Laboratory

3

# Hadoop Installation: Pseudo-distributed mode

## Obecjective

Hadoop can be run in 3 different modes. Different modes of Hadoop are

### 1. Standalone Mode

- Default mode of Hadoop
- HDFS is not utilized in this mode.
- Local file system is used for input and output
- Used for debugging purpose
- No Custom Configuration is required in 3 hadoop(mapred-site.xml,core-site.xml, hdfs-site.xml) files.
- Standalone mode is much faster than Pseudo-distributed mode.

### 2. Pseudo Distributed Mode(Single Node Cluster)

- Configuration is required in given 3 files for this mode
- Replication factory is one for HDFS.
- Here one node will be used as Master Node / Data Node / Job Tracker / Task Tracker
- Used for Real Code to test in HDFS.
- Pseudo distributed cluster is a cluster where all daemons are
- running on one node itself.

### 3. Fully distributed mode (or multiple node cluster)

- This is a Production Phase
- Data are used and distributed across many nodes.
- Different Nodes will be used as Master Node / Data Node / Job Tracker / Task Track

In objective of this experiment is to install Hadoop in Pseudo Distributed Mode.

## Overview

Hadoop is an open source, Java-based programming framework that supports the processing and storage of extremely large data sets in a distributed computing environment. It is part of the Apache project sponsored by the Apache Software Foundation.

Hadoop was created by computer scientists **Doug Cutting and Mike Cafarella in 2006** to support distribution for the Nutch search engine. It was inspired by Google's MapReduce, a software framework in which an application is broken down into

numerous small parts. Any of these parts, which are also called fragments or blocks, can be run on any node in the cluster. After years of development within the open source community, Hadoop 1.0 became publically available in November 2012 as part of the Apache project sponsored by the Apache Software Foundation.

## Procedure

The steps for install Hadoop in Pseudo Distributed Mode(Single Node Cluster) are

## Step1: Installing Java

Hadoop framework is written in Java!!

```
teacher@teacher-virtual-machine:~$ cd ~
# Update the source list

teacher@teacher-virtual-machine:~$ sudo apt-get update
# The OpenJDK project is the default version of Java
# that is provided from a supported Ubuntu repository
.
teacher@teacher-virtual-machine:~$ sudo apt-get install default-
jdk
teacher@teacher-virtual-machine:~$ java -version
java version "1.7.0_65"
OpenJDK Runtime Environment (IcedTea 2.5.3) (7u71-2.5.3-
0ubuntu0.14.04.1)
   OpenJDK 64-Bit Server VM (build 24.65-b04, mixed mode)
```

## Step2: Adding a dedicated Hadoop user

**teacher@teacher-virtual-machine:~$ sudo addgroup hadoop**

**Adding group `hadoop' (GID 1002) ...**
**Done.**
**teacher@teacher-virtual-machine:~$ sudo adduser --ingroup hadoop hduser**
**Adding user `hduser' ...**
**Adding new user `hduser' (1001) with group `hadoop' ...**
**Creating home directory `/home/hduser' ...**
**Copying files from `/etc/skel' ...**
**Enter new UNIX password:**
**Retype new UNIX password:**
**passwd: password updated successfully**
**Changing the user information for hduser**
**Enter the new value, or press ENTER for the default**
**Full Name []:**
**Room Number []:**
**Work Phone []:**
**Home Phone []:**

**Other []:**
**Is the information correct? [Y/n] Y**

## Step3: Installing SSH

**ssh** has two main components:
**1.ssh** : The command we use to connect to remote machines - the client.
2.**sshd** : The daemon that is running on the server and allows clients to connect to the server.
The **ssh** is pre-enabled on Linux, but in order to start **sshd** daemon, we need to install **ssh** first. Use this
command to do that :
**teacher@teacher-virtual-machine:~$ sudo apt-get install ssh**
This will install ssh on our machine. If we get something similar to the following, we can think it is
setup properly:
**teacher@teacher-virtual-machine:~$ which ssh**
**/usr/bin/ssh**
**teacher@teacher-virtual-machine:~$ which sshd**
**/usr/sbin/sshd**

## Step4: **Create and Setup SSH Certificates**

Hadoop requires SSH access to manage its nodes, i.e. remote machines plus our local machine. For our single-node setup of Hadoop, we therefore need to configure SSH access to localhost.So, we need to have SSH up and running on our machine and configured it to allow SSH public key authentication.
Hadoop uses SSH (to access its nodes) which would normally require the user to enter a password. However, this requirement can be eliminated by creating and setting up SSH certificates using the following commands. If asked for a filename just leave it blank and press the enter key to continue.

