BDP: Homework 1: Installing Hadoop and Running WordCount

Due on Wednesday, October 11th, 2017

Joonho Kwon

Hani Ramadhan - 201793254

Contents

1 Preliminary Informations

The purpose of this homework report is to explain about the installation steps needed to run **Hadoop Multi-Node Cluster** on virtual environment. The scheme of the NameNode and DataNodes that will be configured is 1 NameNode and 3 DataNodes. Additional info to add more than 3 DataNodes will be provided also. Below are the information about the environment, which might be considered when installing in other environment:

- Virtual Environment: VirtualBox version 5.0.40 Ubuntu r115130
- Host OS: Elementary OS Loki (Ubuntu based-Distribution)
  - RAM: 4 GB
  - Processor Intel(R) Core (TM) i3-4005U CPU @ 1.70 GHz
- Guest OS: Debian GNU/Linux 9 (without GUI)
  - Assigned Virtual RAM on each guest: 512 MB
  - Assigned Virtual Harddisk on each guest: 8 GB
- Network Interface Used: Bridged Network [2]

This report refers to several sources [5] [4] as guide to install using the described environments, specifically using Debian, VirtualBox, and Apache Hadoop 2.8.x.

2 Setting Up (Pre-Installation)

This section provides information that needed as prerequisites before installing Hadoop. Running Hadoop requires Java Virtual machine to be run and SSH to communicate securely between the nodes.

2.1 Installing Java

The Java version that is going to be installed is Java SE Development Kit 8u144 for linux 64 bits, although is not been yet tested in recent documentation of Hadoop Java version. This package can be acquired from http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html. After downloaded, it could be installed by extracting it to any desired directory. In this case, a folder will be created in `/usr` by using `root` permission. The BASH instructions are shown in Instruction 1.

Instruction 1: Extracting Java Binary

```
tar zxvf jdk-8u144-linux-x64.tar.gz
su
mkdir /usr/java
mv <name of extracted java installation directory> /usr/java
exit
```

The command `su` and `exit` define entering and exiting the `root` user respectively.

2.2 Installing SSH

`SSH` is important to communicate between nodes (especially NameNode to DataNode) without login as another user. In addition, `rsync` is also needed to synchronize the files between nodes. The steps are depicted in Instruction 2.

Instruction 2: Installing SSH and rsync

```
su
apt-get install ssh
apt-get install rsync
exit
```

2.3 Creating Hadoop user account and group

This step is optional, but it works as separation of user permission. This user will be defined as *hduser*. Before the user *hduser* added, a group specifying the users of Hadoop should be created first. Thus, using `root` permission as before, the steps are listed in Instruction 3

Instruction 3: Create Hadoop Group and User Account

```
su
addgroup hadoop
adduser --ingroup hadoop hduser
exit
```

2.4 Configuring SSH on hduser

As explained before, SSH is used to communicate between NameNode and DataNode and using the hduser as user on each machines. But, before fartherly stepping into communicate between each machines, the key of the user in the base machine should be created and added in authorized keys. Make sure that the active user is *hduser* before executing SSH creation and addition. The steps to create and adding the SSH key are described in Instruction 4. When executing the first line, just press ¡Enter¿ key without any text. The result should be similar to Figure 1.

Instruction 4: SSH key creation and addition (hduser)

```
ssh-keygen -t rsa -P ""
cat /home/hduser/.ssh/id_rsa.pub >> /home/hduser/authorized_keys
```

Then, the success of the key creation and addition should be verified using step in Instruction 5. Terminal displaying the result should request ”Yes” or ”No”. Answering ”Yes” drives to not-requesting password when logging into another already authorized *hduser* user account by SSH connection in further request. Instruction 6 provides how to exit the established SSH connection.

Instruction 5: Establishing SSH connection to localhost

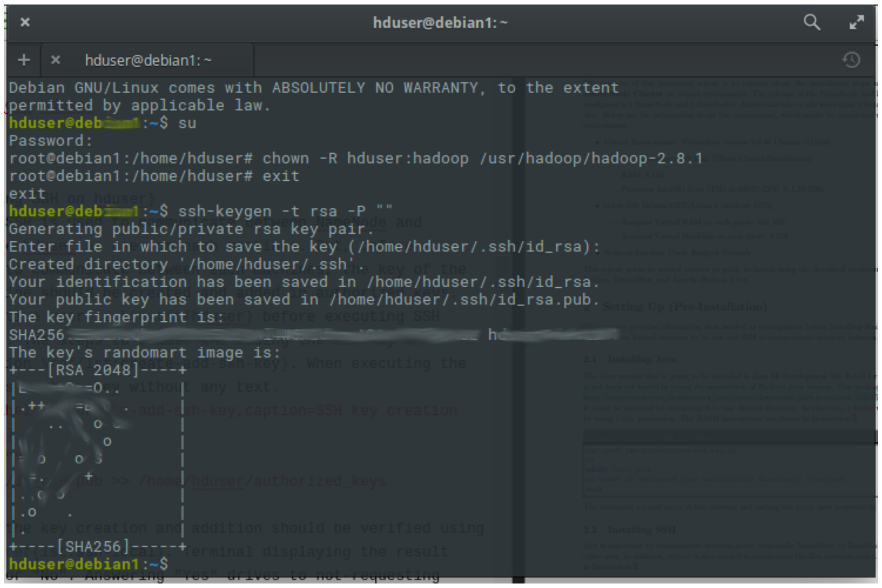
```
ssh localhost
```

Instruction 6: Exiting SSH connection

```
exit
```

3 Setting Up Hadoop 2.8.1

Having all the preparation set up for Hadoop, then we can set up Hadoop for a single machine first as master or NameNode. Later, the slaves or DataNodes will be added using the clone method. Do not clone it yet, because it is easier to set up multi node cluster using steps defined in section 3.5 and 3.7.



```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
hduser@deb[redacted]:~$ su
Password:
root@debian1:/home/hduser# chown -R hduser:hadoop /usr/hadoop/hadoop-2.8.1
root@debian1:/home/hduser# exit
exit
hduser@deb[redacted]:~$ 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:
SHA256 [redacted]
The key's randomart image is:
+---[RSA 2048]----+
[redacted]
+----[SHA256]-----+
hduser@debian1:~$
```

Figure 1: Generating the ssh key result

3.1 About Hadoop 2.8.1

Hadoop is the current version of 2.8.x release line. However, it is still an unstable release. Its stable upgrade (the 2.8.2 version) should be soon released (the planned release date is in August 2017, though). Nevertheless, the 2.8.1 version contains security updates of 2.8.0 version [3].

3.2 Getting Hadoop 2.8.1

Hadoop 2.8.1 can be downloaded from http://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-2.8.1/hadoop-2.8.1.tar.gz and be extracted in desired location. In this case, directory `/usr/hadoop` will be created and used. As usual, the directory will be created using `root` permission. Then, the user *hduser*, which was created before, should be able to modify the directory `/usr/hadoop`. The steps are presented in Instruction 7.

