Exercise 1

MapReduce using the Java Perspective

© Copyright IBM Corporation, 2013

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Lab 1 Mapreduce using the Java Perspective

In this exercises, the student develops a MapReduce application that finds the highest average monthly temperature using the Java perspective in Eclipse.

After completing this hands-on lab, you'll be able to:

- Code a MapReduce application using the Java perspective in Eclipse

Allow 45 minutes to complete this lab.

This version of the lab was designed using the InfoSphere BigInsights 2.1 Quick Start Edition, but has been tested on the 3.0 image. Throughout this lab you will be using the following account login information. If your passwords are different, please note the difference.

	Username	**Password**
VM image setup screen	root	password
Linux	biadmin	biadmin

The assumption is that you have followed the instruction in this course download and untarred BDU_Hadoop_Files.tar. Data in that tar file is required by this exercise.

1.1 Start the BigInsights components

__ 1. Log into your BigInsights image with a userid of **biadmin** and a password of **biadmin.**

__ 2. Start your BigInsights components. Use the icon on the desktop.

__ 3. From a command line (right-click the desktop and select **Open in Terminal**) execute:

```
hadoop fs -mkdir TempData
```

__ 4. Upload some temperature data from the local file system.

```
hadoop fs -copyFromLocal /home/labfiles/SumnerCountyTemp.dat
/user/biadmin/TempData
```

__ 5. You can view this data from the *Files* tab in the Web Console or by executing the following command. The values in the 95th column (354, 353, 353,353, 352...) are the average daily temperatures. They are the result of multiplying the actual average temperature value times 10. (That way you don’t have to worry about working with decimal points.)

```
hadoop fs -cat TempData/SumnerCountyTemp.dat
```

1.2 Define a Java project in Eclipse

__ 1. Start Eclipse using the icon on the desktop. Go with the default workspace.

__ 2. Make sure that you are using the Java perspective. Click **Window->Open Perspective->Other.** Then select **Java.** Click **OK.**

__ 3. Create a Java project. Select **File->New->Java Project.**

__ 4. Specify a *Project name* of **MaxTemp.** Click **Finish.**

__ 5. Right-click the *MaxTemp* project, scroll down and select **Properties.**

__ 6. Select **Java Build Path.**

__ 7. In the *Properties* dialog, select the **Libraries** tab.

__ 8. Click the **Add Library** pushbutton.

__ 9. Select *BigInsights Libraries* and click **Next.** Then click **Finish.** Then click **OK.**

1.3 Create a Java package and the mapper class

__ 1. In the *Package Explorer* expand *MaxTemp* and right-click **src.** Select **New->Package.**

__ 2. Type a *Name* of **com.some.company.** Click **Finish.**

__ 3. Right-click *com.some.company* and select **New->Class.**

__ 4. Type in a *Name* of **MaxTempMapper.** It will be a *public* class. Click **Finish.**

The data type for the input key to the mapper will be *LongWritable.* The data itself will be of type *Text.* The output key from the mapper will be of type *Text.* And the data from the mapper (the temperature) will be of type *IntWritable.*

__ 5. Your class:

__ a. You will need to import *java.io.IOException.*

__ b. Exend Mapper<LongWritable, Text, Text, IntWritable>

__ c. Define a public class called **map.**

__ d. Your code should look like the following:

```
package com.some.company;

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class MaxTempMapper extends
     Mapper<LongWritable, Text, Text, IntWritable> {

   @Override
   public void map(LongWritable key, Text value, Context context)
           throws IOException, InterruptedException {

   }
}
```

1.4 Complete the mapper

You are reading in a line of data. You will want to convert it to a string so that you can do some string manipulation. You will want to extract the month and average temperature for each record.

The month begins at the 22th character of the record (zero offset) and the average temperature begins at the 95th character. (Remember that the average temperature value is three digits.)

__ 1. In the *map* method, add the following code (or whatever code you think is required):

```
String line = value.toString();
String month = line.substring(22,24);
int avgTemp;
avgTemp = Integer.parseInt(line.substring(95,98));
context.write(new Text(month), new IntWritable(avgTemp));
```

__ 2. Save your work.

1.5 Create the reducer class

__ 1. In the *Package Explorer* right-click *com.some.company* and select **New->Class.**

__ 2. Type in a *Name* of **MaxTempReducer.** It will be a *public* class. Click **Finish.**

The data type for the input key to the reducer will be *Text.* The data itself will be of type *IntWritable.* The output key from the reducer will be of type *Text.* And the data from the reducer will be of type *IntWritable.*

__ 3. Your class:

__ a. You will need to import *java.io.IOException.*

__ b. Extend Reducer<Text, LongWritable, Text, IntWritable>

__ c. Define a public class called **reduce.**

__ d. Your code should look like the following:

```
package com.some.company;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class MaxTempReducer extends
     Reducer<Text, IntWritable, Text, IntWritable> {
   @Override
   public void reduce(Text key, Iterable<IntWritable> values, Context
context)
              throws IOException, InterruptedException {
   }
}
```

1.6 Complete the reducer

For the reducer, you want to iterate through all values for a given key. For each value, check to see if it is higher than any of the other values.

