Job Post ✓

Brand Building ✓

Hadoop

2 things
1. lots of data
2. Cluster

Hadoop is an open source framework designed to handle massive amounts of data in a scalable and distributed way

⊖ with earlier soln
→ storage limits
→ Processing issue
→ expensive to add more memory

2000

Doug Cutting

GFS → distributed storage → HDFS

MapReduce → distributed data processing → MR

Hadoop can store and process data across many cheap, commodity hardware m/c working together

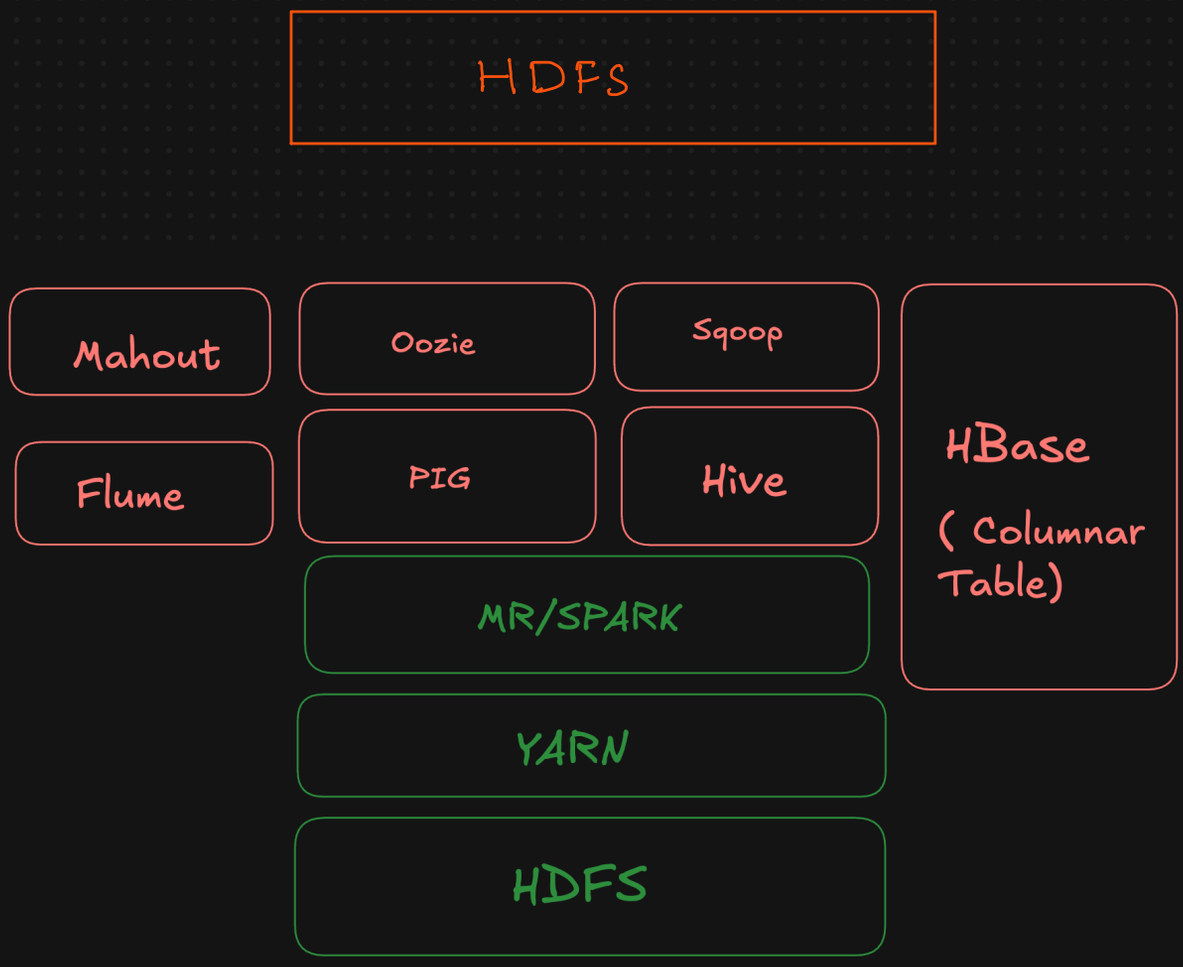
↙ Massive data storage

↘ faster parallel processing

Properties

1. Scalability
2. Fault Tolerance
3. Distributed Processing
4. Cost-effectiveness

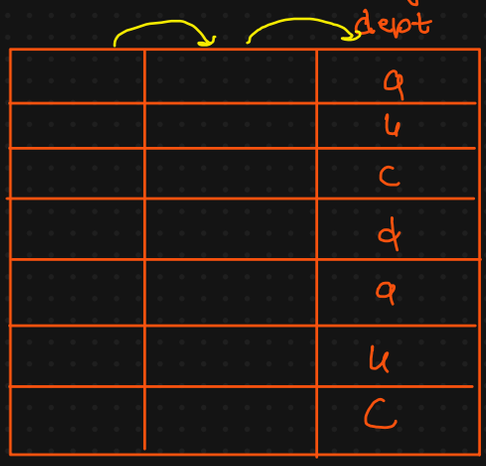
Hadoop Ecosystem / framework



2 imp things

1. Loosely coupled framework :
2. Integration

SQL (write heavy)



