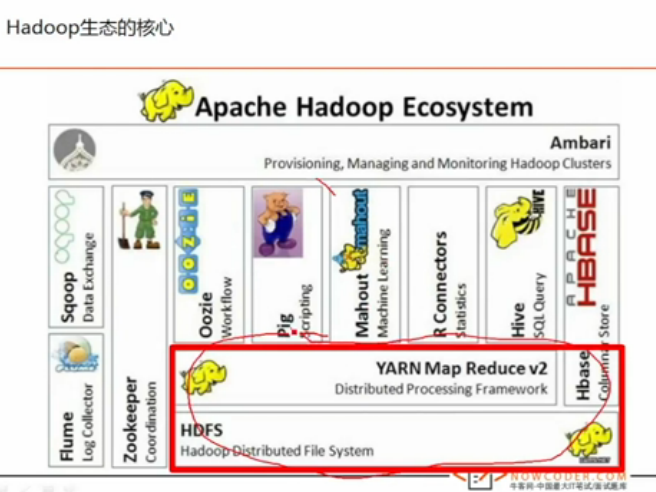
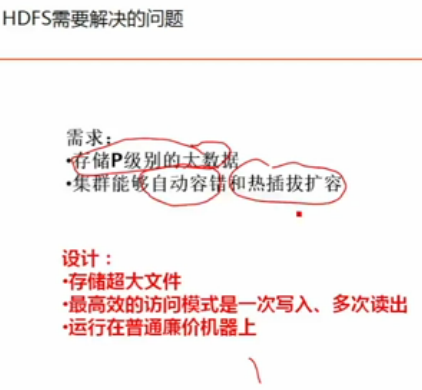


