29/Nov/2023

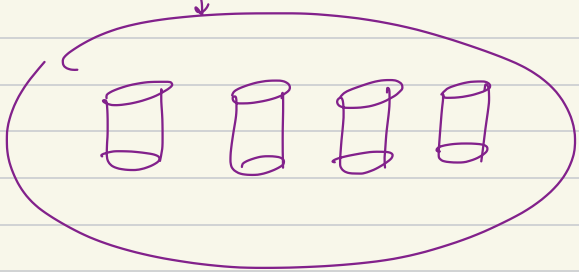
CAP Theorem

Consistency: On every read, the distributed system should return the value of the latest write OR should return an error.

T_1
T_2
T_3
|
|
|
T_4

chronological order

write (A, 100) T_1



never return a stale value ☺

T_2 Read (A) = 100

T_3 Read (A) = 100

T_4 Read (A) => error ☹

T_5 Write (A, 200)

T_6 Read (A) = 200

T7	Write (B, 500)	
T8	Read (A) =	200
T9	Write (B, 700)	
T10	Read (B) =	err
T11	Read (B) =	700

Availability: The system is always available to take your request / answer your request even though the answer might be stale.

T1: Write (A, 200)

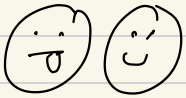
← Okay, done!

T2: Write (B, 400)

← Okay, done!

T3: Write (A, 600)

← Okay, done

T4: Read (A) = 200 

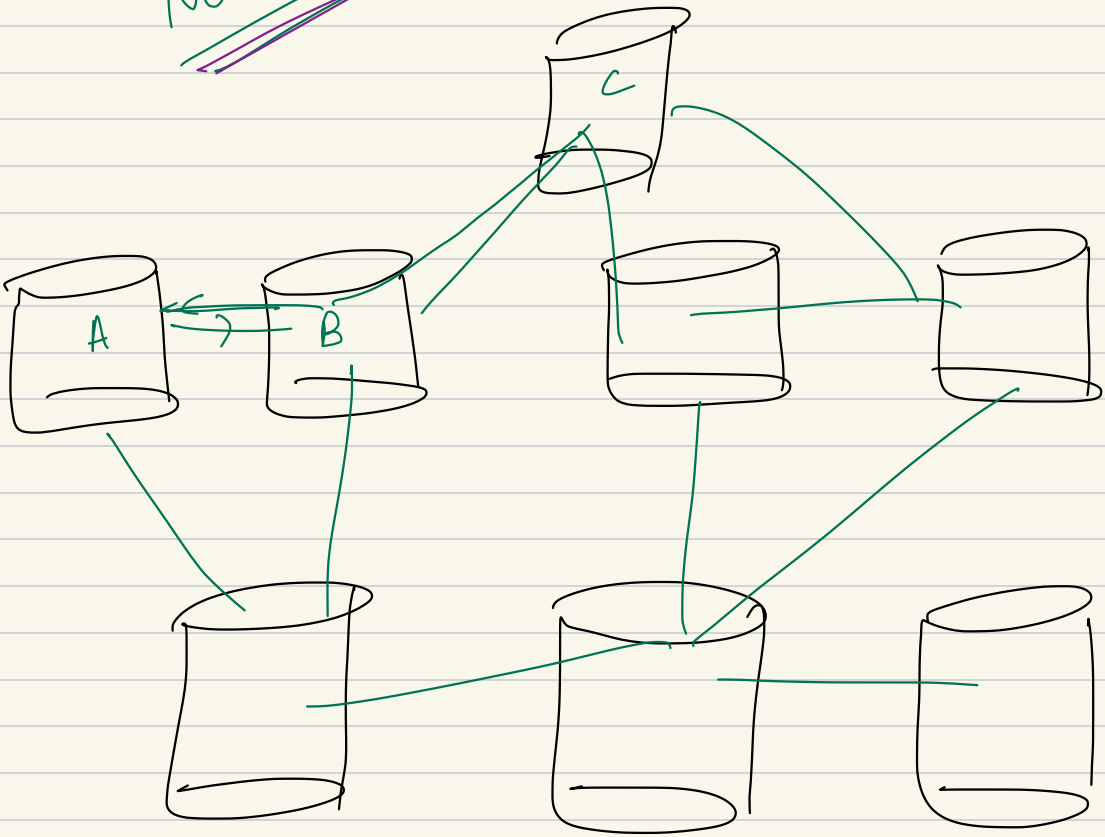
T5: Write (B, 1000)

