Hadoop Installation

Assignment

S Meena Padnekar
Sharon Dcruz
Vishnu NS
Shirfil

2020

Table of Contents

Requirement

- Operating System: Atleast 2 machines pre-installed with Linux OS (Ubuntu).
  (Here we are configuring hadoop with 2 Ubuntu 18.04 systems(a master and a slave system)
- Java 8 package must be installed on all system.

Java 8 Installation in Ubuntu 18.04

- Run the following commands in the terminal
  - To install OpenJDK 8
    sudo apt install openjdk-8-jdk
  - Verify that this is installed with
    java -version
  - Managing Java versions
    sudo update-alternatives –config java
- For more details about installation check this website https://www.digitalocean.com/community/tutorials/how-to-install-java-with-apt-on-ubuntu-18-04

1.Add a Hadoop User

Commands to Add Hadoop User :

- sudo addgroup hadoop_
- sudo adduser –ingroup hadoop_ hduser_
- sudo adduser hduser_ sudo

Login as hadoop User

- su - hduser_

2. SSH Passwordless Login

1. **Generate a pair of public keys on Master machine.**
   - ssh-keygen -t rsa -P ”

```
exam@cusat-HP-dx7380-MT-KF137PA:~$ ssh-keygen -t rsa -P ''
Generating public/private rsa key pair.
Enter file in which to save the key (/home/exam/.ssh/id_rsa):
Your identification has been saved in /home/exam/.ssh/id_rsa.
Your public key has been saved in /home/exam/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:sOwLkUPgKPWk/bPg3MGA/T/spIYXB0mpxrXn2ad8MkI exam@cusat-HP-dx7380-MT-KF137PA
The key's randomart image is:
+---[RSA 2048]----+
|  o .  .         |
| + O  +          |
|o +.*+.o         |
| .  .+B+o.       |
|   .= O+So       |
|   o *.*E . .    |
|    +.++=. o     |
|    ..o=..= .    |
|     oo .. +     |
+----[SHA256]-----+
```

Figure 1

2. SSH Passwordless Login

Run the following commands On Slave Systems

- **Create .ssh directory on slave systems**
  - ssh hduser_@ip mkdir -p .ssh
- **Upload Generated Public Keys to slaves**
  - cat .ssh/id_rsa.pub | ssh hduser_@ip ’cat ».ssh/authorized_keys’
- **Set Permissions on Slave**
  - ssh hduser_@ip "chmod 700 .ssh; chmod 640 .ssh/authorized_keys"

3. Add host names to files

Do the following Procedure

- **Open file /etc/hosts**
  - sudo vi /etc/hosts
- **Add these lines**
  # hadoop
  172.16.6.14 hadoop-master
  172.16.5.153 hadoop-slave1
  172.16.6.115 hadoop-slave2
- **Comment the line**
  127.0.0.1 localhost

To ssh to other system

- ssh hadoop-slave1

Download Hadoop

Link : http://archive.apache.org/dist/hadoop/common/hadoop-3.1.2/hadoop-3.1.2-src.tar.gz

Run the commands

- sudo tar xzf hadoop-3.1.2.tar.gz
- sudo mv hadoop-3.1.2 hadoop
- sudo chown -R hduser_:hadoop_ hadoop

Configure Hadoop

- Modify /.bashrc by Adding the lines
  export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
  export HADOOP_HOME=/usr/local/hadoop
  export PATH=$PATH:$HADOOP_HOME/bin
  export PATH=$PATH:$HADOOP_HOME/sbin
  export HADOOP_MAPRED_HOME=$HADOOP_HOME
  export HADOOP_COMMON_HOME=$HADOOP_HOME
  export HADOOP_HDFS_HOME=$HADOOP_HOME
  export YARN_HOME=$HADOOP_HOME
  export HADOOP_COMMON_LIB_NATIVE_DIR=
  $HADOOP_HOME/lib/native
  export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
- **source this environment configuration using below command**
  - . /.bashrc

Edit hadoop-env.sh

- vi /usr/local/hadoop/etc/hadoop/hadoop-env.sh
- Add JAVA_HOME path
  export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/
- And Save

Edit core-site.xml

- vi /usr/local/hadoop/etc/hadoop/core-site.xml
- Add Property
  ’fs.default.name’ - This specifies the default file system.

```
<property>
    <name>fs.default.name</name>
    <value>hdfs://hadoop-master:9000</value>
</property>
```

