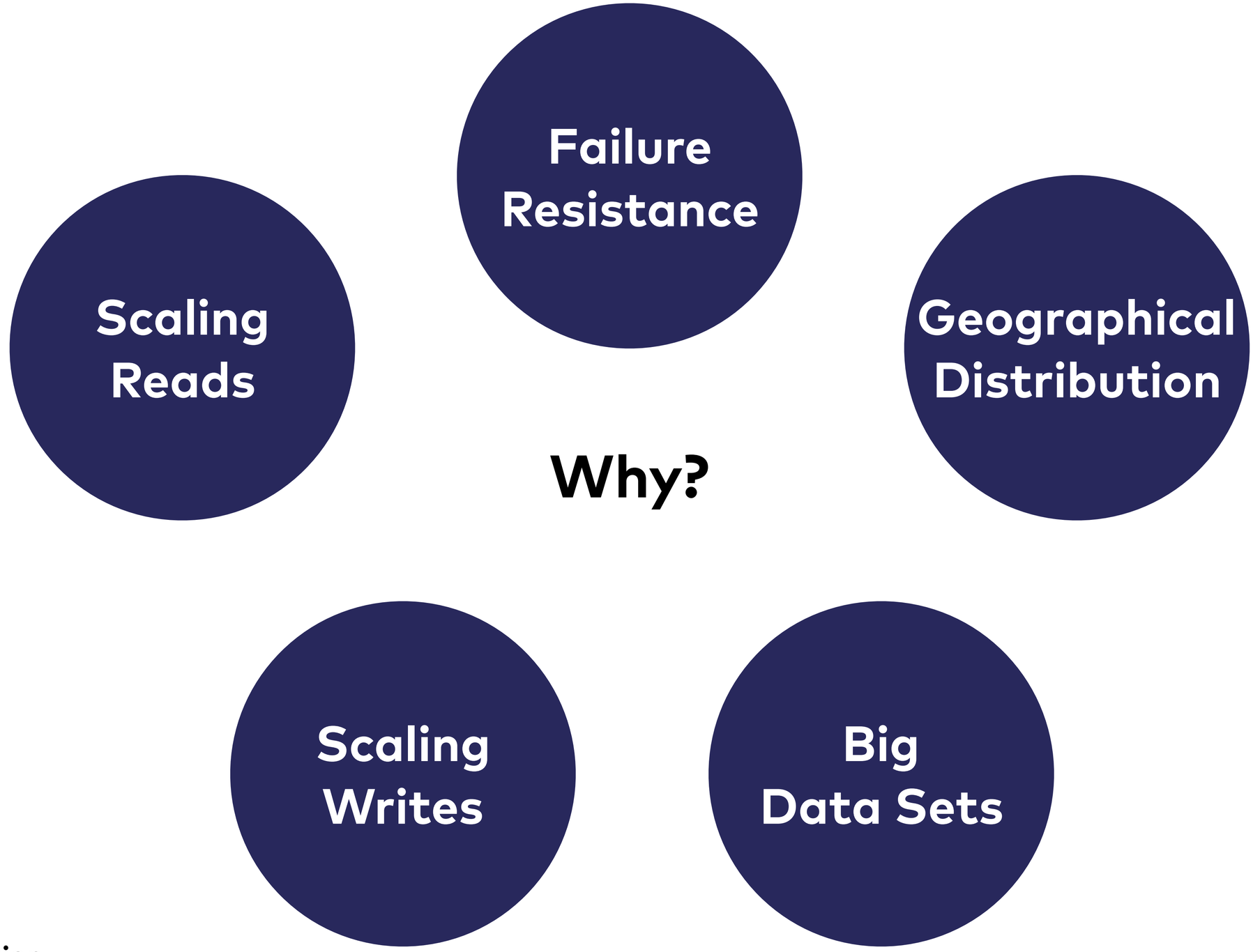
Scaling Data

Lucas Dohmen
Senior Consultant @ INNOQ


Failure Resistance
Scaling Reads
Geographical Distribution
Why?
Scaling Writes
Big Data Sets

Lucas Dohmen

- Senior Consultant at INNOQ
- Everything Web & Databases
- Previously worked at ArangoDB
- http://faucet-pipeline.org

