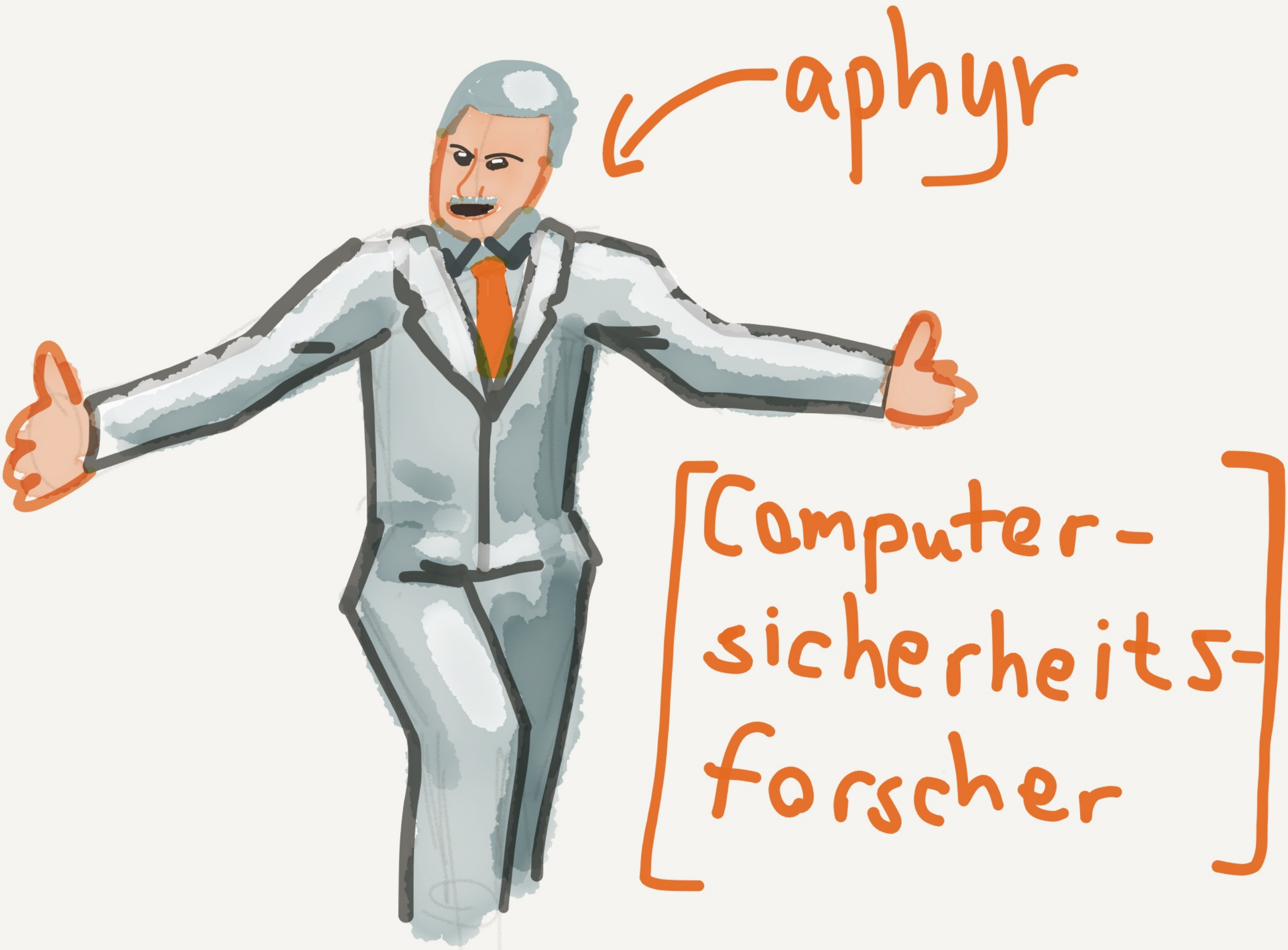
Jepsen

WHAT EVEN ARE COMPUTERS


aphyr
[Computer-
sicherheits-
forscher]

Public API {

} API code

Ruby {

DBs

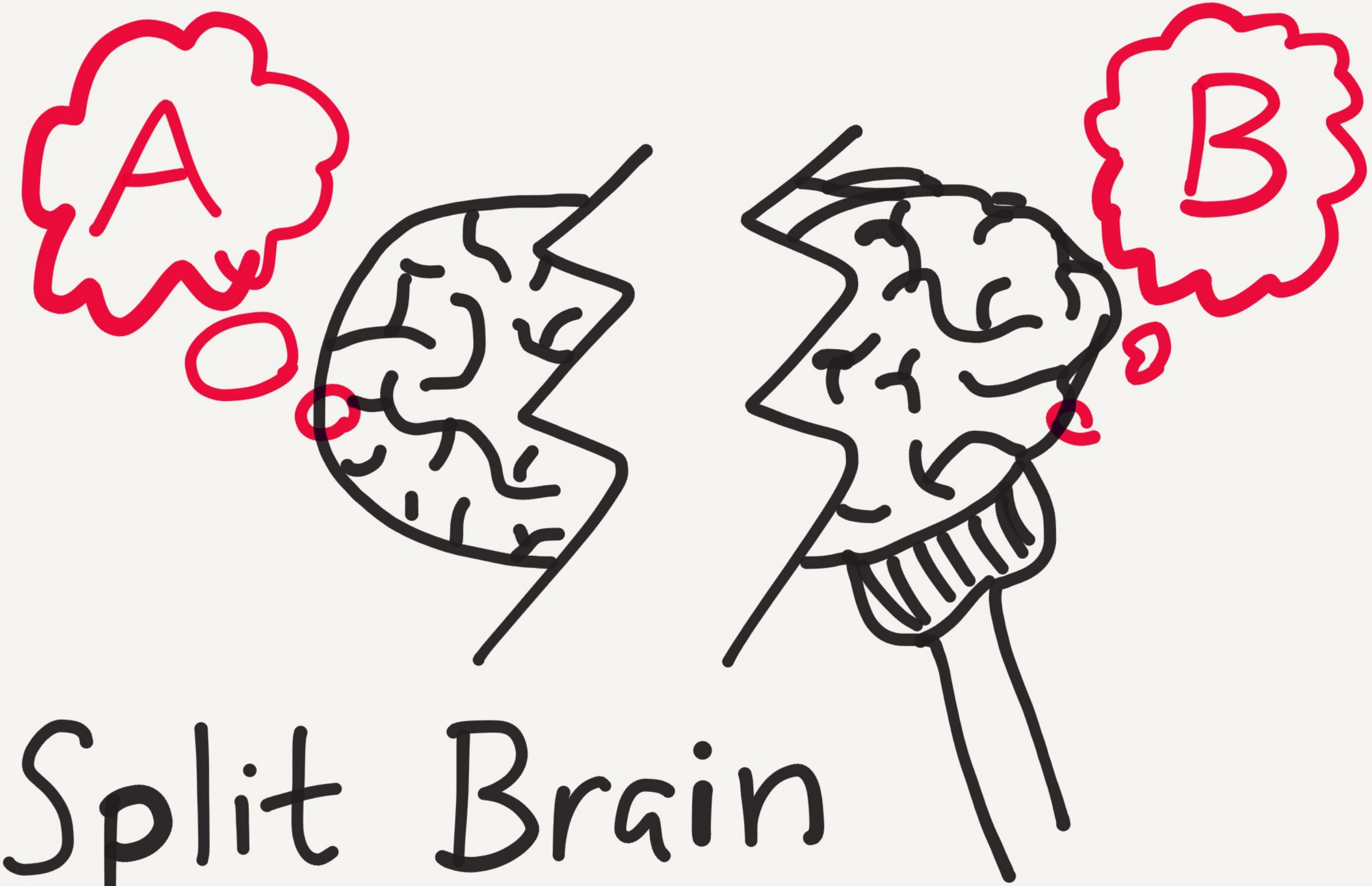
Every thing is fine!

Databases!

Queues!

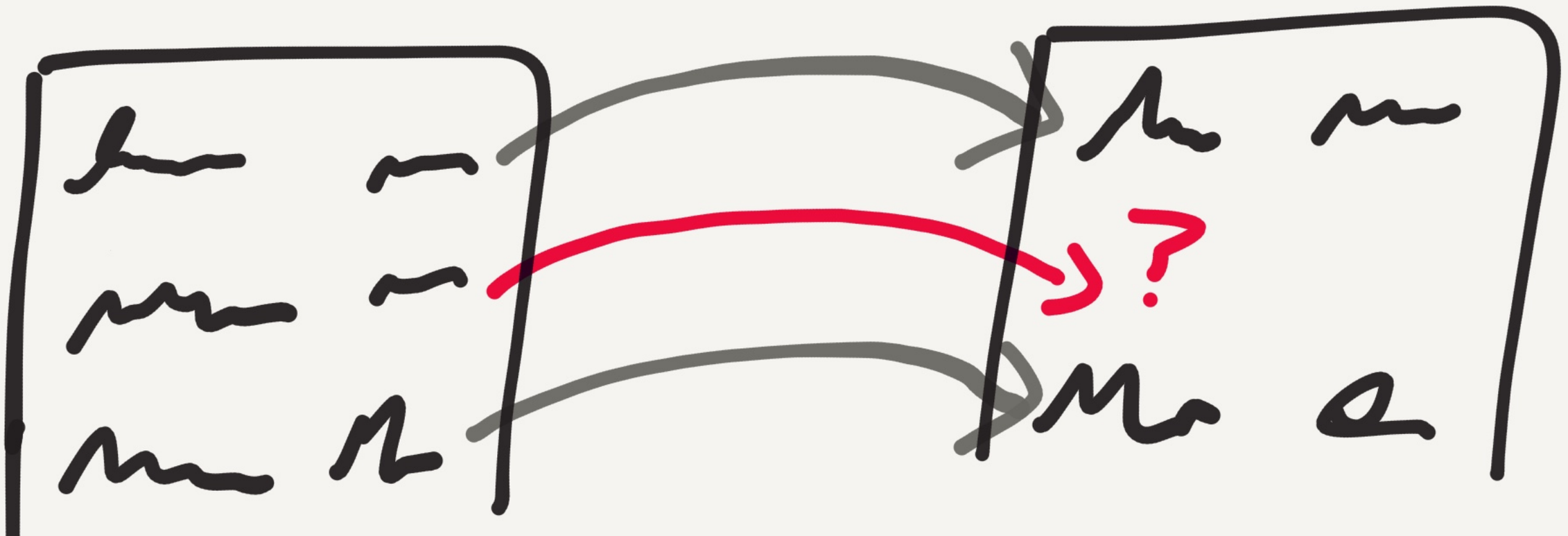
Discovery!

