在 Eclipse 下的 Hadoop0.17.0(MapReduce)作业指导书

杨来 dillony@gmail.com

1. 导言

Hadoop 为分布式编程提供了一个理想的平台，普通的程序员只要理解了分布式的特点，就可以轻易地实现分布式计算，而不需要理解分布式的细节。本文用实例讲解了在 Eclipse 下，使用 Hadoop 对数据集的统计度量的实现过程。

1.1. 预备条件

熟悉 java 编程和 Eclipse，了解 Hadoop，MapReduce，了解 linux；

已经建立了 Hadoop-0.17.0 环境；本文中 master 的 ip 为 10.63.0.60，用户为 root；

数据为实数。

2. 作业目标

2.1. 实现常用的统计度量

具体是对数据集的每个属性进行最小值、最大值、计数、求和、均值、方差等统计，相对应于SQL操作中的min(), max(), cnt(), sum(), avg(), std()（参见6统计知识）；

2.2. 性能要接近线性加速比

由于最小值、最大值、计数、求和 等操作为分布度量，必须遍历每个数据，算法复杂度为 O(n)。均值、方差等代数度量 依赖于计数、求和，复杂度为 O(1)，需要最后执行。

3. 作业方案

3.1. 处理对象

我们希望对.csv 文件格式的实型数据，如下表，进行列向的统计。各种小数形式、科学计数表示的实型数据均应能识别。

9.0000000e+000	1.0000000e+000	2.0000000e+000	4.0000000e+000
3.0000000e+000	8.0000000e+000	4.0000000e+000	8.0000000e+000
6.0000000e+000	5.0000000e+000	9.0000000e+000	1.0000000e+000
5.0000000e+000	6.0000000e+000	9.0000000e+000	2.0000000e+000
9.0000000e+000	8.0000000e+000	4.0000000e+000	2.0000000e+000
7.0000000e+000	9.0000000e+000	9.0000000e+000	2.0000000e+000
5.0000000e+000	7.0000000e+000	1.0000000e+000	6.0000000e+000

3.2. 期望结果

理想的输出结果应该是如下：第一行为最小值、第二行为最大值、第三行为求和、第四行为平方和、第五行为均值、第六行为方差计数。列的数目与输入数据一致。输出的格式为标准的.csv文件，其以逗号为分隔符。

3.0,1.0,1.0,1.0
9.0,9.0,9.0,8.0
44.0,44.0,38.0,25.0
306.0,320.0,280.0,129.0
6.285714,6.285714,5.428571,3.5714285
2.2146692,2.6903703,3.505098,2.572751

4. 作业流程

4.1. MapReduce 插件

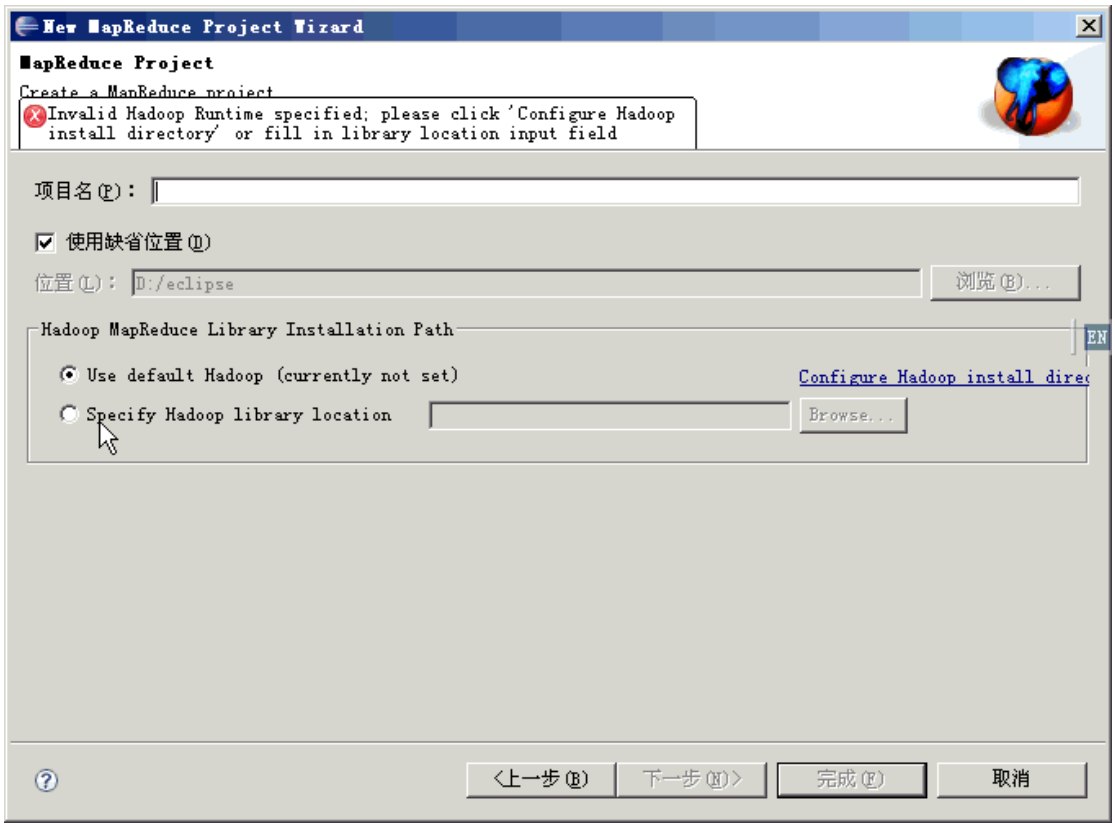
下载和安装

在 http://www.ibm.com/developerworks/cn/opensource/os-cn-hadoop2/mapreduce_plugin.zip 下载 MapReduce 插件，并拷贝到 Eclipse 的 plugin。地址可能会过期失效，尝试查找。可能需要填写某些资料和同意其声明。

