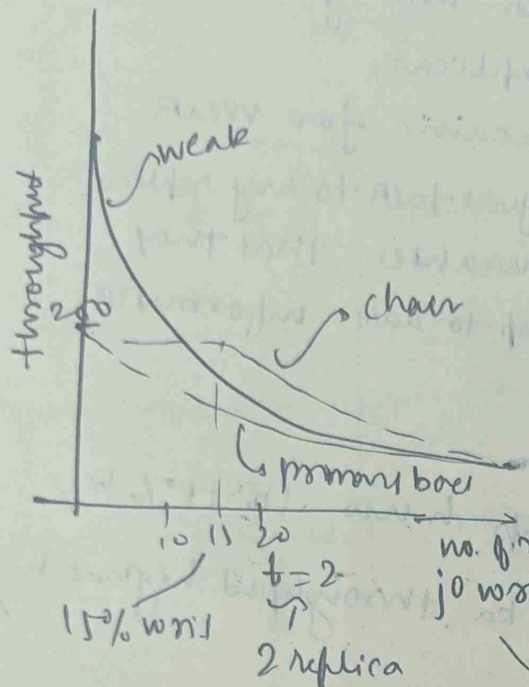
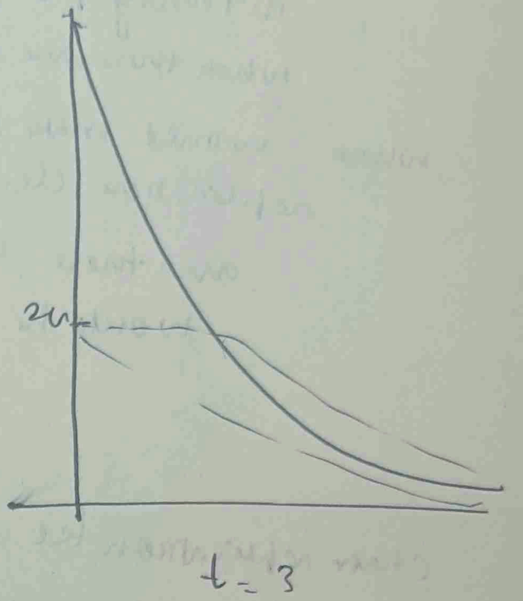
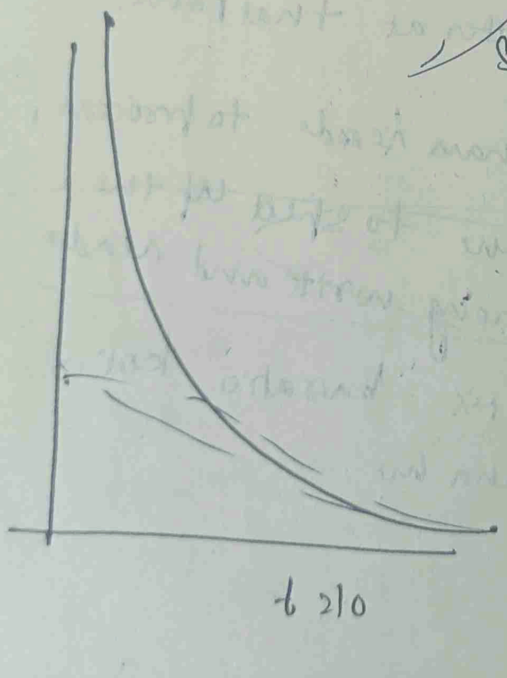
Lec12 pause for breath.





Similar sorta graph.



We can notice that for weak replication,

its throughput gets higher and higher.

when there are more replicas,

which would make sense because for weak

replication client can just talk to any replica.

and there are no guarantee that they

would be getting up-to-date information.

