Lab1 - HDFS, HBase, and Cassandra

Amir H. Payberah
payberah@kth.se

1 Introduction

In this lab assignment you will practice with HDFS, a popluar distributed filesystem, HBase and Cassandra, two known NoSQL databases. Note that the following instruction is tested on a Linux operating system, so if you do not have Linux, you need to install it either on your machine or on a VirtualBox. You can download VirtualBox from its page. You can also find different ready to use Linux distribution images for VirtualBox here. You may get some errors in this task. The last section listed some of them.

2 Installing HDFS

This section presents the steps you need to do to install Apache Hadoop.

1. Download and install Java SDK 8. You can download it from the following link:
   https://www.oracle.com/uk/java/technologies/javase/javase8-archive-downloads.html
2. Download Hadoop MapReduce 3.2.4 from the following link:
   https://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-3.2.4/hadoop-3.2.4.tar.gz
3. Set the following environment variables.

```
export JAVA_HOME=<path to the Java folder>
export HADOOP_HOME=<path to the Hadoop folder>
```

4. Modify `$HADOOP_HOME/etc/hadoop/hadoop-env.sh` as below.

```
export JAVA_HOME=<path to the Java folder>
export HADOOP_LOG_DIR=<path to the Hadoop folder>/hadoop_repo/logs/hadoop
```

5. Modify `$HADOOP_HOME/etc/hadoop/core-site.xml` as below (`hadoop_repo` is a temporary directory, make it if it does not exist).

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
  </property>
  <property>
    <name>hadoop.tmp.dir</name>
    <value><path to the Hadoop folder>/hadoop_repo</value>
  </property>
</configuration>
```

6. Modify `$HADOOP_HOME/etc/hadoop/hdfs-site.xml` as below (There is only one machine here, and the copy can be temporarily configured as "1").

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

7. Modify `$HADOOP_HOME/etc/hadoop/mapred-site.xml` as below (This configuration specifies that the specific execution engine is "yarn").

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
```

8. Modify `$HADOOP_HOME/etc/hadoop/yarn-site.xml` as below.

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

9. Format HDFS! If the format is successful, it cannot be reformatted. Repeated formatting will cause problems. But, if you need to perform formatting multiple times, it is recommended to delete `$HADOOP_HOME/hadoop_repo` contents.

```
$HADOOP_HOME/bin/hdfs namenode -format
```

10. Start and verify Hadoop!

```
$HADOOP_HOME/sbin/start-all.sh
```

If all the above steps are done correctly, you should see the running processing. To do so, execute the `jps` in a terminal to print out the HDFS running processes. View the HFDSs service in browser on `http://127.0.0.1:9870`. To browse HDFS file system directories select `Utilities > Browse the file system`. To view the YARN Service in the browser: http://127.0.0.1:8088. Now, let's try some HDFS commands:

- Create a new directory `/kth` on HDFS

```
$HADOOP_HOME/bin/hdfs dfs -mkdir /kth
```

- Create a file, call it `big`, on your local filesystem and upload it to HDFS under `/sics`

```
touch big
$HADOOP_HOME/bin/hdfs dfs -put big /kth
```

- View the content of `/kth` directory

```
$HADOOP_HOME/bin/hdfs dfs -ls /kth
```

- Determine the size of file `big` on HDFS

```
$HADOOP_HOME/bin/hdfs dfs -du -h /kth/big
```

- Print the first 5 lines of the file `big` to screen (the `big` file is empty, so you can add some lines of text to it before uploading it on the HDFS)

```
$HADOOP_HOME/bin/hdfs dfs -cat /kth/big | head -n 5
```

- Make a copy of the file `big` on HDFS, and call it `big_hdfscopy`

```
$HADOOP_HOME/bin/hdfs dfs -cp /kth/big /kth/big_hdfscopy
```

- Copy the file `big` to the local filesystem and name it `big_localcopy`

```
$HADOOP_HOME/bin/hdfs dfs -get /kth/big big_localcopy
```

- Check the entire HDFS filesystem for inconsistencies/problems

```
$HADOOP_HOME/bin/hdfs fsck /
```

- Delete the file `big` from HDFS

```
$HADOOP_HOME/bin/hdfs dfs -rm /kth/big
```

- Delete the folder `/kth` from HDFS

```
$HADOOP_HOME/bin/hdfs dfs -rm -r /kth
```

3 Installing HBase

Here, we explain how to install HBase in *pseudo-distributed* mode, where all daemons run on a single node

1. Install the SSH-server.

```
sudo apt install openssh-server
```

2. Download Apache HBase 2.4.13 from the following link:
   https://www.apache.org/dyn/closer.lua/hbase/2.4.13/hbase-2.4.13-bin.tar.gz
3. Set the following environment variable.

```
export HBASE_HOME=<path to the HBase folder>
```

4. Modify `$HBASE_HOME/conf/hbase-env.sh` as below.

```
export JAVA_HOME=<path to the Java folder>
```

5. Make a folder on local file system, where zookeeper stores its data.

```
mkdir -p $HBASE_HOME/zookeeper
```

6. Modify `$HBASE_CONF/hbase-site.xml` as below.

```
<configuration>
  <property>
    <name>hbase.rootdir</name>
    <value>hdfs://localhost:9000/hbase</value>
  </property>

  <property>
    <name>hbase.zookeeper.property.dataDir</name>
    <value>/home/amir/Downloads/workspace/hbase-2.4.13/zookeeper</value>
  </property>

  <property>
    <name>hbase.cluster.distributed</name>
    <value>true</value>
  </property>
</configuration>
```

7. Start HBase with the following command. Before starting it, make sure that Hadoop namenode and datanodes are running.

```
$HBASE_HOME/bin/start-hbase.sh
```

8. HBase creates its directory in HDFS. To see the created directory type the following command.

```
$HADOOP_HOME/bin/hdfs dfs -ls /hbase
```

Now, let's go through the following steps to create a table in HBase and test it.