Instruction 7: Installing Hadoop 2.8.1

```
tar zxvf hadoop-2.8.1.tar.gz
su
mkdir /usr/hadoop
mv <name of extracted hadoop-2.8.1 installation directory> /usr/hadoop
chown -R hduser:hadoop /usr/hadoop/hadoop-2.8.1
exit
```

3.3 Master and Slave Scheme and Setup

The installation of Hadoop is done, but it is not ready yet to do some works with it. Meanwhile, let's take a look of general multi-node cluster Hadoop scheme in Figure 3.

The NameNode acts as a *master* of the cluster, which means it is the one who receives the job and assign them to the *workers* or *slaves* named *DataNode*(s). The NameNode's role is not only as the commander of the jobs of each DataNodes, but it communicates about resources management that will be managed by YARN regarding each nodes, whereas called ResourceManager (RM) in the NameNode and NodeManager

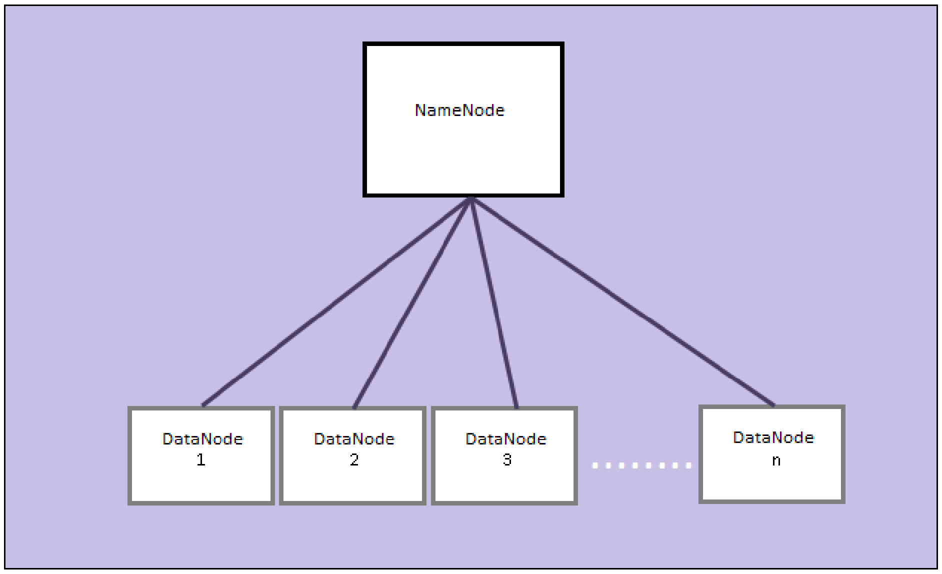


Figure 2: The general scheme of multi node cluster of Hadoop

in the DataNodes.
NameNode can act or *log-in* as *hduser* in each DataNode to run each DataNode's task, given its permission of Hadoop directory. Thus, that's why SSH is important to be installed in each of Hadoop instances, not only the NameNode, but also the DataNodes. This allows NameNode to log in without entering any Hadoop user password in each DataNode.
In this tutorial, the configuration of Hadoop is set out as 1 NameNode and 3 DataNodes. Thus, the scheme of the NameNode and DataNodes will become as illustrated in Figure **??**. Its configuration steps are listed as below:

- Configuring Basic/NameNode Hadoop instance This part focuses in configuring basic instance of Hadoop as well as NameNode instance. It will be cloned later as the first DataNode followed by configuration of several parameters.

- Cloning the basic instance as First DataNode This part focuses in cloning the basic or NameNode instance of Hadoop, which then configured as First DataNode.

- Configuring First DataNode This part focuses in configuring first DataNode's hostname, Hadoop Core, and HDFS. This DataNode then will be used as base clone of other DataNode.

- How to add new DataNode(s) This part focuses in configuring new DataNode's hostname.

- Revisiting the NameNode This part focuses in updates the configuration in routing table, SSH keys, and slaves/DataNode configuration

- Formatting the HDFS

- Running Hadoop

- Running YARN

??.

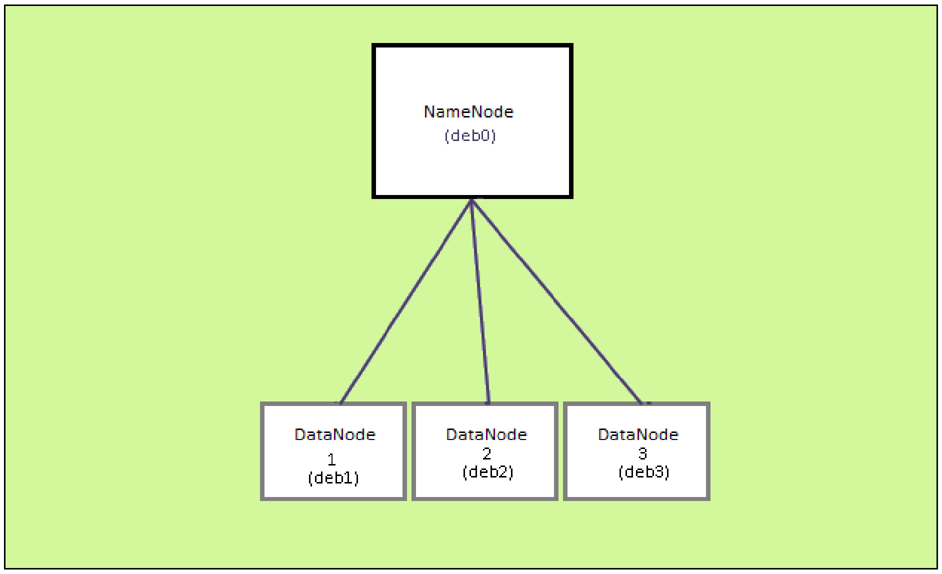


Figure 3: The scheme of multi node cluster of Hadoop of this tutorial

3.4 Configuring Basic/NameNode Hadoop Instance

This section will point out what to be configured in Basic and NameNode Hadoop instance, which consisted of:

- Environment variables
- Routing table
- Hadoop core configuration
- Hadoop HDFS configuration
- Hadoop YARN configuration

3.4.1 Environment variables

The environment variables store the information within the working operating system. This is important in Hadoop since it uses a `$JAVA_HOME` variable to do all of its work. The `$JAVA_HOME` variable is located in `/usr/hadoop/hadoop-2.8.1/etc/hadoop/hadoop-env.sh`, which must be edited. Just try to find out Instruction 8 and replace it with Instruction 9

Instruction 8: hadoop-env.sh content (before)

```
...
export JAVA\_HOME=\{\$JAVA\_HOME\}
...
```

Instruction 9: hadoop-env.sh content (after)

```
...
export JAVA\_HOME=/usr/java/jdk1.8.0_144/
...
```

Optional: Setting .bashrc

The other environment variables that are going to be set up, even though it is optional, are located in `/home/hduser/.bashrc` file in *hduser*'s home directory. The lines depicted in Instruction 10 could be added after the last line. After adding the lines, restart the terminal.

