- LINUX environment
- JAVA

HADOOP → one can't run Hadoop on Windows.
∴ We need VMWare to run it on LINUX environment

As hadoop is written in Java, we need to install Oracle JDK too.

Hadoop runs on both Windows and LINUX, however, LINUX IS THE ONLY SUPPORTED PRODUCTION PLATFORM, Windows is only supported as a development platform.

TYPES OF DATA:

(1) Structured Data:
→ Easy to search and organize, as it is contained into rows and fields

> THE DATA YOU MIGHT STORE IN AN EXCEL SHEET IS CALLED STRUCTURED DATA

This data is managed using Structured Query Language (SQL)
eg: Tables, excel, spreadsheet.
- Has a defined SCHEMA

(2) Semi Structured Data:
This type of data is having some consistent characteristics but doesn't conform to a structure as rigid as is expected with relational database.

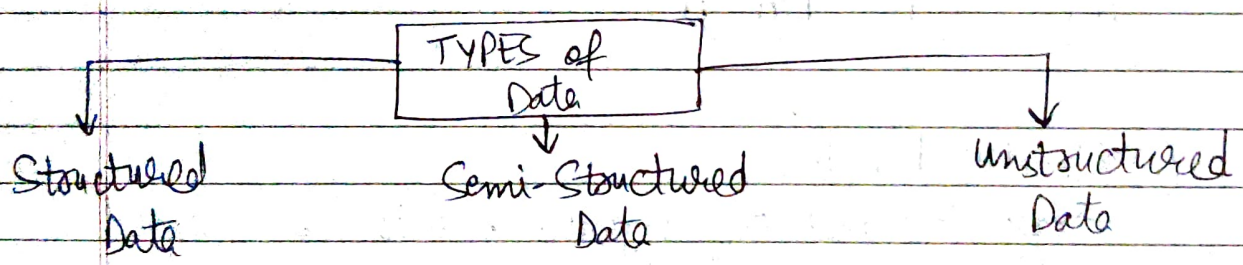
eg:- E-Mail, though the actual content is unstructured, it still contains structured data as Name & Email Address of sender & recipient.

Other examples are: JSON, XML, CSV files

(3) Unstructured Data:

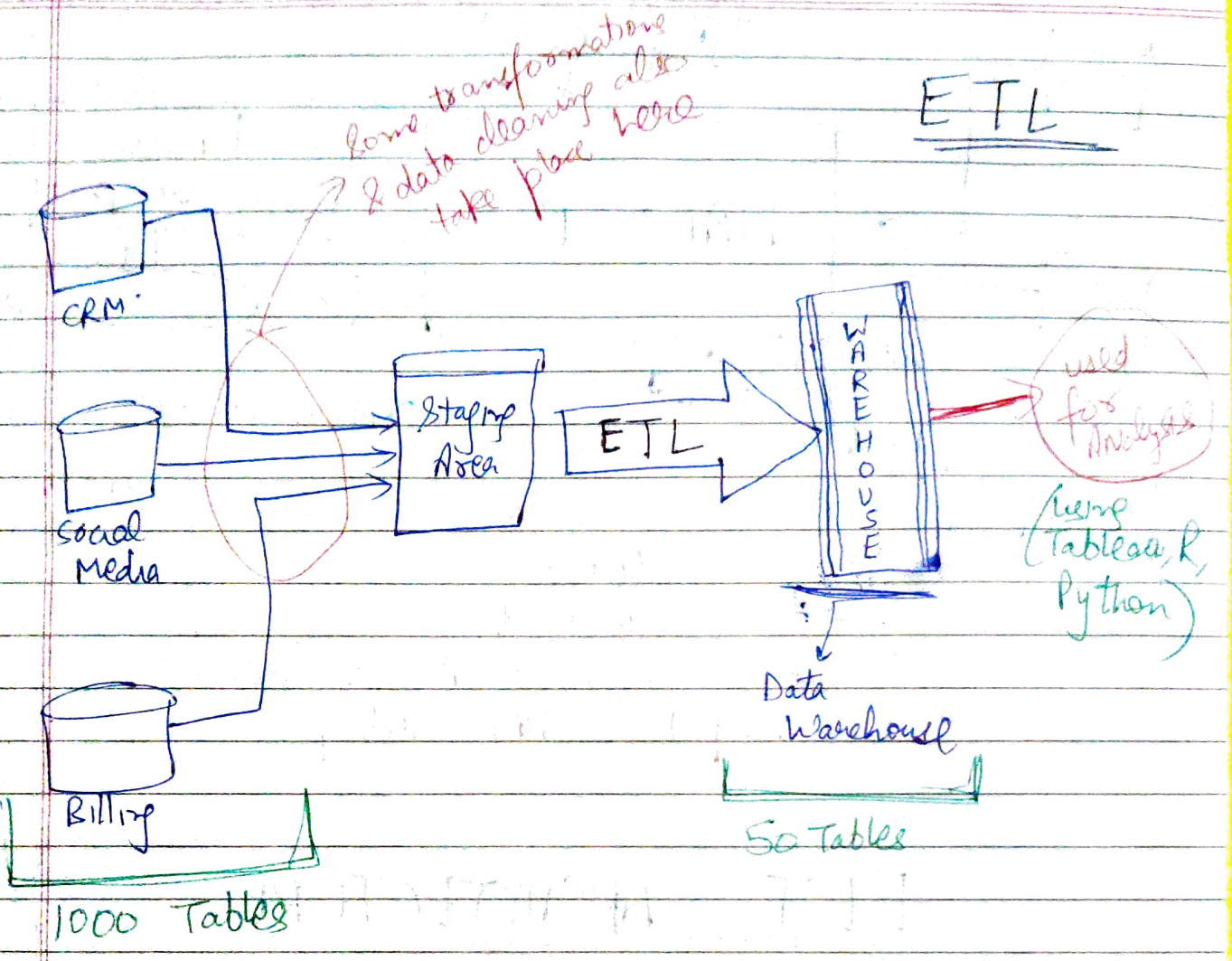
This data can't be contained into a row-column database & doesn't have an associated DATA Model + No Defined Schema is there

- Its queryed using NoSQL
- eg:-
  - .JPG, .mp3, .mp4, doc files, ppt etc



Data WareHouse (ETL)

