Agenda

DynamoDB:

- Review of old ideas
  availability, network partitions, eventual consistency, app-specific conflict resolution

- New ideas:
  - anti-entropy with merkle trees
  - gossip
  - quorum consistency
  - tail latency.

Availability

- Every request will receive a response.

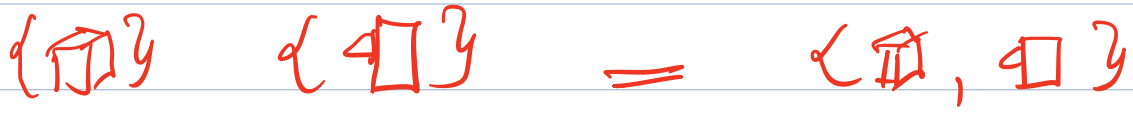
Network Partitions:

- some machines can't talk to each other
- temporary & unintentional

Eventual consistency:

- Liveness property
- Replicas eventually agree if updates stop arriving

Application specific conflict resolution



What was the Dynamo shopping cart bug?
- Deleted items can reappear in the cart. due to union operation.

If not conflict resolution, dynamo uses last-write-Wins

If writes are commutative ⇒ strong convergence.

Vector clocks in dynamo are used to avoid deletion anamolies.

Dealing with replicas that disagree.

Finding out that replicas disagree

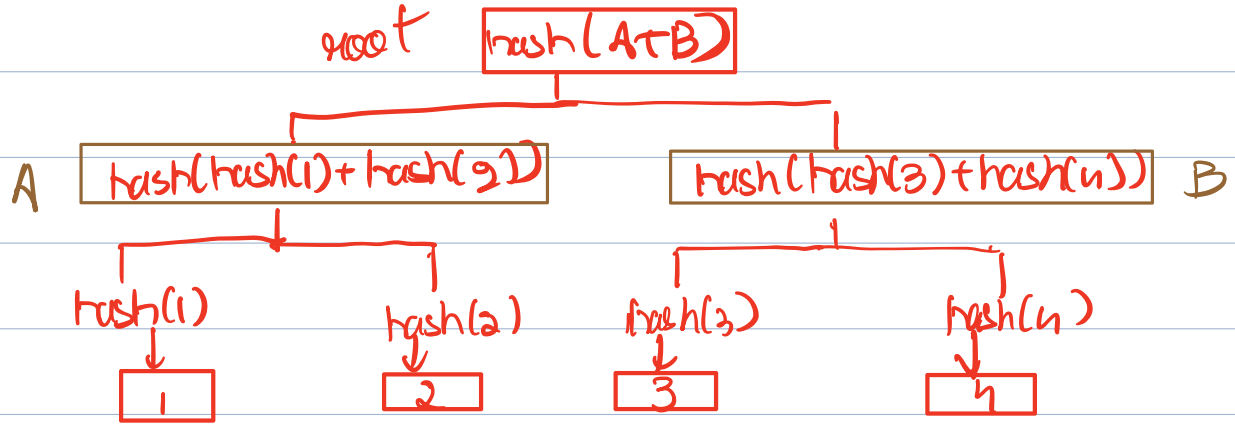
Key-Value store

- Anti entropy - Resolving conflicts in application state
- gossip - resolving conflicts in view state
  - ↳ Synonyms in general, but have different meanings in Dynamo.

↓ who's up?

How does dynamo minimize data transfer cost?
with Merkle Trees. aka Hash trees

Merkle Trees.



Replica 1

x=3 z=5

y=5 q=7

Replica 2.

x=3 z=5

y=4 q=7

Compare hash values at each node. If they disagree, the nodes below has disagreement

Reduces time to parse tree.

Quorum Consistency

How many replicas should a client talk to?
Quorum systems let you configure this!

3 parameters

N - Number of replicas

W - "write quorum" - How many replicas have to ack a write op

R - "read quorum" - How many replicas have to ack to a read op.

For settings, N=3, W=3, R=1
Would this give you strong consistency?

Looks very similar to Primary Backup or chain replication

This approach is called Read-One-Write-All (ROWA)

- If there are concurrent writes (because of another client), the order in which these

writes arrive will determine the value stored in each replica

- Primary Backup involves all writes going to primary, which decides the order of writes, thereby guaranteeing strong consistency. PB has a synchronization point
- there is no such point available in the current scenario. Hence, it is possible that replicas have different values. Therefore, strong consistency is not guaranteed.

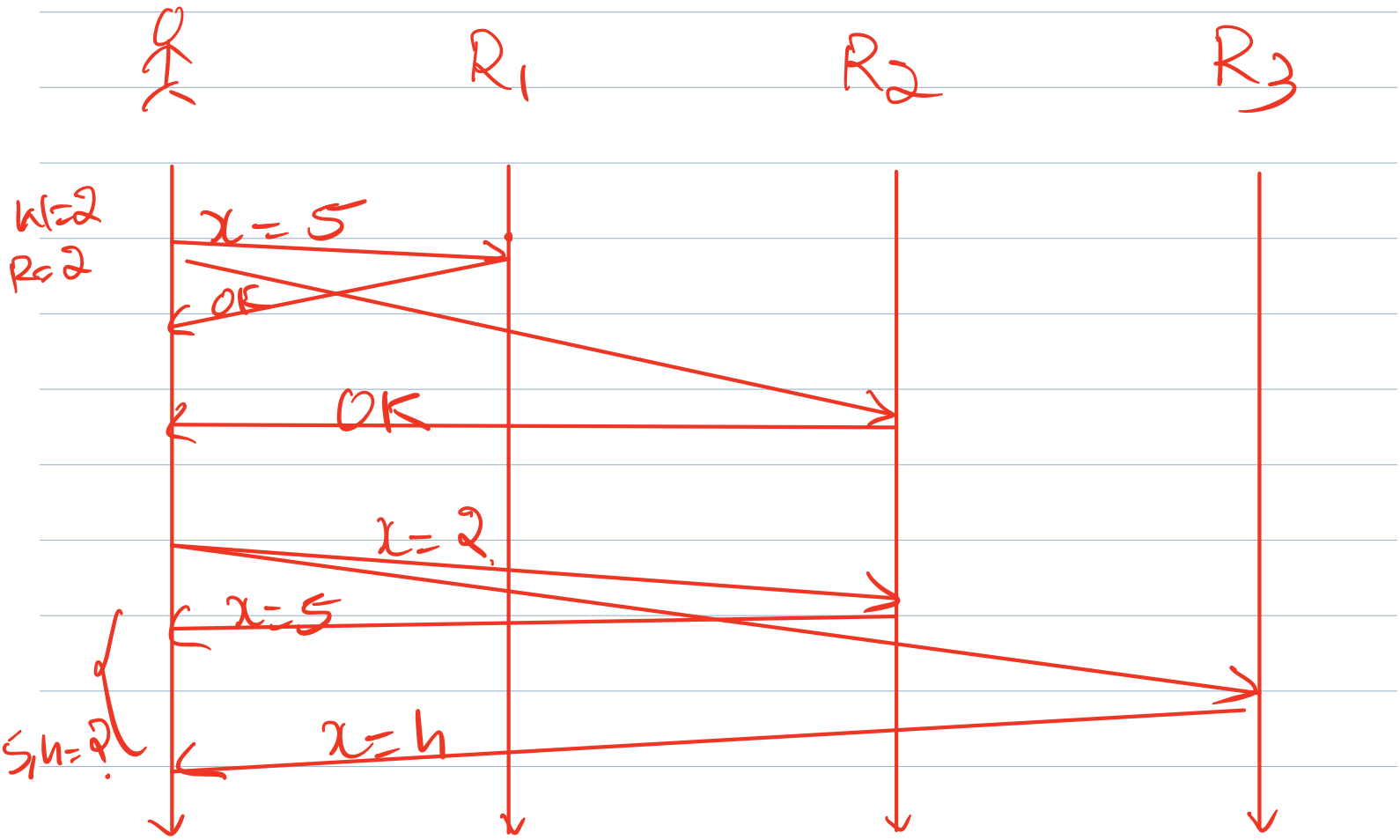
Another reason this is bad:

- If a node crashes / there is a network partition, you cannot do any writes !!!
- Until failure is detected & he remove the failed node, all write ops will be stuck!
- Also, writes are slow since all replicas need to ack a write op.

In dynamo, if you have 3 replicas, suggestions are:

N = 3

W = 2
R = 2



Generally; (R+W > N)

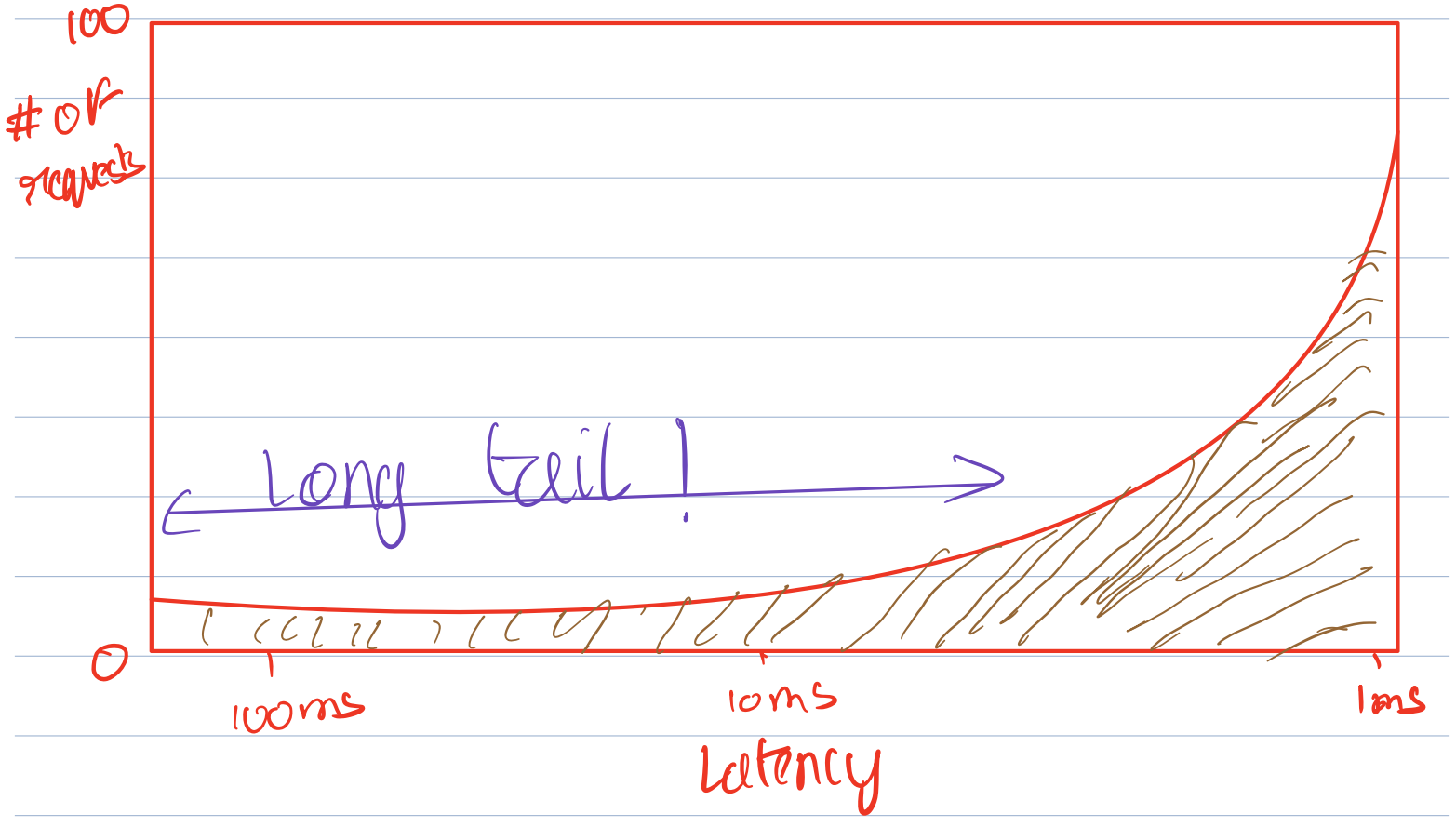
If you have R+W > N, read quorums will intersect with write quorums.

Cassandra :- Supports quorum consistency

AFAIK, dynamo enforces Read-Your-Writes (RYW), by passing metadata containing which replicas received your write. This metadata is forwarded to the next read request to try & read from the same replicas!!

Tail Latency

Latency: Time between start & end of a single action.



- Metrics like "average latency" ignore

slow requests

- latency at 99.9% percentile helps account for slow clients, & make a better system.

∴ **Tail latency at high end of distribution**

99.9% ⇒ what is the time for the 2nd slowest request?