NAME:Aditya Somani CLASS:TE-A ROLL NO:T1851061

ASSIGNMENT -1

Part A : Assignments based on the Hadoop

Aim: To Perform Hadoop Installation (Configuration) on a)Single Node b)Multiple Node

---

Introduction

Apache Hadoop is an open-source software framework written in Java for distributed storage and distributed processing of very large data sets on computer clusters built from commodity hardware. All the modules in Hadoop are designed with a fundamental assumption that hardware failures are common and should be automatically handled by the framework. The core of Apache Hadoop consists of a storage part, known as Hadoop Distributed File System (HDFS), and a processing part called MapReduce. Hadoop splits files into large blocks and distributes them across nodes in a cluster. To process data, Hadoop transfers packaged code for nodes to process in parallel based on the data that needs to be processed. This approach takes advantage of data locality— nodes manipulating the data they have access to— to allow the dataset to be processed faster and more efficiently than it would be in a more conventional supercomputer architecture that relies on a parallel file system where computation and data are distributed via high-speed networking. The base Apache Hadoop framework is composed of the following modules:

- Hadoop Common – contains libraries and utilities needed by other Hadoop modules;
- Hadoop Distributed File System (HDFS) – a distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster;
- Hadoop YARN – a resource-management platform responsible for managing computing resources in clusters and using them for scheduling of users' applications; and
- Hadoop MapReduce – an implementation of the MapReduce programming model for large scale data processing.

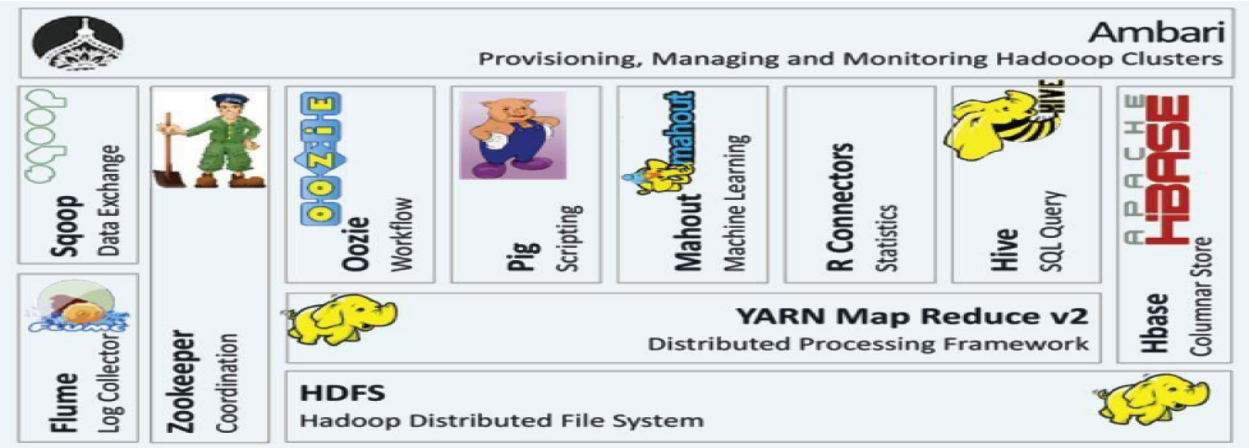

Fig.1: Hadoop Ecosystem

a)Single Node:

Steps for Compilation & Execution

sudo apt-get update
sudo apt-get install openjdk-7-jre-headless
sudo apt-get install openjdk-7-jdk sudo
apt-get install sshsudo apt-get install rsync
Download hadoop from : http://www.eu.apache.org/dist/hadoop/common/stable/hadoop-2.9.0.tar.gz
copy and extract hadoop-2.9.0.tar.gz in home folder

rename the name of the extracted folder from hadoop-2.9.0 to hadoopreadlink

-f /usr/bin/javac

find whether ubuntu is 32 bit (i686) or 64 bit (x86_64) uname
-i
gedit ~/hadoop/etc/hadoop/hadoop-env.sh
add following line in it
for 32 bit ubuntu
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-**i386**
for 64 bit ubuntu
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-**amd64**
save and exit the file
to display the usage documentation for the hadoop script try next command ~/hadoop/bin/hadoop