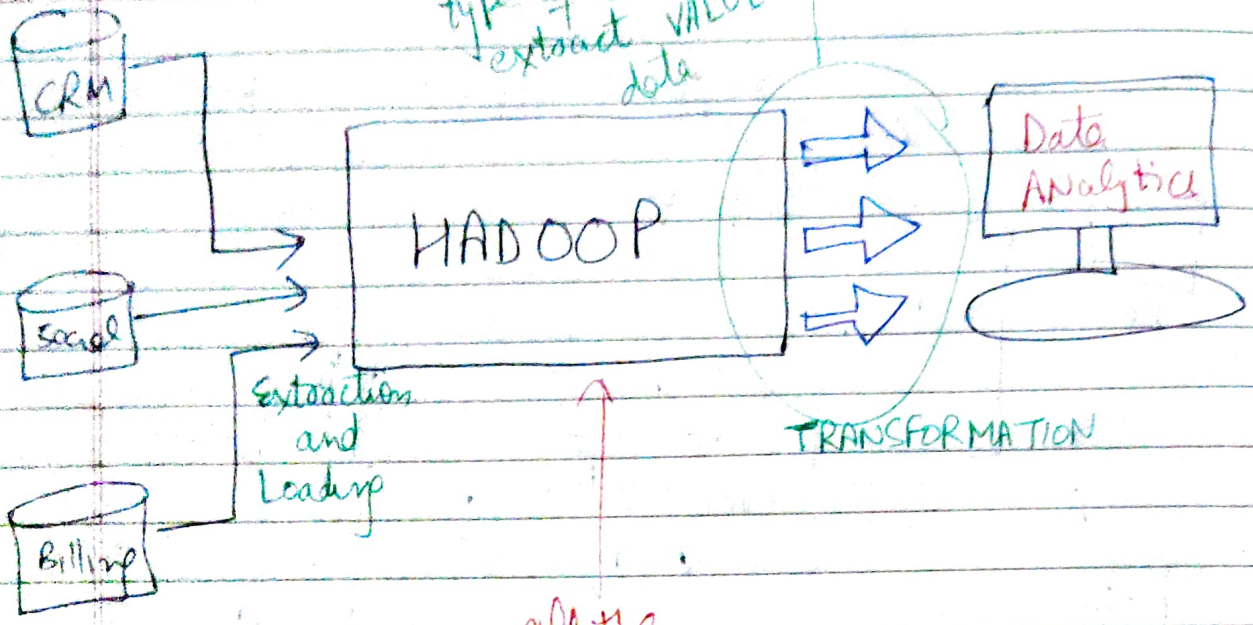
Evergreen
Page No.
Date : / /201



If the data becomes very big, one can't handle it on a data warehouse.
∴ We need a separate framework called Hadoop

Also, as Nowadays data becomes big, the transformation takes a lot of time. Thus the ANALYST are tended to wait for hours.
∴ Rather than ETL, we introduced ELT.

Various jobs like sum, depending upon the type of data to extract VALUE out of data



all the data, be it structured OR unstructured is dumped into HDFS - Hadoop

ELT ARCHITECTURE

Evergreen
Page No.
Date : / /201

→ Hadoop is basically a platform used for storing BIG DATA in a cluster, and is an OPEN SOURCE, DISTRIBUTED platform for Big Data STORING & PROCESSING.

★ Hadoop's Storing Way : (HDFS)
→ Hadoop Distributed file System

★ For Processing, we have MapReduce
→ used for data processing

★ YARN (Yet Another Resource Negotiator)
→ Resource Management

History :-

The Name Hadoop is Not an acronym, it's creater Doug Cutting, explains :

"It is the name my kid gave a stuffed yellow elephant. Short, easy to pronounce & meaningless"

Also, other projects in Hadoop system are also having names which were completely unrelated to their function, often with an elephant or other animal (eg :- PIG).

★ IT WAS INVENTED IN 2006 BY TWO PEOPLE :
MIKE CAFARELLA & DOUG CUTTING

Evergreen
Page No.
Date : / /201

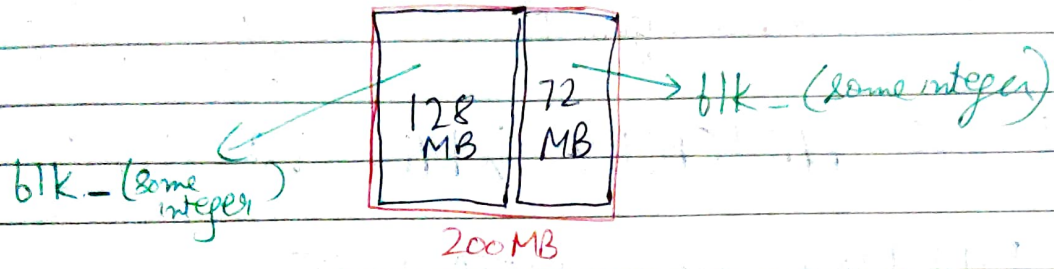
Writing Data in HDFS

* These racks are placed in different geographical regions (so that if one goes down, due to some Natural calamity or something, we can get data from

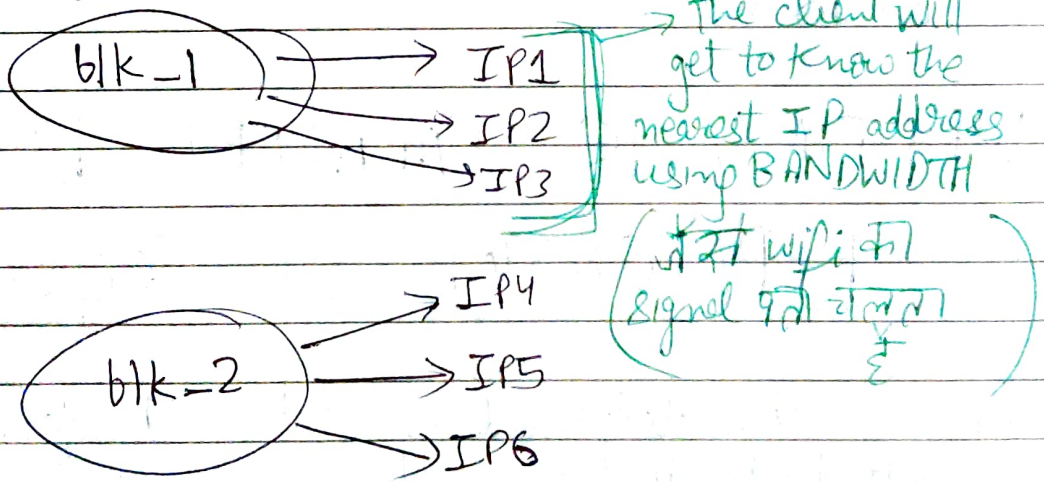
* Data in all the datanodes is not SKEWED, every NODE HAS EQUIVALENT DATA

