Using SequenceFiles and File Compression

Eclipse project: createsequencefile

Java files:

CreateSequenceFile.java (a driver that converts a text file to a sequence file)

ReadCompressedSequenceFile.java (a driver that converts a compressed sequence file to text)

Test data (HDFS): weblog (full web server access log)

Exercise directory: ~/workspace/createsequencefile

JAR File:

readsequencefile.jar

createsequencefile.jar

**In this exercise 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. (You uploaded the access log file to the HDFS ~/weblog directory when you performed the “Using HDFS” exercise.)

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.

**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 ~/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.

1. Complete the fixme file in the createsequencefile project 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 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.

1. Build and test your solution so far. Use the access log as input data, and specify the uncompressedsf directory for output.

NOTE: The CreateUncompressedSequenceFile.java file in the solution package contains the solution for the preceding part of the exercise.

1. Examine the initial portion of the output SequenceFile using the following command:

$ hdfs dfs -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.

1. 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**

1. 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.
2. Compile the code and run your modified MapReduce job. For the MapReduce output, specify the compressedsf directory.

NOTE: The CreateCompressedSequenceFile.java file in the solution package contains the solution for the preceding part of the exercise.

1. 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.

1. 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**

1. Starting with the provided fixme 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.
2. 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.

NOTE: The ReadCompressedSequenceFile.java file in the solution package contains the solution for the preceding part of the exercise.

1. 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**

1. 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 call setCompressOutput (or use the

solution.CreateUncompressedSequenceFile program). Then test setting the mapred.output.compressed option on the command line, e.g.:

$ yarn jar sequence.jar

solution.CreateUncompressedSequenceFile

-Dmapred.output.compressed=true

weblog *outdir*

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

**END**