TP Spark scalability

Daniel Hagimont
hagimont@enseeiht.fr

The goal of this labwork is to experiment with Spark in cluster mode and to evaluate its scalability.

1. Installation

- pre-requisite
    - you should have Java installed and the JAVA_HOME variable defined
    - you should have your ssh keys configured to allow ssh to localhost
- install Hadoop
    - untar the hadoop-2.7.1.tar.gz archive
    - define environment variables

```
        export HADOOP_HOME=<path>/hadoop-2.7.1
        export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH
```

- install Spark

    untar the spark-2.4.3-bin-hadoop2.7.tgz archive
    - define environment variables

```
        export SPARK_HOME=<path>/spark-2.4.3-bin-hadoop2.7
        export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
```

```
hadoop-2.7.1/etc/hadoop/hadoop-env.sh
------------------------------------ ----------
update the JAVA_HOME variable :
export JAVA_HOME=<path-to-java>
```

```
hadoop-2.7.1/etc/hadoop/core-site.xml
------------------------------------ ---------
<configuration>
<property>
   <name>hadoop.tmp.dir</name>
   <value>/tmp/hadoop</value>
   <description>A base for other temporary directories.</description>
</property>
<property>
   <name>fs.defaultFS</name>
   <value>hdfs://master:54310</value>
</property>
</configuration>
```

HERE UPDATE THE MASTER NODE NAME

```
hadoop-2.7.1/etc/hadoop/hdfs-site.xml
----------------------------------- ---------
<configuration>
<property>
   <name>dfs.replication</name>
   <value>1</value>
</property>
<property>
       <name>dfs.block.size</name>
       <value>67108864</value>
</property>
</configuration>
```

```
hadoop-2.7.1/etc/hadoop/slaves
----------------------------- --------
slave1
slave2
```

HERE UPDATE THE SLAVES' NODE NAMES

```
spark-2.4.3-bin-hadoop2.7/conf/slaves.template
-------------------------------------------------------
cp  slaves.template  slaves
add in this file :
slave1
slave2
```

HERE UPDATE THE SLAVES' NODE NAMES

```
spark-2.4.3-bin-hadoop2.7/conf/spark-env.sh.template
---------------------------------------------------------------
cp spark-env.sh.template spark-env.sh
add in this file :
export SPARK_MASTER_HOST=master
# sometimes I had to add an "export JAVA_HOME=…." in this file
```

HERE UPDATE THE MASTER NODE NAME

2. Execution

You also have to replace the master node name in the URL of the file in the WordCount application.

HERE UPDATE THE MASTER NODE NAME

You are given a set of scripts which help starting everything :

comp.sh
------- ---
compile the WordCount.java application

generate.sh

# generate a file data.txt of size 2^x
source generate.sh filesample.txt x
# for example a file of size 8 Gb
source generate.sh filesample.txt 23
# the file "data.txt" is generated in /tmp

start.sh
----
you have to edit the script to update the name of the 2 slaves
starts the overall hadoop cluster (hdfs + spark)
verify with jps that you have
- on slave nodes : one DataNode daemon (hdfs) and one Worker daemon (spark)
- on the master node : one NameNode, one SecondaryNameNode daemon (hdfs) and one Master daemon (spark)

If everything went right, you should be able to observe the datanodes on http://master:50070 and the workers on http://master:8080

copy.sh
-------- -
store the generated file "/tmp/data.txt" in hdfs in /input (/tmp/data.txt)
you can observe the creation of blocks on http://master:50070

run.sh
----
you have to edit the script to update the name of the master
Then, run the script to execute the application

stop.sh
----
stop all the daemons

3. Evaluation

You can manage to produce a file of 8 Gb.
You can evaluate the performance of the WordCount application for 1 slave, 2 slaves, 3 slaves

You can compare with the performance of an iterative version (Count.java).
You can evaluate the performance for 1 slave with 1 code (you must start one slave with "start-slave.sh <url of master> -c 1" on the slave node, instead of using my start.sh script)
You can also experiment with larger files and more slaves.

I obtained : sequential (115s), 1 core (261s), 1 node (83s), 2 nodes (47s)