NameNode is having a block size:

if for a 200MB file, the block size is 128MB, the block will get split into 128MB + 72MB

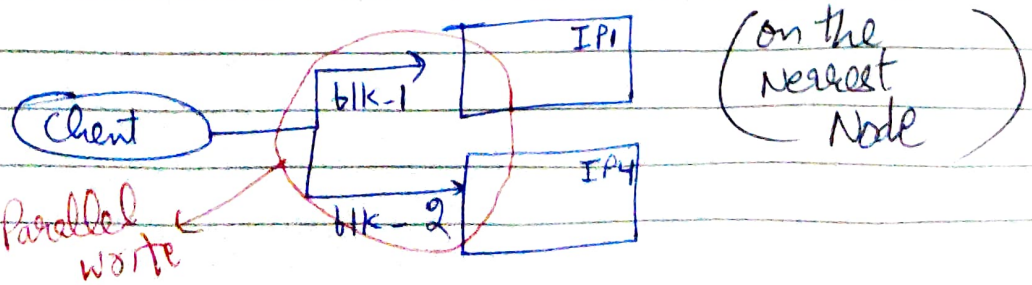


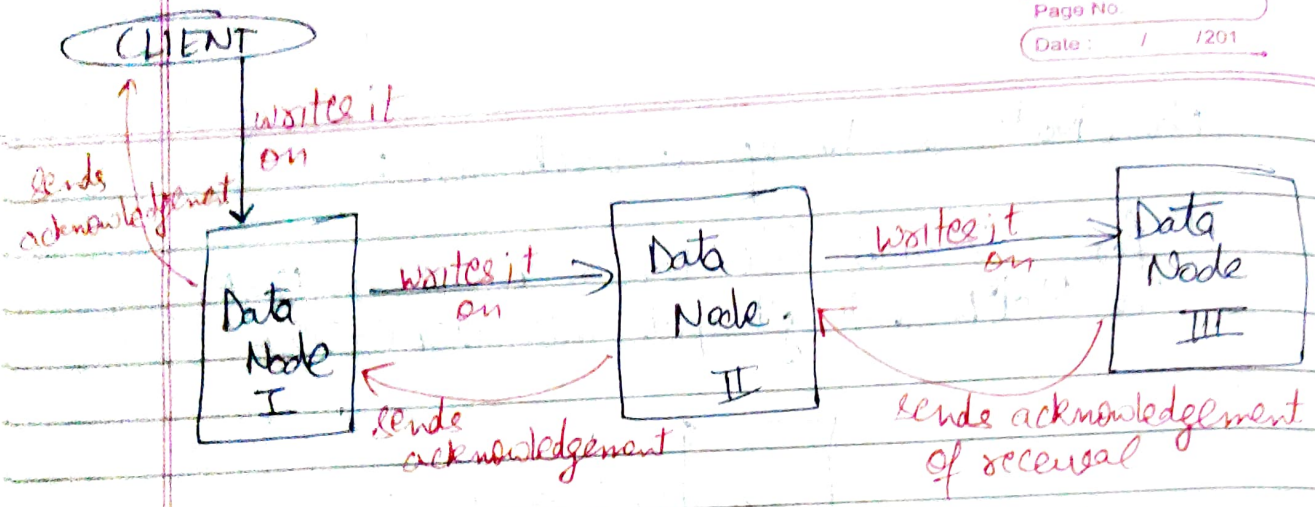
Two blocks on the client Machine will be made & NameNode will send the Number of IP addresses of Data Nodes based on the replication factor (to store it on DataNodes)



Then for each block, the client will check the Nearest IP to it. eg:- for blk-1 the nearest is IP1 & for blk-2 the nearest is IP4

THEN WRITING WILL BE DONE PARALLELY





THIS IS BASICALLY WHAT WE CALL

PIPELINE OF DATA NODES

∴ The Client writes on one Data Node only. Then that Data Node writes on the Next. (sequentially. Replicas are formed)