1. For standalone mode
mkdir input
cp ~/hadoop/etc/hadoop/*.xml input
~/hadoop/bin/hadoop jar ~/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.9.0.jar grep input output 'us[a-z.]+' cat output/*
Our task is done, so remove input and output folders rm
-r input output

 # 2. Pseudo-Distributed mode
get your user name
whoami
remember your user name, we'll use it in the next step gedit
~/hadoop/etc/hadoop/core-site.xml
<configuration>
<property>
<name>fs.defaultFS</name>
<value>hdfs://localhost:1234</value>
</property>

</configuration>

gedit ~/hadoop/etc/hadoop/hdfs-site.xml
<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.name.dir</name>
<value>file:///home/**your_user_name**/hadoop/name_dir</value>
</property>
<property>
<name>dfs.data.dir</name>
<value>file:///home/**your_user_name**/hadoop/data_dir</value>
</property>
</configuration>

#Setup passphraseless/passwordlesssshssh-keygen -t dsa -P

" -f ~/.ssh/id_dsa cat ~/.ssh/id_dsa.pub >>

~/.ssh/authorized_keys export

HADOOP_PREFIX=/home/**your_user_name**/hadoopssh

localhost

type**exit** in the terminal to close the ssh connection (very important) Exit

 # The following instructions are to run a MapReduce job locally.

#Format the filesystem:(**Do it only once**)
~/hadoop/bin/hdfsnamenode -format
#Start NameNode daemon and DataNode daemon:
~/hadoop/sbin/start-dfs.sh
#Browse the web interface for the NameNode; by default it is available at: http://localhost:50070/
#Make the HDFS directories required to execute MapReduce jobs:
~/hadoop/bin/hdfsdfs -mkdir /user
~/hadoop/bin/hdfsdfs -mkdir /user/**your_user_name**
#Copy the sample files (from ~/hadoop/etc/hadoop) into the distributed filesystem folder(input)
~/hadoop/bin/hdfsdfs -put ~/hadoop/etc/hadoop input
#Run the example map-reduce job
~/hadoop/bin/hadoop jar ~/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.1.jar grep input output 'us[a-z.]+'

#View the output files on the distributed filesystem
~/hadoop/bin/hdfsdfs -cat output/*
#Copy the output files from the distributed filesystem to the local filesystem and examine them:
~/hadoop/bin/hdfsdfs -get output output
#ignore warnings (if any) cat
output/*
remove local output folder rm
-r output
remove distributed folders (input & output)
~/hadoop/bin/hdfsdfs -rm -r input output #When
you're done, stop the daemons with
~/hadoop/sbin/stop-dfs.sh

Conclusion: In this way the Hadoop was installed & configured on Ubuntu for BigData.

b)Multiple Node

Install a Multi Node Hadoop Cluster on Ubuntu 14.04

This article is about multi-node installation of Hadoop cluster. You would need minimum of 2 ubuntu machines or virtual images to complete a multi-node installation. If you want to just try out a single node cluster, follow this article onInstalling Hadoop on Ubuntu 14.04.
I used Hadoop Stable version 2.6.0 for this article. I did this setup on a 3 node cluster. For simplicity, i will designate one node as **master,** and 2 nodes as **slaves (slave-1, and slave-2)**. Make sure all slave nodes are reachable from master node. To avoid any unreachable hosts error, make sure you add the slave hostnames and ip addresses in **/etc/hosts** file. Similarly, slave nodes should be able to resolve **master** hostname.

Installing Java on Master and Slaves

```
$ sudo add-apt-repository ppa:webupd8team/java
$ sudo apt-get update
$ sudo apt-get install oracle-java7-installer
# Updata Java runtime
$ sudo update-java-alternatives -s java-7-oracle
```

Disable IPv6

As of now Hadoop does not support IPv6, and is tested to work only on IPv4 networks. If you are using IPv6, you need to switch Hadoop host machines to use IPv4. The Hadoop Wikilink provides a one liner command to disable the IPv6. If you are not using IPv6, skip this step:

```
sudosed -i 's/net.ipv6.bindv6only\ =\ 1/net.ipv6.bindv6only\ =\ 0/' \ /etc/sysctl.d/bindv6only.conf
&&sudo invoke-rc.dprocps restart
```

