Maitreya Kavithkar
TE-IT 8084

DSBDA Assignment 1

Aim:
To install hadoop on a single node.

Theory:

1) What is big data? Explain various applications of big data.

-> Big data is described as volumes of data available in changing levels of complexity, produced at different velocities and changing level of ambiguity, that cannot be processed using conventional technologies, processed methods, algorithms or any commercial off the shelf solutions

Applications of big data-

i) Fraud detection
Fraud detection is a big data application example for businesses which has operations like any type of claims or transaction processing.
Number of times the detection of fraud is concluded long after the fact. At this point the damage has already been done all that's left is to decrease the harm and revise policies to prevent it in the future.
The big data platform can analyze claims and transactions of businesses. They identify large-scale patterns across many transactions or detect anomalous behaviour of some users. This helps in avoiding frauds in the future.

ii) IT log analytics.

An enormous quantity of logs and trace data is generated in IT solutions and IT departments. Many times such data goes unexamined, organizations simply don't have the manpower or resource to go through all such informations.

Big data has the ability to quickly identify large-scale patterns to help diagnosing and preventing problems. It helps organizations with a large IT department.

iii) Call center analytics

Now we turn to customer-facing Big data applications. examples, of which call center analytics are particularly powerful.

Without a Big data solution, much of the insight that a call center can provide will be ignored or exposed later.

By making sense of time/quality resolution metrics, the Big data solutions are able to identify recurring problems or customer and staff behavioural patterns. Big data can also ~~process~~ capture and process call content itself.

iv) Social media analysis.

With the help of social media we can observe the real-time insights into how the market is responding to products and campaigns. With the help of these insights, it is possible for companies to adjust their pricing, promotion and campaign placement to get optimal results.

2) What is hadoop? Explain the features of Hadoop.

→ Hadoop is an open source, Java based programming framework which supports the processing and storage of extremely large sets of data in a distributed computing environment using simple programming models.

Features of Hadoop are-

i) HDFS

HDFS stands for Hadoop distributed file system. It states that the files will be broken into blocks and stored in nodes over the distributed architecture. It provides high-throughput access to application data.

ii) Yarn

Yarn stands for Yet another Resource Negotiator. It is used for job scheduling and managing clusters.

iii) Map Reduce

This is YARN-based system for parallel processing of large dataset using key value pair. The Map task takes input data and converts it into dataset which can be computed in key value pair.

iv) Hadoop common
These java libraries and utilities are used to start hadoop. These are used by other hadoop modules. These libraries provide file system and os level abstractions.

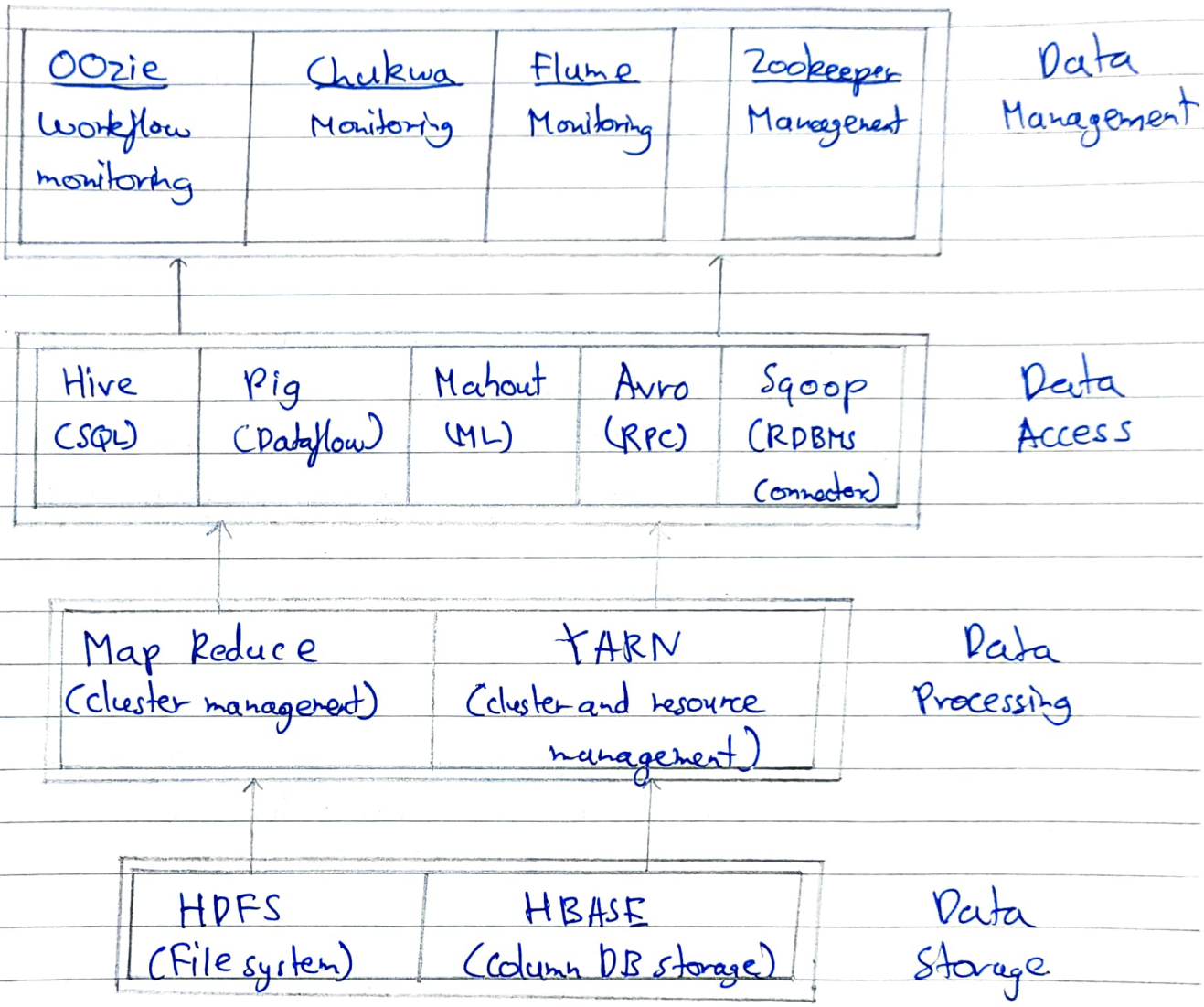
3) Explain the Hadoop ecosystem in detail. (Draw appropriate diagram)

