Xavier Gonzalez Arriola

Big Data Management and Analytics

Session: Introduction to the Hadoop ecosystem

Lecturers: Petar Jovanovic and Sergi Nadal

March 11th, 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. OK

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

3. Upload it to your master node through SCP.

```
bdma08@cluster-rdlab:~$ scp labo1.jar bdma08@node758:.
```

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
real    0m17.742s
user    0m10.208s
sys     0m2.307s
```

(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/bdma08/wines.txt
       ...Total size:        1081969162 B
       ...Total blocks (validated):  9 (avg. block size 120218795 B)

...
```

(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?

```
bdma08@slave1:~$ du -shx data/
517M    data/
bdma08@slave2:~$ du -shx data/
524M    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?

Yes, since factor of 2 is used, 2 replicas are stored, then double size is used overall.

```
bdma08@slave1:~$ du -shx data/
1,1G    data/
bdma08@slave1:~$
du -shx data/
1,1G    data/
```

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?

With 3 replications we would expect to make file creation time longer, which it is, however it will not get to 3 replications since there are only 2 DataNodes, then as we can check only 2 replications are made by hadoop with corresponding under-replicated blocks.

```
Status: HEALTHY
Total size:    1081969162 B
Total dirs:    0
Total files:   1
Total  symlinks:             0
Total blocks (validated):    9 (avg. block size 120218795 B)
Minimally replicated blocks: 9 (100.0 %)
Over-replicated blocks:      0 (0.0 %)
Under-replicated blocks:     9 (100.0 %)
Mis-replicated blocks:       0 (0.0 %)
Default replication factor:  3
Average block replication:   2.0
Corrupt blocks:              0
Missing replicas:            9 (33.333332 %)
Number of data-nodes:        2
Number of racks:             1
```

What do you think those under-replicated blocks mean? Why are they showing up now?

Under-replicated blocks are blocks whith replications number under the specified number (dfs.replication=3 in this case). Then less DataNodes than specified replications number leads to the 9 under-replicated blocks for this case that would be replicated in a third DataNode.