

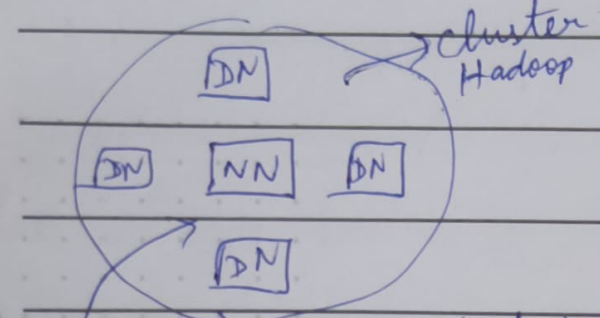
→ Hadoop is NO-SQL database & can handle the ~~un~~ unstructured data)

(Hadoop is ~~also~~ written in Java) Hadoop

NameNode - Master Node
Data Node - Slave

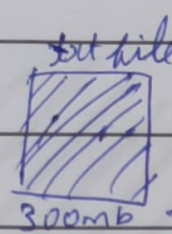
1) HDFS → Storage layer
(Hadoop Distributed Fully distributed System)

2) Map Reduce (YARN) → Processing layer.



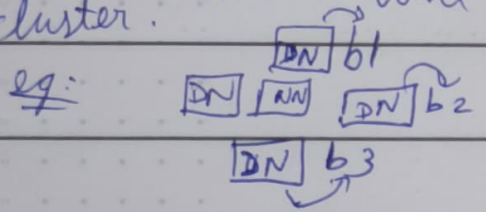
more amount of data in one block = 128mb -

Hadoop FS (Client library) stores a txt file in Hadoop by sending it to NN



3 block required

→ These 3 block will be distributed across the cluster.



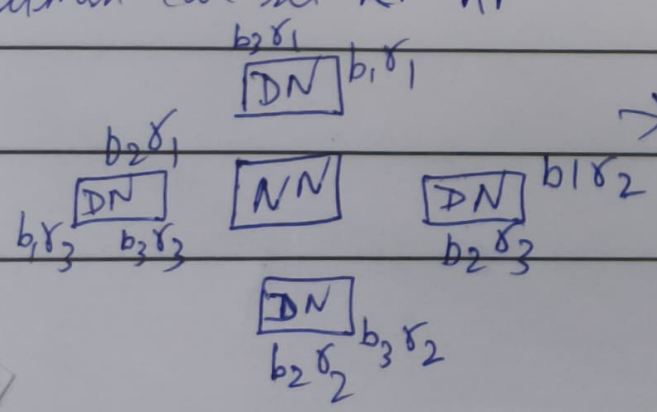
300mb → 128 + 128 + 44

→ Job of Name Node is to allocate blocks to Data Node & Job of Data Node is to store data.

→ Single pt of failure: One of the Data Nodes becomes dysfunctional (default)

→ Replication Factor of Hadoop = 3 (means each block is present three times)
eg: b1, b2 & b3 are present / replicated 3 times

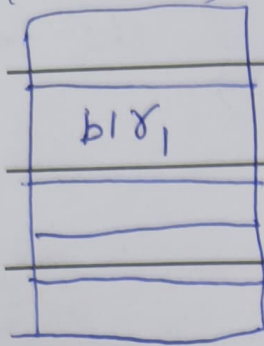
→ admin can set ~~RF~~ RF



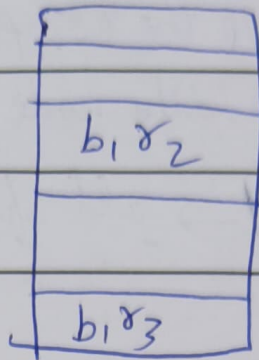
→ This is done to keep data accessible even if one Data Node becomes dysfunctional.

Rack Awareness

Rack-1
(Cluster-1)



Rack-2
(Cluster-2)



Replica-1 is in R1 (Cluster-1) & the same blocks ~~would~~ ~~be~~ other replicas & would be in the Rack-2. So even if Rack-1 is busy with other blocks or goes down completely, the other replicas in Rack-2 can be accessed & used.

- 3 ~~blocks~~ → 3 replicas → So 9 nodes.
- The info of which blocks which replica is present in which Data Node is present in Name Node. (Stores Meta Data)
- If NN goes down then it's again Single pt of Failure.
- In this case, the ~~backup~~ ^{Manual} backup of Name Node is stored in Secondary Name Node in Hadoop-1.
- In Hadoop-2, the data is stored in High Availability Node. This data ~~is created~~ ^{Node} simultaneously made to ~~run~~ ^{run} & ~~created~~ with the Name Node so that no data is lost while in Hadoop-1, if we take backup of Name Node in Secondary NN at 6pm & NN goes down at 7pm, ~~it~~ ^{the} ~~loss~~ of data is lost, this is not the case in Hadoop-2.
- Federation: Collection of Name Nodes with different Name Nodes handling different data.
- High Availability Node is Autobackup of Name Node.
- Hadoop Demons → Name Node, Data Nodes & Secondary Name Nodes.
- ~~Star~~ ~~Java~~ Hadoop modes: Fully distributed, Standalone & Pseudo distributed modes.
- Pseudo Distributed → Setting ^{up} of Hadoop in Virtual Machine.
- Standalone → only for Texting ¹ purpose.

