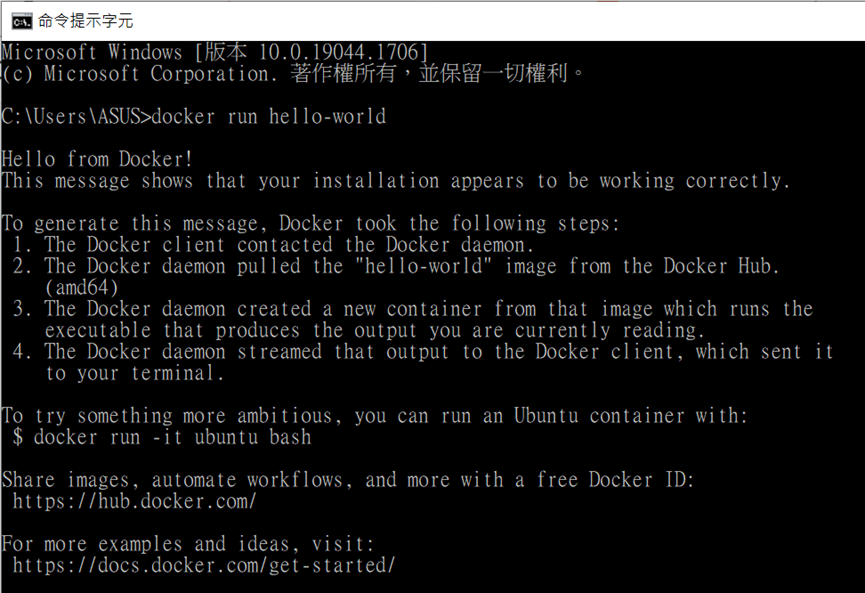
**分布式计算课程实验报告**

学号：10930130448 姓名：吕思勤

一、实验环境配置：hadoop spark

根据老师的安装手册安装即可，安装成功后得到以下界面



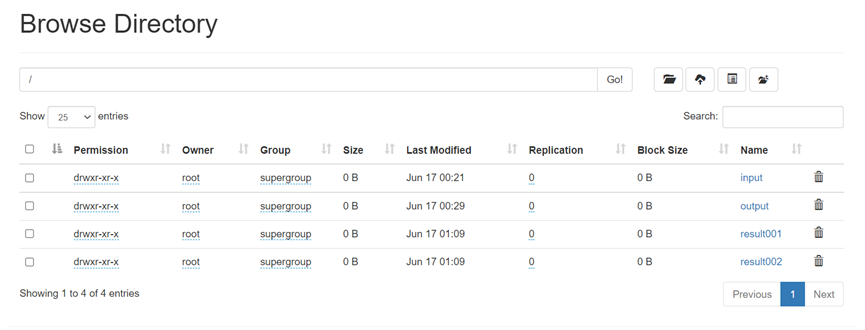


启动实验环境:

一張含有 文字 的圖片

自動產生的描述

HDFS的Web监控服务浏览器网址http://localhost9870/



二、 实验目的

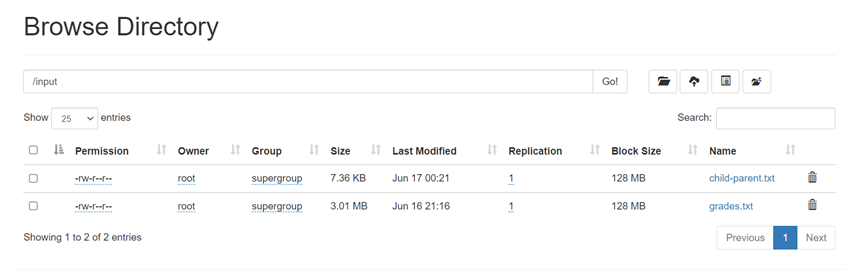
1. 学习基于MapReduce框架的分布式计算程序设计方法。

2. 学习基于Spark框架的分布式计算程序设计方法。

三、 实验题目

准备事件：先将需要处理的文件上传到浏览器

input ( child-parent.txt grades.txt )



题目1

输入文件为学生成绩信息，包含了必修课与选修课成绩，格式如下：

班级1, 姓名1, 科目1, 必修, 成绩1 <br> （注：<br> 为换行符）

班级2, 姓名2, 科目1, 必修, 成绩2 <br>

班级1, 姓名1, 科目2, 选修，成绩3 <br>

………., ………, ………, ……… <br>

编写两个Hadoop平台上的MapReduce程序，分别实现如下功能：

1. 计算每个学生必修课的平均成绩。

* 编写map代码

从数据中选择出**必修**的信息条：key:name value:score

eg: 170315班,史伦泰,计算机图形学,选修,82 史伦泰 82 160301班,邱丰琪,计算机图形学,选修,90 邱丰琪 90 ........