Instruction 10: .bashrc addition lines

```
export JAVA\_HOME=/usr/java/jdk1.8.0\_144/
export HADOOP\_HOME=/usr/hadoop/hadoop-2.8.1/
export PATH=\$PATH:\$HADOOP\_HOME/bin
export PATH=\$PATH:\$HADOOP\_HOME/sbin
```

3.4.2 Routing table

The routing table works as node's navigation route to communicate with other nodes. Since the implemented scheme is the multi cluster node scheme, the node's own address entries will be ignored. Hadoop still do not support the IPV6 setting, which also contained in routing table configuration. Thus, there are several things that should be configured (commented out). Also, the IP Address and the hostname of the machine itself should be added into the table. The routing table can be configured in `/etc/hosts` as `root`. The before and after contents of `/etc/hosts` is depicted as Instruction 11 and Instruction 12 respectively.

Instruction 11: /etc/hosts content (before)

```
127.0.0.1    localhost
127.0.1.1    <hostname>.<something>.<something>.<something>    <hostname>
#The following lines are desirable for IPv6 capable hosts
::1    localhost ip6-localhost ip6-loopback
ff02::1    ip6-allnodes
ff02::2    ip6-allrouters
```

Instruction 12: /etc/hosts content (after)

```
127.0.0.1    localhost
#127.0.1.1    <hostname>.<something>.<something>.<something>    <hostname>
#The following lines are desirable for IPv6 capable hosts
#::1    localhost ip6-localhost ip6-loopback
#ff02::1    ip6-allnodes
#ff02::2    ip6-allrouters
```

Optional: Setting hostname and revisiting /etc/hosts

Essentially, this part of tutorial would not affect the work of Hadoop configuration too much. However, this part is highly useful to make the configuration steps efficient. The hostname of the nodes will be set referring to IP address. The following steps are configurations for the NameNode. These steps will be used again in later DataNode configuration.

1. Check the node's IP Address
   Command `ip a` or `ifconfig` will reveal IP address of the current node. The result and IP address example will be presented in Instruction 13. The IP addresses of the node are `127.0.0.1` and `192.168.0.6`. The latter one is the one that will be used in the next step, since `127.0.0.1` corresponds to localhost.

   Instruction 13: ip a execution

```
hduser@deb0:/home/hduser$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    ↪ qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
    ↪ group default qlen 1000
    link/ether 08:00:27:ae:c5:08 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.6/24 brd 192.168.0.255 scope global enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:feae:c508/64 scope link
       valid_lft forever preferred_lft forever
```

2. Set the hostname
   Hostname is the identifier of the current machine within the network. It is defined in `/etc/hostname`. To edit it, the `root` permission will be required. It only contains a single line (usually) which defines the hostname. Then, the name could be changed as desired as depicted from 14 to 15. Then, while still using the `root` permission, `reboot` command should be executed to observe the change.

Instruction 14: /etc/hostname (before)

```
<something>
```

Instruction 15: /etc/hostname (after)

```
deb0
```

3. Revisit the routing tables
   Using the preferred hostname, it is essential to update the routing tables configured in `/etc/hosts`. Only by adding up lines represented in Instruction 16 (highlighted text was the new addition), which contains new addition of host route using the IP address and hostname from previous steps.

Instruction 16: /etc/hosts addition

```
127.0.0.1    localhost
192.168.0.6 deb0
#127.0.1.1    <hostname>.<something>.<something>.<something>    <hostname>
#The following lines are desirable for IPv6 capable hosts
#::1     localhost ip6-localhost ip6-loopback
#ff02::1    ip6-allnodes
#ff02::2    ip6-allrouters
```

3.4.3 Hadoop core configuration

Core configuration of Hadoop is used for every Hadoop instance as its base of execution. The configuration file can be found in `/usr/hadoop/hadoop-2.8.1/etc/hadoop/core-site.xml` (or in `$HADOOP_HOME/etc/hadoop/core-site.xml` if the `.bashrc` was already configured).
Since the Hadoop directories already owned by *hduser*, changing the permission to `root` would not be needed (and may lead to several errors). The XML file must be written as illustrated in Instruction 17. Then, the

file must be saved. The common port used for HDFS is 54310 or 9000 as seen at fourth line.**(Note: this is configured for multi cluster node, it differs from the single cluster node)**

Instruction 17: core-site.xml configuration (NameNode)

```
<configuration>
    <property>
        <name>fs.default.name</name>
        <value>hdfs://deb0:54310</value>
    </property>
    <property>
        <name>hadoop.tmp.dir</name>
        <value>/home/hduser/hadoop_data/hd-data/tmp</value>
    </property>
    <property>
        <name>fs.checkpoint.dir</name>
        <value>/home/hduser/hadoop_data/hd-data/snn</value>
    </property>
    <property>
        <name>dfs.data.dir</name>
        <value>/home/hduser/hadoop_data/hd-data/dn</value>
    </property>
</configuration>
```

Then, the directory `hadoop_data` should be created inside the *hduser* home directory. Using the *hduser* privileges, its creation is executed using Instruction 18.

Instruction 18: Creating hadoop_data directory (role: hduser)

```
mkdir /home/hduser/hadoop_data
```

After that, temporary files of Hadoop in `/tmp/hadoop` should be deleted using `root` privileges as depicted in Instruction 19

Instruction 19: Deleting /tmp/hadoop (role:root)

```
rm -r /tmp/hadoop*
```

This configuration will be used in both NameNode and DataNode configuration. But, it is later modified in the DataNode configuration.

3.4.4 Hadoop HDFS configuration

HDFS configuration of Hadoop is used to set some of the Hadoop Filesystem parameters. The configuration file can be found in `/usr/hadoop/hadoop-2.8.1/etc/hadoop/hdfs-site.xml` (or in `$HADOOP_HOME/etc/hadoop/hdfs-site.xml` if the `.bashrc` was already configured).
Similar to core configuration, only *hduser* permission needed to edit the file. The XML file must be written as illustrated in Instruction 20. The value of the replication should be set lower than or equal to the number of DataNode(s) prepared. As explained in 3.3, this tutorial will prepare 1 NameNode and 3 DataNodes. Thus, the value 3 as seen in fourth line of **??** could be written in the configuration, however the value 4 or higher should not be used. **(Note: this is configured for multi cluster node, it differs from the single cluster node)**

Instruction 20: hdfs-site.xml configuration (NameNode)

