Week 2
=======

HDFS recap

talking about the labs

Linux commands

HDFS commands

How distributed computing work (map-reduce style)

HDFS
======

block size = 128 mb

500 mb file

1st block - 128 mb
2nd block - 128 mb
3rd block - 128 mb
4th block - 116 mb

name node will hold a table
=====================

file1	block1	DN1
file1	block2	DN2
file1	block3	DN3
file1	block4	DN4

client wants to read file1 from hdfs

the request will go to the namenode

name node will tell from where to read the blocks

consider like a 1000 page book

you have an index page at the start of the book

1000 data pages

your index page is more like your name node

and your actual pages are more like your data nodes

HDFS Spark YARN
storage processing Resource manager

1 name node
4 data nodes

data nodes
4000 nodes…

why the block size is 128 mb?

what happens if we reduce the block size

64 mb

1 GB file - default block size is 128 mb - 8
1 GB file - default block size is 64 mb - 16
1 GB file - default block size is 32 mb - 32

if we reduce the block size leads to more number of blocks

1000 node cluster

if you choose a block size which is very low…

lets say block size is 1 mb

you have a file of 10 gb

10240 blocks

your name node has to keep the metadata of all of it in its table..

your namenode will get overburdened with these many entries..

what if we increase the block size…

1 GB file - default block size is 128 mb - 8
1 GB file - default block size is 256 mb - 4
1 GB file - default block size is 512 mb - 2

128 mb is a good number…

if you data grows then you can add more data nodes

when your datanodes increase to a certain level then your metadata also would have grown up..

Namenode federation - we can have more than one name node. Metadata is split across these namenodes…

Namenode federation is to give scalability

Data node failure
Replication factor - 3

Name node fails
Secondary name node

secondary name node - fault tolerance
namenode fedration - scalability

rack

10 datanodes in bangalore (1 rack)

10 datanodes in san francisco (1 rack)

10 datanodes in australia (1 rack)

3 copies are not stored in a single rack

Name node
Data node
Replication factor
Block size - 128 mb
name node federation
secondary name node
rack

Practise Labs

hadoop

you can manually install all the services

ready made VM's (virtual machine)

cloudera quickstart VM

where things are preinstalled…

Oracle Virtual box

you can install the Cloudera VM

Pseudo distributed setup

Namenode & Datanode is running on the same VM.

your system will get very slow, and you will not get the feel of production environment.

Multinode cluster - Fully Distributed.

1. Pseudo distributed - Cloudera quickstart VM
2. fully distributed

6 months lab access..

3rd party

what you can practise on that lab?

Linux commands
HDFS commands
python
DSA
Pyspark
Kafka
Structured streaming
Hive

ADF, Databricks, other cloud services..

For these cloud services you would create your own cloud account and you will practise.

Gateway node: https://g02.itversity.com
username: itv005857
password: 87420n7zlb00a5i76j5k79plfyyf40te

Name node:
http://m01.itversity.com:9870/

10 Tb of harddisk

3 data nodes

each datanode - 9 TB for HDFS
- 1 TB for local

30 TB

DFS - 27 TB (part of hdfs)
Non DFS - 3 TB (part of local storage)

Lab access - 6 months..

what if you want to extend this lab further.

6 months - 2300 Rs / $32

==========

Gateway node: https://g02.itversity.com
username: itv005857
password: 87420n7zlb00a5i76j5k79plfyyf40te

Linux commands
==========

pwd (present working directory)
/home/itv005857

whoami (the username)

linux follow a tree like structure

/ (top most directory, parent of all)

in linux we say directory and in windows we say as folder

cd is to change directory

if I have to go to my home directory
cd ~

cd .. (parent)

cd - (to go to previous directory)

cd ~

cd ../.. (will take me to the root directory)

cd . (currect)

from your macbook open the terminal

ssh itv005857@g02.itversity.com

87420n7zlb00a5i76j5k79plfyyf40te

absolute vs relative path

cd /data

/data/trendytech

/data/retail_db

absolute means you give the path starting from the /

the complete path
cd /data/retail_db/departments/

relative path means the path relative to the current location where we are..
cd ../departments/

ls command
==========

it is for listing the files and directories

blue color indicates directories
green color indicates executables
black color denotes normal file

if I press the up arrow I get the previous command

ls

ls -l

ls -lt (newest file first)

ls -ltr (oldest file first)

ls -l (gives ascending order based on dictionary)

ls -lr (gives descending order based on dictionary)

cd /data/retail_db

- at the start indicates a file

d at the start indicates a directory

ls -R (recursively print all the directories and files)

ls -R /data/nyse_all/

ls -a (to print all files including the hidden files)

hidden files start with .