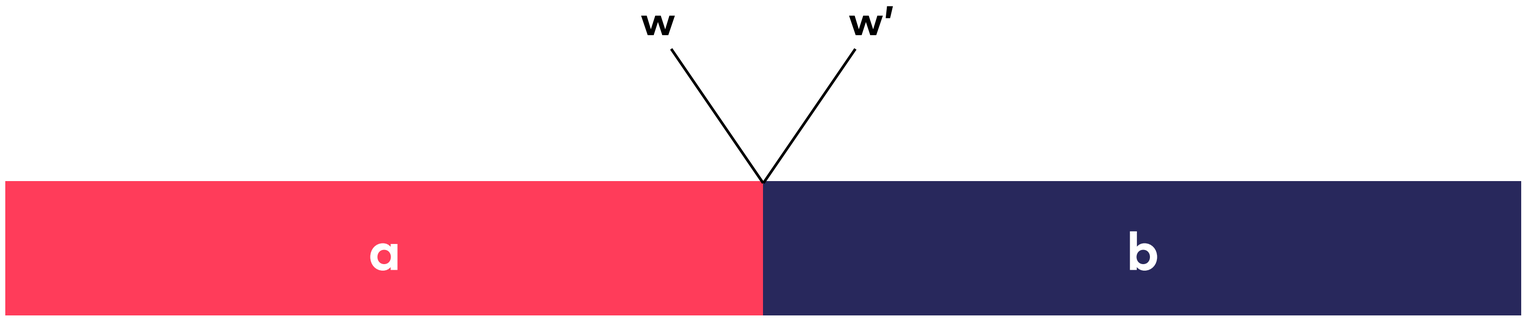
Structure

1. Consistency
2. Scaling
3. Trouble

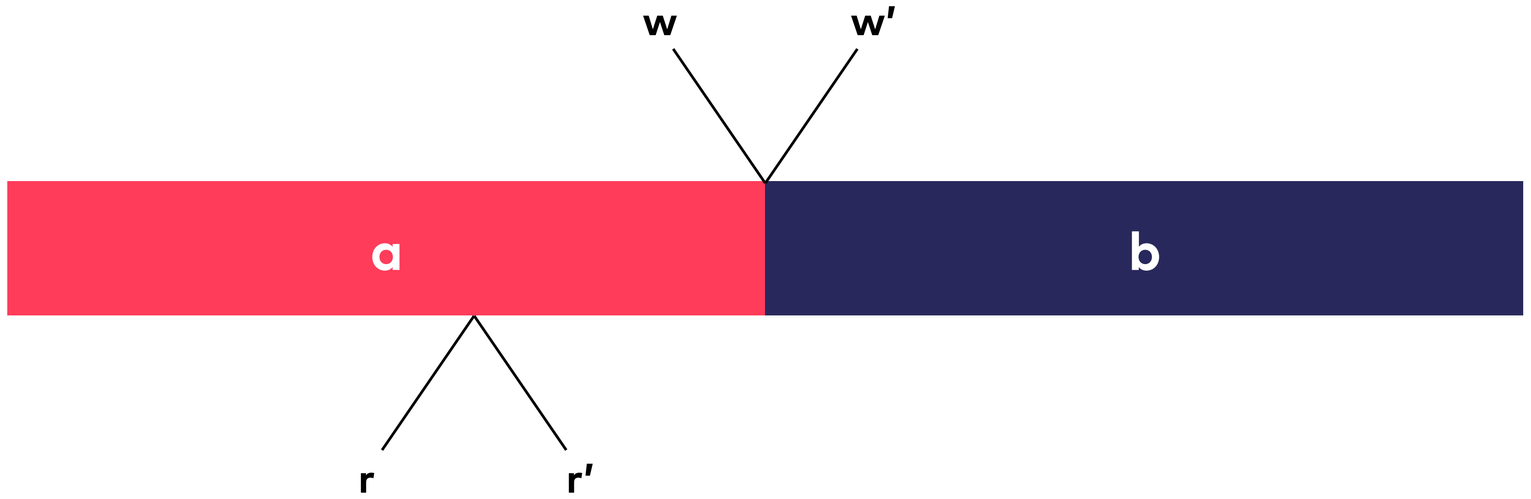
Part 1: Consistency

无

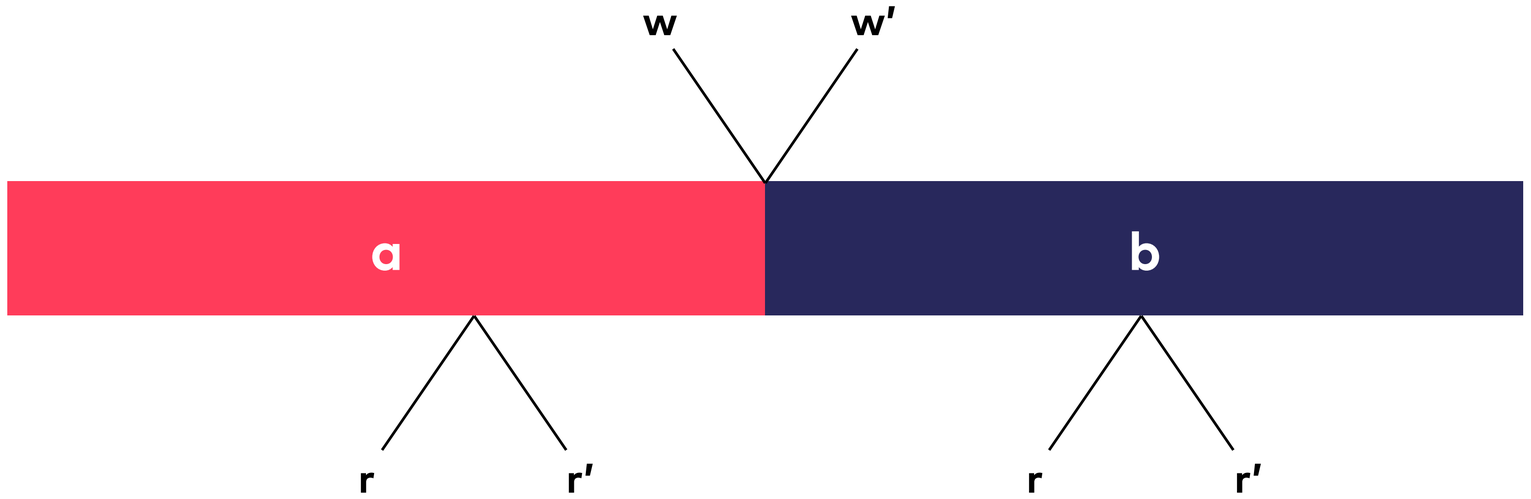
Linearizable



