Date   Day

HDFS

→ Hadoop Distributed File System used to store BigData.
→ HDFS stores metadata on a dedicated server called NameNode / MasterNode
→ Application data are stored on other servers called DataNodes / Slaves
→ All servers are fully connected and communicate with each other using TCP-based protocols.
→ DataNodes in HDFS donot use data protection mechanism such as RAID
↳ "Redundant Array of Inexpensive Disk" is a data storage virtualization technology that combines multiple physical disk drive compone into one or more logical units for the purpose of data redundancy, performance improvements.

▷ ARCHITECTURE :
→ Files and directories are represented on the NameNode by inodes, which record attributes like permissions, modifications and access times.
→ The file content / data is divided into chunks of 128 MB.
→ Each block of data is replicated at multiple DataNodes (mostly three nodes) to prevent loss of data

Page No.

→ Also stores the modification log of the image called the journal

contains the metadata & list of blocks belongi to each file

Date ______ Day ______

• NameNode :

→ Mantains the namespace tree

→ Determine the mapping of file blocks to DataNode (the physical location of data.)

→ Stores the metadata

→ Collect block repots from DataNodes on block locations.

→ Keeps the entire namespace in RAM for fast access

→ An HDFS client wanting to read a file 1st contacts the Name node for locations of data blocks and then reads the block content from DataNode closest to the client.

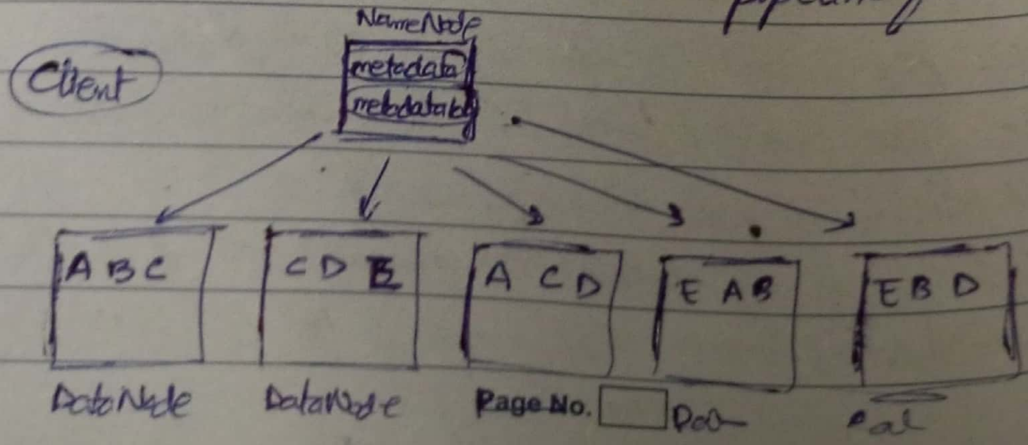
→ Can have thousands of DataNodes and tens of thousands of HDFS clients / cluster.

→ Each DataNode may execute multiple applicat tasks concurrently.

→ When writing the data, the client req the NameNode to nominate a suite of three DN to host the block replicas

→ The client then writes to DN in a pipeline fashion.

→



Page No.

Heartbeat [illegible] total storage capacity (fraction of storage in use and the no. of data transfers currently in progress.

Date Day

• DataNodes :

→ The DataNodes are responsible for serving read and write requests from the file system's clients.

→ The DataNodes also perform block creation, deletion and replication upon instruction from the NN.

→ DN periodically send back block reports to NN.

→ NN and DN communication is called Heartbeat

→ DN send heartbeats to NN to confirm that DN is operating and the block replicas it hosts are available

↳ DN sends Heartbeat after every 3 seconds

↳ Every 10th HB is a block report

↳ NN builds metadata from block reports

↳ ~~HCP~~ If HB not send in 10 mins the NN considers the DN to be out of service

• Block Reports :

→ A DataNode identifies block replicas in its possession to the NameNode by sending a block report. A block report contains the block_id, the generation stamp and the length for each block replica the server hosts

→ Blockreports provide the NN with an uptodate view of where block replicas are located on the cluster and NN constructs & maintains latest metadata for Blockreports.

