https://www.youtube.com/watch?v=ueRwfeC_VRc replication started at 55:57 ## Replication - strongest type of consistency - full transparency - client can't tell its replicated (data always consistent no bad copies) - assuming crash model (node won't come back online) - ==Primary-Backup replication== 1976 - i.e. mongoDB, MySQL, PostgreSQL, Redis, HDFS, RabbitMQ, ZooKeeper, etcd, job schedulers, distrib. databases, etc. - pick primary node P and n backup nodes - when client writes on P: - broadcast to all replicas in parallel - each replica gets write committed and ACKs to primary - if all replicas ACK received, considered **committed** and **delivered** on primary - primary sends ACK to client - if replica fails (no ACK) - kicks replica forever (untrustworthy) - if primary fails - heartbeat between primary and replicas - no heartbeat (consensus) replicas detect and vote new leader - i.e. Raft leader election - most up-to-date replica wins vote (i.e. WAL=write-ahead log tracks all requests, their ids, and results, and replay when primary receives request) - request id = client token (hash timestamp etc.) - must store WAL on another node and replicated backup WAL node - if prioritize availability over correctness (client doesn't care if inconsistent/old data, just wants response) - then okay to omit WAL, CAP theorem

```
- when client reads --> backups not used, primary only
- if anything fails, client retries entire process
    - variations in alg: primary retries to replicas (as long as
idempotent), accepts some % of replicas ACKS received, etc.)
- pros/cons?
    - **fault tolerance:** decent recovery
    - **data locality**: bad. no locality just 1 primary
    - **load balancing**: bad. primary does all requests
    - primary is bottleneck esp if bandwidth, cpu-intensive
```

- ==chain replication== 2004
    - less common, more niche/research, DynamoDB
    - no primary/replicas, now head (writes), tail (reads), middle
    - when client writes
        1. client writes to head (which receives all requests)
        2. head sends to next in chain (M1)
        3. M2->M3->….->T (tail)
        4. once tail receives, considered committed (all replicas received)
        5. T–ACK–>client
    - when client reads
        - directly reads from tail, and tail responds with 3
    - pros/cons?
        - **fault tolerance:** decent recovery
        - **data locality**: bad. no locality just 1 primary
        - **load balancing**: better, one node reads, one node writes
    - if node fails, like removal from linked list.
        - ![[Pasted image 20241103123203.png]]
        - if head dies: ![[Pasted image 20241103123253.png]]
            - elect new head from middle nodes, M2
            - redirect M1 –> T and redirect new head H* –> M1
            - ![[Pasted image 20241103123322.png]]
        - if tail dies: same idea, elect tail, i.e. M1
            - ![[Pasted image 20241103123504.png]]
            - ![[Pasted image 20241103123615.png]]
        - if middle node dies, simply skip over the missing link
            - ![[Pasted image 20241103123708.png]]
- ==chain vs primary-backup==
    - both use 2 RTT
    - if constant latency & all nodes same network:
        - L = one-way time from A–>B
        - Primary-Backup:
            - writes = 4L: L(client->P) + 2L(all parallel replicas + ACKs) + L (P–ACK–>client)
            - reads = 2L: L(client->P) + L(P->client)
        - Chain:
            - writes = (N+2)L: L(client->H) + NL(sequential chain) + L(T–ACK–>client)
            - reads = 2L: L(client->T) + L(T->client)

            - the more nodes, the more latency, the more redundancy

CDNs

- are example of replication
- caching: copies everywhere around world for quick access
    - first lvl replication for locality ![[Pasted image 20241103124616.png]]
- failover: if local node crash, CDN schedules backup takeover
    - ![[Pasted image 20241103124541.png]]
    - second lvl of replication just for backups
    - also used for load-balancing among edge servers
    - ![[Pasted image 20241103124710.png]]
    - re-routing (CS 118)
- 5:00 https://youtu.be/qpfmF4g24Oo?si=HOqYlNJe8QhTJkKy&t=287
-