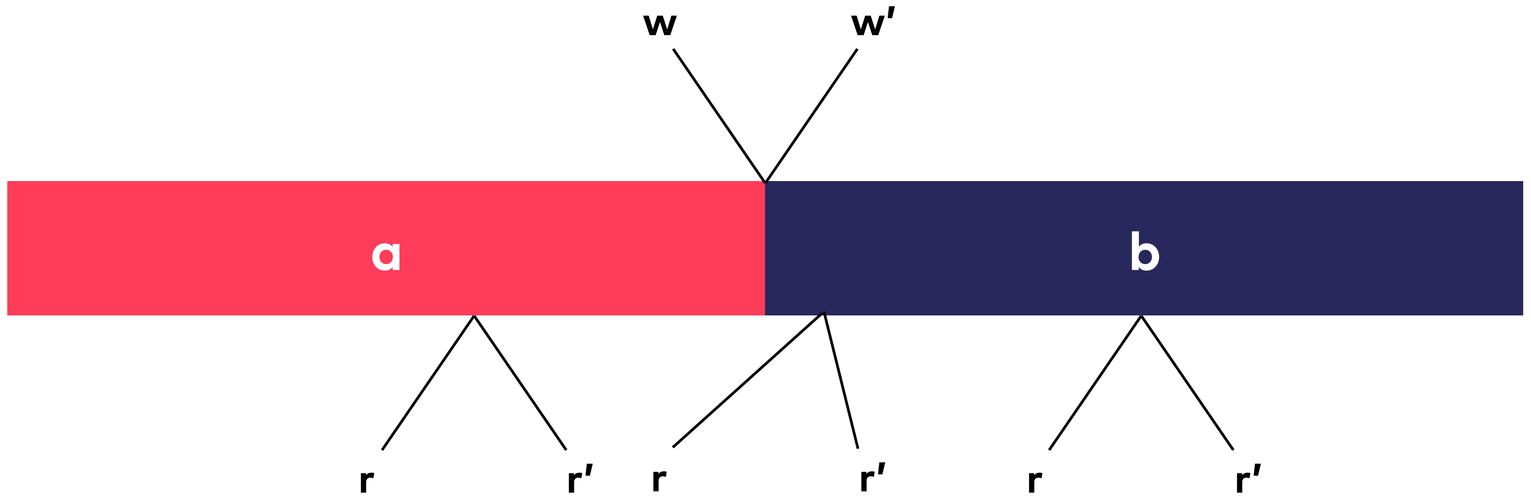
Linearizable



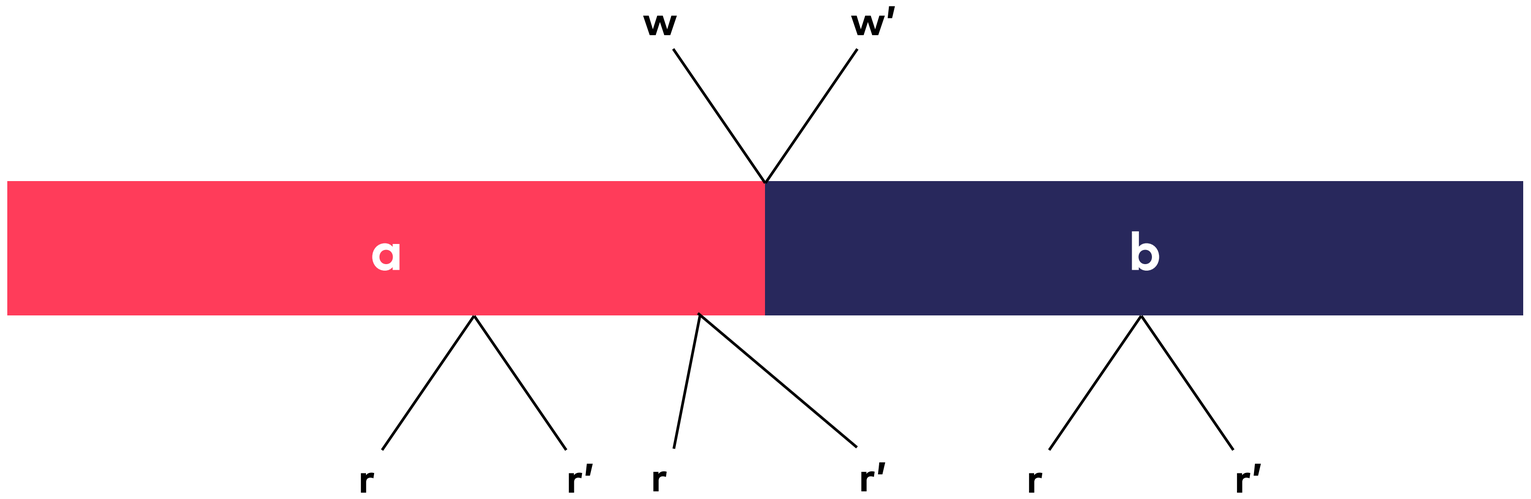
Linearizable



Linearizable



Linearizable



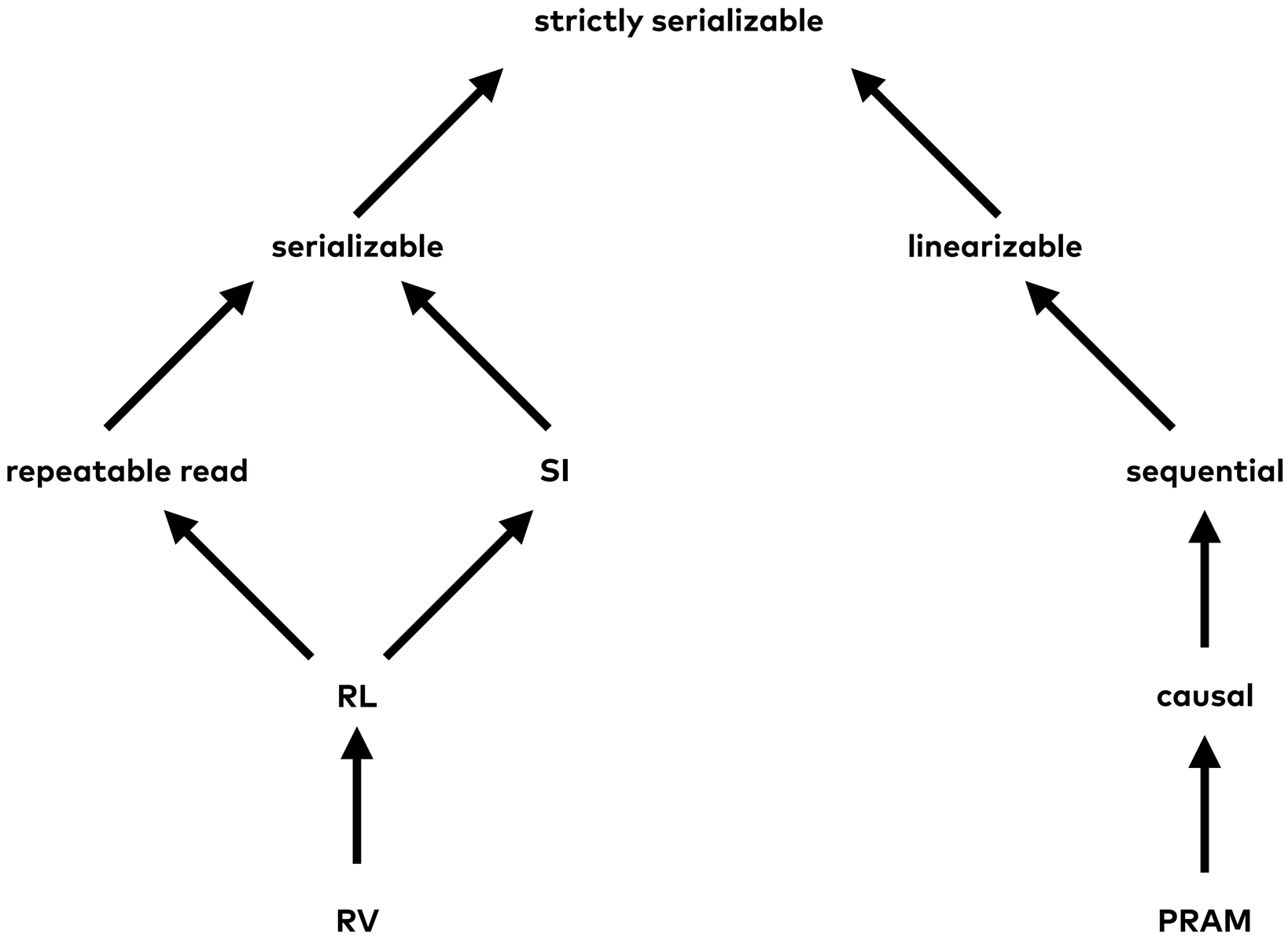
Consistency Models: Which histories are valid?