← okay, done

T6: Read (B) = 1000

Partition Tolerance :

Network Partition



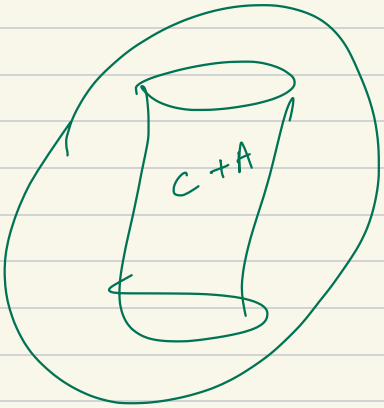
In a cluster of machines, machines talk to each other over network, and given network is unreliable, if for some time, 2 machines are not able to talk to each other, we call that event network partitioning.

CAP Theorem

It says whenever your system faces network partitioning, you have to prioritize either strong consistency

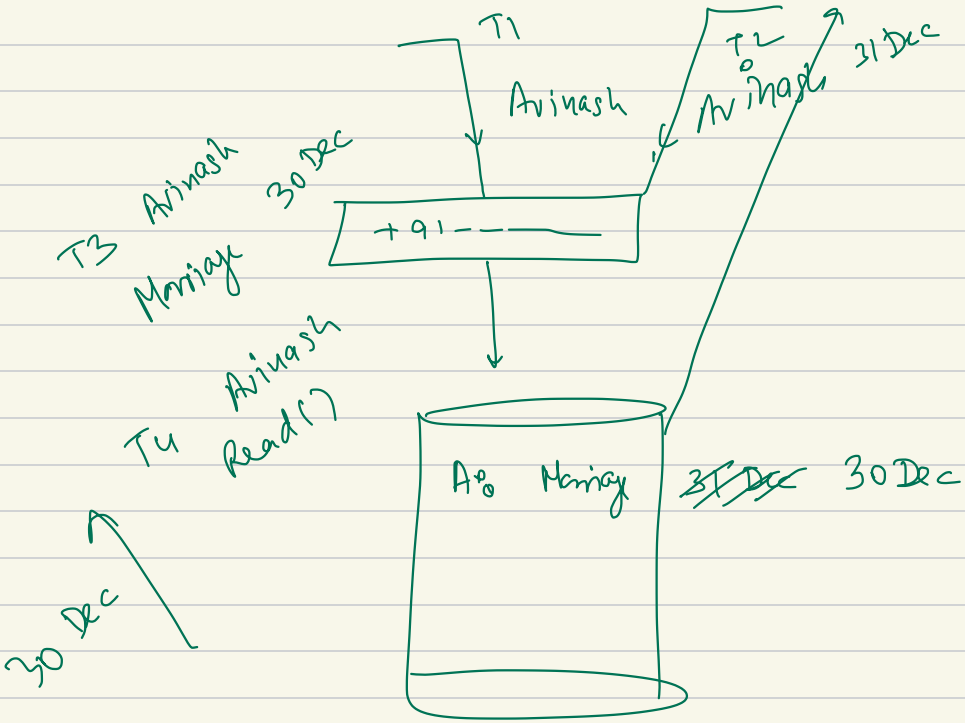
OR

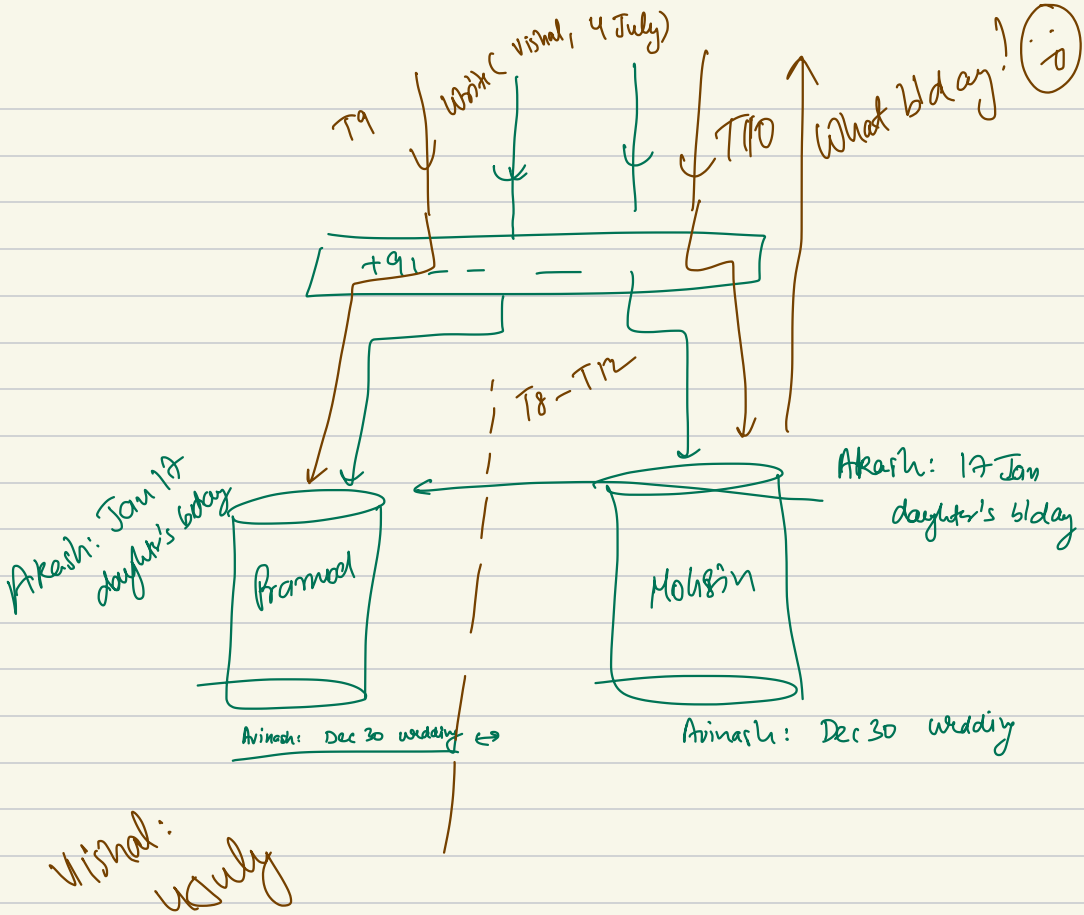
Full availability



Example:

Pramod → event reminder service




T9
Visit (Vishal, 4 July)
T10
What bday!
+91
T8 - T12
Akash: Jan 17
daughter's bday
Pramod
Mohsin
Akash: 17 Jan
daughter's bday
Avinash: Dec 30 wedding
Avinash: Dec 30 wedding
Vishal:
4 July

Examples:

Real Time Multiplayer Gaming Applications

Av > Con

Banking

Con > Availa

Comments on Facebook Post

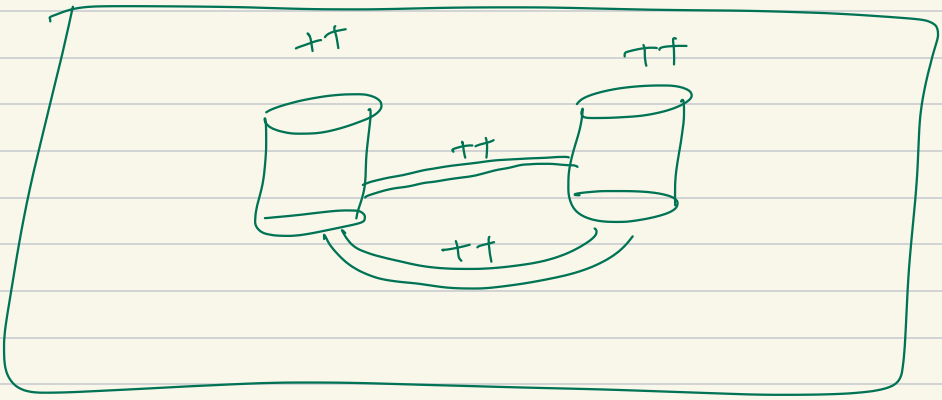
Avail > Cons

Video Ingestion on hotstar

Cons > Avail

Spanner database – Google

- proprietary hardware
- super redundant network cables



C and Av

when partition

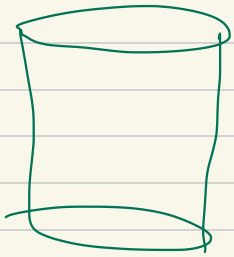
C > A

CAP

when P happens

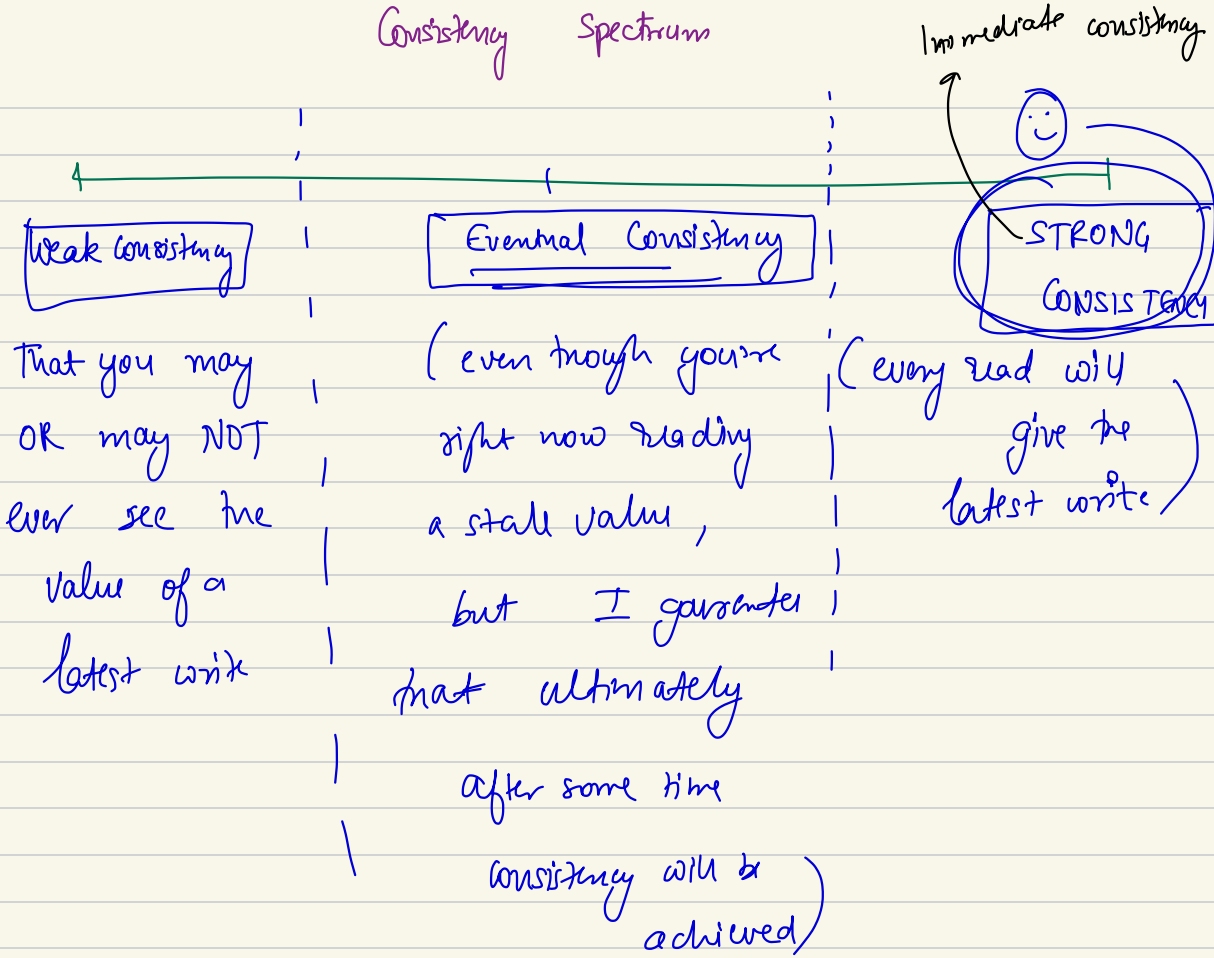
CP OR AP

CA only when P won't happen



1 MySQL DB

Consistency Spectrum



Availability Spectrum

✓ Triple 9 availability = 99.9% of the time

downtime ≃ 9 hours / year.

✓ Four 9 availability

99.99%

downtime ≃ 1 hour / year.

$$\boxed{\frac{0.01}{100} \times 365 \times 24 \text{ hours / year}}$$

skip

PACELC Theorem

— extension to CAP Theorem

Partition → Availability OR Consistency

Even otherwise → Latency ✓ vs Consistency ✓

STORAGE LAYER

- Archival / Cold Storage
- Sharding

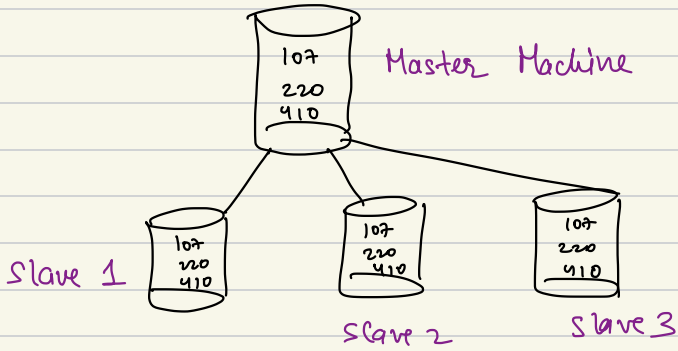
Replication

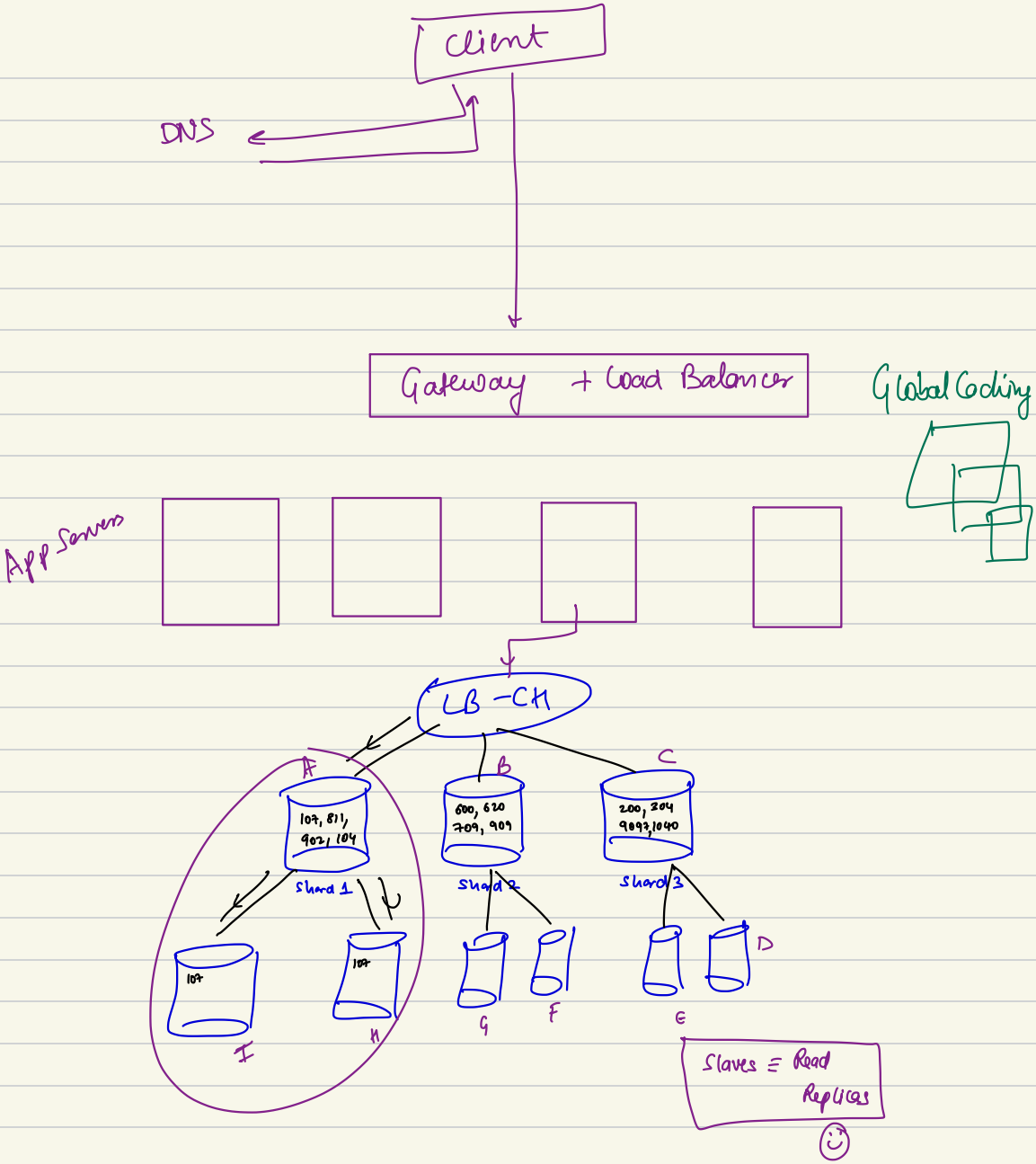
copying data

Replication

1. Master Slave Replication

Master-Slave ≡ Leader-follower





Why do we need slaves??