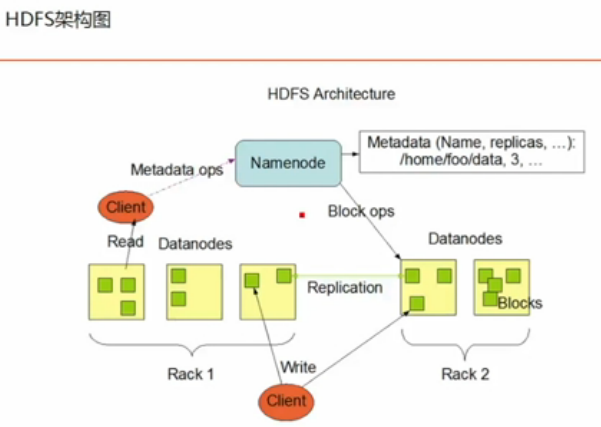
主要讲红色部分，特征的抽取、处理用到的一些工具。

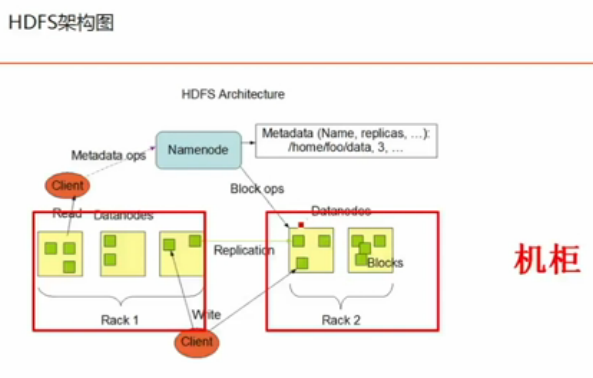


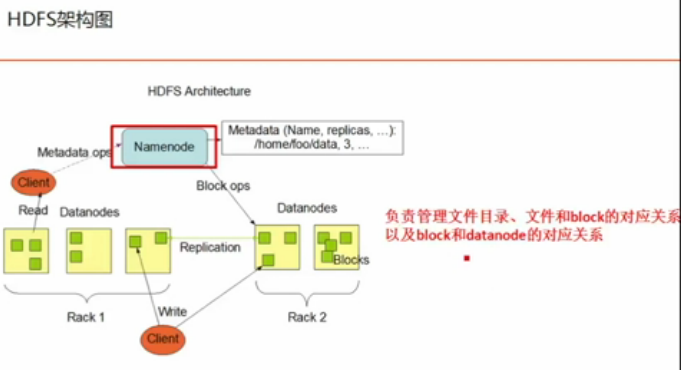


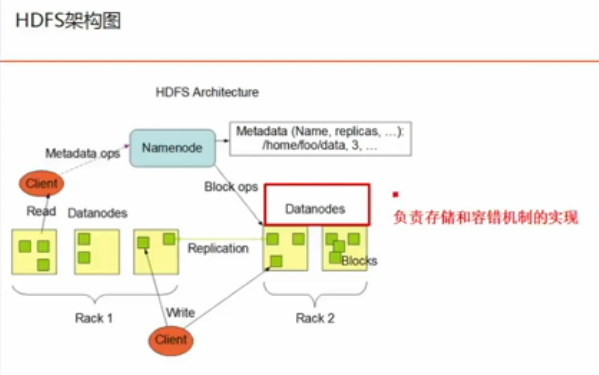


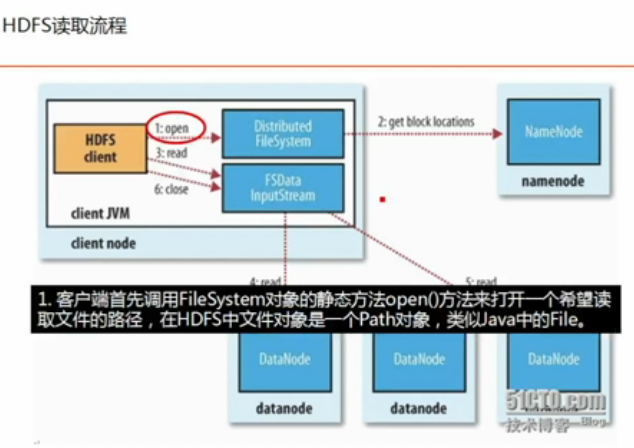


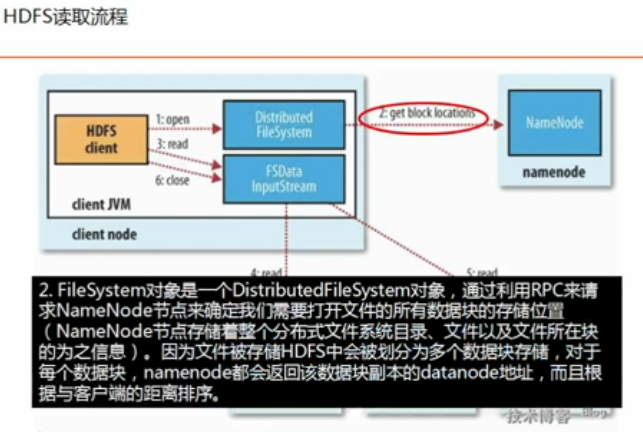