Read() => a
Write(b)
Read() => b
Read() => b

Read() => a
Write(b)
Read() => a
Read() => a

Read() => b
Write(b)
Read() => b
Read() => b

https://aphyr.com/posts/313-strong-consistency-models

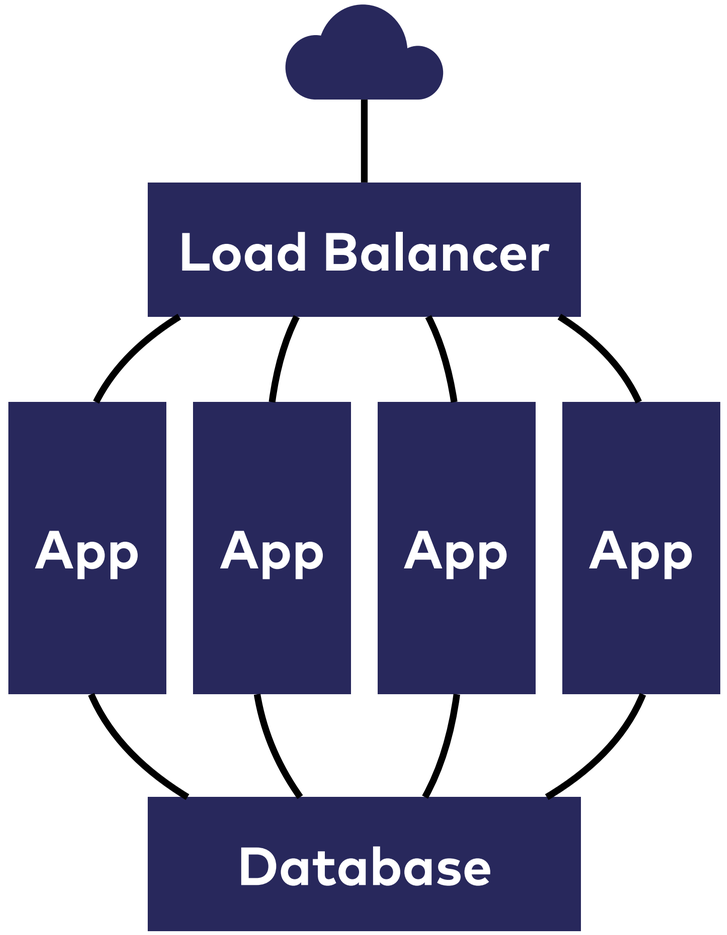


Highly Available Transactions: Virtues and Limitations – Bailis et al.

Part 2: Scaling

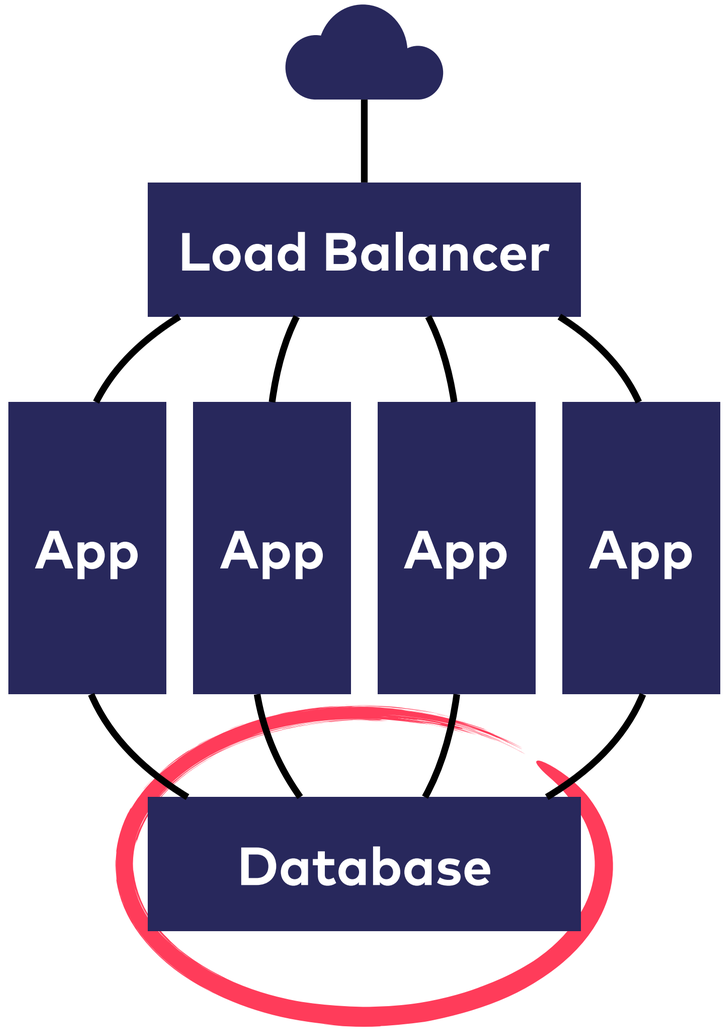
How do we scale web applications?

- Share nothing between application servers
- Put behind a load balancer
- Add servers



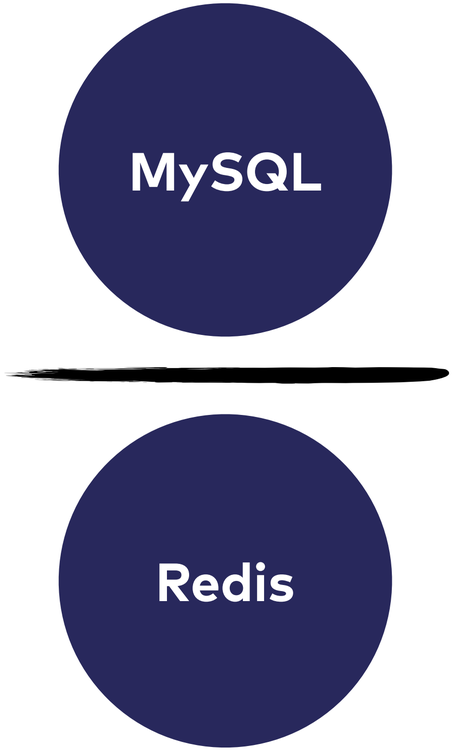
How do we scale web applications?

- Share nothing between application servers
- Put behind a load balancer
- Add servers



Share Nothing for Databases?

- Possible & Underused
- Separate databases for separate data
- If we need to join data, we need to join in the application



Replication

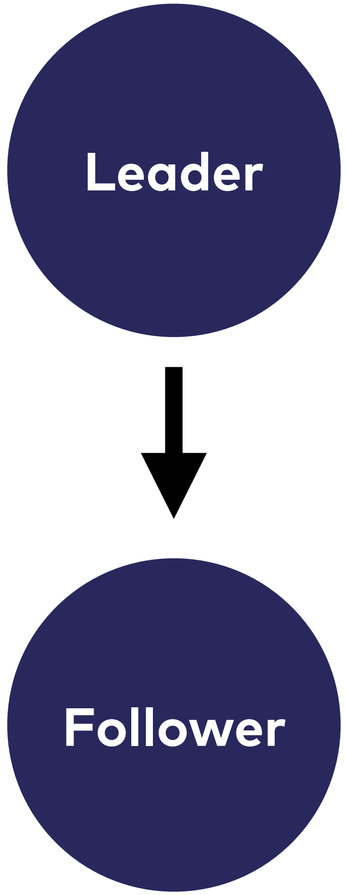
Replication

=

Same data on multiple nodes

Single Leader

- Failover
- Read scaling
- No write scaling



Sync or Async Replication?

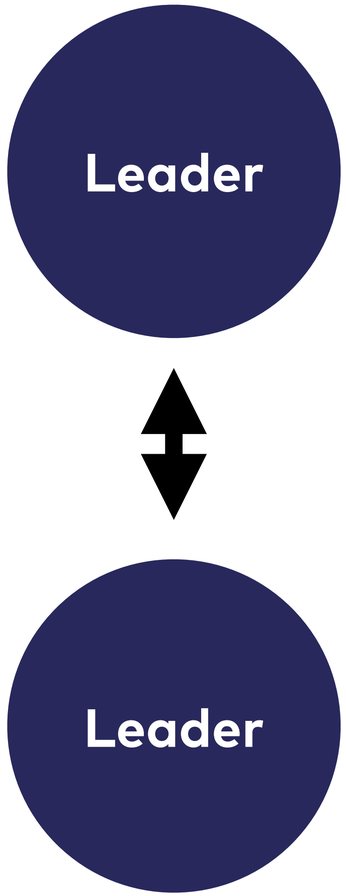
- Trade-off between consistency & speed
- Sync: Every follower we add decreases performance
- Async: If our leader dies and the replication is not done, we have lost acknowledged data

Examples

- Redis
- MariaDB
- PostgeSQL
- MongoDB

Multi Leader

- Failover
- Read & write scaling



Write Conflicts

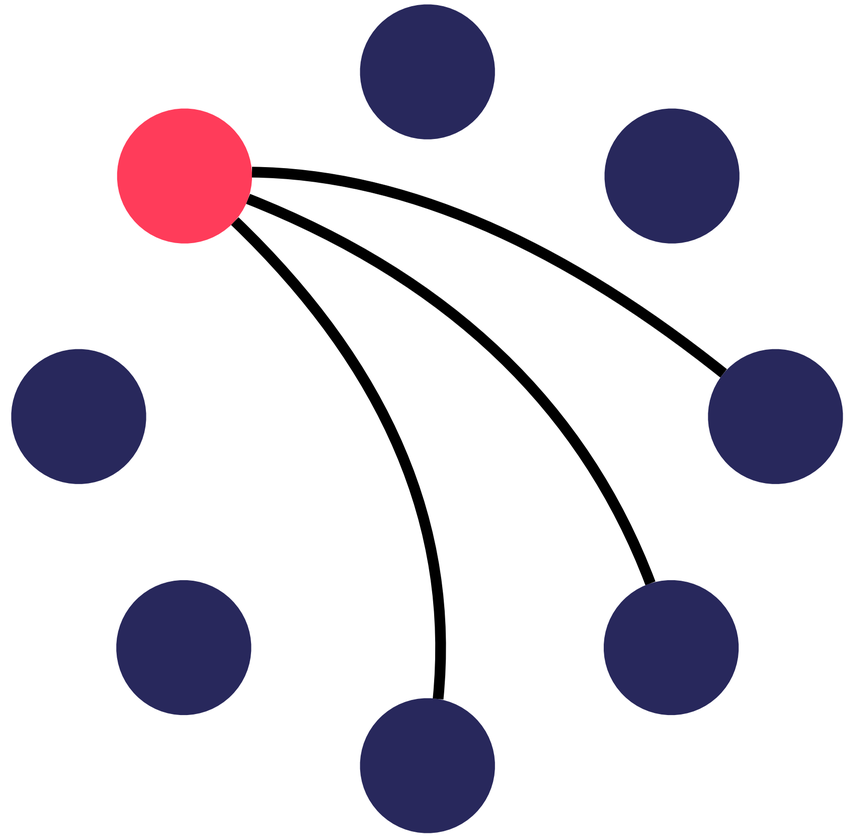
- Two leaders can accept a conflicting write
- We usually resolve them when reading
- Do we have all information we need to resolve a conflict at read time?

Examples

- CouchDB
- Percona Server for MySQL
- ArangoDB

Leaderless

- Failover
- Read & write scaling



Quorum

- Clients write to multiple nodes at once
- When more than n nodes acknowledged the write, the write is successful (n is the write quorum)
- When we read, we read from m nodes (m is the read quorum)

Examples

- riak
- Cassandra
- aerospike

Sharding

Sharding

=

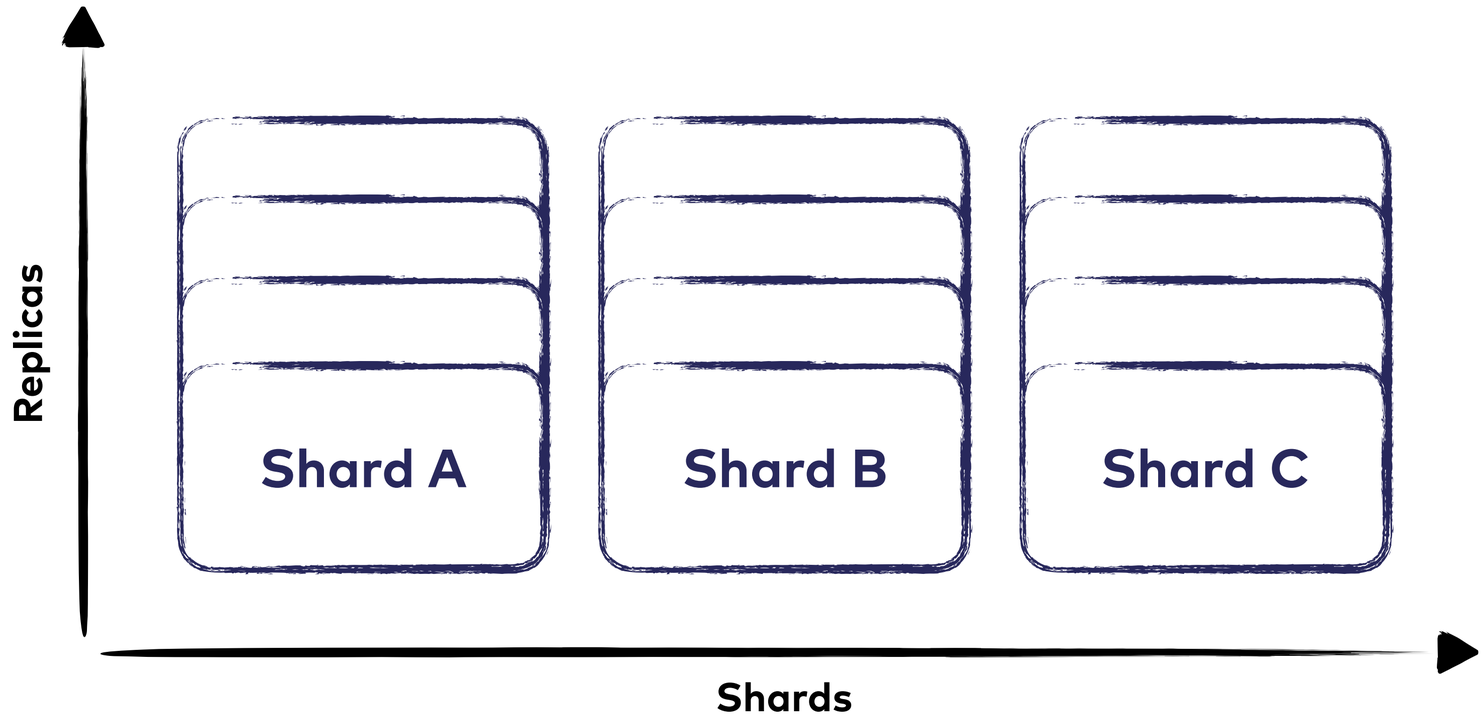
Each node only has part of the data

Sharding by Primary Key

Sharding by Hashed Primary Key

- Equal distribution to all shards

Combining Replication & Sharding



Part 3: Trouble

SHARED MUTABLE STATE IS EVIL

Clocks are monotonic & synchronized

leap seconds

Clocks are monotonic & synchronized

NTP fails

leap seconds

Clocks are monotonic & synchronized

NTP fails

leap seconds

Clocks are monotonic & synchronized

NTP Sync ⇒ Going back in time

NTP fails

leap seconds

NTP is an estimation

Clocks are monotonic & synchronized

NTP Sync ⇒ Going back in time

NTP fails

leap seconds

NTP is an estimation

Clocks are monotonic & synchronized

NTP Sync ⇒ Going back in time

DO NOT USE WALL CLOCKS FOR ORDERING

Solution:
Vector Clocks

The network is reliable

Problem	IP	TCP
Reordered Messages	✘	✔ (Sequence Numbers)
Lost Messages	✘	✔ (ack)
Duplicated Messages	✘	✔ (Sequence Numbers)
Delayed Messages	✘	✘