__ 1. Add the following code (or your variation) to the *reduce* method.

```
int maxTemp = Integer.MIN_VALUE;
for (IntWritable value: values) {
maxTemp = Math.max(maxTemp, value.get());
}
context.write(key, new IntWritable(maxTemp));
```

__ 2. Save your work.

1.7 Create the driver

__ 1. In the *Package Explorer* right-click *com.some.company* and select **New->Class.**

__ 2. Type in a *Name* of **MaxMonthTemp.** It will be a *public* class. Click **Finish.**

The *GenericOptionsParser()* will extract any input parameters that are not system parameters and place them in an array. In your case, two parameters will be passed to your application. The first parameter is the input file. The second parameter is the output directory. (This directory must not exist or your MapReduce application will fail.)

Your code should look like this:

```
package com.some.company;
import org.apache.hadoop.conf.Configuration;
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.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

import com.some.company.MaxTempReducer;
import com.some.company.MaxTempMapper;

public class MaxMonthTemp {

   public static void main(String[] args) throws Exception {
      Configuration conf = new Configuration();

      String[] programArgs =
          new GenericOptionsParser(conf, args).getRemainingArgs();
      if (programArgs.length != 2) {
         System.err.println("Usage: MaxTemp <in> <out>");
         System.exit(2);
      }
      Job job = new Job(conf, "Monthly Max Temp");
      job.setJarByClass(MaxMonthTemp.class);
      job.setMapperClass(MaxTempMapper.class);
      job.setReducerClass(MaxTempReducer.class);

      job.setOutputKeyClass(Text.class);
      job.setOutputValueClass(IntWritable.class);

      FileInputFormat.addInputPath(job, new Path(programArgs[0]));

      FileOutputFormat.setOutputPath(job, new Path(programArgs[1]));

      // Submit the job and wait for it to finish.
      System.exit(job.waitForCompletion(true) ? 0 : 1);

   }
}
```

__ 3. Save your work.

1.8 Create a JAR file

__ 1. In the *Package Explorer,* expand the *MaxTemp* project. Right-click *src* and select **Export.**

__ 2. Expand *Java* and select *JAR file.* Click **Next.**

__ 3. Click the *JAR file* **browse** pushbutton. Type in a name of **MyMaxTemp.jar**. Keep *biadmin* as the folder. Click **OK.**

__ 4. Click **Finish.**

To save time, you are not going to run your application now. You will add a combiner class and then run the application.

1.9 Add a combiner function

__ 1. Return to your Eclipse development environment. You are going to add a combiner function to your application. In a real world multi-node cluster, this would allow some reducing functions to take place on the mapper node and lessen the amount of network traffic.

__ 2. Look at the code for **MaxMonthTemp.java.** Add the following statement after the *job.setMaperClass(MaxTempMapper.class);* statement.

```
job.setCombinerClass(MaxTempReducer.class);
```

__ 3. Save your work.

1.10 Recreate the JAR file

__ 1. In the *Project Explorer,* expand the *MaxTemp* project. Right-click *src* and select **Export.**

__ 2. Expand *Java* and select *JAR file.* Click **Next.**

__ 3. Click the *JAR file* **browse** pushbutton. Type in a name of **MyMaxTemp.jar**. Keep *biadmin* as the folder. Click **OK.**

__ 4. Click **Finish.** Override and replace the previous JAR.

1.11 Run your application

__ 1. You need a command line. You may have to open a new one. Change to *biadmin's* home directory.

```
cd ~
```

__ 2. Execute your program. At the command line type:

```
hadoop jar MyMaxTemp.jar com.some.company.MaxMonthTemp
/user/biadmin/TempData/SumnerCountyTemp.dat /user/biadmin/TempDataOut
```

__ 3. Close the open edit windows in Eclipse**.**

End of exercise

NOTES

NOTES

© Copyright IBM Corporation 2013.

The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. This information is based on current IBM product plans and strategy, which are subject to change by IBM without notice. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way.

IBM, the IBM logo and ibm.com are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Please Recycle