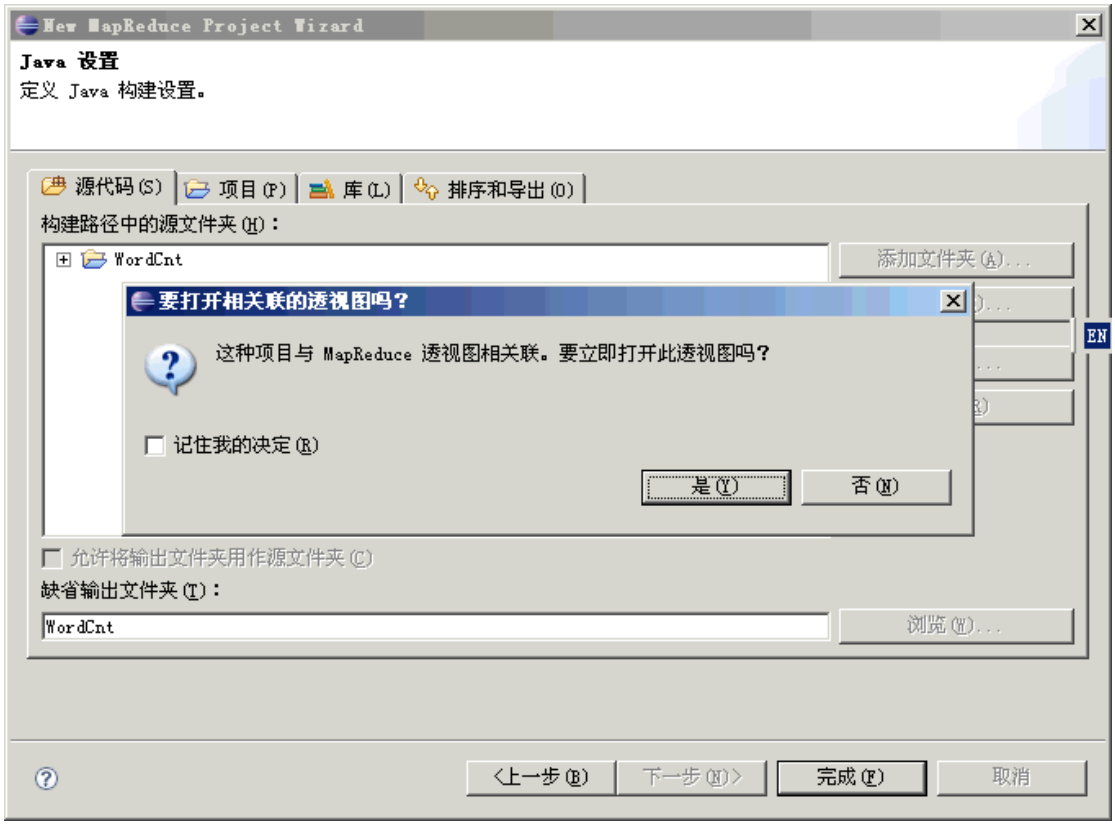
用 MapReduce 插件编程



如图，在“新建”时，会出现“选择向导”，选择“MapReduce Project”，点击“下一步”；



对话框会提示没有 Hadoop 的运行库，也就是外部 JAR；选择“Specify Hadoop library location”，给出“hadoop-xxxxx.core.jar”所在的目录；“xxxxx”表示版本。



MapReduce 透视图必须要选“是”，否则后续流程无法进行。

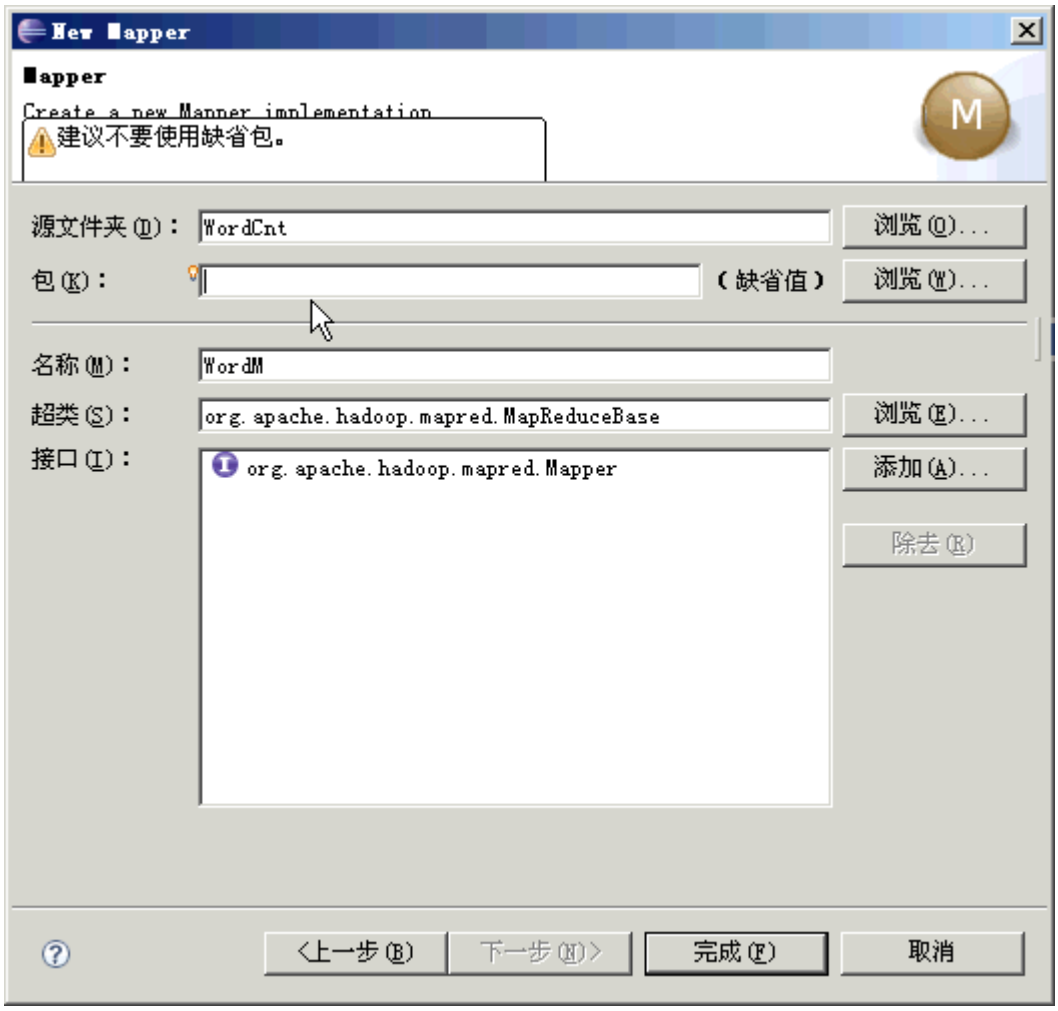
新建 Mapper 类



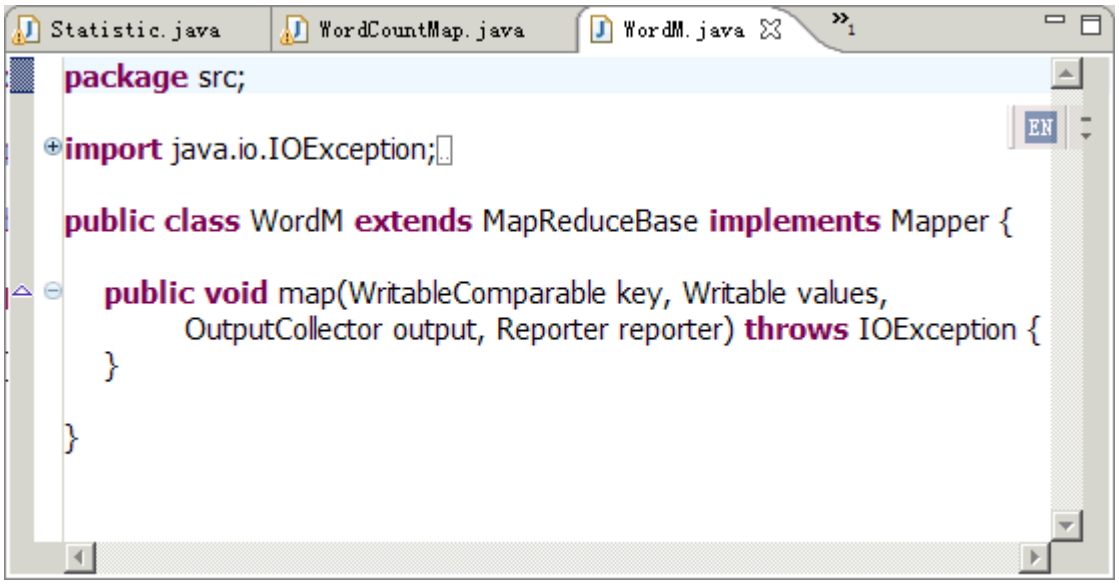
在“项目资源管理器”中对刚产生的项目点右键，出现如图菜单，选择“其他”。之后又会出现“选择向导”，如下图，找到“Mapper”，点击“下一步”；



