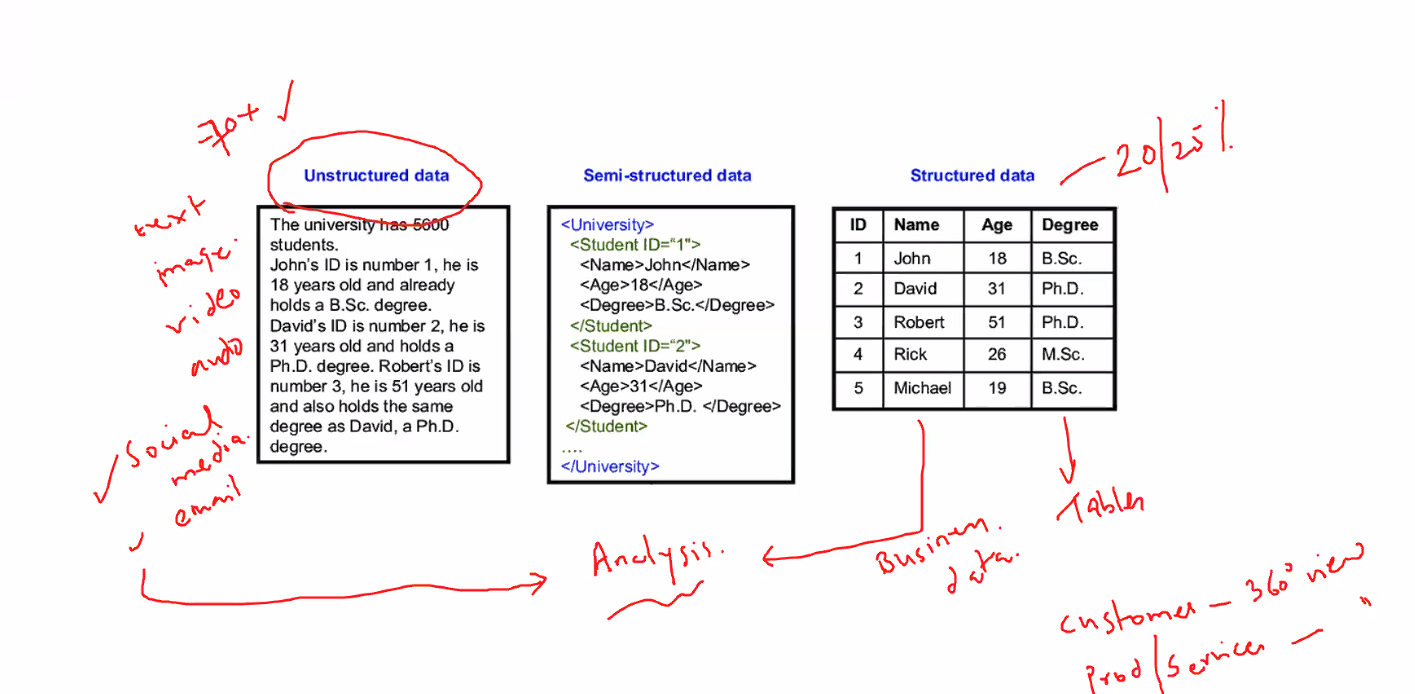
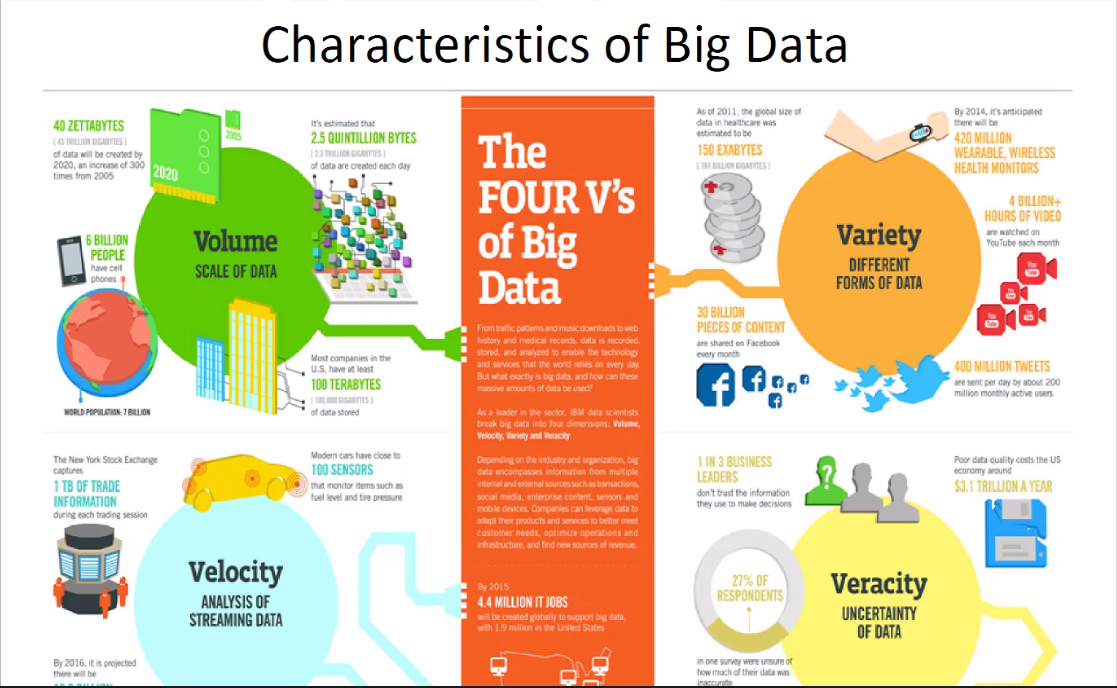
**Big Data**

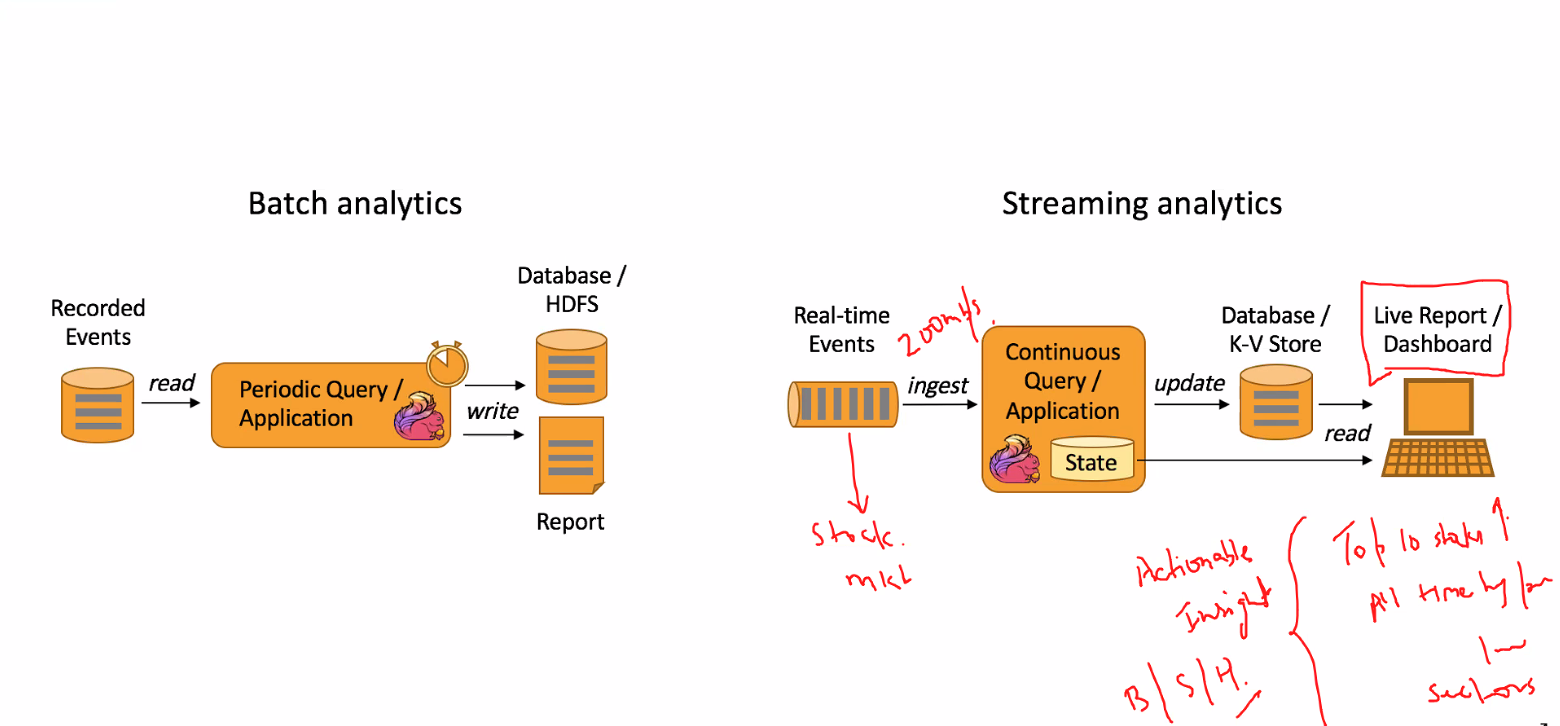
pwd

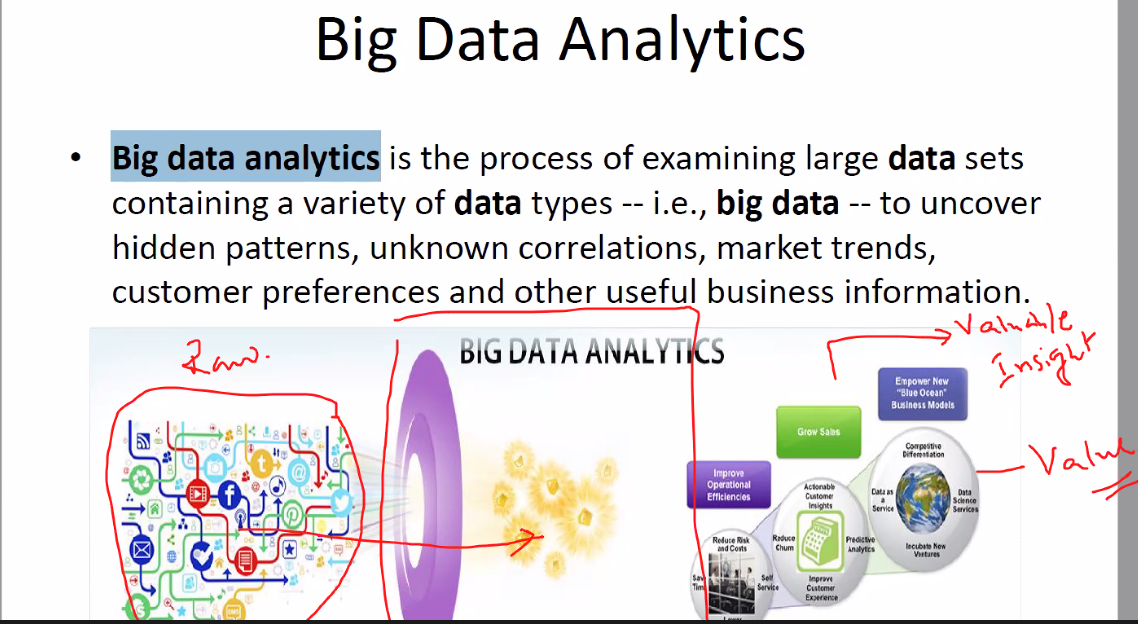
ls

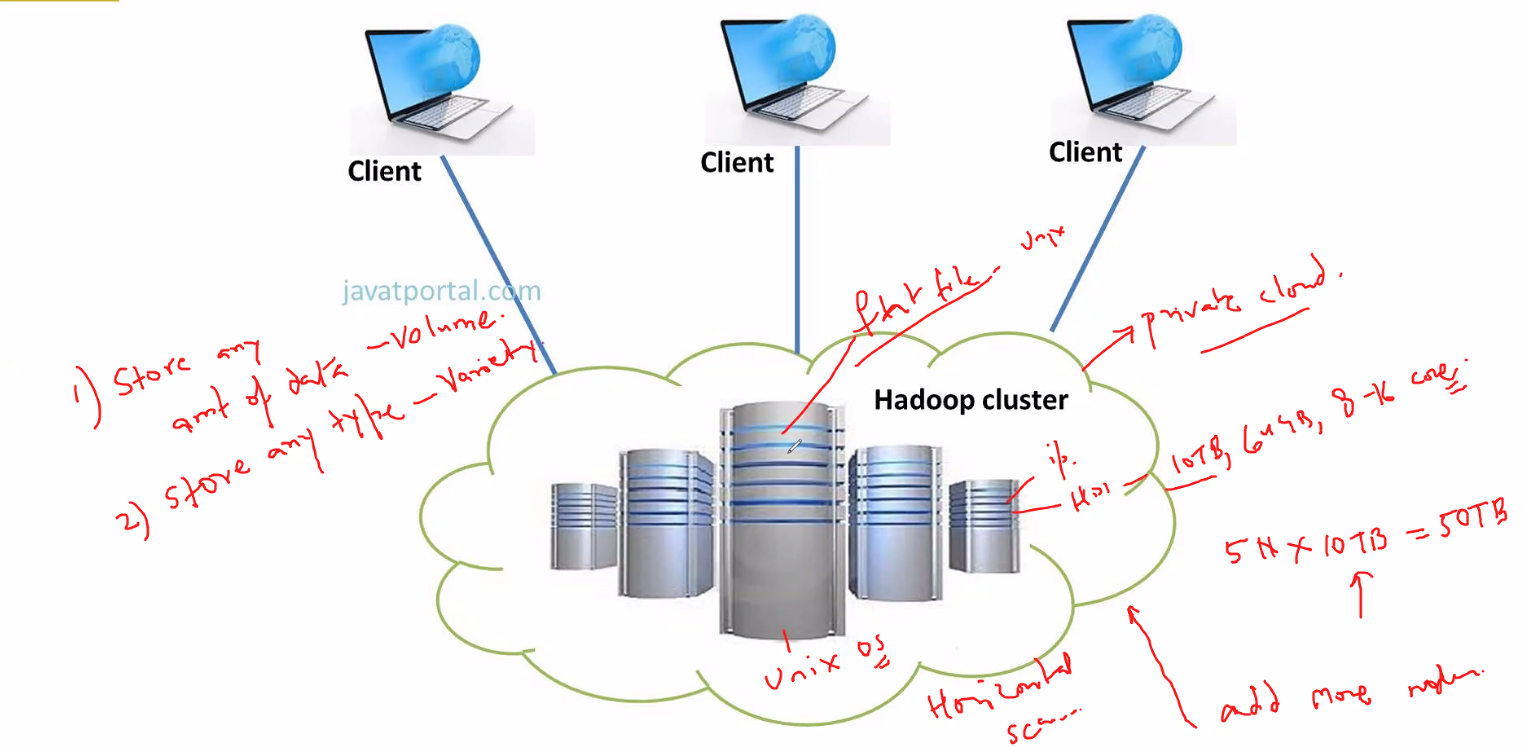




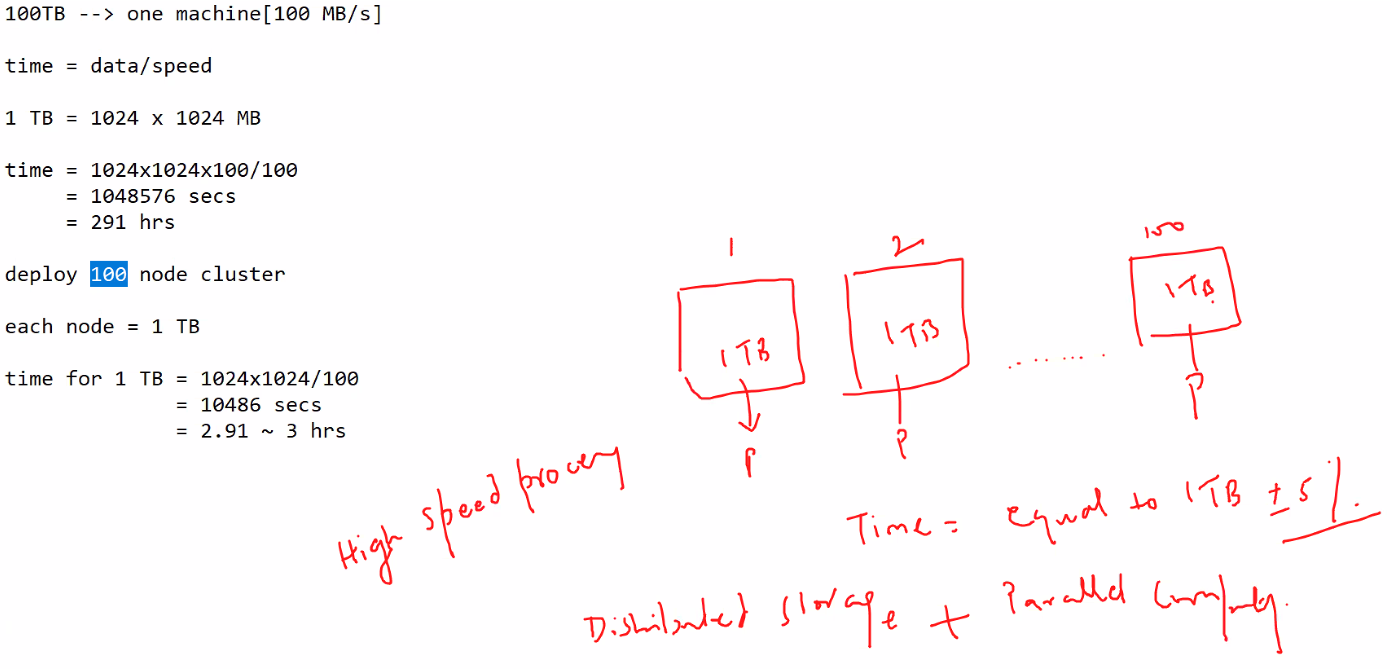
Veracity: Cleansing, preprocessing, validating

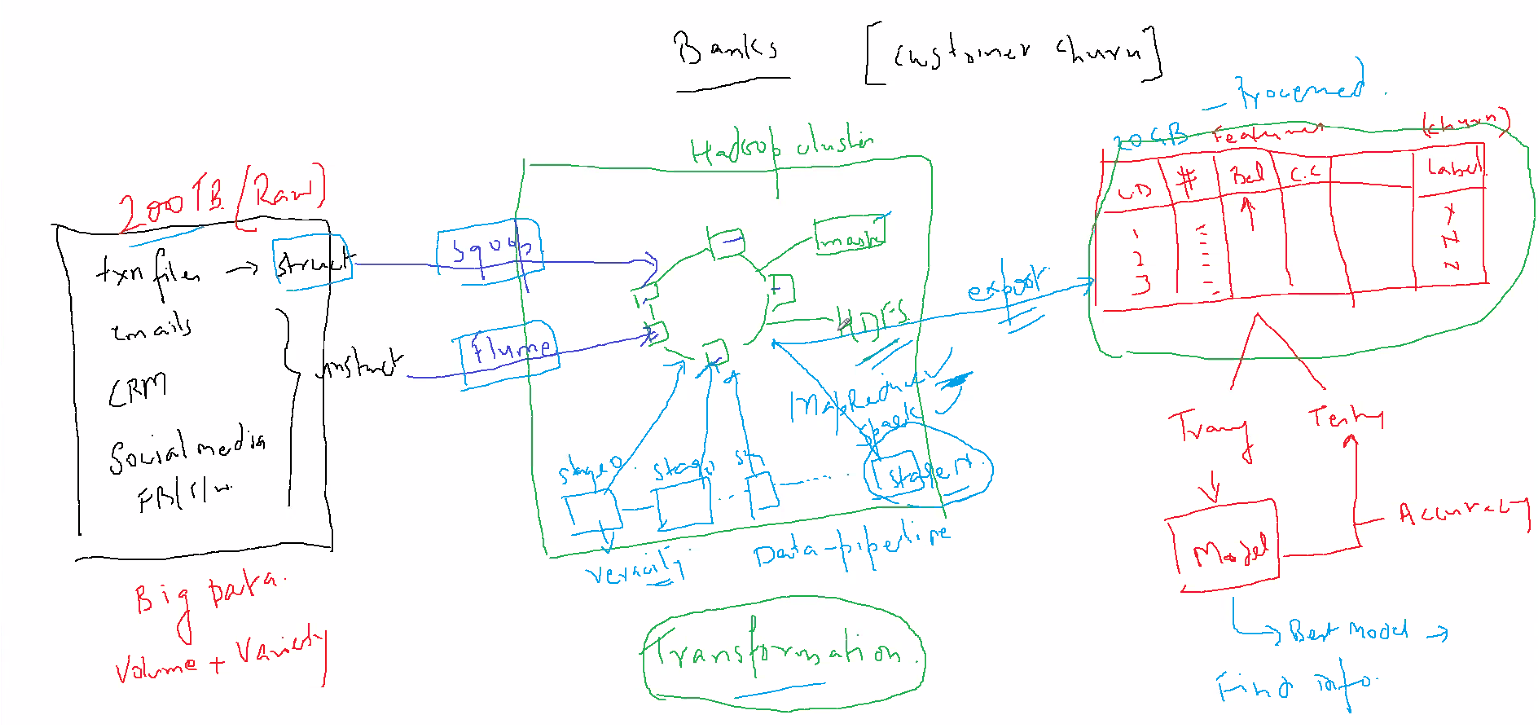






Hadoop -> Transform Unstructured data to structured processed data (processed in distributed storage and parallel computing is used)





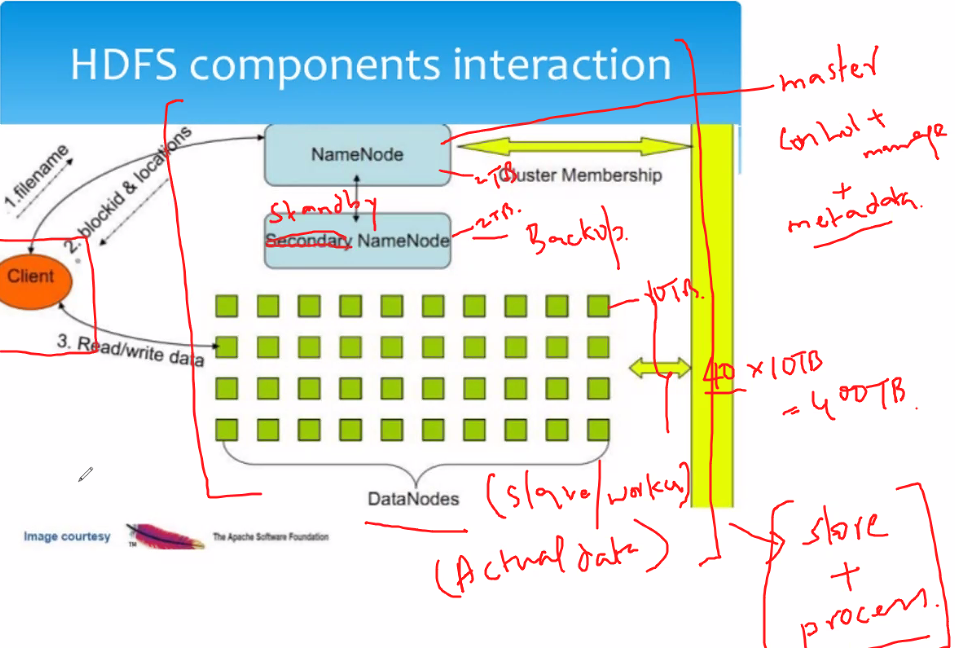
Sqoop => pull structured data faster

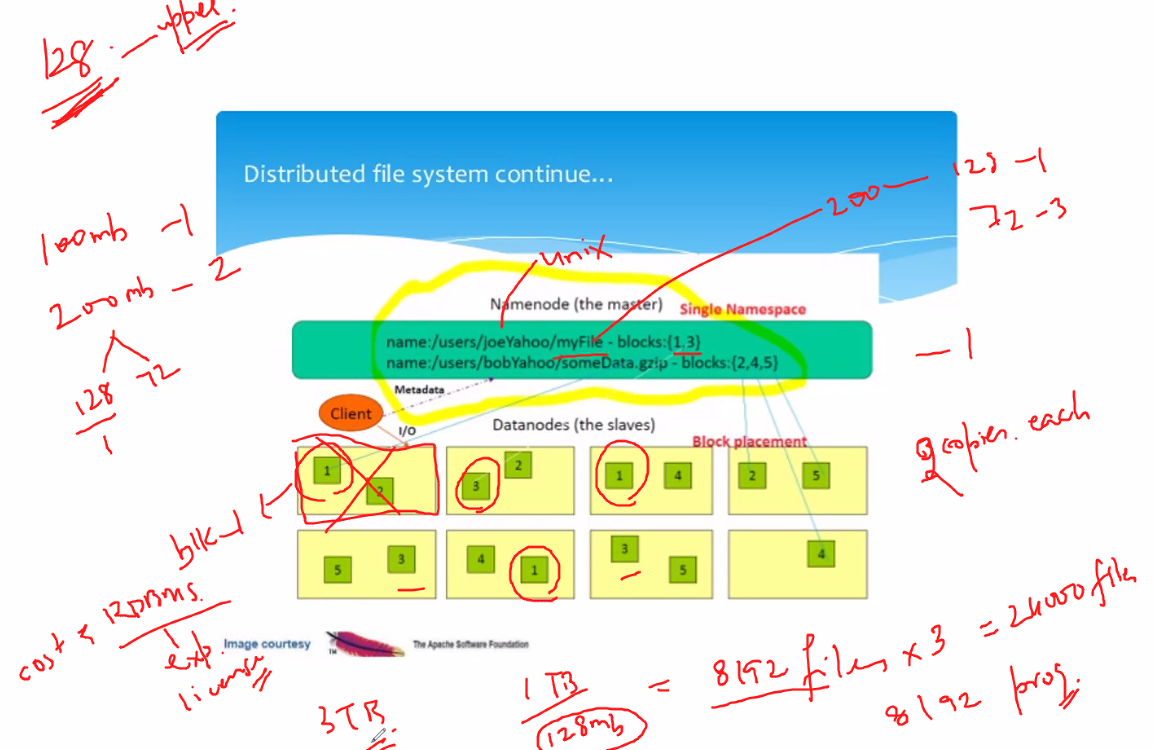
Flume => pull unstructured data faster

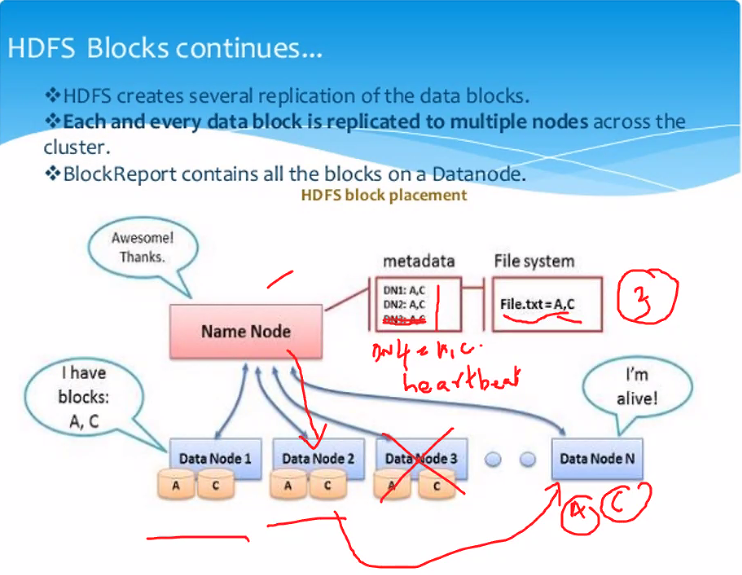
MapReduce/Spark => Algorithms to process distributed data

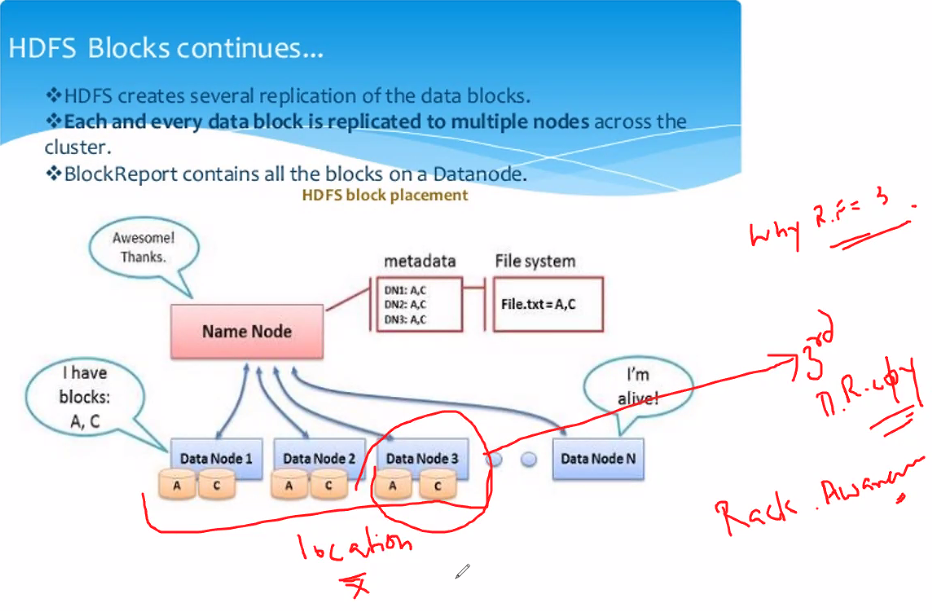
Yarn MapReduce written in Java

Hive => SQL query to Java





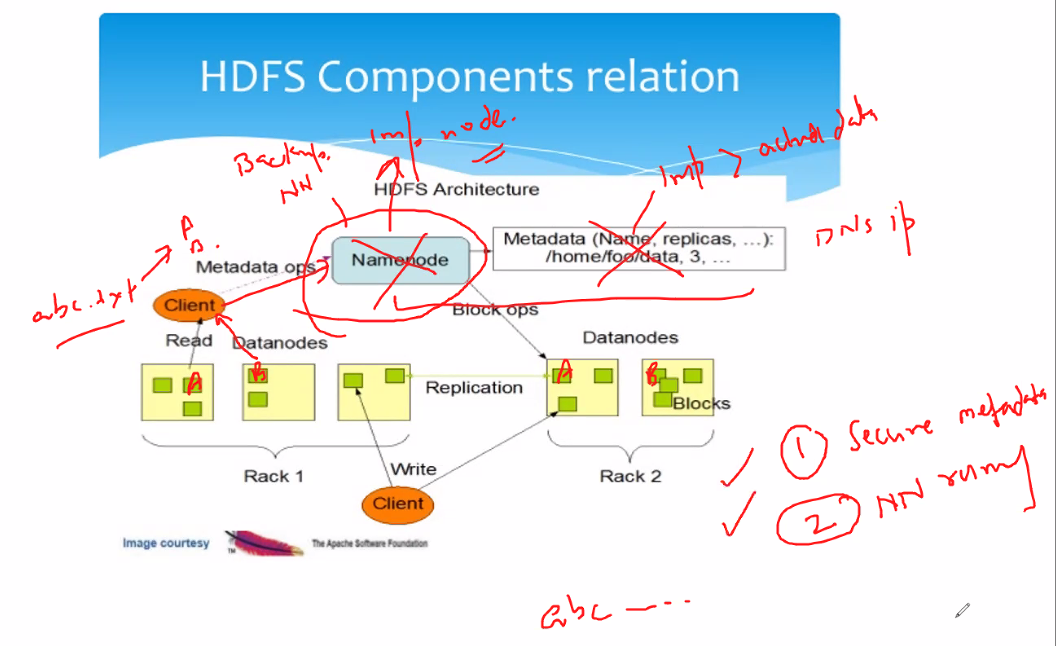


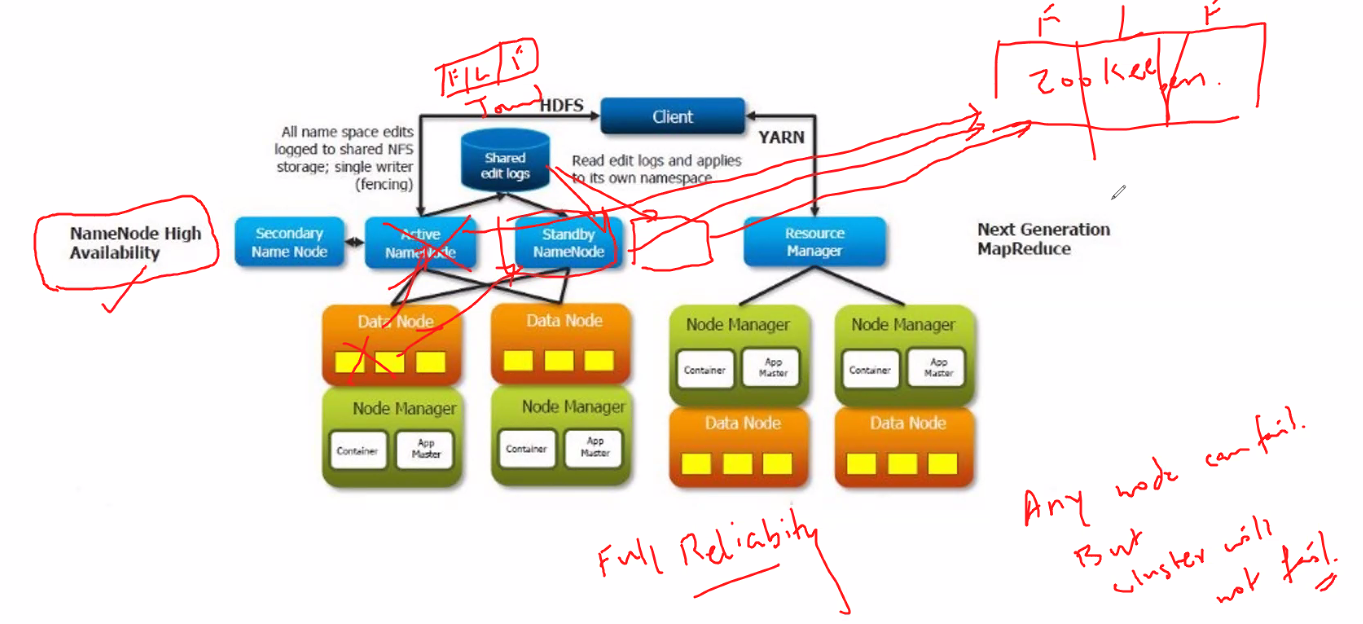


3rd copy => Different rack/location

DR => Disaster recovery

RF => Replication factor





**Linux commands**

lsb\_release –a

pwd => current working directory

machinename@hostname

~ denotes home/user directory

cd /

pwd

ls

ls –l => current directory files and folders

d represents directory

- represents files

cd home => change present directory to home

ls

ls -l

cd cloudera

pwd

mkdir folder1

ls

ls -l

cd folder1

ls

gedit file1.txt

ls -l

user, group & others

r – read => 4

w – write => 2

x – execute => 1

chmod 764 file1.txt

ls -l

chmod 724 file1.txt

ls -l

chmod 777 file1.txt

ls -l

chmod 700 file.txt

ls -l

7 = 4+2+1 (rwx)

user – group – others (permissions)

gedit (graphic edit system)

gedit file1.txt

ls -l

chmod 500 file1.txt

ls -l

gedit file1.txt

chmod 700 file1.txt

gedit file1.txt

cat file1.txt (read file)

chmod 300 file1.txt

ls -l

cat file1.txt

chmod 700 file1.txt

cat file1.txt

nano file1 (create new file without extension different from file1.txt)

Ctrl+X to exit

Enter Y for yes to save

ls -l

flat file system (no data type)

file names are case sensitive

nano File1

ls -l

cat File1.txt

pwd

cd ..

mkdir folder2

cp folder1/file1.txt folder2

cd folder2

ls -l

cd ..

mv folder1/file1 folder2

cd folder1

ls -l

cd ..

ls -l

cd folder2

ls –l

rm file1

ls –l

cd..

rm –R folder2

cd folder2

hadoop fs –ls /

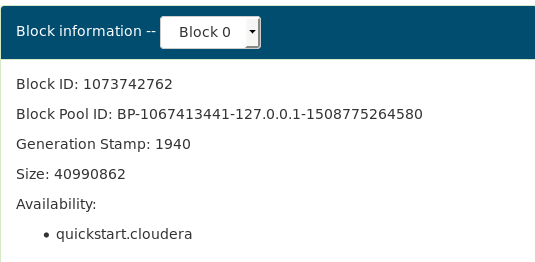
hadoop fs –mkdir /user/cloudera/imart

ls –l NYSE.csv

ls –l eclipse.gz

hadoop fs –put NYSE.csv /user/cloudera/imart

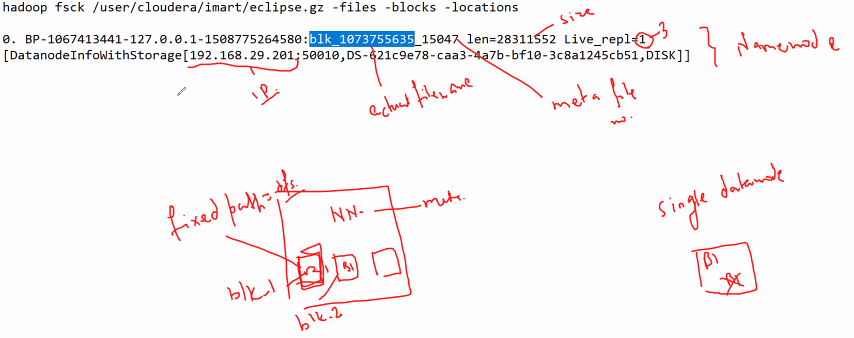
hadoop fs –put eclipse.gz /user/cloudera/imart



Block pool id => Name node(root) id

Name node contains only metadata, data nodes consist of data in blocks

hadoop fsck /user/cloudera/imart/eclipse.gz –files –blocks -locations



cd /var/lib/hadoop-hdfs/cache/hdfs/dfs/

sudo su (get root access to access data dir)

cd data

cd current/

cd blockpoolid

cd current/

cd finalized/

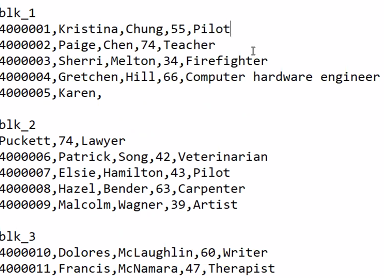
cd subdir0/

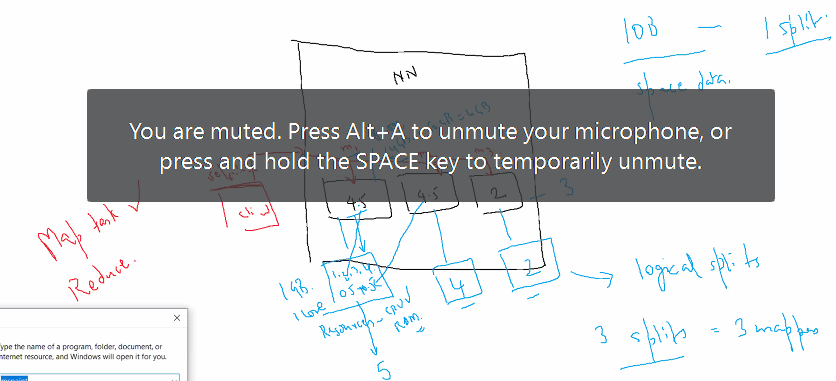
find -name blk\_blockid

cd subdir(founddir)

gedit blk\_blockid

Store any kind of info due to physical split in hadoop.



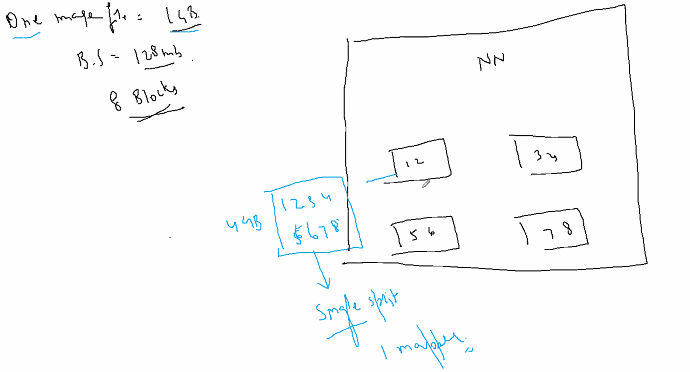


Containers => memory allocation, logical allocation of resources, check resource availability

IO operation, edit access open etc => RAM/CPU processors

Mapper => process one record at a time, does not take half or incomplete records borrows other half records and then process, at runtime logical/input split is created.

Save store file => disk



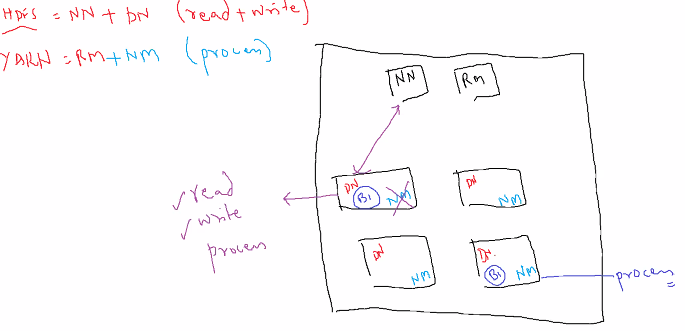
One split because image data cannot be split logically and is complete as one file as a whole.

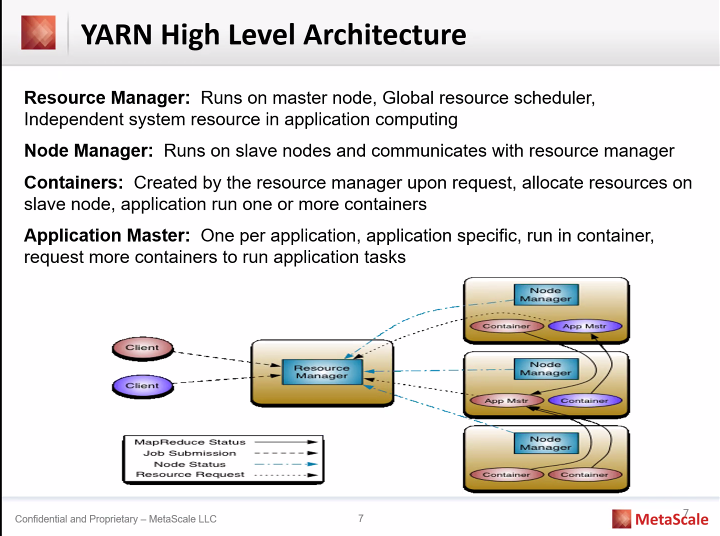
YARN => Yet Another Resource negotiator, takes code to where the data resides.

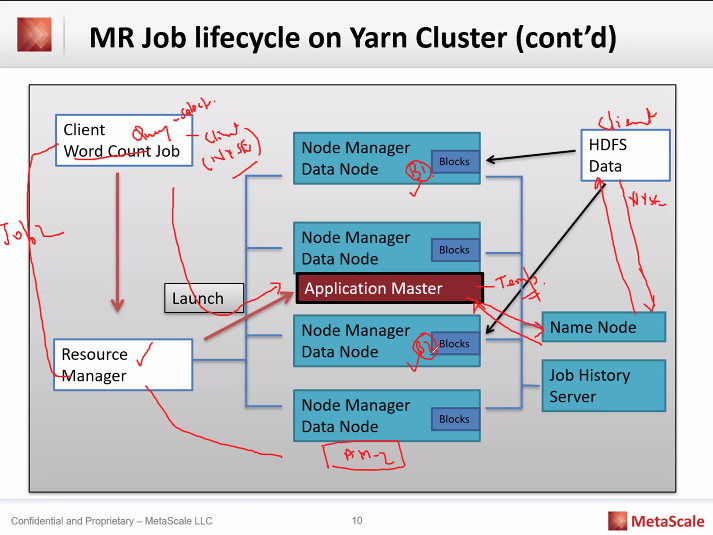
Master – Resource manager (1 per cluster) Request for processing such as queries goes to rm.

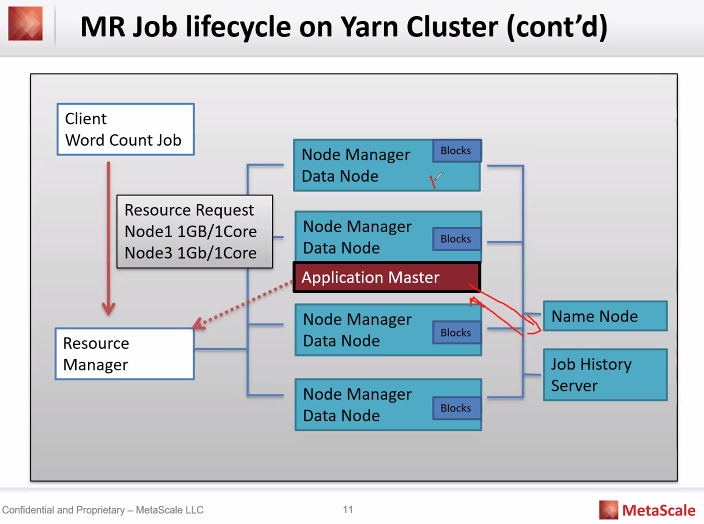
Request for read/write goes to name node of hdfs.

Slave – node manager (1 per data node) for processing data

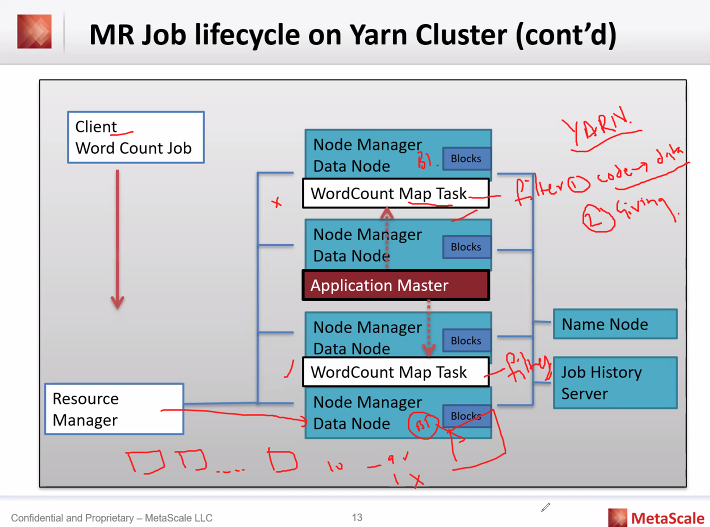








Application master for temporary processing, coordinating, managing flow of the process (check for resources getting metadata from name node, sharing mapper code to node manager).

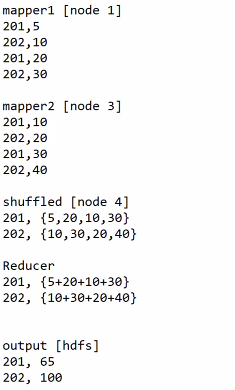


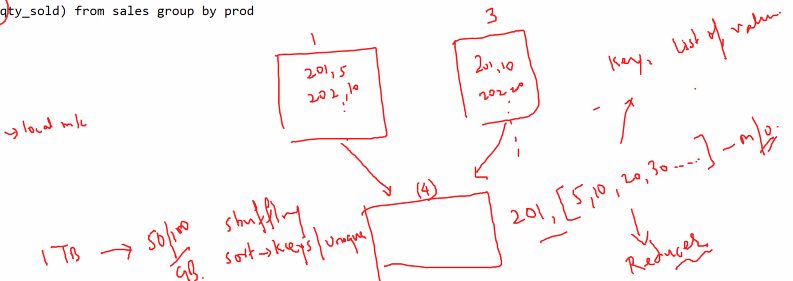
MapReduce:

Mapper: program to filter the data => key, value pair

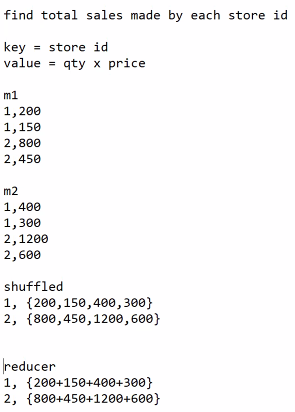
Reducer: program to process the mapper data.







Mapper output: key, [list of values] (Reducer input)

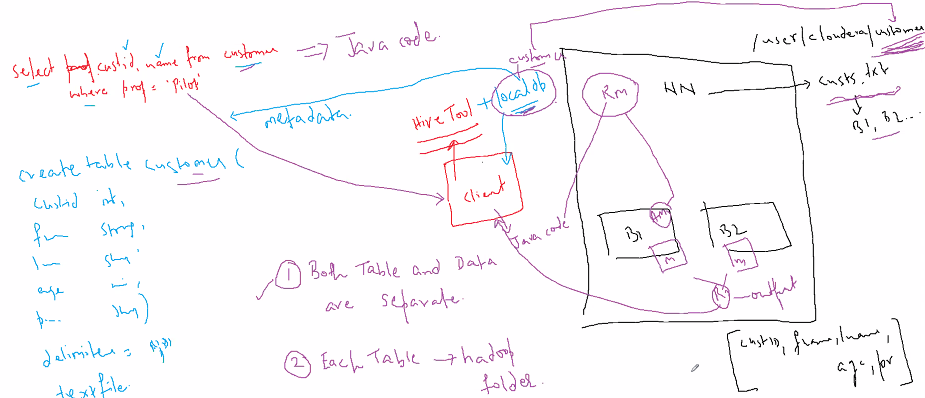




For value creation or derivation of value using multiple cols mapper can be used.

Physical split => logical split => Yarn + MapReduce

Hive => converts sql queries internally into java code.



Database is created in local database and a .db file is created under /user/hive/warehouse.

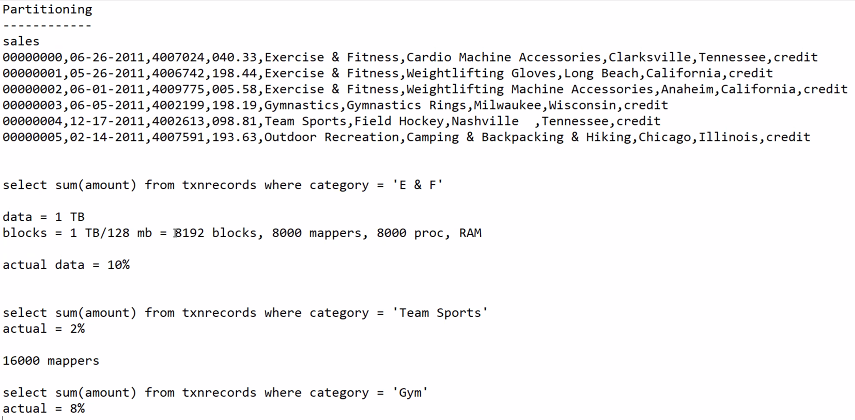
hadoop fs –put NYSE.csv /user/hive/warehouse/imarticus.db/nyse for mapping data to hive databse folder.

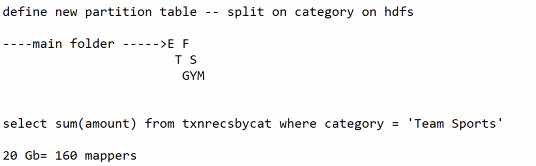
/user/cloudera/hive/results1

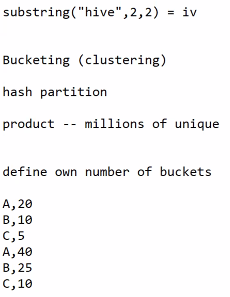
Airline analysis project.

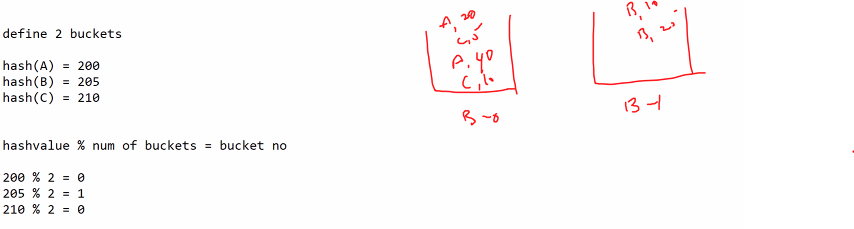
Optimization in hive

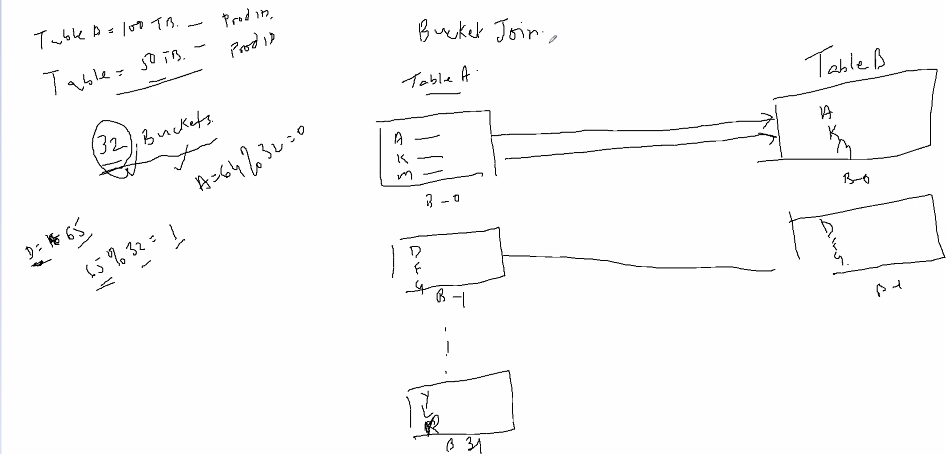
Make the process faster than actual speed, less memory consumption.





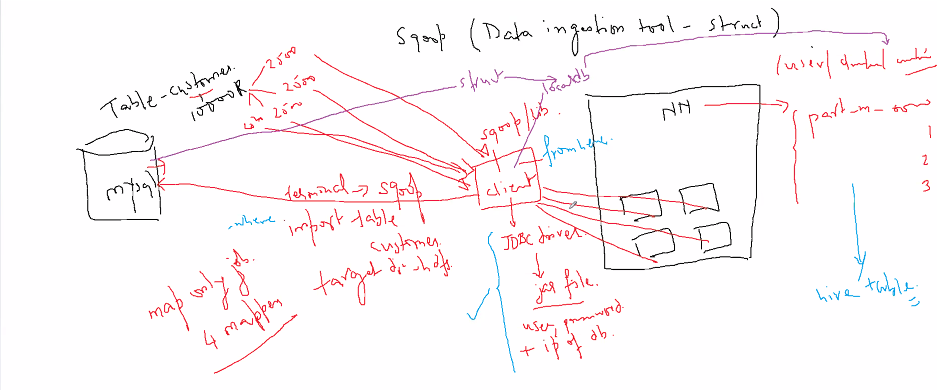




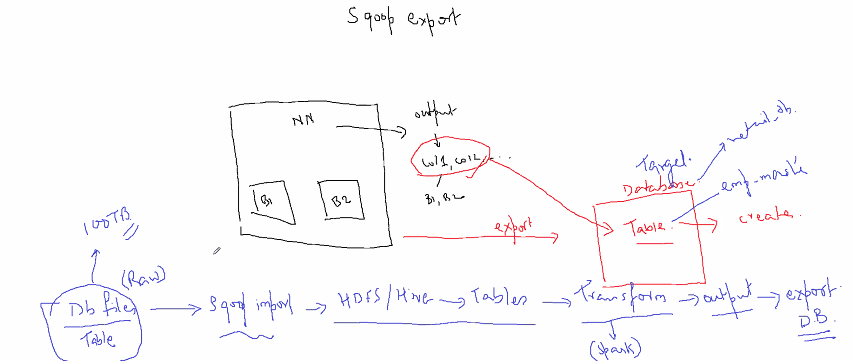




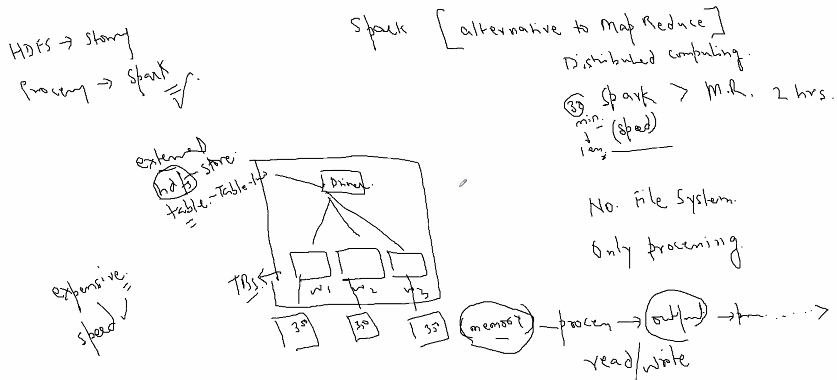
Sqoop:



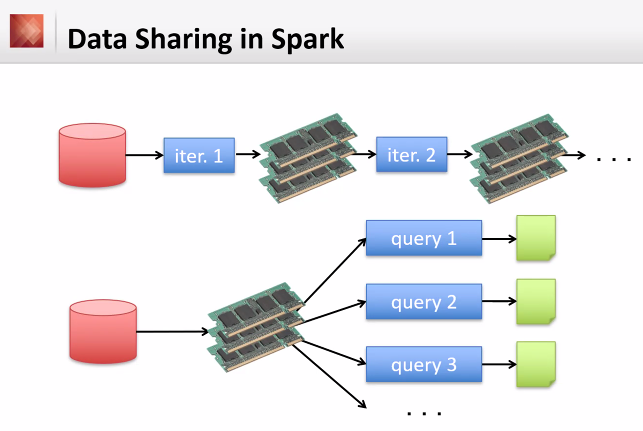
Load data from local database splitting it by blocks, then map the data to the hive database.

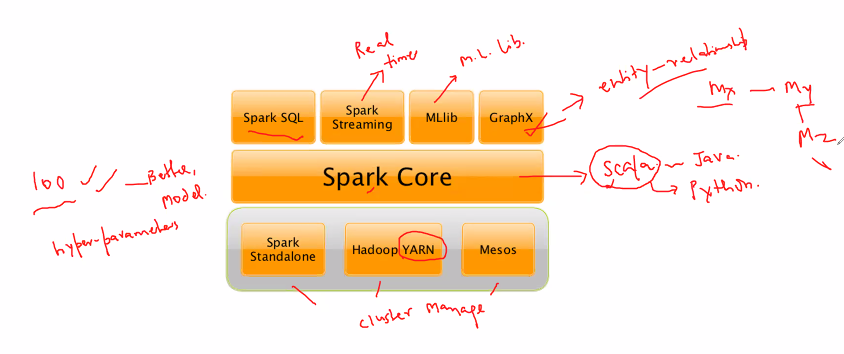


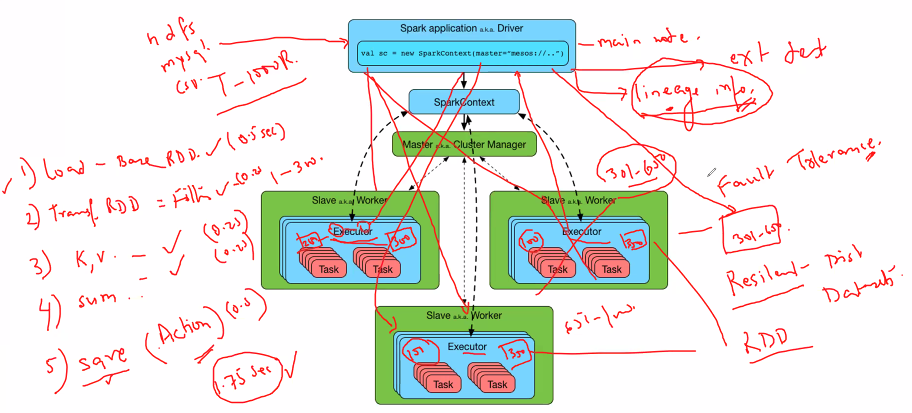
Spark:



Process or write data to memory, limitation very large amount of data expensive.







Base RDD to store data, intermediate results in transform RDDs. RDD => Memory

Lineage info stores which records/data belongs to which worker node. Relaunch of data on new worker node on fault tolerance using lineage info. Occupy memory for only while processing & process starts only when action is executed.