+

Database

OLTP

Columnar

Seek time

Data Warehouse

OLAP

Task for students ⇒ Master Basic Linux Command

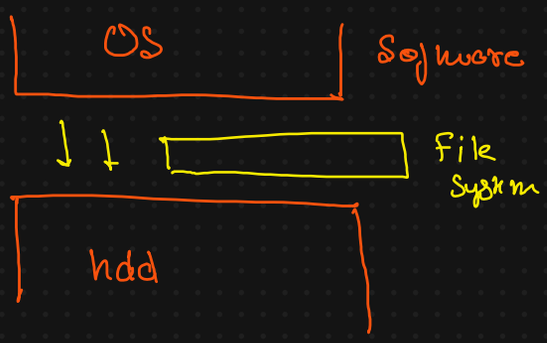
HDFS

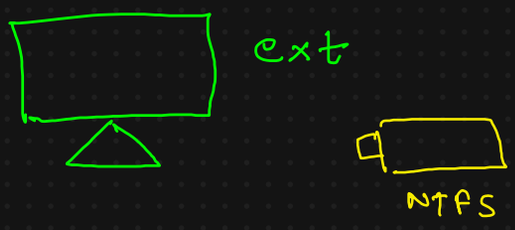
hadoop distributed file system

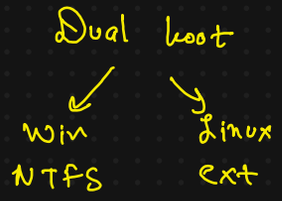
GFS

1. What is a file system?

Windows	NTFS / FAT32
Apple	APFS
Linux	ext







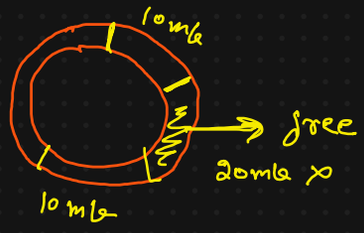
2. What is a block?