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>3</value>
    </property>
</configuration>
```

This configuration will be used in both NameNode and DataNode configuration. But, it is later modified in the DataNode configuration.

3.4.5 Hadoop YARN configuration

YARN (Yet Another Resource Negotiator) configuration of Hadoop is used to set some of the Hadoop Resource Management parameters. The configuration file can be found in `/usr/hadoop/hadoop-2.8.1/etc/hadoop/yarn-site.xml` (or in `$HADOOP_HOME/etc/hadoop/yarn-site.xml` if the `.bashrc` was already configured).
Similar to core and HDFS configuration, only *hduser* permission needed to edit the file. The XML file must be written as illustrated in Instruction 21. The value of the replication should be set lower than or equal to the number of DataNode(s) prepared. As explained in 3.3, this tutorial will prepare 1 NameNode. As in the hostname editing step, the NameNode is named "deb0". Thus, the value "deb0" as seen in fourth line of 21 represents the IP address of the NameNode. **(Note: this is configured for multi cluster node, it differs from the single cluster node)**

Instruction 21: yarn-site.xml configuration (NameNode/DataNode)

```
<configuration>
    <property>
        <name>yarn.resourcemanager.hostname</name>
        <value>deb0</value>
        <description>The hostname of the RM.</description>
    </property>
    <property>
        <name>yarn.nodemanager.aux-services</name>
     <value>mapreduce_shuffle</value>
    </property>
</configuration>
```

This configuration will be used in both NameNode and DataNode configuration. DataNode's YARN configuration does not differ from the NameNode's YARN configuration.

3.4.6 Hadoop MapReduce configuration

MapReduce configuration of Hadoop is used to set some of the Hadoop MapReduce parameters. Those parameters will be used as the base of WordCount program. Since the `mapred-site.xml` does not exist in `/usr/hadoop/hadoop-2.8.1/etc/hadoop/` (or in `$HADOOP_HOME/etc/hadoop/`, the Instruction 22 must be executed.

Instruction 22: Copying mapred-site.xml configuration from mapred-site.xml.template

```
cat /usr/hadoop/hadoop-2.8.1/etc/hadoop/mapred-site.xml.template >> /usr/hadoop/hadoop
    ↪ -2.8.1/etc/hadoop/mapred-site.xml
```

Similar to core and HDFS configuration, only *hduser* permission needed to edit the file. The XML file must be written as illustrated in Instruction 23. The value of the replication should be set lower than or equal to the number of DataNode(s) prepared. As explained in 3.3, this tutorial will prepare 1 NameNode. As in the hostname editing step, the NameNode is named "deb0". Thus, the value "deb0" as seen in fourth line of 23 represents the IP address of the NameNode. **(Note: this is configured for multi cluster node, it differs from the single cluster node)**

Instruction 23: mapred-site.xml configuration (NameNode/DataNode)

```
<configuration>
     <property>
          <name>mapreduce.framework.name</name>
          <value>yarn</value>
     </property>
     <property>
          <name>mapred.job.tracker</name>
          <value>deb0:54311</value>
     </property>
</configuration>
```

This configuration will be used in both NameNode and DataNode configuration. DataNode's MapReduce configuration does not differ from the NameNode's MapReduce configuration.

3.4.7 Creating logs directory

To keeping track of the Hadoop execution, it is mandatory to create `logs` directory which located in `/usr/hadoop/hadoop-2.8.1/` (or in `$HADOOP_HOME` by Instruction 24. The *hduser* user account should be active when creating it.

Instruction 24: Creating directory logs directory