在下图中，提示“建议不要使用缺省包”。应该为类指定一个包的路径，方便代码的封装。本例的名称为“WordM”，包为“src”。



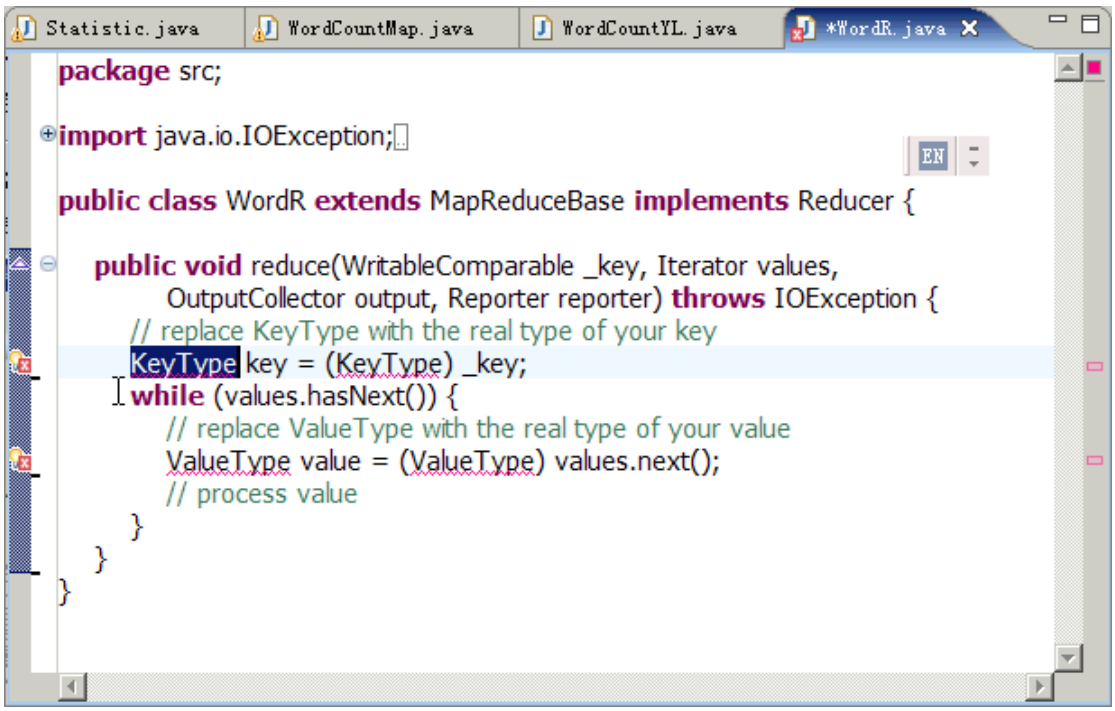
之后，该插件会为新类生成必要的框架，如下图。



新建 Reducer 类

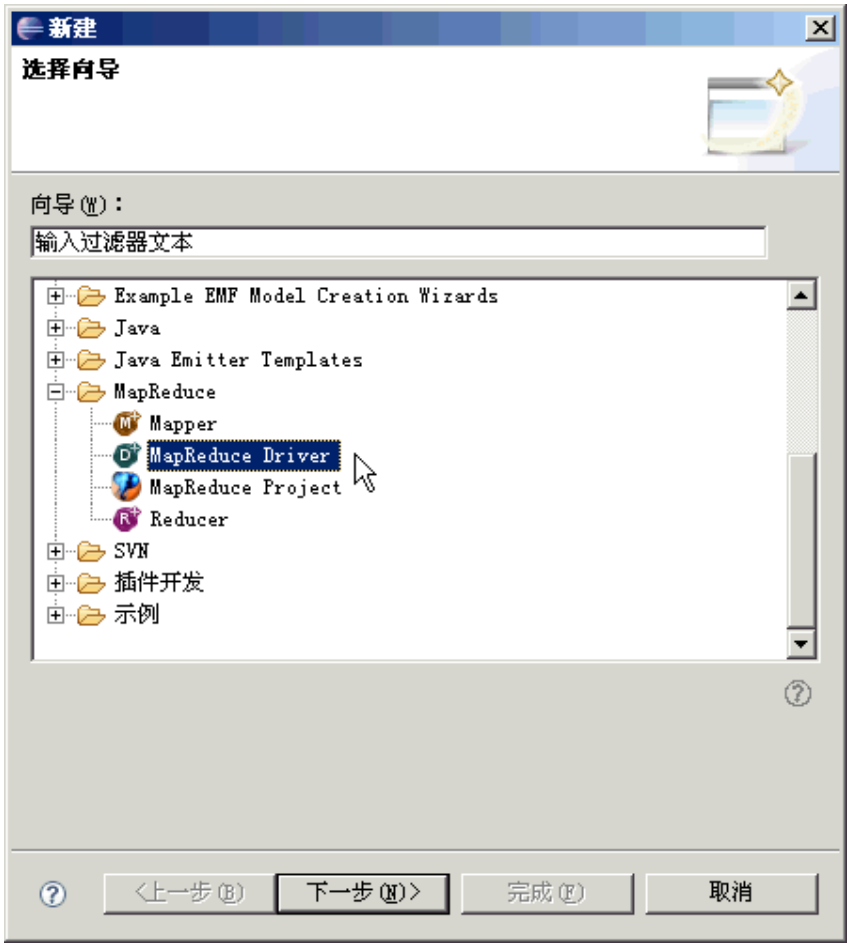


在新建的对话框中，选择“Reducer”。

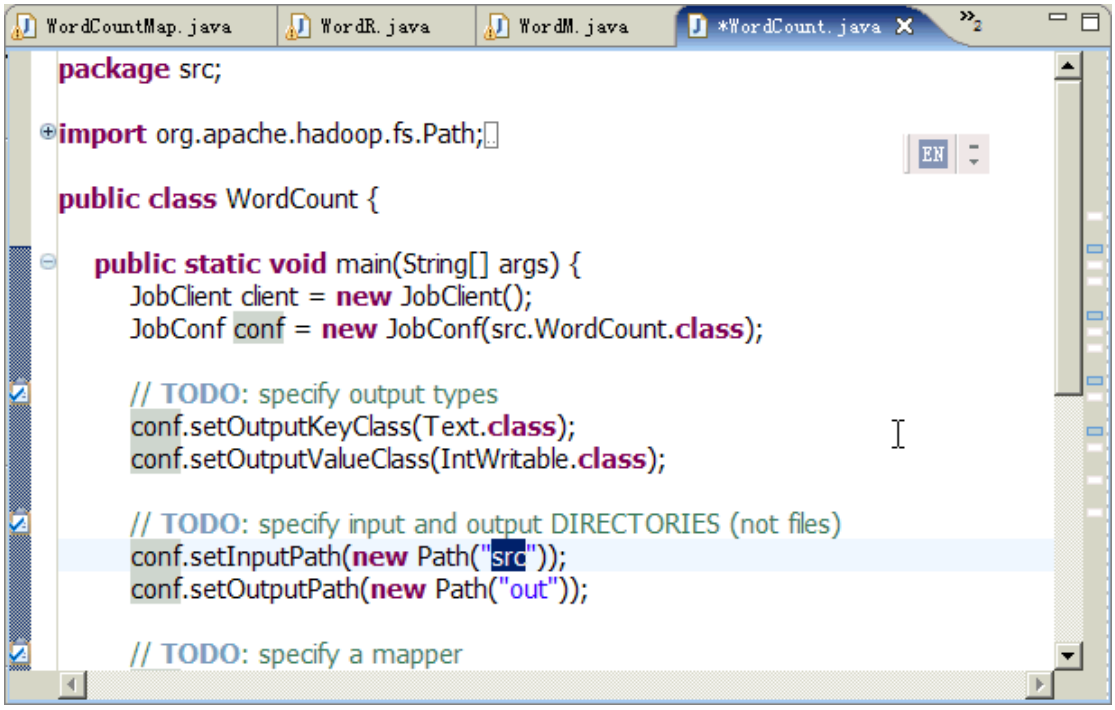


如上图，程序架构已经产生，按照设计添加代码即可。

新建主类



在新建的对话框中，选择“MapReduce Driver”，即可生成必要的代码。



如上图，代码都已经产生，需要按照设计，更改输入和输出。向导产生的代码是用于计算词频的，所以“KeyClass”为“Text.class”，“ValueClass”为“IntWritable.class”，实际可以按照需要，使用其它 Writable 类。

```
conf.setJobName("WordCountYL");

// TODO: specify output types
conf.setOutputKeyClass(Text.class);
conf.setOutputValueClass(IntWritable.class);

// TODO: specify input and output DIRECTORIES (not files)
conf.setInputPath(new Path("YangLai"));
conf.setOutputPath(new Path("WordCountYL"));

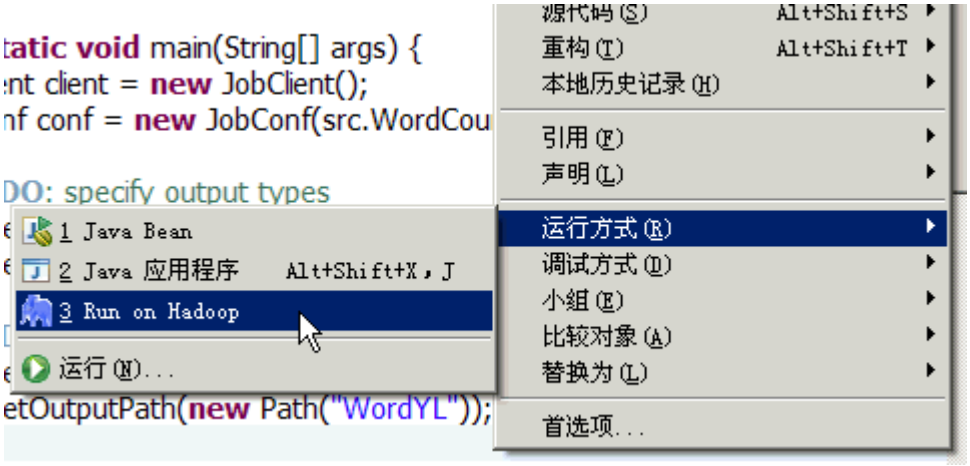
// TODO: specify a mapper
conf.setMapperClass(org.apache.hadoop.mapred.lib.IdentityMapper.class);

// TODO: specify a reducer
conf.setReducerClass(org.apache.hadoop.mapred.lib.IdentityReducer.class);

client.setConf(conf);
```

