Installing Hadoop

Emanuele Fusco (`fusco@di.uniroma1.it`)

Prerequisites

You need a *nix system (Linux, Mac OS X, …) with a working installation of Java 1.7, either OpenJDK or the Oracle JDK. See, e.g.:

`http://www.webupd8.org/2012/01/install-oracle-java-jdk-7-in-ubuntu-via.html`

for installation instructions of the Oracle JDK on Ubuntu Linux. Otherwise, follow the standard instructions on the Oracle website:

`http://www.oracle.com/technetwork/java/javase/downloads/index.html`

You can check your current Java version by typing `java -version`

Install instructions

Download Hadoop 2.6.0 (i.e., file `hadoop-2.6.0/hadoop-2.6.0.tar.gz`) from you favorite mirror:

`http://www.apache.org/dyn/closer.cgi/hadoop/common/`

From now on assume we are user `john`, installing hadoop on our home folder `/home/john`

Extract the downloaded archive

```
tar xvf hadoop-2.6.0.tar.gz

mv hadoop-2.6.0 ~/
```

Set system variables and the path for executable files:

```
nano ~/.profile
```

and add the following lines:

```
export YARN_HOME=/home/john/hadoop-2.6.0

export PATH=$YARN_HOME/bin:$YARN_HOME/sbin:$PATH

export HADOOP_HOME=$YARN_HOME

export HADOOP_MAPRED_HOME=$YARN_HOME

export HADOOP_COMMON_HOME=$YARN_HOME

export HADOOP_HDFS_HOME=$YARN_HOME

export HADOOP_CONF_DIR=$YARN_HOME/etc/hadoop
```

Be careful to change `/home/john/` appropriately into your installation path.

Also, add to `.profile` the path to your Java installation. For OS X the following line should work:

```
export JAVA_HOME=$(/usr/libexec/java_home)
```

For Ubuntu and Oracle JDK the following line should work:

```
export JAVA_HOME=/usr/lib/jvm/java-7-oracle
```

For the changes made in the `.profile` file to become effective, open a new shell (or log out and back in). To check if everything is ok, try: `cd $YARN_HOME`

The `JAVA_HOME` export should also be repeated in the `hadoop-env.sh` file:

```
cd $YARN_HOME/etc/hadoop/

nano hadoop-env.sh
```

and change the line containing text `export JAVA_HOME=${JAVA_HOME}`

to match the export for `JAVA_HOME` made in the `.profile` file.

Preparing folders for the HDFS

We will use folder `/home/john/yarnData/` for the distributed filesystem of Hadoop.

First create the folders:

```
mkdir ~/yarnData

mkdir ~/yarnData/namenode

mkdir ~/yarnData/datanode
```

Then tell Hadoop where the namenode and datanode folders are:

```
cd $YARN_HOME/etc/hadoop/

nano hdfs-site.xml
```

Add the following properties between the `<configuration>` tags:

```
<property>
      <name>dfs.replication</name>
      <value>1</value>
</property>
<property>
      <name>dfs.namenode.name.dir</name>
      <value>file:/home/john/yarnData/namenode</value>
</property>
<property>
      <name>dfs.datanode.data.dir</name>
      <value>file:/home/john/yarnData/datanode</value>
      </property>
```

In the lines above, be careful to change `/home/john/` appropriately into your installation path.

Now format the namenode:

```
      hdfs namenode -format
```

Set the configuration parameters

Change dir to $YARN_HOME/etc/hadoop/:

```
      cd $YARN_HOME/etc/hadoop/
```

Edit the `core-site.xml` file in `$YARN_HOME/etc/hadoop/`

```
      nano core-site.xml
```

and add the following property between the `<configuration>` tags:

```
      <property>
            <name>fs.default.name</name>
            <value>hdfs://localhost:9000</value>
      </property>
```

Then edit the `yarn-site.xml` file and add the following properties:

```
      <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>
```

Copy the template file `mapred-site.xml.template` on file `mapred-site.xml` and add the following property:

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

Set ssh for password-less authentication

Generate an ssh public/private rsa key pair. With all defaults, the following command

```
       ssh-keygen -t rsa
```

will create a public key in `~/.ssh/id_rsa.pub` and a private key in `~/.ssh/id_rsa`

Commands

```
cd ~

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

add the newly generated public key to the authorized keys file, allowing access to your account via ssh without having to type a password.

Check if everything works by typing `ssh localhost`. This command should return something like:

```
$ ssh localhost

The authenticity of host 'localhost (::1)' can't be established.

RSA key fingerprint is 60:b9:c6:8c:29:58:1c:5f:0b:96:cf:82:6f:0c:
94:6d.

Are you sure you want to continue connecting (yes/no)?
```

Type yes. If the connection is refused, it may be the case that the ssh daemon is not running. In that case, you can start it depending on your operating system (e.g., on Mac OS X go to `System preferences/Sharing` and flag `Remote login`; on Ubuntu just type `apt-get install ssh-server`).

Firing it up

You should now be all set and ready to start your single-node pseudo-distributed hadoop cluster:

```
start-dfs.sh

start-yarn.sh
```

Check for any error message and check that command

```
jps
```

returns something similar to:

```
95579 ResourceManager

94607 NameNode

6815 Jps

94801 DataNode

95723 NodeManager

94950 SecondaryNameNode
```

Run some task (e.g. compute a few digits of π):

```
yarn jar $YARN_HOME/share/hadoop/mapreduce/hadoop-mapreduce-
examples-               2.2.0.jar bbp 1 4 2 /pi

hadoop fs -ls /pi
```

The listing should return something like

```
  0 2014-04-02 14:43 /pi/out
106 2014-04-02 14:43 /pi/pi.txt
  2 2014-04-02 14:43 /pi/pi_1_4.hex
```

And you can check the value computed by issuing the following command:

```
hadoop fs -cat /pi/pi.txt
```

Remember to stop yarn and dfs when you are done with using hadoop. Commands

```
stop-yarn.sh
stop-dfs.sh
```

should do the work.

Troubleshooting

In case something goes wrong, check for error messages in the `stderr` (or in the `syslog`) log files:

```
find $YARN_HOME -name stderr |xargs cat | less
```

for hints on what the error is related to.

More configuration parameters

The default values of many configuration parameters for hadoop can be found at the following link:

```
http://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-
              mapreduce-client-core/mapred-default.xml
```

Some parameters you are likely to change are the number of concurrent map and reduce tasks and memory limits for mappers and reducers:

```
<property>
      <name>mapreduce.job.maps</name>
      <value>4</value>
</property>
<property>
      <name>mapreduce.job.reduces</name>
      <value>2</value>
</property>
<property>
      <name>mapred.reduce.child.java.opts</name>
      <value>-Xmx2048m</value>
</property>
<property>
      <name>mapred.map.child.java.opts</name>
```

```
            <value>-Xmx2048m</value>
      </property>
```

Setting the above properties in the `mapred-site.xml` file would instruct hadoop to run 4 mappers and two reducers concurrently and would increase the memory limits for both mappers and reducers to 2Gb.

Acknowledgements

This short guide is partially based on the tutorial available from:

`http://practicalcloudcomputing.com/post/26448910436/install-and-run-hadoop-yarn-in-10-easy-steps`