```
mkdir $HADOOP_HOME/logs
```

3.5 Cloning the basic instance as First DataNode

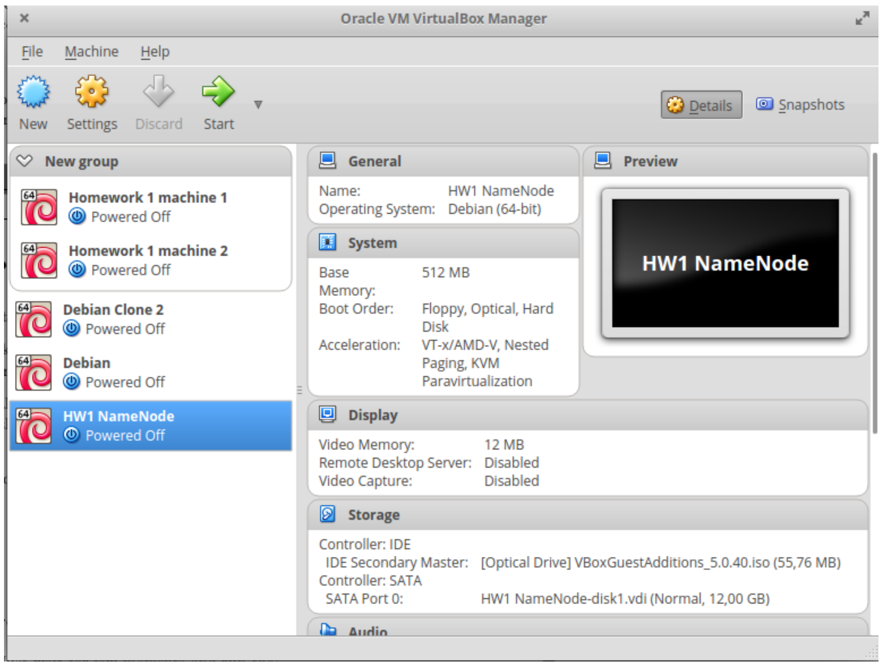
Cloning virtual machine in VirtualBox is as easy as few clicks [1]. These steps will be also later referred in adding new DataNode. Starting from the VirtualBox GUI, the steps are as belows:

1. Right-Click the preferred machine as cloning base, then click "Clone". In this case: the machine which hostname is "deb0" (HW1 NameNode) as depicted in Figure 4
2. **New machine name**. Set the new machine name and make sure the "Reinitialize the MAC address of all network cards" is Checked, then Click "Next¿". In this case: the new machine name is "HW1 DataNode1" as depicted in Figure 5
3. **Clone type**. Choose "Full Clone", then Click "Next¿" as depicted in Figure6
4. Just wait until it finishes as depicted in Figure ...

3.6 Configuring First DataNode

3.6.1 Configuring Hostname

The first things to do are starting the machine, checking IP address, and changing the hostname. The step to check IP Address is similar to Instruction 13. The desired IP address might not appear at first since it need to be fetched automatically from the main routing table. Just wait for few moments and run the `ip a` command again.



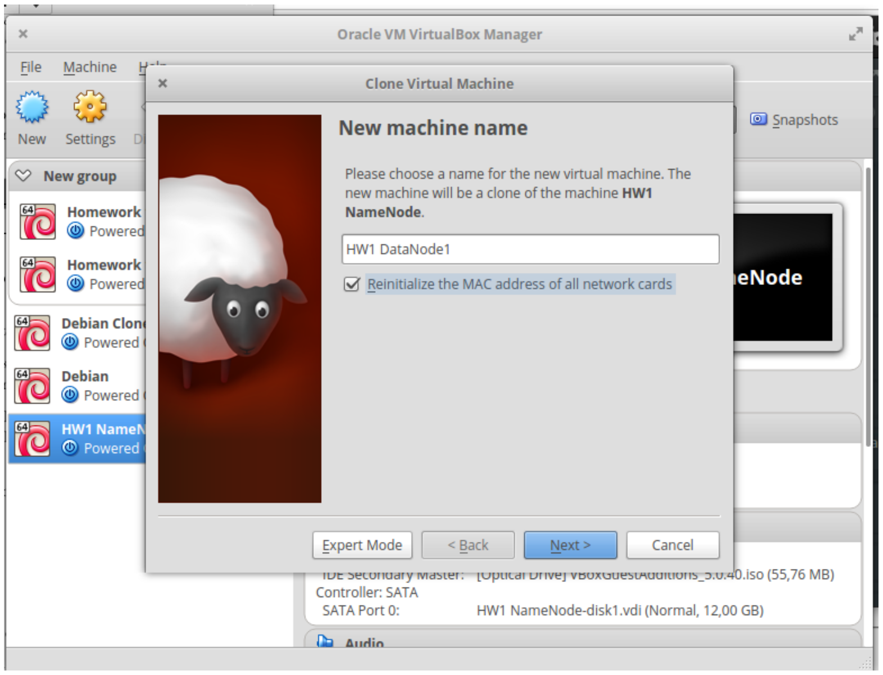
Name: HW1 NameNode
Operating System: Debian (64-bit)

Base Memory: 512 MB
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, KVM Paravirtualization

Video Memory: 12 MB
Remote Desktop Server: Disabled
Video Capture: Disabled

Controller: IDE
IDE Secondary Master: [Optical Drive] VBoxGuestAdditions_5.0.40.iso (55,76 MB)
Controller: SATA
SATA Port 0: HW1 NameNode-disk1.vdi (Normal, 12,00 GB)

Figure 4: VirtualBox Home GUI Appearance



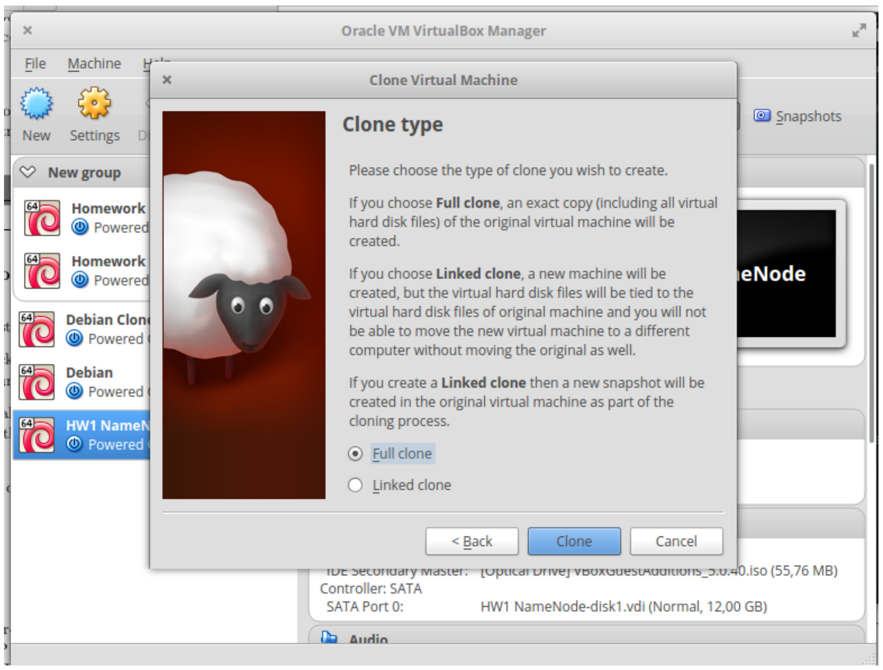
New machine name

Please choose a name for the new virtual machine. The new machine will be a clone of the machine **HW1 NameNode**.

HW1 DataNode1

☑ Reinitialize the MAC address of all network cards

Figure 5: VirtualBox Naming the Clone



Clone type

Please choose the type of clone you wish to create.

If you choose **Full clone**, an exact copy (including all virtual hard disk files) of the original virtual machine will be created.

If you choose **Linked clone**, a new machine will be created, but the virtual hard disk files will be tied to the virtual hard disk files of original machine and you will not be able to move the new virtual machine to a different computer without moving the original as well.

If you create a **Linked clone** then a new snapshot will be created in the original virtual machine as part of the cloning process.

[✓ Full clone] Linked clone

IDE Secondary Master: [Optical Drive] VBoxGuestAdditions_5.0.40.iso (55,76 MB)
Controller: SATA
SATA Port 0: HW1 NameNode-disk1.vdi (Normal, 12,00 GB)

Figure 6: VirtualBox Clone Type Window

Then, the hostname of this new machine should be also configured. The instructions used here are similar to Instruction 14 to 15. This hostname will be should be changed to ”deb1” although any name is also good. Finally, reboot and keep track of the IP address and the hostnames. They will be used later in revisiting the NameNode and DataNodes.

3.6.2 Configuring Core and HDFS Configuration

There is a slight change in the `core-site.xml`. The contents of `core-site.xml` should not contain the `hadoop.tmp.dir`, `fs.checkpoint.dir`, and `dfs.data.dir`. Then, using instructions similar to 17, the contents of the new `core-site.xml` of the DataNode should appear like Instruction 25.

Instruction 25: mapred-site.xml configuration (DataNode)

```
<configuration>
     <property>
          <name>fs.default.name</name>
          <value>hdfs://deb0:54310</value>
     </property>
</configuration>
```

Next, there is also slight change in the `hdfs-site.xml`. The value of the replication in DataNode should be changed to 1. Then, the contents of DataNode’s `hdfs-site.xml` should appear like Instruction 26.

Instruction 26: hdfs-site.xml configuration (DataNode)

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

```
</configuration>
```

3.7 Adding New DataNode

This part is similar to part 3.5 and 3.6. The differences are listed below.

1. Referring to section 3.5, the basic instance in this case is: the machine which hostname is ”deb1” (HW1 DataNode1)

2. Since the number of DataNode to be added is 3, 1 already configured, then 2 must be cloned. Let's name them ”HW1 DataNode2” and ”HW1 DataNode3”. Just clone them in any order.

3. Referring to section 3.7, the only step need to be done is checking IP Address and configuring hostname. The `core-site.xml` and `hdfs-site.xml` do not need to be configured because their contents would be same.

Note: if the DataNode scheme is not equal to 3, any customization regarding those steps is applicable, just repeat the cloning until desired number achieved.

3.8 Revisiting the NameNode and all DataNodes

3.8.1 Reconfiguring the /etc/hosts

Before this part started, all machines including NameNode and DataNodes should be running and acquired their IP Address. Then, the first directions to be done is configuring the routing table. Each machine's `/etc/hosts` should contain every machine's (NameNode and DataNodes) IP Address and hostname. Similar to steps in section 3.4.2 to edit the `/etc/hosts` before (as root), the desired `/etc/hosts` in each machine should be like Instruction 27.

Instruction 27: /etc/hosts content (NameNode and All DataNodes)