**teacher@teacher-virtual-machine:~$ : su hduser**
**Password:**

**teacher@teacher-virtual-machine:~$ : ssh-keygen -t rsa -P ""**
**Generating public/private rsa key pair.**
**Enter file in which to save the key (/home/hduser/.ssh/id_rsa):**
**Created directory '/home/hduser/.ssh'.**
**Your identification has been saved in /home/hduser/.ssh/id_rsa.**
**Your public key has been saved in /home/hduser/.ssh/id_rsa.pub.**
**The key fingerprint is:**
**50:6b:f3:fc:0f:32:bf:30:79:c2:41:71:26:cc:7d:e3 hduser@teacher**
**The key's randomart image is:**
**+--[ RSA 2048]----+**
**| .oo.o |**
**| . .o=. o |**
**| . + . o . |**
**| o = E |**
**| S + |**

```
| . + |
| O + |
| O o |
| o.. |
+-----------------+
```

**hduser@teacher:/home/k$**

**cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys**

The second command adds the newly created key to the list of authorized keys so that Hadoop can use ssh without prompting for a password.

We can check if ssh works:

**hduser@teacher:/home/k$ ssh localhost**

**The authenticity of host 'localhost (127.0.0.1)' can't be established.**
**ECDSA key fingerprint is e1:8b:a0:a5:75:ef:f4:b4:5e:a9:ed:be:64:be:5c:2f.**
**Are you sure you want to continue connecting (yes/no)? yes**
**Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.**
**Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-40-generic x86_64)**

# Step5: **Install Hadoop**

hduser@teacher**:~$wget** http://mirrors.sonic.net/apache/hadoop/common/hadoop-2.6.0/ **hadoop-2.6.0.tar.gz**

**hduser@teacher:~$ tar xvzf hadoop-2.6.0.tar.gz**

We want to move the Hadoop installation to the **/usr/local/hadoop** directory

using the following command:

```
hduser@teacher:~/hadoop-2.6.0$ sudo mv * /usr/local/hadoop
[sudo] password for hduser:
hduser is not in the sudoers file. This incident will be reported.
```

Oops!... We got:

```
"hduser is not in the sudoers file. This incident will be
reported."
```

This error can be resolved by logging in as a root user, and then add **hduser** to **sudo**:

```
hduser@teacher:~/hadoop-2.6.0$ su k
Password:

hduser@teacher:/home/hduser$ sudo adduser hduser sudo
[sudo] password for k:
Adding user `hduser' to group `sudo' ...
Adding user hduser to group sudo
Done.
```

Now, the **hduser** has root priviledge, we can move the Hadoop installation to the **/usr/local/hadoop**directory without any problem:

```
hduser@teacher:/home/hduser$ sudo su hduser
hduser@teacher:~/hadoop-2.6.0$ sudo mv * /usr/local/hadoop
hduser@teacher:~/hadoop-2.6.0$ sudo chown -R hduser:hadoop
/usr/local/hadoop
```

# Step6: **Setup Configuration Files**

The following files will have to be modified to complete the Hadoop setup:
1.~/.bashrc
2./usr/local/hadoop/etc/hadoop/hadoop-env.sh
3./usr/local/hadoop/etc/hadoop/core-site.xml
4./usr/local/hadoop/etc/hadoop/mapred-site.xml.template
5./usr/local/hadoop/etc/hadoop/hdfs-site.xml

## 1. ~/.bashrc

Before editing the **.bashrc** file in our home directory, we need to find the path where Java has been installed to set the **JAVA_HOME** environment variable using the following command:
**hduser@teacher: update-alternatives --config java**
**There is only one alternative in link group java (providing /usr/bin/java): /usr/lib/jvm/java-7-openjdk-amd64/jre/bin/java**
**Nothing to configure.**
Now we can append the following to the end of **~/.bashrc**:

**hduser@teacher:~$ vi ~/.bashrc**

**#HADOOP VARIABLES START**
**export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64**
**export HADOOP_INSTALL=/usr/local/hadoop**
**export PATH=$PATH:$HADOOP_INSTALL/bin**
**export PATH=$PATH:$HADOOP_INSTALL/sbin**
**export HADOOP_MAPRED_HOME=$HADOOP_INSTALL**
**export HADOOP_COMMON_HOME=$HADOOP_INSTALL**
**export HADOOP_HDFS_HOME=$HADOOP_INSTALL**
**export YARN_HOME=$HADOOP_INSTALL**
**export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native**
**export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"**
**#HADOOP VARIABLES END**

**hduser@teacher:~$ source ~/.bashrc**
note that the JAVA_HOME should be set as the path just before the '.../bin/':

**hduser@ubuntu-VirtualBox:~$ javac -version**
**javac 1.7.0_75**

**hduser@ubuntu-VirtualBox:~$ which javac**
**/usr/bin/javac**

**hduser@ubuntu-VirtualBox:~$ readlink -f /usr/bin/javac/usr/lib/jvm/java-7-openjdk-amd64/bin/javac**

## 2. /usr/local/hadoop/etc/hadoop/hadoop-env.sh

We need to set **JAVA_HOME** by modifying **hadoop-env.sh** file.
**hduser@teacher:~$ vi /usr/local/hadoop/etc/hadoop/hadoop-env.sh**
**export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64**
Adding the above statement in the **hadoop-env.sh** file ensures that the value of JAVA_HOME variablewill be available to Hadoop whenever it is started up.

## 3. /usr/local/hadoop/etc/hadoop/core-site.xml:

The **/usr/local/hadoop/etc/hadoop/core-site.xml** file contains configuration properties that Hadoop uses when starting up.

This file can be used to override the default settings that Hadoop starts with.
**hduser@teacher:~$ sudo mkdir -p /app/hadoop/tmp**
**hduser@teacher:~$ sudo chown hduser:hadoop /app/hadoop/tmp**
Open the file and enter the following in between the <configuration></configuration> tag:
**hduser@teacher:~$ vi /usr/local/hadoop/etc/hadoop/core-site.xml**

```
<configuration>
<property>
<name>hadoop.tmp.dir</name>
<value>/app/hadoop/tmp</value>
<description>A base for other temporary directories.</description>
</property>
<property>
<name>fs.default.name</name>
<value>hdfs://localhost: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>
```

## 4 /usr/local/hadoop/etc/hadoop/mapred-site.xml

By default, the **/usr/local/hadoop/etc/hadoop/** folder contains
**/usr/local/hadoop/etc/hadoop/mapred-site.xml.template**
file which has to be renamed/copied with the name **mapred-site.xml**:

**hduser@teacher:~$ cp /usr/local/hadoop/etc/hadoop/mapred-site.xml.template /usr/local /hadoop/etc/ hadoop /mapred-site.xml**

