Running Hadoop On Fedora Linux (Multi-Node Cluster)

Table of Contents:

1. What we are going to do

Setting up a *multi-node* Hadoop cluster using the Hadoop Distributed File System (HDFS) on Fedora Linux.

2. Approach and structure

From two single-node clusters to a multi-node cluster – We will build a multi-node cluster using two Fedora boxes in this tutorial. In my humble opinion, the best way to do this for starters is to install, configure and test a “local” Fedora setup for each of the two Fedora boxes, and in a second step to “merge” these two single-node clusters into one multi-node cluster in which one Fedora box will become the designated master (but also act as a slave with regard to data storage and processing), and the other box will become only a slave. It’s much easier to track down any problems you might encounter due to the reduced complexity of doing a single-node cluster setup first on each machine.

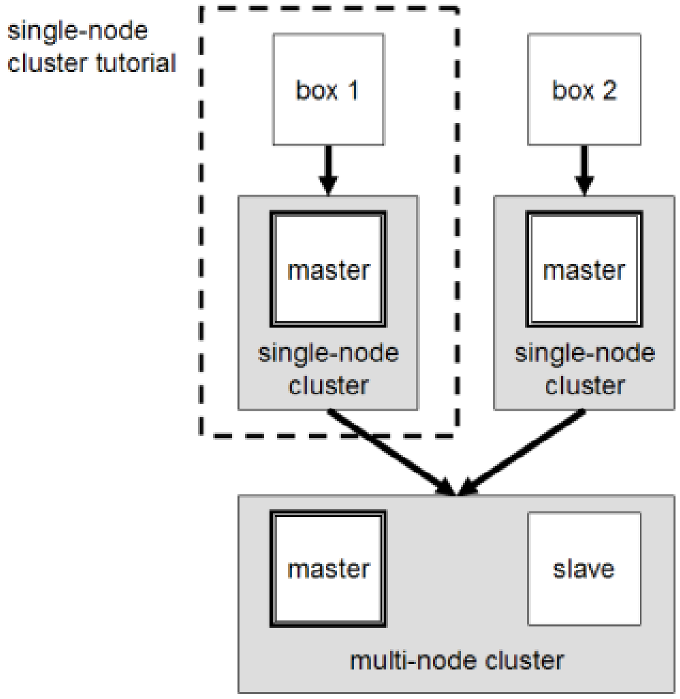

Fig . approach and structure

3. Prerequisites

Configure single-node clusters first.

We will make one single node (dellnode1.pictlibrary) as master and another single node (dellnode2.pictlibrary) as slave.

Shutdown each single-node cluster with */bin/stop-all.sh* before continuing if you haven’t done so already.

.

4. Networking

Add the IP address 10.1.1.1 to the master machine and 10.1.1.2 to the slave machine.
Update /etc/hosts on **both** machines with the following lines:

```
[hduser@dellnode1~]$ su
[root@localhost ~] # vi /etc/hosts
```

```
10.1.1.1    dellnode1.pictlibrary         #master
10.1.1.2    dellnode2.pictlibrary         #slave
```

5. SSH access

The hduser user on the master (aka hduser@master) must be able to connect to its own user account on the master – i.e. ssh master in this context and not necessarily ssh localhost to the hduser user account on the slave (aka hduser@slave) via a password-less SSH login.

```
[root@localhost ~] # su hduser
[hduser@dellnode1~]$ ssh-copy-id -i $HOME/.ssh/id_rsa.pub hduser@dellnode2.pictlibrary
```

Now test connection from master to master

```
[hduser@dellnode1 ~] $ ssh dellnode1.pictlibrary
The authenticity of host ' dellnode1.pictlibrary (10.1.1.1)' can't be
established.
RSA key fingerprint is
3b:21:b3:c0:21:5c:7c:54:2f:1e:2d:96:79:eb:7f:95.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'dellnode1.pictlibrary' (RSA) to the list
of known hosts.
Linux dellnode1.pictlibrary 2.6.20-16-386 #2 Sat Jan 7 20:16:13 UTC
2012 i686
...
[hduser@dellnode1 ~] $
```

