Aims

This exercise aims to get you to:

- Install and configure Hadoop MapReduce
- Practice HDFS operations
- Test Hadoop MapReduce with the pseudo-distributed mode

Background

In the examples below, we have used the $ sign to represent the prompt from the command interpreter (shell). The actual prompt may look quite different on your computer (e.g. it may contain the computer's hostname, or your username, or the current directory name). Whenever the word “edit” is used, this means that you could use your favorite text editor (e.g.vim, emacs, gedit, etc.).

A virtual machine running Ubuntu 22.04 is provided. Both user name and password is comp9313. You can download the image from the following links:

https://mega.nz/file/SqIz1Jpb#Ay5ioC4EkiQgZVuVYDUL6hfO2LiBvsJxjTXX1qAnxrg

https://drive.google.com/file/d/1ymUkS422jiNnEKU2witPb2fIL8wf6eME/view

The sudo password is also comp9313 in the system. Please follow the instructions in “VM image.pdf” to download and configure the virtual machine.

Today’s lab aims to let you practice how to install and configure Hadoop.

Configure Hadoop and HDFS

0. Install Java (if already installed the correct version, you can skip this step)

Hadoop 3.3.2 requires Java 8 or 11. Install Java jdk by the following command:

```
$ sudo apt install openjdk-11-jdk
```

1. Download Hadoop and Configure HADOOP_HOME

Download the Hadoop package by the command (use the aarch64 package for Apple M1):

```
$ wget https://dlcdn.apache.org/hadoop/common/hadoop-3.3.2/hadoop-3.3.2.tar.gz
```

Then unpack the package:

```
$ tar xvf hadoop-3.3.2.tar.gz
```

Now you have Hadoop installed under `~/hadoop-3.3.2`. We need to configure this folder as the working directory of Hadoop, i.e., `HADOOP_HOME`.

Use the following command to install gedit if it is not installed yet (sudo password is comp9313):

```
$ sudo apt install gedit
```

Open the file `~/.bashrc` using gedit (or use vim or emacs if you are familiar with them):

```
$ gedit ~/.bashrc
```

Then add the following lines to the **end** of this file:

```
export HADOOP_HOME=/home/comp9313/hadoop-3.3.2
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
```

Save the file, and then run the following command to take these configurations into effect:

```
$ source ~/.bashrc
```

Important: Check if the HADOOP_HOME is correctly configured by:

```
$ echo $HADOOP_HOME
```

You should see:

```
/home/comp9313/hadoop-3.3.2
```

2. Configure HDFS

We first open the hadoop environment file, hadoop-env.sh, using:

```
$ gedit $HADOOP_CONF_DIR/hadoop-env.sh
```

and add the following to the **end** of this file

```
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
```

Then open the HDFS core configuration file, core-site.xml, using:

```
$ gedit $HADOOP_CONF_DIR/core-site.xml
```

Note that it is in xml format, and every configuration should be put in between <configuration> and </configuration>. You need to add the following lines:

```
<property>
        <name>hadoop.tmp.dir</name>
        <value>file:/home/comp9313/hadoop-3.3.2/tmp</value>
</property>

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

For more configuration details please refer to:

https://hadoop.apache.org/docs/r3.3.2/hadoop-project-dist/hadoop-common/core-default.xml

Finally open the configuration file hdfs-site.xml, using:

```
$ gedit $HADOOP_CONF_DIR/hdfs-site.xml
```

You need to add the following lines between <configuration> and </configuration>:

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

<property>
        <name>dfs.namenode.name.dir</name>
        <value>file:/home/comp9313/hadoop-3.3.2/tmp/dfs/name</value>
</property>

<property>
        <name>dfs.datanode.data.dir</name>
        <value> file:/home/comp9313/hadoop-3.3.2/tmp/dfs/data</value>
</property>
```

Now you have already done the basic configuration of HDFS, and it is ready to use.

For more configuration details please refer to;

https://hadoop.apache.org/docs/r3.3.2/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml

Start HDFS

0. install and configure ssh (if already installed and configured, you can skip this step)

If you cannot ssh to localhost, you need to do the following configurations:

```
$ sudo apt install ssh
$ ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
$ chmod 0600 ~/.ssh/authorized_keys
```

1. Work in the Hadoop home folder.

```
$ cd $HADOOP_HOME
```

Format the NameNode (the master node):

```
$ hdfs namenode -format
```

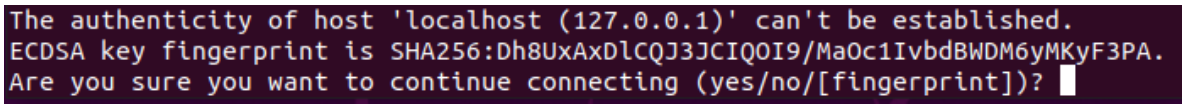
You should see the output like below if successful:

```
2021-09-01 14:21:49,601 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
2021-09-01 14:21:49,757 INFO namenode.FSNamesystem: Stopping services started for active state
2021-09-01 14:21:49,758 INFO namenode.FSNamesystem: Stopping services started for standby state
2021-09-01 14:21:49,767 INFO namenode.FSImage: FSImageSaver clean checkpoint: txid=0 when meet shutdown.
2021-09-01 14:21:49,777 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at comp9313-VirtualBox/127.0.1.1
************************************************************/
```