Setting up a Hadoop User

Hadoop talks to other nodes in the cluster using no-password ssh. By having Hadoop run under a specific user context, it will be easy to distribute the ssh keys around in the Hadoop cluster. Lets's create a user **hadoopuser**on **master** as well as **slave** nodes.

```
# Createhadoopgroup
$ sudoaddgrouphadoopgroup
# Createhadoopuser user
$ sudoadduser —ingrouphadoopgrouphadoopuser
```

Our next step will be to generate a ssh key for password-less login between master and slave nodes. Run the following commands only on **master** node. Run the last two commands for each slave node. Password less ssh should be working before you can proceed with further steps.

```
# Login as hadoopuser
$ su - hadoopuser
#Generate a ssh key for the user
$ ssh-keygen -t rsa -P ""
#Authorize the key to enable password less ssh
$ cat /home/hadoopuser/.ssh/id_rsa.pub >> /home/hadoopuser/.ssh/authorized_keys $ chmod
600 authorized_keys
#Copy this key to slave-1 to enable password less ssh
$ ssh-copy-id -i ~/.ssh/id_rsa.pub slave-1
#Make sure you can do a password less ssh using following command. $ ssh
slave-1
```

Download and Install Hadoop binaries on Master and Slave nodes

Pick the best mirror site to download the binaries fromApache Hadoop, and download the stable/hadoop-2.6.0.tar.gz for your installation. **Do this step on master and every slave node.** You can download the file once and the distribute to each slave node using scp command.

```
$ cd /home/hadoopuser
$ wget http://www.webhostingjams.com/mirror/apache/hadoop/core/stable/hadoop-2.2.0.tar.gz $
tar xvf hadoop-2.2.0.tar.gz
$ mv hadoop-2.2.0 hadoop
```

Setup Hadoop Environment on Master and Slave Nodes

Copy and paste following lines into your .bashrc file under /home/hadoopuser. **Do this step on master and every slave node.**

```
# Set HADOOP_HOME
export HADOOP_HOME=/home/hduser/hadoop
# Set JAVA_HOME
```

```
export JAVA_HOME=/usr/lib/jvm/java-7-oracle
# Add Hadoop bin and sbin directory to PATH
export PATH=$PATH:$HADOOP_HOME/bin;$HADOOP_HOME/sbin
```

Update hadoop-env.sh on Master and Slave Nodes

Update JAVA_HOME in /home/hadoopuser/hadoop/etc/hadoop/hadoop_env.sh to following. **Do this step on master and every slave node.**

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

Common Terminologies

Before we start getting into configuration details, lets discuss some of the basic terminologies used in Hadoop.

- **Hadoop Distributed File System**: A distributed file system that provides high-throughput access to application data. A HDFS cluster primarily consists of a NameNode that manages the file system metadata and DataNodes that store the actual data. If you compare HDFS to a traditional storage structures (e.g. FAT, NTFS), then NameNode is analogous to a Directory Node structure, and DataNode is analogous to actual file storage blocks.
- **Hadoop YARN**: A framework for job scheduling and cluster resource management.
- **Hadoop MapReduce**: A YARN-based system for parallel processing of large data sets.

Update Configuration Files

Add/update core-site.xml on **Master and Slave nodes** with following options. Master and slave nodes should all be using the same value for this property **fs.defaultFS,** and should be pointing to master node only.

/home/hadoopuser/hadoop/etc/hadoop/core-site.xml(Other Options)

```
<property>
<name>hadoop.tmp.dir</name>
<value>/home/hadoopuser/tmp</value>
<description>Temporary Directory.</description>
</property>

<property>
<name>fs.defaultFS</name>
<value>hdfs://master:54310</value>
<description>Use HDFS as file storage engine</description></property>
```

Add/update mapred-site.xml on **Master node only** with following options.

/home/hadoopuser/hadoop/etc/hadoop/mapred-site.xml(Other Options)

```
<property>
<name>mapreduce.jobtracker.address</name>
<value>master: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>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
<description>The framework for running mapreduce jobs</description></property>
```

Add/update hdfs-site.xml on **Master and Slave Nodes.** We will be adding following three entries to the file.