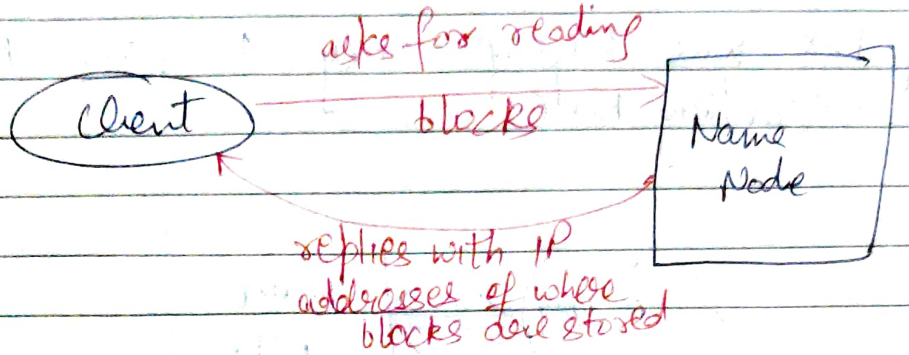
WRITING IS PARALLEL BUT

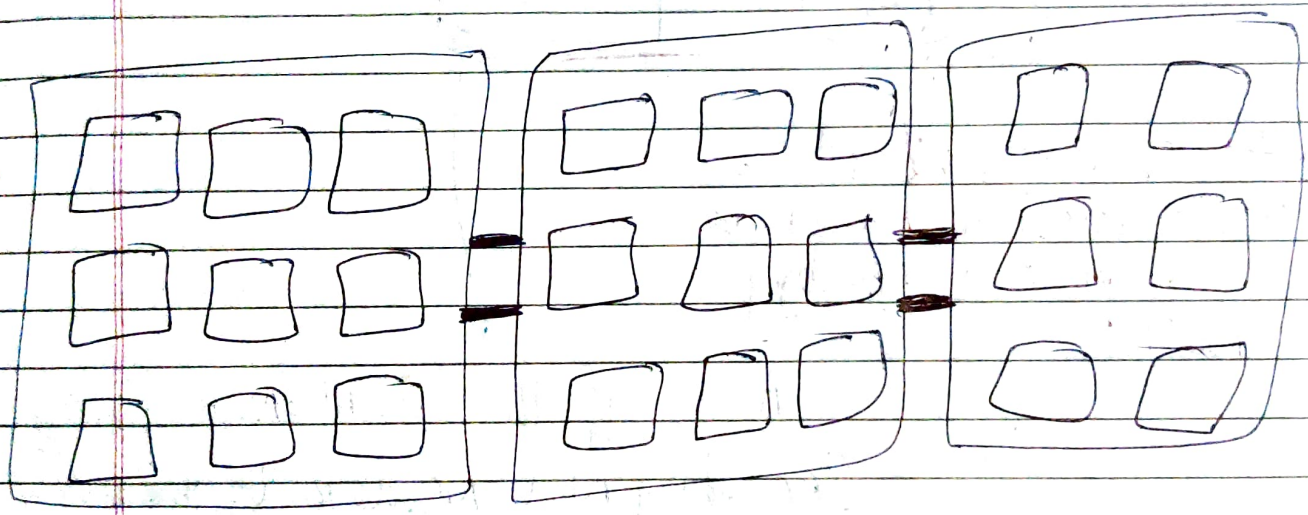
REPLICATION IS SEQUENTIAL

Reverse acknowledgement is sent back from one data node to other when it receives the data.

Everygreen
Page No.
Date : / /201

Reading Data in HDFS

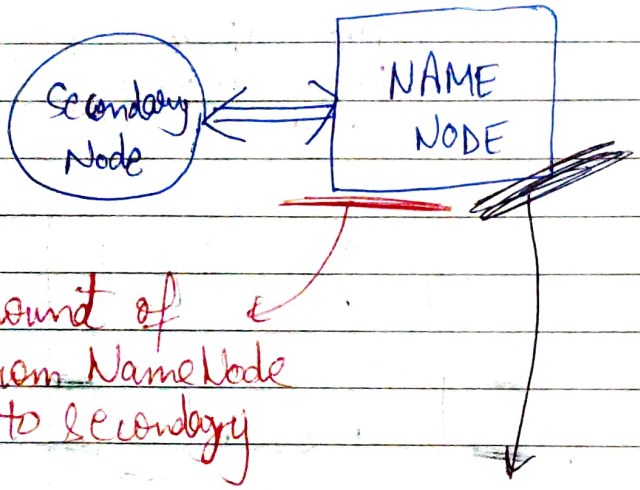




READING IS DONE SEQUENTIALLY

HADOOP 1.X VS HADOOP 2.X

Hadoop 1.x was having only 1 Active Node (Name Node) & one Secondary Node



After small amount of time data from NameNode was copied to Secondary Node

ALSO, if the NAMENode failed, Secondary Node Needs to be manually replaced with NameNode & till then the complete cluster went down.

THUS, IN HADOOP 1.X, NAMENODE WAS THE SINGLE POINT OF FAILURE IN CASE OF HADOOP.

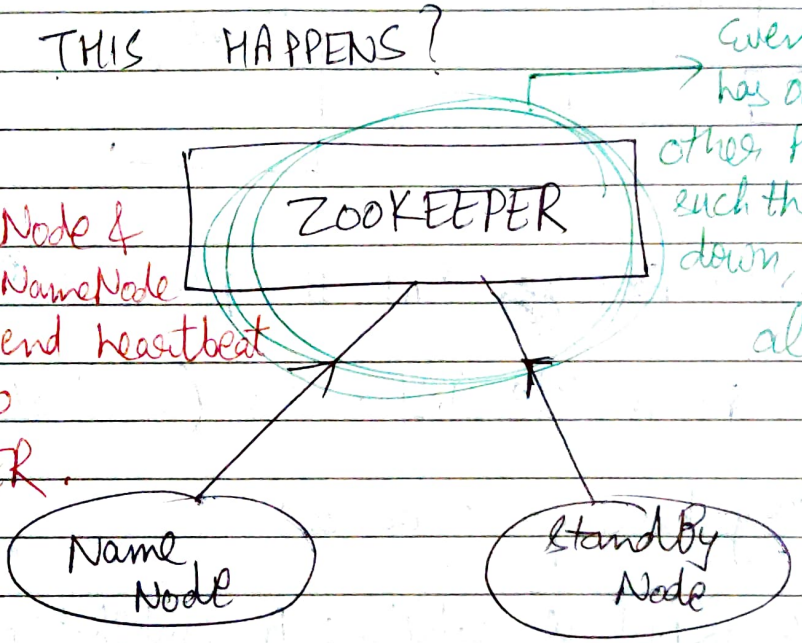
IN HADOOP 2.X, WE INTRODUCED THE CONCEPT OF STANDBY NODE

NAME NODE

STANDBY NODE

When NameNode Crashes down in Hadoop 2.x, the Standby Node immediately replaces it (without any lag)
(~~Stat~~ UPS replace ~~that~~ & Computer Power ~~off~~)

⭐ HOW THIS HAPPENS?



Given the zookeeper has ONE ACTIVE & other PASSIVE NODES, such that if one goes down, other gets alive.

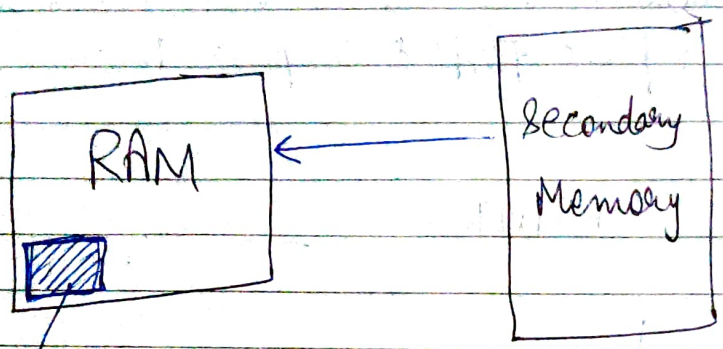
Both NameNode & StandBy NameNode used to send heartbeat signals to ZOOKEEPER.