Start HDFS in the virtual machine using the following command:

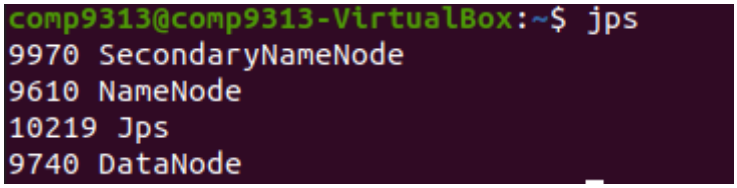
```
$ start-dfs.sh
```

If you see below,



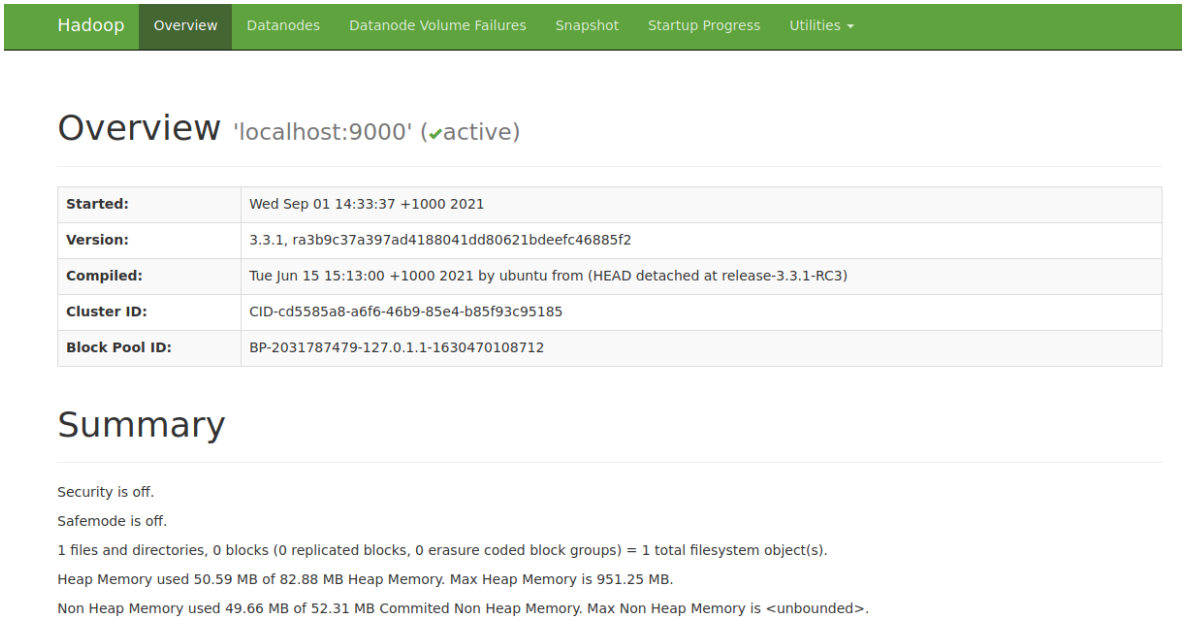
you just need to input “yes” to continue.

2. Use the command “jps” to see whether Hadoop has been started successfully. You should see something like below:



Note that you should have “NameNode”, “DataNode” and “SecondaryNameNode”.

3. You can browse the web interface for the information of NameNode and DataNode at: http://localhost:9870. You will see:



Using HDFS

1. Make the HDFS directories required to execute MapReduce jobs:

```
$ hdfs dfs -mkdir -p /user/comp9313
```

Folders are created upon HDFS, rather than local file systems. After creating these folders, the `/user/comp9313` is now the default working folder in HDFS. That is, you can create/get/copy/list (and more operations) files/folders without typing `/user/comp9313` every time. For example, we can use

```
$ hdfs dfs -ls
```

instead of

```
$ hdfs dfs -ls /user/comp9313
```

to list files in `/user/comp9313`.