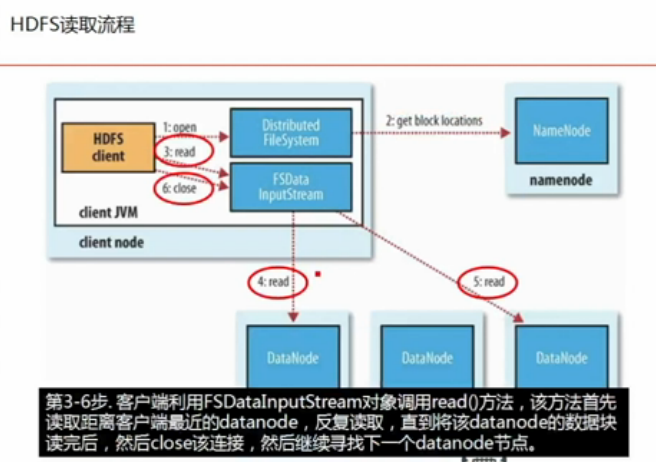


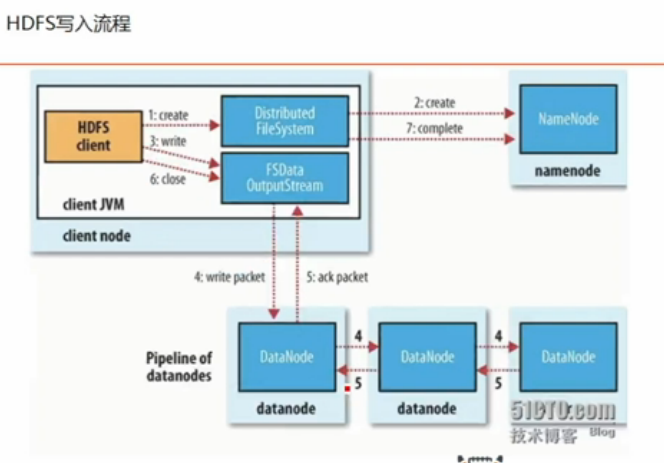


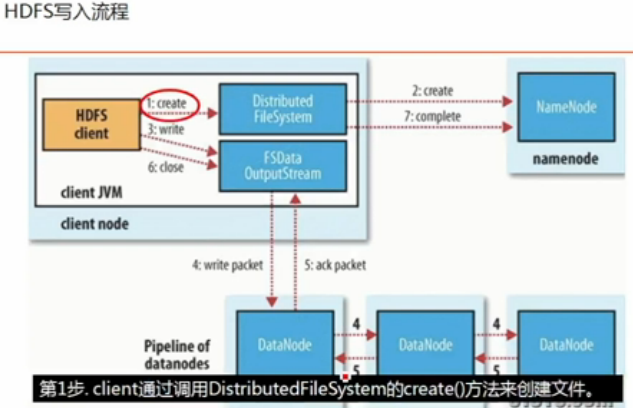


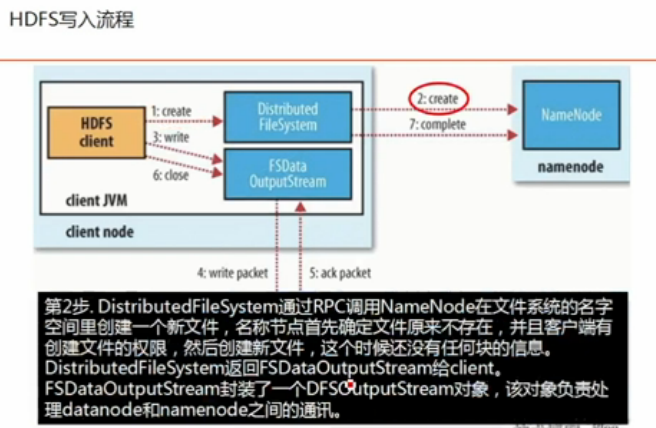


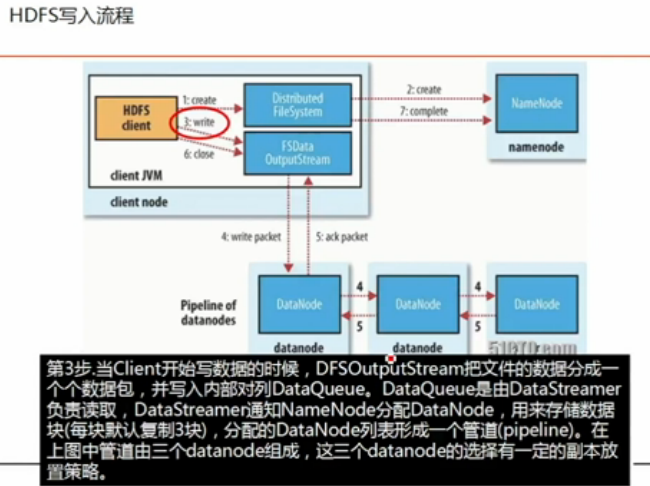


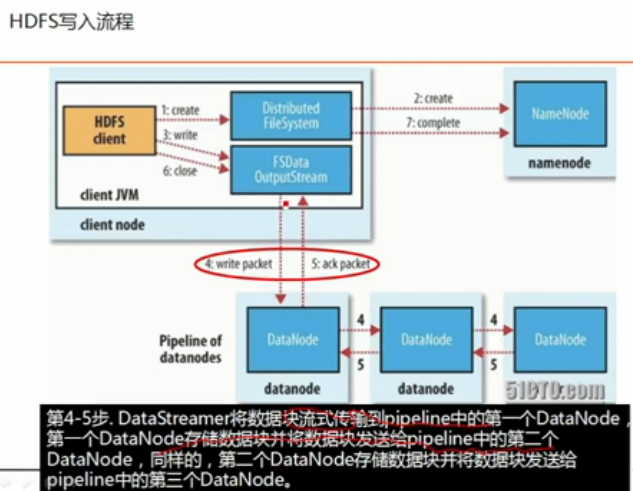


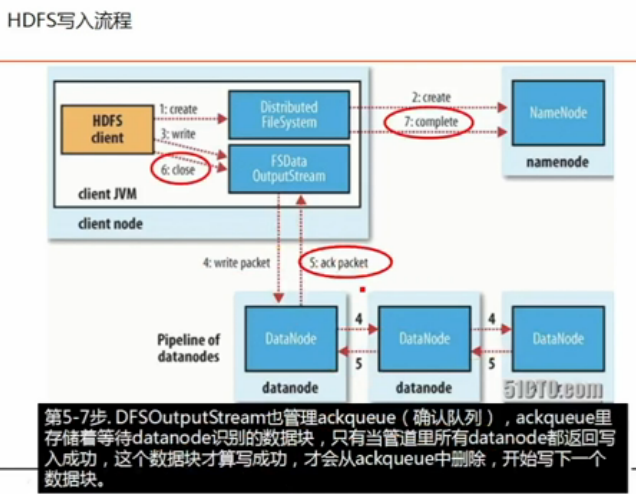