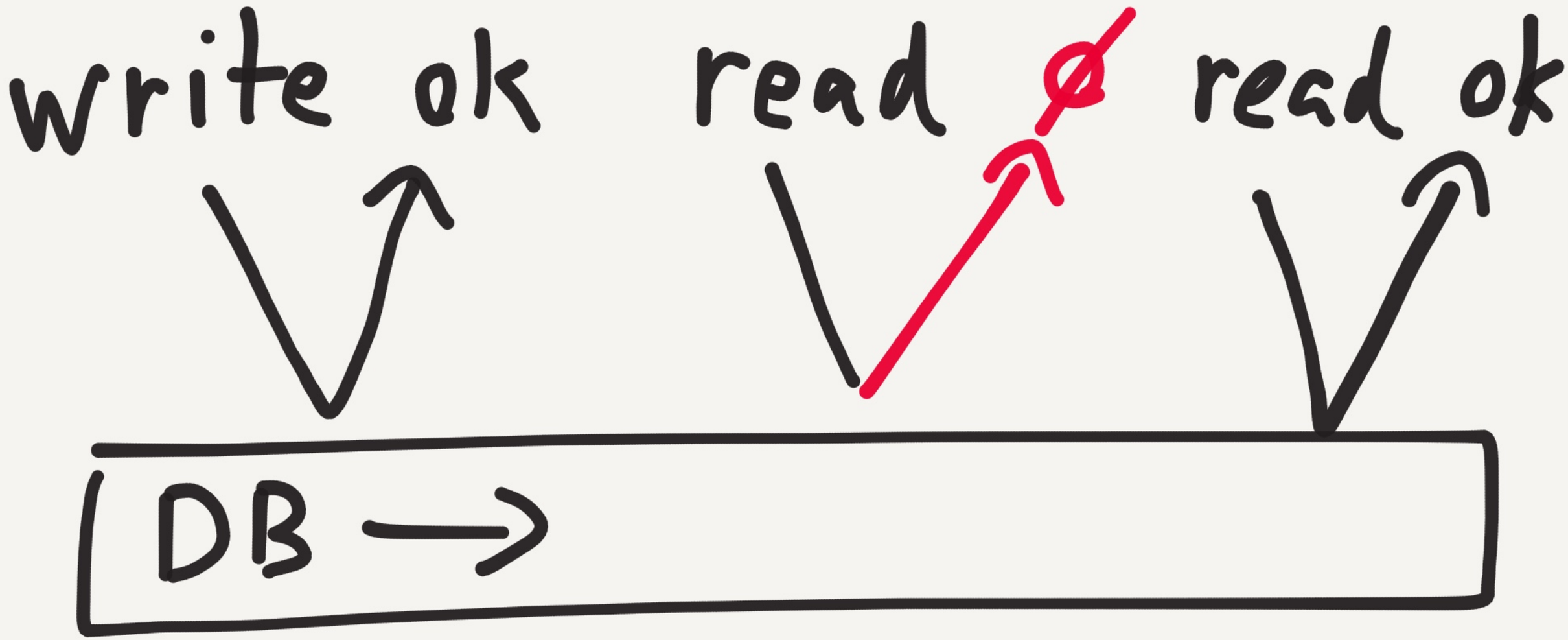
THE HORROR



Split Brain

Broken Foreign Keys

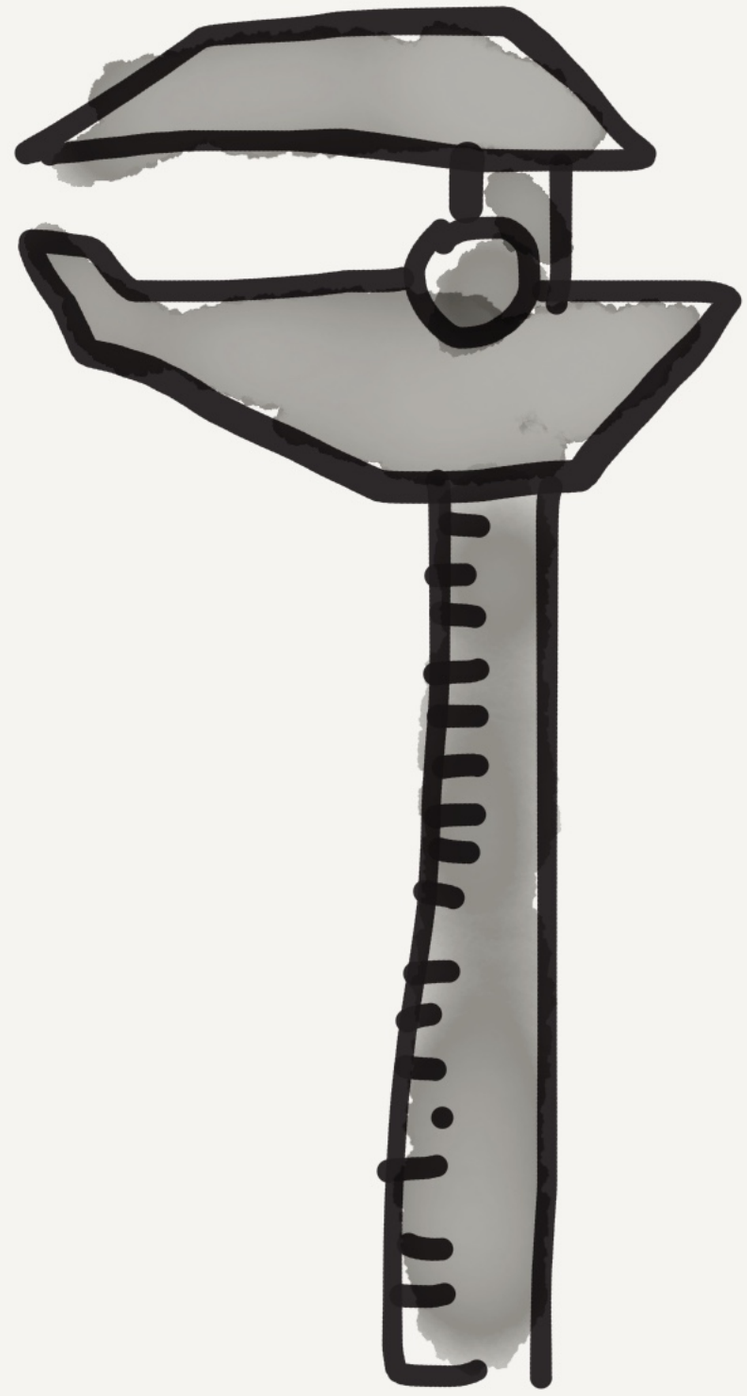




Anomalies

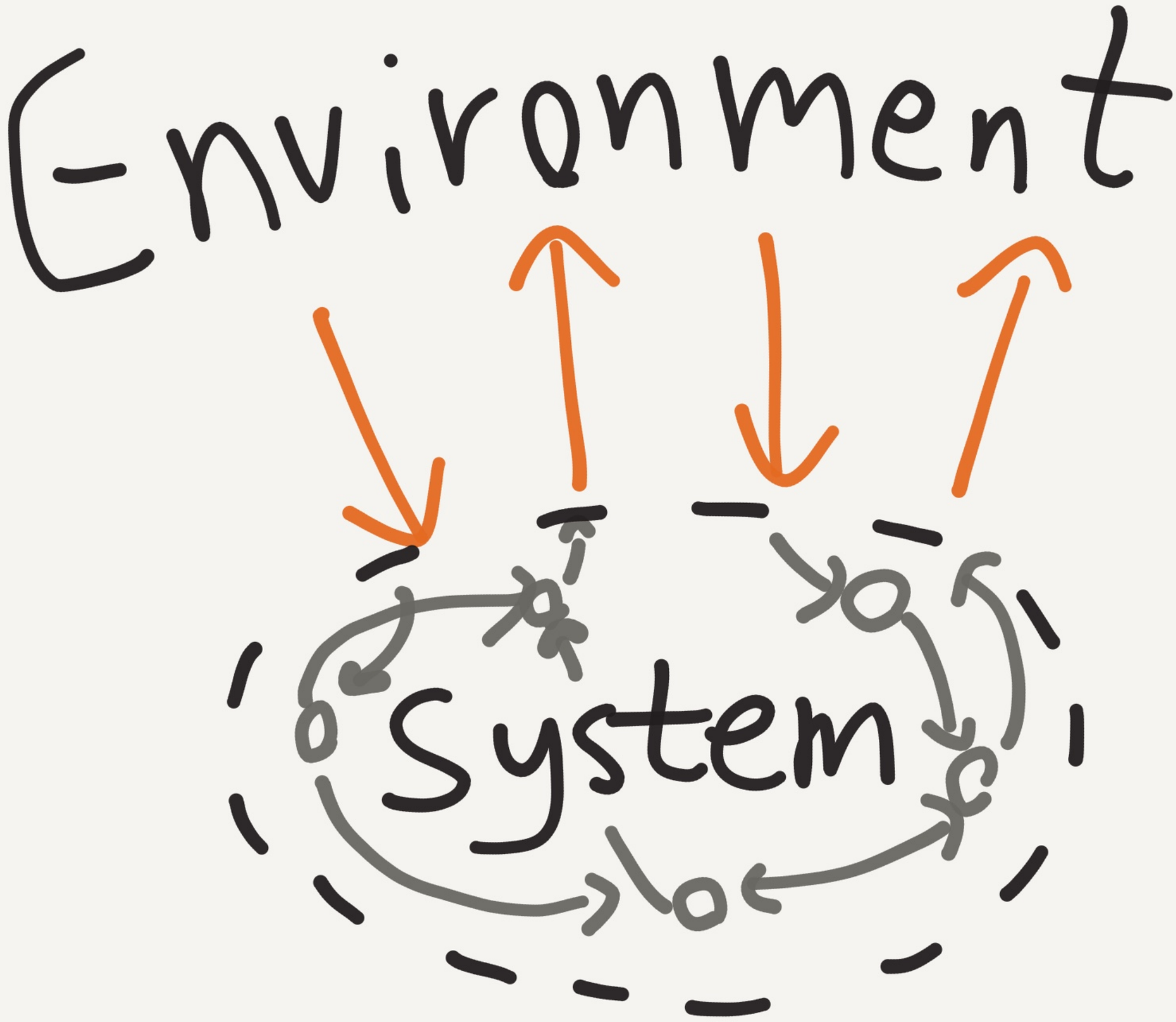
How do you
know if a
system is safe?

