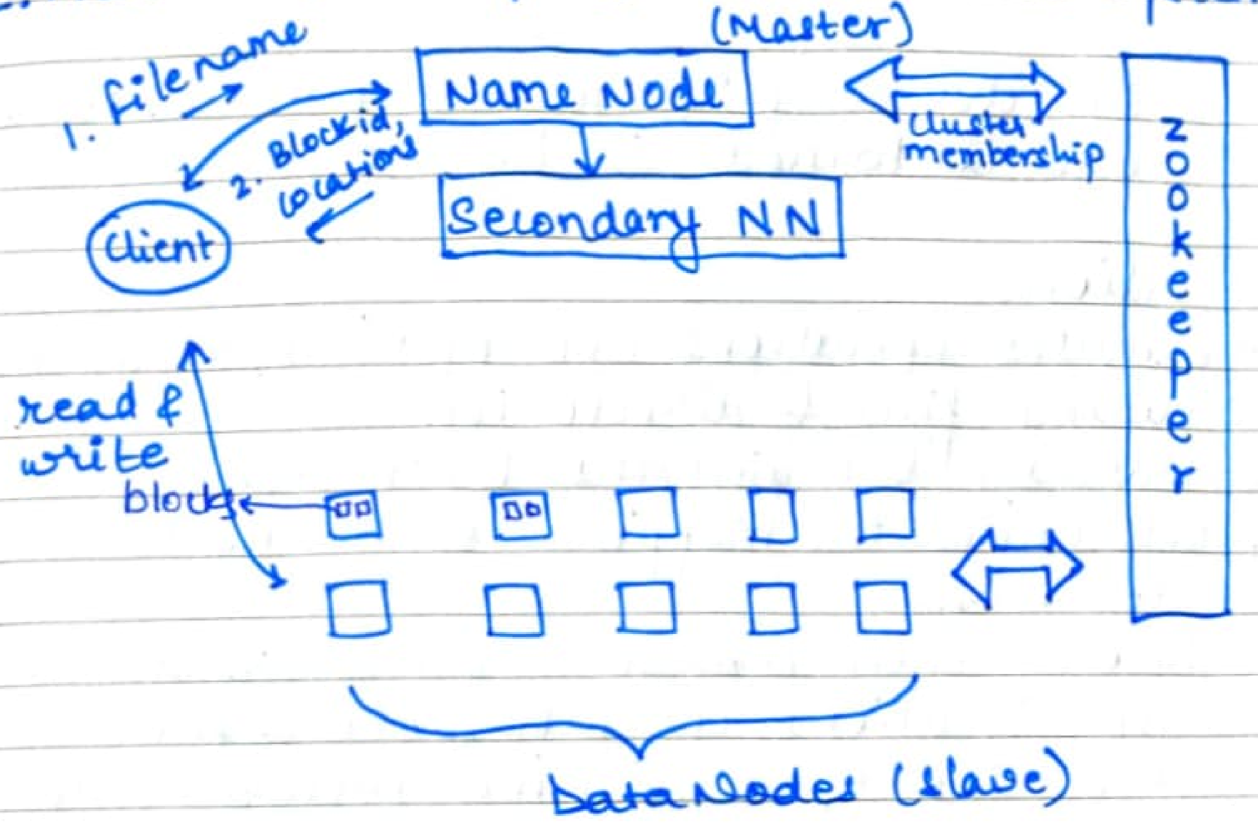
★ Hadoop Architecture

Application layer	Map-Reduce
Resource mgmt. layer	YARN
Storage layer	HDFS

1. HDFS (Hadoop Distributed File System).



→ - It is a master-slave architecture.

- Internally file gets divided into blocks whose default size is 128mb & saved on diff. datanodes depending on replication factor.

1. Name Node

→ Single master node

→ It maintains & manages the file system namespace by executing operations like, opening, renaming & closing of files.

→ Keeps Metadata of info. being file permission, names & location of each block.