If the Zookeeper didn't received any signals from NameNode, then it will activate STANDBY node at that instant.

⭐ THE METADATA OF NAMENODE WAS ALSO EXACTLY SAME AT STANDBY NODE. HOW DID THEY MAINTAINED THE EXACT SAME COPY ALWAYS

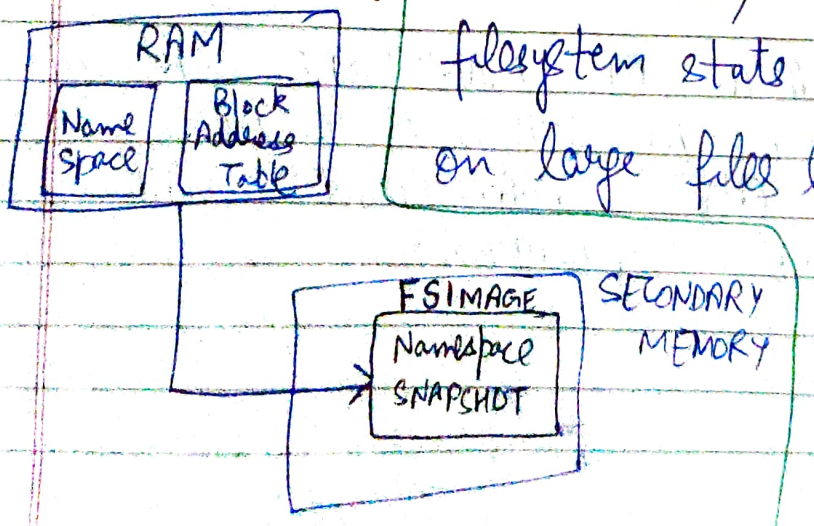
So, NameNode contains 2 things:

1. FSImage
2. Edit Logs

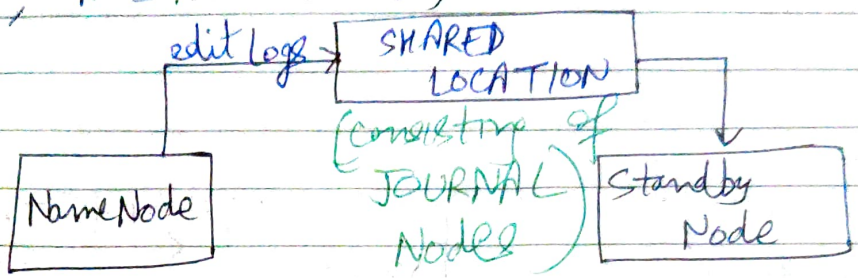


⭐ FSImage: RAM में पड़ी होती है file system की complete information (so that the users may have faster access).

But, as RAM is volatile; A snapshot of the File System is also saved in the Secondary Memory called FS Image. HOWEVER, updating filesystem state repeatedly on large files like FSImage is highly IMPRACTICAL



∴ We have 'Edit Log' files, which are WRITE AHEAD Log files. (इसमें जो भी चीज़ FS Image में edit करने वाली है, वो डाली जाती है)



The NameNode stores modifications to the file system as a log appended to a native file system file, EDITS

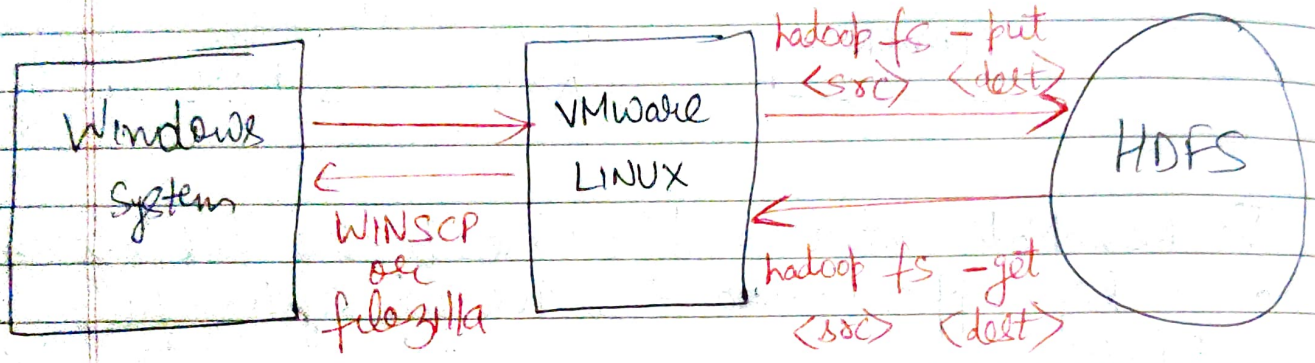
When a namenode starts up, it reads HDFS state from an image file, fsimage, and then applies edits from EDIT LOG file

It then writes new HDFS state to the fsimage & starts normal operation with an empty edits file

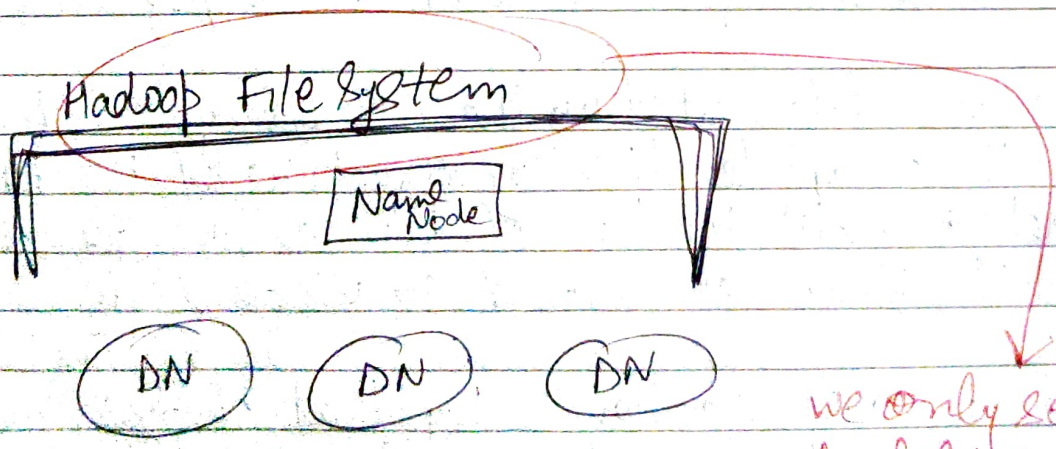
FSImage is a file stored on the OS filesystem that contains the complete directory structure of the HDFS with details about location of data on datablocks & which blocks are stored on which Node.