Smallest unit of data storage in a file system

Block size ⇒ depends on file system

NTFS → 4kb
Hadoop/HDFS → 128 mb

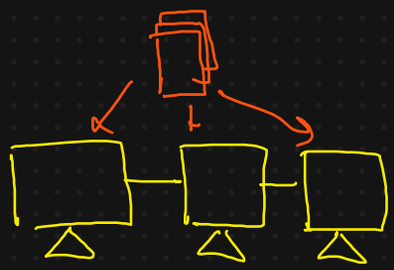
10 kb → 3 block



50 mb

3. Types of file system.

Standalone



Distributed

4. Cluster and Node ?

5. Process & Daemon Process ?

hadoop is a process

JP1
JP2 } HDFS
JP3
JP4
JP5 } MR

6. Metadata

data about data

Image

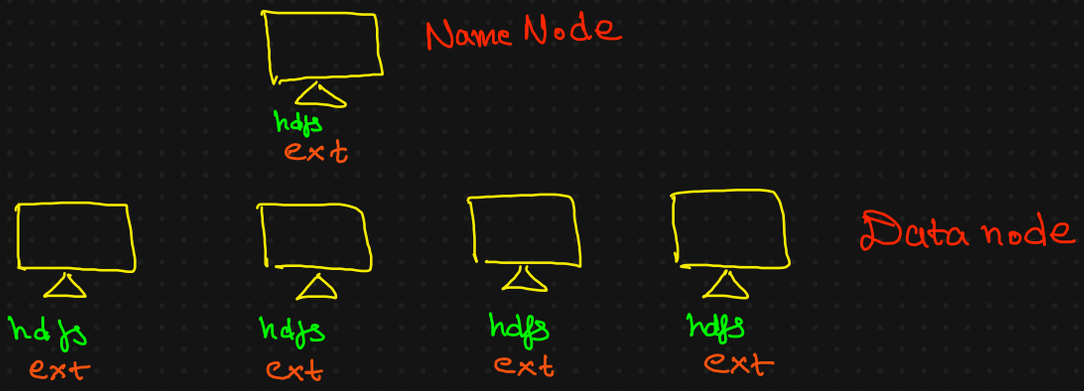
Size
Created at
Pixels
dimensions

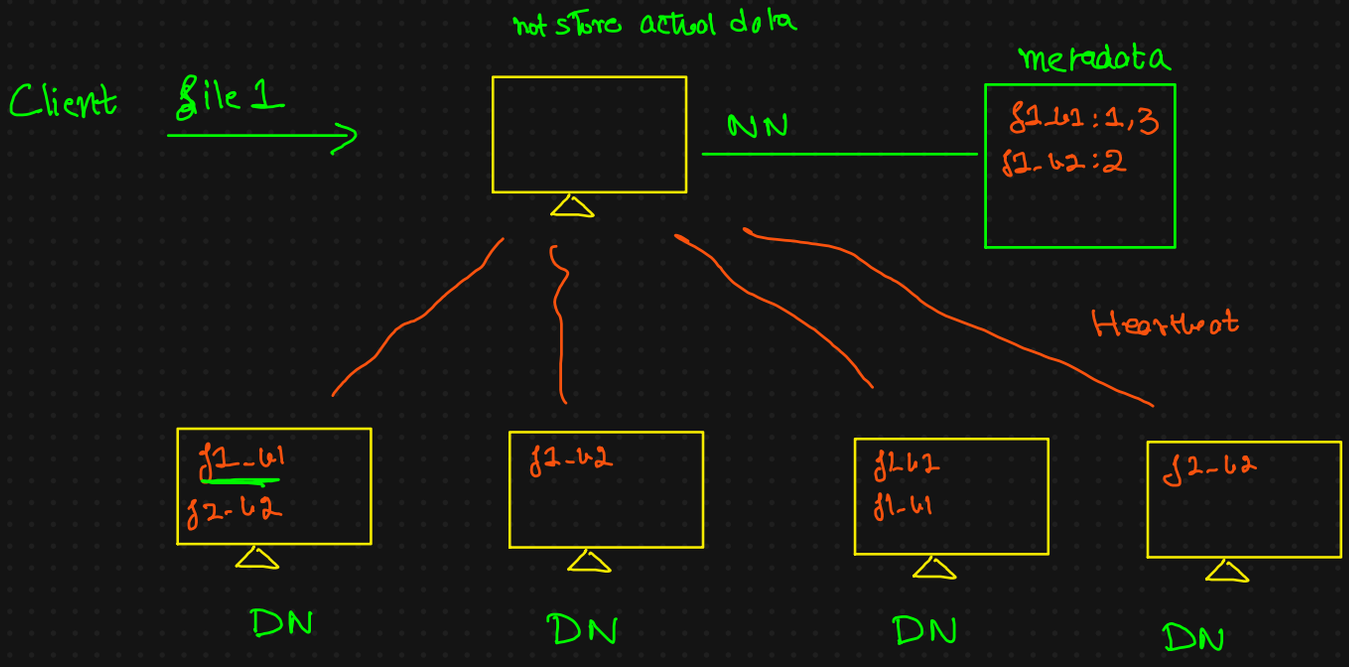
metadata

7. Replication

making copies of data

HDFS

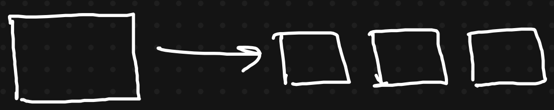




replication factor = 3

lgb

Blocks



128 mb by default size

256 mb → 2 blocks

Bazooka / Gun to kill an ant

Key Points

1. efficient for large storage file
2. Distributed data

1kb
64mb
64kb
127mb
⇒ 1 block

1kb * 10 → 10 blocks

Q: Can we inc or dec. block size?

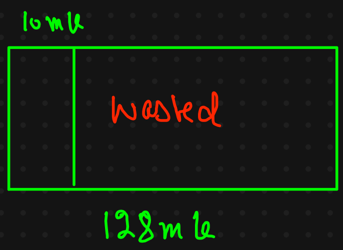
Yes, default is 128mb

1. decrease → 10mb

↑↑ parallelism but also ↑↑ metadata

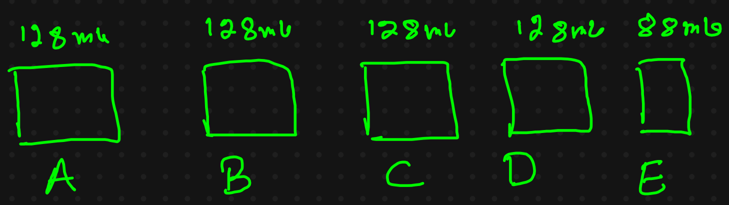
index metadata — Chapters 10 pages

metadata ↑↑ — 1 pages



2. increase : parallelism

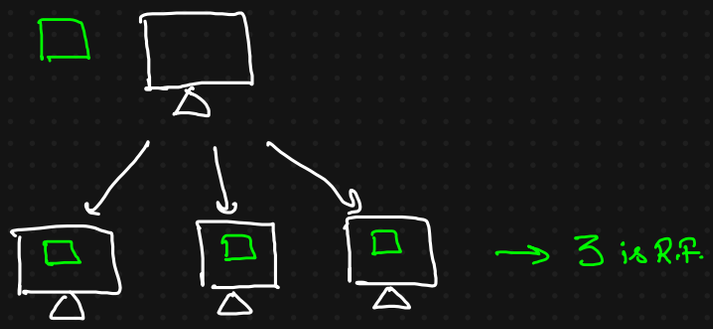
600mb file



900mb file

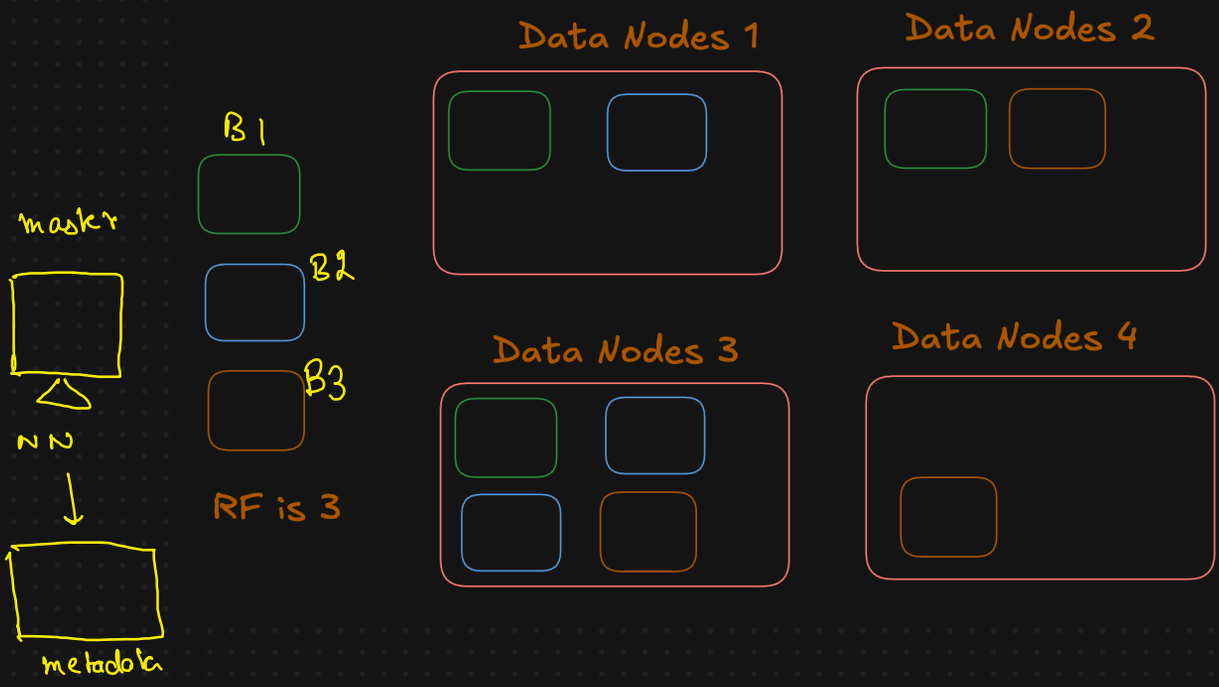
129mb file and 64mb block size → 3 blocks

