Indira Gandhi Delhi Technical University for Women

(Established by Govt. of Delhi vide Act 09 of 2012) Kashmere Gate, Delhi – 110006

PRACTICAL FILE

BIG DATA AND NOSQL

MCA – 203

Submitted By:
Shweta Rawat
Roll No. 06304092023
MCA (IT), Semester - 3

Submitted To:
Dr. Praveen

INDEX

PRACTICAL 1

Objective: Download and Configure Hadoop for Colab.

Theory: Apache Hadoop is an open-source platform designed for the distributed storage and processing of vast datasets across computer clusters. It effectively handles large-scale data with its key components:

1. **HDFS (Hadoop Distributed File System):** HDFS ensures reliable storage by distributing and replicating large files across multiple nodes, providing high-speed access tailored for big data storage and retrieval.
2. **MapReduce:** This processing model enables parallel computation by breaking tasks into smaller jobs that run concurrently. This method boosts efficiency and shortens overall processing time.
3. **YARN (Yet Another Resource Negotiator):** YARN manages cluster resources and schedules tasks, improving scalability and optimizing resource usage.
4. **Hadoop Ecosystem:** Hadoop integrates with tools like Apache Hive for SQL-style queries, Apache Pig for data transformation, Apache HBase for NoSQL storage, and Apache Spark for real-time in-memory processing.

Hadoop is pivotal in Big Data analytics, allowing organizations to store and process enormous datasets efficiently and affordably. Its design emphasizes fault tolerance and horizontal scalability, making it well-suited for complex analyses in distributed environments.

1. Download JAVA.

Download JAVA

```
[1]  !apt-get install openjdk-11-jdk-headless -qq > /dev/null
```

2. Make sure that no partial downloads or corrupted files interfere with the process are left over from previous installation attempts.

Make sure that no partial downloads or corrupted files interfere with the process are left over from previous installation attempts.

```
[2]  !rm -rf hadoop* /usr/local/hadoop
```

3. Download HADOOP

Download HADOOP

```
!wget -c "https://archive.apache.org/dist/hadoop/common/hadoop-3.3.2/hadoop-3.3.2.tar.gz" -O hadoop.tar.gz
!tar -xzf hadoop.tar.gz
!mv hadoop-3.3.2 /usr/local/hadoop
```

```
--2024-11-22 08:22:14--  https://archive.apache.org/dist/hadoop/common/hadoop-3.3.2/hadoop-3.3.2.tar.gz
Resolving archive.apache.org (archive.apache.org)... 65.108.204.189, 2a01:4f9:1a:a084::2
Connecting to archive.apache.org (archive.apache.org)|65.108.204.189|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 638660563 (609M) [application/x-gzip]
Saving to: 'hadoop.tar.gz'

hadoop.tar.gz       100%[===================>] 609.07M  15.7MB/s    in 60s

2024-11-22 08:23:14 (10.2 MB/s) - 'hadoop.tar.gz' saved [638660563/638660563]
```

4. Verify the downloaded file

Verify the downloaded file.

```
!ls -lh /usr/local/hadoop
```

```
total 112K
drwxr-xr-x 2 501 dialout 4.0K Feb 21  2022 bin
drwxr-xr-x 3 501 dialout 4.0K Feb 21  2022 etc
drwxr-xr-x 2 501 dialout 4.0K Feb 21  2022 include
drwxr-xr-x 3 501 dialout 4.0K Feb 21  2022 lib
drwxr-xr-x 4 501 dialout 4.0K Feb 21  2022 libexec
-rw-r--r-- 1 501 dialout  23K Jan 15  2022 LICENSE-binary
drwxr-xr-x 2 501 dialout 4.0K Feb 21  2022 licenses-binary
-rw-r--r-- 1 501 dialout  15K Jan 15  2022 LICENSE.txt
-rw-r--r-- 1 501 dialout  29K Jan 31  2022 NOTICE-binary
-rw-r--r-- 1 501 dialout 1.6K Dec  2  2021 NOTICE.txt
-rw-r--r-- 1 501 dialout  175 Dec  2  2021 README.txt
drwxr-xr-x 3 501 dialout 4.0K Feb 21  2022 sbin
drwxr-xr-x 4 501 dialout 4.0K Feb 21  2022 share
```

