Practice 5: Writing a Partitioner

Files and Directories Used in this Exercise

Eclipse project or package name: `practice_5`

Java files:

`MonthPartitioner.java` (Partitioner)

`ProcessLogs.java` (driver)

`CountReducer.java` (Reducer)

`LogMonthMapper.java` (Mapper)

Test data:

`testlog` (partial data set for testing)

In this Exercise, you will write a MapReduce job with multiple Reducers, and create a Partitioner to determine which Reducer each piece of Mapper output is sent to.

The Problem

In this program we will count the number of hits for each different IP address in month.. The final output will be 12 files, one for each month of the year: January, February, and so on. Each file will contain a list of IP address, and the number of hits from that address *in that month*.

We will accomplish this by having 12 Reducers, each of which is responsible for processing the data for a particular month. Reducer 0 processes January hits, Reducer 1 processes February hits, and so on.

Note: we are actually breaking the standard MapReduce paradigm here, which says that all the values from a particular key will go to the same Reducer. In this example, which is a very common pattern when analyzing log files, values from the same key (the IP address) will go to multiple Reducers, based on the month portion of the line.

Write the Mapper

Starting with `LogMonthMapper.java`, write a Mapper that maps a log file output line to an IP/month pair.

The Mapper should emit a Text key (the IP address) and Text value (the month). E.g.:

Input: `96.7.4.14 - - [24/Apr/2011:04:20:11 -0400] "GET /cat.jpg HTTP/1.1" 200 12433`

Output key: `96.7.4.14`

Output value: `Apr`

Hint: in the Mapper, you may use a regular expression to parse to log file data if you are familiar with regex processing. Otherwise, just copy the code from the `solution` package.

Remember that the log file may contain unexpected data – that is, lines that do not conform to the expected format. Be sure that your code copes with such lines.

Write the Partitioner

Modify the `MonthPartitioner.java` stub file to create a Partitioner that sends the (key, value) pair to the correct Reducer based on the month.

Remember that the Partitioner receives both the key and value, so you can inspect the value to determine which Reducer to choose.

Modify the Driver

1. Modify your driver code to specify that you want 12 Reducers.
2. Configure your job to use your custom Partitioner.

Test your Solution

Build and test your code in Eclipse. Use the testlog data as your input. Your output directory should contain 12 files named `part-r-000`*xx*. Each file should contain IP address and number of hits for month *xx*.

Hints:

- Write unit tests for your Partitioner!
- In Eclipse, you are using a small version of the access log in the data/`testlog`.
- The test data may not include all months, so some result files will be empty. □

To test your program against the cluster, remember to add the data to HDFS

Go to the directory with the data:

```
$ cd ~/Desktop/datasets
```

Create a new directory HDFS to hold your log data:

```
$ hadoop fs -mkdir weblog
```

Unzip the access_log file in datasets and put it into HDFS:

```
$ gunzip -c access_log.gz | hadoop fs -put - weblog/access_log
```

Run the hadoop fs -ls command to verify that the log file is in your HDFS home directory:

```
$ hadoop fs -mkdir testlog
```

Create a test file and put it into HDFS:

```
$ gunzip -c access_log.gz | head -n 5000 | hadoop fs -put - testlog/test_access_log
```

This is the end of the Exercise