**ICS 490: Big Data Storage**

**Homework Assignment #3**

**Due: See Syllabus Total: 50 Points**

In this assignment, you will analyze a log file from a web server to count the number of hits made from each unique IP address.

**Step 1:** Write the mapper, reducer, and driver code so that the final output of your program should be a file containing a list of IP addresses, and the number of hits from that address. The main idea is to examine the input data file to learn about the format of the input. Your mapper then will be mainly parsing an input line to extract the IP address. You can just discard lines that do not conform to the expected format.

**Step 2: Use a combiner** to reduce the number of data exchanged between mappers and reducers.

Use: job.setCombinerClass(XXXXXX.class);

In the Driver class. The XXXXXX is your combiner class name.

**Step 3: Use a counter** to count the number of times gifs, jpegs, and other resources have been retrieved. Your job will report three figures: number of gif requests, number of jpeg requests, and number of other requests. Use a counter group such as ImageCounter, with names gif, jpeg and other.

i.e: public static enum ImageCounter

**Step 4: Use a partitioner** to modify your final output such that, you will perform a similar task, but the final output should consist of 12 files, one each for each month of the year: January, February, and so on. Each file will contain a list of IP addresses, and the number of hits from that address in that month. You can 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 that you *may* need to change your reducer implementation)

Use: job.setPartitionerClass(XXXXXX.class);

job.setNumReduceTasks(12);

In the Driver class. The XXXXXX is your partitioner class name.

Why 12?

**Test data:** weblogs\_small

**Note: combiners, counters, and partitioners are additions to the basic map-reduce job and they will be covered in class. I highly recommend that you complete step 1 before, and use the second week to add features to your basic implementation.**

**What to Submit:** Upload to D2L files that contain:

* A files that include the following Java files for LogAnalysisMapper, LogAnalysisReducer, LogAnalysisDriver,LogAnalysisCombiner, and LogAnalysisPartitioner
* A word document with a table the compares the values of the counters that are displayed after running your job in the following cases:

Case 1: Just Mapper and reducer

Case 2: Mapper, combiner, and reducer

Case 3: Mapper, partitioner, and reducer

Case 4: Mapper, combiner, partitioner, and reducer

So Your table should have 4 columns as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Counter Name | Case 1 | Case 2 | Case 3 | Case 4 |

* Highlight table rows that have different values for the counters. For each highlighted row, explain briefly why the values are different.

.

**Example Table: see below (note the numbers are just made up, it is an example only)**

**Question 3:** Comparison table and Observation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Counter Name | Case 1 | Case 2 | Case 3 | Case 4 |
| FILE: Number of bytes read | 67 | 78 | xxxx | 23 |
| FILE: Number of bytes written | 67 | 78 | xxxx | 16 |
| HDFS: Number of bytes written | 67 | 78 | xxxx | 21 |
| Total time spent by all map tasks (ms) | 67 | 78 | xxxxx | 64 |
| Total time spent by all reduce tasks (ms) | 67 | 78 | xxxxx | 340 |
| Combine input records | 0 | 78 | 7 | 204 |
| Combine output records | 0 | 78 | 0 | 10 |
| Reduce shuffle bytes | 67 | 78 | xxxxx | 23 |
| Reduce input records | 67 | 78 | xxxxx | 30 |
| Reduce output records | 67 | 8 | xxxxx | 370 |
| Spilled Records | 67 | 2 | xxxx | 240 |
| Shuffled Maps | 16 | 18 | 9 | 8 |
| Merged Map outputs | 16 | 18 | 8 | 7 |
| Total committed heap usage (bytes) | 67 | 27 | 6640 | 380 |