- These are small so stored in memory of NN, allowing faster access to data.
- These are stored in fs_image file

- The changes performed to file sys. namespace are contained in Edit log.

• Functions

→ Executes operations like opening, renaming & closing files & directories.

→ Manages & maintains Data Nodes.

→ Determines the mapping of blocks to a file to DN.

→ Records each change made to FS ~~Image~~ namespace.

→ Keeps locations of each block a file.

→ Takes care of replication factor of all block

→ Receives heartbeat & block reports from all DN, ensuring they are alive.

→ If a DN fails, NN chooses new DN for new replicas.

In Hadoop 2.0, high availability feature is added i.e two or more NN runs in the cluster. in standby configuration.

2. Data Node

→ slave nodes storing actual data in form of blocks on diff. DN.

- Functions

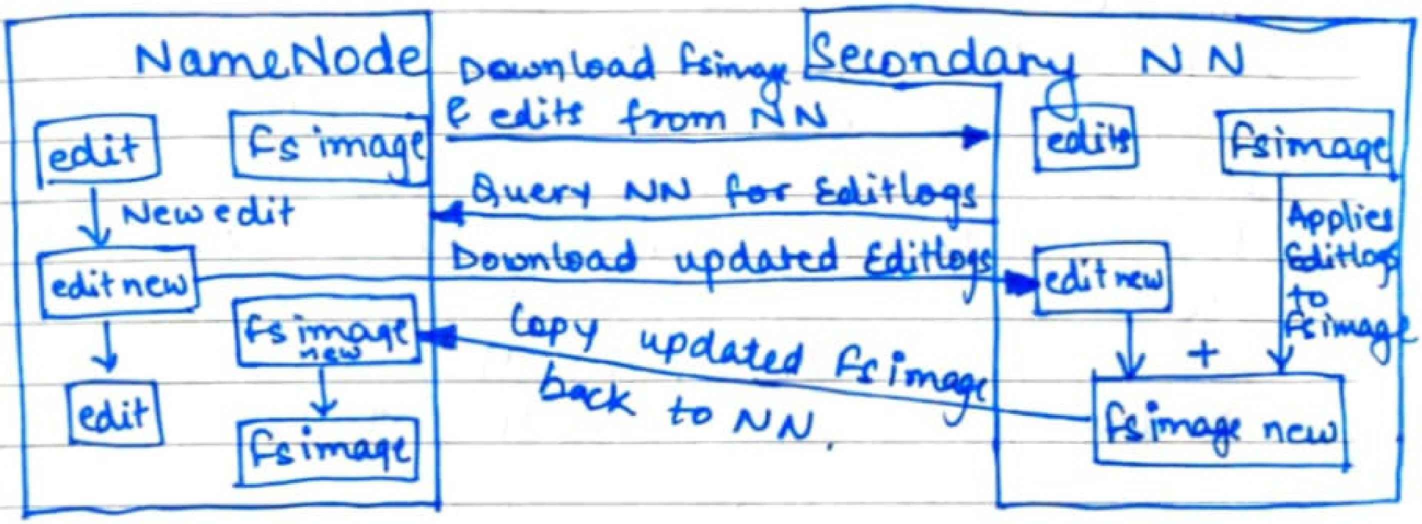
→ Responsible for serving client's read/write requests

→ Based on instruction from NN, DN performs block creation, replication, & deletion.

→ sends heartbeat to NN to report health of DN.

→ Also sends block reports to Name N. to report the list of block it contains whenever it restarts.

3. Secondary Name Node.



→ It works as a helper to NN but doesn't replace NN.

→ When NN starts, the NN merges fs image & editlogs file to restore current fs namespace. Since the NN runs continously for long time w/o restart so, the size of edit logs becomes too large. This will result in long restart time of NN next time

→ Secondary NN solves this issue.

Working:

1 → S.N.N downloads the fsimage & edit logs from NN.

2 → It periodically applies edit logs to fsimage & refreshes the edit logs.