5. Setup environment variable for JAVA for the COLAB session.

Setup environment variable for JAVA for the COLAB session.

```
[5] import os
    os.environ["JAVA_HOME"]="/usr/lib/jvm/java-11-openjdk-amd64"
    os.environ["HADOOP_HOME"]="/usr/local/hadoop"
    os.environ["PATH"] += os.pathsep + "/usr/local/hadoop/bin"
```

```
!echo "export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64" >> /usr/local/hadoop/etc/hadoop/hadoop-env.sh
```

6. Extract the Hadoop tarball.

Extract the Hadoop tarball

[No Title]

```
[7]  !tar -xzf hadoop.tar.gz
     !mv hadoop-3.3.2 /usr/local/hadoop
```

7. Create a Hadoop user and adjust permissions accordingly.

Create a hadoop user and adjust permissions accordingly.

```
[8]  !adduser --disabled-password --gecos "" hadoopuser
```

```
Adding user `hadoopuser' ...
Adding new group `hadoopuser' (1000) ...
Adding new user `hadoopuser' (1000) with group `hadoopuser' ...
Creating home directory `/home/hadoopuser' ...
Copying files from `/etc/skel' ...
```

8. Set Permissions for Hadoop: Change the ownership of the Hadoop directory to the new user.

```
[9]  !chown -R hadoopuser:hadoopuser /usr/local/hadoop
```

9. Set environment variables.

```
os.environ['HADOOP_HOME']='/usr/local/hadoop'
os.environ['PATH'] += os.pathsep + '/usr/local/hadoop/bin'
os.environ['PATH'] += os.pathsep + '/usr/local/hadoop/sbin'
os.environ['HDFS_NAMENODE_USER'] = 'hadoopuser'
os.environ['HDFS_DATANODE_USER'] = 'hadoopuser'
os.environ['HDFS_SECONDARYNAMENODE_USER'] = 'hadoopuser'
os.environ['YARN_RESOURCEMANAGER_USER'] = 'hadoopuser'
os.environ['YARN_NODEMANAGER_USER'] = 'hadoopuser'
```

10. Configure now Hadoop

```
[10] !sed -i '/<configuration>/a <property>\n <name>fs.defaultFS</name>\n <value>hdfs://localhost:9000</value>\n</property>' /usr/local/hadoop/etc/hadoop/core-site.xml
```

```
[11] !sed -i '/<configuration>/a <property>\n <name>dfs.replication</name>\n <value>1</value>\n</property>' /usr/local/hadoop/etc/hadoop/hdfs-site.xml
```

11. Format the HDFS to prepare it for use

```
[12] !sudo -u hadoopuser /usr/local/hadoop/bin/hdfs namenode -format

WARNING: /usr/local/hadoop/logs does not exist. Creating.
2024-11-22 08:35:57,219 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = 7d3bc1e77c66/172.28.0.12
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 3.3.2
STARTUP_MSG:   classpath = /usr/local/hadoop/etc/hadoop:/usr/local/hadoop/share/hadoop/common/lib/kerby-pkix-1.0.1.jar:/usr/local/hadoop/share/hadoop/
STARTUP_MSG:   build = git@github.com:apache/hadoop.git -r 0bcb014209e219273cb6fd4152df7df713cbac61; compiled by 'chao' on 2022-02-21T18:39Z
STARTUP_MSG:   java = 11.0.25
************************************************************/
2024-11-22 08:35:57,320 INFO namenode.NameNode: registered UNIX signal handlers for [TERM, HUP, INT]
2024-11-22 08:35:57,667 INFO namenode.NameNode: createNameNode [-format]
2024-11-22 08:35:59,285 INFO namenode.NameNode: Formatting using clusterid: CID-225cd62a-13e4-4a2b-8094-f99403b9a7f6
2024-11-22 08:35:59,347 INFO namenode.FSEditLog: Edit logging is async:true
2024-11-22 08:35:59,472 INFO namenode.FSNamesystem: KeyProvider: null
2024-11-22 08:35:59,476 INFO namenode.FSNamesystem: fsLock is fair: true
2024-11-22 08:35:59,477 INFO namenode.FSNamesystem: Detailed lock hold time metrics enabled: false
2024-11-22 08:35:59,552 INFO namenode.FSNamesystem: fsOwner                 = hadoopuser (auth:SIMPLE)
2024-11-22 08:35:59,552 INFO namenode.FSNamesystem: supergroup              = supergroup
```