注意，要将缺省的“setMapperClass”和“setReducerClass”改为刚才自己产生的类，比如“WordM”和“WordR”。

在 Hadoop 上运行



如上图，在主类的窗口点击鼠标右键，出现菜单。选择“Run on Hadoop”，程序执行的情况将会在控制台反映出来，相当方便。

MapReduce 透视图

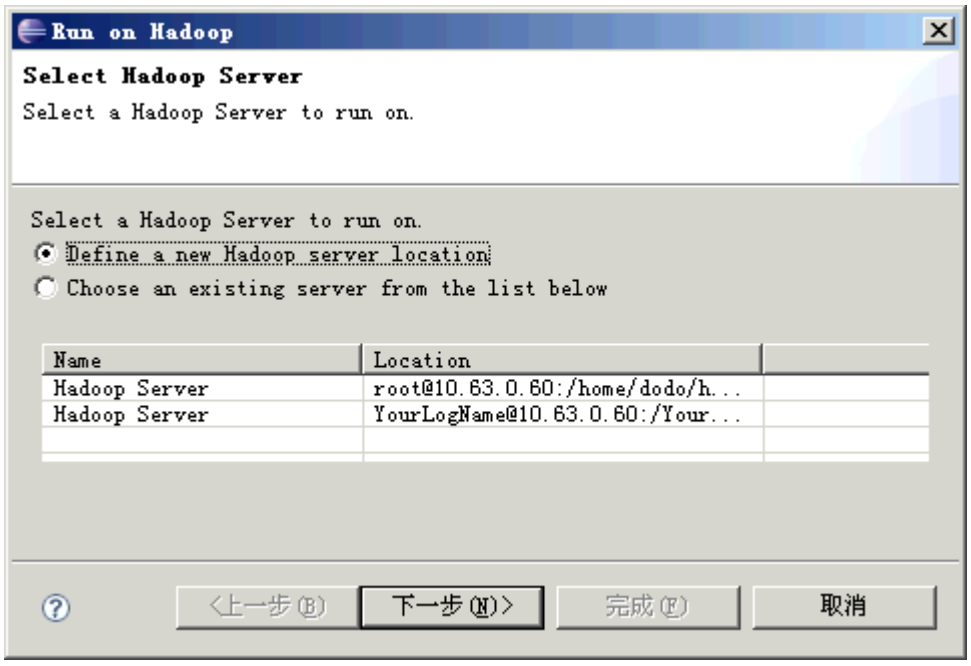
为了能够使用该插件诸多功能，必须选择 MapReduce 透视图，如下。



确定之后，在控制台旁边会有新的大象图标出现，如右图。

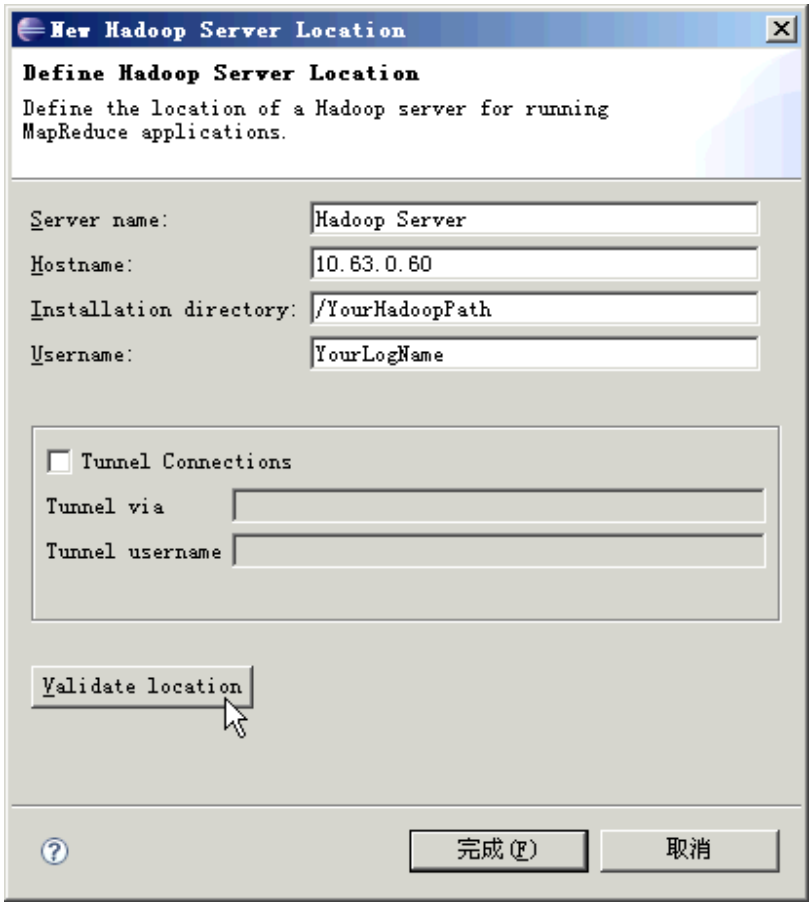


点击大象图标会出现“Select Hadoop Server”的对话框，如下图。

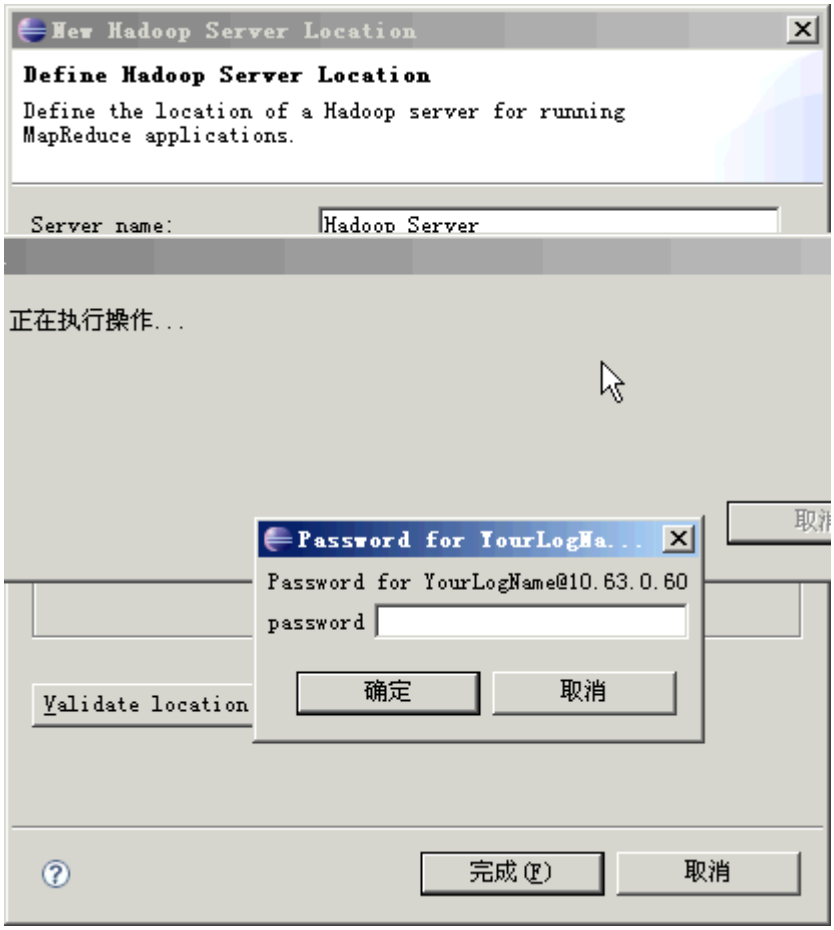


如下图，填写“Hostname”、“Installation directory”、“Username”。

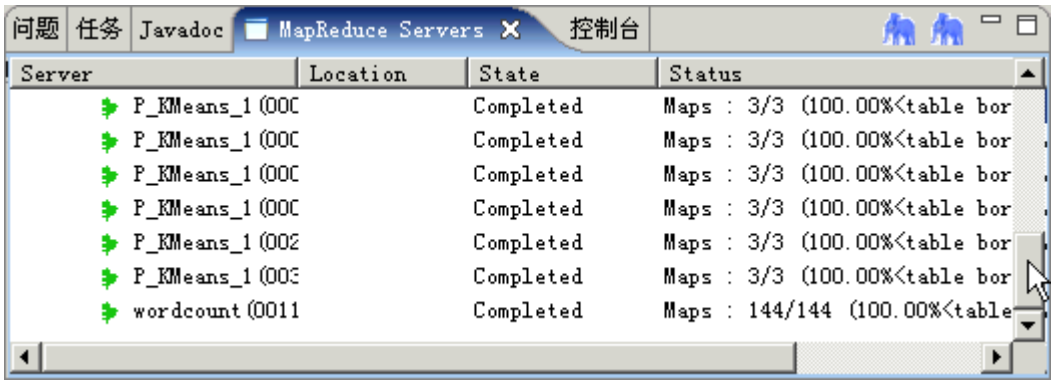
这里需要提醒的是，Hadoop 中象“Installation directory”之类的参数要尽量简短，否则会在许多地方造成不必要的麻烦。



点击“Validate location”来确认资料准确无误，然后再点击完成。如下图，会要求密码。

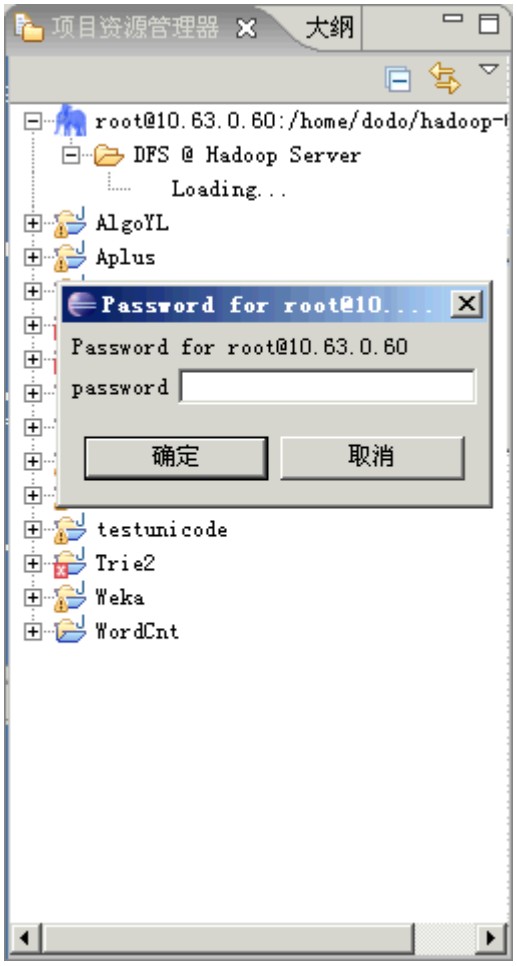


在控制台的旁边，会有“MapReduce Servers”出现，其指示当前 hadoop 上的任务情况。

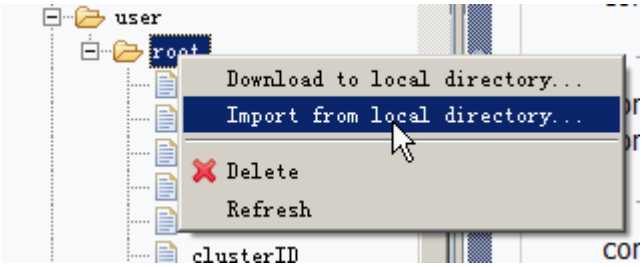


操作 DFS 的目录

在“项目资源管理器”中有 DFS 的目录可以看到，该功能非常有用，将取代许多 linux 下的拷贝、分发命令。第一次访问需要密码。如下图



在 DFS 目录中右键就可以出现下图菜单，可以上传和下载目录到 DFS。注意是整个目录，而不是文件。其中的“Delete”能够方便地删除 DFS 目录。从性能考虑，程序运行之后的结果不会马上显示在 DFS 目录，需要用“Refresh”来刷新。



4.2. 统计度量的 MapReduce 实现

设计

由于要求在一次扫描后求得所有属性（假设为*LenFields*个）的统计度量，我们需要保存个这*LenFields*个属性的最小值、最大值、计数、求和、平房和，然后计算均值、方差，所以在所有的Mapper中需要有 4**LenFields*个全局的Key，来汇集最后的Value。鉴于Hadoop最后由Reducer产生的结果是若干行<key, value>，如4.3中间数据part-00000，结果不容易被使用，需要用csv文件规范。

我们要用 Hadoop 的整型 IntWritable 作为 Key，才能方便处理 *LenFields* 个属性的最小值、最大值、计数、求和、平方和。采用 iFields[][]来统一表示所有的 Key 值，除计数外。计数对所有属性都一样，只用单个变量 *iwCnt* 来 Map。

以下代码版本为 Hadoop-0.15.0。

Mapper 代码

```
package src;

import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.Writable;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.mapred.JobConf;

public class StatM extends MapReduceBase implements Mapper,Parameter {
    /** The iFields[][] is the Array for Keys to present the min(), max(), sum() of each field in the input.
     * iFields[0] for min()
     * iFields[1] for max()