-> Hadoop ecosystem is a platform or a suite which provides various services to solve big data problems. It includes Apache projects and various commercial tools and solutions. Following are the components that collectively form a hadoop ecosystem-

i) HDFS - Hadoop Distributed File System
ii) YARN - Yet Another Resource Negotiator.
iii) Spark - In memory data processing.
iv) PIG, HIVE - Query based processing of data.
v) HBASE - NoSQL database
vi) Mahout, Spark MLLib - algorithm libraries.
vii) Solar, Lucene - searching and indexing
viii) Zookeeper - Managing cluster.
ix) OOzie - job scheduling.

Hadoop revolves around data and hence makes its synthesis easier.

Hadoop ecosystem diagram



4) Explain the following components of hadoop in detail.

a) Namenode
b) Datanode
c) Secondary namenode.

→ a) Namenode-

The namenode is used to manage the namespace of file system. For all files and directories, Namenode presents a file system tree and metadata. This information is stored continuously on a local disk in the form of two files: the namespace image and the edit log. The Namenode is also called the master.

The namenode doesn't hold the actual data or any dataset. The namenode is considered as a single point of failure because when the namenode is down then the hadop cluster is not accesible to other users.

The namenode has the information about the list of the blocks and its location for anygiven file in HDFS. With the help of this information namenode identify how to build the file from blocks.

The configuration of namenode requires a lot of RAM because the block locations are in the main memory.

b) Data node -

Unlike namenode the datanode stores the actual data in HDFS.

The Datanode is also called as the slave. Datanode is constantly communicating with the namenode.

When a datanode starts up it tells itself to the namenode the list of blocks it holds.

The availability of data on the cluster will be unaffected if any of the datanode is down.

The configuration of datanode requires a lot of hard disk space, because the datanode contains the actual data.

c) Secondary namenode.

Secondary namenode is considered as a dedicated node in HDFS cluster The main function of the secondary namenode is to take checkpoints of the file system metadata available on namenode.

Secondary namenode is not considered as a backup namenode as it does not do any of the functions that namenode does rather it only stores checkpoints.

As we know if namenode goes down then complete HDFS is lost. To resolve this problem secondary namenode is implemented whose main function is to store a copy of FsImage file and edits log file.

FsImage is used to store the snapshot of HDFS metadata at a particular point of time whereas edit log file is used to maintain a log of transactions which includes records for every change that occurs to file metadata. When you map FsImage and Editlog you will reciene the current status of the file system metadata.

Another function of the secondary namenode is to keep the editlog file small. This is due to whenever the namenode restarts, the most recent status of FsImage is constructed by providing the edit records on the most recently saved copy of FsImage, because namenode combines FsImage and Editlog files only at the start up stage.

5) Explain the following command in detail (syntax of the command and one example each)

a) appendToFile
b) copyFromLocal
c) copyToLocal
d) count
e) touchz
f) get
g) put
h) moveFromLocal
i) rm
j) ls
k) cat.

