Apache Hadoop – A course for undergraduates

Homework Labs, Lecture 4

Lab: Using ToolRunner and Passing Parameters

Files and Directories Used in this Exercise

Eclipse project: `toolrunner`

Java files:

`AverageReducer.java` (Reducer from AverageWordLength)

`LetterMapper.java` (Mapper from AverageWordLength)

`AvgWordLength.java` (driver from AverageWordLength)

Exercise directory: `~/workspace/toolrunner`

In this Exercise, you will implement a driver using ToolRunner.

Follow the steps below to start with the Average Word Length program you wrote in an earlier lab, and modify the driver to use ToolRunner. Then modify the Mapper to reference a Boolean parameter called `caseSensitive`; if true, the mapper should treat upper and lower case letters as different; if false or unset, all letters should be converted to lower case.

Modify the Average Word Length Driver to use Toolrunner

1. Copy the Reducer, Mapper and driver code you completed in the "Writing Java MapReduce Programs" lab earlier, in the `averagewordlength` project.

> **Copying Source Files**
>
> You can use Eclipse to copy a Java source file from one project or package to another by right-clicking on the file and selecting Copy, then right-clicking the new package and selecting Paste. If the packages have different names (e.g. if you copy from `averagewordlength.solution` to `toolrunner.stubs`), Eclipse will automatically change the `package` directive at the top of the file. If you copy the file using a file browser or the shell, you will have to do that manually.

2. Modify the `AvgWordLength` driver to use ToolRunner. Refer to the slides for details.

   a. Implement the `run` method

   b. Modify `main` to call `run`

3. Jar your solution and test it before continuing; it should continue to function exactly as it did before. Refer to the *Writing a Java MapReduce Program* lab for how to assemble and test if you need a reminder.

Modify the Mapper to use a configuration parameter

4. Modify the `LetterMapper` class to

   a. Override the `setup` method to get the value of a configuration parameter called `caseSensitive`, and use it to set a member variable indicating whether to do case sensitive or case insensitive processing.

b. In the `map` method, choose whether to do case sensitive processing (leave the letters as-is), or insensitive processing (convert all letters to lower-case) based on that variable.

Pass a parameter programmatically

5. Modify the driver's `run` method to set a Boolean configuration parameter called `caseSensitive`. (Hint: Use the `Configuration.setBoolean` method.)

6. Test your code twice, once passing `false` and once passing `true`. When set to true, your final output should have both upper and lower case letters; when false, it should have only lower case letters.

Hint: Remember to rebuild your Jar file to test changes to your code.

Pass a parameter as a runtime parameter

7. Comment out the code that sets the parameter programmatically. (Eclipse hint: Select the code to comment and then select Source > Toggle Comment). Test again, this time passing the parameter value using `-D` on the Hadoop command line, e.g.:

```
$ hadoop jar toolrunner.jar stubs.AvgWordLength \
-DcaseSensitive=true shakespeare toolrunnerout
```

8. Test passing both `true` and `false` to confirm the parameter works correctly.

This is the end of the lab.

Lab: Using a Combiner

Files and Directories Used in this Exercise

Eclipse project: `combiner`

Java files:

`WordCountDriver.java` (Driver from WordCount)

`WordMapper.java` (Mapper from WordCount)

`SumReducer.java` (Reducer from WordCount)

Exercise directory: `~/workspace/combiner`

In this lab, you will add a Combiner to the WordCount program to reduce the amount of intermediate data sent from the Mapper to the Reducer.

Because summing is associative and commutative, the same class can be used for both the Reducer and the Combiner.

Implement a Combiner

1. Copy `WordMapper.java` and `SumReducer.java` from the `wordcount` project to the `combiner` project.

2. Modify the `WordCountDriver.java` code to add a Combiner for the WordCount program.

3. Assemble and test your solution. (The output should remain identical to the WordCount application without a combiner.)

This is the end of the lab.