     * iFields[2] for sum()
     * iFields[3] for sum2() */
```

```
    private static IntWritable iFields[][] = new IntWritable[LenStat][LenFields];
    private static float min[] = new float[LenFields];
    private static float max[] = new float[LenFields];
    private static IntWritable iwCnt = new IntWritable(LenStat*LenFields);
    private final static FloatWritable one = new FloatWritable(1);

    public StatM(){
        for (int i=0; i<LenStat; i++){
            for (int j=0; j<LenFields; j++){
                iFields[i][j] = new IntWritable(i*LenFields+j);
            }
        }
        for (int j=0; j<LenFields; j++){
            min[j] =    fMax;  // the maximum integer
            max[j] = fMin; // the minimum integer
        }
    }
    public void map(WritableComparable key, Writable values,
            OutputCollector output, Reporter reporter) throws IOException {
        StringTokenizer st = new StringTokenizer(values.toString().toLowerCase()," \t,;");
        float iTmp;
        for (int j=0; j<LenFields; j++){   /** handle each field. */
            iTmp = Float.parseFloat(st.nextToken());
            if (min[j]>iTmp){     /** for min(), this judgement just output about 37
<key,value> pairs in 100,000 records. */
                min[j] = iTmp;
                output.collect(iFields[0][j],new FloatWritable(min[j]));
            }
            if (max[j]<iTmp){     /** for max() */
                max[j] = iTmp;
                output.collect(iFields[1][j],new FloatWritable(max[j]));
            }
            output.collect(iFields[2][j], new FloatWritable(iTmp));   /** for sum() */
            output.collect(iFields[3][j], new FloatWritable(iTmp*iTmp));/** for sum2() */
        }
        output.collect(iwCnt, one); /** for cnt() */
    }
}
```