Measure your Systems

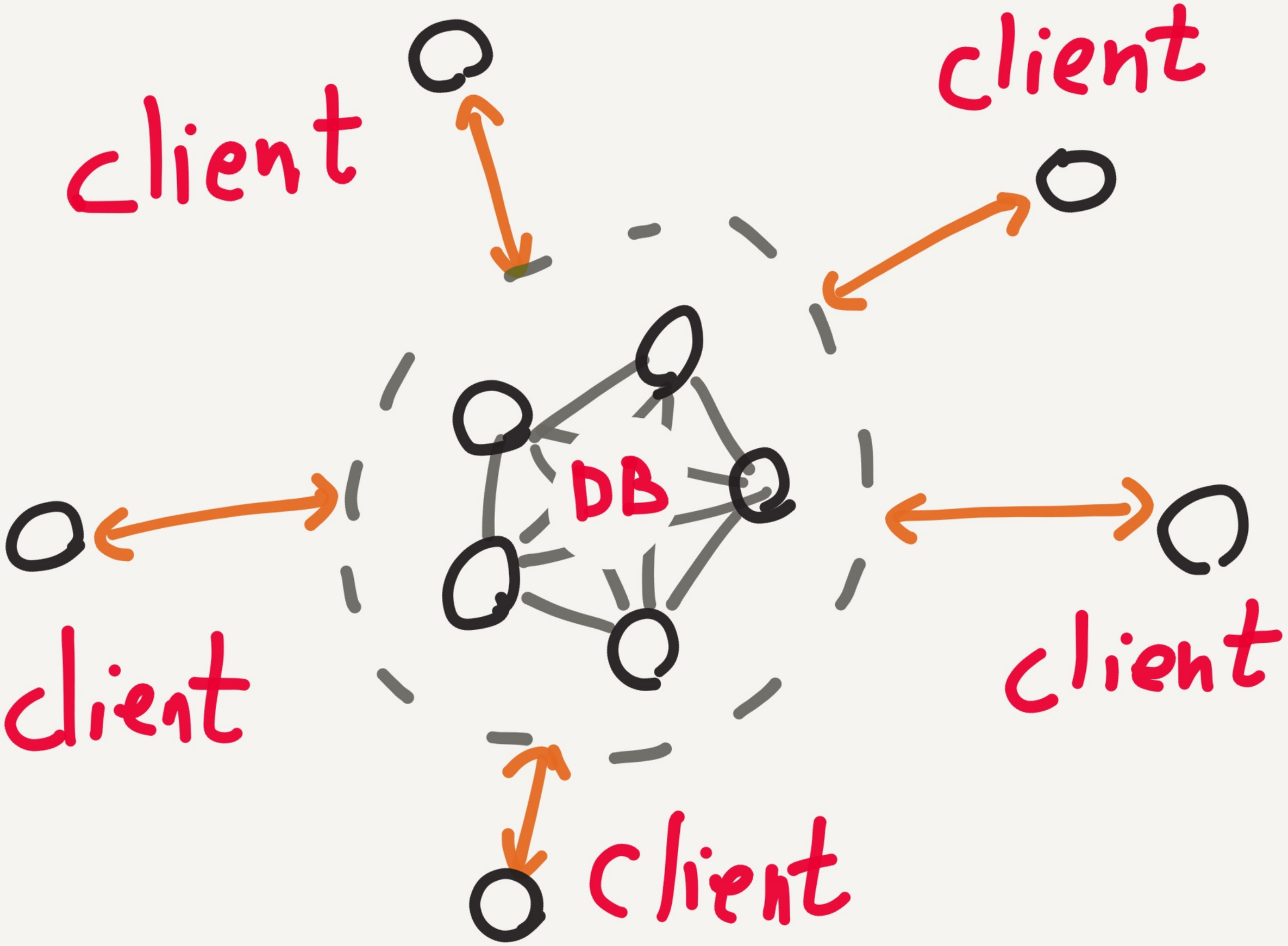


Jepsen

github.com/aphyr/jepsen


Environment
System

-INVARIANTS-

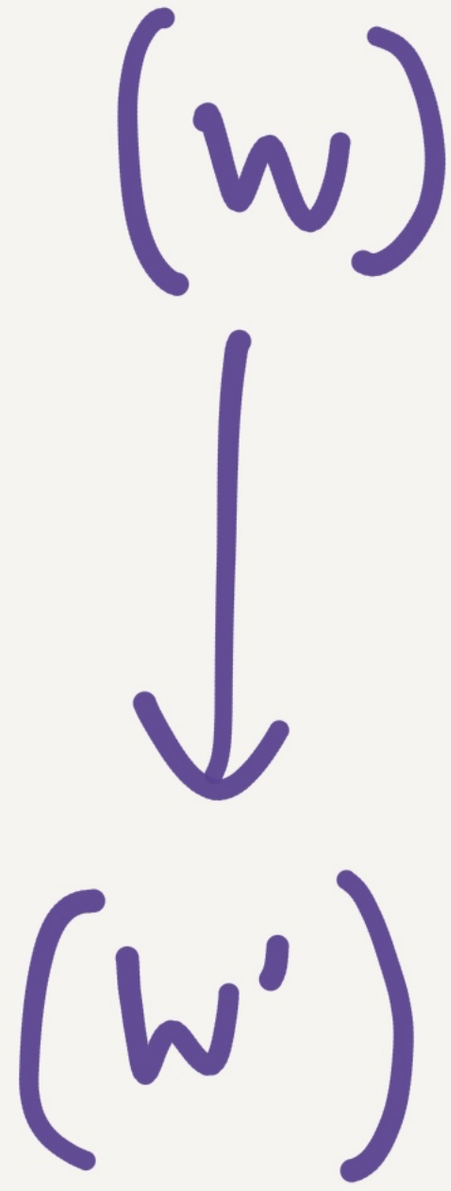

client
client
DB
client
client
client

client: w—w' r—r' w—w'

DB:

client : w —— w' w —— - - -

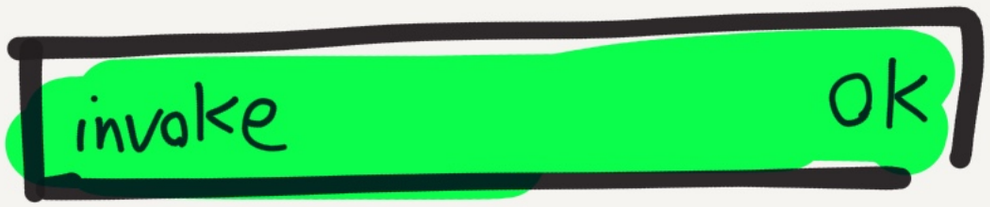
Clients Generate random operations (w) and apply them to the system



invoke ok

invoke fail

involce ? ? info ? ? ?

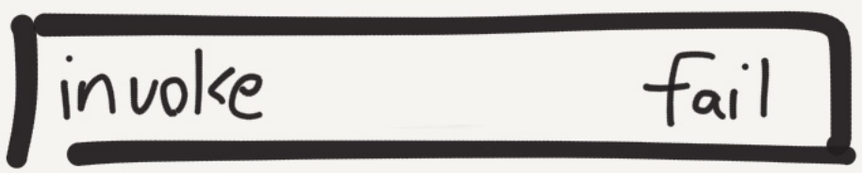

invoke
ok


invoke
fail


invoke
ok


invoke
info


invoke
ok


invoke
fail


invoke
ok


invoke
info

invoke ok

invoke fail

invoke ok

invoke info

invoke ok

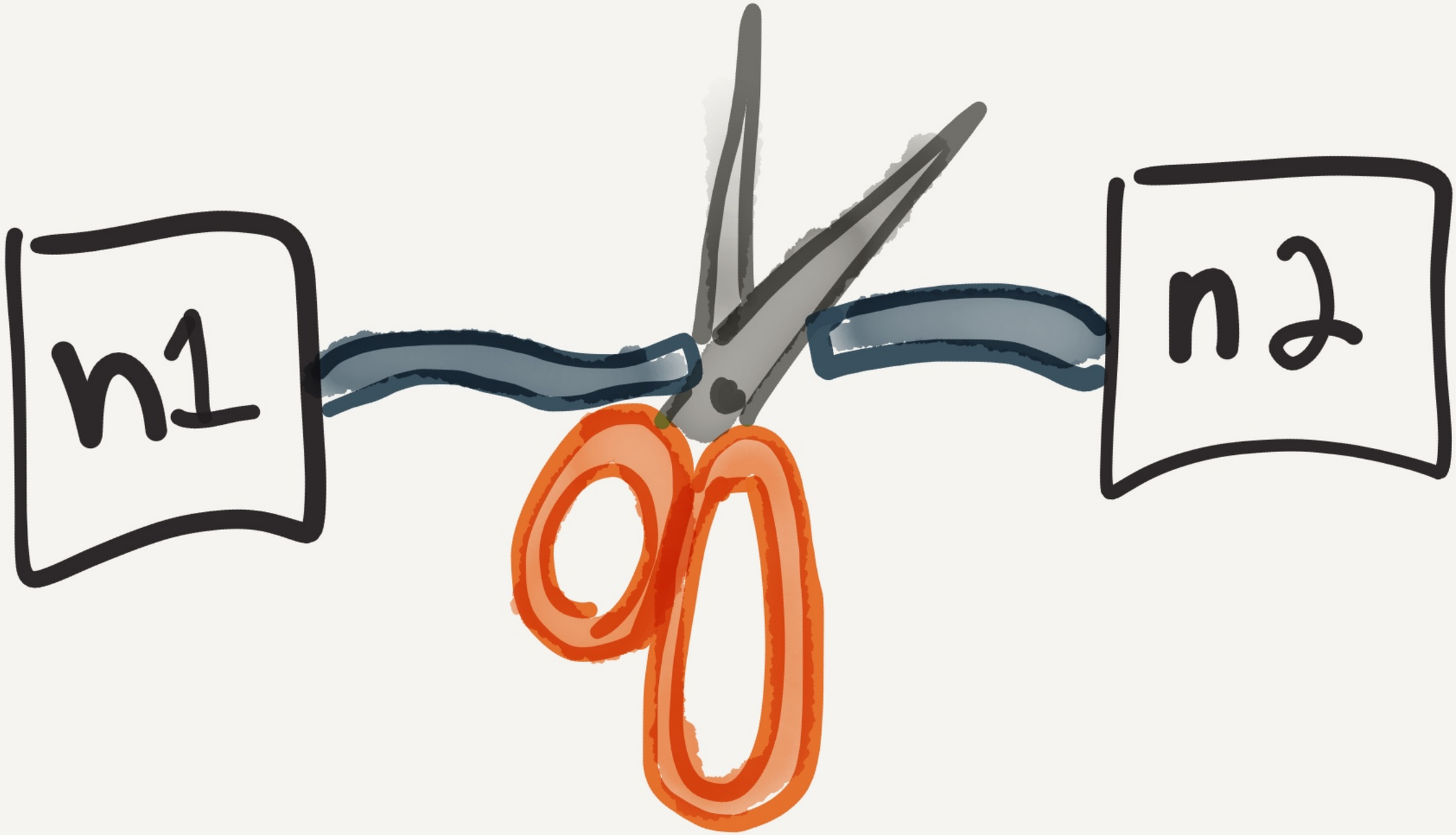
invoke fail

invoke ok

invoke ok

invoke info

1. Generate ops
2. Record history
3. Verify history is consistent w/ model



Partitions!

“So, what have you found?”

Riak

5/13

LWW → lost write

CRDTs → safe

5/13

Mongo

Data loss at all

Write Concerns

Redis Sentinel

Split brain,

massive write loss

Cassandra

9/13

- LWW write loss
- Row isolation broken
- Transaction deadlock, data loss

NuoDB

9/13

Beat CAP by buffer-ing all requests in RAM during partition

Kafka

In-sync Replica Set could shrink to 0 nodes, causing msg loss.

Zookeeper

Works.

etcd / Consul

6/14

stale reads

Elasticsearch

Loses documents in every class of partition tested.

RabbitMQ

Split brain, massive

message loss

Aerospike

Claims "ACID", was really LWW.