The **mapred-site.xml** file is used to specify which framework is being used for MapReduce.
We need to enter the following content in between the <configuration></configuration> tag:

```
<configuration>
<property>
<name>mapred.job.tracker</name>
<value>localhost: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>**

## 5 /usr/local/hadoop/etc/hadoop/hdfs-site.xml

The **/usr/local/hadoop/etc/hadoop/hdfs-site.xml** file needs to be configured for each host in the cluster that is being used.It is used to specify the directories which will be used as the **namenode** and the **datanode** on that host. Before editing this file, we need to create two directories which will contain the namenode and the datanode for this Hadoop installation.

This can be done using the following commands:
**hduser@teacher:~$ sudo mkdir -p /usr/local/hadoop_store/hdfs/namenode**
**hduser@teacher:~$ sudo mkdir -p /usr/local/hadoop_store/hdfs/datanode**
**hduser@teacher:~$ sudo chown -R hduser:hadoop /usr/local/hadoop_store**

Open the file and enter the following content in between the <configuration></configuration> tag:
**hduser@teacher:~$ vi /usr/local/hadoop/etc/hadoop/hdfs-site.xml**
**<configuration>**
**<property>**
**<name>dfs.replication</name>**
**<value>1</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>**
**<property>**
**<name>dfs.namenode.name.dir</name>**
**<value>file:/usr/local/hadoop_store/hdfs/namenode</value>**
**</property>**
**<property>**
**<name>dfs.datanode.data.dir</name>**
**<value>file:/usr/local/hadoop_store/hdfs/datanode</value>**
**</property>**
**</configuration>**

## Step7: **Format the New Hadoop Filesystem**

Now, the Hadoop file system needs to be formatted so that we can start to use it. The format command should be issued with write permission since it creates **current** directory under **/usr/local/hadoop_store/hdfs/namenode** folder:

**hduser@teacher:~$ hadoop namenode -format**
**DEPRECATED: Use of this script to execute hdfs command is deprecated.**
**Instead use the hdfs command for it.**

**15/04/18 14:43:03 INFO namenode.NameNode: STARTUP_MSG:**
**---------------------------------**

**---------------------------------**

**15/04/18 14:43:12 INFO util.ExitUtil: Exiting with status 0**
**15/04/18 14:43:12 INFO namenode.NameNode: SHUTDOWN_MSG:**
**/**************************************************************
**SHUTDOWN_MSG: Shutting down NameNode at laptop/192.168.1.1**
**************************************************************/**

Note that **hadoop namenode -format** command should be executed once before we start using Hadoop.If this command is executed again after Hadoop has been used, it'll destroy all the data on the Hadoop file system

# Step8: Starting Hadoop

Now it's time to start the newly installed single node cluster.
We can use **start-all.sh** or (**start-dfs.sh** and **start-yarn.sh**)

**hduser@teacher:~$ cd /usr/local/hadoop/sbin**
**hduser@teacher:/usr/local/hadoop/sbin$ ls**

| | | |
|---|---|---|
| **distribute-exclude.sh** | **start-all.cmd** | **stop-balancer.sh** |
| **hadoop-daemon.sh** | **start-all.sh** | **stop-dfs.cmd** |
| **hadoop-daemons.sh** | **start-balancer.sh** | **stop-dfs.sh** |
| **hdfs-config.cmd** | **start-dfs.cmd** | **stop-secure-dns.sh** |
| **hdfs-config.sh** | **start-dfs.sh** | **stop-yarn.cmd** |
| **httpfs.sh** | **start-secure-dns.sh** | **stop-yarn.sh** |
| **kms.sh** | **start-yarn.cmd** | **yarn-daemon.sh** |
| **mr-jobhistory-daemon.sh** | **start-yarn.sh** | **yarn-daemons.sh** |
| **refresh-namenodes.sh** | **stop-all.cmd** | **slaves.sh stop-all.sh** |

**hduser@teacher:/usr/local/hadoop/sbin$ sudo su hduser**

**hduser@teacher:/usr/local/hadoop/sbin$ start-all.sh**

This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
15/04/18 16:43:13 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Starting namenodes on [localhost]
localhost: starting namenode, logging to /usr/local/hadoop/logs/hadoop-hduser-namenodelaptop. out
localhost: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hduser-datanodelaptop.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-hdusersecondarynamenode-laptop.out
15/04/18 16:43:58 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hduser-resourcemanagerlaptop. out
localhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hduser-nodemanagerlaptop.out

We can check if it's really up and running:

**hduser@teacher:/usr/local/hadoop/sbin$ jps**

**9026 NodeManager**
**7348 NameNode**
**9766 Jps**
**8887 ResourceManager**
**7507 DataNode**

The output means that we now have a functional instance of Hadoop running on our VPS (Virtual private server).

## Step9: **Stopping Hadoop**

**$ pwd**

**/usr/local/hadoop/sbin**

**$ ./stop-all.sh**
**It will stop all Hadoop demons**

# Hadoop Web Interfaces

Let's start the Hadoop again and see its Web UI:

```
hduser@teacher:/usr/local/hadoop/sbin$ start-all.sh
```

**http://localhost:50070/ - web UI of the NameNode daemon**

## Questions

1. What is Hadoop?
2. Define HDFS?
3. Define Mapreduce?
4. What is Name node?
5. What is Data node?
6. What is the default block size in HDFS?
7. What does it means:" Streaming accessing pattern"
8. Mention the configuration files used in Hadoop Installation?
9. What is the port number for web interface url for Hadoop?
10. Who is the developer of Hadoop?
11. What are the other alternatives for Hadoop?

## Conclusion

After successful completion of the program the Students are able to understand various modes of Hadoop installation .They are able to install Hadoop in Pseudo distributed mode.


Prepared By

Name D. Srinivas

Branch Department of Computer Science and Engineering

College MVR College of Engineering and Technology