IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-4, April 2019 E-ISSN: 2347-2693

Apache Hadoop: A Guide for Cluster Configuration & Testing

Ankit Shah[1*], Mamta Padole[2]

[1] Dept. of Information Technology, Shankersinh Vaghela Bapu Institute of Technology, Gandhinagar, India
[2] Dept. of Computer Science and Engineering, The Maharaja Sayajirao University of Baroda, Vadodara, India

[*]*Corresponding Author: shah_ankit101@yahoo.co.in , Tel.: +91-9824584855*

DOI: https://doi.org/10.26438/ijcse/v7i4.792796 | Available online at: www.ijcseonline.org

Accepted: 19/Apr/2019, Published: 30/Apr/2019

***Abstract*—** For Big Data processing, analyzing and storing Apache Hadoop is widely adopted as a framework. Hadoop facilitates processing through MapReduce, analyzing using Apache Spark and storage using the Hadoop Distributed File System (HDFS). Hadoop is popular due to its wide applicability and easy to run on commodity hardware functionality. But the installation of Hadoop on single and distributed cluster always remains a headache for the new developers and researchers. In this paper, we present the step by step process to run Hadoop on a single node and also explain how it can be used as a distributed cluster. We have implemented and tested the Hadoop framework using single node and cluster using ten (10) nodes. We have also explained primary keywords to understand the concept of Hadoop.

***Keywords*—**Apache Hadoop, Hadoop Cluster Configuration, Hadoop Testing, Hadoop Implementation

I. INTRODUCTION

Every day and year pass we are generating data. The data which is generated, it is not just a data but the data which is beyond our expectation and imagination. Knowingly or unknowingly we are part of these Big Data. More data doesn't just let us see more, it allows us to see new, better and different. The data has gone from a stock to a flow, from stationary and static to fluid and dynamic. So Big Data processing is not just a challenge but it also opens a new door for technology and betterment of humanity.

The Big Data is an all-encompassing term for any collection of data sets so large and complex that it becomes difficult to process using on-hand data management tools or traditional data processing applications [1]. Apache Hadoop [2] is the most suitable open source ecosystem of distributed processing of Big Data. Google's MapReduce [3] is the best-proposed programming framework for Big Data processing solution under the umbrella of Hadoop. Hadoop is not just software but it is a framework of tools for processing and analyzing Big Data.

Applications involving Big Data need enormous memory space to load the data and high processing power to execute them. Individually, the traditional computing systems are not sufficient to execute these big data applications but, cumulatively they can be used to meet the needs. This cumulative power for processing Big Data Applications can be achieved by using Distributed Systems with Map-Reduce model under the Apache Hadoop framework. Mere implementation of the application on Distributed Systems may not make optimal use of available resources [4].

In this paper, we try to provide complete configuration information so that any layman can just follow the steps of configuration for the Hadoop cluster setup.

Rest of the paper is organized as follows, Section I contains the introduction of Big Data and the need for Hadoop, Section II describes the Hadoop Ecosystem, Section III gives configuration settings of Hadoop single node setup, Section IV contains the essential steps of the Hadoop cluster setup, and Section V concludes research work with future directions.

II. HADOOP ECOSYSTEM

Hadoop is open source software comprising of framework of tools. These tools provide support for executing big data applications. Hadoop has very simple architecture. Hadoop 2.0 version primarily consists of three components as shown in fig.1:

1. HDFS (Hadoop Distributed File System) [5]: HDFS supports distributed storage on cluster nodes. HDFS is the core component of Hadoop which deals for data storage and block placement.
2. YARN (Yet Another Resource Negotiator) [6]: YARN is basically a resource manager. The task of

YARN is to manage the resources of the cluster and schedule the jobs.