Elasticsearch 1.5.0

5/15

Still loses data

in every test case

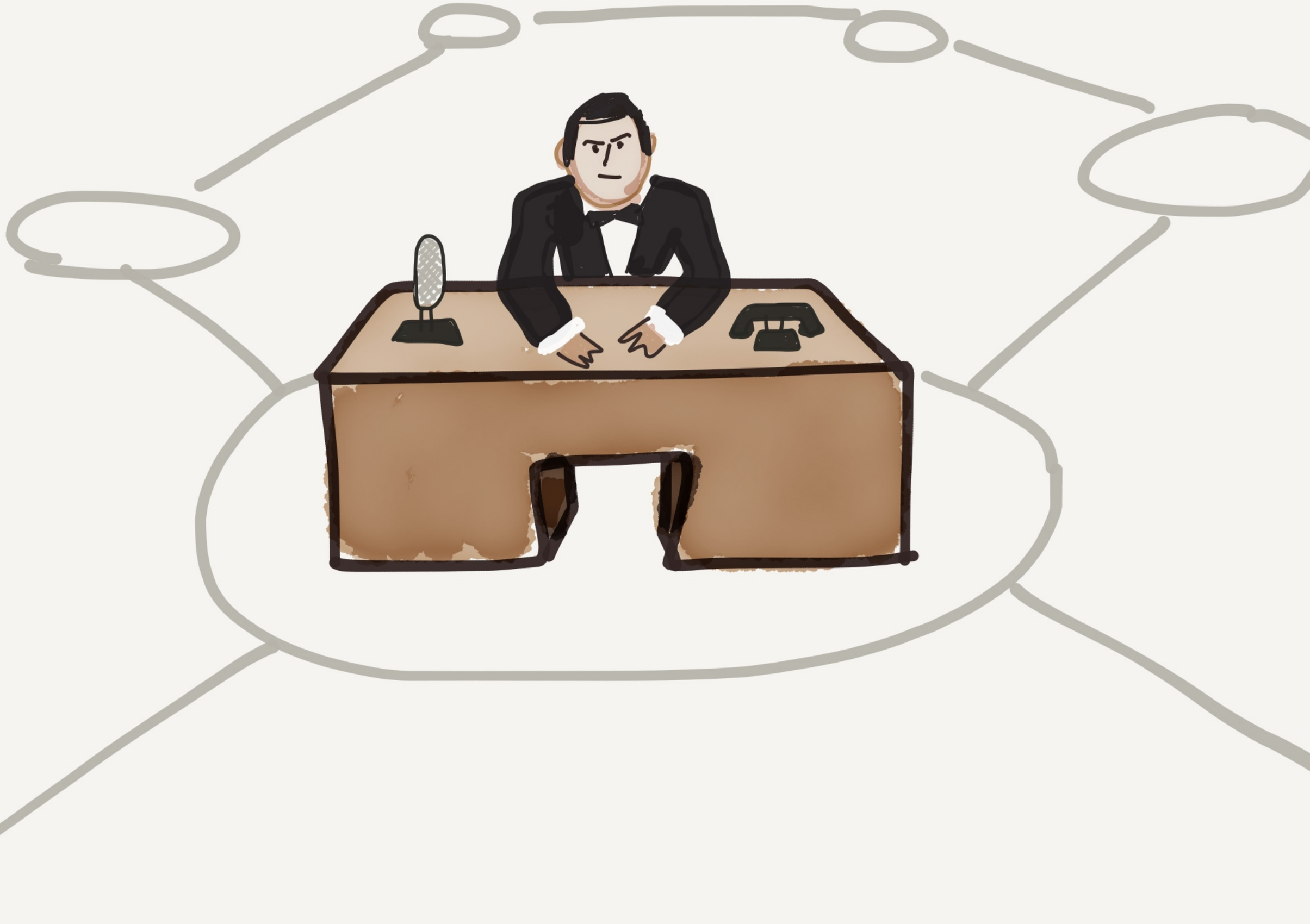
MongoDB 2.6.7

5/15

stale reads

dirty reads

As an industry, we are woefully underprepared to handle distributed systems failure modes