EditLogs is a transactional log that changes the HDFS or any actions performed on clusters.

It records the changes since the last FSImage was created, it then merges the changes into FSImage file to create a new FSImage file.



So there are three different FILE systems present, changes on one of them will Not reflect in others



However, in reality there are a number of blocks & replicas made which are distributed on the clusters on different Nodes.

Evergreen
Page No.
Date : / /201

TO KNOW THE METADATA of A FILE IN HDFS CLUSTER:
we use comomand `fsck`

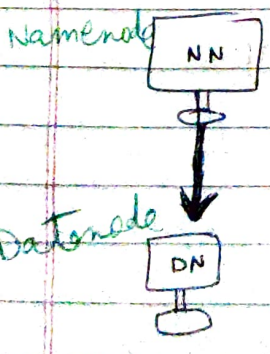
hadoop fsck (location of file in HDFS)

- files
- blocks
- locations

optional

TO CHANGE THE REPLICATION FACTOR OF A SPECIFIC FILE IN HDFS:

```
hadoop fs -setrep 3 /user/location
```



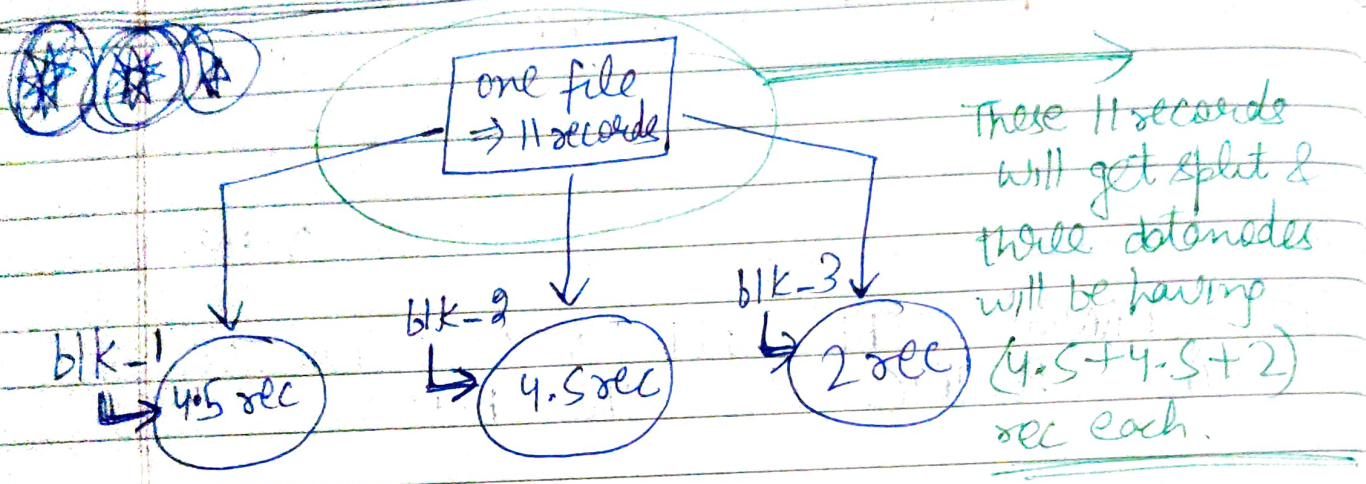
→ In a single Node Cluster, if you make replication factor as 2,

(1) On the NameNode level it will get changed to 2.

(2) However, on a datanode level there will be one replica only (which can be seen using fsck command)

HADOOP DOESN'T ALLOW TO KEEP 2 REPLICAS ON SAME NODE

(3) However, if a new Node is added to the cluster, then that Node will contain the other replicas.



However, when the processing of this data will take place, splits will be made.

split 1 ⇒ (4.5 rec) + (0.5) rec
↳ from blk-1 ↳ from blk-2

split 2 ⇒ 4 rec → remaining blk-2

split 3 ⇒ 2 rec → blk-3

THE NUMBER OF MAPPERS THAT WILL BE USED FOR DATA PROCESSING HAS NO RELATIONSHIP WITH THE NUMBER OF BLOCKS THE FILE IS STORED INTO

Evergreen
Page No.
Date: / /201

Q(1) Suppose we have:
1 image file of 1GB,
it will be stored on 8 different datanodes (if blocksize = 128MB). Now, if we want to do processing on this file, How many mappers will be there?

→ As it is one image file, it will be considered as one record only

∴ All the data from different datanodes will come at one place & then processing will take place

∴ THERE WILL BE ONLY ONE MAPPER, BUT 8 DIFFERENT BLOCKS WILL BE THERE

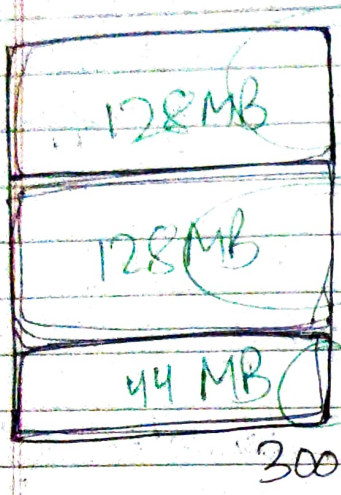
Q(2) Suppose we have a file of 300MB, record length 50mb & Block Size 128MB