Chain replication ke liye bump hai 10-15% h, -

~~So~~ jaha chain replication ka throughput highest h,

why would that be better at that point.

writes takes longer than reads to process,

So, if you would have ~~to split up the load~~ equally among write and reads

to shayad 10-15% ka ratio koi

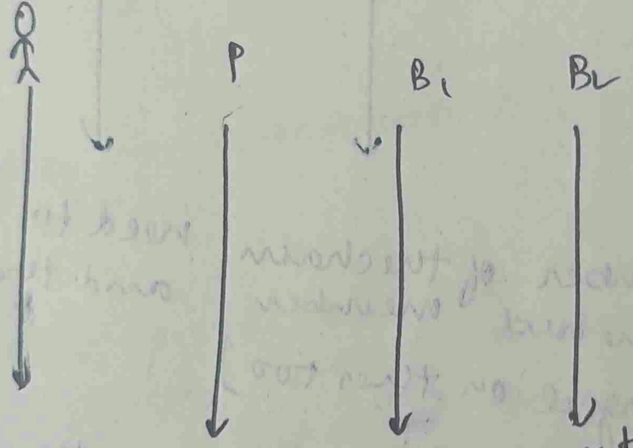
eqm laa raha hai

The main thing I want to point out is that,
Chain replication provides an advantage over primary backup replication.

And it turns out that 10-15% writes is actually a pretty common number for typical application.

How do you deal with failure?

Let's talk about primary backup.

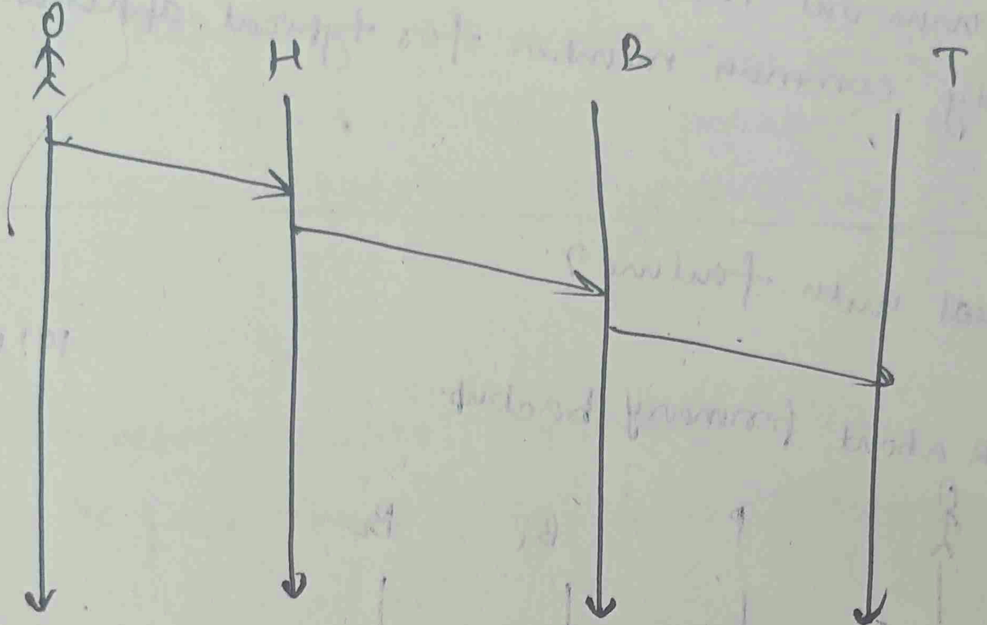


The Client has to know whom to talk to, like primary,
and if the primary were to fail and in that case somebody else would have to take over as the primary then somebody would have to be in charge to telling the Client that there's a new primary.

So, we will some sort of proxy, to whom client contact and it's the proxy job to know things and primary.