12. Start services now

```
[13] !sudo -u hadoopuser /usr/local/hadoop/bin/hdfs --daemon start namenode
     !sudo -u hadoopuser /usr/local/hadoop/bin/hdfs --daemon start datanode
     !sudo -u hadoopuser /usr/local/hadoop/bin/hdfs --daemon start secondarynamenode
     !sudo -u hadoopuser /usr/local/hadoop/bin/yarn --daemon start resourcemanager
     !sudo -u hadoopuser /usr/local/hadoop/bin/yarn --daemon start nodemanager
```

13. Check whether all services started or not.

```
[14] !jps

6337 Jps
6145 ResourceManager
6241 NodeManager
6006 DataNode
5943 NameNode
6077 SecondaryNameNode
```

PRACTICAL 2

Objective: To perform some basic HDFS operations like creating new files in new directories.

Theory: The Hadoop Distributed File System (HDFS) provides several command-line operations to manage files within a Hadoop cluster. Below are some essential HDFS commands for working with files and directories:

1. **Viewing Files and Directories**
   - **Command:** hdfs dfs -ls /path
   - This command lists all files and directories in the specified HDFS path, showing details such as file permissions, ownership, size, and last modified date.
2. **Creating a Directory**
   - **Command:** hdfs dfs -mkdir /path/directory_name
   - Use this command to create a new directory in HDFS at the given path, helping to keep files organized.
3. **Transferring Files to HDFS**
   - **Command:** hdfs dfs -put /local_path/filename /hdfs_path
   - The -put option uploads files from the local file system to HDFS, making them available for distributed storage and processing.
4. **Retrieving Files from HDFS**
   - **Command:** hdfs dfs -get /hdfs_path/filename /local_path
   - The -get command is used to download files from HDFS to the local file system, enabling local access for further analysis or use.

Code and Output:

1. Create directories in HDFS

```
[16] !sudo -u hadoopuser /usr/local/hadoop/bin/hdfs dfs -mkdir -p /user/hadoopuser
     !sudo -u hadoopuser /usr/local/hadoop/bin/hdfs dfs -mkdir -p /user/hadoopuser1
     !sudo -u hadoopuser /usr/local/hadoop/bin/hdfs dfs -mkdir -p /user/hadoopuser_hadoopuser1
```

2. Check if the directories have been created or not.

```
[17] !sudo -u hadoopuser /usr/local/hadoop/bin/hdfs dfs -ls /user/

Found 3 items
drwxr-xr-x   - hadoopuser supergroup          0 2024-11-22 08:51 /user/hadoopuser
drwxr-xr-x   - hadoopuser supergroup          0 2024-11-22 08:51 /user/hadoopuser1
drwxr-xr-x   - hadoopuser supergroup          0 2024-11-22 08:51 /user/hadoopuser_hadoopuser1
```

You should see all the directories you created in the previous step.

3. Create a file hadoop.txt with content “This is my first Hadoop file” and upload it in Hadoop.

```
[18] !echo "This is my first Hadoop file"> hadoop.txt
     !sudo -u hadoopuser /usr/local/hadoop/bin/hdfs dfs -put hadoop.txt /user/hadoopuser/
```

4. Check files have been created and uploaded.

```
[20] !sudo -u hadoopuser /usr/local/hadoop/bin/hdfs dfs -ls /user/hadoopuser/

Found 1 items
-rw-r--r--   1 hadoopuser supergroup         29 2024-11-22 08:52 /user/hadoopuser/hadoop.txt
```

5. Create more than one file and upload.

```
[21] !echo "This is file 1"> My_file1.txt
     !echo "This is file 2"> My_file2.txt
     !sudo -u hadoopuser /usr/local/hadoop/bin/hdfs dfs -put My_file1.txt My_file2.txt /user/hadoopuser/
```

6. Check if both the files have been successfully uploaded in hadoop HDFS or not.

```
[22] !sudo -u hadoopuser /usr/local/hadoop/bin/hdfs dfs -ls /user/hadoopuser/

Found 3 items
-rw-r--r--   1 hadoopuser supergroup         15 2024-11-22 08:52 /user/hadoopuser/My_file1.txt
-rw-r--r--   1 hadoopuser supergroup         15 2024-11-22 08:52 /user/hadoopuser/My_file2.txt
-rw-r--r--   1 hadoopuser supergroup         29 2024-11-22 08:52 /user/hadoopuser/hadoop.txt
```

PRACTICAL 3

Objective: To perform basic YARN operations.

