Lab: Using SequenceFiles and File Compression

In this lab you will practice reading and writing uncompressed and compressed SequenceFiles.

First, you will develop a MapReduce application to convert text data to a SequenceFile. Then you will modify the application to compress the SequenceFile using Snappy file compression.

When creating the SequenceFile, use the full access log file for input data.

After you have created the compressed SequenceFile, you will write a second MapReduce application to read the compressed SequenceFile and write a text file that contains the original log file text.

Instructions

1. Input Data

   Unpack the access_log.gz file and upload it to HDFS. You can unpack the file with the following command:

   ```
   gunzip access_log.gz
   ```

2. Source code

   You will find the source files in the `src/stubs` directory. You will also find a `src/hints` directory that contains a copy of the source files modified to make the assignment easier to complete, and a `src/solution` directory that contains a copy of the source files modified to complete the assignment.

3. Complete the `CreateSequenceFile` and `ReadCompressedSequenceFile` classes in the `src/stubs` directory to transform web logs into sequence files and back to text.

4. Assemble and test your solution.

Write a MapReduce program to create sequence files from text files

1. Determine the number of HDFS blocks occupied by the access log file:
   1. In a browser window, start the Name Node Web UI. The URL is

      `http://localhost:50070`

   2. Click "Browse the filesystem."
   3. Navigate to the `/user/training/weblog/access_log` file.
   4. Scroll down to the bottom of the page. The total number of blocks occupied by the access log file appears in the browser window.
2. Complete the `CreateSequenceFile` stub file to read the access log file and create a SequenceFile. Records emitted to the SequenceFile can have any key you like, but the

values should match the text in the access log file. (Hint: You can use a Map-only job using the default Mapper, which simply emits the data passed to it.)

Note: If you specify an output key type other than `LongWritable`, you must call `job.setOutputKeyClass()` – *not* `job.setMapOutputKeyClass()`. If you specify an output value type other than `Text`, you must call `job.setOutputValueClass()` – *not* `job.setMapOutputValueClass()`.

3. Build and test your solution so far. Use the access log as input data, and specify the `uncompressedsf` directory for output.
4. Examine the initial portion of the output SequenceFile using the following command:

```
$ hadoop fs -cat uncompressedsf/part-m-00000 | less
```

Some of the data in the SequenceFile is unreadable, but parts of the SequenceFile should be recognizable:

   - The string `SEQ`, which appears at the beginning of a SequenceFile
   - The Java classes for the keys and values
   - Text from the access log file

5. Verify that the number of files created by the job is equivalent to the number of blocks required to store the uncompressed SequenceFile.

Compress the Output

6. Modify your MapReduce job to compress the output SequenceFile. Add statements to your driver to configure the output as follows:
   - Compress the output file.
   - Use block compression.
   - Use the Snappy compression codec.
7. Compile the code and run your modified MapReduce job. For the MapReduce output, specify the `compressedsf` directory.
8. Examine the first portion of the output SequenceFile. Notice the differences between the uncompressed and compressed SequenceFiles:
   - The compressed SequenceFile specifies the `org.apache.hadoop.io.compress.SnappyCodec` compression codec in its header.
   - You cannot read the log file text in the compressed file.
9. Compare the file sizes of the uncompressed and compressed SequenceFiles in the `uncompressedsf` and `compressedsf` directories. The compressed SequenceFiles should be smaller.

Write another MapReduce program to uncompress the files

10. Starting with the `ReadCompressedSequenceFile` stub file, write a second MapReduce

program to read the compressed log file and write a text file. This text file should have the same text data as the log file, plus keys. The keys can contain any values you like.

11. Compile the code and run your MapReduce job. For the MapReduce input, specify the `compressedsf` directory in which you created the compressed SequenceFile in the previous section. For the MapReduce output, specify the `compressedsftotext` directory.
12. Examine the first portion of the output in the `compressedsftotext` directory.

    You should be able to read the textual log file entries.

Optional: Use command line options to control compression

13. If you used ToolRunner for your driver, you can control compression using command line arguments. Try commenting out the code in your driver where you set the compress. Then test setting the `mapred.output.compressed` option on the command line, e.g.:

```
$ hadoop jar sequence.jar stubs.CreateSequenceFile \
-Dmapred.output.compressed=true weblog outdir
```

14. Review the output to confirm the files are compressed.