The final step is to test the SSH setup by connecting with user hduser from the master to the user account hduseron the slave.

```
[hduser@dellnode1 ~] $ ssh dellnode2.pictlibrary
The authenticity of host ' dellnode2.pictlibrary (10.1.1.2)' can't be
established.
RSA key fingerprint is
5d:24:a3:c0:11:5c:6c:54:5f:1e:23:96:29:3b:6f:25.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'dellnode2.pictlibrary' (RSA) to the list
of known hosts.
Linux dellnode2.pictlibrary 2.6.20-16-386 #2 Sat Jan 7 20:16:13 UTC
2012 i686
...
[hduser@dellnode2 ~] $
```

6. Hadoop

Cluster Overview

The master node will also act as a slave because we only have two machines available in our cluster but still want to spread data storage and processing to multiple machines.

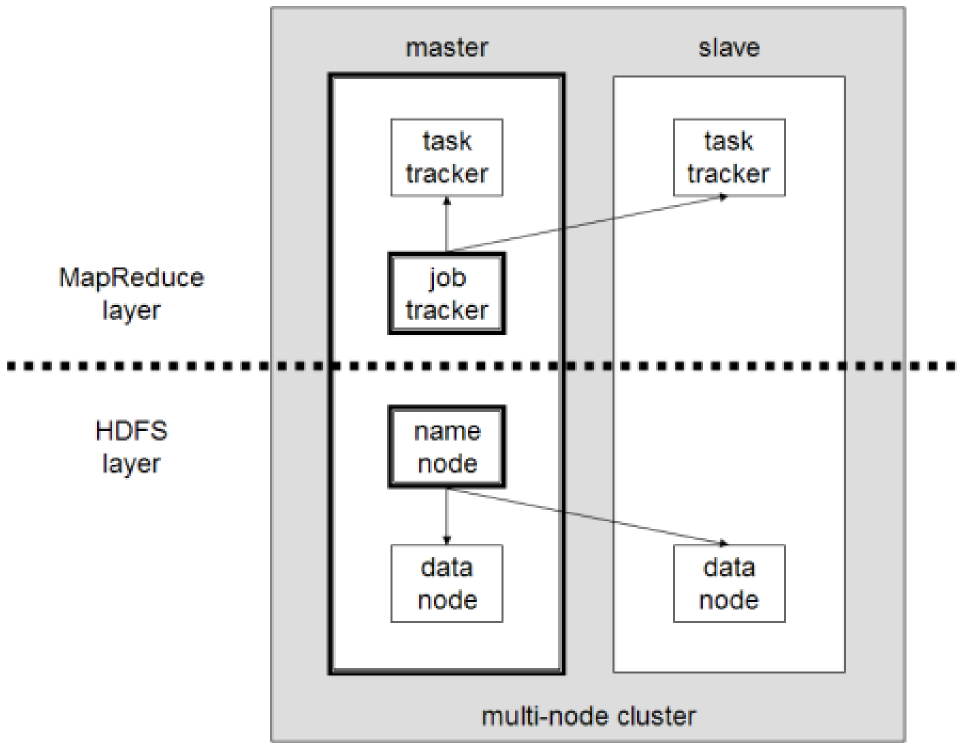

Fig. the final multi-node cluster.

The master node will run the “master” daemons for each layer: NameNode for the HDFS storage layer, and JobTracker for the MapReduce processing layer. Both machines will run the “slave” daemons: DataNode for the HDFS layer, and TaskTracker for MapReduce processing layer. Basically, the “master” daemons are responsible for coordination and management of the “slave” daemons while the latter will do the actual data storage and data processing work.