Data Node failures

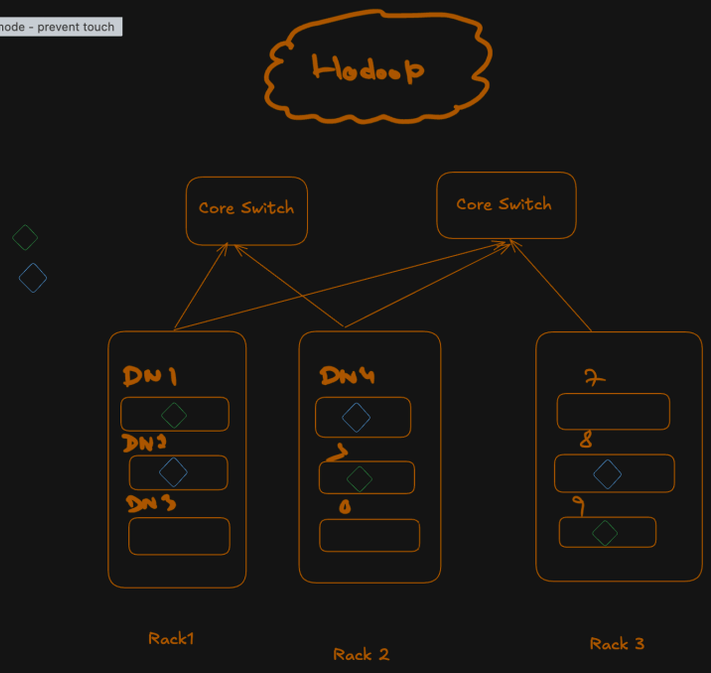


1. Replication

R.F. = 3 ⇒ 1 original
2 copies



Pen mode - prevent touch



1. Fault tolerance
2. Optimized Network Traffic

Q:

1. Temporary failure
2. Permanent failure

   Name node failure

   +

   hdfs Practical
3. YARN + MR

Linux + hdfs) Quick Guide

Temporary failures in Data Node

- → Network outage
- → Software issue
- → Reboot for maintenance

Process during this failure

1. Detection

   temporary failure — heartbeat — upto 10.5 min

2. Replication

   Name node identifies the blocks stored on unavailable DN

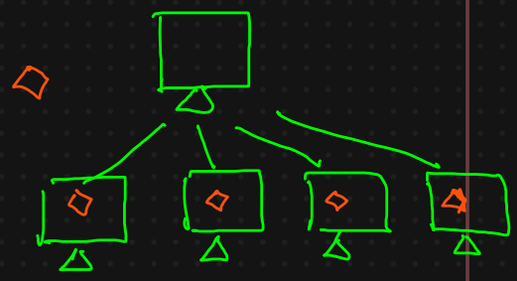
   marks them as under-replicated. → updating metadata

3. When the DN recovers

   the DN sends a Block report

   extra replica is deleted.

   The metadata is updated dynamically to ensure consistency.



The client will have no idea about failure

Permanent failure

→ Hardware failure

→ Disk corruption

→ Decommissioning the node

> 10 min heartbeat

Process during Permanent failure

1. Detection

   NN will declare DN as dead due to missing heartbeats

2. Replication

- re-replicate the lost blocks to other data nodes.
- NN updates its metadata to point to new copies & remove reference from previous failed DN

3. If failed Machine comes back:

   the returning DN is not trusted & its blocks are ignored and cleaned up.

Aspect	Temporary Failure	Permanent Failure
Failure Cause	Network issues, software crashes, or reboots.	Hardware failures, disk corruption, or decommissioning.
Detection	DataNode stops sending heartbeat signals; marked as temporarily unavailable.	DataNode stops sending heartbeat signals for an extended period; marked as dead.
Replication Trigger	Blocks on the failed DataNode are marked as "under-replicated," and new replicas are created.	Blocks on the failed DataNode are treated as lost, and new replicas are created.
Returning DataNode Behavior	Sends a block report to the NameNode. Excess replicas are deleted to restore the replication factor.	Returning DataNode is not trusted. Its blocks are treated as stale and eventually cleaned up.
Metadata Updates	Metadata dynamically updates to include new replicas and remove excess copies.	Metadata permanently removes blocks from the failed DataNode and updates mappings for new replicas.
Library Analogy	A locked bookshelf temporarily restricts access. Extra copies are reconciled when it becomes available.	A destroyed bookshelf is permanently removed from the catalog, and its books are replaced elsewhere.

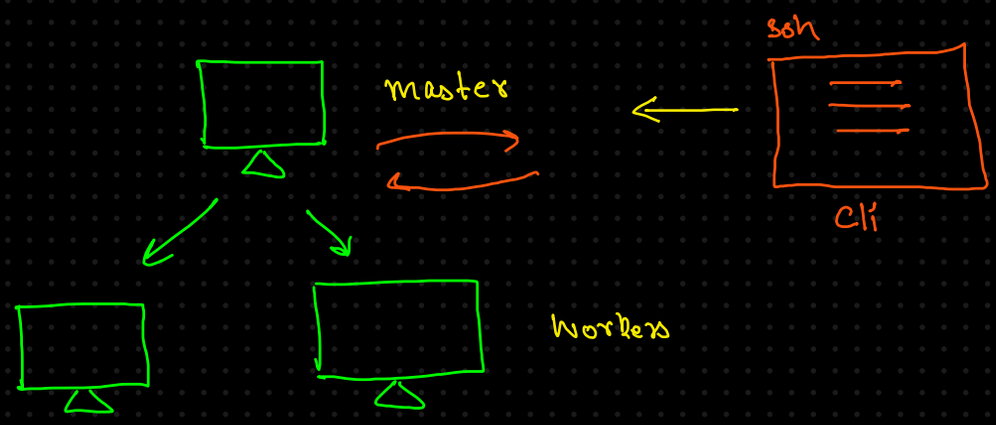
Few Takeaways

1. HDFS handles failures gracefully:
2. Metadata is the backbone of hdfs.
3. Replication factor is strictly maintained.

1. Google cloud → $300 credits + 3 month trial

⇒ Credit/debit
UPI
Netbanking

2. Dataproc → managed Hadoop clusters
   ↳ enable the dataproc API



software

data engineering

Name node failures

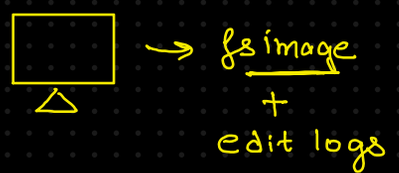
SPOF → Single point of failure

NN is the most critical component of your HDFS

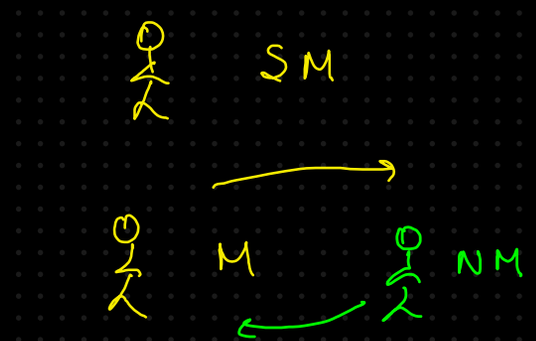
Q: What is the difference between a secondary NN & a Standby NN?

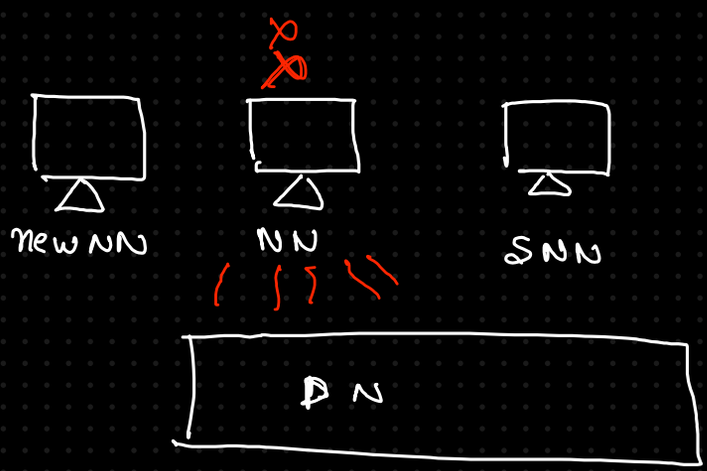
Secondary Name node

It is a helper node that checkpoints the primary Name node.



When our main NN dies, SNN will have all information to bring up a new NN upto the point where it can behave like previous NN. It is not a failover mechanism, it cannot take role of NN.





Some downtime will be there till the new NN has been updated via SNN.

Librarian ⇒ helper

Standby NN

Standby NN takes role of active as soon as Active Name Node gets killed

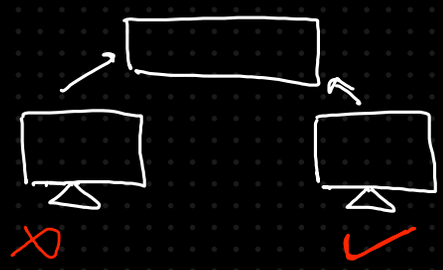
→ hot backup for NN
→ constant connection
→ makes copy of namespace image in memory

1. High Availability

1. Standby NN will maintain a synchronized copy of metadata in memory.
2. Synchronization happens via shared edits: your ANN and Standby NN has access to a shared directory (Journal Node)