ls -R -a (recursively print all files and folders including hidden ones)

ls -Ra

ls -aR

ls -latR /data

we covered

ls
cd

========

how to create an empty file - touch

touch file1

owner group others

rwx

r - read (4)
w - write (2)
x - execute (1)

6 6 4

owner - read and write
group - read and write
others - read

6 4 4

chmod 777 file1

chmod 764 file1

touch is to create a new empty file
also it can modify the timestamp to the current timestamp

cat is to see the content of the file

mkdir is to create a new directory

mkdir dir1

mkdir dir2 dir3 dir4

rmdir is to remove a directory

rmdir dir1 (it will remove the directory as it is empty)

but if the directory is not empty it will show error.

rm filename will remove the file

rm dir (it will give error saying its a directory)

rm -R dir2

cp command

it is used to copy files and directories

cp file1 file2

cp file1 dir3

so cp works like a copy paste

cp -R dir3 dir4

mv command
==========

=> is used to move a file from one directory to another (cut paste)

=> rename a file

vi samplefile

press i button on keyboard - insert mode

then write the content

press esc

shift + :

wq (w for save, q for quit)

head

tail

cat /data/retail_db/orders/*

headers.csv
order_id, timestamp, customer_id, order_status

orders.csv
1,2013-07-25 00:00:00.0,11599,CLOSED
2,2013-07-25 00:00:00.0,256,PENDING_PAYMENT
3,2013-07-25 00:00:00.0,12111,COMPLETE
4,2013-07-25 00:00:00.0,8827,CLOSED

```
order_id, timestamp, customer_id, order_status
1,2013-07-25 00:00:00.0,11599,CLOSED
2,2013-07-25 00:00:00.0,256,PENDING_PAYMENT
3,2013-07-25 00:00:00.0,12111,COMPLETE
4,2013-07-25 00:00:00.0,8827,CLOSED
```

cat file1

cat > file2

cat >> file1 (is for append)

du -h /data/trendytech

for searching there is a command (grep)

HDFS Commands
==========

hadoop fs (will give you the list of all commands)

hdfs dfs (will give the same results)

ls (we are listing files on the gateway node)

in case of my local
/home/itv005857

hadoop fs -ls /user/itv005857

in local - /home/itv005857
in hdfs - /user/itv005857

hadoop fs -ls
hadoop fs -ls /user/itv005857

whenever we fire a hdfs command mostly it will interact with the namenode

ls

however a few commands will even talk to the datanodes

cat, tail

hadoop fs -ls / (root directory of hdfs)

hadoop fs -ls -t -r / (reverse time)

hadoop fs -ls -S -h /order_results (based on size)

hadoop fs -ls /user | grep itv872598

hadoop fs -mkdir /user/itv005857/retail_db

hadoop fs -mkdir /user/itv005857/dir1/dir2 (will not work)

hadoop fs -mkdir -p /user/itv005857/dir1/dir2

I want to bring file from my local to hdfs

/data/trendytech bigLog.txt

from local to hdfs (data folder)

put
copyFromLocal

hadoop fs -put <local file path> <hdfs path>
hadoop fs -copyFromLocal <local file path> <hdfs path>

from hdfs to local

copyToLocal
get

what if we have to move a file from one location in hdfs to other other location in hdfs

cp (copy from one location in hdfs to other location in hdfs)

mv (move from one location in hdfs to other location in hdfs)

hadoop fs -df -h /user/itv005857

hdfs fsck datanew/biglog.txt -files -blocks -locations

Few things to remember regarding the lab
==========

in our gateway node we have a data folder inside root

cd /data ---- gateway node

hadoop fs -ls /public ---- hdfs

to practise sql

go to your lab.. open the terminal

mysql -u retail_user -h ms.itversity.com -p
itversity

show databases;

use retail_db;

for retail_db we do not have the write permission

use retail_export for creating your tables

mysql -A -u retail_user -h ms.itversity.com -p

we have learnt HDFS,

AWS - amazon S3
Azure - ADLS gen2

datalakes in cloud

HDFS + Pyspark

ADLS gen2 + Databricks

pyspark - week3

databricks - week11

HDFS - distributed file system (blocks)

ADLS gen2/ amazon S3 - Object based storage

id - a unique identifier
value - the content
metadata - who can access the file, what kind of data is stored..

