UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
School of Professional & Executive Development

Big Data Management and Analytics
Session: Introduction to the Hadoop ecosystem

Lecturers: Petar Jovanovic and Sergi Nadal

October 28th, 2016

1 Required Tools

- SSH and SCP client
- `eclipse` IDE with JDK 7 and Maven plugin installed

2 Tasks To Do Before The Session

As you will probably have already been told, laboratory sessions are thought to include some self-work material prior to the sessions themselves. However, we are still introducing the course so not previous work for this first session.

3 Part A: Examples & Questions (1h)

Today, we will have a 1-hour lecture called **Introduction to the Hadoop ecosystem**. Here, we will introduce the Big Data big picture, how the Hadoop ecosystem pops up in there and we will finally go through the details of its main technology stack.

4 Part B: In-class Practice (2h)

4.1 Exercise 1 (1h): Setting-up of Hadoop basics

Follow the enclosed guide to set up and start up a Hadoop/HDFS cluster to work on later on. Let the lecturer know when this is complete.

4.2 Exercise 2 (15min): Data generation

1. Compile the Java project you have been given. You can follow the instructions attached to this document.

2. Make a runnable JAR out of it (this JAR file is called *labo1.jar* in the examples below).

3. Upload it to your master node through SCP.

4. Generate, for instance, 1 GB of data and load it into HDFS. You are free to generate larger volumes of data, but note that will take more time on moving data from one node to another without adding much to your knowledge. Play a little bit and check on what information is displayed in the web interface.

```
hadoop-2.5.1/bin/hadoop jar labo1.jar write -standard -size 1 > wines.txt
hadoop-2.5.1/bin/hdfs dfs -put wines.txt
```

4.2.1 Exercise 3 (45min): Block splitting and replication

1. With replication factor of 1:

   (a) Load again the data file you previously generated into HDFS. How long did it take? First, remove the previous one.

   ```
   hadoop-2.5.1/bin/hdfs dfs -rm wines.txt
   time hadoop-2.5.1/bin/hdfs dfs -D dfs.replication=1 -put wines.txt
   ```

   (b) Now try to explore a little bit more on what has been going on. What is the size of the file in HDFS? How many blocks have been stored? What is the average block size? Discuss if such results make sense to you.

   ```
   hadoop-2.5.1/bin/hdfs fsck /user/bdma**/wines.txt
   ```

   (c) Now log into both DataNodes and try to explore the directory where you configured Hadoop to store the data. How much of the file is stored on each node?

   ```
   du -shx data/
   ```

2. With replication factor of 2:

   (a) Repeat the same steps as with replication factor of 1. Remove the previous file.

   ```
   hadoop-2.5.1/bin/hdfs dfs -rm wines.txt
   time hadoop-2.5.1/bin/hdfs dfs -D dfs.replication=2 -put wines.txt
   ```

   (b) Are there any differences in the results you obtain? If so, what are these differences?

3. With replication factor of 3:

(a) Now we start understanding how replication works, try to anticipate what is going to happen with replication factor of 3. What volume of data you expect to end up being physically stored?

(b) Repeat the same steps as with replication factor of 1 and 2. Remove the previous file.

```
hadoop-2.5.1/bin/hdfs dfs -rm wines.txt
time hadoop-2.5.1/bin/hdfs dfs -D dfs.replication=3 -put wines.txt
```

(c) Do these new results coincide with what you previously guessed? What do you think those under-replicated blocks mean? Why are they showing up now?

5 Part C: Optional Practice (3h)

5.1 Block size

Again, work with the same data file you previously generated and load it several times on the HDFS but with different block sizes. For instance, to load it with 32 MB block size and to read it, run:

```
time hadoop-2.5.1/bin/hdfs dfs -D dfs.blocksize=32m -copyFromLocal wines.txt
       wines.txt.32m
time hadoop-2.5.1/bin/hdfs dfs -cat wines.txt.32m > /dev/null
```

Complete table 1 and answer:

- Does the number of blocks make sense to you?
- What about the insertion and reading time?
- Imagine instead of having 1 GB file, you have a file occupying several TB. Would you use the default 128MB as block size? Why?

Block size	Insertion time	Reading time	Number of blocks
2 MB			
8 MB			
32 MB			
128 MB			
512 MB			
2 GB			

Table 1: Block size table

5.2 Sequence file and compression

This time let's explore a bit on sequence files and how compression works on their different levels.

1. Load 1 GB file with a sequence file format into HDFS. As we mentioned in the first hour, sequence files are made of key-values and therefore it is important to define the data type of such key-value before starting loading data. For the sake of simplicity, both values in this practical work are *Text* which is a Hadoop class to represent text. Run:

   ```
   hadoop-2.5.1/bin/hadoop jar labo1.jar write -hdfsSequence -size 1 wines.1g.seq
   ```

2. Take a look at it with the following command:

   ```
   hadoop-2.5.1/bin/hdfs -cat wines.1g.seq
   ```

   - What does it look like? Is it readable?
   - What do you think it is happening?

3. Now read the file again by means of the following command:

   ```
   hadoop-2.5.1/bin/hadoop jar labo1.jar read -hdfsSequence wines.1g.seq
   ```

   - Is it readable now?
   - What do you think the first column appearing is?
   - Is it something given automatically or do we need to specify it?
   - How is it built?
   - ***Hint: Follow the code at the classes* MyHDFSSequenceFileWriter *and* MyHDFSSequenceFileReader *for answers.***

4. Fill table 2. 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 1 GB size 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				
RECORD				
BLOCK				

Table 2: Compression on sequence files