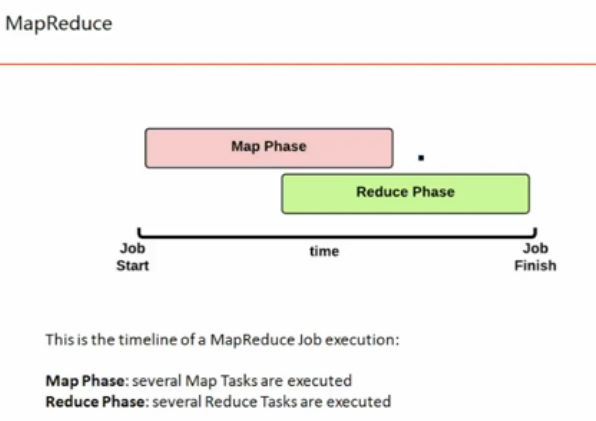


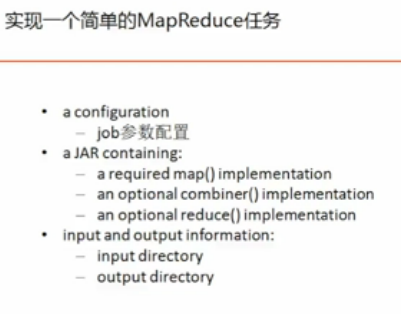


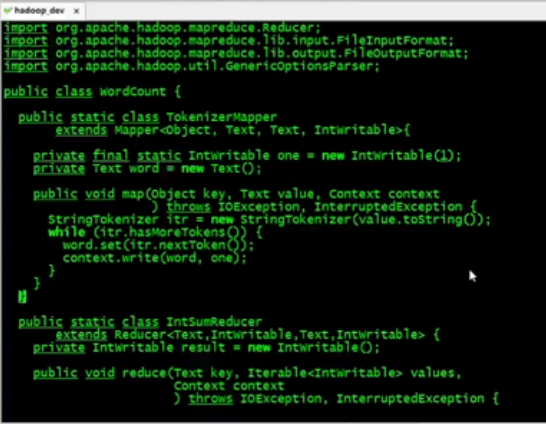


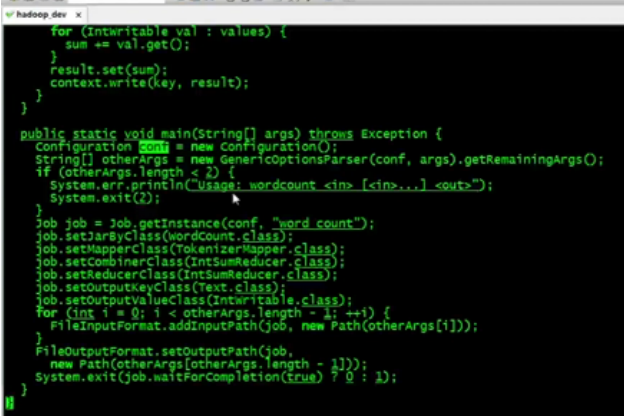
Hadoop fs –cp是单进程的命令

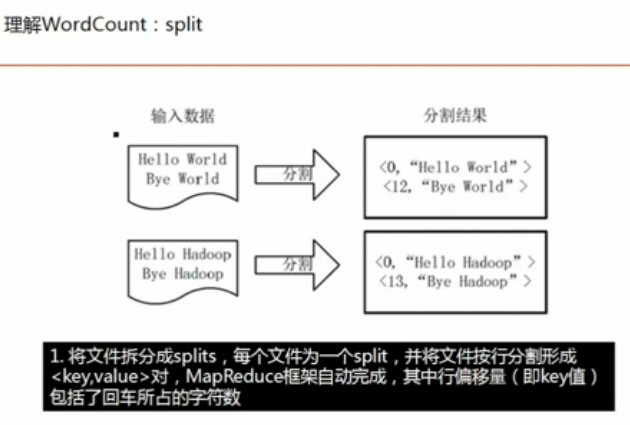
Hadoop distcp 会将任务转成mapreduce任务，一次性起上千个进程去做拷贝