Reducer 代码

```
package src;

import java.io.IOException;
import java.util.Iterator;

import org.apache.hadoop.io.IntWritable;
```

```
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;

public class StatR extends MapReduceBase implements Reducer,Parameter {

    public void reduce(WritableComparable _key, Iterator values,
            OutputCollector output, Reporter reporter) throws IOException {
        IntWritable K = (IntWritable) _key;
        FloatWritable fElement;
        float fTmp;
        float min =   iMax;
        float max = iMin;
        float sum = 0;
        long iCnt = 0;
        int iCategory = K.get() / LenFields; // restore the category from Keys.
        switch (iCategory){
        case 0 :/** min()*/
            while (values.hasNext()){
                fElement = (FloatWritable) values.next();
                fTmp = fElement.get();
                if (  min > fTmp){
                    min =fTmp;
                }
            }
            output.collect(K, new FloatWritable(min));
            break;
        case 1 :/** max() */
            while (values.hasNext()){
                fElement = (FloatWritable) values.next();
                fTmp = fElement.get();
                if (  max < fTmp){
                    max =fTmp;
                }
            }
            output.collect(K, new FloatWritable(max));
            break;
        case 2 :/** sum() */
            while (values.hasNext()){
                fElement = (FloatWritable) values.next();
                sum += fElement.get();
```

```
            }
            output.collect(K, new FloatWritable(sum));
            break;
        case 3 : /** sum2() */
            while (values.hasNext()){
                fElement = (FloatWritable) values.next();
                sum += fElement.get();
            }
            output.collect(K, new FloatWritable(sum));
            break;
        case 4 :
                while (values.hasNext()){
                    fElement = (FloatWritable) values.next();
                    iCnt +=   fElement.get();
                }
                output.collect(K, new FloatWritable(iCnt));
                break;
        }   // switch
    }
}
```

Main 代码

```
package src;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.util.StringTokenizer;
import java.lang.Math;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;

public class StatMain implements Parameter{

    public static void main(String[] args) {
        JobClient client = new JobClient();
        JobConf conf = new JobConf(src.StatMain.class);
        // TODO: The setOutputKeyClass() define the type of Key and Value.
        conf.setOutputKeyClass(IntWritable.class);
```

```
        conf.setOutputValueClass(FloatWritable.class);
        // TODO: specify input and output DIRECTORIES (not files)
        conf.setInputPath(new Path("Yang1"));
        conf.setOutputPath(new Path("YangCnt"));
        // TODO: Use the name of your mapper.
        conf.setMapperClass(StatM.class);
        // TODO: Use the name of your reducer.
        conf.setReducerClass(StatR.class);
        conf.setNumReduceTasks(1);//This results in the file “part-00000”. The more the
number, the more the files, i.e. “part-00001”, “part-00002”...
        client.setConf(conf);
        try {
            JobClient.runJob(conf);
            int iNum;
            float fNum, iCntRecord=1;
            float fStat[][] = new float[LenStat][LenFields];
        // TODO: Read the reduced file “part-00000” from DFS.
            String OutPath = "YangCnt";
             Path pathStat = new Path(OutPath,"part-00000");
            FileSystem FileSys = FileSystem.get(new Configuration());
            BufferedReader bin = new BufferedReader(new InputStreamReader
( FileSys.open( pathStat)));
            String values;
            int iRow,iCol;
            while ((values=bin.readLine())!=null){
                StringTokenizer st = new StringTokenizer
( values.toString() .toLowerCase()," \t,");
                iNum = Integer.parseInt(st.nextToken());
                iRow = iNum / LenFields;
                iCol = iNum % LenFields;
                if (iRow == LenStat){
                    iCntRecord = Float.parseFloat(st.nextToken());
                }else{
                    fNum = Float.parseFloat((st.nextToken()));
                    fStat[iRow][iCol] = fNum;
                }
            }
            bin.close();
        // TODO: Transform file “part-00000” into .csv file in DFS.
            Path pathStatcsv = new Path(OutPath,"Statistic.csv");
            BufferedWriter bout = new BufferedWriter(new OutputStreamWriter
( FileSys.create(pathStatcsv)));
            /** for the min(), max(), sum(), sum2(). */
            for (int i=0; i<LenStat-1; i++){
                bout.write(Float.toString(fStat[i][0]));
                for (int j=1; j<LenFields; j++){
```

```
                    bout.write(',');
                    bout.write(Float.toString(fStat[i][j]));
                }
                bout.write('\n');
            }
            /** count the avg() */
            bout.write(Float.toString(fStat[2][0]/iCntRecord));
            for (int j=1; j<LenFields; j++){
                bout.write(',');
                bout.write(Float.toString(fStat[2][j]/iCntRecord));
            }
            bout.write('\n');

            /** count the stdev() */
            float DevSQ = (fStat[3][0]-fStat[2][0]*fStat[2][0]/iCntRecord)/(iCntRecord-1);
            bout.write(Float.toString((float)Math.sqrt(DevSQ)));
            for (int j=1; j<LenFields; j++){
                bout.write(',');
                DevSQ = (fStat[3][j]-fStat[2][j]*fStat[2][j]/iCntRecord)/(iCntRecord-1);
                bout.write(Float.toString((float)Math.sqrt(DevSQ)));
            }
            bout.write('\n');

            bout.flush(); bout.close();//*/
        } catch (Exception e) {
            e.printStackTrace();
        }
        System.out.println("Success!");
    }

}

package src;

public interface Parameter {
    /**
     * LenFields is the length of fields, or columns; it need to be modified for varied
     */
    public final int LenFields = 4;
    /**
     * iFields[0] for min()
     * iFields[1] for max()
     * iFields[2] for sum()
     * iFields[3] for sum2()
     */
    public final int LenStat = 4;
```

```
    public final int iMin = 0x80000000;
    public final int iMax = 0x7FFFFFFF;
    public final float fMin = (float)-3.4E38;
    public final float fMax = (float)3.4E38;
}
```

4.3. 中间数据 part-00000

由于 Reducer 产生的结果只能是 key 和 value 两列，*LenFields* 个属性的统计度量数组 iFields[][] 被展开。最后一个值对表示所有属性的计数，因为全部相同，只写一次。

```
0   3.0
1   1.0
2   1.0
3   1.0
4   9.0
5   9.0
6   9.0
7   8.0
8   44.0
9   44.0
10  38.0
11  25.0
12  306.0
13  320.0
14  280.0
15  129.0
16  7.0
```

