TP2: Deploying and configuring *Yarn* on a multi node

Alessio Pagliari and Shadi Ibrahim
March, 2022

The goal of this TP is to study the implementation and the operation of the Hadoop 3 Platform using Yarn as a resource manager. Deploy the platform on multiple nodes. Start exploring the HDFS by adding and removing files. Finally, we will run simple examples using the MapReduce paradigm.

Exercise 1: Installing Yarn platform on Grid5000

The goal of this exercise is to learn how to set up and configure a multi-node Yarn installation so that you can quickly perform simple operations using Hadoop MapReduce.

Question 1.1

First make a reservation to a Grid5000 cluster. You can check available nodes in the various sites by the following website: https://intranet.grid5000.fr/oar/Rennes/drawgantt-svg/. Change the site name to check availabilities in different sites.

For a better TP experience it would be best to reserve three nodes, reserve them for the duration of the TP (2 hours):

```
$ oarsub -I -p "cluster='sagittaire'" -l /host=3,walltime=2:0:0 -t deploy
```

This time deploy a Debian 11 image:

```
$ kadeploy3 -f $OAR_NODE_FILE -e debian9-x64-base -k
```

Question 1.2

Required software for Hadoop on Linux include:

- Java must be installed.
- ssh must be installed and sshd must be running to use the Hadoop scripts that manage remote Hadoop daemons.

Configure one of the two nodes, remember that to connect to a node you need to do `ssh root@nodename`.

Install Java and configure system variables:

```
# apt-get update
# apt-get install default-jdk
```

You can check if the ssh daemon runs using the `ps` command:

```
# ps aux | grep sshd
```

If needed, you can install ssh as follows:

```
# sudo apt-get install ssh
```

To avoid typing your password each time you start the Hadoop daemons, you should create in each node a couple of `ssh` keys thanks to the `ssh-keygen` command (if not already done so), then add the public key to the authorized keys. This can be done from the site frontend:

```
$ ssh root@node-1 'ssh-keygen -t rsa -P ""'
$ ssh root@node-2 'ssh-keygen -t rsa -P ""'
...
$ ssh root@node-1 'cat ~/.ssh/id_rsa.pub' | ssh root@node-1 'cat >> ~/.ssh/authorized_keys'
$ ssh root@node-2 'cat ~/.ssh/id_rsa.pub' | ssh root@node-1 'cat >> ~/.ssh/authorized_keys'
...
```

N.B. you should copy the public key of each node inside the `authorized_keys` files of every node.

Check the success of this step by connecting in ssh between your nodes
When running this command, you should not be prompted to type any password.

Question 1.3

Now download hadoop-3.3.2.tar to the master node, to do so use the following command:

```
wget https://dlcdn.apache.org/hadoop/common/hadoop-3.3.2/hadoop-3.3.2.tar.gz
```

To run Hadoop needs a `JAVA_HOME` environment variable specifies the directory of your `JVM`. Add the following lines at the end of your `$HOME/.bashrc` file:

```
export JAVA_HOME=<java path>
export HADOOP_HOME=<your Hadoop directory path>
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
export HDFS_NAMENODE_USER="root"
export HDFS_DATANODE_USER="root"
export HDFS_SECONDARYNAMENODE_USER="root"
export YARN_RESOURCEMANAGER_USER="root"
export YARN_NODEMANAGER_USER="root"
```

To make take into account these changes in your open terminal type:
`. source ~/.bashrc`

Question 1.4

Before starting the Yarn platform, edit its configuration files located in the `etc/hadoop/` folder. As you are using your own machine, you must configure Hadoop as follows:

- In `workers` file specify the name of your worker machines: `node-2` and `node-3`
- In `etc/hadoop/hadoop-env.sh` set the JAVA_HOME variable consistently with your environment, for instance:

  Change
  `# export JAVA_HOME=/usr/lib/j2sdk1.5-sun`
  To
  `export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64`

- In `etc/hadoop/core-site.xml` add:

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://node-1:9000</value>
  </property>
</configuration>
```

  This indicates the name of the machine and the associated network port on which the NameNode process will run. You should check that the 9000 port on your machine is not already in use:
  `netstat -a | grep tcp | grep LISTEN | grep 9000`

- In `etc/hadoop/hdfs-site.xml` add:

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
</configuration>
```

- In `etc/hadoop/mapred-site.xml` add:

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
  <property>
    <name>yarn.app.mapreduce.am.env</name>
    <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
  </property>
  <property>
    <name>mapreduce.map.env</name>
    <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
  </property>
  <property>
    <name>mapreduce.reduce.env</name>
    <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
  </property>
</configuration>
```

- In `etc/hadoop/yarn-site.xml` add:

```
<configuration>
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>node-1</value>
  </property>

  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>

  <property>
    <name>yarn.nodemanager.aux-services.mapreduce_shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
  </property>
</configuration>
```

Finally, **copy** the hadoop folder from node-1 to node-2 or repeat the configurations on the second node.

Question 1.5

We will now start the platform, and start all daemons:

- Format a new distributed-filesystem:
  $ `bin/hdfs namenode -format`
- Start the HDFS daemons:
  $ `sbin/start-dfs.sh`
- The hadoop daemon log output is written to the `HADOOP DIRECTORY PATH/logs`. Check them
- A nice command for checking if the expected Hadoop processes are running is `jps`. Try it when the HDFS processes are running!
- Start the ResourceManager daemon and NodeManager daemons:
  $ `sbin/start-yarn.sh`

Question 1.6

To Browse the web interface for the NameNode and the ResourceManager; by default they are available at:

- ResourceManager - http://localhost:8088/
- NodeManger - http://localhost:8042/

NOTE: you need to create an ssh tunnel like you did in TP1:

```
$ ssh <username>@access.grid5000.fr -N -L50070:<hadoop_master>.<site>.grid5000.fr:50070
```

Exercise 2: Using the Hadoop distributed file System (HDFS)

The goal of this exercise is to learn how to perform simple operations using the Hadoop Distributed File System (HDFS).

The FileSystem (FS) shell is invoked by `bin/hadoop fs <args>`. All the FS shell commands take path `URIs` as arguments. The `URI` format is `scheme://autority/path`. For HDFS the scheme is `hdfs`, and for the local filesystem the scheme is `file`. The scheme and authority are optional: *an HDFS file or directory such as `/parent/child` can be specified as*

- `hdfs://namenodehost/parent/child` OR as
- `/parent/child`

FS shell:

```
hdfs dfs -cat URI [URI ...] –Copies source paths to stdout.
hdfs dfs -copyFromLocal <localsrc> URI
hdfs dfs -copyToLocal [-ignorecrc] [-crc] URI <localdst>
hdfs dfs -get [-ignorecrc] [-crc] <src> <localdst>
hdfs dfs -ls <args>
hdfs dfs -mkdir <paths>
hdfs dfs -put <localsrc> ...  <dst>
hdfs dfs -rmr URI [URI ...]
```

Question 2.1

During start up the NameNode loads the file system state from the fsimage and the edits log file. It then waits for DataNodes to report their blocks so that it does not prematurely start replicating the blocks though enough replicas already exist in the cluster. During this time NameNode stays in Safemode. Safemode for the NameNode is essentially a read-only mode for the HDFS cluster, where it does not allow any modifications to file system or blocks. Normally the NameNode leaves Safemode automatically after the DataNodes have reported that most file system blocks are available.

```
bin/hdfs dfsadmin -safemode get/enter/leave
```

Question 2.2

We will now generate (in the local file system) three files loremIpsum-2000, loremIpsum-1000 and loremIpsum-100. Their sizes are 2000MB, 1000MB and 100MB respectively. For this you use three times `loremIpsum` and `generator.sh` script file provided (also available here: http://people.rennes.inria.fr/Shadi.Ibrahim/TP1-Resources.zip):

- `./generator.sh loremIpsum 2000`
- `./generator.sh loremIpsum 1000`
- `./generator.sh loremIpsum 100`

Now copy the three files to HDFS (`put`) and then show their content (`cat`) and finally remove them (`rmr`).

Exercise 3: Running a MapReduce program on Yarn

The goal of this exercise is to execute two MapReduce examples, typically used for

benchmarking, which come with the default Hadoop distribution. At this point we will simply run the compiled programs (although it is not necessary for this exercise, if you are curious to see how the map and reduce functions are programmed, the code of this programs is available at the Github page:
https://github.com/apache/hadoop/tree/trunk/hadoop-mapreduce-project/hadoop-mapreduce-examples/src/main/java/org/apache/hadoop/examples).

Question 3.1
Run the wordcount example:

- `bin/hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-examples-*.jar wordcount input output`

where input is the directory containing the input files (e.g., loremIpsum-2000) while the output is the directory that will be created to store the results.

Use the 1000 MB data set and check the output. Then, inspect the log files generated by this job.

Test also the other files of different size and compare the performance.
(Optional) You can also try to generate larger datasets, e.g. 5GB or 10GB.

Question 3.2
Run the gep example:

- `bin/hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-examples-*.jar grep input output 'keyword'`

where input is the directory containing the input files (e.g., loremIpsum-1000) while the output is the directory that will be created to store the results.

Question 3.3 (Optional)
Run the Pi estimator:

- `bin/hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-examples-*.jar pi maps samples_per_map`

where maps represents the number of map tasks used and *samples_per_map* the number of points processed by each map task (i.e. the number of reducers).

Run the Pi estimator with 20 maps and 10000 *samples_per_map*. Check the log files.

Exercise 4 (Optional): Prepare for future deployments

Question 4.1

Create a disk image of the configured machine:

From g5k site frontend:

```
$ tgz-g5k -m <configured_node> -f my-hadoop-image.tgz
$ kaenv3 -p debian11-x64-base -u deploy > my-hadoop-environment.yaml
```

Open and edit `my-hadoop-environment.yaml` by changing: *name*, *description*, *author* and *image file* to point to the previously created tgz image.

Test the re-deployment with the following command:

```
$ kadeploy3 -f $OAR_NODE_FILE -a my-hadoop-env.yaml -k
```

Question 4.2

Create a script that automatically:

- configure the ssh keys on the reserved nodes
- configure the nodes addressed on Hadoop configuration file and distribute it on all nodes
- start remotely the Hadoop cluster
- create the ssh bridges to connect to the web interface

A skeleton for the script with some basics pre-configuration is available at `/home/alpagliari/public/DeployHadoopYarn.sh` **from the Rennes site**, otherwise available at the following link: http://people.rennes.inria.fr/Shadi.Ibrahim/DeployHadoopYarn.zip. In the file you will find the list of TODOs to follow.

Now you're ready to deploy everything in TP3!