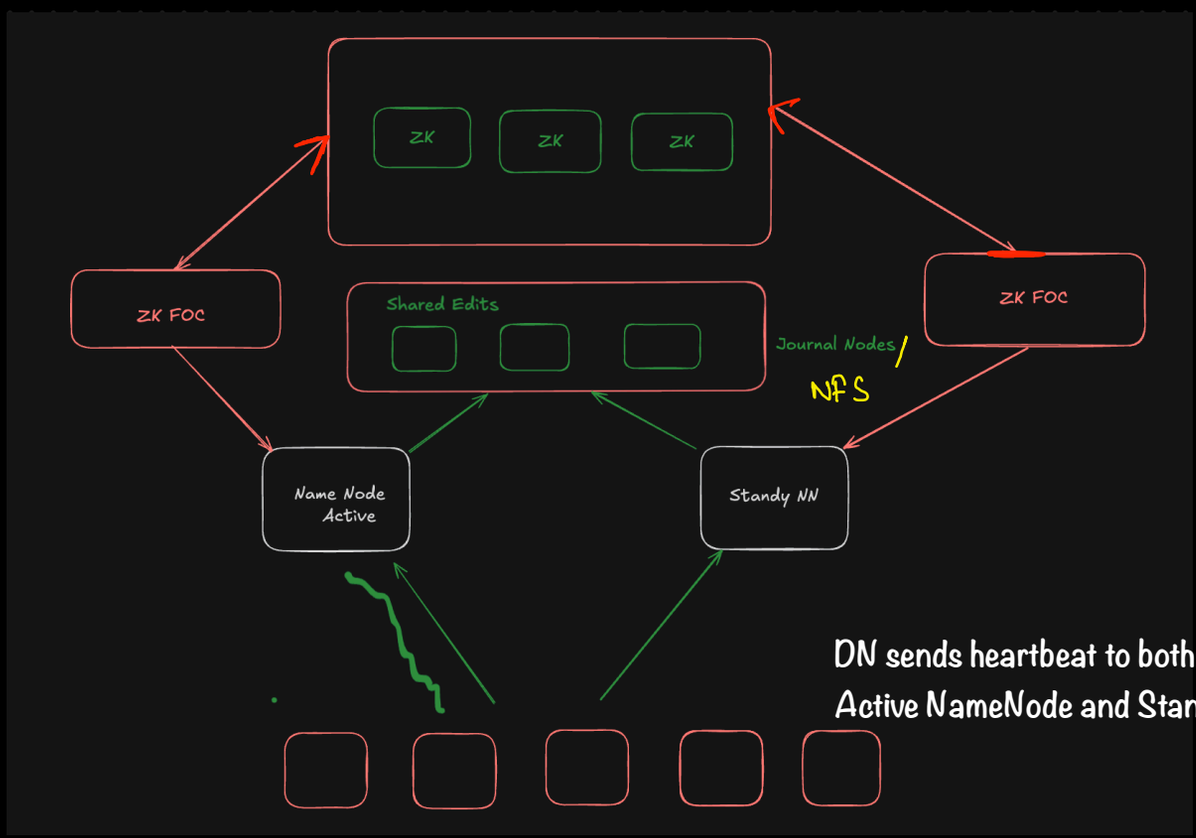
2. Failover mechanism

Standby Name node can take role of active NN, ensuring minimum downtime & data loss.



Aspect	Secondary NameNode	Standby NameNode
Role	Helper for checkpointing and edits log management.	Hot backup for Active NameNode, enabling seamless failover.
High Availability	Does not provide HA; cannot replace the NameNode.	Ensures HA by taking over the Active NameNode's role on failure.
Synchronization	No continuous synchronization; works periodically.	Continuously synchronized with the Active NameNode.
Failover Capability	Not capable of failover; only provides updated fsimage.	Fully capable of taking over in case of Active NameNode failure.
Introduced In	Hadoop 1.x	Hadoop 2.x

Hadoop High Availibilty architecture



Component	Role	Explanation
Active NameNode	Main controller managing HDFS namespace and block information.	Handles all client operations like file creation, deletion, and metadata management.
Standby NameNode	Hot backup of the Active NameNode, ready to take over during failure.	Synchronizes its state with the Active NameNode using edit logs stored in JournalNodes.
DataNodes (DN)	Store actual data blocks and handle data read/write operations.	Send regular block reports to both Active and Standby NameNodes.
JournalNodes (JN)	Maintain a shared edit log that keeps both NameNodes synchronized.	Facilitate consistent updates to the Standby NameNode to ensure it has the latest changes in the HDFS namespace.
Zookeeper (ZK)	Coordinates failover and prevents split-brain scenarios.	Monitors NameNode states and triggers leader election during failover.
FailoverController	Monitors the health of NameNodes and initiates failover.	Works with Zookeeper to perform automatic failover when the Active NameNode is unresponsive.
Heartbeat Mechanism	Periodic signals exchanged between components to check health.	Ensures timely detection of failures in NameNodes or DataNodes.
Block Reports	Periodic reports from DataNodes to NameNodes about stored blocks.	Ensures both Active and Standby NameNodes have accurate information about the location of data blocks in the cluster.
Clients	Users or applications interacting with the Hadoop cluster.	Access HDFS files through the Active NameNode. Failover is transparent to clients, ensuring uninterrupted service.