@Override

protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text, LongWritable>.Context context)

throws IOException, InterruptedException {

String line = value.toString();

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

String name = splited[1];//名字

String kind = splited[3];//必修or选修

Integer score = Integer.parseInt(splited[4]);//分数

if(kind.equals("必修")){

context.write(new Text(name), new LongWritable(score));

}

}

* 编写reduce代码

将新k2，v2-----> k3,v3

eg 张哥<100,90,80> ----> 求平均数 张哥 90=（100+90+80）/3

@Override

protected void reduce(Text k2, Iterable<LongWritable> v2s,

Reducer<Text, LongWritable, Text, DoubleWritable>.Context context) throws IOException, InterruptedException {

long count = 0L;

long sum = 0L;

double aver;

for (LongWritable v2 : v2s) {//遍历求和

count += v2.get();

sum = sum + 1;

}

aver = count\*1.0/sum;//防止结果取整数

DoubleWritable v3 = new DoubleWritable(aver);

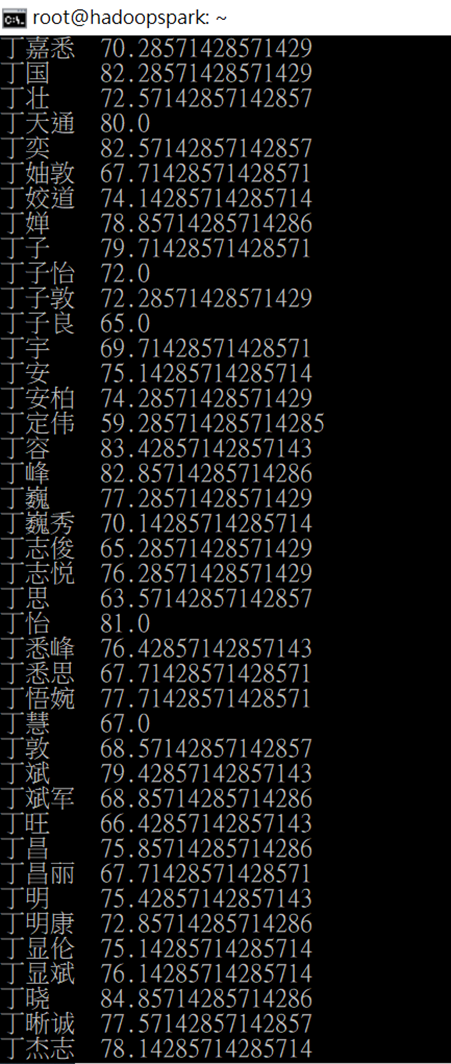
context.write(k2, v3);

}

* 打包jar包--->11.jar（先clean再package）
* 运行：

hadoop jar ./target/11.jar com.org.xidian.MapReduceWordCountDemo /input/grades.txt /output/output11

* 运行结果



1. 按科目统计每个班的平均成绩。

* 编写map代码

从数据中选择出信息条：key:subject+"\t"+banji+"\t" value:score

eg: 170315班,史伦泰,计算机图形学,选修,82 计算机图形学 170315班 82 160301班,邱丰琪,计算机图形学,选修,90 计算机图形学 160301班 90 ........

@Override

protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text, LongWritable>.Context context)

throws IOException, InterruptedException {

String line = value.toString();

String[] splited = line.split(",");//分离

String banji = splited[0];

String subject = splited[2];

String feature = subject+"\t"+banji+"\t";

Integer score = Integer.parseInt(splited[4]);

context.write(new Text(feature), new LongWritable(score));

}

* 编写reduce代码

将新k2，v2-----> k3,v3

eg 数学 11班<100,90,80> ----> 求平均数 数学 11班 90=（100+90+80）/3

@Override

protected void reduce(Text k2, Iterable<LongWritable> v2s,

Reducer<Text, LongWritable, Text, DoubleWritable>.Context context) throws IOException, InterruptedException {

long count = 0L;

long sum = 0L;

double aver;

for (LongWritable v2 : v2s) {//遍历相加

count += v2.get();

sum = sum + 1;

}

aver = count\*1.0 /sum;//取平均

DoubleWritable v3 = new DoubleWritable(aver);