Date   Day

→ The 1st BR is sent immediately after the DN registration.

→ BR are sent every hour.

• Failure Recovery:

→ NN doesnt directly calls DN. It uses replies
heartbeats to send instr. to the DN.

→ The instr. include cmds to:

① Replicate block to other nodes
   • DataNode died
   • Copy data to local

② Remove local block replicas

③ Re-register or to shut down the node

→ So when DN died, NN will notice and instruct other DN to replicate data to new DN.

• Image:

→ The FsImage is a snapshot of the entire file system metadata, stored as a single file on the disk.

→ Contains info about the directories, files, their permissions and block locations.

→ The NN loads the image into the memory at startup to reconstruct the file system's metadata.

ate Day

• Journal :
→ A sequential log of all changes made to the file system metadata (eg: file creation, deletion..)
→ Tracks update since the last checkpoint.
→ If the journal grows too large, it slows down recovery, making checkpoint essential.

• Checkpoint :
→ A process where the current metadata (combining the FsImage and edits from the journal) is saved to create a new, conistent image.
→ This task is periodically performed by Secondary NameNode.

In order to preserve the checkpoint or journal HDFS can be configured to store the checkpoint and journal in multiple storage directories.

• Startup Process :
When the name node starts :
1. It reads the checkpoint from disk to initialize the namespace image
2. It replays the journal, applying all recorded changes to update the image to it's latest state
3. After replaying the journal

Page No.

Date ______ Day

- A new checkpoint (update image) is created
- The journal is cleared, starting fresh for new transaction.

- Fault Tolerance:

→ If the checkpoint or journal bcms corrupted or unavailable, the filesystem metadata might be partially/completely lost

→ To prevent this

↳ Multiple storage directories can be configured for checkpoint & journal

↳ Best Practices:

↳ Place directories on different volumes to avoid single-vol failure.

↳ Store one directory on a remote NFS server to protect against node failures.

- CheckpointNode:

→ This node periodically merges the existing checkpoint and journal to create a new checkpoint & clears the journal.

→ How it works:

↳ The CheckpointNode downloads the current checkp and journal files from the NameNode

↳ It merges them locally to create an updated checkpoint

Page No.

Date   Day

↳ The update CP is sent back to NN.

→ Where it Runs:
↳ Usually on a seperate host from the NN, ensuring redundancy in case of node failure.

• Backup Node:
→ Like the CN, the BackupNode also creates periodic CP, but it goes further by maintaining an in-memory, up-to-date image of the file system namespace.
→ The image is always syncronized with NN.
→ How it's Better?
↳ The BN doesn't need to download the CP and journal from NN bcz it already has the latest metadata in memory.
↳ This makes CPing faster & more efficent.
→ Act as a read-only NameNode.
↳ Can handle oper. like metadata queries.
↳ It cannot modify the namespace nor manage block locations.
→ If NN fails, the BN' image in memory & checkpoint on disk is a record of the latest namespace state.

→ HDFS snapshots are created to safeguard the file system during software upgrades, reducing the r of data corruption caused by bugs or human erro Snapshots persistently save the current state of bot data & metadata, allowing adminstrators to rollback to the pre-upgraded state if issue arises, ensurin data integrity & recovery.

⊳ FILE I/O OPERATIONS :

- HDFS follows a "single writer, multiple reader" model.
- File writting & Lease :

→ A client writing to a file is granted a lease, ensuring exclusive write access.

