UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
School of Professional & Executive Development

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

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 (15h)

In the first 15 minutes, we will first clarify the main objectives of this lab. We will then learn more on how you can configure HDFS to partition your input data and why is this important. Then we will discuss the load balancing in HDFS cluster, how we can improve it, and see the scenarios when this is important.

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

3.1 Exercise 1 (1h): Block size

Work with the same data file you generated in the previous lab session (10 million rows) and load it on the HDFS, but with different block sizes. For instance, to load it with 32 MB block size, run:

```
hadoop-2.7.4/bin/hdfs dfs -D dfs.blocksize=32m -copyFromLocal wines.txt wines.txt.32m
```

And to read it (with obtaining reading time):

```
time hadoop-2.7.4/bin/hdfs dfs -cat wines.txt.32m > /dev/null
```

Complete table 1.

Note: Due to different factors that can introduce variability in reading/writing times from execution to execution, we repeat an operation and draw our conclusions from the obtained mean values.

Therefore, for each block size, first load data into HDFS, then execute reading operation 10 times and record in table 1 reading time for each execution (**RTi**). Analyze the obtained results and answer the following questions.

Block size	#blocks	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10
2 MB	628	19.730s	27.173s	23.050s	25.707s	16.444s	16.085s	18.467s	18.465s	16.608s	16.552s
32 MB	40	30.735s	17.716s	13.320s	10.536s	13..775s	12.409s	13.662s	12.861s	12.891s	11.282s
128 MB	10	18.333s	27.409s	10.381s	18.481s	12.128s	13.717s	9.565s	15.074s	13.192s	13.768s
2 GB	1	18.031s	19.357s	10.534s	19.745s	19.449s	10.248s	10.359s	10.095s	18.648 s	19.907s

Table 1: Block size table

- **Does the number of blocks make sense to you?**

  Answer: sí porque es en cada caso el redondeo superior de la division entre el tamaño del archivo y el tamaño definido de los bloques

- **What about the reading time? How is it affected by diffent block sizes?**

  Answer: sí porque es en cada caso el redondeo superior de la division entre el tamaño del archivo y el tamaño definido de los bloques
  Cuanto menos bloques mejor, hasta el limite de tener menos bloques que numero de nodos, que es el ultimo caso.

- **Imagine instead of having 1 GB file, you have a file occupying several TB. Would you use the default 128MB as block size? Why?**

  Answer: No porque tendría muchos bloques en cada nodo y el tiempo de lectura sería peor. Por ejemplo, con 1TB tendría 10.000 bloques.

3.2 Exercise 2 (1h): Load distribution in HDFS

Now, let us explore more about how HDFS distributes the data over cluster nodes and on what the load balance depends.

Besides previously generated file (10 million rows), generate also files with 10 000 and 1 million rows.

1. Load these files into HDFS with replication factor 1 (previously remove all the other files you previously loaded).

2. When loading data into HDFS, for each input size check different block sizes (16MB, 64MB, and 128MB).

Complete table 2.

Now, read the table and discuss how do you think HDFS is distributing data inside the cluster.

Answer: HDFS is radomly distributing blocks into the datanodes

What do you think affects the load distribution inside the cluster? In which cases the load is more balanced?

Input size (#rows)	**Block size**	**Load on slave1** #blocks	**Load on slave2** #blocks
10M	16MB		
10M	64MB		
10M	128MB		
1M	16MB		
1M	64MB		
1M	128MB		
10K	16MB		
10K	64MB		
10K	128MB		

Table 2: Load distribution table

Answer: The amount of blocks in which the file is divided (the more blocks the more balanced they can be distributed). They were more balanced when the block size was 16MB, because it divided the file into more blocks, aunque pensamos que fue por suerte y la tendencia es que el file de 10M con bloques de 16MB, que es el que tiene en total más bloques sea el más balanceado.

3.3 Exercise 3 (45min): Improving the load distribution

Next, we will try to use some HDFS admin tools to improve the distribution of data inside the cluster.

Notice that this is important in the cases when some of your cluster nodes are heavily overused (i.e., have considerably larger amounts of data in comparison to others).

This may happen either while loading/deleting your data from the cluster, and more noticeably when you dynamically add new nodes to your cluster in order to scale up.

To exemplify this, let us reconfigure our HDFS cluster to have only one node (slave1). Use the HDFS setup guidelines and follow these exact steps:

1. Log in to your master node.
2. Stop the HDFS cluster.
3. Exclude for now the slave2 node from the cluster. To do this, change the *hadoop-2.7.4/etc/hadoop/slaves* file to include only slave1.
4. Copy the new "slaves" configuration to slave1.
5. Start the HDFS cluster.

Once we reconfigured the cluster, let's load our data to HDFS. Load the previously generated file with 10 million rows to HDFS using only 1 replica.

```
hadoop-2.7.4/bin/hdfs dfs -D dfs.replication=1 -put wines.10M.txt
```

Next, let us add a new node to our cluster (Note that this is typically done to scale up the data processing by increasing the level of parallelism):

1. Stop the HDFS cluster.
2. Add the slave2 node to the cluster. To do this, change the *hadoop-2.7.4/etc/hadoop/slaves* file to include both slave1 and slave2.
3. Copy the new "slaves" configuration to both slave1 and slave2.
4. Start the HDFS cluster.

How many blocks are now stored on the slave1 and slave2 nodes?

Answer:

Do you think that with this setup we actually scaled up the processing of data from the previously loaded file? Explain why.

Answer:

Luckily, for such scenarios HDFS gives us an admin tool called Balancer that can improve the situation.

Let us now run the HDFS Balancer tool and see what happens[1] (Notice that we use 1% threshold to make the balancer more sensitive given the smaller size of data we are dealing with.)

```
hadoop-2.7.4/bin/hdfs balancer -threshold 1
```

How many blocks are now stored on the slave1 and slave2 nodes?

Answer:

Do you think that now we improved the situation? Explain why.

Answer:

Additional comments:

[1] Find more information about the balancer and its input parameters here: `https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HDFSCommands.html#balancer`