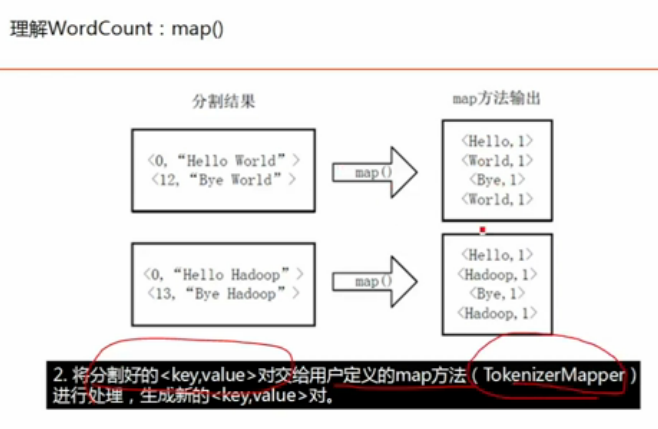


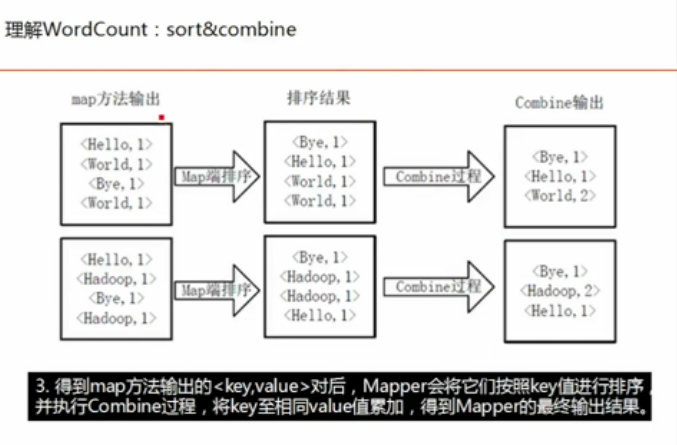


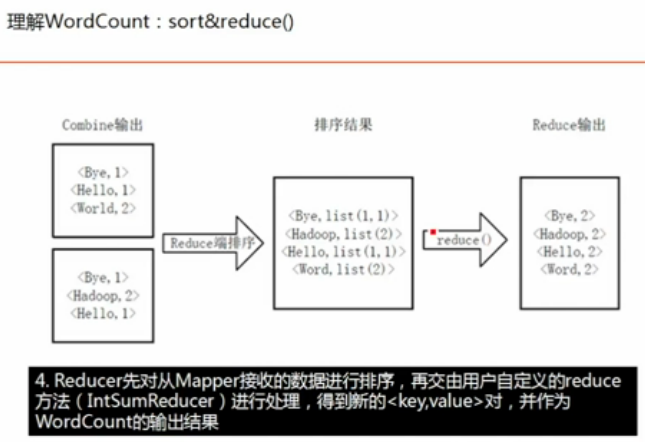


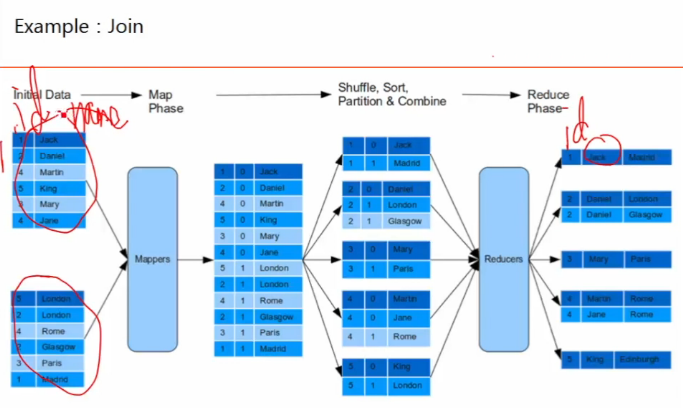


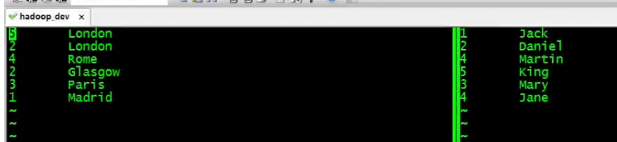


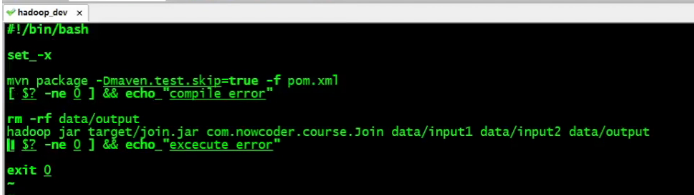




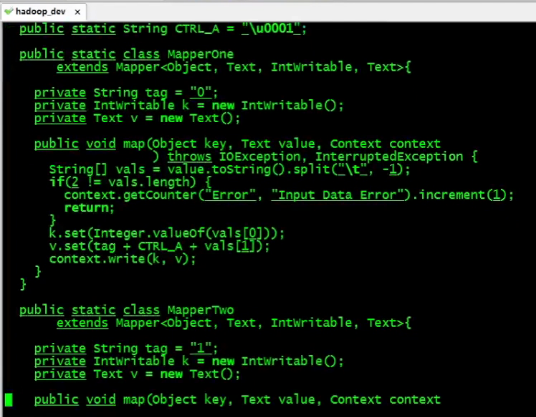


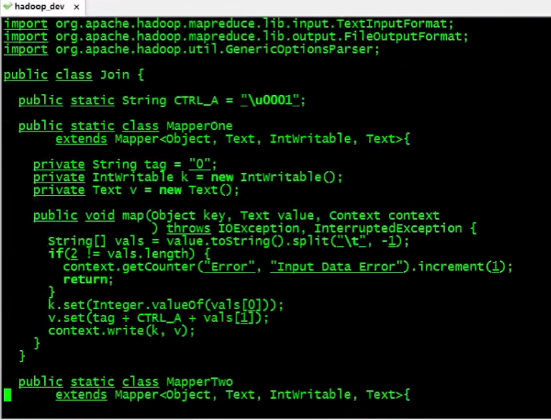


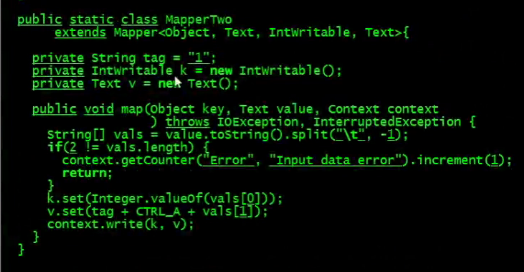


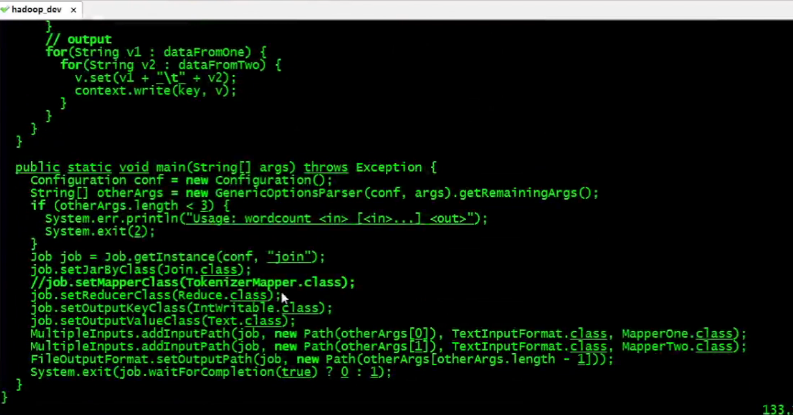


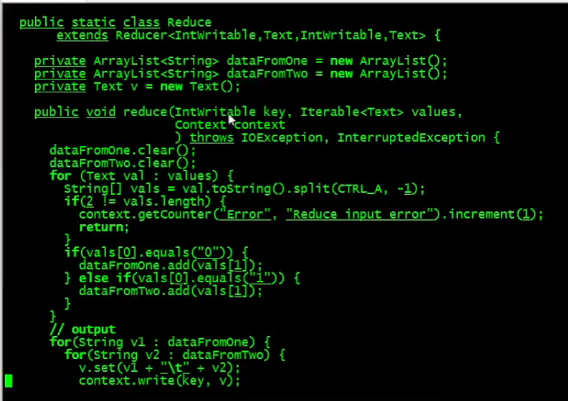


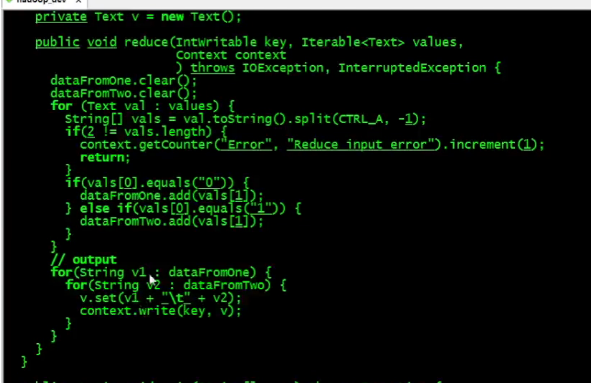


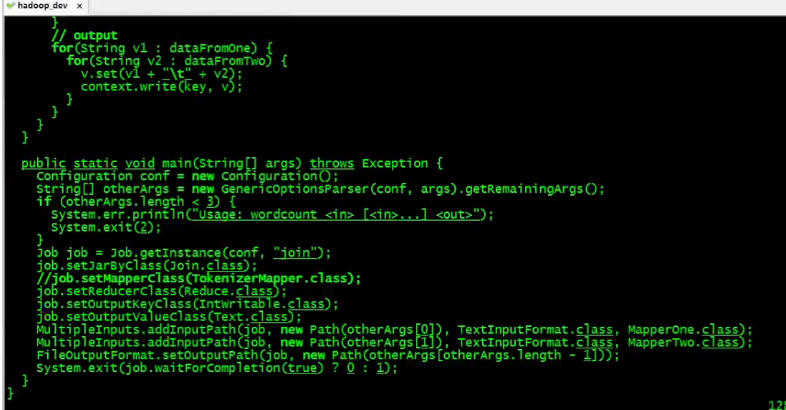


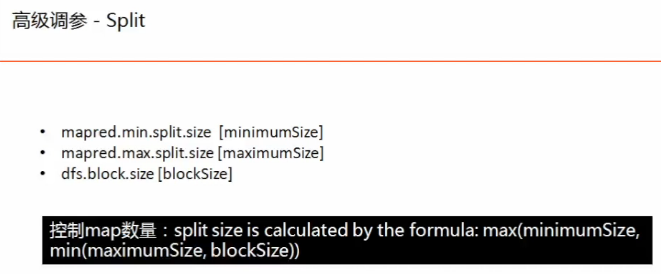


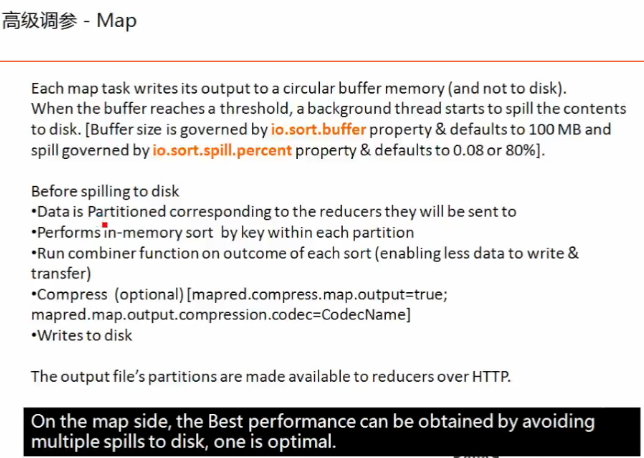


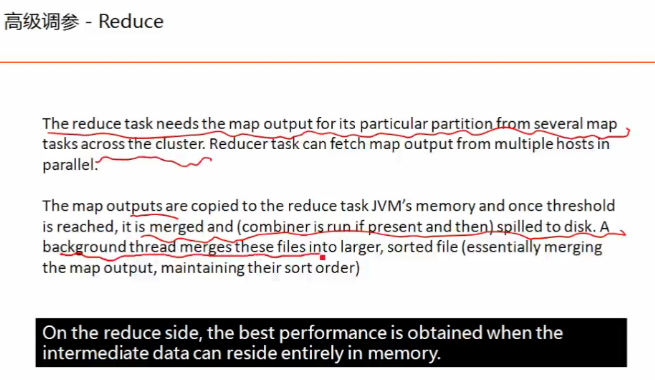


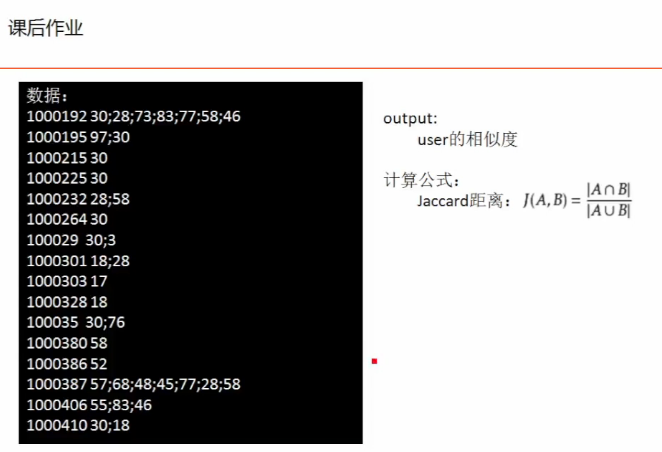












输入数据：

1000033 30

1000035 30

1000038 30

100006 3;30

1000062 30

1000079 30;45;28;46;83

1000094 30

1000095 97

1000104 30

1000105 28

1000122 62;52

1000153 30

1000158 28

1000176 30

1000183 30

1000192 30;28;73;83;77;58;46

1000195 97;30

1000215 30

1000225 30

1000232 28;58

1000264 30

100029 30;3

1000301 18;28