(i) Number of records

No. of Records = $\frac{300}{50}$

= 6 records

(ii) No. of Blocks

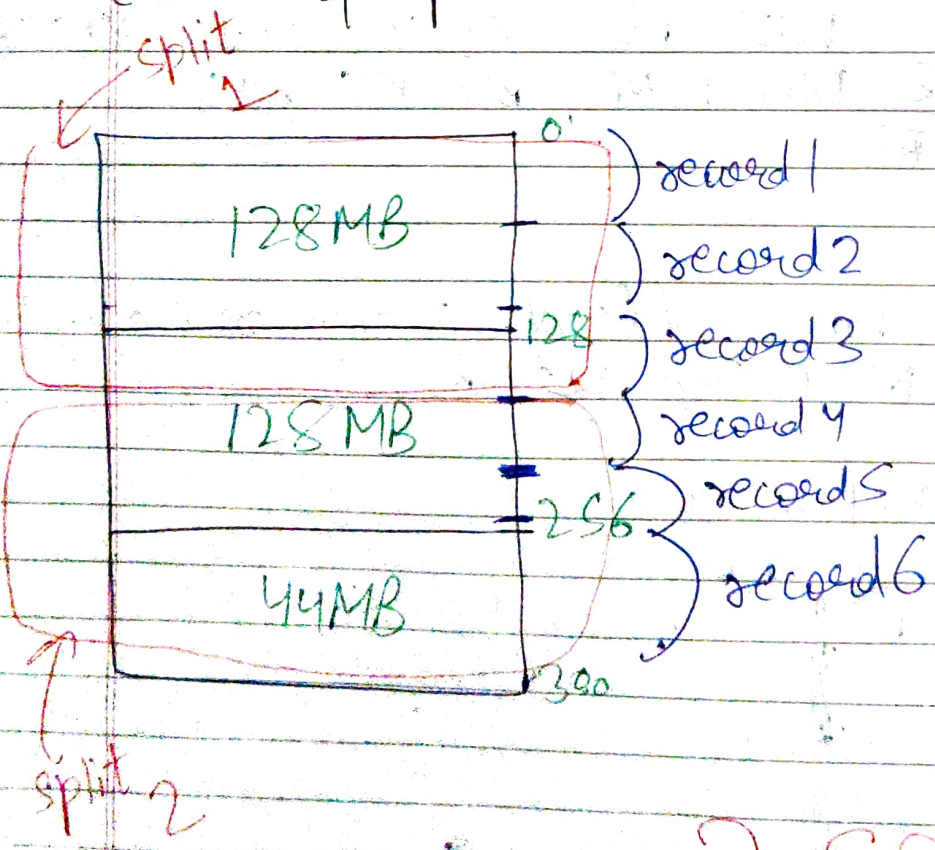


$$= \left\lceil \frac{\text{file size}}{\text{size of Block}} \right\rceil$$

$$= \left\lceil \frac{300}{128} \right\rceil = 3$$

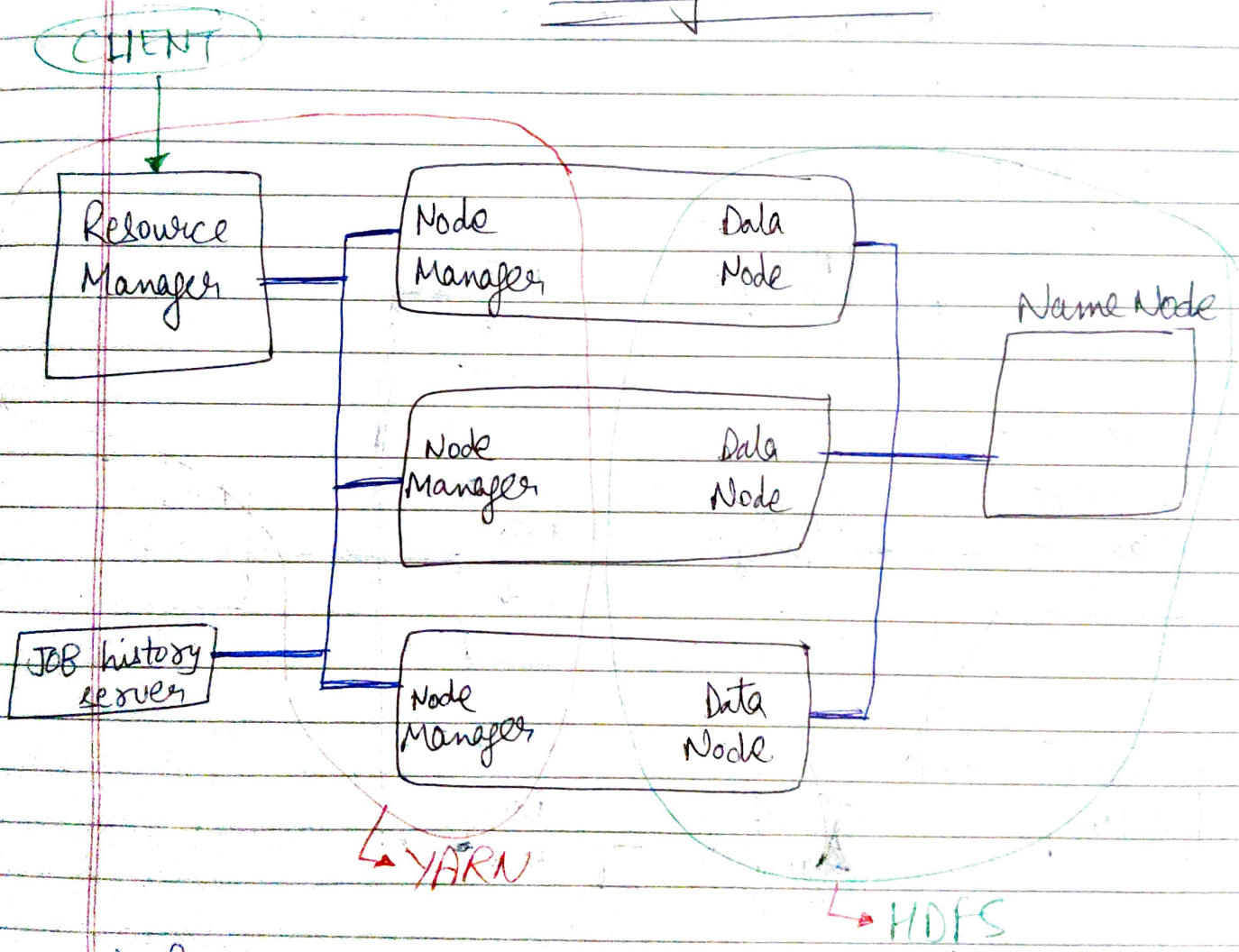
3 BLOCKS

(iii) No. of Splits



2 SPLITS

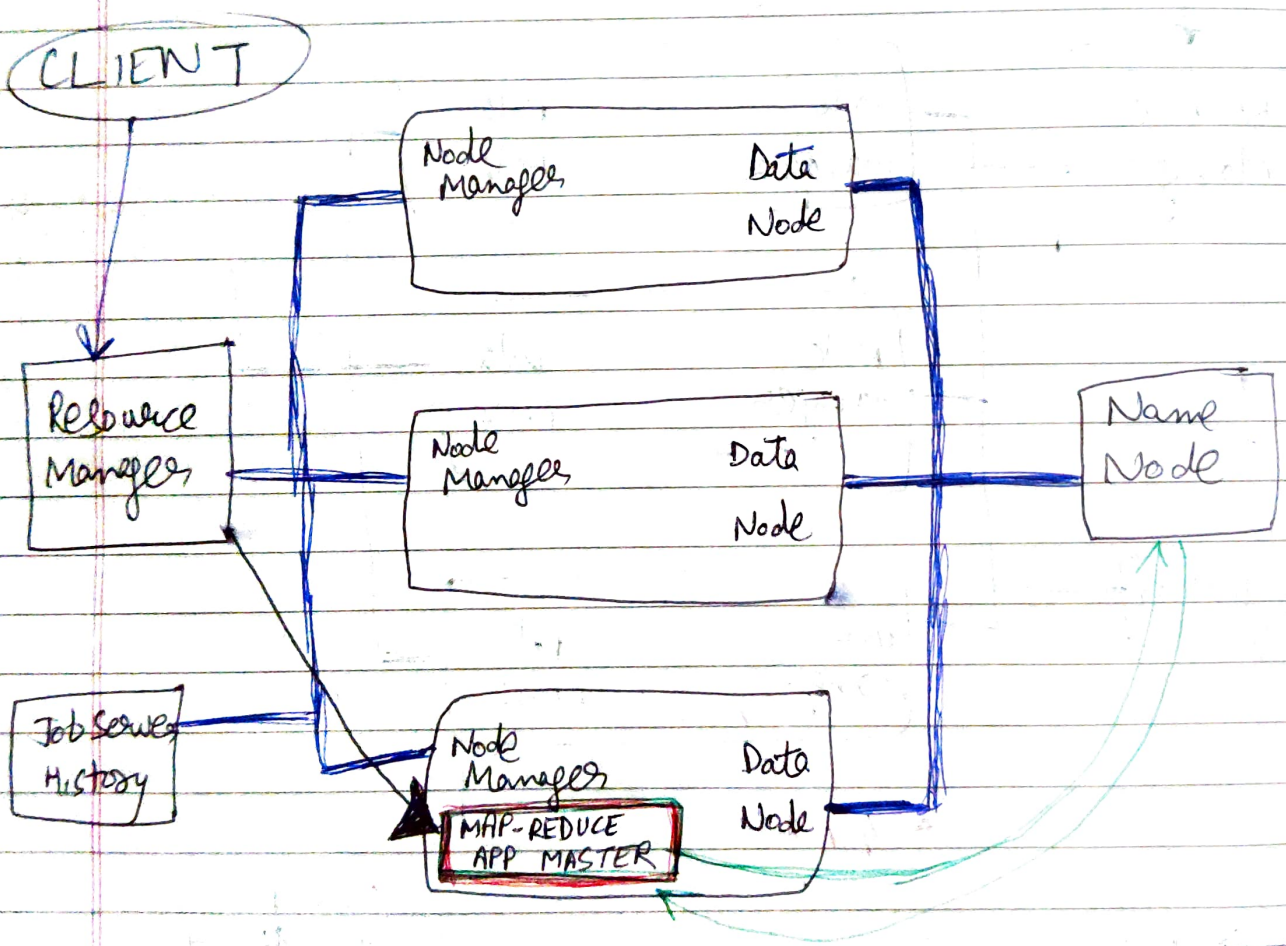
YARN — Yet ANOther Resource Negotiator



→ Resource Manager is the one with direct contact with Client
∴ Client will give the query to Resource Manager.

As resource manager will be receiving a lot of queries at a time, it will NOT process the queries himself

Not to overburden itself, Resource Manager will select one datanode (depending upon the amount of RAM & cores it has), and Make it the APP Master for the query (with lifecycle till the job runs)



Now, this app Master will firstly ask the NameNode to tell where the data required to be processed is present.

Then, after the APP MASTER has found the

LOCATION where blocks are stored, it will request the Resource Manager to create containers there (on the place where blocks are there

CONTAINERS IS BASICALLY ALLOCATION OF RESOURCES (like having 1GB Memory & 1 core of that datanode reserved for data processing)