3. MapReduce [3]: MapReduce the programming framework to execute parallel tasks on cluster nodes.

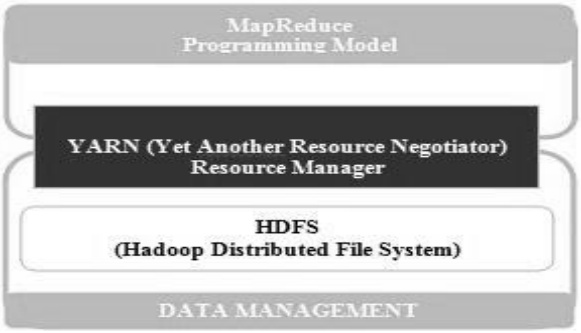

Figure 1. Hadoop Core Components

III. HADOOP SINGLE NODE SETUP

For Hadoop setup we use Hadoop 2.7.2 and ubuntu 14.04 version.

Required Software

1. JavaTM 1.5 + versions, preferably from Sun, must be installed

2. ssh must be installed and sshd must be running to use the Hadoop scripts that manage remote Hadoop daemons. (By default part of Linux system)

3. Hadoop: http://hadoop.apache.org/releases.html (version 2.7.2 (binary) – Size 202 MB)

Steps for Installation [For version: Jdk- 1.7, Hadoop-2.7.2]

1. Install Linux/Ubuntu (If using Windows, use Vmware player and Ubuntu image)

2. First, update the package index & for that command is

sudo apt-get update
sudo apt-get install ssh
sudo gedit /etc/ssh/sshd_config

3. Install Java

Hadoop framework required the jave environment. You can check the version available to your system using below command.

$ java –version

Figure 2. Java version installed

If JAVA not installed you can install by following commands:

sudo apt-get install default-jdk
or
sudo apt-get install openjdk-7-jre

4. Hadoop Version 2.7.2 Download

For our configuration we have used 2.7.2. User can use the latest version. Most of the parameters are unchanged in newer version too.

$wget
http://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-2.7.2/hadoop-2.7.2.tar.gz

But the file size is 202 MB so it's preferable to download it using some downloader and copy the **hadoop-2.7.2-src.tar.gz** file in your home folder.

5. Extract the hadoop-2.7.2-src.tar.gz directory manually or using command:

tar -xvf hadoop-2.7.2.tar.gz

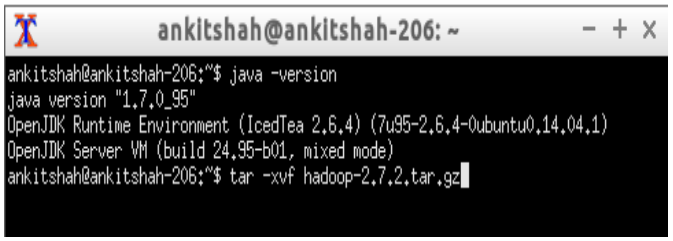

Figure 3. Extract Hadoop 2.7.2

This will create folder (directory) named: hadoop-2.7.2

6. Cross check hadoop directory

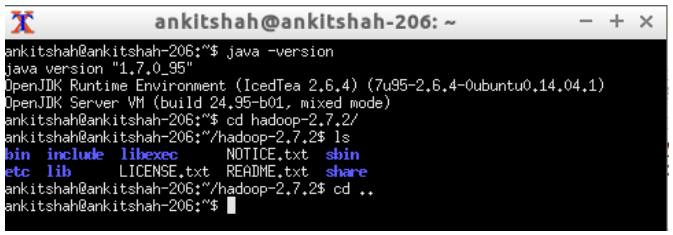

Figure 4. Hadoop 2.7.2 directory

7. Create Hadoop User for common access

It is important to create same username on **all machines** to avoid multiple password entries.

$sudo adduser hduser
$sudo adduser hduser sudo

8. Move Hadoop to usr/local directory

$sudo mv hadoop-2.7.2 /usr/local/

9. Move Hadoop to usr/local directory

sudo gedit /etc/hosts

10. Update the '.bashrc' file to add important Apache Hadoop environment variables for user.

a) Change directory to home.
$ cd
b) Edit the file
$ sudo gedit .bashrc (this command will open one file)

----Set Hadoop environment Variables - @ the end of File----

export HADOOP_HOME=/usr/local/hadoop-2.7.2
export HADOOP_CONF_DIR=/usr/local/hadoop-2.7.2/etc/hadoop
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-i386
export PATH=$PATH:/$HADOOP_HOME/bin
export PATH=$PATH:/$HADOOP_HOME/sbin
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"

c) Source the .bashrc file to set the hadoop environment variables without having to invoke a new shell:
$. ~/.bashrc