1000303 17

1000328 18

100035 30;76

1000380 58

1000386 52

1000387 57;68;48;45;77;28;58

1000406 55;83;46

1000410 30;18

1000419 30

1000427 30

1000432 30

1000433 30;58

1000438 30

1000439 30

1000457 23;28

1000461 28;77

1000481 28

1000485 30

1000489 18;30

1000527 30;28

1000530 62;28

1000546 30

1000554 30

1000559 30

1000561 30

100057 28

1000571 30

1000584 18

1000596 83;24;30

1000597 8

1000602 30

1000628 30

1000634 33;88;58;46;16

1000670 81;30

1000678 30

1000684 30

10007 35;30

1000710 28;30;58

1000721 8;97

1000740 30

1000774 83;30

1000779 77;28;59;30

1000800 77

1000804 8

1000808 30

1000815 30

1000840 30;18

1000851 45;58;83;78;46;30;38

1000868 8

1000880 28;83;30

1000890 30

1000892 30;28

1000894 97;30

1000904 83;28;30;44

100093 30;28;83

1000943 30

1000954 30

1000959 77

1000965 22;28;83;30

1000976 30

100098 30

1000987 30

1000991 77

1000992 30

1001004 8;79;30;58;28

1001006 30

1001046 83;28;30

1001057 30;17

1001072 30

1001105 28;18

1001129 84;14;90;91;29;21;9;44;24;89;8;42;41;40;25;37;30;16;97;52;62;56;80;83;36;26;73;64;32;27;67;65;79;87;17

1001140 58;44;30

1001144 30

1001147 30

1001154 30

1001163 30

1001167 18;8

利用3个mr处理

Driver代码：

package com.nowcoder.course;

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.Text;

import org.apache.hadoop.mapreduce.Job;

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

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

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

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

import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;

public class Lesson1 extends Configured implements Tool {

@Override

public int run(String[] args) throws Exception {

// 1.step 1

Job job1 = Job.getInstance(getConf(), "step1");

job1.setJarByClass(Lesson1.class);

job1.setMapperClass(Mapper1.class);

job1.setReducerClass(Reducer1.class);

job1.setOutputKeyClass(Text.class);

job1.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(job1, new Path(args[0]));

FileOutputFormat.setOutputPath(job1, new Path(args[1] + "/step1"));

job1.waitForCompletion(true);