4.4. 控制台输出结果

```
08/07/03 00:43:53 INFO mapred.FileInputFormat: Total input paths to process : 1
08/07/03 00:43:54 INFO mapred.JobClient: Running job: job_200807011008_0524
08/07/03 00:43:55 INFO mapred.JobClient:  map 0% reduce 0%
08/07/03 00:43:58 INFO mapred.JobClient:  map 33% reduce 0%
08/07/03 00:43:59 INFO mapred.JobClient:  map 66% reduce 0%
08/07/03 00:44:01 INFO mapred.JobClient:  map 100% reduce 0%
08/07/03 00:45:04 INFO mapred.JobClient:  map 100% reduce 100%
08/07/03 00:45:05 INFO mapred.JobClient: Job complete: job_200807011008_0524
08/07/03 00:45:05 INFO mapred.JobClient: Counters: 9
08/07/03 00:45:05 INFO mapred.JobClient:   Job Counters
08/07/03 00:45:05 INFO mapred.JobClient:     Launched map tasks=3
08/07/03 00:45:05 INFO mapred.JobClient:     Launched reduce tasks=1
08/07/03 00:45:05 INFO mapred.JobClient:   Map-Reduce Framework
08/07/03 00:45:05 INFO mapred.JobClient:     Map input records=7
08/07/03 00:45:05 INFO mapred.JobClient:     Map output records=100
08/07/03 00:45:05 INFO mapred.JobClient:     Map input bytes=490
```

```
08/07/03 00:45:05 INFO mapred.JobClient:        Map output bytes=800
08/07/03 00:45:05 INFO mapred.JobClient:        Reduce input groups=17
08/07/03 00:45:05 INFO mapred.JobClient:        Reduce input records=100
08/07/03 00:45:05 INFO mapred.JobClient:        Reduce output records=17
Success!
```

5. 注意

5.1. 范型的问题

如果不注意范型，则在4.2统计度量的MapReduce实现的代码虽然能够运行，但是会有警告提示。将原来的代码修改为如下，可解决问题。

Mapper 的范型

```
public class StatM extends MapReduceBase
implements Mapper<Object, Object, IntWritable, FloatWritable>,Parameter {
    /*
    public void map(WritableComparable key, Writable values,
            OutputCollector output, Reporter reporter) throws IOException {
            */
    public void map(Object key, Object values,
            OutputCollector<IntWritable, FloatWritable> output,
            Reporter reporter) throws IOException {
```

Reducer 的范型

```
public class StatR
extends MapReduceBase
implements Reducer< IntWritable, FloatWritable, IntWritable, FloatWritable>,Parameter {
    /*
    public void reduce(WritableComparable _key, Iterator values,
            OutputCollector output, Reporter reporter) throws IOException {*/
    public void reduce(IntWritable _key, Iterator values,
            OutputCollector< IntWritable, FloatWritable> output,
            Reporter reporter) throws IOException {
```

5.2. Hadoop-0.17.0 的变化

经测试，原来的 hadoop-0.15.0 程序到 hadoop-0.17.0 的变动有以下：

1、在 eclipse 中需要将外部包 hadoop-0.15.0-core.jar 更换为 hadoop-0.17.0-core.jar，该包位于 hadoop-0.17.0.tar.gz 中；hadoop-0.17.1.tar.gz 需要到网络下载。

2、如果不改变类型，会有错误，认为没有实现 Reducer.reduce()和 Mapper.map()

```
 /*  在 hadooop-0.15.0 中的 Reducer.reduce(WritableComparable, Iterator, OutputCollector,
Reporter)
 public void reduce(WritableComparable _key, Iterator values,
   OutputCollector output, Reporter reporter) throws IOException {*/

// 在 hadooop-0.17.0 中的 Reducer.reduce(Object, Iterator, OutputCollector, Reporter)
 public void reduce(Object _key, Iterator values,
   OutputCollector output, Reporter reporter) throws IOException {
 /* 在 hadooop-0.15.0 中的 Reducer.reduce(WritableComparable, Iterator, OutputCollector,
Reporter)
 public void map(WritableComparable key, Writable values,
   OutputCollector output, Reporter reporter) throws IOException {
   */
// 在 hadooop-0.17.0 中的 Reducer.reduce(Object, Iterator, OutputCollector, Reporter)
 public void map(Object key, Object values,
   OutputCollector output, Reporter reporter) throws IOException {
```

3、原来的输入目录和输出目录的设置需要换另外的函数：

```
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
  // TODO: specify input and output DIRECTORIES (not files)

  //conf.setInputPath(new Path("Yang1"));// 原来 0.15.0
  //conf.setOutputPath(new Path("YangCnt"));
  FileInputFormat.setInputPaths(conf, new Path("yangl/Yang1"));
  FileOutputFormat.setOutputPath(conf, new Path("yangl/YangCnt"));
```

有些代码被认为过时，需要改动，例如

```
//conf.setInputPath(new Path("Yang1"));
//conf.setOutputPath(new Path("YangCnt"));
FileInputFormat.setInputPaths(conf, new Path("yangl/Yang1"));
FileOutputFormat.setOutputPath(conf, new Path("yangl/YangCnt"));
```