Chronos

Like Cron, but

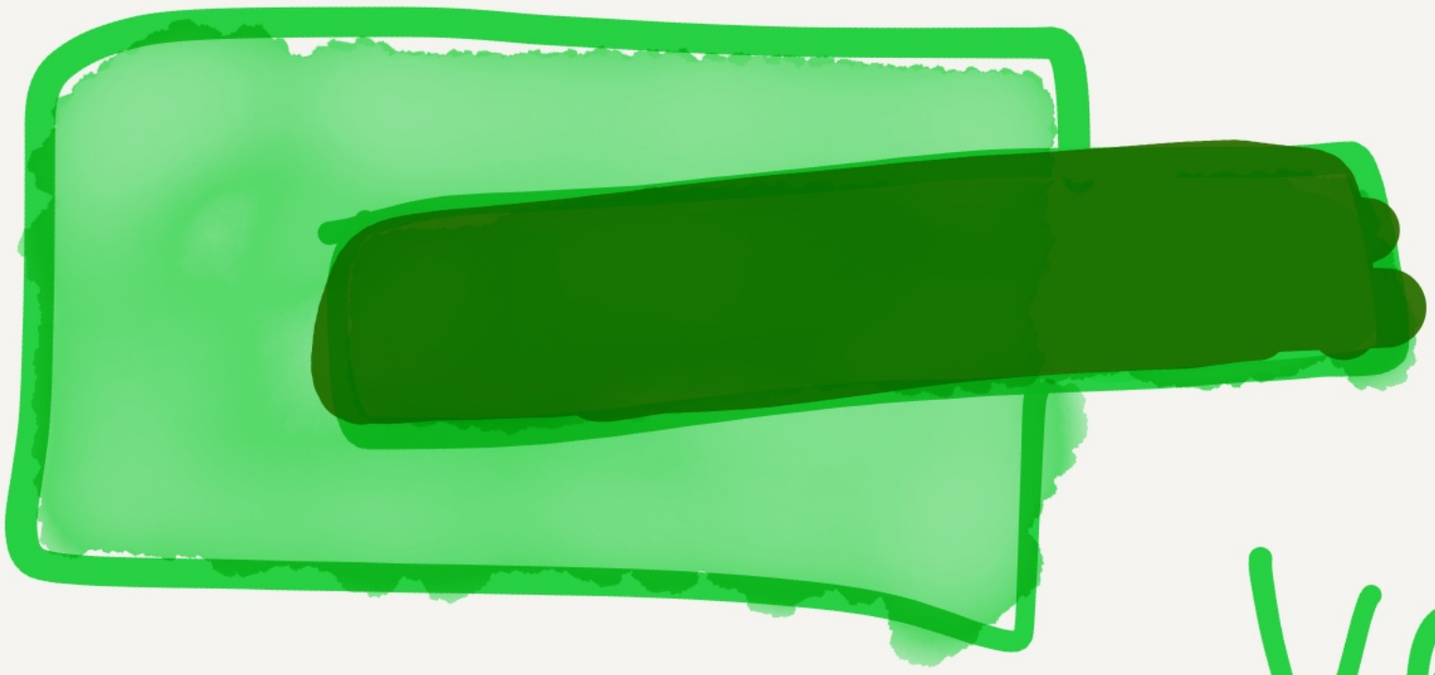
DISTRIBUTED

Chronos

Zookeeper

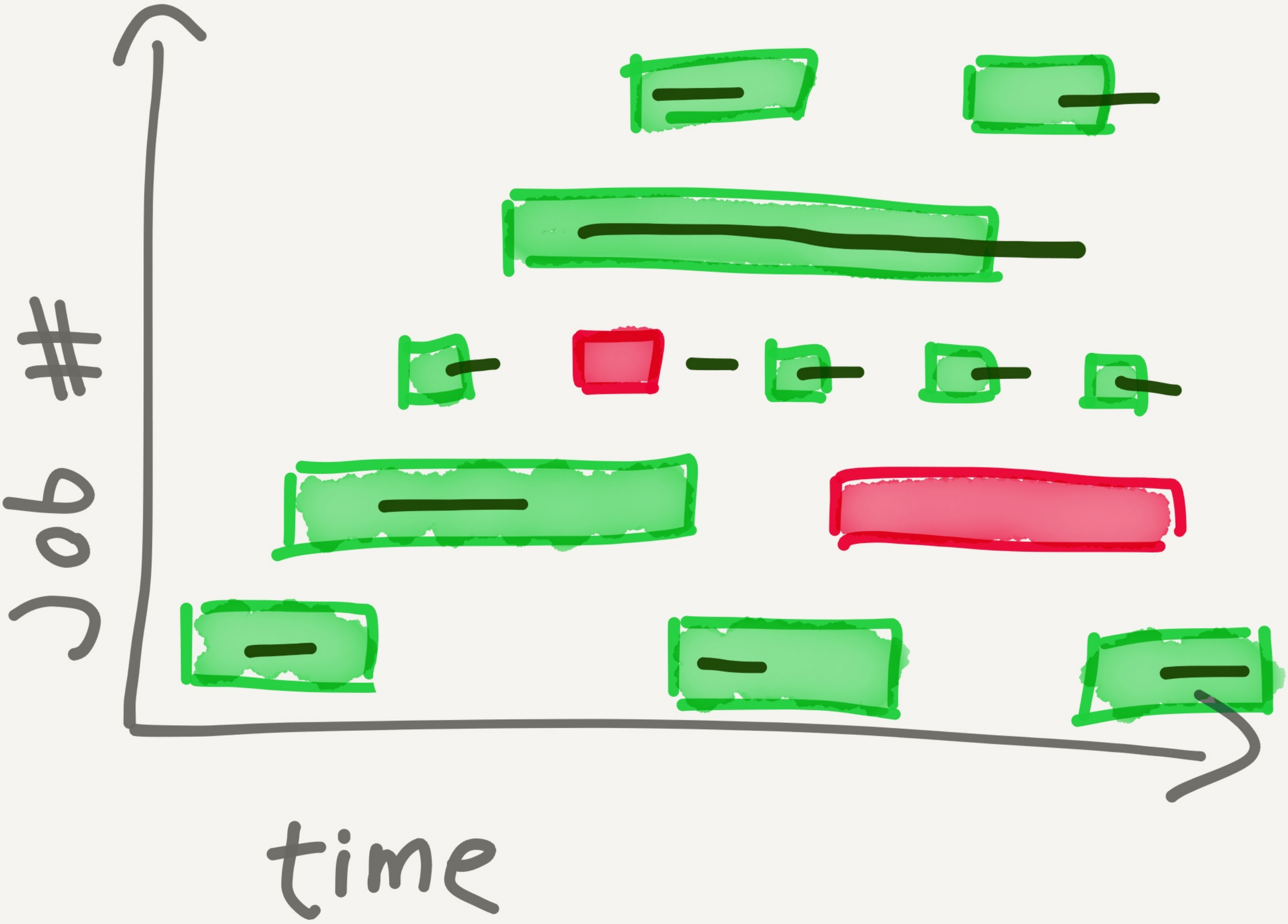
Mesos Masters

Mesos Slaves



valid

invalid

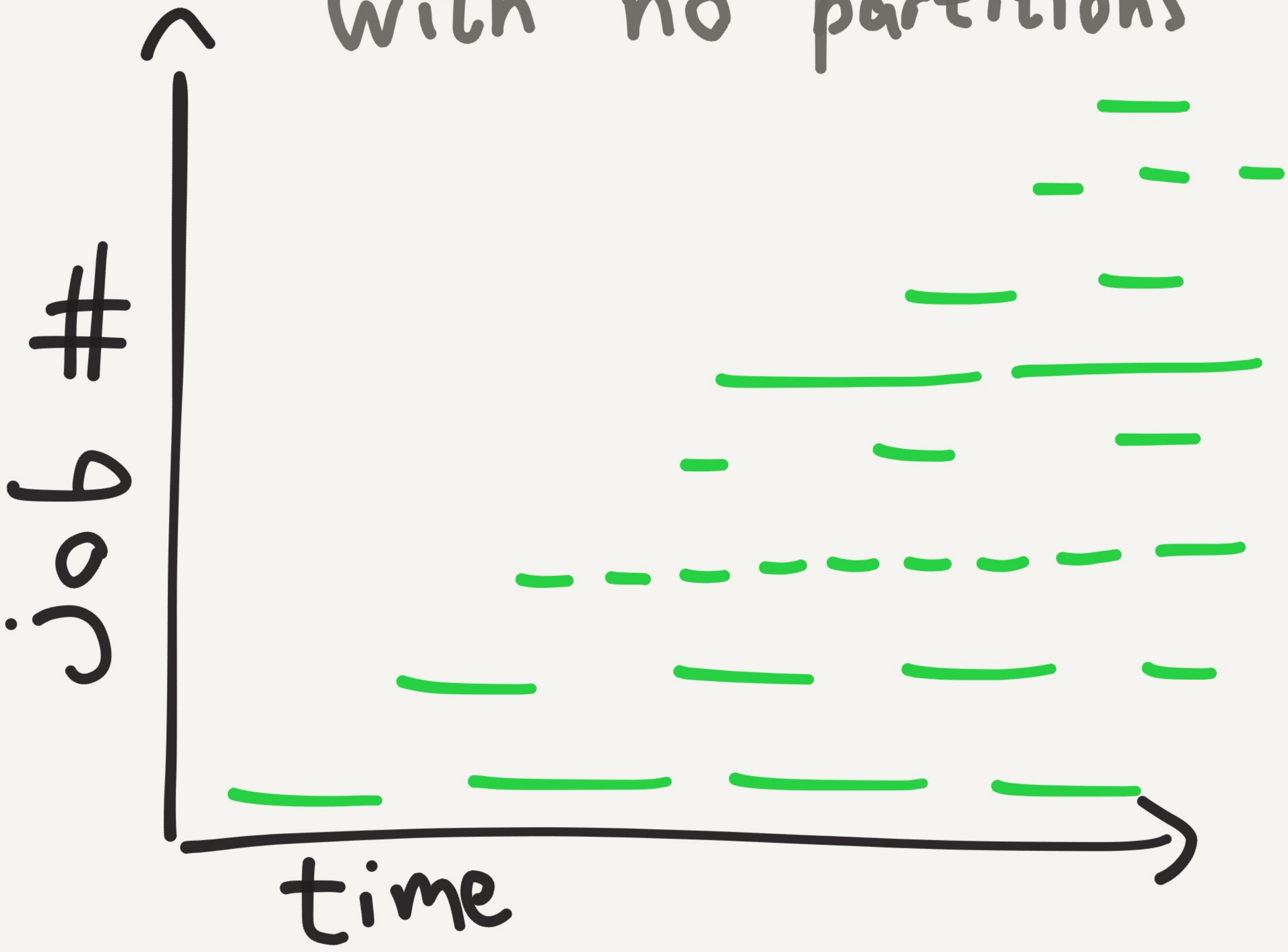

#
Job
time

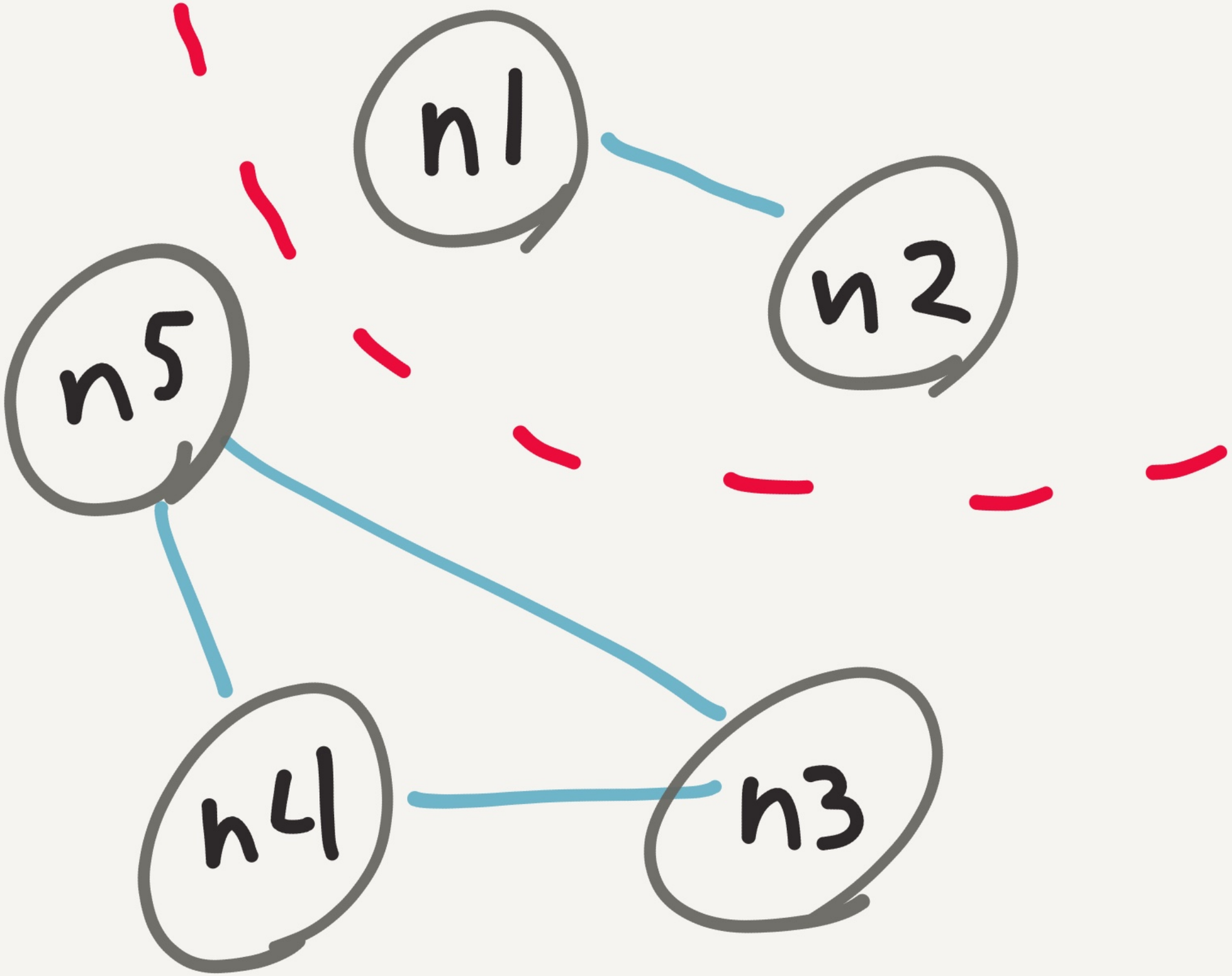
Chronos & Mesos
processes will crash
on losing ZK conn.

(By design)

(Except when they don't)

with no partitions




n1
n2
n5
n4
n3

Partition

Job

time

1: Partition occurs

2: Chronos loses ZK

3: Chronos pauses

4: Partition ends

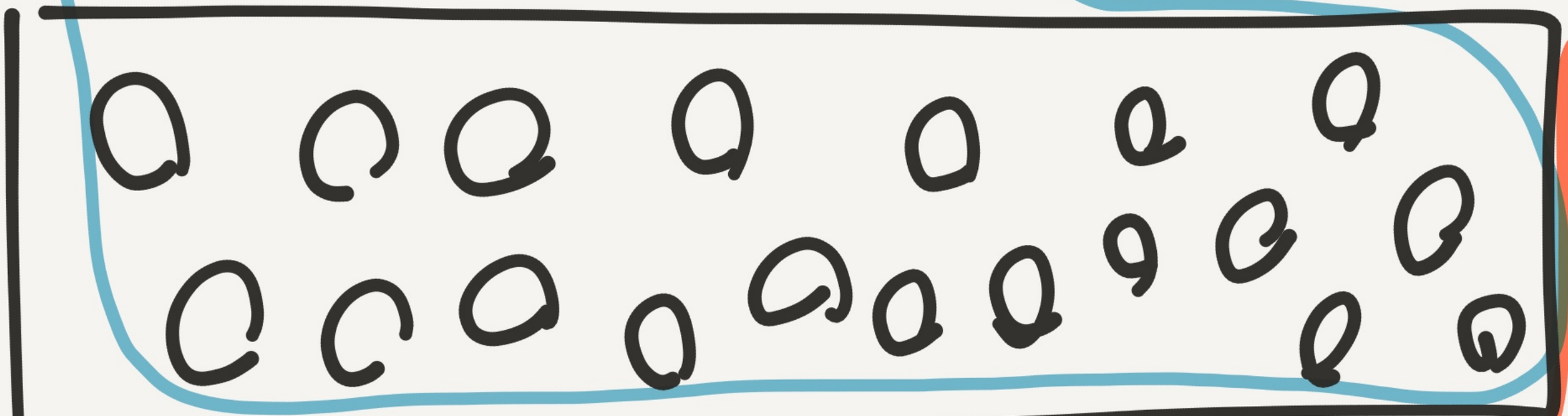
5: Re-election

Why can't the new Chronos leader run any tasks?