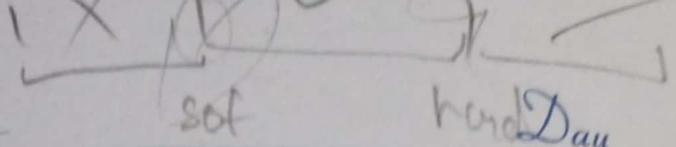
→ The lease is periodically renewed via a heartbeat sent to NN.

→ Once the file is closed, the lease is revoked, and no further modifications can be made (except appending).

- Lease Limits :

↳ Soft Limits : Ensures exclusive access for a set duration; after this, another client may preempt the lease if the writer fails to renew it.

Means another client can [illegible], the lease.



↳ Hard Limits : After one hour without renewal, HDFS assumes that writer has quit, closes the file, and recovers (take back) the lease.

↳ The new client can only request for lease after softy limit expiry & after hard limit expiry only then he can write.

• Concurrent Reads :
The writer's lease doesnot block other clients from reading the file, allowing many readers to access it simultaneously.

1. File Write Process :

1. Block Allocation :
• The NN allocates a new block with an unique ID and selects a list of DN to host it replicas
• The DN form a pipeline to minimize network distance

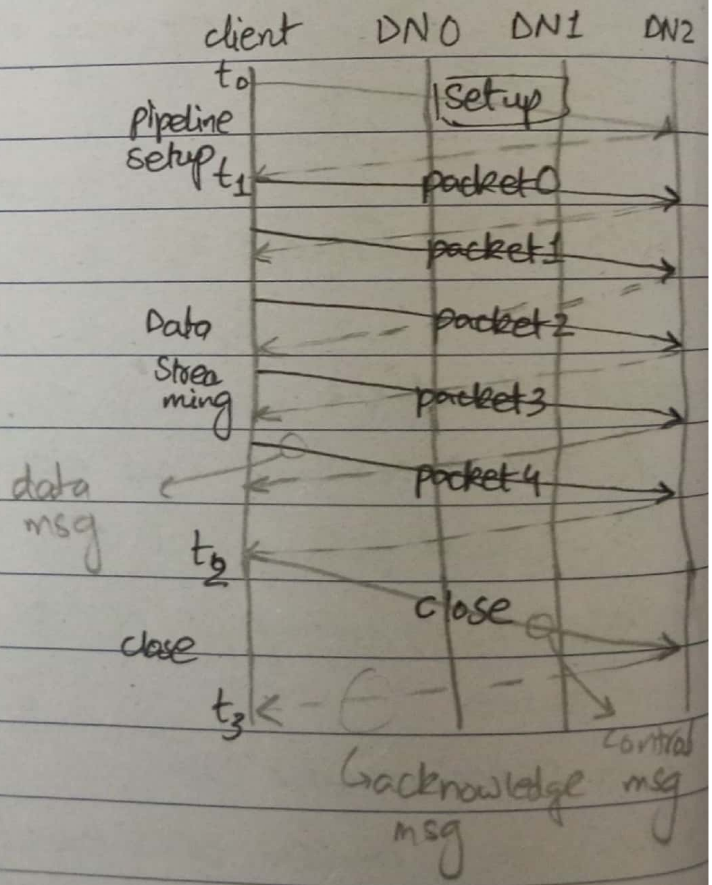
2. Data Transmission :
• The client buffers data locally.
• ~~Once the buffer (typically 64KB) is full, data is split into packets and pushed into the pipeline~~
• Bytes are pushed into the pipeline as a sequence of packets.

Date   Day

- After a packet buffer is filled, the data are p
to the pipeline
- The next packet is pushed, after recieving the acknow. from prev one.

3. Visiblity of data:
- Data is not guaranteed to be visible to new reade until the file is closed.
- The hflush operation ensures visiblity by waiting for acknowledgments from all DN in the pipelines.

4. Pipleline Stages:



2. File Read Process:
1. A client req the block location from the NN.
2. The client reads data directly from one of the DN hosting the block.