→ Hadoop can't create new files & can only move file from local to hadoop.

Hadoop VM Commands

- ① ~~start~~ start-dfs.sh → To start & Configure hadoop
- ② jps — To list all the components of the hadoop system (ie, name node, data node... etc)
- ③ hadoop fs ~~add~~ -ls / → all files.
(hadoop fs -ls /)
- ④ ~~hadoop~~ hadoop fs ~~add~~ ~~hadoop~~ ~~folder~~ → hadoop folder.
- ④ hadoop fs -mkdir /del_aug → Creates a new directory.
- ⑤ Go to localhost:50070 to go to the hadoop website, then click Utilities & then click browse the file system to see if ~~a file~~ the dir has been added or not.
- ⑥ hadoop fs -put file1.txt /del_aug → To move file1.txt from local to hadoop. Go back to browser to check if file has been added to hadoop or not.
- ⑦ Distribution & replication has no meaning while running in the VM because it's pseudodistributed mode & there is only 1 real node (ie local laptop) so, RF=1. (can check RF in browser also)
- ⑧ hadoop fs -get '/hadoop path' '/local path' to get folder from hadoop & save it to local.

Additionally, you can ~~ad~~ rename the file's name before copying it to local by adding the filename at the end of the local path ie '/local path/filename'

→ Hadoop can't create new files & can only move files from local to hadoop.

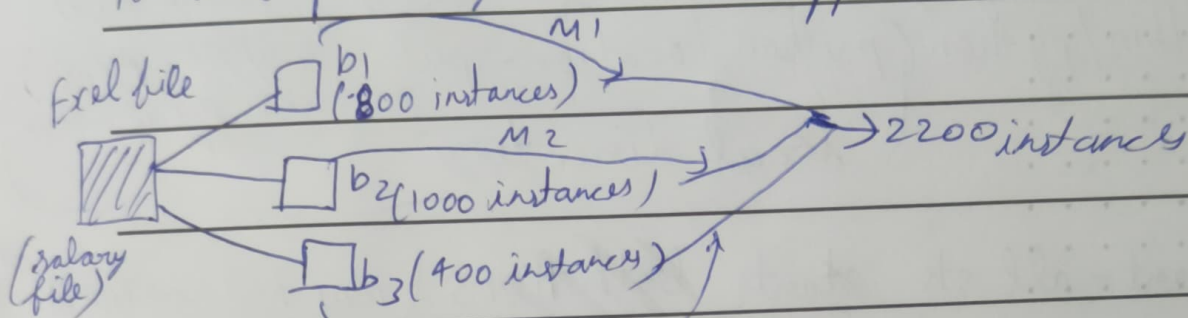
Hadoop VM Commands:

- ① ~~start~~ start-dfs.sh → To start & Configure hadoop
- ② jps - To list all the components of the hadoop system (ie, name node, data node... etc)
- ③ hadoop fs ~~ls~~ -ls → all files.
(hadoop fs -ls /)
- ④ ~~hadoop fs~~ ~~mk~~ hadoop folder.
- ④ hadoop fs -mkdir /del_aug → Creates a new directory
browser then,
- ⑤ Go to localhost:50070 to go to the hadoop website, then click Utilities & then click browse the file system to see if ~~a file~~ the dir has been added or not.
- ⑥ hadoop fs -put file1.txt /del_aug → To move file1.txt from local to hadoop. Go back to browser to check if file has been added to hadoop or not.
- ⑦ Distribution & replication has no meaning while running in the VM because it's pseudodistributed mode & there is only 1 real node (ie local laptop) so, RF=1. (Can check RF in browser also)
- ⑧ hadoop fs -get '/hadoop path' '/local path' To get folder from hadoop & save it to local.

Additionally, you can ~~ad~~ rename the file's name before copying it to local by adding the filename at the end of the local path ie '/local path/filename'.

⑨ hadoop fs - rm can be used to remove a folder/dir from hadoop. Same for all linux commands \rightarrow just do hadoop fs ~~rm~~ - linux command to run the linux command for hadoop.
(eg: hadoop fs ~~ls~~ - del * to list all dirs) starting with 'del'.

* Map Reduce: (get another Resource Negotiator)
Parallel processing network. Mappers \rightarrow m_1, m_2, m_3



\rightarrow Mappers runs queries on only their assigned blocks parallelly & the aggregate them later.

MR2 (hadoop-2): Daemons are:

(1) Resource Manager (Master) - Manages cluster w/ resources & scheduling of jobs

(2) Node Manager (slave) \rightarrow Does all the allocated jobs

(3) Application Master \rightarrow Present for each jobs
 \downarrow
(Negotiates with Resource manager to schedule jobs.)