The replicas themselves have to know who is right

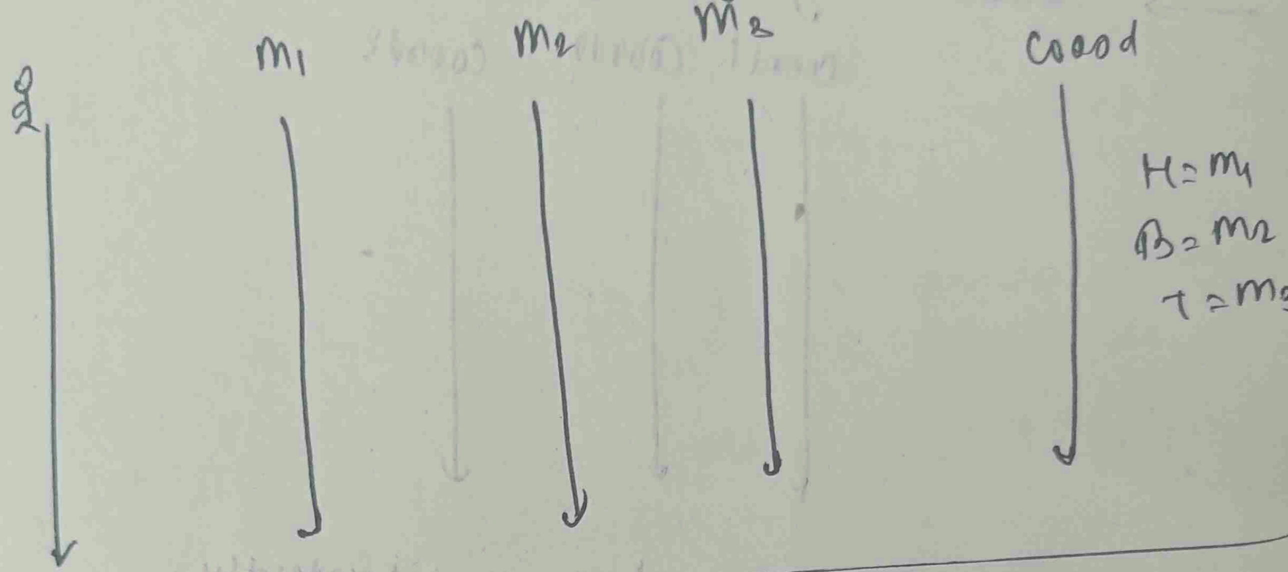
for eg, in chain replication



(Each member of the chain need to know who is the next member and they should agree on this too)

So, there should be presence of something, some kind of coordinator, whose job is to know who all the replicas are involved and what roles are they playing at a given time and keep everybody else informed of those roles.

The co-ordinator also should be able to detect failures and when a replica fail, it's the co-ordinator job to tell everybody else what the new configuration is going to be.



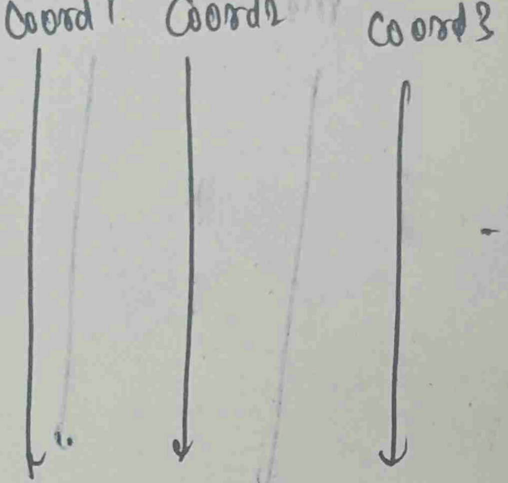
Q So what does the co-ordinator do, if the head process fails ?

→ then, the co-ordinator just has to take the head process out of the chain and it just make the successor the new head of the chain and tell the client who is the new head.

and if tail fails the removes tails and make the predecessor as the new tail.

Q) What if the co-ordinator fails??

⟶ ~~have a~~ maybe having a bunch of co-ordinators



which you hope fails independently,

but then these coordinators have to be consistent with each other,

So how do you keep them consistent with each other

↳ well are you going to implement primary backup or chain replication among those co-ordinators among them as well

and then have co-ordinators for that replicas and so on

... bhai ye ek infinite loop h fir do!

W

This problem leads to
"Consensus"

Remember, that the whole point of both chain replication and primary backup replication was to provide strong consistency btw replicas,

↓

If we wanna have strong consistency btw replicas, which is what there protocols give you and we also wanna be tolerant to faults

↓

then ultimately we end up being reliant on a consensus protocol.

"Consensus is hard and expensive"
↳ and that's one reason for not having strong consistency

"We will start with consensus after midterm"