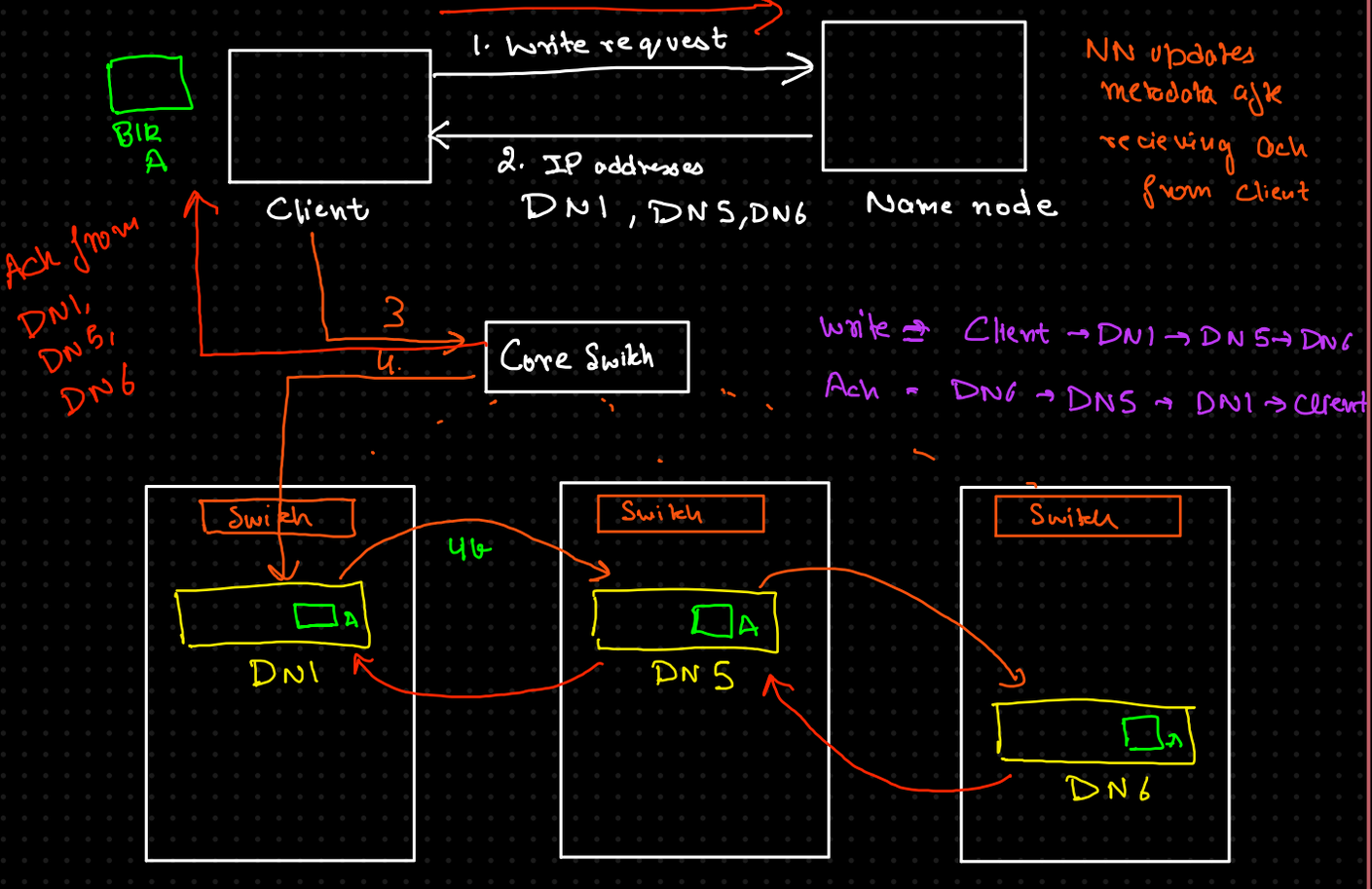
HDFS

- → Architecture of hdfs
- → Replication, blocks, rack
- → DataNode failures
- → Name Node failures
- → hdfs write
- → hdfs read