From the ***Hadoop documentation***:

Typically one machine in the cluster is designated as the NameNode and another machine the as JobTracker, exclusively. These are the actual “master nodes”. The rest of the machines in the cluster act as both DataNode and TaskTracker. These are the slaves or “worker nodes”.

Configuration

1. *conf/masters (master only)*

The conf/masters file defines on which machines Hadoop will start *secondary NameNodes* in our multi-node cluster. In our case, this is just the master machine. The primary NameNode and the JobTracker will always be the machines on which you run the bin/start-fs.sh and bin/start-mapred.sh scripts, respectively.

Note that you can also start an Hadoop daemon manually on a machine via bin/hadoop-daemon.sh start [namenode | secondarynamenode | datanode | jobtracker | tasktracker], which will not take the conf/masters and conf/slaves files into account.

The machine on which bin/start-dfs.sh is run will become the *primary* NameNode.

On master, update conf/masters

```
[hduser@dellnode1 ~] $ cd $HADOOP_HOME
[hduser@dellnode1 hadoop] $ vi conf/masters
```

Replace localhost by name of master node (in this case dellnode1.pictlibrary) to this file as follows:

```
dellnode1.pictlibrary
```

2. *conf/slaves (master only)*

This conf/slaves file lists the hosts, one per line, where the Hadoop slave daemons (DataNodes and TaskTrackers) will be run. We want both the master box and the slave box to act as Hadoop slaves because we want both of them to store and process data.

On master, update conf/slaves

```
 [hduser@dellnode1 hadoop] $ vi conf/slaves
```

Replace localhost by name of slave nodes (also include master name in slave list) to this file as follows:

```
dellnode1.pictlibrary
dellnode2.pictlibrary
#anotherslave03
#anotherslave04
```

Important: You have to change the configuration files conf/core-site.xml, conf/mapred-site.xml and conf/hdfs-site.xml on ALL machines as follows.

3. *conf/core-site.xml (all machines)*

We have to change the fs.default.name variable (in conf/core-site.xml) which specifies the NameNode(the HDFS master) host and port. In our case, this is the master machine.

```
[hduser@localhost hadoop] $ vi conf/core-site.xml
```

```
<configuration>
<property>
  <name>fs.default.name</name>
  <value>hdfs://dellnode1.pictlibrary:54310</value>
  <description>The name of the default file system.  A URI whose
  scheme and authority determine the FileSystem implementation.  The
  uri's scheme determines the config property (fs.SCHEME.impl) naming
  the FileSystem implementation class.  The uri's authority is used
to
  determine the host, port, etc. for a filesystem.</description>
</property>
</configuration>
```

We have to change the **mapred.job.tracker** variable (in conf/mapred-site.xml) which specifies theJobTracker (MapReduce master) host and port. Again, this is the master in our case

```
[hduser@localhost hadoop] $ vi conf/mapred-site.xml
```

```
<configuration>
<property>
  <name>mapred.job.tracker</name>
  <value> dellnode1.pictlibrary:54311</value>
  <description>The host and port that the MapReduce job tracker runs
  at.  If "local", then jobs are run in-process as a single map
  and reduce task.
  </description>
</property>
</configuration>
```

We change the **dfs.replication** variable (in conf/hdfs-site.xml) which specifies the default block replication. It defines how many machines a single file should be replicated to before it becomes available. If you set this to a value higher than the number of slave nodes (more precisely, the number of DataNodes) that you have available, you will start seeing a lot of (Zero targets found, forbidden1.size=1) type errors in the log files.

The default value of dfs.replication is 3. However, we have only two nodes available, so we set dfs.replication to 2.

```
[hduser@localhost hadoop] $ vi conf/hdfs-site.xml
```

```
<configuration>
<property>
  <name>dfs.replication</name>
  <value>2</value>
  <description>Default block replication.
  The actual number of replications can be specified when the file is
created.
  The default is used if replication is not specified in create time.
  </description>
</property>
</configuration>
```

Formatting the HDFS filesystem via the NameNode

Before we start our new multi-node cluster, we have to format Hadoop's distributed filesystem (HDFS) for the NameNode. You need to do this the first time you set up a Hadoop cluster.