The network is reliable

The network is ~~reliable~~

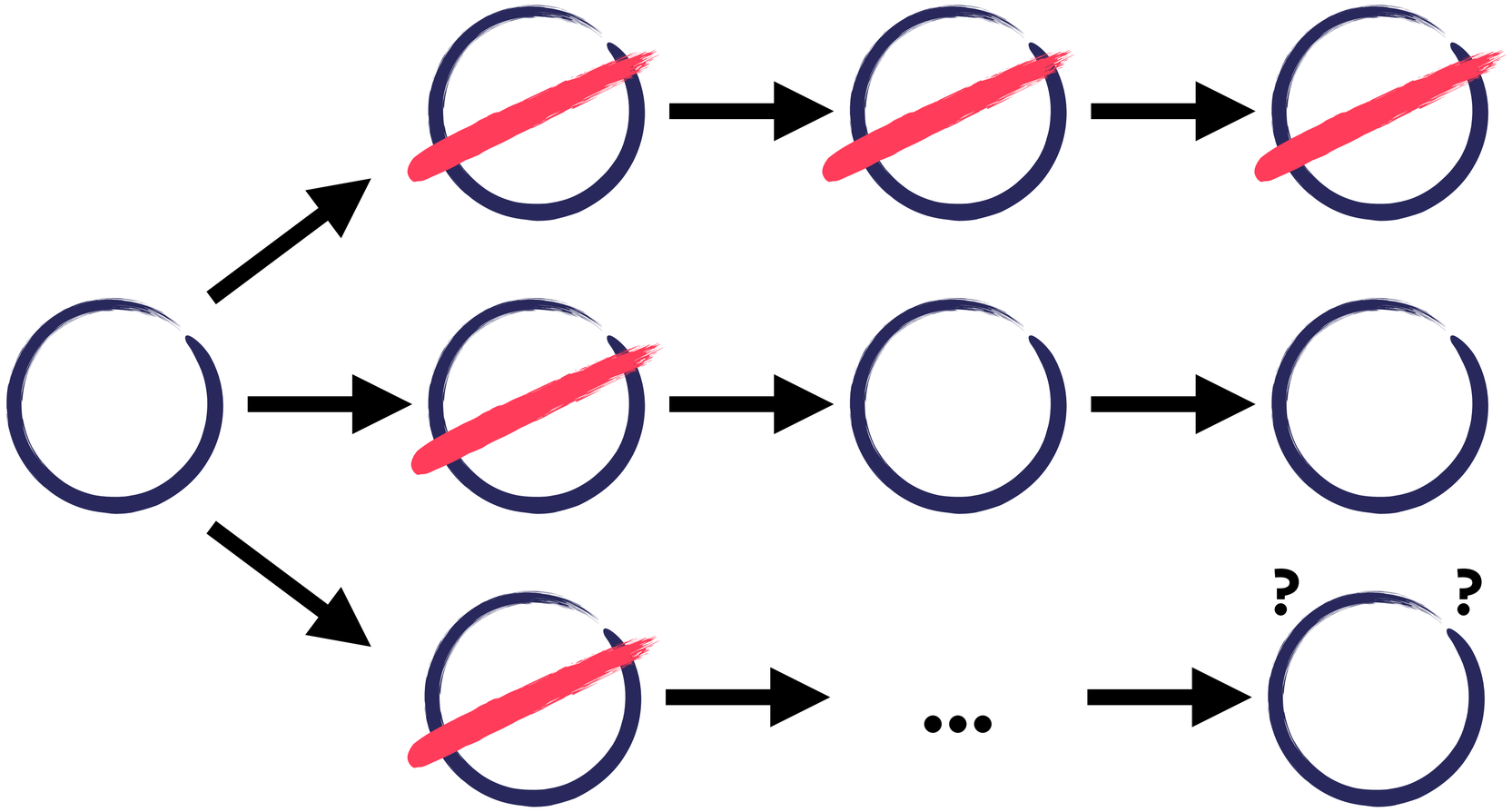
packages can take a
looooong time

The network is ~~reliable~~

packages can take a
looooong time

the network can fail
partially/entirely

The network is ~~reliable~~



Node Failure

Node Recovery

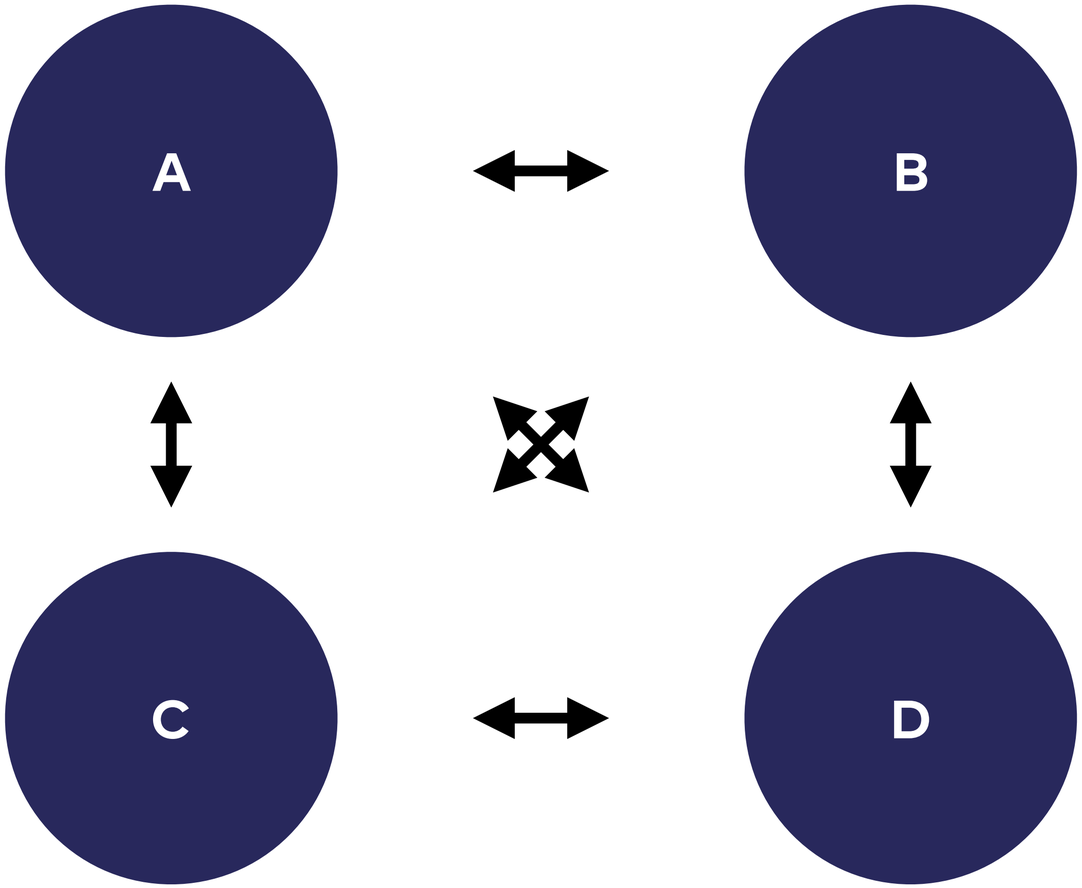
Crash Amnesia

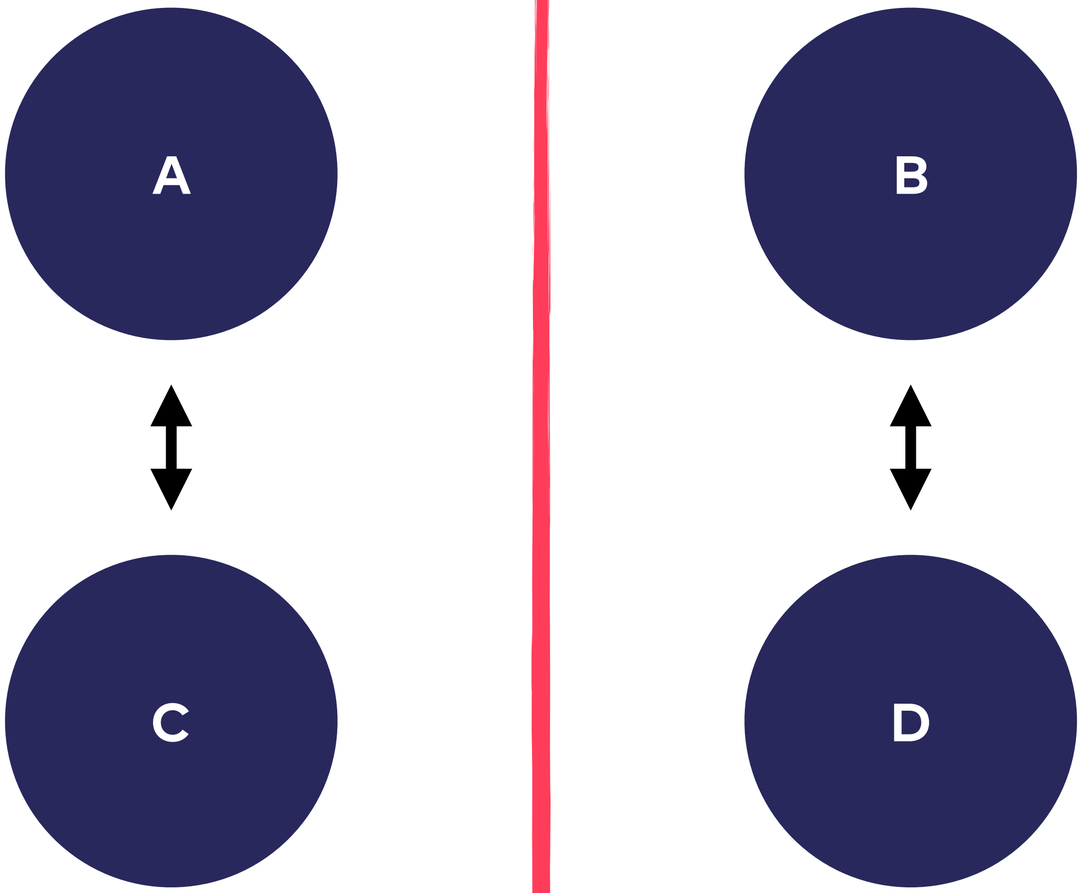
Availability

Availability vs. Consistency

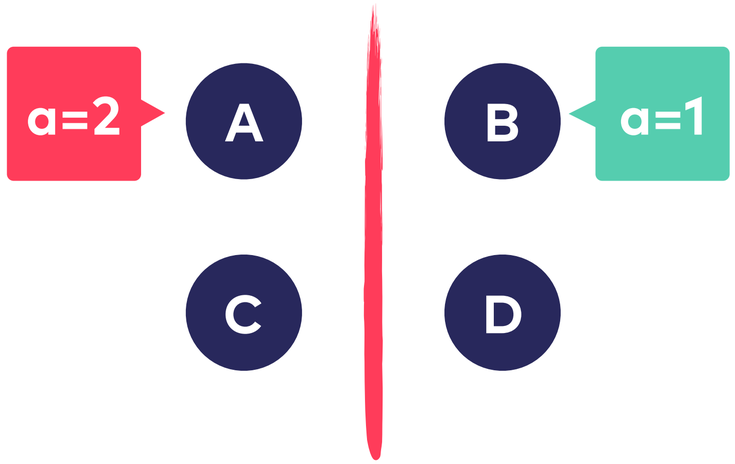
YOUR
NODES
WILL
FAIL

YOUR
NETWORK
WILL
FAIL


A
B
C
D


A
B
C
D

You have two choices

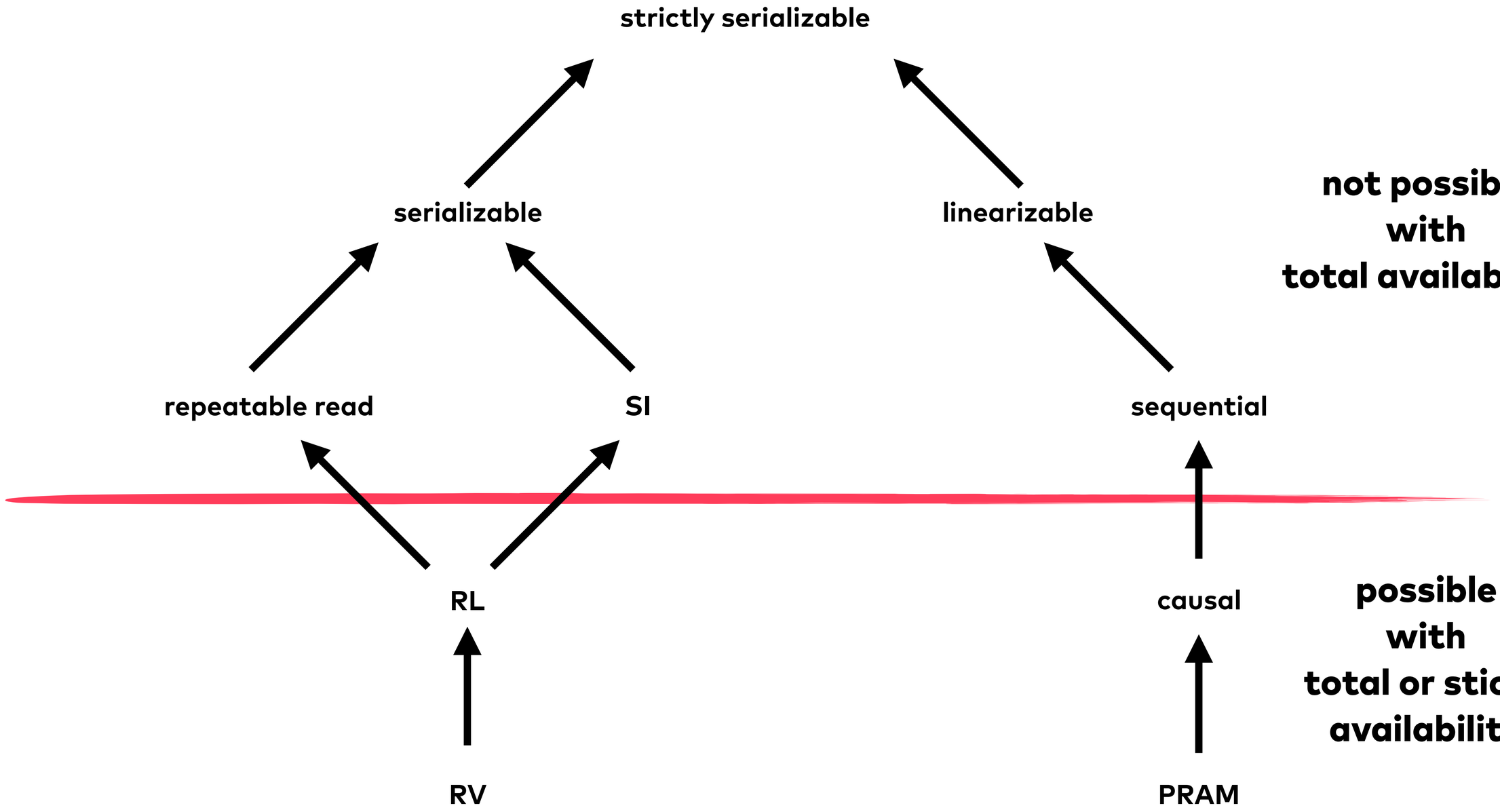


- Stop taking requests
- Not available, but consistent

CP

- Continue taking requests
- Available, but not consistent

AP



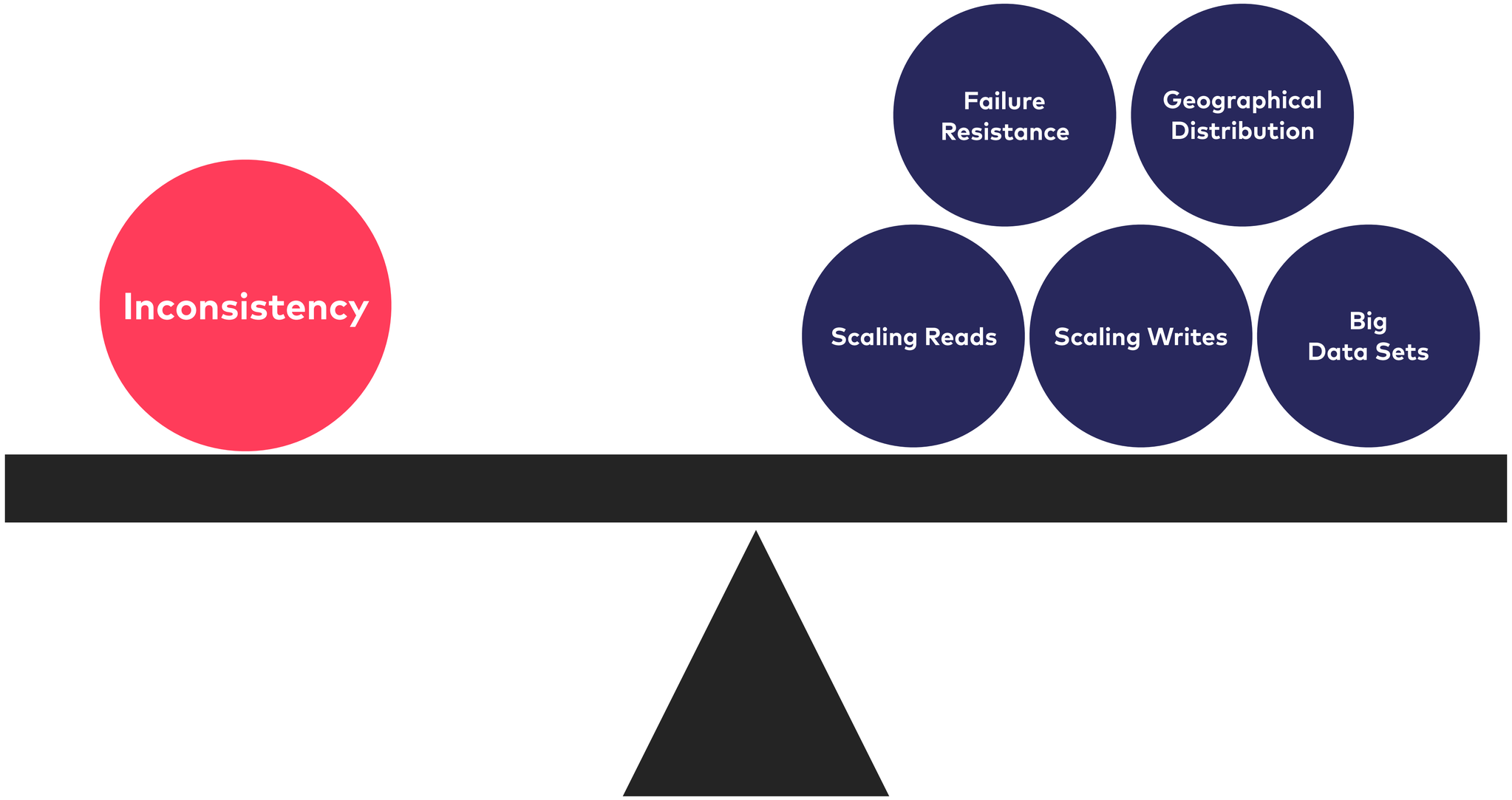
Highly Available Transactions: Virtues and Limitations – Bailis et al.

Wrap-Up

Remember!

- Nodes will fail
- The network will fail
- Clocks aren't reliable

What are your requirements?



Thank you!

- @moonbeamlabs on Twitter
- Photo Credit
  - Slide 5: Shoot N' Design on Unsplash
  - Slide 11: Andy Hall on Unsplash
  - Slide 30: Hermes Rivera on Unsplash