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 Hadoop-HDFS-0.1.jar write -plainText 10000 wines.10000.txt
java -jar Hadoop-HDFS-0.1.jar write -sequenceFile 10000 wines.10000.seq
java -jar Hadoop-HDFS-0.1.jar write -avro 10000 wines.10000.avr
java -jar Hadoop-HDFS-0.1.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: El plain text se muestra como una sucesión de filas y es legible. En cambio, el sequential file no es legible, ya que es un fichero binario. Avro también devuelve un

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

     Answer: Lo que sucede es que todos los formatos, excepto el texto plano, almacenan los datos en formato binario. Por lo tanto, es imposible leerlos directamente sin

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

```
java -jar Hadoop-HDFS-0.1.jar read -sequenceFile wines.10000.seq
```

   - **Is your file readable now?**

     Answer: El archivo en formato secuencial ahora sí es legible y es como estar consultando el texto plano

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

     Answer: La primera columna representa la clave de cada registro clave-valor dentro del sequence file

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

     Answer: Es algo especificado por el programa de escritura en nuestro caso es la columna Type

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

  Answer: Para cada registro, se genera un buffer para la clave, que será la columna Type, y otro buffer para el valor, que contiene el resto de las columnas especificadas

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 (the rest of the command is like in the previous cases): `write -plainText 1 winesplainText.txt`.

Next, answer the following questions:

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

Answer: Para el plein text són 128 bytes. Pare el sequence file són 219 bytes. Para Avro són 887 bytes. Para parquet són 2865 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: Para Avro el overhead es de 6.93, para parquet es de 22.38 y para sequence file es de 1.71

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

Answer: En el caso del Sequence File, debido a que contiene metadatos en el encabezado (header) y guarda sync markers que permiten la búsqueda de registros de

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: El tamaño de los archivos varía según el formato utilizado:Plain text: 131,465 bytes.Sequence File: 140,651 bytes.Avro: 146,192 bytes.Parquet: 32,183

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

Answer: Para Sequence File el overhead es de 1.07.Para Avro el overhead es de 1.11.Para Parquet el overhead es de 0.24

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

Answer: El overhead para todos los tipos de archivo se ha visto reducido, pero especialmente para Avro y Parquet, ya que guardan los datos en binario, lo que ocupa menos

3.3 Exercise 3 (1h): Data compression

Fill table 1. Here you need to change (actually uncomment) the code at the class *MyHDFSSequenceFileWriter*[1] (remember to recompile and redeploy it after the change) in order to write a sequence file of 2 million rows but by using different compression levels (line 43 - `NONE`, line 44 - `RECORD`, line 45 - `BLOCK`). Initially, it is set to NONE! Importantly, remember to recompile and redeploy the project, following the instructions in the first session, and produce new JAR file (for convenience you can rename it by adding the corresponding suffix: '*-NONE*', '*-RECORD*', *'-BLOCK'*).

Finally explain the meaning of each compression level and discuss about the results you obtain.

Level	Insertion time	Reading time	#Blocks	File size
NONE	13.546	2.479S	3	281.65
RECORD	8.392S	5.016S	2	192.12
BLOCK	4.669S	3.871S	1	2.17M

Table 1: Compression on sequence files

Reasoning on Table 1 El tiempo está medido en segundos y el tamaño del archivo en MB. La lectura del fichero se ha realizado utilizando el

[1] .../src/main/java/bdma/labos/hadoop/writer/MyHDFSSequenceFileWriter.java

Additional comments: