Will start at 9:10 PM

→ (1) CAP Theorem

(2) PACELC Theorem ⇐

(3) Master Slave Architecture

→ Consistent

→ Eventually Consistent

→ Never consistent

---

(1) Case Study (Analogy)

(2) Example of a real computer system.

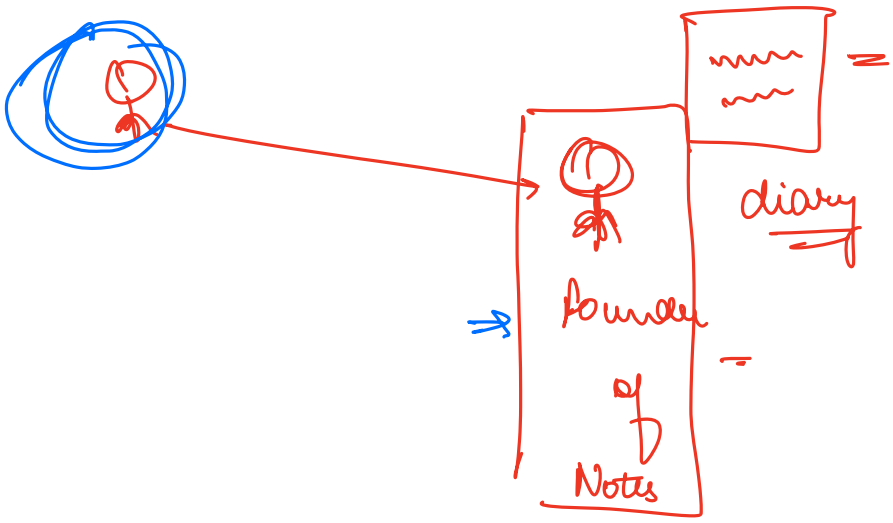
Notes

⇒ helpline number

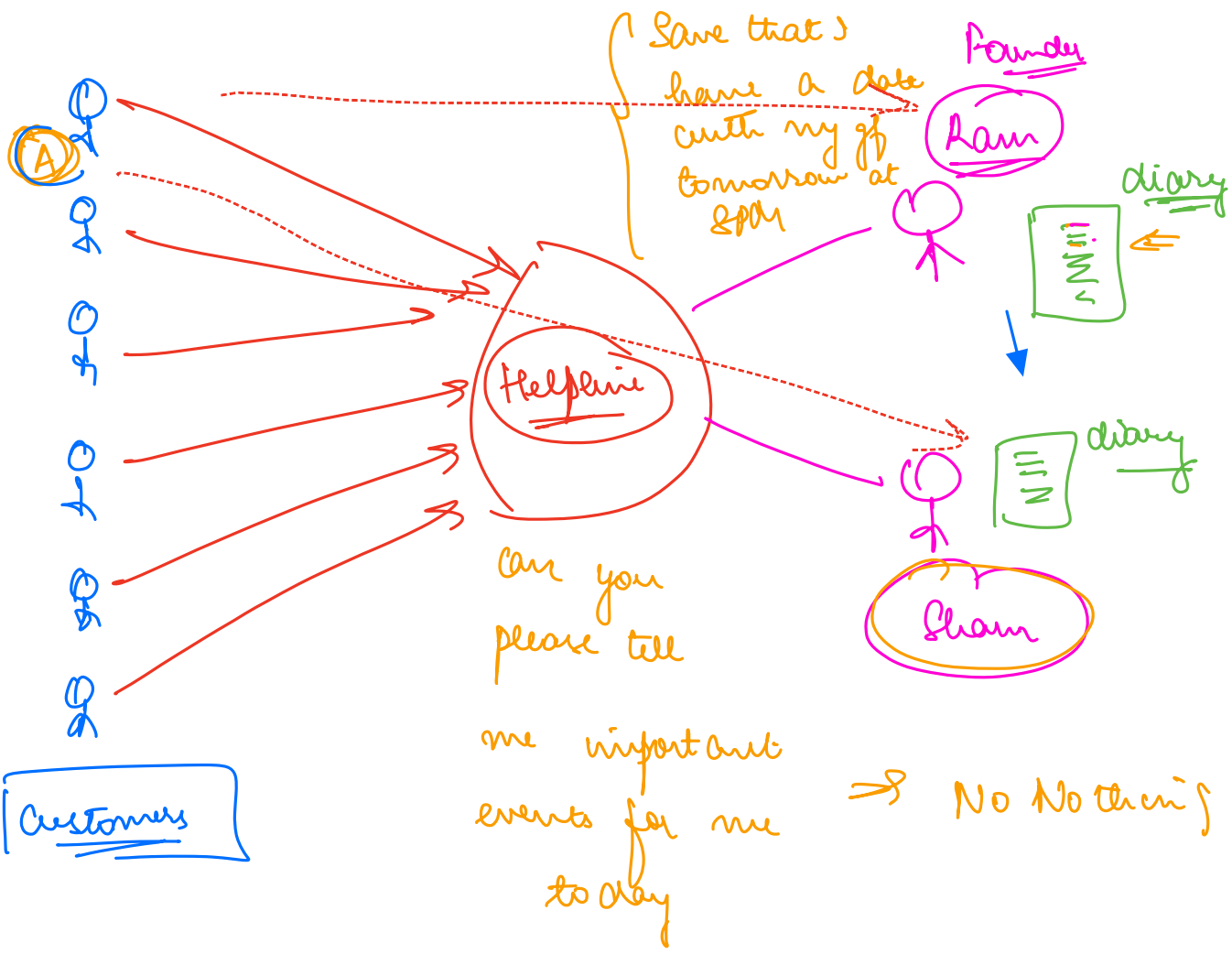
⇒ 12345

⇒ Call them and tell them whatever you want to remember / make a note of

⇒ call them and ask about your notes



→ When # of users increase, customers will have to wait a lot

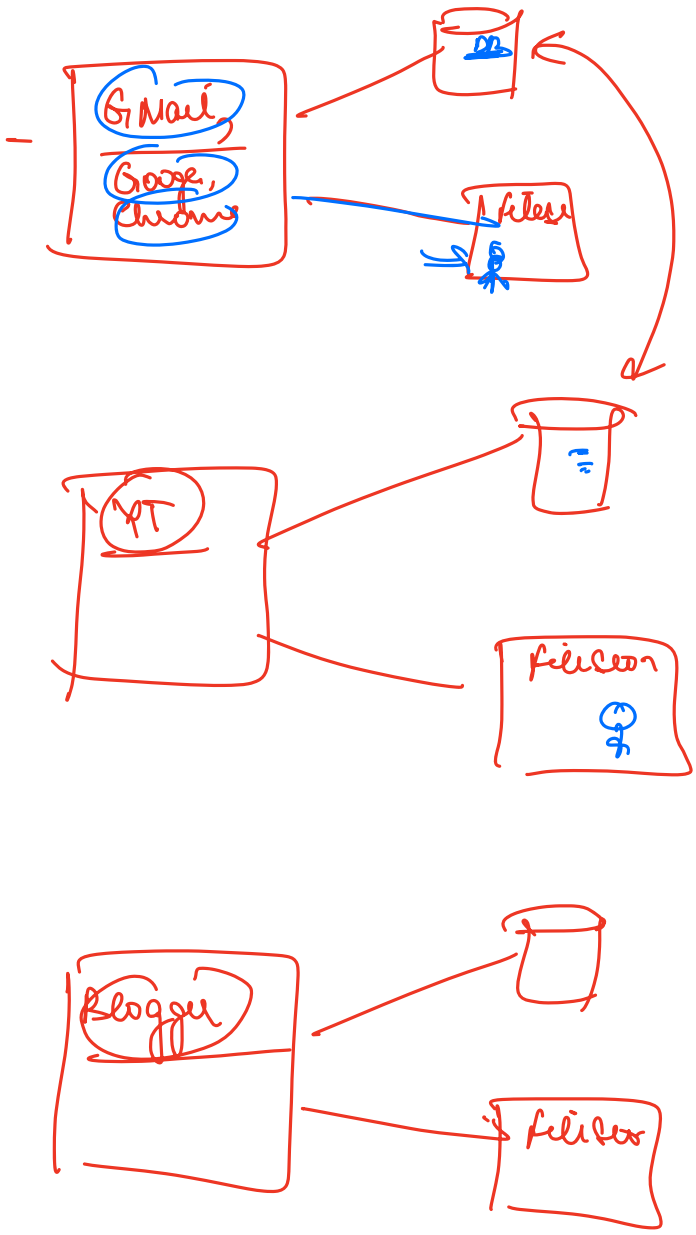


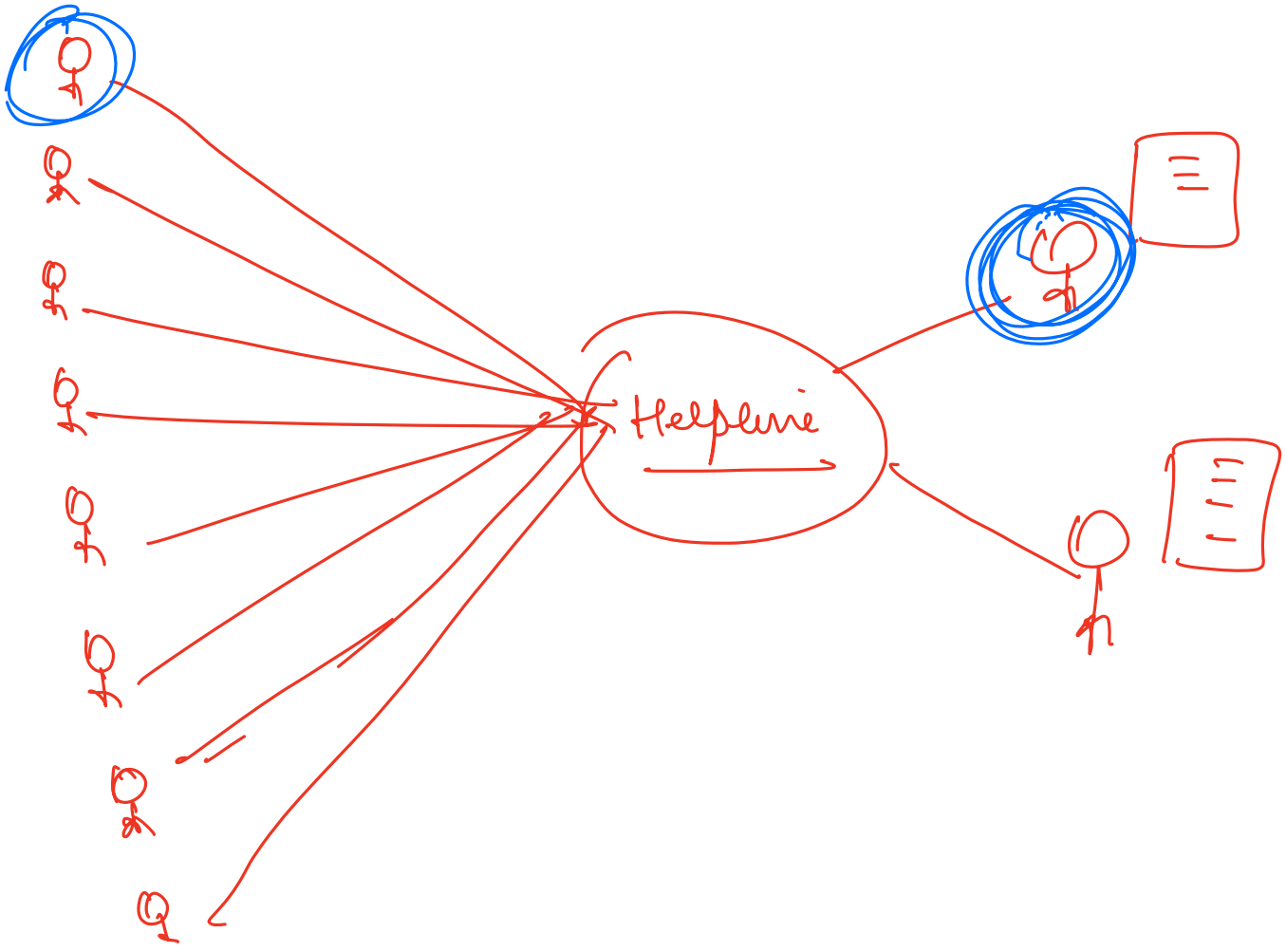
① Part of info is being stored at diff places. If the same info is not present anywhere it can lead to wrong o/p

Real World Example

GMail

→ update profile pic
(take 24-48 hrs to reflect across all google services)





→ client will rely on receiver to tell them that their request has been successfull registered.

earlier

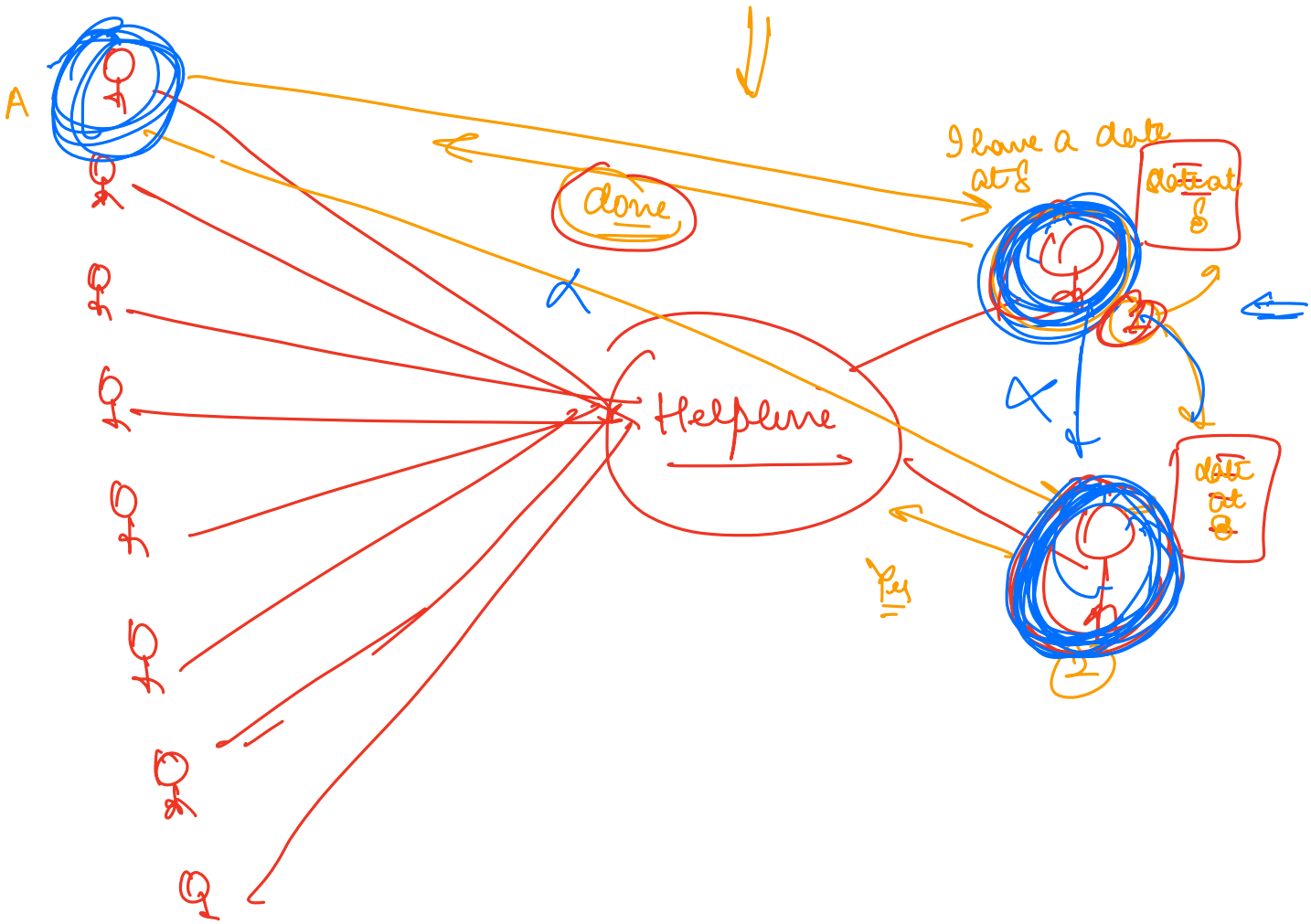
→ emp put on their diary

→ told client success

now

→ emp has to first store the data in theirs as well as others' diary.

→ return success to client.



Will there be any inconsistency in this case: No

⇒ Service is unavailable

Case 1 — Inconsistency

Case 2 — Unavailability

If the unavailable employee is unavailable for everyone:

- a.) other employees
- b.) customers

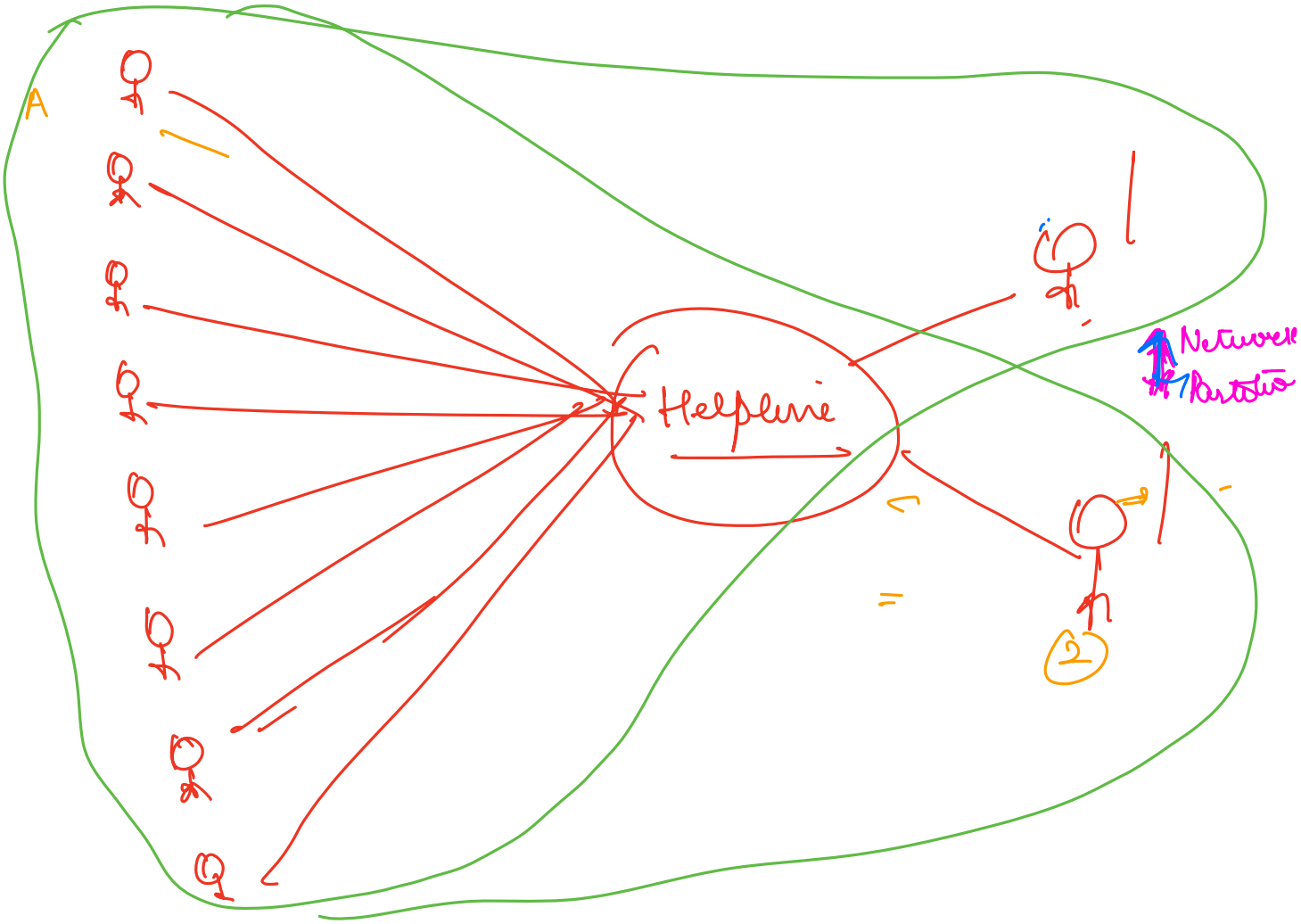
] ⇒ This assumption may not be true.

⇒ I am okay w/o storing data in their diary

⇒ No inconsistency as they aren't responding to customers either.

⇒ When the employee is back up, they just sync their diary with other employees and only after that start work

Available + consistent

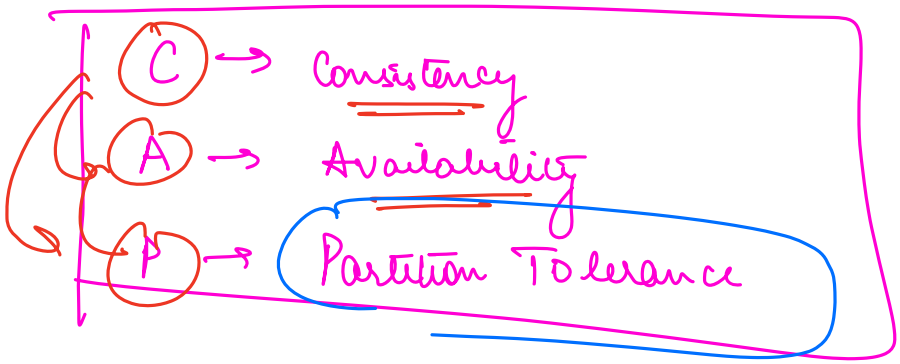


The assumption we have is not really true:

(i) intercom is down

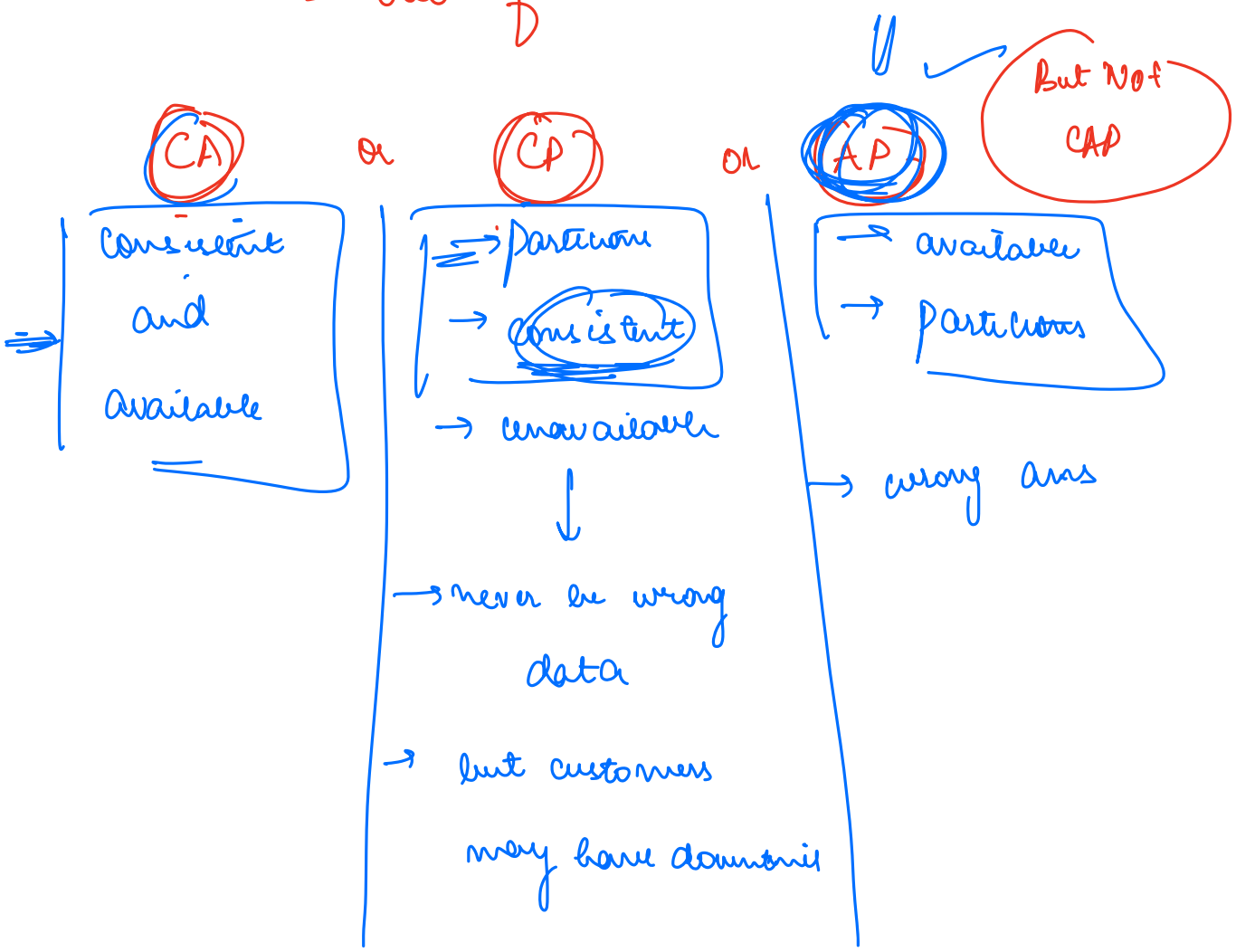
(ii) n/w is available

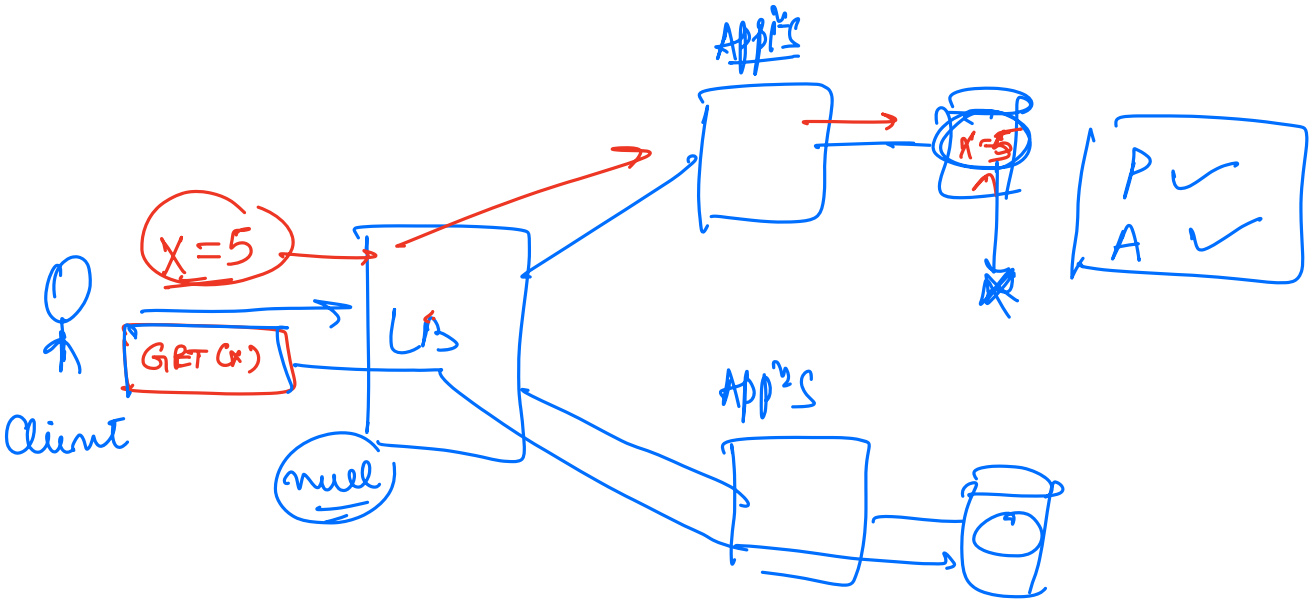
CAP Theorem



→ No System can have all of three

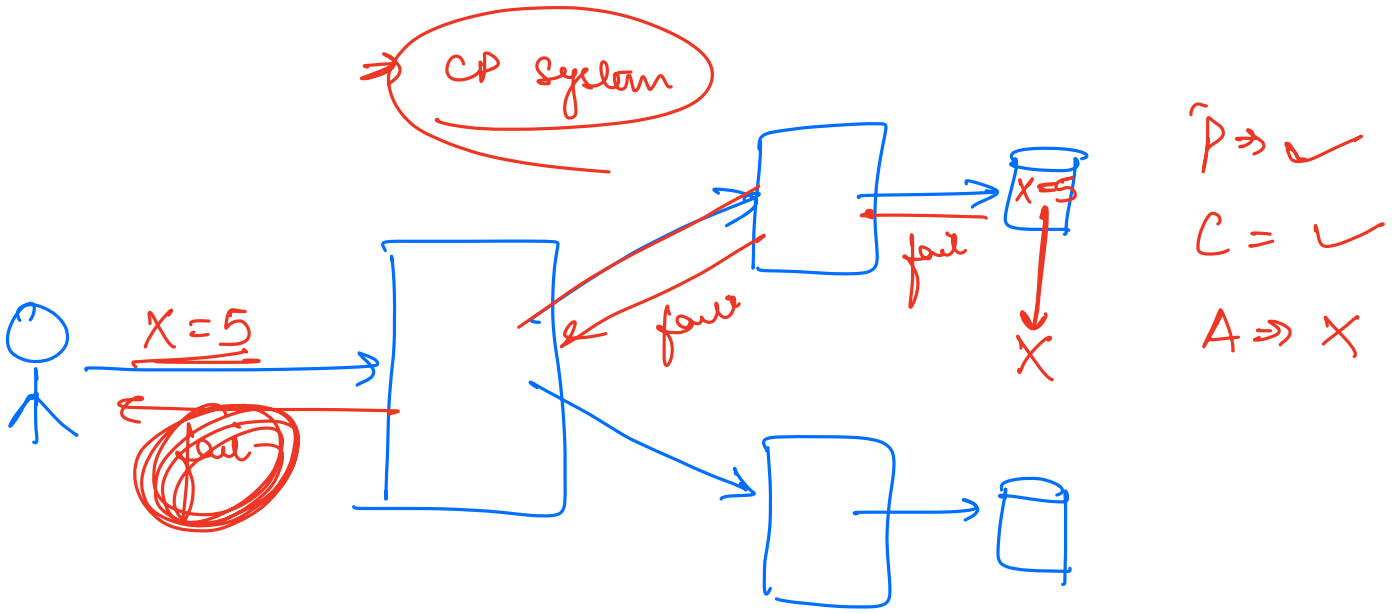
→ Any Software System can have only 2 out of three 3

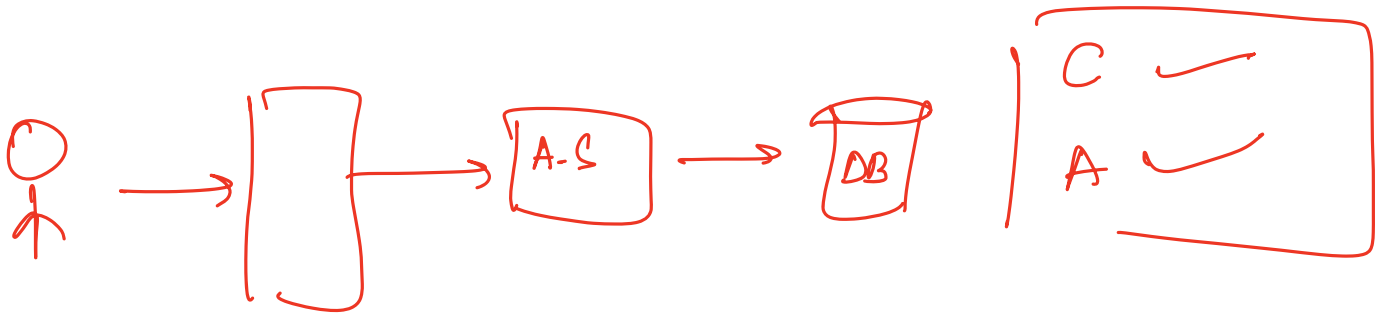




Partitions are inevitable

N/W is highly unreliable





⇒ any system can be only 2 out of 3.

⇒ can be no system that is consistent as well as available as well can have partitions

Not possible to have system w/o partitions.

⇒ N/W is unreliable

→ delays

→ go down.

CAP ⇒ For system that can have partitions, your system can be either consistent or available, but not both

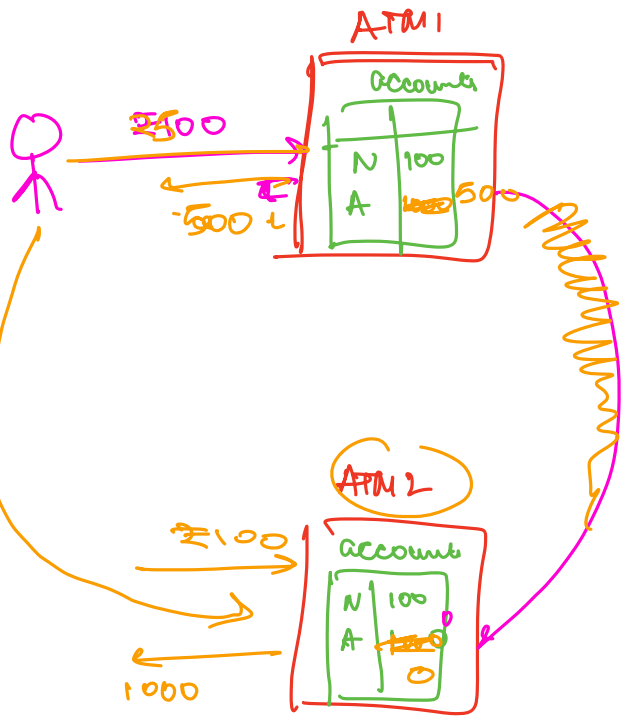
⇒ Whenever we will design systems we will ask:

~~CA~~

{ CP / AP } ⇒ Does consistency matter more or availability

① Banking System : Consistency

S/W that runs on ATM machine

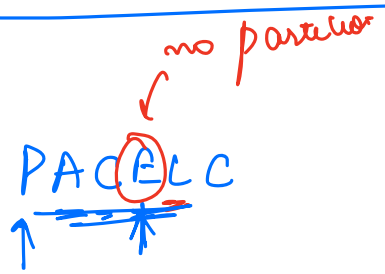


C ✓
A ✓
P ✗

	C	A
Bank	✓	
FB News Feed		✓
Slack / Teams	✗	✓
WApp / FB Messenger	✓	✗

→ Sometimes decision b/w consistency and availability is a business decision

---

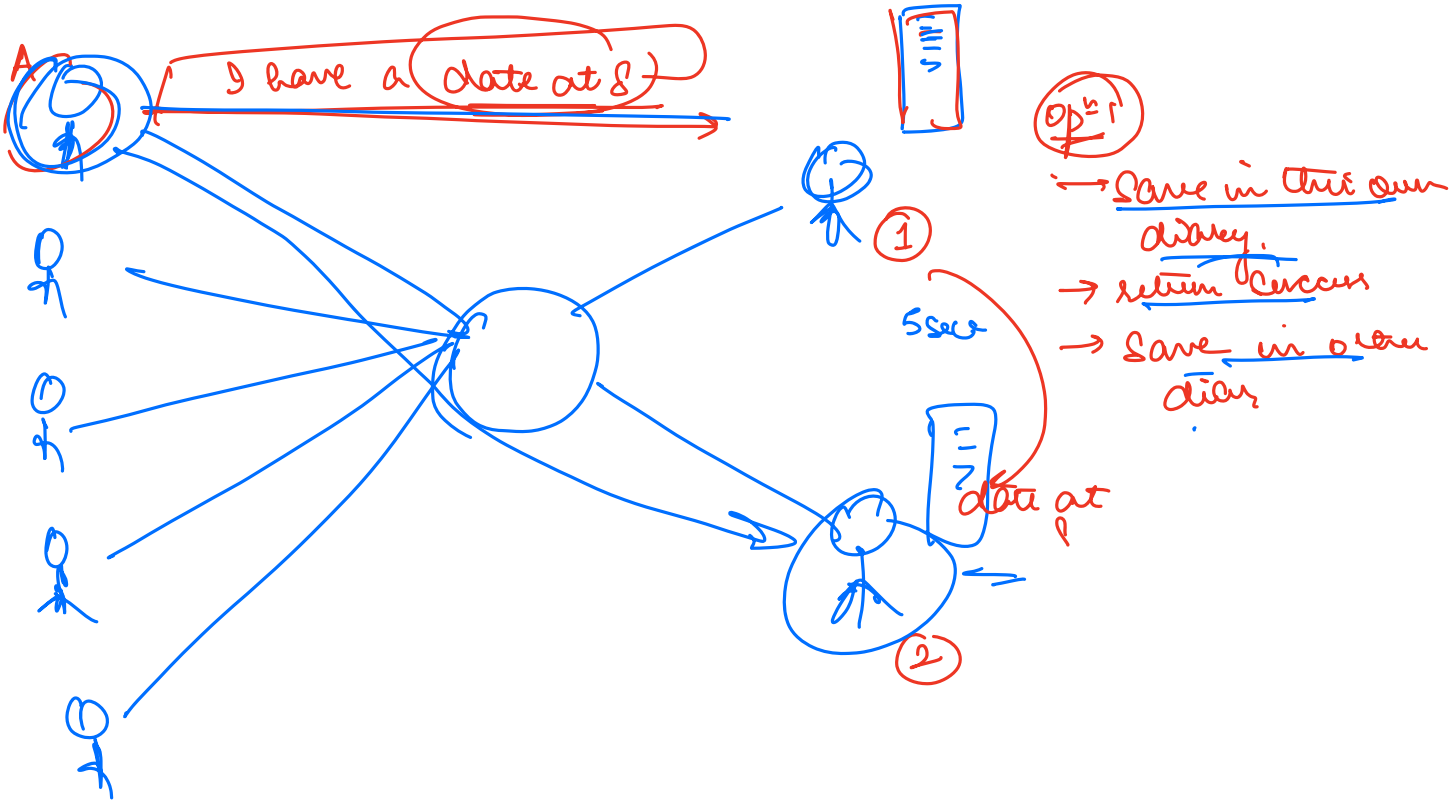


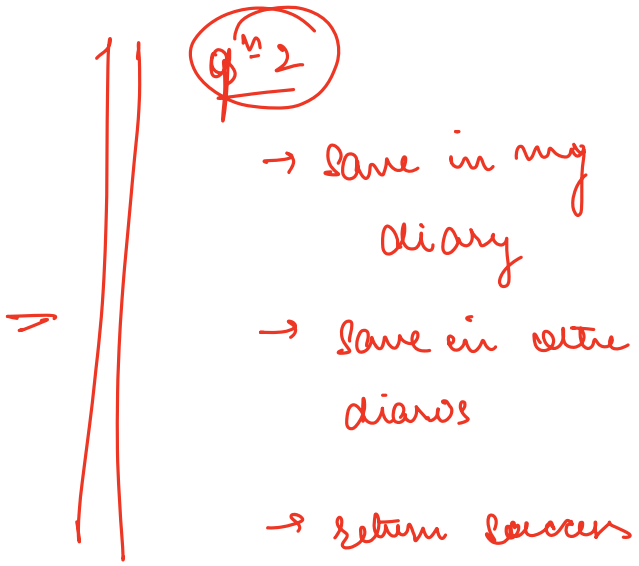
If you have partitions:

worry b/w availability and consistency

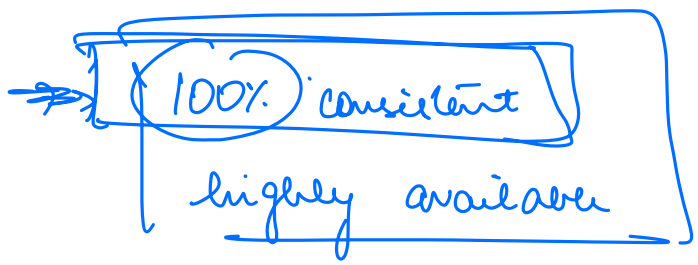
Else:

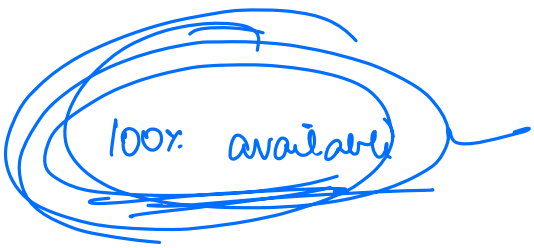
worry b/w high latency OR consistency




q^n 2
→ Save in my diary
→ Save in other diaries
→ return success


PACELC


100% consistent
highly available

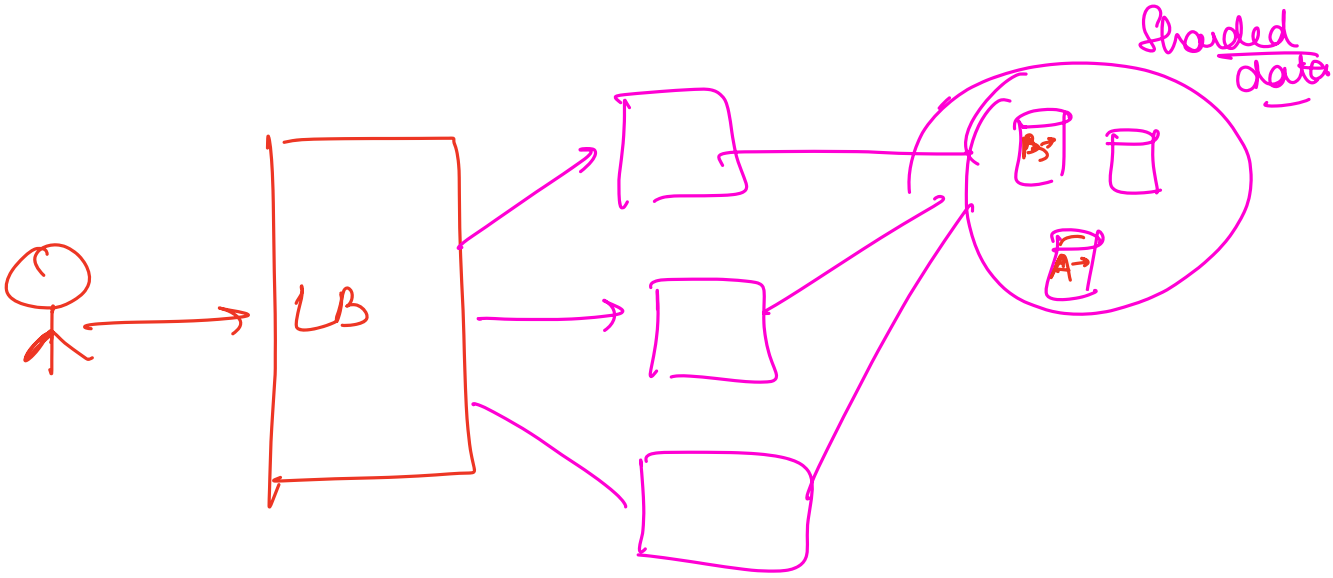

100% available


10:30 PM

⇒ 40 mins

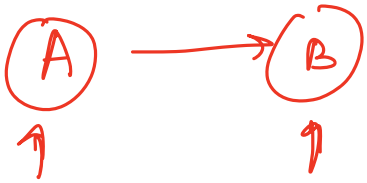

Master Slave

→ typically we don't have app^n server level mapping for DB machines



(1) Bank

↳ account details of 2 users are on 2 diff shard.

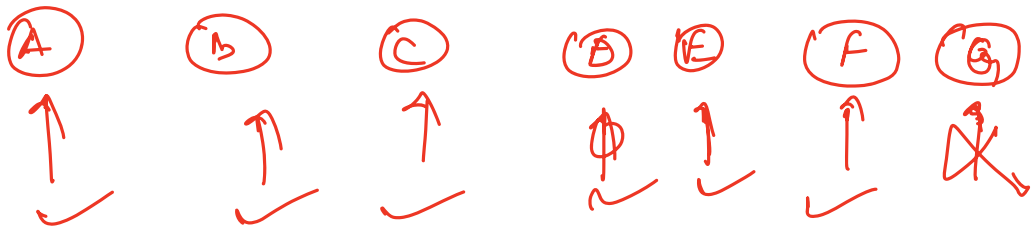


Consistency

↳ delay

(2) News Feed

↳ all posts across all of your friends can be present on diff machines.



⇒ collate from others ⇒ query

Available ✓

Consistent ✗

In a DB machine, high level there are 2 types of queries ⇒ Read

→ Write

typically ratio is not same.

100 : 1

eg

Posts ⇒ Read

Orders in Stock Market ⇒ 1:1

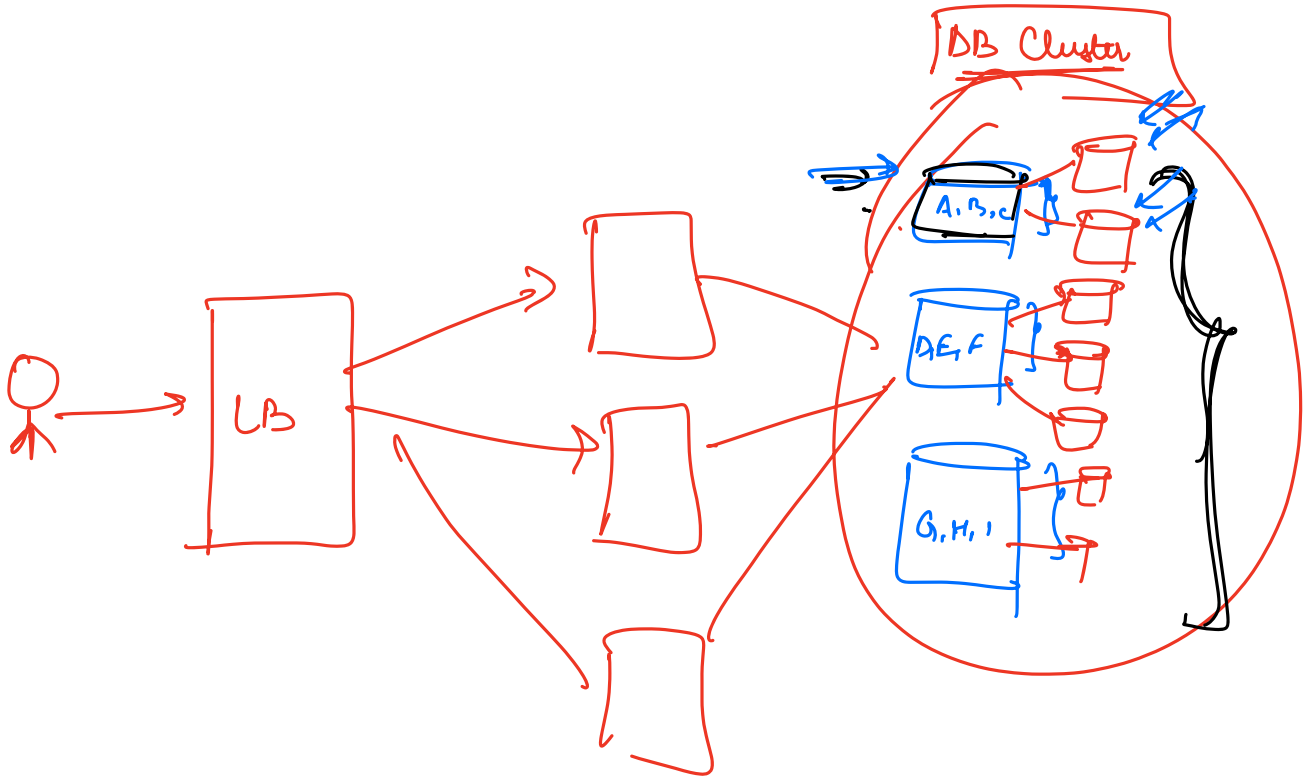
Analytics / Tracking ⇒ Write

⇒ No DB in the world can handle large # of writes as well large # of reads very fast

DB Systems

⇒ 1000 reads : 1000 writes =

→ 1 machine handles



→ each of the shards has some data (diff data)

→ ideally want to have backups of every data

(1) create more shards.

↳ # query on each shard reduces

doesn't always work (when specific rows getting lot of queries)

not the most cost intensive sol^n

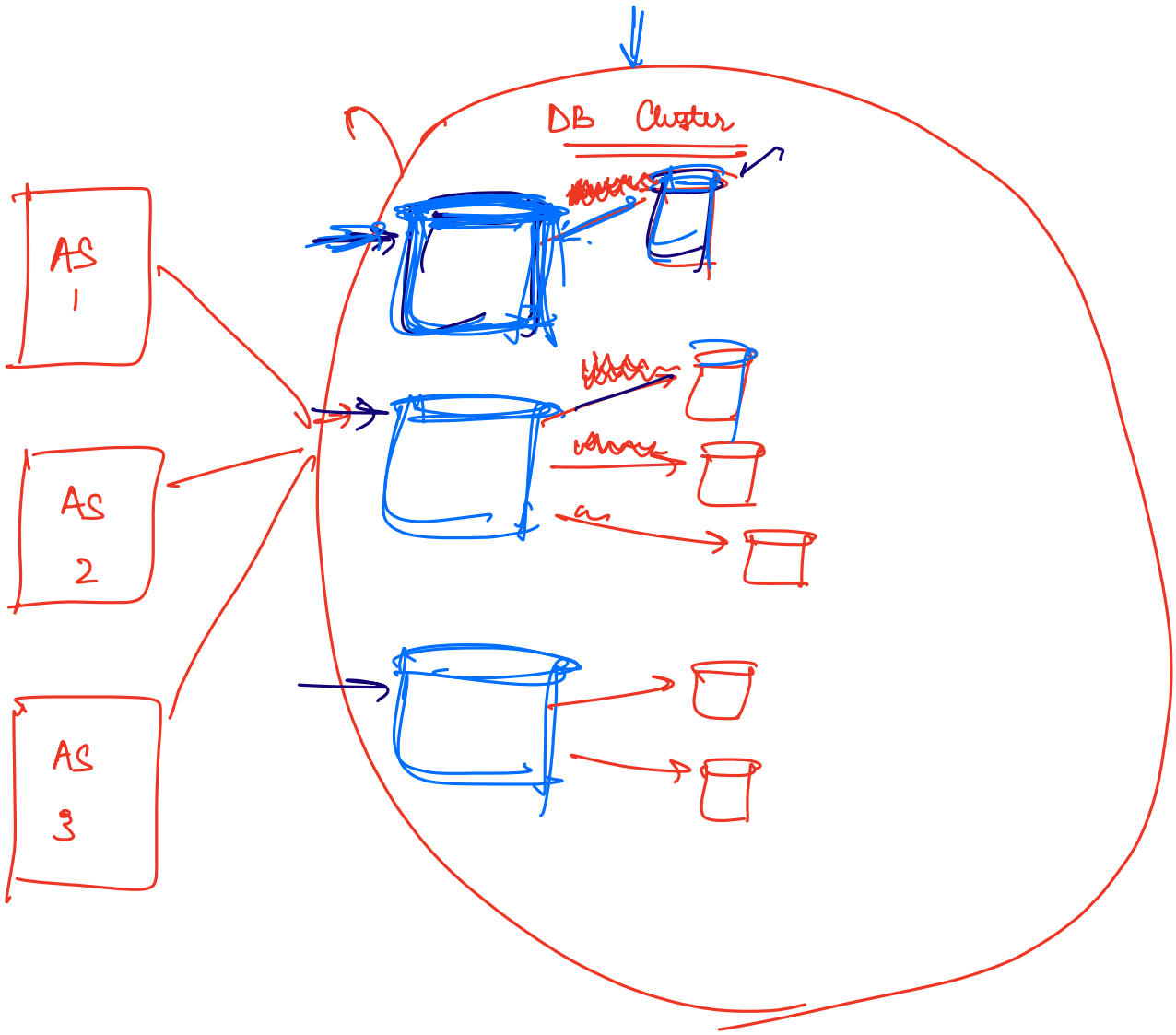
(1) We need Backups

(2) 1 machine may not be able to handle large # queries and sharding data further might not be sol^n

⇓

for each shard, create more replica

⇒ to handle load on that shard



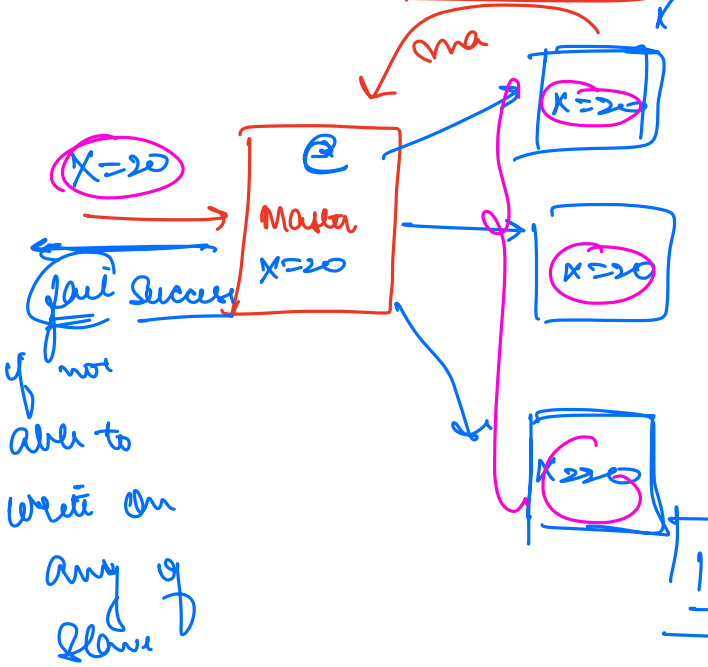
(1) 1 of these servers will handle all of the write queries ⇒ Master DB

(2) other servers will act as backup of. handle read queries ⇒ Slave DB

Backup ⇐

More read queries ⇐

100% consistent M/S setup



1. write to all slaves as well as you
2. Success.

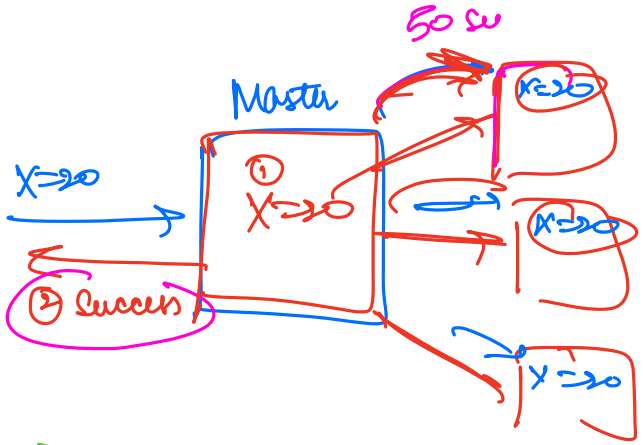
→ Read can happen from any machine

→ Consistent

Write → slower

Write to all or none

100% Available System



20s →

Available

Consistent X

→ Only save to master

→ return Success

→ Save to other machines

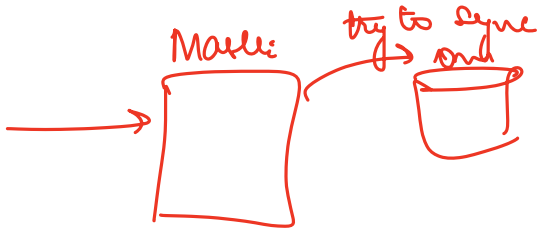
- 100% Available

eventually consistent ✓

→ after some time all machines will end with same value.

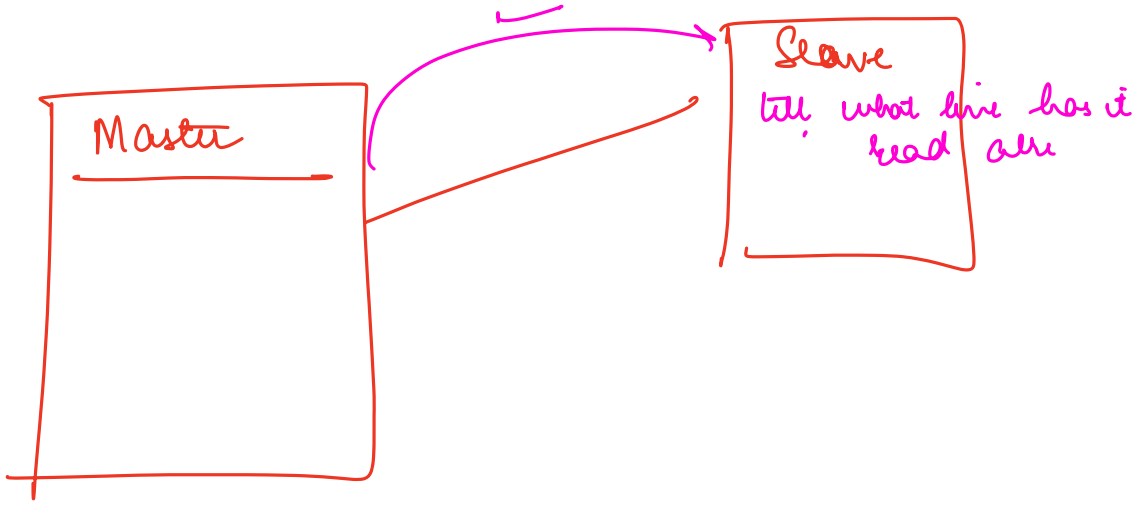
never consistent ✗

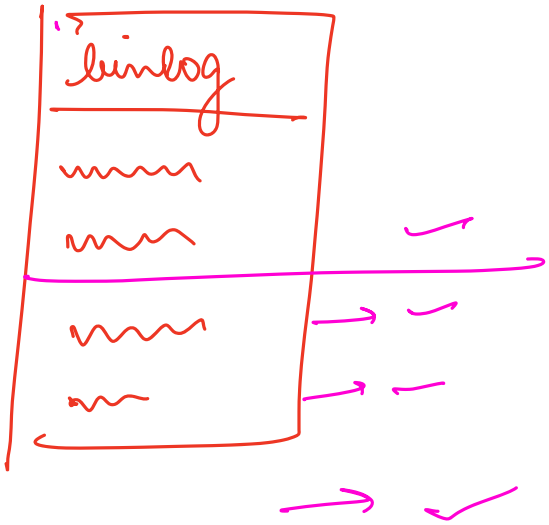
→ it can happen that all replicas are never in sync

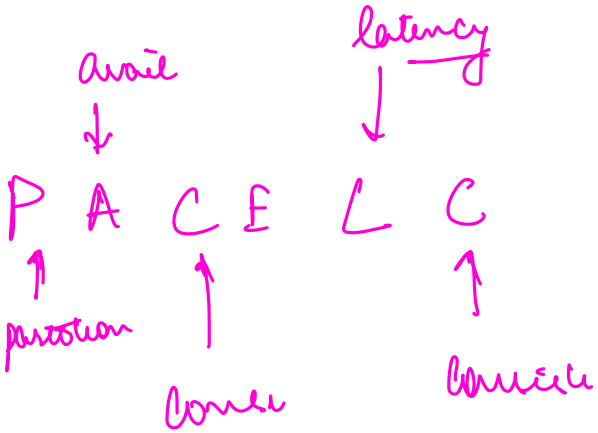


✓ —

→ Multi Master Setup


Master
Slave
till what bin log has it read also


bin log


avail
latency
P A C E L C
partition
Consi
Consiste