Edit hdfs-site.xml

- vi /usr/local/hadoop/etc/hadoop/hdfs-site.xml
- Add Properties
  - dfs.replication: indicates how many times data is replicated in the cluster

    <name>dfs.replication</name>

    <value>2</value>
  - dfs.permissions : set permission to false

    <name>dfs.permissions</name>

    <value>false</value>
  - dfs.namenode.name.dir : Specify namenode directory

    <name>dfs.namenode.name.dir</name>

    <value>/usr/local/hadoop/hdfs/namenode</value>
  - dfs.datanode.name.dir : Specify datanode directory

    <name>dfs.datanode.name.dir</name>

    <value>/usr/local/hadoop/hdfs/datanode</value>

Edit yarn-site.xml

- vi /usr/local/hadoop/etc/hadoop/yarn-site.xml
- Add properties
  1. name>yarn.nodemanager.aux-services</name>
     <value>mapreduce_shuffle</value>
  2. <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
     <value>org.apache.hadoop.mapred.ShuffleHandler</value>
  3. <name>yarn.resourcemanager.resource-tracker.address</name>
     <value>hadoop-master:8025</value>
  4. <name>yarn.resourcemanager.scheduler.address</name>
     <value>hadoop-master:8030</value>
  5. <name>yarn.resourcemanager.address</name>
     <value>hadoop-master:8040</value>

Edit mapred-site.xml

- vi /usr/local/hadoop/etc/hadoop/mapred-site.xml
- Add Property

```
<property>
    <name>mapred.job.tracker</name>
    <value>hadoop-master:54311</value>
</property>
```

Configure Master and Slave

- Add Master in file /usr/local/hadoop/etc/hadoop/masters
  hadoop-master
- Add Slave system in file /usr/local/hadoop/etc/hadoop/workers
  hadoop-slave1
  hadoop-slave2

Duplicate Config Files on Each Node

1. Copy the Hadoop binaries to worker nodes:
   ```
   tar -zcvf hadoop.tar.gz /usr/local/hadoop/
   scp hadoop.tar.gz hadoop-slave1:/home/hadoop
   scp hadoop.tar.gz hadoop-slave2:/home/hadoop
   ```
2. Connect to hadoop-slave1 via SSH
   ```
   ssh hadoop-slave1
   ```
3. Unzip the binaries, rename the directory, and exit node1 to get back on the node-master:
   ```
   tar -xzf hadoop.tar.gz
   mv hadoop /usr/local/
   logout
   ```
4. Repeat steps 2 and 3 for all slave machines.

Run and monitor HDFS

- Start and Stop HDFS
  - start-dfs.sh
  - stop-dfs.sh
- Start and Stop Yarn
  - start-yarn.sh
  - stop-yarn.sh
- Monitor your HDFS Cluster
  - hdfs dfsadmin -report
- Monitor YARN
  - yarn node -list
  - yarn application -list
- Format HDFS
  - hdfs namenode -format
- Point your browser to **http://hadoop-master:9870**

Master Configuration

Hadoop | Overview | Datanodes | Datanode Volume Failures | Snapshot | Startup Progress | Utilities

Overview 'hadoop-master:9000' (active)

Started:	Tue Mar 10 09:42:27 +0530 2020
Version:	3.2.1, rb3cbbb467e22ea829b3808f4b7b01d07e0bf3842
Compiled:	Tue Sep 10 21:26:00 +0530 2019 by rohithsharmaks from branch-3.2.1
Cluster ID:	CID-b4b38b89-3a6a-4bcd-a3e5-73db4393f6c3
Block Pool ID:	BP-1173495593-172.16.7.53-1580896540152

Summary

Security is off.

Safemode is off.

35 files and directories, 13 blocks (13 replicated blocks, 0 erasure coded block groups) = 48 total filesystem object(s).

Heap Memory used 60.76 MB of 175.5 MB Heap Memory. Max Heap Memory is 832 MB.

Non Heap Memory used 57.49 MB of 58.75 MB Commited Non Heap Memory. Max Non Heap Memory is <unbounded>.