Bug #520

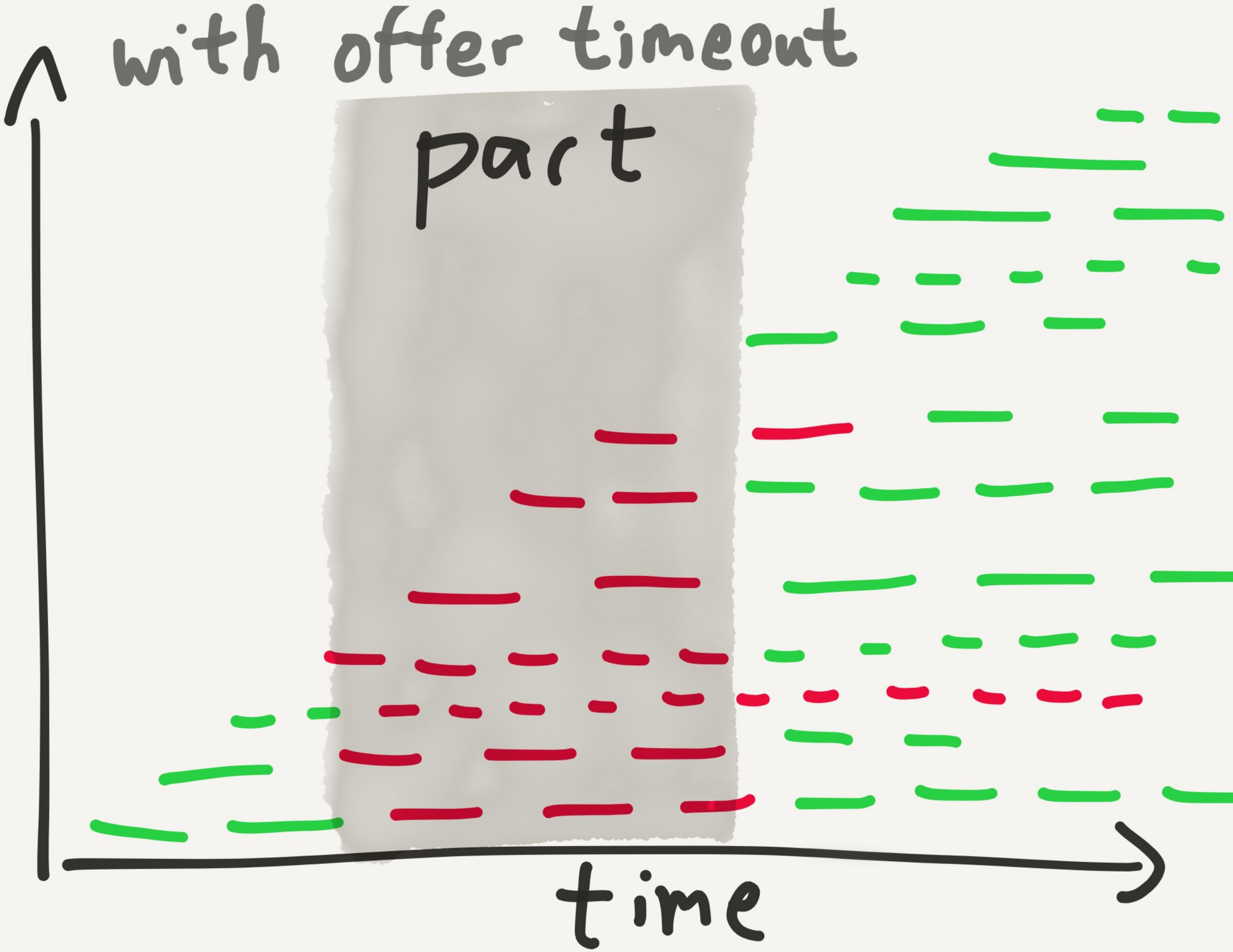
Chronos registers with a new framework ID

Old Chronos



New Chronos

--offer_timeout=30s


with offer timeout
part
time

with offer_timeout

part

time

Mesos-3280

Race condition in node recovery

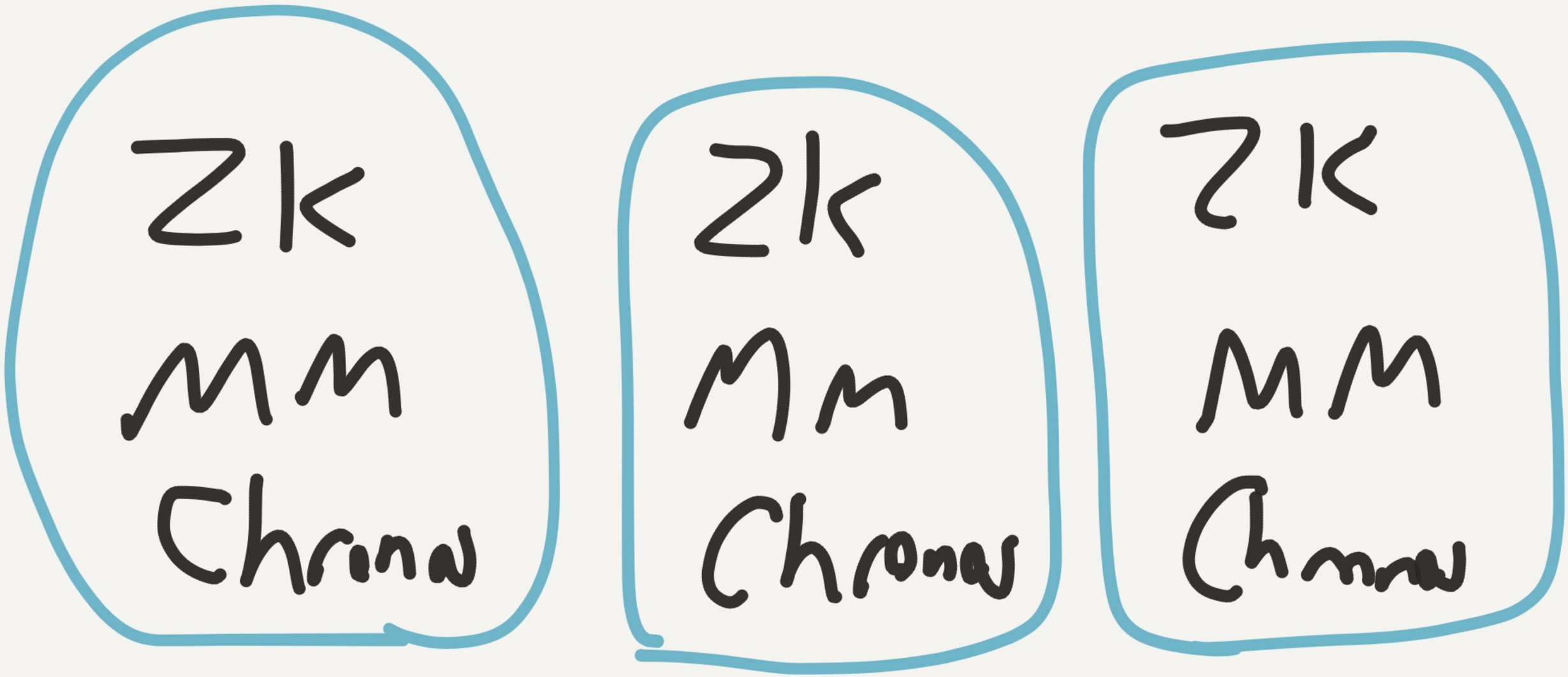
Recovering nodes silently ignore promise $b write ops — Paxos rounds can stall forever

Now fixed in Mesos, pending in Chronos

Chronos

Recommendations

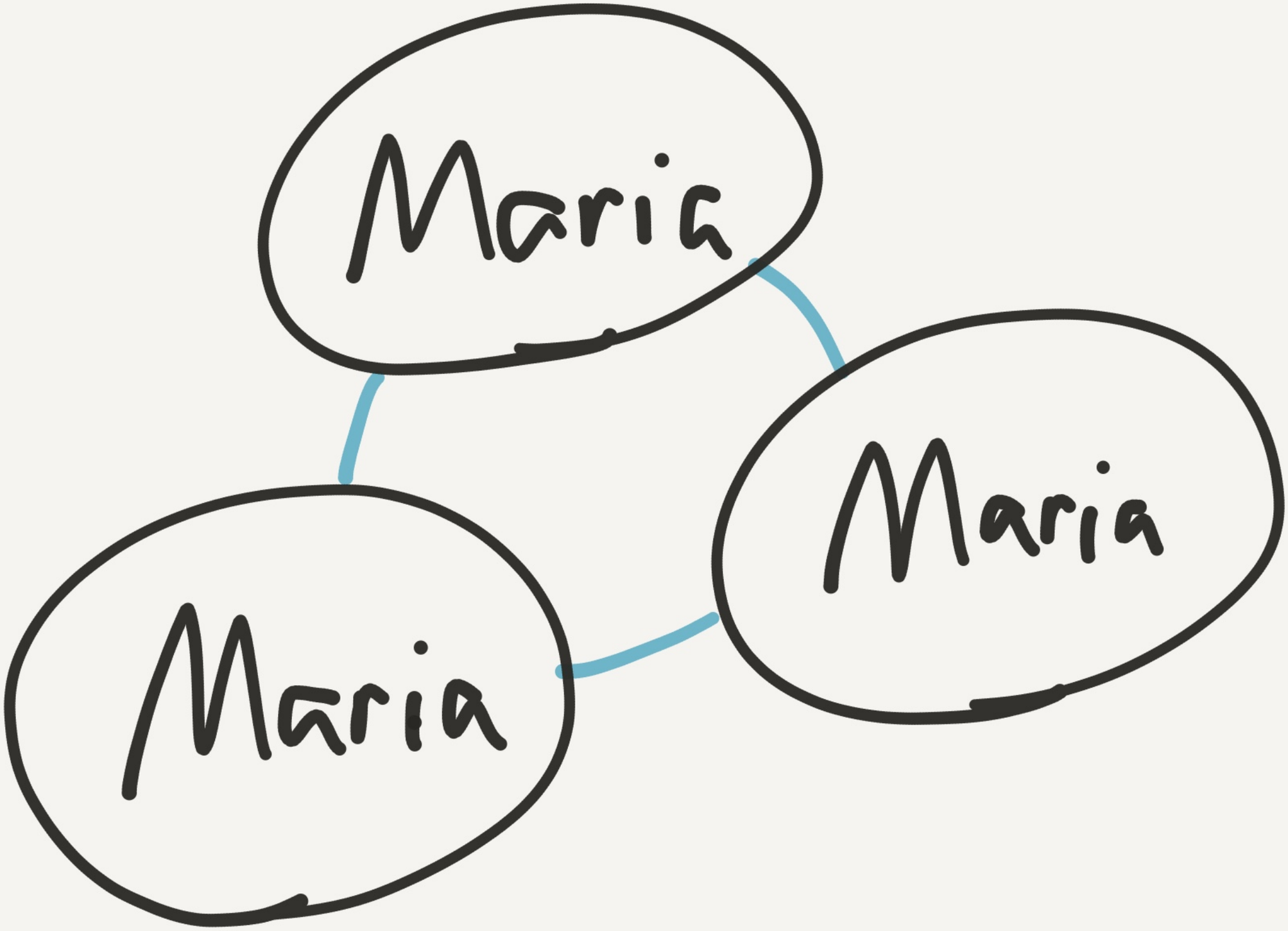
Colocate quora!