// 2.step 2

Job job2 = Job.getInstance(getConf(), "step2");

job2.setJarByClass(Lesson1.class);

job2.setReducerClass(Reducer2.class);

job2.setOutputKeyClass(IntWritable.class);

job2.setOutputValueClass(Text.class);

MultipleInputs.addInputPath(job2, new Path(args[1] + "/step1"), TextInputFormat.class, Mapper21.class);

MultipleInputs.addInputPath(job2, new Path(args[0]), TextInputFormat.class, Mapper22.class);

FileOutputFormat.setOutputPath(job2, new Path(args[1] + "/step2"));

job2.waitForCompletion(true);

// 3.step 3

Job job3 = Job.getInstance(getConf(), "step3");

job3.setJarByClass(Lesson1.class);

job3.setReducerClass(Reducer3.class);

job3.setOutputKeyClass(IntWritable.class);

job3.setOutputValueClass(Text.class);

MultipleInputs.addInputPath(job3, new Path(args[1] + "/step2"), TextInputFormat.class, Mapper31.class);

MultipleInputs.addInputPath(job3, new Path(args[0]), TextInputFormat.class, Mapper22.class);

FileOutputFormat.setOutputPath(job3, new Path(args[1] + "/step3"));

job3.waitForCompletion(true);

return 0;

}

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

Configuration conf = new Configuration();

int res = ToolRunner.run(conf, new Lesson1(), args);

System.exit(res);

}

}

Mr1代码：

package com.nowcoder.course;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class Mapper1 extends Mapper<Object, Text, Text, IntWritable>{

private final static IntWritable one = new IntWritable(1);

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

String[] parts = line.toString().split("\t");

//String movie = parts[0];

String[] userIds = parts[1].split(";");

for(int i = 0; i < userIds.length; i++){

for(int j = 0; j < userIds.length; j++){

if(Long.parseLong(userIds[i]) < Long.parseLong(userIds[j])){

context.write(new Text(userIds[i] + "," + userIds[j]), one);

}

}

}

}

}

package com.nowcoder.course;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class Reducer1 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 sum = 0;

for (IntWritable val : values) {

sum += val.get();

}

result.set(sum);

context.write(key, result);

}

}

Mr2代码：

package com.nowcoder.course;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class Mapper21 extends Mapper<Object, Text, IntWritable, Text> {

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

String[] parts = line.toString().split("\t");

if (parts.length < 2) {

return;

}

String[] userIds = parts[0].split(",");

if (userIds.length < 2) {

return;

}

context.write(new IntWritable(Integer.parseInt(userIds[0])), new Text(userIds[1] + ","

+ parts[1]));

}

}

package com.nowcoder.course;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class Mapper22 extends Mapper<Object, Text, IntWritable, Text> {

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

String[] parts = line.toString().split("\t");

if(parts.length < 2){

return;

}

String[] userIds = parts[1].split(";");

for (int i = 0; i < userIds.length; i++) {

context.write(new IntWritable(Integer.parseInt(userIds[i])), new Text("1"));

}

}

}

package com.nowcoder.course;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

public class Reducer2 extends Reducer<IntWritable, Text, IntWritable, Text>{

public void reduce(IntWritable userId, Iterable<Text> values, Context context) throws IOException,

InterruptedException {

int sum = 0;

List<String> neighbors = new ArrayList<String>();

for (Text val : values) {

String valStr = val.toString();

if(valStr.contains(",")){

neighbors.add(valStr);

} else {

sum += Integer.parseInt(valStr);

}

}

StringBuffer result = new StringBuffer();

for (String neighbor : neighbors) {

if(result.length() > 0){

result.append(";");

}

result.append(neighbor);

result.append(",");

result.append(sum);

}

context.write(userId, new Text(result.toString()));

}

}

Mr3代码：

package com.nowcoder.course;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class Mapper31 extends Mapper<Object, Text, IntWritable, Text> {

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

String[] parts = line.toString().split("\t");

if (parts.length < 2) {

return;

}

String[] userInfos = parts[1].split(";");

for (String userInfo : userInfos) {

String[] infos = userInfo.split(",");

context.write(new IntWritable(Integer.parseInt(infos[0])), new Text(parts[0] + ","

+ infos[1] + "," + infos[2]));

}

}

}

package com.nowcoder.course;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

public class Reducer3 extends Reducer<IntWritable, Text, Text, Text> {

public void reduce(IntWritable userId, Iterable<Text> values, Context context)

throws IOException, InterruptedException {

int sum = 0;

List<String> neighbors = new ArrayList<String>();

for (Text val : values) {

String valStr = val.toString();

if (valStr.contains(",")) {

neighbors.add(valStr);

} else {

sum += Integer.parseInt(valStr);

}

}

for (String neighbor : neighbors) {

String[] parts = neighbor.split(",");

if (parts.length < 3) {

continue;

}

//jaccard

double corr =

Double.parseDouble(parts[1])

/ (Double.parseDouble(parts[2]) + sum - Double.parseDouble(parts[1]));

context.write(new Text(parts[0] + "," + userId), new Text(corr + ""));

}

}

}