输出

```
08/07/09 01:35:16 INFO mapred.FileInputFormat: Total input paths to process : 1
08/07/09 01:35:18 INFO mapred.JobClient: Running job: job_200807061815_0316
08/07/09 01:35:19 INFO mapred.JobClient:   map 0% reduce 0%
08/07/09 01:35:31 INFO mapred.JobClient:   map 1% reduce 0%
08/07/09 01:35:36 INFO mapred.JobClient:   map 3% reduce 0%
08/07/09 01:35:43 INFO mapred.JobClient:   map 4% reduce 0%
08/07/09 01:35:52 INFO mapred.JobClient:   map 5% reduce 0%
08/07/09 01:35:53 INFO mapred.JobClient:   map 6% reduce 0%
08/07/09 01:35:58 INFO mapred.JobClient:   map 7% reduce 0%
08/07/09 01:36:12 INFO mapred.JobClient:   map 8% reduce 0%
08/07/09 01:36:13 INFO mapred.JobClient:   map 9% reduce 0%
08/07/09 01:36:19 INFO mapred.JobClient:   map 10% reduce 0%
08/07/09 01:36:23 INFO mapred.JobClient:   map 11% reduce 0%
08/07/09 01:36:29 INFO mapred.JobClient:   map 13% reduce 0%
08/07/09 01:36:30 INFO mapred.JobClient:   map 14% reduce 0%
```

```
08/07/09 01:36:38 INFO mapred.JobClient:  map 15% reduce 0%
08/07/09 01:36:43 INFO mapred.JobClient:  map 17% reduce 0%
08/07/09 01:36:45 INFO mapred.JobClient:  map 18% reduce 0%
08/07/09 01:36:50 INFO mapred.JobClient:  map 19% reduce 0%
08/07/09 01:36:54 INFO mapred.JobClient:  map 20% reduce 0%
08/07/09 01:36:59 INFO mapred.JobClient:  map 21% reduce 0%
08/07/09 01:37:03 INFO mapred.JobClient:  map 22% reduce 0%
08/07/09 01:37:06 INFO mapred.JobClient:  map 23% reduce 0%
08/07/09 01:37:09 INFO mapred.JobClient:  map 24% reduce 0%
08/07/09 01:37:11 INFO mapred.JobClient:  map 25% reduce 0%
08/07/09 01:37:16 INFO mapred.JobClient:  map 26% reduce 0%
08/07/09 01:37:21 INFO mapred.JobClient:  map 27% reduce 0%
08/07/09 01:37:23 INFO mapred.JobClient:  map 28% reduce 0%
08/07/09 01:37:26 INFO mapred.JobClient:  map 29% reduce 0%
08/07/09 01:37:28 INFO mapred.JobClient:  map 30% reduce 0%
08/07/09 01:37:36 INFO mapred.JobClient:  map 31% reduce 0%
08/07/09 01:37:39 INFO mapred.JobClient:  map 32% reduce 0%
08/07/09 01:37:41 INFO mapred.JobClient:  map 33% reduce 0%
08/07/09 01:37:48 INFO mapred.JobClient:  map 34% reduce 0%
08/07/09 01:37:51 INFO mapred.JobClient:  map 35% reduce 0%
08/07/09 01:37:57 INFO mapred.JobClient:  map 35% reduce 2%
08/07/09 01:38:00 INFO mapred.JobClient:  map 36% reduce 2%
08/07/09 01:38:02 INFO mapred.JobClient:  map 37% reduce 7%
08/07/09 01:38:05 INFO mapred.JobClient:  map 38% reduce 7%
08/07/09 01:38:09 INFO mapred.JobClient:  map 39% reduce 7%
08/07/09 01:38:12 INFO mapred.JobClient:  map 40% reduce 7%
08/07/09 01:38:16 INFO mapred.JobClient:  map 41% reduce 7%
08/07/09 01:38:17 INFO mapred.JobClient:  map 41% reduce 9%
08/07/09 01:38:21 INFO mapred.JobClient:  map 42% reduce 9%
08/07/09 01:38:25 INFO mapred.JobClient:  map 44% reduce 9%
08/07/09 01:38:30 INFO mapred.JobClient:  map 45% reduce 9%
08/07/09 01:38:31 INFO mapred.JobClient:  map 46% reduce 9%
08/07/09 01:38:36 INFO mapred.JobClient:  map 47% reduce 9%
08/07/09 01:38:41 INFO mapred.JobClient:  map 48% reduce 9%
08/07/09 01:38:47 INFO mapred.JobClient:  map 49% reduce 9%
08/07/09 01:38:50 INFO mapred.JobClient:  map 51% reduce 9%
08/07/09 01:38:52 INFO mapred.JobClient:  map 52% reduce 9%
08/07/09 01:38:57 INFO mapred.JobClient:  map 53% reduce 9%
08/07/09 01:39:02 INFO mapred.JobClient:  map 54% reduce 9%
08/07/09 01:39:07 INFO mapred.JobClient:  map 55% reduce 9%
08/07/09 01:39:10 INFO mapred.JobClient:  map 57% reduce 9%
08/07/09 01:39:15 INFO mapred.JobClient:  map 58% reduce 9%
08/07/09 01:39:17 INFO mapred.JobClient:  map 59% reduce 9%
08/07/09 01:39:22 INFO mapred.JobClient:  map 60% reduce 9%
08/07/09 01:39:25 INFO mapred.JobClient:  map 61% reduce 9%
08/07/09 01:39:30 INFO mapred.JobClient:  map 62% reduce 9%
```

```
08/07/09 01:39:31 INFO mapred.JobClient:   map 63% reduce 9%
08/07/09 01:39:36 INFO mapred.JobClient:   map 64% reduce 9%
08/07/09 01:39:37 INFO mapred.JobClient:   map 65% reduce 9%
08/07/09 01:39:42 INFO mapred.JobClient:   map 66% reduce 9%
08/07/09 01:39:50 INFO mapred.JobClient:   map 67% reduce 9%
08/07/09 01:39:53 INFO mapred.JobClient:   map 68% reduce 11%
08/07/09 01:39:55 INFO mapred.JobClient:   map 69% reduce 11%
08/07/09 01:40:02 INFO mapred.JobClient:   map 71% reduce 11%
08/07/09 01:40:07 INFO mapred.JobClient:   map 71% reduce 14%
08/07/09 01:40:10 INFO mapred.JobClient:   map 72% reduce 14%
08/07/09 01:40:12 INFO mapred.JobClient:   map 74% reduce 14%
08/07/09 01:40:17 INFO mapred.JobClient:   map 75% reduce 14%
08/07/09 01:40:23 INFO mapred.JobClient:   map 78% reduce 19%
08/07/09 01:40:26 INFO mapred.JobClient:   map 79% reduce 19%
08/07/09 01:40:27 INFO mapred.JobClient:   map 82% reduce 19%
08/07/09 01:40:33 INFO mapred.JobClient:   map 84% reduce 19%
08/07/09 01:40:41 INFO mapred.JobClient:   map 84% reduce 21%
08/07/09 01:40:42 INFO mapred.JobClient:   map 85% reduce 21%
08/07/09 01:40:43 INFO mapred.JobClient:   map 86% reduce 21%
08/07/09 01:40:47 INFO mapred.JobClient:   map 87% reduce 21%
08/07/09 01:40:48 INFO mapred.JobClient:   map 88% reduce 21%
08/07/09 01:40:54 INFO mapred.JobClient:   map 89% reduce 21%
08/07/09 01:41:03 INFO mapred.JobClient:   map 90% reduce 26%
08/07/09 01:41:04 INFO mapred.JobClient:   map 91% reduce 26%
08/07/09 01:41:12 INFO mapred.JobClient:   map 92% reduce 26%
08/07/09 01:41:17 INFO mapred.JobClient:   map 93% reduce 26%
08/07/09 01:41:22 INFO mapred.JobClient:   map 94% reduce 26%
08/07/09 01:41:26 INFO mapred.JobClient:   map 95% reduce 26%
08/07/09 01:41:34 INFO mapred.JobClient:   map 96% reduce 26%
08/07/09 01:41:37 INFO mapred.JobClient:   map 97% reduce 26%
08/07/09 01:41:42 INFO mapred.JobClient:   map 97% reduce 28%
08/07/09 01:41:47 INFO mapred.JobClient:   map 98% reduce 28%
08/07/09 01:41:59 INFO mapred.JobClient:   map 98% reduce 30%
08/07/09 01:42:01 INFO mapred.JobClient:   map 99% reduce 30%
08/07/09 01:42:16 INFO mapred.JobClient:   map 100% reduce 30%
08/07/09 01:42:26 INFO mapred.JobClient:   map 100% reduce 100%
08/07/09 01:42:27 INFO mapred.JobClient: Job complete: job_200807061815_0316
08/07/09 01:42:27 INFO mapred.JobClient: Counters: 17
08/07/09 01:42:27 INFO mapred.JobClient:     Job Counters
08/07/09 01:42:27 INFO mapred.JobClient:       Launched map tasks=16
08/07/09 01:42:27 INFO mapred.JobClient:       Launched reduce tasks=1
08/07/09 01:42:27 INFO mapred.JobClient:       Data-local map tasks=13
08/07/09 01:42:27 INFO mapred.JobClient:       Rack-local map tasks=1
08/07/09 01:42:27 INFO mapred.JobClient:     Map-Reduce Framework
08/07/09 01:42:27 INFO mapred.JobClient:       Map input records=10000000
08/07/09 01:42:27 INFO mapred.JobClient:       Map output records=90001540
```

```
08/07/09 01:42:27 INFO mapred.JobClient:     Map input bytes=888610373
08/07/09 01:42:27 INFO mapred.JobClient:     Map output bytes=720012320
08/07/09 01:42:27 INFO mapred.JobClient:     Combine input records=90001540
08/07/09 01:42:27 INFO mapred.JobClient:     Combine output records=1108
08/07/09 01:42:27 INFO mapred.JobClient:     Reduce input groups=17
08/07/09 01:42:27 INFO mapred.JobClient:     Reduce input records=1108
08/07/09 01:42:27 INFO mapred.JobClient:     Reduce output records=17
08/07/09 01:42:27 INFO mapred.JobClient:   File Systems
08/07/09 01:42:27 INFO mapred.JobClient:     Local bytes read=48344
08/07/09 01:42:27 INFO mapred.JobClient:     Local bytes written=68340
08/07/09 01:42:27 INFO mapred.JobClient:     HDFS bytes read=888663634
08/07/09 01:42:27 INFO mapred.JobClient:     HDFS bytes written=261
```

Map&Reduce: 431500

Reading part-00000: 433

Writing .csv: 106
Success!

6. 统计知识

MIN()

返回一组值中的最小值。

MAX()

返回一组值中的最大值。

SUM()

返回某一单元格区域中所有数字之和。计算公式为： $\sum x$。

SUM2()

返回某一单元格区域中所有数字的平方之和。用于计算STD()。计算公式为： $\sum x^2$。

STD()

估算样本的标准偏差。标准偏差反映相对于平均值 (mean) 的离散程度。

其参数是总体中的样本。此处标准偏差的计算使用“无偏差”或“n-1”方法。

函数 STDEV 的计算公式如下：$\text{sqrt}[(\sum x^2 - (\sum x)^2/n) / (n-1)]$

AVG()

计算平均值（算术平均值），反映数据的中心。计算公式为： $(\sum x)/n$。

CNT()

计算数据的个数，即 n。