```
127.0.0.1    localhost
#127.0.1.1    <hostname>.<something>.<something>.<something>    <hostname>
192.168.0.6    deb0
192.168.0.11 deb1
192.168.0.13 deb2
192.168.0.12 deb3

#The following lines are desirable for IPv6 capable hosts
#::1    localhost ip6-localhost ip6-loopback
#ff02::1    ip6-allnodes
#ff02::2    ip6-allrouters
```

3.8.2 Sending NameNode's SSH Key to DataNodes

To access the DataNodes *hdusers* account, NameNode must send its SSH public key to the NameNodes. In addition, it must send its SSH public key to itself and address `0.0.0.0` as secondary NameNode This can be performed using Instruction 28. The answer should be ”yes” when prompted. To test it, the command `ssh hduser@<hostname>` can be used. Then, to exit the SSH connection command `exit` can be used.

Instruction 28: Sending NameNode's public key to all DataNodes and locals

```
ssh-copy-id -i $HOME/.ssh/id_rsa.pub hduser@deb0
ssh-copy-id -i $HOME/.ssh/id_rsa.pub hduser@deb1
```

```
ssh-copy-id -i $HOME/.ssh/id_rsa.pub hduser@deb2
ssh-copy-id -i $HOME/.ssh/id_rsa.pub hduser@deb3
ssh-copy-id -i $HOME/.ssh/id_rsa.pub hduser@0.0.0.0
```

3.8.3 Updating contents of slaves/DataNodes configuration

It is required to inform the Hadoop NameNode its DataNodes. This configuration resides in `/usr/hadoop/hadoop-2.8.1/etc/hadoop/slaves` or `$HADOOP_HOME/etc/hadoop/slaves`. Some lines should be appended in it as in Instruction 29, specifying all the hostnames or IP address of the DataNodes.

Instruction 29: Adding DataNodes hostnames

```
deb1
deb2
deb3
```

3.9 Formatting the HDFS

One last step before running Hadoop is formatting the filesystem which resides in Hadoop directory as HDFS. This step is listed in Instruction 30. This should be done as *hduser* in NameNode (deb0).

Instruction 30: Formatting HDFS

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

Note: Should there be any node added after this step, the whole Hadoop configuration must be repeated from Section 3.7 to 3.9 before running Hadoop.

3.10 Running Hadoop

Hadoop execution file is located in the `sbin` in the Hadoop folder, which can be executed using Instruction 31

Instruction 31: Starting Hadoop

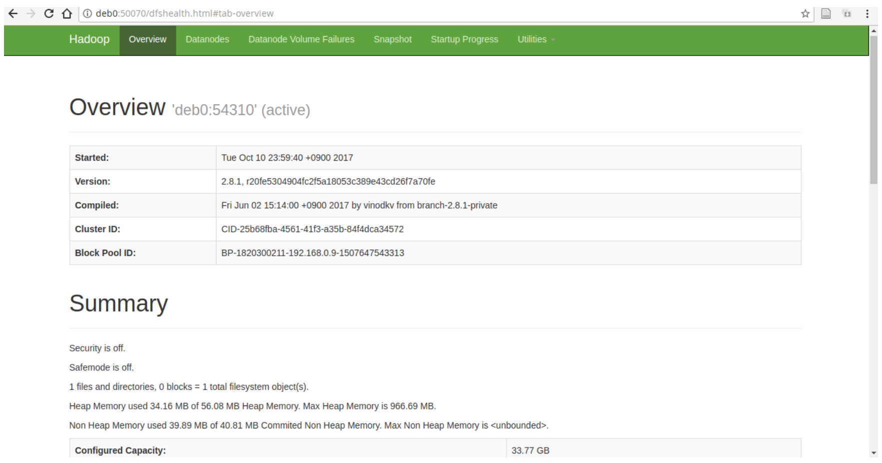
```
$HADOOP_HOME/sbin/start-dfs.sh
```

The results should be depicted as in Figure7.

```
Starting namenodes on [deb0]
deb0: starting namenode, logging to /usr/hadoop/hadoop-2.8.1/logs/hadoop-hduser-namenode-deb0.out
deb1: starting datanode, logging to /usr/hadoop/hadoop-2.8.1/logs/hadoop-hduser-datanode-deb1.out
deb2: starting datanode, logging to /usr/hadoop/hadoop-2.8.1/logs/hadoop-hduser-datanode-deb2.out
deb3: starting datanode, logging to /usr/hadoop/hadoop-2.8.1/logs/hadoop-hduser-datanode-deb3.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /usr/hadoop/hadoop-2.8.1/logs/hadoop-hduser-secondarynamenode-deb0.out
```

Figure 7: The terminal output window when executing start-dfs.sh

In the other hand, the running nodes' status can be checked via web interface in `http://deb0:50070/` as depicted in Figure 8



Overview 'deb0:54310' (active)

Started:	Tue Oct 10 23:59:40 +0900 2017
Version:	2.8.1, r20fe5304904fc2f5a18053c389e43cd26f7a70fe
Compiled:	Fri Jun 02 15:14:00 +0900 2017 by vinodkv from branch-2.8.1-private
Cluster ID:	CID-25b68fba-4561-41f3-a35b-84f4dca34572
Block Pool ID:	BP-1820300211-192.168.0.9-1507647543313

Summary

Security is off.

Safemode is off.

1 files and directories, 0 blocks = 1 total filesystem object(s).

Heap Memory used 34.16 MB of 56.08 MB Heap Memory. Max Heap Memory is 966.69 MB.

Non Heap Memory used 39.89 MB of 40.81 MB Commited Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	33.77 GB

Figure 8: NameNodes Web Interface

3.11 Running YARN

YARN execution file is located in the `sbin` in the Hadoop folder, which can be executed using Instruction 32

Instruction 32: Starting YARN

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

The results should be depicted as in the Figure9. In the other hand, the running nodes' resource status can