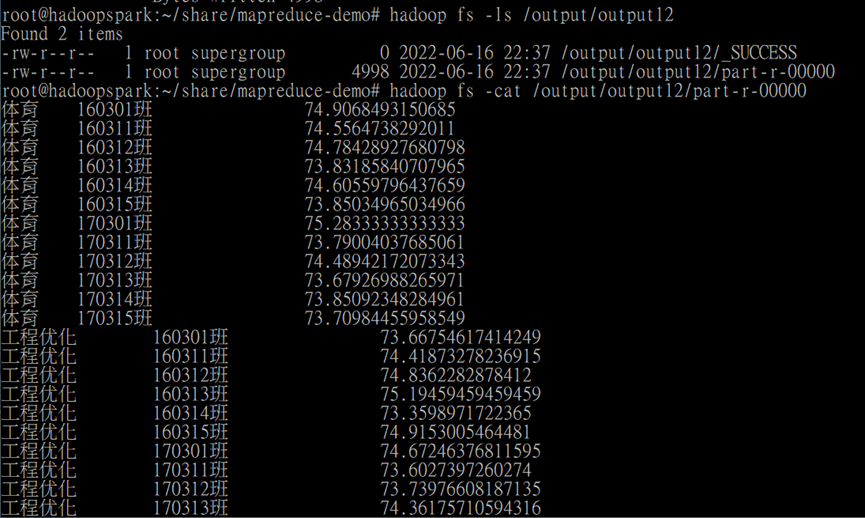
context.write(k2, v3);

}

* 打包jar包--->12.jar（先clean再package）
* 运行：

hadoop jar ./target/12.jar com.org.xidian.MapReduceWordCountDemo /input/grades.txt /output/output12

* 运行结果



题目2

输入文件的每一行为具有父子/父女/母子/母女/关系的一对人名，例如：

Tim, Andy <br>

Harry, Alice <br>

Mark, Louis <br>

Andy, Joseph <br>

……….., ………… <br>

假定不会出现重名现象。

编写Hadoop平台上的MapReduce程序，找出所有具有grandchild-grandparent关系的人名组。

* 编写map代码

从数据中选择出信息条： key:child value:"<"+parent

​ key:parent value:">"+child

eg: 儿a，父a1 儿a <父a1 父a1 >儿a ........

@Override

protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text, Text>.Context context)

throws IOException, InterruptedException {

String line = value.toString();

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

String child = splited[0];

String parent = splited[1];

context.write(new Text(child), new Text("<"+parent));

context.write(new Text(parent), new Text(">"+child));

}

* 编写reduce代码

将新k2，v2-----> k3,v3

eg 儿a< <a1,<a2,>a3,<a4,>a5 > 转化1 a1,a2,a4 a3,a5 转化2 a3-a1 a3-a2 a3-a4 a5-a1 a5-a2 a5-a4

@Override

protected void reduce(Text k2, Iterable<Text> v2s,

Reducer<Text, Text, Text, Text>.Context context) throws IOException, InterruptedException {

ArrayList<Text> grandparent = new ArrayList<Text>();

ArrayList<Text> grandchild = new ArrayList<Text>();

for (Text v2 : v2s) {//对各个values中的值进行处理

String ss = v2.toString();

String s = ss.substring(0, 1);

if (s.equals("<")) {

grandchild.add(new Text(ss.substring(1)));

} else {

grandparent.add(new Text(ss.substring(1)));

}

}

//再将grandparent与grandchild中的东西，一一对应输出。

for (int i = 0; i < grandchild.size(); i++) {

for (int j = 0; j < grandparent.size(); j++) {

context.write(grandchild.get(i), grandparent.get(j));