-> a) appendToFile -
Appends single src, or multiple srcs from local file system to the destination file system. Also reads input from stdin and appends to destination file system

syntax -> hdfs dfs -appendToFile <localsrc> ... <dst>
eg -> hdfs dfs -appendToFile localfile /user/hadoop/hadoopfile

b) copyFromLocal -
Similar to put command, except that the source is restricted to a local file reference. The -f option will overwrite the destination if it already exists.

syntax -> hdfs dfs -copyFromLocal <localsrc> URI

c) copyToLocal -
Similar to get command, except that the destination is restricted to a local file reference.

syntax -> hdfs dfs -copyToLocal [ignorecrc] [-crc] URI <localdst>

d) count -
Counts the number of directories, files and bytes under the path that matches the specified file pattern. The output columns with -count are: DIR_COUNT, FILE_COUNT, CONTENT_SIZE FILE_NAME

Syntax -> hdfs dfs -count [-q] <paths>
eg -> hdfs dfs -count -q
hdfs://nn1.example.com/file1

e) touchz -
Creates a file of zero length

Syntax -> hdfs dfs -touchz URI [URI...]
eg -> hadoop -touchz pathname.

f) get -
Copy files to the local file system. Files that fail the CRC check may be copied with the -ignorecrc option. Files and CRCs may be copied using the -crc option.

syntax -> hdfs dfs -get [ignorecrc] [-crc] <src>
<localdst>

eg -> hdfs dfs -get /user/hadoop/file localfile

g) put -
Copy single src, or multiple srcs from local file system to the destination file system. Also reads input from stdin and writes to destination file system.

Syntax -> hdfs dfs -put <localsrc> ... <dst>
eg -> hdfs dfs -put localfile /user/hadoop/
hadoopfile

h) moveFromLocal -
Similar to put command, except that the source localsrc is deleted after its copied.

syntax -> dfs -moveFromLocal <localsrc> <dst>

i) rm -
Deletes files specified as args. Only deletes non-empty directory and files. If the -skipTrash option is specified, the trash if enabled will be bypassed and the specified files will be deleted immediately. This can be useful when it is neccesary to delete files from an over-quota directory.

syntax -> hdfs dfs -rm [-skipTrash] URI [URI...]
eg -> hdfs dfs -rm hdfs://nn.example.com/file
/user/hadoop/emptydir.

j) ls -
For a file it returns stat on the file and for directory it returns list of its direct children as in unix.

syntax -> hdfs dfs -ls <args>
eg -> hdfs dfs -ls /user/hadoop/file1

k) cat -
Copies source paths to stdout.

syntax -> hdfs dfs -cat URI [URI...]
eg -> hdfs dfs -cat hdfs://nn1.example.com/file1
hdfs://nn2.example.com/file2

6) Write the detailed steps required for configuration of Hadoop on single node.

-> Commands for Hadoop installation on single node ->

i) java --version

ii) sudo apt-get update

iii) sudo addgroup hadoop

iv) sudo adduser -ingroup hadoop hduser.

v) sudo apt-get install openssh-server
sudo adduser hduser sudo

vi) su -hduser

vii) ssh -keygen -t rsa -P "" "

viii) cat $HOME/.ssh/id_rsa.pub >> $HOME/ssh/
authorized_keys.

ix) ssh localhost

x) exit

xi) cd /home/student/Downloads/

xii) sudo tar -xvzf hadoop-2.9.0.tar.gz

xiii) sudo mv hadoop-2.9.0 /usr/local

xiv) cd.. x3

xv) sudo chown -R hduser /usr/local

xvi) sudo gedit ~/.bashrc

xvii) source ~/.bashrc

xviii) sudo nano /usr/local/hadoop-2.9.0/etc/hadoop/
hadoop-env.sh

xix) Change JAVA HOME

xx) sudo nano /usr/local/hadoop-2.9.0/etc/hadoop/core-site.xml
(insert given code)

xxi) sudo nano /usr/local/hadoop-2.9.0/etc/hadoop/hdfs-site.xml
(insert given code)

xxii) sudo nano /usr/local/hadoop-2.9.0/etc/hadoop/yarn-site.xml
(insert given code)

xxiii) sudo cp mapred-site.xml.template mapred-site.xml.

xxiv) sudo nano /usr/local/hadoop-2.9.0/etc/hadoop/mapred-site.xml
(insert given code.)

xxv) sudo mkdir -p /usr/local/hadoop_tmp

xxvi) sudo mkdir -p /hdfs/namenode

xxvii) sudo mkdir -p /hdfs/datanode

xxviii) sudo chown -R hduser /usr/local/hadoop_tmp

xxix) hdfs namenode -format

xxx) start-dfs.sh

xxxi) start-yarn.sh

xxxii) http://localhost:50070

Conclusion:
Hadoop installation on single node has been completed successfully.