CONTAINERS WILL BE MADE UP ON ONLY THOSE DATA NODES WHERE DATA IS PRESENT

Now, when these containers are created, it's the duty of NODE MANAGER to then process the data on the containers.

Evergreen
Page No.
Date : / /20

CHANGE BLOCK SIZE

To change block size, we Need to update the hdfs-site.xml.
HDFS-SITE.XML is found in the folder ⇒

/usr/lib/hadoop/etc/hadoop

Basically in the lib folder

We're supposed to add the property

```
<property>
    <name>dfs.blocksize</name>
    <value>27M</value>
</property>
```

Once the value is changed, the cluster restart is required for the change of effect, WHICH WILL BE APPLIED TO NEW FILES ONLY,

EXISTING FILES BLOCK SIZE DOES NOT CHANGE

NAME NODE SAFEMODE

SafeMode in Apache Hadoop is the maintainance state of NameNode, during which NameNode doesn't allow any modifications to the file system.

HDFS IS IN READ-ONLY MODE AND DOESN'T REPLICATE OR DELETE DATA BLOCKS.

What if all the nodes that holds certain blocks of data are down? HDFS has a safety mechanism in which a certain percentage of unavailable blocks makes the HDFS go in a safe Mode.

To know the status of NameNode:

```
hadoop dfsadmin -safemode get
```

To manually enter safemode:

```
hadoop dfsadmin -safemode enter
```

To Exit Safemode

```
hadoop dfsadmin -safemode leave
```