Why do we need replications

☆ We need replication to achieve fault tolerance

[To create redundancy]

Added advantage of M-S Replication ☺

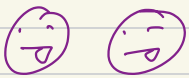
✓ Beyond redundancy / fault tolerance

+

✓ Also allows for scaling up the reads.

In Master slave Replication, whenever the Master dies, There is a leader election protocol which elects One of the slaves as the new master.

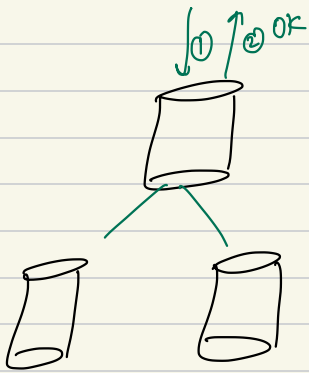
If after sometime, the died master comes back up, they join as a slave



Master slave Replication

Different replication strategies → Different Outcome.

(A) Master slave is highly available but is weak consistency



Step 1: Master takes the write Immediately ack

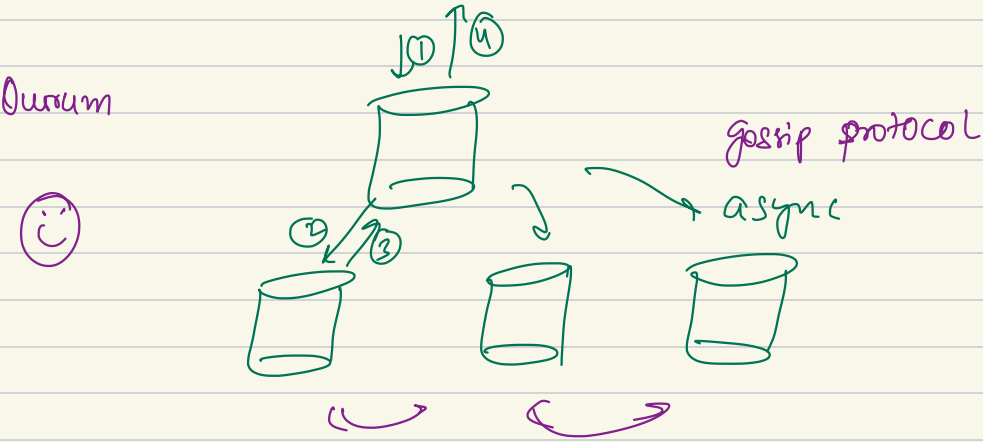
Step 2: Try to copy to slaves.

ex: splunk → logs

Least latency

② Master Slave is highly, eventually consistent

Quorum

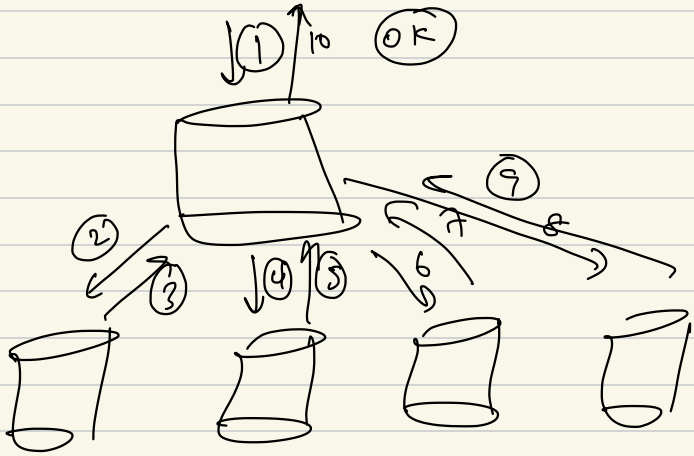


eg. Facebook Posts / Facebook comments

More latency

(3) Master slave setup

that compromises availability but gives strong consistency



PROS:

1. strong consistency

CONS:

1. highest latency
2. Availability will be compromised

Even if 1 slave is unavailable / having a network partition, we will have to reject the user write request by saying we are unavailable right now.

eg.

Question

1. Consistency vs Availability

   ↑ ↑ OR ↑

2. Consistency vs Latency

   High consistency OR Low Latency ☹