Do not format a running Hadoop NameNode, this will cause all your data in the HDFS filesytem to be erased.

```
[hduser@ dellnode1 hadoop] $ hadoop namenode -format

12/02/01 11:42:15 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = dellnode1.pictlibrary/10.1.1.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 0.20.203.0
STARTUP_MSG:   build =
http://svn.apache.org/repos/asf/hadoop/common/branches/branch-0.20-
security-203 -r 1099333; compiled by 'oom' on Wed May  4 07:57:50 PDT
2011
************************************************************/
Re-format filesystem in /app/hadoop/tmp/dfs/name ? (Y or N) Y
12/02/01 11:42:16 INFO util.GSet: VM type       = 32-bit
12/02/01 11:42:16 INFO util.GSet: 2% max memory = 17.77875 MB
12/02/01 11:42:16 INFO util.GSet: capacity      = 2^22 = 4194304
entries
12/02/01 11:42:16 INFO util.GSet: recommended=4194304, actual=4194304
12/02/01 11:42:16 INFO namenode.FSNamesystem: fsOwner=hduser
12/02/01 11:42:16 INFO namenode.FSNamesystem: supergroup=supergroup
12/02/01 11:42:16 INFO namenode.FSNamesystem:
isPermissionEnabled=true
12/02/01 11:42:17 INFO namenode.FSNamesystem:
dfs.block.invalidate.limit=100
12/02/01 11:42:17 INFO namenode.FSNamesystem:
isAccessTokenEnabled=false accessKeyUpdateInterval=0 min(s),
accessTokenLifetime=0 min(s)
12/02/01 11:42:17 INFO namenode.NameNode: Caching file names occuring
more than 10 times
12/02/01 11:42:17 INFO common.Storage: Image file of size 112 saved
in 0 seconds.
12/02/01 11:42:17 INFO common.Storage: Storage directory
/app/hadoop/tmp/dfs/name has been successfully formatted.
12/02/01 11:42:17 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at
dellnode1.pictlibrary/10.1.1.1
************************************************************/
[hduser@ dellnode1 hadoop] $
```

Starting the multi-node cluster

Starting the cluster is done in two steps. First, the HDFS daemons are started: the NameNode daemon is started on `master`, and DataNode daemons are started on all slaves (here: `master` and `slave`). Second, the MapReduce daemons are started: the JobTracker is started on `master`, and TaskTracker daemons are started on all slaves (here: `master` and `slave`).

1. **HDFS daemons**

Run the command /bin/start-dfs.sh on the machine you want the (primary) NameNode to run on. This will bring up HDFS with the NameNode running on the machine you ran the previous command on, and DataNodes on the machines listed in the conf/slaves file.

we will run bin/start-dfs.sh on master:

```
[hduser@dellnode1 hadoop] $ start-dfs.sh
starting namenode, logging to /usr/local/hadoop/bin/../logs/hadoop-
hduser-namenode-master.out
slave: Ubuntu 10.04
slave: starting datanode, logging to
/usr/local/hadoop/bin/../logs/hadoop-hduser-datanode-slave.out
master: starting datanode, logging to
/usr/local/hadoop/bin/../logs/hadoop-hduser-datanode-master.out
master: starting secondarynamenode, logging to
/usr/local/hadoop/bin/../logs/hadoop-hduser-secondarynamenode-
master.out
[hduser@ dellnode1 hadoop] $
```

At this point, the following Java processes should run on `master`…

```
[hduser@dellnode1 hadoop] $ jps
14799 NameNode
15314 Jps
14880 DataNode
14977 SecondaryNameNode
[hduser@dellnode1 hadoop] $
```

(the process IDs don't matter of course)

and the following on `slave`.

```
[hduser@dellnode2 hadoop] $ jps
15183 DataNode
15616 Jps
[hduser@dellnode2 hadoop] $
```

2. MapReduce daemons

Run the command /bin/start-mapred.sh on the machine you want the JobTracker to run on. This will bring up the MapReduce cluster with the JobTracker running on the machine you ran the previous command on, and TaskTrackers on the machines listed in the conf/slaves file.