Write operation happens in hdfs

3 steps process

1. Interact with NN
2. moving actual data to DN
3. Acknowledgement



Fault tolerance & Failure handing

1. Datanode failed during write

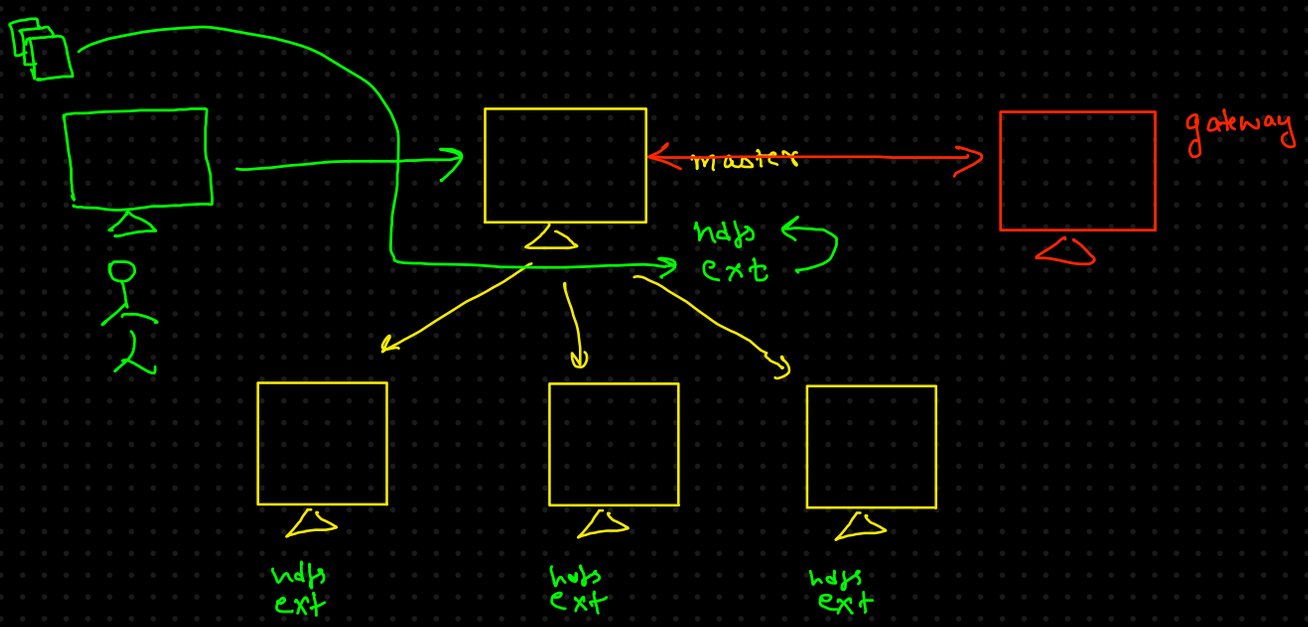
9:30 ~ Hadoop cluster

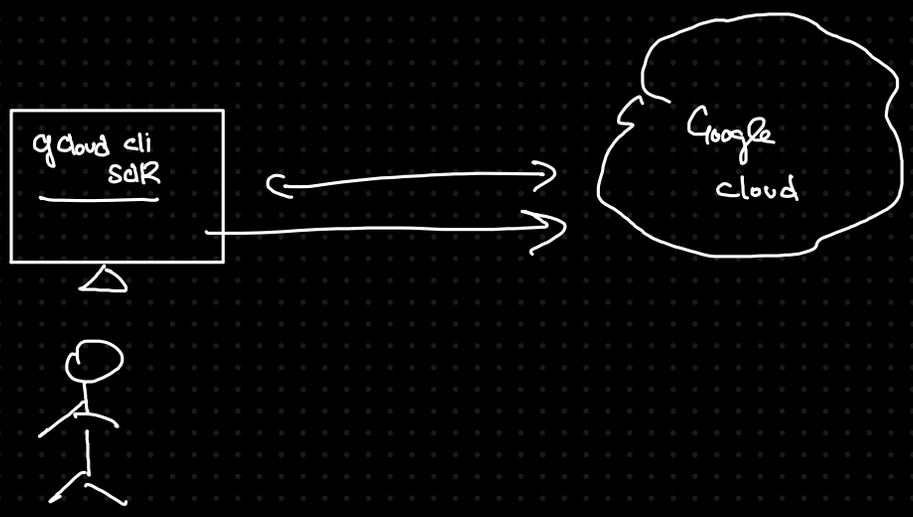
Read HDFS . MR, YARN
↳ extra class

Linux, hadoop commands
↳ Recordings

LinkedIn :

1. Create a hadoop cluster
2. property
3. google sclR
4. Upload a file to dataproc clusters
5. See the blocks and everything





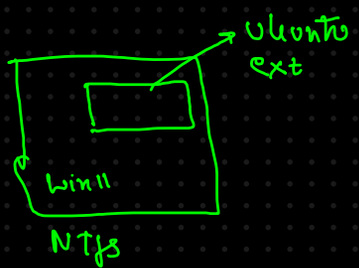
Windows

↳ Powershell
cmd

+ putty

* Windows Terminal

or



Today Class

Map Reduce

YARN

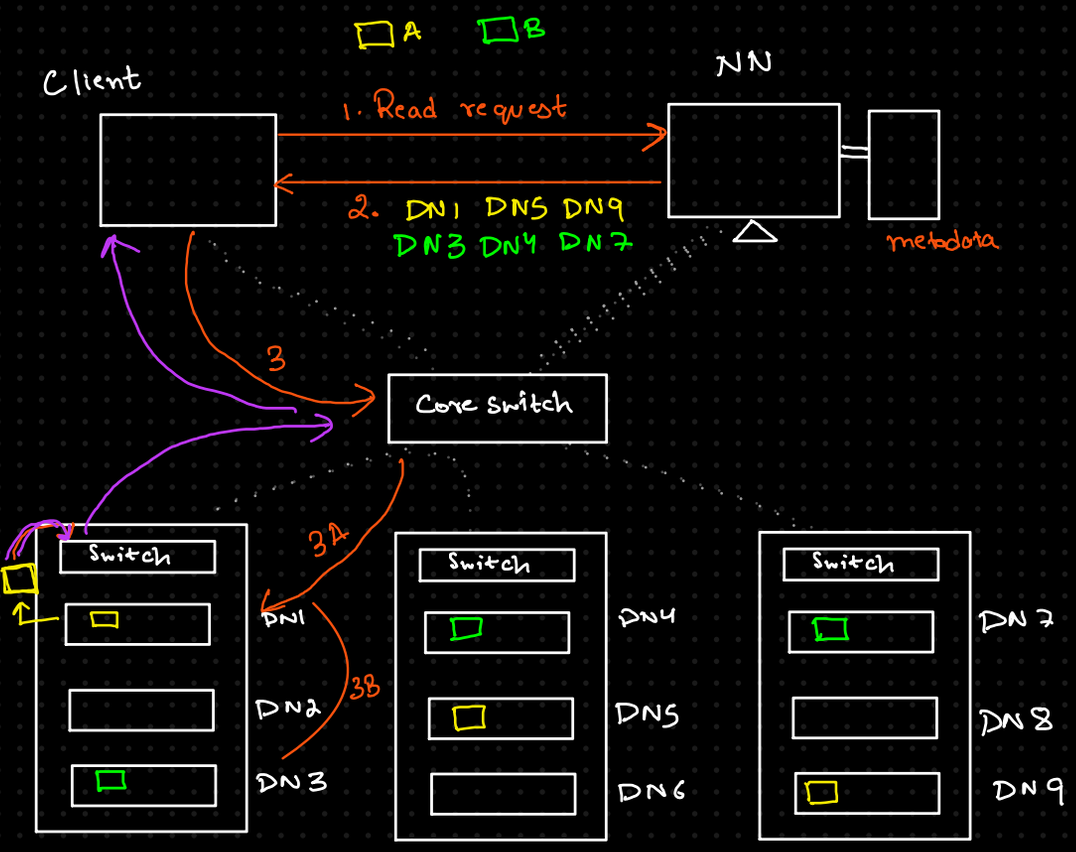
Spark

Please revise the last atleast 3 classes

Dont code with me

Linux hdfs

Reading in HDFS



3 step process

1. read request
2. Namenode provides access /loc. of DN
3. Client reads from the datanodes

* In read, there is no acknowledgement

Setting up our hadoop cluster

1. Local - pseudo distributed.
2. Local in house cluster
3. Cloudera VM / Oracle / IBM
4. Cloud → AWS, GCP, Azure

Diff b/w HDFS and cloud based storage

In our HDFS, storage $ compute are tightly computed.

SSD / HDD vs buying a laptop