3. → The updated fsimage is then ~~sent~~ used by NN so that NN doesn't have to reapply edit logs during restart.

IMP 4 → This keeps edit log small & Reduces the NN restart time.

→ If NN fails, the last saved fsimage on the S.N.N can be used to recover metadata

→ S.NN performs regular checkpoints in HDFS

4. Checkpoint Node

→ It periodically creates checkpoints of namespace.

→ It first downloads fsimage & edits from Active NN. Then it merges them locally, & uploads the new image back to active NN.

→ stores in directory having same structure as NN's directory. By this the checkpointed image is always available to NN.

- Diff. b/w SNN & checkpoint Node.

→ SNN does not upload the merged fsimage with editlogs to active NN.

→ Checkpoint NN uploads the merged new imge to NN.

5. Backup Node

→ Same functionality as Checkpoint Node.

→ Keeps an in-memory, up-to-date copy of file system namespace.

→ It is always synchronized with NN.

→ More efficient as it only needs to save namespace into local fsimage file & reset edits.

→ One per NN.

6. Rack Awareness

→ Rack → Collection of around 40-50 machines (DNs) connected using same network switch.

→ Rack awareness → concept of choosing the closest node based on the Rack information

→ NN follows rack awareness algorithm to store replica of files in diff. racks to provide latency & fault tolerance.

★ Write Operation

1→ Client communicates with NN for metadata →

2→ The NN responds with no. of blocks, locations, replicas to client.

3→ Client interacts with DN.

4 → The client first sends block A to DN 1 along with ip of other two DNs where replicas will be stored.

5 → After the block is stored in DN 1 then it copies the file to DN 2 in same rack, happens through rack switch.

6 → Now DN 2 copies the file to DN 3 on diff. rack, happens thr. out-of-rack switch.

→ When Data N. recives block from client, it sends write confirmation to NN.

→ Same process will be repeated for each block of file.

★ Read Operation

1. → Client communicates with NN for metadata

2. → The NN responds with the locations of DN containing blocks to client.

3 → Client interacts with DN & starts reading parallely based on info. received by NN.

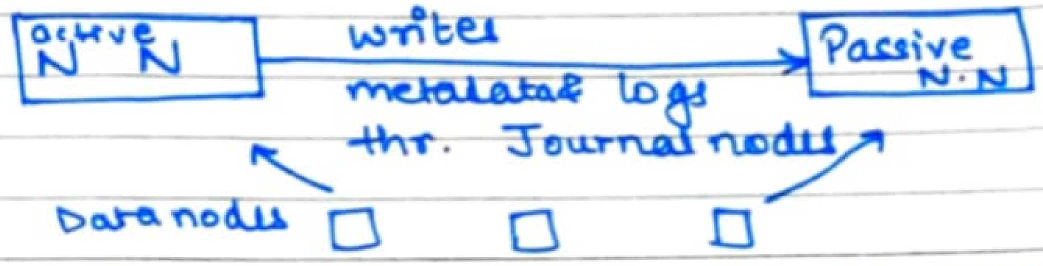
4 → When client receives all blocks of file, it combines these blocks into the form of an original form.

- Features

1. Cost effective : Commodity hardware.
2. Stores large datasets | variety & vol. of data.
3. Fault tolerance : (replications)
4. Reliability : (If one node fails, replaces it w/o shutting whole cluster)

5. High availability (2 or more NNs)
6. Scalability (adding nodes on fly).
7. Data Integrity (checks the data to original data at the time of storing for correctness)
8. Data locality (computation to data).

★ High Availability



Journal nodes (At least 3 JN).

→ Active NN writes edit logs to journal nodes & then to passive NN.

→ Passive is in continous sync with active NN.

→ To remove ambiguity of which will be the active ~~[illegible]~~ NN, fencing process is performed by Journal node which decides which NN will be the writer.

→ DN sends heartbeat to both NN but receives cmds of active NN only. (as well as block loc. info.).