We will run bin/start-mapred.sh on master:

```
[hduser@dellnode1 hadoop] $ start-mapred.sh
starting jobtracker, logging to /usr/local/hadoop/bin/../logs/hadoop-
hadoop-jobtracker-master.out
slave: dellnode2.pictlibrary
slave: starting tasktracker, logging to
/usr/local/hadoop/bin/../logs/hadoop-hduser-tasktracker-slave.out
master: starting tasktracker, logging to
/usr/local/hadoop/bin/../logs/hadoop-hduser-tasktracker-master.out
[hduser@ dellnode1 hadoop] $
```

The following Java processes should run on `master`…

```
[hduser@dellnode1 hadoop] $ jps
16017 Jps
14799 NameNode
15686 TaskTracker
14880 DataNode
15596 JobTracker
14977 SecondaryNameNode
[hduser@dellnode1 hadoop] $
```

and the following on `slave`.

```
[hduser@dellnode2 hadoop] $ jps
15183 DataNode
15897 TaskTracker
15616 Jps
[hduser@dellnode2 hadoop] $
```

Stopping the multi-node cluster

Like starting the cluster, stopping it is done in two steps. The workflow is the opposite of starting, however. First, we begin with stopping the MapReduce daemons: the JobTracker is stopped on `master`, and TaskTracker daemons are stopped on all slaves (here: `master` and `slave`). Second, the HDFS daemons are stopped: the NameNode daemon is stopped on `master`, and DataNode daemons are stopped on all slaves (here: `master` and `slave`).

1. MapReduce daemons

Run the command /bin/stop-mapred.sh on the JobTracker machine. This will shut down the MapReduce cluster by stopping the JobTracker daemon running on the machine you ran the previous command on, and TaskTrackers on the machines listed in the conf/slaves file.

We will run bin/stop-mapred.sh on master:

```
[hduser@dellnode1 hadoop] $ stop-mapred.sh
stopping jobtracker
slave: dellnode2.pictlibrary
master: stopping tasktracker
slave: stopping tasktracker
[hduser@ dellnode1 hadoop] $
```

(Note: The output above might suggest that the JobTracker was running and stopped on slave, but you can be assured that the JobTracker ran on master.)

At this point, the following Java processes should run on master…

```
[hduser@dellnode1 hadoop] $ jps
14799 NameNode
18386 Jps
14880 DataNode
14977 SecondaryNameNode
[hduser@ dellnode1 hadoop] $
```

…and the following on `slave`.

```
[hduser@dellnode2 hadoop] $ jps
15183 DataNode
18636 Jps
[hduser@ dellnode2 hadoop] $
```

2. HDFS daemons

Run the command /bin/stop-dfs.sh on the NameNode machine. This will shut down HDFS by stopping the NameNode daemon running on the machine you ran the previous command on, and DataNodes on the machines listed in the conf/slaves file.

In our case, we will run bin/stop-dfs.sh on master:

```
[hduser@dellnode1 hadoop] $ stop-dfs.sh
stopping namenode
slave: dellnode2.pictlibrary
slave: stopping datanode
master: stopping datanode
master: stopping secondarynamenode
[hduser@ dellnode1 hadoop] $
```

(again, the output above might suggest that the NameNode was running and stopped on `slave`, but you can be assured that the NameNode ran on `master`)
At this point, the only following Java processes should run on `master`…

```
[hduser@dellnode1 hadoop] $ jps
18670 Jps
[hduser@ dellnode1 hadoop] $
```

…and the following on `slave`.

```
[hduser@dellnode2 hadoop] $ jps
18894 Jps
[hduser@ dellnode2 hadoop] $
```