This is actually not a daemon.

Commands:

① Start - yarn.sh to start the Map reducer

② hadoop jar ~~class~~ hadoop-2.9.2 / logs / yarn-haduser

~~resource manager~~ / usr / hadoop-2.9.2 / share / hadoop / tools
lib / hadoop-streaming-2.9.1 jar - input "User path to the file / file.txt" - output "path to mapper.py" - output "path to reducer"

(Hadoop Vendors: Apache, Cloudera, IBM Big Insights)

before the prev page commands, do

chmod a+x / chmod 777 (give permission to your mapper file)

chmod a+x mapper.py reducer.py

#1 -> Shabong -> To run a executable python file
|usr/bin/python (python location - put in mapper.py & reducer.py if not in local directory)

-> Start-all.sh starts both HDFS & Map Reduce YARN

HIVE (Use ; to get out of newline/loop)

① Start-all.sh (before this go to the local directory where you want to start hive)

② hive

③ hive > show databases; to show all databases.

④ hive > set hive.cli.print.current.db=true;

⑤ hive > create database; del-aug (to create new database)

⑥ hive > use del-aug; > Create table employee (name string, salary float, dept string)

⑦ Again go to localhost:50070 to check progress.

⑧ ~~hive~~ hive (del-aug) > create table employee (string name, float salary, string dept)

Continue (Creating a table ~~salary~~ float salary string dept)

⑨ hive (del-aug) > row format delimited;

do together (don't include at end of any line) fields terminated by ',';

⑩ hive (del-aug) > show tables;

(Note: Sometimes don't use `(-)` while specifying a path)

(11) ~~drop~~ hive (del-aug) > drop table employee;
(to delete an existing table)

(12) Create a ~~csv~~ ^{Comma separated} file in .txt in local

(13) ~~load~~ hive (del-aug) > load data ^{local} inpath ~~path~~
↙ 'path of file' into table
(path = 'employee.txt') employee;

(14) Select * from employee

(15) describe formatted employee; to check if all data are loaded in correct format into the table

(16) set hive.cli.print-header=true; is used to

exercise: print the header names of the table as well

(17) Drop table employee, then push ~~the~~ emp.txt files to ^{load data from} hadoop & then

(18) load data inpath 'HDFS path of files' into ^{hadoop} table employee after creating table employee

to hive

go to HDFS browser to get HDFS path of files

& path was '/del-aug/employee.txt'

(use hadoop fs -put filename.txt /hadoop dir path
to move files from local to (del-aug)
hadoop first)

(19) Stop-all.sh is used to stop all daemons of hadoop.
→ includes all files whose name starts with emp

(20) hadoop fs -put emp* into '/del-aug'

→ This happens only for Hive Internal Table.

(21) Hive moved data from ~~the~~ original '/del-aug' directory in Hadoop system to '/user/hduser' directory within the same hadoop system.

(22) Hive's Table, database & other files & data can be viewed by accessing the hive directory within the ~~hive~~ hadoop system.

→ To solve this problem, we create an external table

→ To do this: (name)

> Create external table ~~create~~ employee (name string, salary float, dept string)

> row format delimited

> fields terminated by ','

> location '/del-aug';

→ 4 lines

(location in HDFS where the data files emp.txt, emp2.txt, employee.txt are stored)

(23) If you put the ^{external} table's ~~location~~ into the

~~the~~ same location as the data files, then you don't need to load data into the table ~~the~~ because the data is automatically loaded into the external table.

(Note: The location should only have data files & the directory shouldn't have any other files or ~~sub~~ sub-directory. If it has a subdirectory Hive will throw an error.)

24) * Creating Partitions in the table:

hive (del-aug) > create table part-emp (name string, sal string, dept string)

> partitioned by (country string, state string)

> row format delimited

> fields terminated by ',';

> show partition (table name) part-emp; → shows the partitions

~~create~~ > load ~~data~~ local (emp.txt) data inpath 'file path' into table part-emp
(or could load data from HDFS)

> partition (Country = 'US', State = 'NY');

→ Now → select * from part-emp; to see the changes.

→ For ~~ext Table~~ external Table

do → > create external table t1 (t1 str, ...)

> partitioned by (t2 str, ...)

> row format

> field terminated by ',';


Now type, (table name)

> alter table ~~part-emp~~ part-emp

To add
dif partitions
just change
locations accordingly
everytime

> add partition (country = 'US', State = 'NY')

> location 'path' (location of the table)

 make a table of
(25) To ~~select~~ only Karnataka employees & do
create table tbl as select * from ~~employees~~^{part-emp}
→ All this we did till where state = 'KA';

~~To do this~~ now is static partitioning.

→ Dynamic Partitioning:

① set hive.exec.dynamic.partition = true;
(for strict partitioning to prevent unnecessary
load & usage of Map Reducers / mappers)

② set hive.exec.dynamic.partition.mode = nonstrict;
(non-strict → optional)