UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
School of Professional & Executive Development

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

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: Objectives & Questions (15min)

In the first 15 minutes, we will first clarify the main objectives of this lab. We will then learn about the role of HDFS in the Big Data stack and introduce its main components. We will then overview how HDFS takes care of partitioning of input data and ensures the fault-tolerance in the cluster.

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

3.1 Exercise 1 (1h): Setting-up of Hadoop

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.

3.2 Exercise 2 (45min): Data generation

For the purpose of this lab session, we prepared a Java program that generates the data files of desired size, ready to be uploaded to HDFS. Follow carefully these instructions to obtain the project code, compile it and run it to generate the data.

1. **Obtain** the Java project code from `git` into your master node.
    - Enter your cluster and access the `master` node.
    - Get Hadoop-HDFS project code from `git` by executing the following command:

      ```
      git clone https://fpc-git.upc.es/upcschool-lab/hadoop-hdfs
      ```

As a result, you should be able to see the directory `hadoop-hdfs` inside your master.
Important: In the case you get `server certificate verification failed` error when running `git clone`, run the following com-mand to temporarily disable the certificate check for git and re-peat the above `git clone` command.

```
git config --global http.sslVerify false
```

After you finish the download, enable it again by running:

```
git config --global http.sslVerify true
```

2. If you do not have Java set up on your machine (i.e., if "`echo $JAVA_HOME`" command returns empty result), follow these guidelines to set it up.

   - **Only** if you do not have Java JDK 1.8 on you master node, first execute the following commands (it should be there from HDFS Setup process):

     ```
     cp tarballs/jdk-8u144-linux-x64.tar.gz ~/.
     tar xf jdk-8u144-linux-x64.tar.gz
     ```

   - Next, set up the `JAVA_HOME` environment variable (NOTE: you will have to set this up every time you startup your master!). Run the following command (*replacing ** with your bdma user*):

     ```
     export JAVA_HOME="/home/bdma**/jdk1.8.0_144/"
     ```

3. **Compile** the Java project and create the JAR file for generating data.

   - Enter to the `hadoop-hdfs` project directory

     ```
     cd ~/hadoop-hdfs
     ```

   - Once inside the project directory, run the following maven command to compile and build the project:

     ```
     mvn clean package
     ```

     The first time maven will download all required dependencies, thus it is very advisable to have enough free space in your master node. If the process succeeds (i.e., `BUILD SUCCESS` message), you will have the generated JAR in the `target` directory inside the project, specifically

```
~/hadoop-hdfs/target/Hadoop-HDFS-0.1.jar
```

- To work more conveniently with the geenrated JAR file, move it to the home of the master node and rename:

  ```
  mv ~/hadoop-hdfs/target/Hadoop-HDFS-0.1.jar ~/Hadoop-HDFS.jar
  ```

4. Once you have your JAR file produced, you can use it to **generate** data to be uploaded to HDFS. Generate, for instance, a file with 10 million rows 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.

   - First, **run** the JAR file to generate a data file of a given size in your master node:

     ```
     java -jar ~/Hadoop-HDFS.jar write -plainText 10000000 wines.txt
     ```

   - Finally, upload the generated file to HDFS:

     ```
     ~/hadoop-2.7.4/bin/hdfs dfs -put wines.txt
     ```

     Play a little bit and check what information is displayed in the web interface[1].

3.2.1 Exercise 3 (1h): Block splitting and replication

1. With replication factor of 1:

   (a) Load again the data file you previously generated into HDFS (first, remove the previous one).

   ```
   ~/hadoop-2.7.4/bin/hdfs dfs -rm wines.txt
   time ~/hadoop-2.7.4/bin/hdfs dfs -D dfs.replication=1 -put wines.txt
   ```

   How long did it take?

   Answer: 28.458s

   (b) Now try to explore a little bit more on what has been going on.

   ```
   ~/hadoop-2.7.4/bin/hdfs fsck /user/bdma**/wines.txt
   ```

   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.

   Answer: 1314.9 MB,10 blocks,131.4 MB

[1] After starting your HDFS cluster, the HDFS web interface should be available at port 50070. Thus, you should check your email to see what public address opens to your master:50070

(c) Now log into both DataNodes and try to explore the directory where you configured Hadoop to store the data.

```
du -shx ~/data/
```

How much of the file is stored on each node? Does such distribution of data benefits the parallelism of applications executing in the cluster?

Answer: 4 blocks en el slave1 y 6 blocks en el slave2.
Este distibución beneficia el paralelismo ya que puedes leer de los dos nodos a la vez y por lo tanto el la

2. With replication factor of 2:

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

   ```
   ~/hadoop-2.7.4/bin/hdfs dfs -rm wines.txt
   time ~/hadoop-2.7.4/bin/hdfs dfs -D dfs.replication=2 -put wines.txt
   ```

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

   Answer: Ahora tenemos los 10 blocks copiados en ambos nodos. Ahora es un poco mas optimo ya que como cada nodo tiene todos los blockes puedo repartir la

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 be physically stored?**

   Answer: Tendremos la misma cantidad de datos que con replica 2, es decir los 10 blocks en los dos nodos

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

   ```
   ~/hadoop-2.7.4/bin/hdfs dfs -rm wines.txt
   time ~/hadoop-2.7.4/bin/hdfs dfs -D dfs.replication=3 -put wines.txt
   ```

   (c) **Discuss the results obtained. What "under-replicated blocks" message means? Why is it showing up now?**

   Answer: No hay diferencias con el resultado esperado. Son el numero de blockes que no cumplen la replicidad que se ha indicado que en nuestro caso son los 10 blocks

---

Additional comments: Si alguna casilla no se muestra correctamente, es devido al interprete de pdf que se este usando