Date Day

3. Check sums

↳ HDFS verifies block integrity using checksums stored on the DataNode

↳ If corruption is detected, the client notifies the NN & fetches ~~the~~ an alternate replica.

3. Data Integrity With Checksums:

1. Checksum generation:

- When writing data, the client computes checksums for each block & sends them to DN along with the data.
- DN stores these checksums in a separate metadata file.

2. Checksum verification:

- During reads the client computes the checksum for the data it recieves and compares it with stored one
- If there is a mismatch, it detects corruption

3. Corruption Handling:

- The client inform NN about the corrupt replica
- The client then goes to differ replica.

NOTE Page No.

Date   Day

▷ REPLICA MANAGEMENT :

1. Block Placement :

- Cluster Topology : Nodes are grouped into racks connected via switches, with faster communicatio within racks than b/w them.
- Rack Awareness :
  - HDFS estimates network bandwidth based on distance b/w nodes
  - A rack identification script helps the NN determi a DN's rack
- Default Placement Policy :
  - First replica on the writer's node
  - 2nd & 3rd replicas on two nodes in a diff rac
  - No DN hosts more than one replica of a block
  - No rack has more than two replicas ~~unless~~ provided there are sufficent racks on the cluster
- Pipeline and Proximity :
  - Data is written in a pipeline order
  - Blocks are read based on the proximity to the client.

Date Day

2. Replication Management :

- Under-replication :
  - The NN adds under-rep blocks to a priority queue
  - Blocks with a single replica are the highest priority for replication

- Over-replication :
  - Extra replicas are removed, preferring to :
    - Keep replicas on diff racks
    - Remove from DN with least available space

- Objective : Maintain balanced storage while ensuring data availiblity.

3. Balancer : (An Application Program)

- Purpose : Balances disk space utilization accross DN without reducing replicas or rack diversity.
- Operation :
  - Moves replicas from highly utilized nodes to underutilized nodes
  - Ensures no impact on data availablity during balancing.

Date Day

4. Block Scanner:

- Purpose: Periodically runs on each DN and
block data integrity using checksums.
- Operation:
  - Adjust read bandwidth to complete scans w
  a set time
  - Marks corrupt blocks and informs the NN.
  - Corrupt replicas are flagged but not delete
  immediately.

5. Decommissioning:

- Process:

1. Cluster administrator marks a DN for decommi-
using an exclude list.
2. The DataNode
   - Stops recieving new replicas
   - Continues serving read request
3. The NN replicates it's blocks to other nodes
4. Once all blocks are replicated, the DN enters
decomissioned state and can be removed safely.

Infinix NOTE

"Practice At Yahoo" Day

Date

▹ Durability of Data :

◦ Uncorrelated Node Failure :
Replication of data three times is a robust guard against loss of data due to uncorrelated node failures.

◦ Correlated Node Failures :
The failure of a rack or a core switch. HDFS can tolerate losing a rack switch (each block has a replica on some other rock because no more than two replicas are stored on same rack.)

◦ Loss of electrical Power to Cluster :
A large cluster will loose a handfull of blocks during a power-on restart.

▹ Automated Failover :
↳ Plan :
↳ Zookeeper, Yahoo's distributed consensus technology to build an automated failover soluti.
· Scalability of the NameNode.
↳ Solution :
↳ Our near-term sol to scalibility is to allow multiple namespaces (and NameNodes) to share the physical storage within a cluster

NOTE

Date ____________ Day ____

↳ Drawbacks :

↳ The main drawback of multiple indep namesp is the cost of managing them.

▷ ADVANTAGES OF HDFS

① Load Balancing: As data is distributed, load on datanodes is also distributed. This allows improved performance.

② High Availability: By default HDFS replicates 3 copies of each block. This ensures fault tolerance and high availability.

③ Data Localization: Replicated copies of data facilitates localized access of data, which reduces network latency.

• But HDF entails additional cost of data storage.