Figure 2

Datanode Configuration

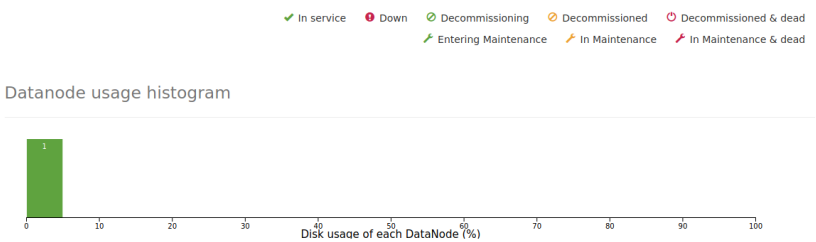

Figure 3

Hadoop File System

Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities

Browse Directory

/ Go!

Show 25 entries Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	dr.who	supergroup	0 B	Mar 10 10:28	0	0 B	MM
drwxr-xr-x	hduser_	supergroup	0 B	Feb 27 16:46	0	0 B	Matrix
drwxr-xr-x	hduser_	supergroup	0 B	Feb 27 14:50	0	0 B	Product
drwxr-xr-x	hduser_	supergroup	0 B	Mar 10 10:29	0	0 B	Res
drwxr-xr-x	hduser_	supergroup	0 B	Feb 05 15:45	0	0 B	hduser_
drwxr-xr-x	hduser_	supergroup	0 B	Feb 27 15:51	0	0 B	output
drwxr-xr-x	hduser_	supergroup	0 B	Feb 28 11:07	0	0 B	result

Figure 4

Matrix Multiplication

- Matrix multiplication or the matrix product is a binary operation that produces a matrix from two matrices.
- If A is an $n \times m$ matrix and B is an $m \times p$ matrix, their matrix product AB is an $n \times p$ matrix, in which the m entries across a row of A are multiplied with the m entries down a column of B and summed to produce an entry of AB.

Matrix Multiplication Map Function

Algorithm for Map Function

1. for each element m_{ij} of M do
   produce (key,value) pairs as ((i,k), (M,j,m_{ij}), for k=1,2,3,.. up to the number of columns of N
2. for each element n_{jk} of N do
   produce (key,value) pairs as ((i,k),(N,j,n_{jk}), for i = 1,2,3,.. Upto the number of rows of M.
3. return Set of (key,value) pairs that each key (i,k), has list with values (M,j,m_{ij}) and (N, j,n_{jk}) for all possible values of j.

Matrix Multiplication Reduce Function

Algorithm for Reduce Function

1. for each key (i,k) do
    - sort values begin with M by j in listM
    - sort values begin with N by j in listN
    - multiply m_{ij} and n_{jk} for j^{th} value of each list
    - sum up m_{ij} x n_{jk} return (i,k), $\sum_{j=1} m_{ij} \times n_{ij}$

Matrix Multiplication Program

- Download the program from [3]
- Compile the java programs:
  - **javac -d . Map.java Reduce.java MatrixMultiply.java**
- Set CLASSPATH for java program
  - **export CLASSPATH="$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-core-3.2.1.jar:$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-common-3.2.1.jar:$HADOOP_HOME/share/hadoop/common/hadoop-common-3.2.1.jar: /matrixMult/MatrixMultiply/*:$HADOOP_HOME/lib/*"**
- Create Jar file
  - **jar -cvf MatrixMultiply.jar -C operation/ .**

Matrix Multiplication Program

- Create a file in HDFS:
  hdfs dfs -mkdir /Matrix
- Create M and N matrix file:
  cat M

  M,0,0,1
  M,0,1,2
  M,1,0,3
  M,1,1,4
- copy M and N matrix file to this file
  hdfs dfs -put M /Matrix/
  hdfs dfs -put N /Matrix/

Matrix Multiplication Program

- Run the jar file:
  hadoop jar MatrixMultiply.jar www.ehadoopinfo.com.MatrixMultiply /Matrix/* /result/
- Find the Result in file result/part-r-00000

M matrix

Figure 5

N Matrix

Figure 6

Resultant Matrix

Figure 7

Reference

- How to Install and Set Up a 3-Node Hadoop Cluster
  https://www.linode.com/docs/databases/hadoop/how-to-install-and-set-up-hadoop-cluster/
- Hadoop Mapreduce Examples: Create your First Program
  https://www.guru99.com/create-your-first-hadoop-program.html
- MR5. Matrix Multiplication using MapReduce Programming in Java.
  http://www.ehadoopinfo.com/2017/10/mr5-matrix-multiplication-using.html

Thanks