- Wrappers for Chronos + Mesos daemons
- schedule_horizon
- offer_timeout
- Monitor your jobs
- Monitor daemon restarts

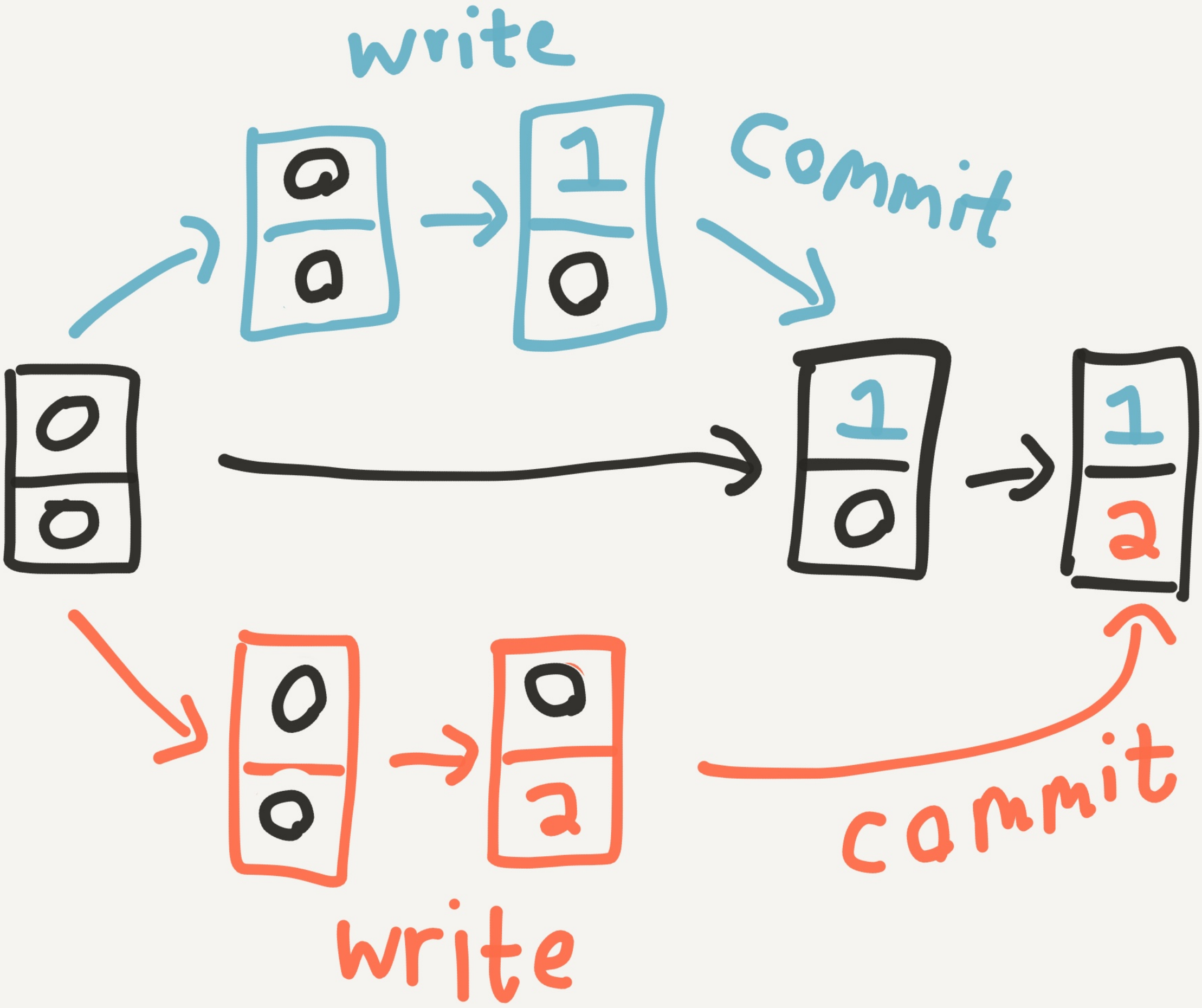
GALERA CLUSTER

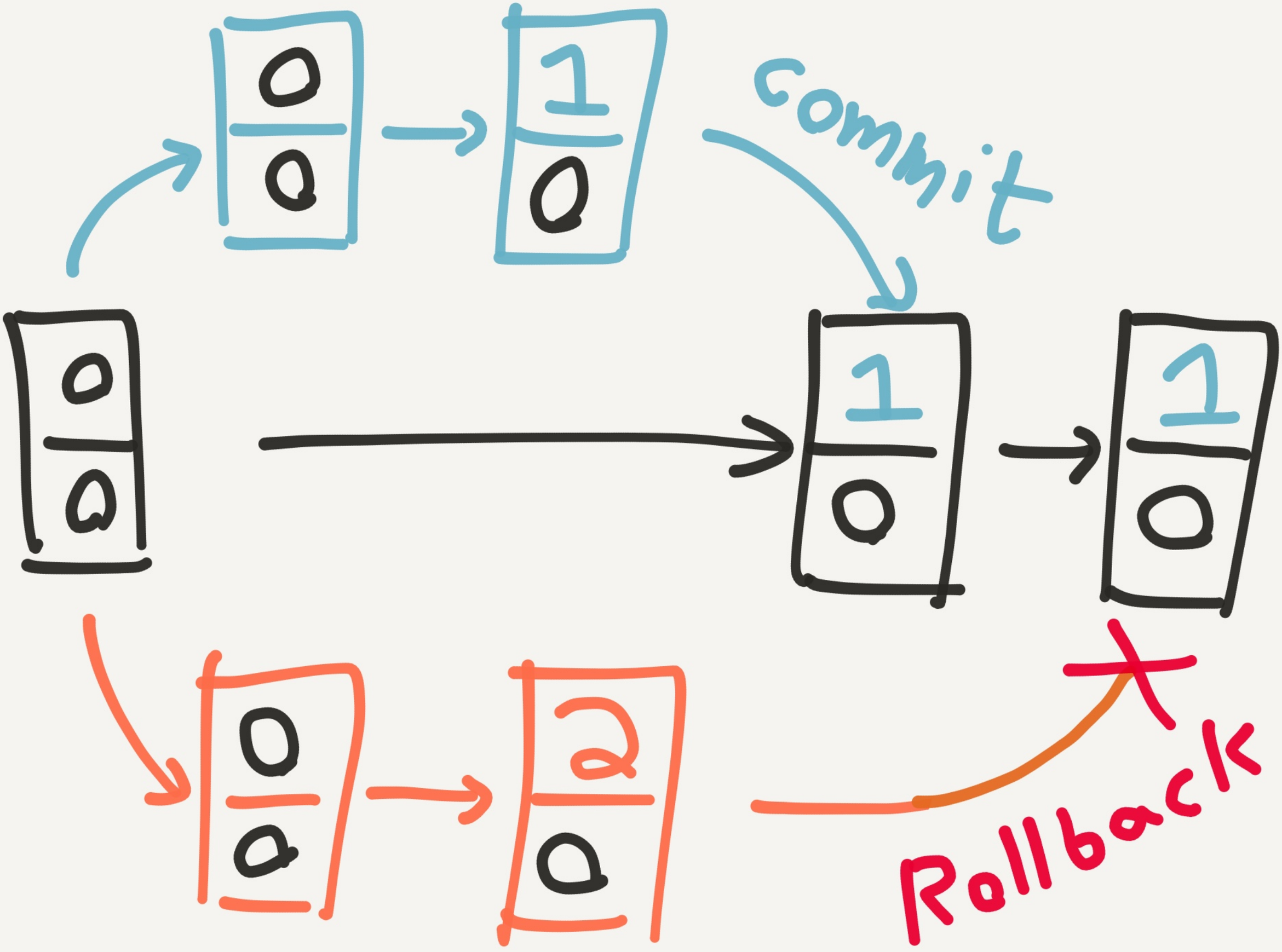
Symmetric replication for MySQL, MariaDB, Percona XtraDB, ...


Maria
Maria
Maria

Between nodes, claims to provide

Snapshot Isolation


write
0
0
0
0
1
0
Commit
1
0
1
0
2
commit
0
0
0
2
write


0
0
1
0
commit
0
0
1
0
1
0
0
0
2
0
Rollback

"SI level occurs between Repeatable Read and Serializable"

(so, use 1SR, get SI)

serializability 1SR

snapshot isolation SI (wrong!)

repeatable read RR

read committed RC

read uncommitted RU

1SR

(A3) RR

SI (A5B P6)