```
starting yarn daemons
starting resourcemanager, logging to /usr/hadoop/hadoop-2.8.1/logs/yarn-hduser-resourcemanager-deb0.out
deb1: starting nodemanager, logging to /usr/hadoop/hadoop-2.8.1/logs/yarn-hduser-nodemanager-deb1.out
deb3: starting nodemanager, logging to /usr/hadoop/hadoop-2.8.1/logs/yarn-hduser-nodemanager-deb3.out
deb2: starting nodemanager, logging to /usr/hadoop/hadoop-2.8.1/logs/yarn-hduser-nodemanager-deb2.out
```

Figure 9: The terminal output window when executing start-yarn.sh

be checked via web interface in `http://deb0:8088/` as depicted in Figure 10

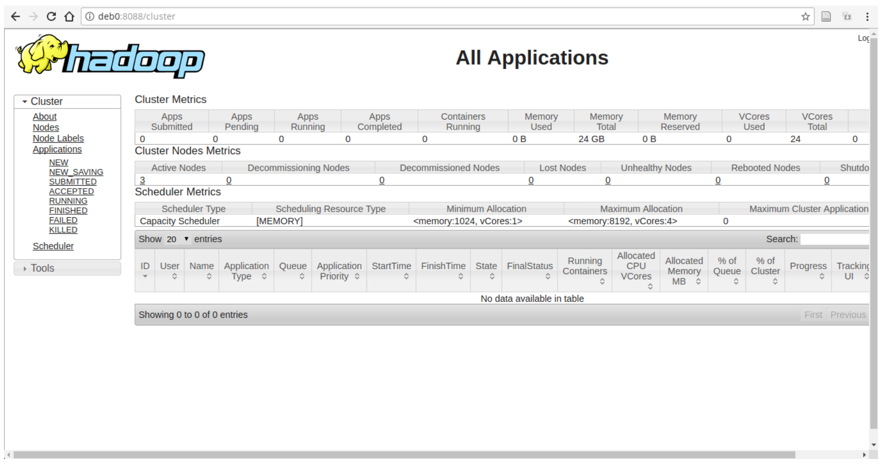
3.12 Debugging Hadoop

As the `logs` directory created in `$HADOOP_HOME` of each machine, all the INFO, WARN, and ERROR logs are recorded. Thus, the behaviour of NameNode and each DataNode can be investigated through the logs. The example of the log file in "deb0" NameNode is depicted in Figure 11.

3.13 Stopping YARN and Hadoop

To stop the YARN and Hadoop completely, YARN must be stopped first using Instruction 33.

Instruction 33: Stopping YARN



deb0:8088/cluster

All Applications

Cluster
- About
- Nodes
- Node Labels
- Applications
  - NEW
  - NEW_SAVING
  - SUBMITTED
  - ACCEPTED
  - RUNNING
  - FINISHED
  - FAILED
  - KILLED
- Scheduler

Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	
0	0	0	0	0	0 B	24 GB	0 B	0	24	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdo
3	0	0	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0

Show 20 entries Search:

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	% of Queue	% of Cluster	Progress	Tracking UI
No data available in table																

Showing 0 to 0 of 0 entries First Previous

Figure 10: Running Cluster Web Interface

```
/usr/hadoop/hadoop-2.8.1/logs/hadoop-hduser-namenode-deb0.log
/usr/hadoop/hadoop-2.8.1/logs/hadoop-hduser-secondarynamenode-deb0.log
```

Figure 11: The example of log files in "deb0"

```
$HADOOP_HOME/sbin/stop-yarn.sh
```

The results should be depicted as in the Figure 12.
Then, Hadoop could be stopped first using Instruction 34.

```
stopping yarn daemons
stopping resourcemanager
deb0: stopping nodemanager
deb1: nodemanager did not stop gracefully after 5 seconds: killing with kill -9
deb2: nodemanager did not stop gracefully after 5 seconds: killing with kill -9
deb3: nodemanager did not stop gracefully after 5 seconds: killing with kill -9
no proxyserver to stop
```

Figure 12: The result of stopping YARN execution

Instruction 34: Stopping Hadoop

```
$HADOOP_HOME/sbin/stop-dfs.sh
```

The results should be depicted as in the Figure 13.

4 Running The MapReduce Job

The MapReduce is one of Hadoop's main application. **To run it, Hadoop and YARN must be run in the background** using steps mentioned before.

4.1 About the WordCount

The WordCount is a default MapReduce program which is counted as HelloWorld program of Hadoop MapReduce. It is a class located in `share/hadoop/mapreduce/hadoop-mapreduce-examples-2.`

```
Stopping namenodes on [deb0]
deb0: stopping namenode
deb3: stopping datanode
deb1: stopping datanode
deb2: stopping datanode
Stopping secondary namenodes [0.0.0.0]
0.0.0.0: stopping secondarynamenode
```

Figure 13: The result of stopping Hadoop execution

8.1.jar file of Hadoop directory.

4.2 Input file

The input file is a publicly available text file acquired from http://bmidb.cs.stonybrook.edu/publicdata/big.txt containing texts from ebooks. The size is about 6 MB. This file is stored in home folder of *hduser* and named "big.txt".

4.3 Creating input directory in HDFS

Before the WordCount program executed, Word needs two arguments: `input` and `output`. Those two arguments specifies the input directory and output directory in HDFS respectively. However, the `input` directory should be created first while the `output` directory will be automatically created when the WordCount program runs.
To create the input directory, while the Hadoop run in background. Instruction 35 must be executed. Make sure that the current directory is the home directory of *hduser*

Instruction 35: Creating input directory in HDFS

```
hdfs dfs -mkdir /user
hdfs dfs -mkdir /user/hduser
hdfs dfs -mkdir /user/hduser/input
```

4.4 Copying input file into HDFS

Then, the input file "big.txt" can be copied using Instruction 36.

Instruction 36: Copying input file into HDFS

```
hdfs dfs -put big.txt /user/hduser/input
```

4.5 Running the WordCount program

After that, the WordCount could be run using Instruction 37.

Instruction 37: Running Wordcount in Hadoop

```
hadoop jar /usr/hadoop/hadoop-2.8.1/share/hadoop/mapreduce/hadoop-mapreduce-examples
    ↪ -2.8.1.jar wordcount input output
```

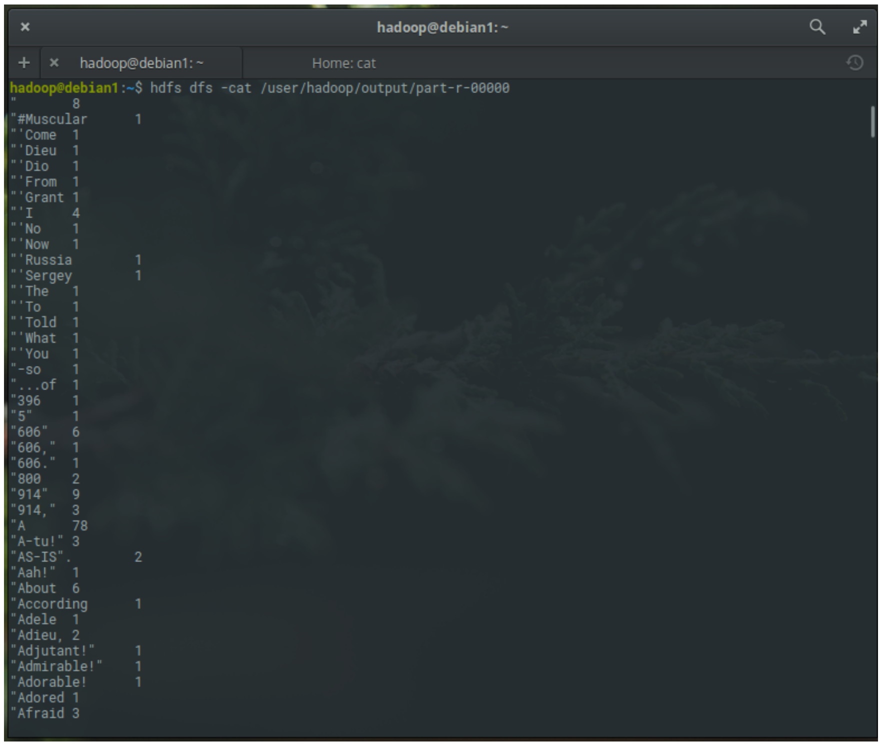
The result in the terminal window should be similar to Figure 14 and Figure 15.