Distributed file system vs Object based storage

hdfs vs amazon s3/ adls gen2

HDFS is not persistent but amazon s3/ adls gen2 are persistent.

4 node hadoop cluster

4 datanodes

what if you shut down the cluster

your storage is tightly coupled with compute

if you just want to store but not compute...

amazon s3/ adls gen2

they are not coupled with compute

hadoop - 4 node cluster (hdfs datalake)

other hadoop cluster - 8 node cluster

amazon s3/ adls gen2 - any number of clusters can use this data..

Mapreduce
=========

HDFS - storage

Mapreduce theory is very important

but practicals are not important

Map Reduce

2 phases - Map Reduce

(K,V) -> Map -> (K,V)

(K,V) -> Reduce -> (K,V)

(rollno, studentname)
(1, Satish)
(2, Kapil)

Mapreduce is a programming paradigm

traditional programs work when you data is kept on one machine

if the file size is 500 mb then there are 4 blocks.

so 4 mappers will run

1 GB file - 8 blocks

8 mappers will run but 4 will run in parallel

is the data going to the code?

or the code is going to the data?

code is generally very small but data is big

principal of data locality... it means the data is processed on the same machine where it is kept.

the output from mapper is not the final output.. but its an intermediate output..

the output of all the mappers go to one other machine where reduce activity will take place.

mapper gives you parallelism

4 mappers and 1 reducer

file1 (500 mb)

hello my name is sumit
i love to teach big data
big data is quite interesting
people call me sumit sir
hello this is me

the output
==========
(hello,2)
(sumit,4)

(teach,10)

500 mb file will be divided into 4 blocks

```
========
hello my name is sumit
i love to teach big data          b1
sumit sir
==========
big data is quite interesting   b2
big data is nice
=======
people call me sumit sir          b3
people are good
========
hello this is me                        b4
hello hi
===========
```

map and reduce understands only key value

record reader


```
========
hello my name is sumit
i love to teach big data          b1
sumit sir
==========
big data is quite interesting   b2
big data is nice
=======
people call me sumit sir          b3
people are good
========
hello this is me                        b4
hello hi
===========
```


input to the record reader is

hello my name is sumit

and output is
(0, hello my name is sumit)
(1, i love to teach big data)
(2, sumit sir)

(0, hello my name is sumit hello)
(1, i love to teach big data)
(2, sumit sir)

| mapper

output from mapper will be

```
mapper - 1
(hello,1)
(my,1)
(name,1)
(is,1)
(sumit,1)
(hello,1)

mapper - 2       output from all mappers goes to other machine
(hello,1)
(my,1)


mapper - 3
(sumit,1)
(hello,1)


mapper - 4
(hello,1)
(sumit,1)
(hello,1)


the mappers output is shuffled and brought to one other machine

this is done so that reducer can work on it..


map -> shuffled  -> Sorting

DN2
=====
(hello,1)
(my,1)
(name,1)
(is,1)
(sumit,1)
(hello,1)
(hello,1)
(my,1)
(sumit,1)
(hello,1)
(hello,1)
(sumit,1)
(hello,1)


(hello,1)
(hello,1)
(hello,1)
(hello,1)
(hello,1)
(hello,1)
(is,1)
(sumit,1)
(sumit,1)
```

(hello,{1,1,1,1,1,1})
(is,{1})
(sumit,{1,1})

map -> shuffle -> sort -> reduce

Map
Reduce
we as a programmer write the code

but shuffle and sort is taken care by the framework

reduce code takes the list of values and sums it up

(hello,6)
(is,1)
(sumit,2)

hello how are you

(0, hello how are you)

(hello,{1,1,1,1,1})
(how,{1,1})
(are,{1,1,1})
(you,{1,1,1,1,1})

record reader -> map -> shuffle -> sort -> reduce

map reduce