11. Setup the Hadoop Cluster

11.1 Configure JAVA_HOME

Configure JAVA_HOME in 'hadoop-env.sh'. This file specifies environment variables that affect the JDK used by Apache Hadoop 2.0 daemons started by the Hadoop start-up scripts:

$cd $HADOOP_CONF_DIR
$sudo gedit hadoop-env.sh

Update the JAVA_HOME to:

export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-i386

11.2 Configure the Default File system
The 'core-site.xml' file contains the configuration settings for Apache Hadoop Core such as I/O settings that are common to HDFS, YARN and MapReduce. Configure default files system

(Parameter: fs.default.name) used by clients in core-site.xml

$cd $HADOOP_CONF_DIR
$sudo gedit core-site.xml

Add the following line in between the configuration tag:

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

11.3 Configure MapReduce framework
This file contains the configuration settings for MapReduce. Configure mapred-site.xml and specify framework details.

$sudo gedit mapred-site.xml

Add the following line in between the configuration tag:

<configuration>
<property>
<name>mapred.job.tracker</name>
<value>localhost:54311</value>
</property>
</configuration>

11.4 Create NameNode and DataNode directory
Create DataNode and NameNode directories to store HDFS data.

$sudo mkdir -p /usr/local/hadoop_tmp/hdfs/namenode
$sudo mkdir -p /usr/local/hadoop_tmp/hdfs/datanode
$sudo chown -R hduser /usr/local/hadoop_tmp

11.5 Configure the HDFS
HDFS configuration helps to create number of replicas for the data. It also runs daemons in NameNode and DataNode for HDFS configurations. Here hdfs-site.xml is required to setup which is fully customizable. So we can have used replication factor as 1. Usually default is 3.

$sudo gedit hdfs-site.xml

Add the following line in between the configuration tag:

<property>

<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.namenode.name.dir</name>
<value>file:/usr/local/hadoop_tmp/hdfs/namenode
</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value> file:/usr/local/hadoop_tmp/hdfs/datanode
</value>
</property>

11.6 Start the DFS services
Once all parameters are set we are ready to go. But before that it is required to format the HDFS namenode for enabling to store and manage HDFS directories along with default OS file system. Command for formatting the HDFS namenode is give below:

To format the file-system, run the command:

$hdfs namenode –format
(To execute command: Go to Hadoop folder > Go to bin folder)
Or
$hadoop namenode –format (deprecated from latest Hadoop version)
$start-all.sh
http://localhost:50070/
$stop-all.sh

IV. HADOOP CLUSTER SETUP

```
$sudo gedit /etc/hosts
```

Add hostnames with IP address

```
$cd $HADOOP_CONF_DIR
$sudo gedit hdfs-site.xml
```

<property>
<name>dfs.replication</name>
<value>2</value>
</property>
<property>
<name>dfs.namenode.name.dir</name>
<value>file:/usr/local/hadoop_tmp/hdfs/namenode
</value>
</property>

```
$cd $HADOOP_CONF_DIR
$sudo gedit core-site.xml
```

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

```
$sudo gedit mapred-site.xml
```

<configuration>
<property>
<name>mapreduce.job.tracker</name>
<value>hadoopmaster:54311</value>
</property>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>

```
$sudo gedit yarn-site.xml
```

<property>
<name>yarn.resourcemanager.hostname</name>
<value>hadoopmaster</value>
<description>The hostname of the RM.</description>
</property>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
<description>shuffle service that needs to be set for Map Reduce to run </description>
</property>
<property>

<name>yarn.resourcemanager.scheduler.address
</name>
<value>hadoopmaster:8030</value>
</property>
<property>
<name>yarn.resourcemanager.address</name>
<value>hadoopmaster:8032</value>
</property>
<property>
<name>yarn.resourcemanager.webapp.address</name>
<value>hadoopmaster:8088</value>
</property>
<property>
<name>yarn.resourcemanager.resourcetracker.address
</name>
<value>hadoopmaster:8031</value>
</property>

```
$sudo gedit /usr/local/hadoop-2.7.2/etc/hadoop/slaves
```

Now create clone of Master

Master

```
$sudo gedit /usr/local/hadoop-2.7.2/etc/hadoop/masters
$sudo rm -r /usr/local/hadoop_tmp
$sudo mkdir /usr/local/hadoop_tmp
$sudo mkdir -p /usr/local/hadoop_tmp/hdfs/namenode
$sudo chown -R hduser /usr/local/hadoop_tmp
If required check for the chmod also
$sudo chmod 755 -R /usr/local/hadoop-2.7.2
$sudo chown -R hduser /usr/local/hadoop-2.7.2
$hdfs namenode -format
```

All Slaves

```
$cd $HADOOP_CONF_DIR
$sudo gedit hdfs-site.xml
```

```
<property>
<name>dfs.replication</name>
<value>2</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>file:/usr/local/hadoop_tmp/hdfs/datanode
</value>
</property>
```

```
$sudo rm -rf  /usr/local/hadoop_tmp
$sudo mkdir -p  /usr/local/hadoop_tmp
$sudo mkdir -p  /usr/local/hadoop_tmp/hdfs/datanode
$sudo chown -R hduser /usr/local/hadoop_tmp (change ownership)
$sudo chmod 755 -R /usr/local/hadoop-2.7.2 (check user access)
$sudo chown -R hduser /usr/local/hadoop-2.7.2
$sudo reboot
```

Master

```
$sudo /etc/init.d/networking restart
$ssh-keygen -t rsa -P ""
$cat
$HOME/.ssh/id_rsa.pub>>$HOME/.ssh/authorized_keys
$ssh-copy-id -i ~/.ssh/id_rsa.pub hduser@slave1 (do it for all slaves)
$ssh hadoopmaster
$ssh slave1 (should be able to login w/o password)
```

```
$start-all.sh
```

http://hadoopmaster:50070/

Figure 5. Hadoop 2.7.2 dashboard

V. CONCLUSION AND FUTURE SCOPE

In this paper, we give detailed steps to install Hadoop standalone and distributed cluster. We try to provide all setup parameters which are customized and the user can set it as per the convenience. Moreover, this complete guide will help to create a robust cluster setup for the homogeneous and heterogeneous cluster. The future researcher can avail the benefit of this setup configuration for their researches.

REFERENCES

[1] Forbes Welcome, https://www.forbes.com/sites/gilpress/2014/09/03/12-big-data-definitions-whats-yours/#487d104413ae (Access on March 30, 2019)

[2] Hadoop, http://hadoop.apache.org (Access on March 30, 2019)

[3] Dean, J. and Ghemawat, S., MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1), pp.107-113 (2008).

[4] Shah A., Padole M. (2019) Performance Analysis of Scheduling Algorithms in Apache Hadoop. In: Shukla R., Agrawal J., Sharma S., Singh Tomer G. (eds) Data, Engineering and Applications. Springer, Singapore

[5] Shvachko, K., Kuang, H., Radia, S. and Chansler, R., 2010, May. The hadoop distributed file system. In *MSST* (Vol. 10, pp. 1-10).

[6] Vavilapalli, V.K., Murthy, A.C., Douglas, C., Agarwal, S., Konar, M., Evans, R., Graves, T., Lowe, J., Shah, H., Seth, S. and Saha, B., (2013). Apache hadoop yarn: Yet another resource negotiator. In Proceedings of the 4th annual Symposium on Cloud Computing (p.5). ACM.

Authors Profile

Mr. Ankit Shah received his Bachelors in Information & Technology Engineering in 2009. He received his Masters in Computer Science & Engineering. Currently, he is pursuing his PhD in the Computer Science & Engineering from The Maharaja Sayajirao University of Baroda, India. His research interests include big data processing, distributed computing, and IoT. He has 8 years of teaching experience.

Dr. Mamta Padole is PhD in Computer Science & Engineering, and is currently working as an Associate Professor in the Department of Computer Science & Engineering, The Maharaja Sayajirao University of Baroda, India. She has vast experience in teaching and research. Her research interests include Distributed computing, Fog computing, IoT and BioInformatics. She has over 20 years of teaching / industry experience.