```
17/10/10 21:09:52 INFO client.RMProxy: Connecting to ResourceManager at deb0/192.168.0.6:8032
17/10/10 21:09:56 INFO input.FileInputFormat: Total input files to process : 1
17/10/10 21:09:57 WARN hdfs.DataStreamer: Caught exception
java.lang.InterruptedException
at java.lang.Object.wait(Native Method)
at java.lang.Thread.join(Thread.java:1252)
at java.lang.Thread.join(Thread.java:1326)
at org.apache.hadoop.hdfs.DataStreamer.closeResponder(DataStreamer.java:927)
at org.apache.hadoop.hdfs.DataStreamer.endBlock(DataStreamer.java:578)
at org.apache.hadoop.hdfs.DataStreamer.run(DataStreamer.java:755)
17/10/10 21:09:57 INFO mapreduce.JobSubmitter: number of splits:1
17/10/10 21:09:57 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1507637359884_0001
17/10/10 21:09:59 INFO impl.YarnClientImpl: Submitted application application_1507637359884_0001
17/10/10 21:09:59 INFO mapreduce.Job: The url to track the job:
http://deb0:8088/proxy/application_1507637359884_0001/
17/10/10 21:09:59 INFO mapreduce.Job: Running job: job_1507637359884_0001
17/10/10 21:10:19 INFO mapreduce.Job: Job job_1507637359884_0001 running in uber mode : false
17/10/10 21:10:19 INFO mapreduce.Job: map 0% reduce 0%
17/10/10 21:10:39 INFO mapreduce.Job: map 100% reduce 0%
17/10/10 21:10:53 INFO mapreduce.Job: map 100% reduce 100%
17/10/10 21:10:54 INFO mapreduce.Job: Job job_1507637359884_0001 completed successfully
17/10/10 21:10:54 INFO mapreduce.Job: Counters: 49
File System Counters
FILE: Number of bytes read=1269408
FILE: Number of bytes written=2811649
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=6501072
HDFS: Number of bytes written=951085
HDFS: Number of read operations=6
HDFS: Number of large read operations=0
HDFS: Number of write operations=2
Job Counters
Launched map tasks=1
Launched reduce tasks=1
Data-local map tasks=1
Total time spent by all maps in occupied slots (ms)=16350
Total time spent by all reduces in occupied slots (ms)=10885
Total time spent by all map tasks (ms)=16350
Total time spent by all reduce tasks (ms)=10885
Total vcore-milliseconds taken by all map tasks=16350
Total vcore-milliseconds taken by all reduce tasks=10885
Total megabyte-milliseconds taken by all map tasks=16742400
Total megabyte-milliseconds taken by all reduce tasks=11146240
```

Figure 14: The output in terminal window of WordCount execution (part 1)

```
Map-Reduce Framework
Map input records=128444
Map output records=1095562
Map output bytes=10815949
Map output materialized bytes=1269408
Input split bytes=111
Combine input records=1095562
Combine output records=82264
Reduce input groups=82264
Reduce shuffle bytes=1269408
Reduce input records=82264
Reduce output records=82264
Spilled Records=164528
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=448
CPU time spent (ms)=8630
Physical memory (bytes) snapshot=311062528
Virtual memory (bytes) snapshot=3856465920
Total committed heap usage (bytes)=138354688
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=6500961
File Output Format Counters
Bytes Written=951085
```

Figure 15: The output in terminal window of WordCount execution (part 2)



```
hadoop@debian1:~$ hdfs dfs -cat /user/hadoop/output/part-r-00000
"       8
"#Muscular      1
"'Come  1
"'Dieu  1
"'Dio   1
"'From  1
"'Grant 1
"'I     4
"'No    1
"'Now   1
"'Russia        1
"'Sergey        1
"'The   1
"'To    1
"'Told  1
"'What  1
"'You   1
"-so    1
"...of  1
"396    1
"5"     1
"606"   6
"606,"  1
"606."  1
"800    2
"914"   9
"914,"  3
"A      78
"A-tu!" 3
"AS-IS".        2
"Aah!"  1
"About  6
"According      1
"Adele  1
"Adieu, 2
"Adjutant!"     1
"Admirable!"    1
"Adorable!      1
"Adored 1
"Afraid 3
```

Figure 16: The output of MapReduce WordCount

4.6 Getting the output

The output file resides in `/user/hduser/output` directory of HDFS. The small portion of the output is depicted in Figure 16. To acquire the output file from HDFS to local, the Instruction 38 can be executed while Hadoop is running. The output in local file is stored as `wordcount-output.txt` in home directory of *hduser*.

Instruction 38: Copying output file from HDFS to local

```
hdfs dfs -get  /user/hduser/output wordcount-output.txt
```

4.7 Removing the output directory in HDFS

In further execution of MapReduce, it is required to delete the output directory in HDFS to prevent some errors. To perform the deletion, Instruction 39 can be executed.

Instruction 39: Deleted output directory in HDFS

```
hdfs dfs -get  /user/hduser/output wordcount-output.txt
```

References

[1] Virtualbox chapter1.First steps, . URL http://www.virtualbox.org/manual/ch01.html#clone.

[2] Virtualbox chapter6.Virtual networking, . URL https://www.virtualbox.org/manual/ch06.html.

[3] HadoopJavaVersions - Hadoop Wiki, . URL https://wiki.apache.org/hadoop/HadoopJavaVersions.

[4] How to Setup Hadoop Multi-Node Cluster on Ubuntu, October 2016. URL https://linoxide.com/cluster/setup-hadoop-multi-node-cluster-ubuntu/.

[5] Mathias Montantin. Installation d'Hadoop sur Debian, September 2017. URL https://cdiese.fr/installation-dhadoop-sur-debian/.