UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
School of Professional & Executive Development

Big Data Management and Analytics
Session: Distributed Storage - Hadoop III

Lecturers: Petar Jovanovic and Josep Berbegal

1 Tasks To Do Before The Session

It is important that you: (1) carefully read the **instruction sheet** for this lab session, (2) introduce yourself to the **lab's main objectives**, (3) understand the **theoretical background**, and (4) get familiar with the **tools being used**.

2 Part A: Examples & Questions (15min)

In the first 15 minutes, we will first clarify the main objectives of this lab. We will then learn how HDFS stores your data in the disk and overview different possible formats to do so, including their benefits and overhead. Finally, we will also see how HDFS can compress your data to optimize the storage space.

3 Part B: In-class Practice (2h 45min)

In this exercise you will use the provided Java program in order to generate data with different formats. The run configuration arguments are the following (in the same order): *format number_of_tuples path_for_file*. Precisely, we will explore the following formats (in bold the argument that will be used in the run configuration):

1. Plain text (-plainText).
2. Sequence files (-sequenceFile).
3. Avro (-avro).
4. Parquet (-parquet).

3.1 Exercise 1 (1h): Writing and Reading in different formats

1. Generate files with 10000 rows as a plain text, and then as well with Sequence files, Avro, and Parquet formats, and insert them into HDFS. Run:

```
java -jar labo1.jar write -plainText 10000 wines.10000.txt
java -jar labo1.jar write -sequenceFile 10000 wines.10000.seq
java -jar labo1.jar write -avro 10000 wines.10000.avr
java -jar labo1.jar write -parquet 10000 wines.10000.prq
```

   Then, load these files into your HDFS using the put command.

2. Read each of your files, with the following standard commands (-cat):

```
hadoop-2.7.4/bin/hdfs dfs -cat wines.10000.txt
hadoop-2.7.4/bin/hdfs dfs -cat wines.10000.seq
hadoop-2.7.4/bin/hdfs dfs -cat wines.10000.avr
hadoop-2.7.4/bin/hdfs dfs -cat wines.10000.prq
```

   - **What do your files look like now? Are they readable?**

     Answer: Text file: fully readable
     Sequence file: partially readable
     Avro: non readable

   - **What do you think is happening?**

     Answer: Sequence file: partially readable, because the header is binary.
     Avro and parquet are both fully binary files

3. Now, read the file stored in the SequenceFile format again by means of the following command:

```
java -jar labo1.jar read -sequenceFile wines.10000.seq
```

   - **Is your file readable now?**

     Answer: Yes, it is readable.

   - **What do you think the first column represents?**

     Answer: It is the key.

   - **Is it something given automatically or do we need to specify it?**

     Answer: It needs to be specified at the moment the sequence file is being generated.

- **How is it built?**
  Hint: Follow the code at the classes MyHDFSSequenceFileWriter *and* MyHDFSSequenceFileReader *for answers.*

  Answer: Assigns one value to the key (either type_1, type_2 or type_3) based on the WineInfo and random generated value at Generator class

3.2 Exercise 2 (45min): Format overhead

3.2.1 Exercise 2.1

Now, generate 1 tuple for each format. To this end, use the appropriate run configuration for each. For instance, you would use the following parameters for the plain text case: `write -plainText 1 winesplainText.txt`.

Next, answer the following questions:

(a) **What is the exact size for each file?**

Answer: Text file: 133 bytes
Sequence file: 217 bytes
Avro: 888 bytes

(b) **For Sequence files, Avro, and Parquet what is the overhead ratio with respect to plain text (i.e., the result of the fraction sizeOfFormatX/sizeOfPlainText)?**

Answer: For avro, it is 6.67.
For sequence file, it is 1.63.
For parquet, it is 21.54

(c) **Why? Elaborate on why you observe this file size with respect to the properties of the used format.**

Answer: Sequence and avro file contain headers, and parquet contain header, footer and a metadata into each rowgroup

3.2.2 Exercise 2.2

Second, generate 1000 tuples for each format and answer the following questions:

(a) **What is the exact size for each file?**

Answer: Text file: 130979 bytes
Sequence file: 141026 bytes
Avro: 146259 bytes

(b) **For Sequence files, Avro and Parquet what is the fraction of overhead with respect to plain text?**

Answer: For avro, it is 1.07.
For sequence file, it is 1.11.
For parquet, it is 0.12

(c) **If the overhead ratio is different, explain why.**

Answer: The overhead ratio doesn't have that difference as the previous exercise, the overhead is much lower, because the header, which is the additional information

3.3 Exercise 3 (1h): Data compression

Fill table 1. Here you need to change (actually uncomment) the code at the class *MyHDFSSequenceFileWriter* (remember to recompile and to upload it) in order to write a sequence file of 2 million rows but by using different compression levels. **Finally explain the meaning of each compression level and discuss about the results you obtain.**

Level	Insertion time	Reading time	#Blocks	File size
NONE	0m11.8	0m5.9	3	28166
RECORD	0m9.4	0m5.3	2	20148
BLOCK	0m4.6	0m4.2	1	29485

Table 1: Compression on sequence files

Reasoning on Table 1 First of all, for the #Blocks above, the number of replication has not been set, so HDFS assumed the default replcation 3

Additional comments: The numbers in file size are in measured in bytes.