2. Make a directory input to store files:

```
$ hdfs dfs -mkdir input
```

Remember `/user/comp9313` is our working folder. Thus, the directory `input` is created under `/user/comp9313`, that is: `/user/comp9313/input`. Check the `input` folder exists using

```
$ hdfs dfs -ls
```

3. Copy the input files into the distributed filesystem:

```
$ hdfs dfs -put $HADOOP_HOME/etc/hadoop/*.xml input
```

We will copy all xml files in the directory of `$HADOOP_HOME/etc/hadoop` on the local file system to the directory of `/user/comp9313/input` on HDFS. After you copy all the files, you can use the following command to list the files in `input`:

```
$ hdfs dfs -ls input
```

4. Please find more commands of HDFS operations here:

https://hadoop.apache.org/docs/r3.3.2/hadoop-project-dist/hadoop-common/FileSystemShell.html

and try these commands to operate the HDFS files and/or folders. At least you should familiar with the following commands in this lab:

```
get, put, cp, mv, rm, mkdir, cat
```

Running MapReduce in the pseudo-distributed mode

Now Hadoop has been configured to the pseudo-distributed mode, where each Hadoop daemon runs in a separate Java process. This is useful for debugging.

1. The hadoop-mapreduce-examples-3.3.2.jar is a package of classic MapReduce implementations including `wordcount, grep, pi,` etc. You can explore by checking the available applications as:

```
$ hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.2.jar
```

2. Choose one that you are interested in and investigate the specific usage. For example, to run the pi example with 16 maps and 100000 samples:

```
$ hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.2.jar
pi 16 10000
```

You will see the results:

```
        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=1888
        File Output Format Counters
                Bytes Written=97
Job Finished in 15.499 seconds
Estimated value of Pi is 3.14127500000000000000
```

If you want to search for all the strings starting with 'dfs' in the xml files stored in HDFS, you can try the following command:

```
$ hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.2.jar grep input output 'dfs[a-z.]+'
```

Please see here http://www.cyberciti.biz/faq/howto-use-grep-command-in-linux-unix/ if you are not familiar with the grep command. The results are stored in the directory "`output`" in HDFS. Copy the output files from the distributed filesystem to the local filesystem and examine the results:

```
$ hdfs dfs -get output output
$ cat output/*
```

Or, you can examine them on HDFS directly:

```
$ hdfs dfs -cat output/*
```

You can see the results like below:

```
comp9313@comp9313-VirtualBox:~$ hdfs dfs -cat output/*
1       dfsadmin
1       dfs.replication
1       dfs.namenode.name.dir
1       dfs.datanode.data.dir
```

3. Check the usage of wordcount by running:

```
$ hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.2.jar wordcount
```

You will notice that the application would need an input file and an output file as arguments, which are the inputs and outputs respectively. Thus, you can use the following command to count the frequency of words from files in our `input` folder and write the results to our `output` folder:

```
$ hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.2.jar wordcount input output
```

Warning: Note that if output already exists, you will meet an exception. You need to either delete `output` on HDFS:

```
$ hdfs dfs -rm -r output
```

Or, you use another folder to store the results (e.g., `output2`). Then the results can be checked using `cat` as you did before.

Execute a job on YARN

1. Configurations

If we want to run the job in a real distributed environment, we need to borrow a hand from YARN, which manages all the computing nodes and resources of Hadoop. On a single computer, we can also run a MapReduce job on YARN in a pseudo-distributed mode by setting a few parameters and running ResourceManager daemon and NodeManager daemon in addition.

We first configure the MapReduce to use the YARN framework. Open the mapred-site.xml :

```
$ gedit $HADOOP_CONF_DIR/mapred-site.xml
```

and then add the following lines (still in between <configuration> and </configuration>):

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

<property>
  <name>yarn.app.mapreduce.am.env</name>
  <value>HADOOP_MAPRED_HOME=$HADOOP_MAPRED_HOME</value>
</property>

<property>
  <name>mapreduce.map.env</name>
  <value>HADOOP_MAPRED_HOME=$HADOOP_MAPRED_HOME</value>
</property>

<property>
  <name>mapreduce.reduce.env</name>
  <value>HADOOP_MAPRED_HOME=$HADOOP_MAPRED_HOME</value>
</property>
```

Then open the yarn-site.xml to configure yarn:

```
$ gedit $HADOOP_CONF_DIR/yarn-site.xml
```

and add the following lines:

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

2. Start YARN:

```
$ start-yarn.sh
```

3. Try jps again, you will see “NodeManager” and “ResourceManager”, and these are the main daemons of YARN.

```
comp9313@comp9313-VirtualBox:~$ jps
8498 ResourceManager
8627 NodeManager
8277 SecondaryNameNode
7944 NameNode
8909 Jps
8079 DataNode
```

4. Run the grep or wordcount example again.

You may observe that now the runtime is longer. Compared to the non-distributed execution, YARN is now managing resources and scheduling tasks. This causes some overheads. However, YARN allows us to deploy and run our applications in a cluster with up to thousands of machines, and process very large data in the real world.

If you do not want to use YARN, you can first stop YARN by running “stop-yarn.sh”, and then rename the mapred-site.xml file by:

```
$ mv $HADOOP_CONF_DIR/mapred-site.xml $HADOOP_CONF_DIR/mapred-site.xml.bak
```

5. Browse the web interface (for supervision and debugging) for the ResourceManager at: http://localhost:8088/.