Theory: YARN (Yet Another Resource Negotiator) is a key component of the Apache Hadoop ecosystem that handles resource management and job scheduling for applications running on a Hadoop cluster. Introduced in Hadoop 2.0, YARN addressed the limitations of the original MapReduce system by separating resource management from data processing.

This separation enables multiple types of data processing applications to run simultaneously on the same cluster.

Key Components of YARN

1. ResourceManager (RM): The central authority for managing cluster resources, with two main subcomponents:

- Scheduler: Allocates resources to applications based on policies like capacity and fairness. It does not monitor or restart failed tasks.
- Application Manager: Handles job lifecycles, accepts job submissions, and collaborates with the Scheduler to allocate containers for running applications.

2. NodeManager (NM): A daemon running on each cluster node, responsible for container management, resource monitoring (CPU, memory, disk), and reporting resource usage to the ResourceManager.

3. ApplicationMaster (AM): A framework-specific library that requests resources from the ResourceManager and works with the NodeManager to execute and track tasks.

4. Containers: Containers represent physical resources (e.g., CPU, memory) allocated to tasks on a node. Each application operates within its container to ensure efficient resource utilization.

Basic YARN Operations

1. Listing Running Applications:

- Command: yarn application -list
- Displays all currently running applications, including details like application ID, name, user, type, and status (RUNNING, ACCEPTED, etc.).

2. Checking Cluster Status:

- Command: yarn node -list
- Lists all nodes in the cluster by their status (RUNNING, LOST, DECOMMISSIONED, etc.), aiding in cluster health monitoring.

3. Viewing Application Logs:

- Command: yarn logs -applicationId <application_id>
- Retrieves logs for a specific application to assist in debugging and reviewing execution details.

Code and Output:

1. Check the status of YARN

```
[23] !yarn application -list

2024-11-22 09:05:56,585 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
Total number of applications (application-types: [], states: [SUBMITTED, ACCEPTED, RUNNING] and tags: []):0
                Application-Id      Application-Name        Application-Type          User           Queue                   State
```

2. Check with queue resources.

```
[24] !yarn queue -status default

2024-11-22 09:06:08,013 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
Queue Information :
Queue Name : default
        State : RUNNING
        Capacity : 100.00%
        Current Capacity : .00%
        Maximum Capacity : 100.00%
        Default Node Label expression : <DEFAULT_PARTITION>
        Accessible Node Labels : *
        Preemption : disabled
        Intra-queue Preemption : disabled
```

3. Command to track jobs:

```
!yarn application -list -appStates RUNNING
!yarn application -list -appStates FINISHED
!yarn application -list -appStates ALL

2024-11-22 09:06:20,564 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
Total number of applications (application-types: [], states: [RUNNING] and tags: []):0
                Application-Id      Application-Name        Application-Type          User           Queue                   State             Final-State             Progress
2024-11-22 09:06:22,643 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
Total number of applications (application-types: [], states: [FINISHED] and tags: []):0
                Application-Id      Application-Name        Application-Type          User           Queue                   State             Final-State             Progress
2024-11-22 09:06:24,714 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
Total number of applications (application-types: [], states: [NEW, NEW_SAVING, SUBMITTED, ACCEPTED, RUNNING, FINISHED, FAILED, KILLED] and tags: []):0
                Application-Id      Application-Name        Application-Type          User           Queue                   State             Final-State             Progress
```

4. To check the logs for more details. Use the following command to see the logs of specific applications.

```
[26] !yarn logs -applicationId <application_id>

/bin/bash: -c: line 1: syntax error near unexpected token `newline'
/bin/bash: -c: line 1: `yarn logs -applicationId <application_id>'
```