}

}

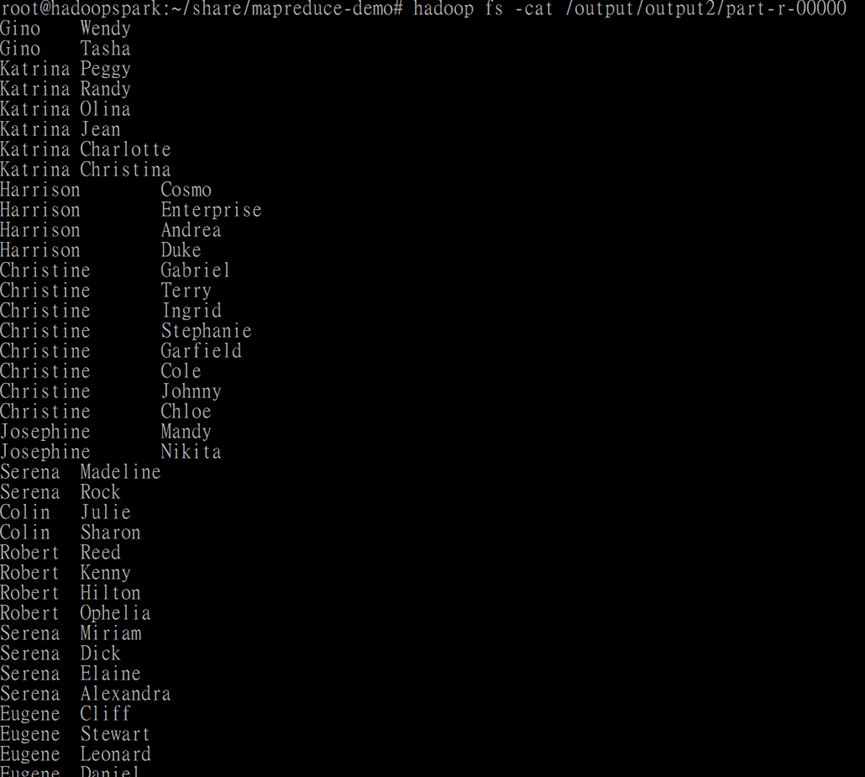
}

* 打包jar包--->2.jar（先clean再package）
* 运行：

hadoop jar ./target/2.jar com.org.xidian.MapReduceWordCountDemo /input/child-parent.txt /output/output2

* 运行结果

hadoop fs -cat /output/output2 /part-r-00000



题目3

输入文件为学生成绩信息，包含了必修课与选修课成绩，格式如下：

班级1, 姓名1, 科目1, 必修, 成绩1 <br> （注：<br> 为换行符）

班级2, 姓名2, 科目1, 必修, 成绩2 <br>

班级1, 姓名1, 科目2, 选修，成绩3 <br>

………., ………, ………, ……… <br>

编写一个Spark程序，同时实现如下功能：

1. 计算每个学生必修课的平均成绩。

2. 统计学生必修课平均成绩在：90~100,80~89,70~79,60~69和60分以下这5个分数段的人数。

* 先读入数据

from pyspark import SparkConf,SparkContext

conf = SparkConf().setMaster("local").setAppName("spark")

sc = SparkContext(conf=conf)

textData = sc.textFile("/input/grades.txt")

* 筛选必修课信息条，并选择学生名字和分数列

def select(x):

m = x.split(",")

name = m[1]

score = int(m[4])

return (name, score)

lines = textData.filter(lambda line : "必修" in line).map(lambda x:select(x))

* 每个学生必修课的平均成绩

avgData = lines.mapValues(lambda x : (x,1)).reduceByKey(lambda x,y : (x[0]+y[0],x[1] + y[1])).mapValues(lambda x : int(x[0] / x[1]))

avgData.saveAsTextFile("/result001")

* 统计分数段人数

def sort(x):

if(x>=90 and x<=100):

return ("A", 1)

if(x>=80 and x<=89):

return ("B", 1)

if(x>=70 and x<=79):

return ("C", 1)

if(x>=60 and x<=69):

return ("D", 1)

if(x<60):

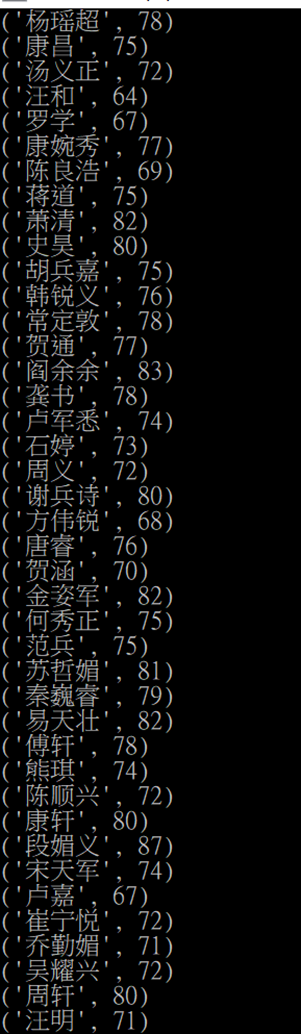
return ("E", 1)

fData = avgData.map(lambda x:sort(x[1])).reduceByKey(lambda x,y : (x+y))

fData.saveAsTextFile("/result002")

* 运行：spark-submit spark.py
* 运行结果

hadoop fs -cat /result001/part-00000



hadoop fs -cat /result002/part-00000