- Start the HBase shell

```
$HBASE_HOME/bin/hbase shell
```

- Create a table called `test` with the column family `cf`

```
# ' is a single quotation
create 'test', 'cf'
```

- Use the command `describe` to get the description of the table

```
describe 'test'
```

- Print the information about your table

```
list 'test'
```

- Put data into your table (the first insert is at `row1`, column `cf:a`, with a value of `value1`). Columns in HBase are comprised of a column family prefix, e.g., `cf`, followed by a colon and then a column qualifier suffix, e.g., `a`.

```
put 'test', 'row1', 'cf:a', 'value1'
put 'test', 'row2', 'cf:b', 'value2'
put 'test', 'row3', 'cf:c', 'value3'
```

- Scan the table for all data at once

```
scan 'test'
```

- To get a single row of data at a time, use the `get` command

```
get 'test', 'row1'
```

- If you want to delete a table or change its settings, as well as in some other situations, you need to disable the table first, using the `disable` command. You can re-enable it using the `enable` command

```
disable 'test'
enable 'test'
```

- To delete a table, use the `drop` command.

```
disable 'test'
drop 'test'
```

- Exit the HBase shell

```
exit
```

4 Installing Cassandra

Here, we show how to install and test Cassandra.

1. Download Cassandra 3.11.13 from the following link
   https://dlcdn.apache.org/cassandra/3.11.13/apache-cassandra-3.11.13-bin.tar.gz
2. Download and install Anaconda. You can download it from the following link:
   https://www.anaconda.com/products/distribution
3. Set the following environment variables.

```
export CASSANDRA_HOME=<path to the Cassandra folder>
export PYTHONPATH=<path to the Python folder>
```

4. Start Cassandra in the foreground.

```
$CASSANDRA_HOME/bin/cassandra -f
```

   After running the above command you may get an error as below:

```
Error occurred during initialization of VM
Could not reserve enough space for object heap
Error: Could not create the Java Virtual Machine.
Error: A fatal exception has occurred. Program will exit.
```

   This is due to heap memory that Cassandra wants to set, which is by defauly 1/4 of the total memory of the machine. You can it by editing `$CASSANDRA_HOME/etc/cassandra/default.conf/cassandra-env.sh` as below:

```
MAX_HEAP_SIZE="2048M"
HEAP_NEWSIZE="400M"
```

5. Start the `cqlsh` prompt.

```
$CASSANDRA_HOME/bin/cqlsh
```

6. Now, let's create a *keyspace*. A keyspace is similar to a schema/database in the RDBMS world. To create a keyspace execute the following CQL command. The `WITH REPLICATION` part of the command states that the `wordcount_keyspace` keyspace should use a simple replication strategy and will only have one replica for all data inserted into the keyspace.

```
create keyspace wordcount_keyspace
      with replication = {'class': 'SimpleStrategy', 'replication_factor': 1};
```

7. Print the list of keyspaces in Cassandra.

```
describe keyspaces;
```

8. Now let's create a column family. In order to create a column family, first you will need to navigate to the `wordcount_keyspace` keyspace. Then, create the `Words` table.

```
use wordcount_keyspace;
create table Words (word text, count int, primary key (word));
```

9. Insert a row into the above column family.

```
insert into Words(word, count) values('hello', 5);
```

10. Now, let's look at the table now.

```
select * from Words;
```

Let's examine what happened as a result of creating and inserting a row into the `Words` table. This will require us to flush data from a memtable to disk thus creating an SSTable on disk. We will use a utility called `nodetool` to help us flush data to disk.

```
$CASSANDRA_HOME/bin/nodetool flush wordcount_keyspace
```

Here is how Cassandra stores data (all keyspace and SSTable related data) on disk. By default it stores data at `data/data` in `$CASSANDRA_HOME`. However you can change it, by updating the `data_file_directories` at `$CASSANDRA_HOME/conf/cassandra.yaml`. The directory structure and component files have the following structure:

- `Data.db`: this is the base data file for the SSTable. All other SSTable related files can be generated from this file.
- `CompressionInfo.db`: it holds information about the uncompressed data length.
- `Filter.db`: the serialized bloom filter.
- `Index.db`: an index to the row keys with pointers to their position in the data file.
- `Summary.db`: SSTable index summary.
- `Statistics.db`: statistical metadata about the content of the SSTable.
- `TOC.txt`: a file which contains a list of files outputted for each SSTable.

Now, we can see what the underlying format looks like. The `sstabledump` is a utility that can be used to convert a binary SSTable file into a JSON. Let's convert data inserted into our `Words` table into JSON. You should replace the `<NUMS>` with what you see on your machine.

```
chmod +x $CASSANDRA_HOME/tools/bin/sstabledump
$CASSANDRA_HOME/tools/bin/sstabledump $CASSANDRA_HOME/data/data/wordcount_keyspace/words-<NUMS>/me-1-big-Data.db
```

5 Some Error Messages

- Port 22: Connection refused. To solve:
  1. Remove SSH.

```
sudo apt-get remove openssh-client openssh-server
```

2. Install SSH again.

```
sudo apt-get install openssh-client openssh-server
```

- Hadoop "Permission denied (publickey,password,keyboard-interactive)" warning. To solve:
  1. Generate a new keygen.

```
ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
```

  2. Register the new keygen.

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```