- **dfs.replication**– Here I am using a replication factor of 2. That means for every file stored in HDFS, there will be one redundant replication of that file on some other node in the cluster.
- **dfs.namenode.name.dir –** This directory is used by Namenode to store its metadata file. Here i manually created this directory /hadoop-data/hadoopuser/hdfs/namenode on master and slave node, and use the directory location for this configuration.
- **dfs.datanode.data.dir –** This directory is used by Datanode to store hdfs data blocks. Herei manually created this directory /hadoop-data/hadoopuser/hdfs/datanode on master and slave node, and use the directory location for this configuration.

/home/hadoopuser/hadoop/etc/hadoop/hdfs-site.xml(Other Options)

```
<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>
<property>
<name>dfs.namenode.name.dir</name>
<value>/hadoop-data/hadoopuser/hdfs/namenode</value>
<description>Determines where on the local filesystem the DFS name node should store the name table(fsimage). If this is a comma-delimited list of directories then the name table is replicated in all of the directories, for redundancy.
</description>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>/hadoop-data/hadoopuser/hdfs/datanode</value>
<description>Determines where on the local filesystem an DFS data node should store its blocks. If this is a comma-delimited list of directories, then data will be stored in all named directories, typically on different devices. Directories that do not exist are ignored.  </description>
</property>
```

Add yarn-site.xml on **Master and Slave Nodes**. This file is required for a Node to work as a Yarn Node. Master and slave nodes should all be using the same value for the following properties, and should be pointing to master node only.

/home/hadoopuser/hadoop/etc/hadoop/yarn-site.xml

```
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.resourcemanager.scheduler.address</name>
<value>master:8030</value>
</property>
<property>
<name>yarn.resourcemanager.address</name>
<value>master:8032</value>
</property>
<property>
<name>yarn.resourcemanager.webapp.address</name>
<value>master:8088</value>
</property>
<property>
<name>yarn.resourcemanager.resource-tracker.address</name>
<value>master:8031</value>
</property>
<property>
<name>yarn.resourcemanager.admin.address</name>
<value>master:8033</value>
</property>
```

Add/update **slaves** file on Master node only. Add just name, or ip addresses of master and all slave node. If file has an entry for localhost, you can remove that. This file is just helper file that are used by hadoop scripts to start appropriate services on master and slave nodes.

/home/hadoopuser/hadoop/etc/hadoop/slave

```
master slave-1
slave-2
```

Format the Namenode

Before starting the cluster, we need to format the Namenode. Use the following command only on **master node**:

```
$ hdfsnamenode –format
```

Start the Distributed Format System

Run the following on **master node** command to start the DFS.

```
$ ./home/hadoopuser/hadoop/sbin/start-dfs.sh
```

You should observe the output to ascertain that it tries to start datanode on slave nodes one by one. To validate the success, run following command on master nodes, and slave node.

```
$ su - hadoopuser
$ jps
```

The output of this command should list ***NameNode,SecondaryNameNode, DataNode*** on **master** node, and ***DataNode***on all slave nodes. If you don't see the expected output, review the log files listed in Troubleshooting section.

Start the Yarn MapReduce Job tracker
Run the following command to start the Yarn mapreduce framework.

```
$ ./home/hadoopuser/hadoop/sbin/start-yarn.sh
```

To validate the success, run **jps** command again on master nodes, and slave node.The output of this command should list ***NodeManager,ResourceManager*** on **master** node, and ***NodeManager,***on all **slave** nodes. If you don't see the expected output, review the log files listed in Troubleshooting section.

Review Yarn Web console
If all the services started successfully on all nodes, then you should see all of your nodes listed under Yarn nodes. You can hit the following url on your browser and verify that:http://master:8088/cluster/nodes

Lets's execute a MapReduce example now
You should be all set to run a MapReduce example now. Run the following command

```
$ hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.2.0.jar pi 30 100
```

Once the job is submitted you can validate that its running on the cluster by accessing following url. http://master:8088/cluster/apps

Conclusion: Thus we have tested successfully multimode cluster.