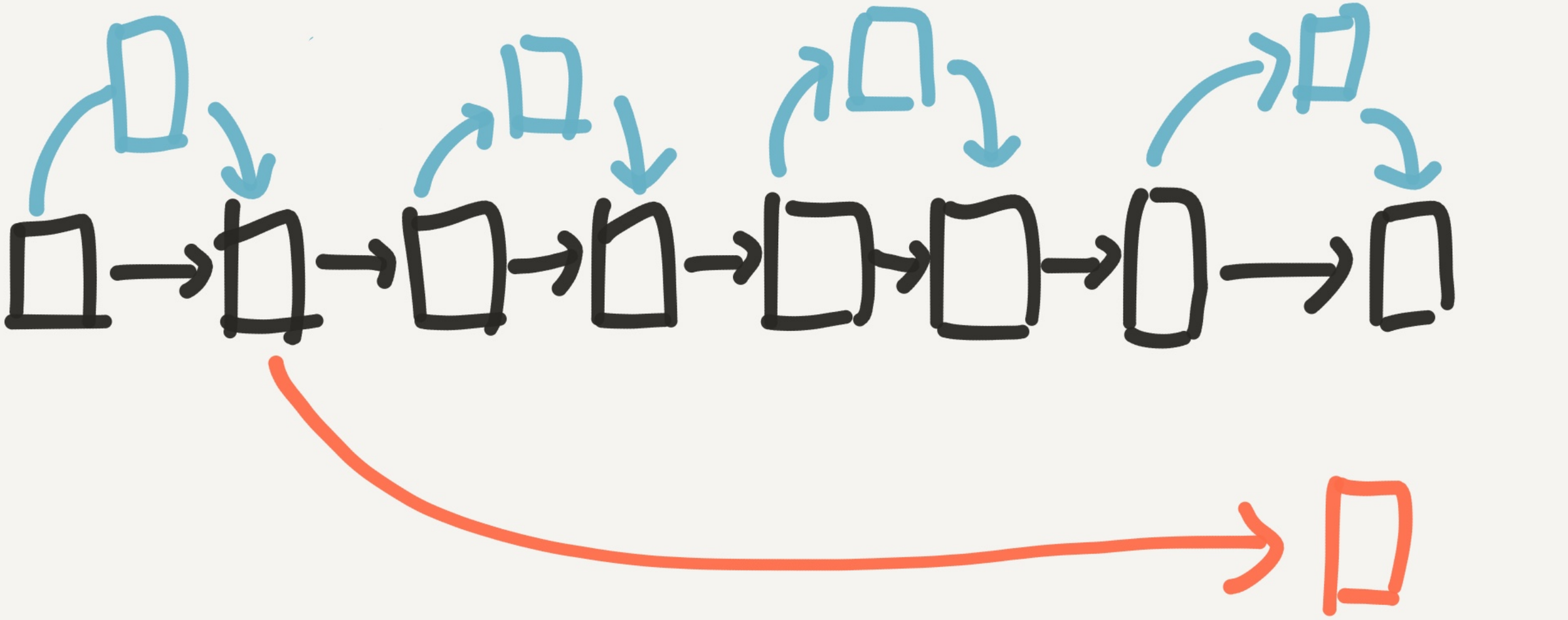
RC

RU

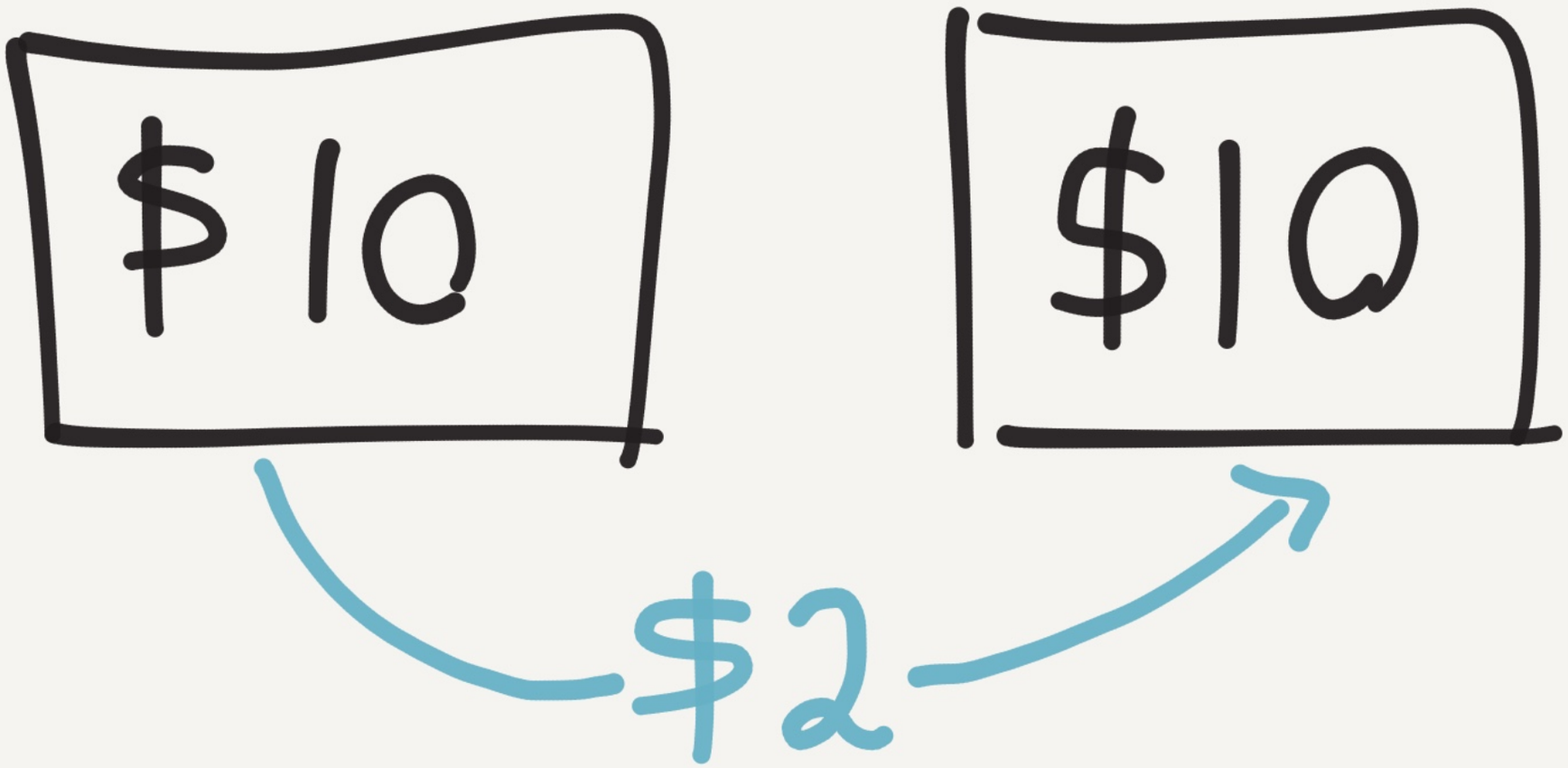
OK...so how

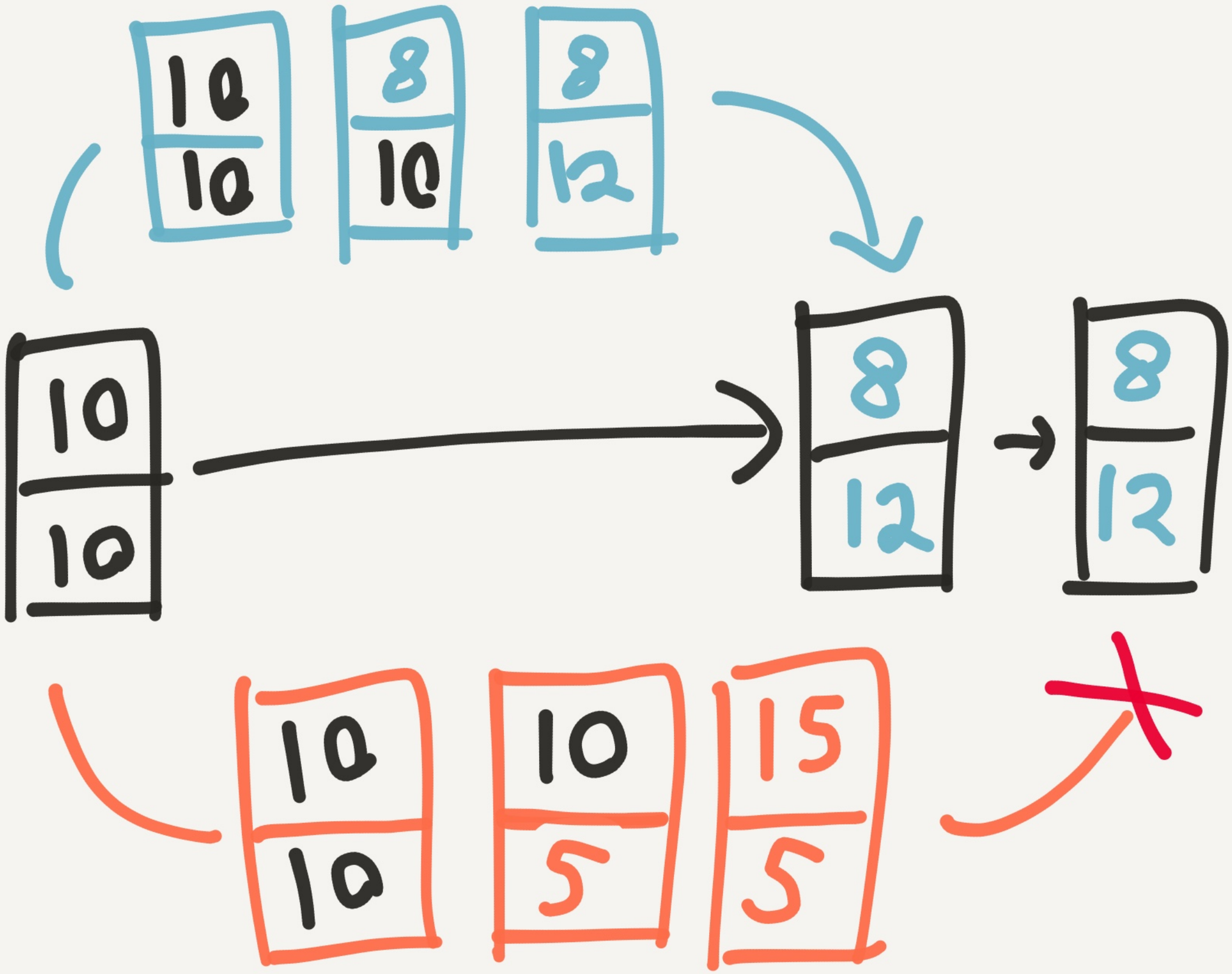
do we test it?

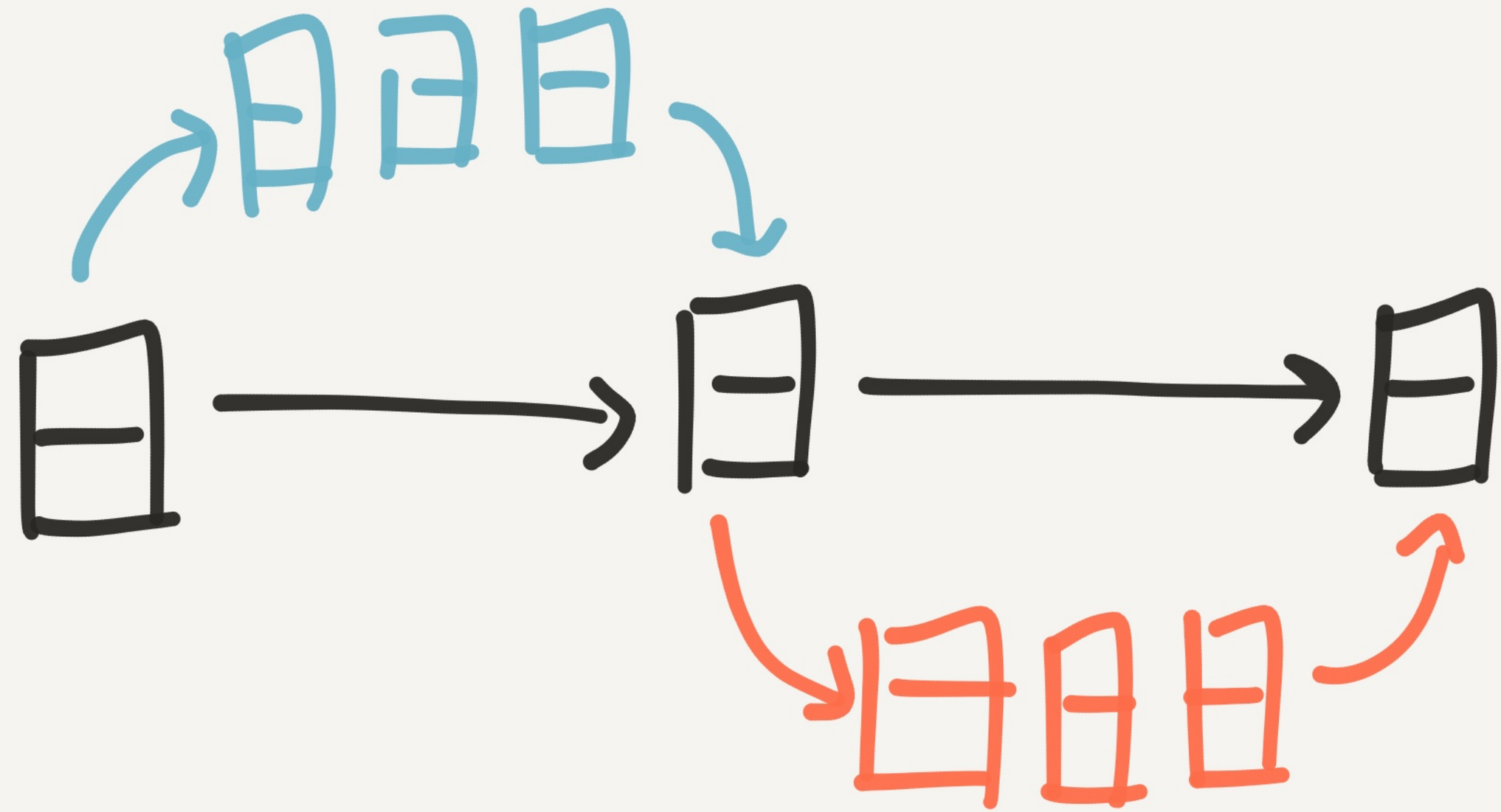
Can't use our linearizability checker...


stale read

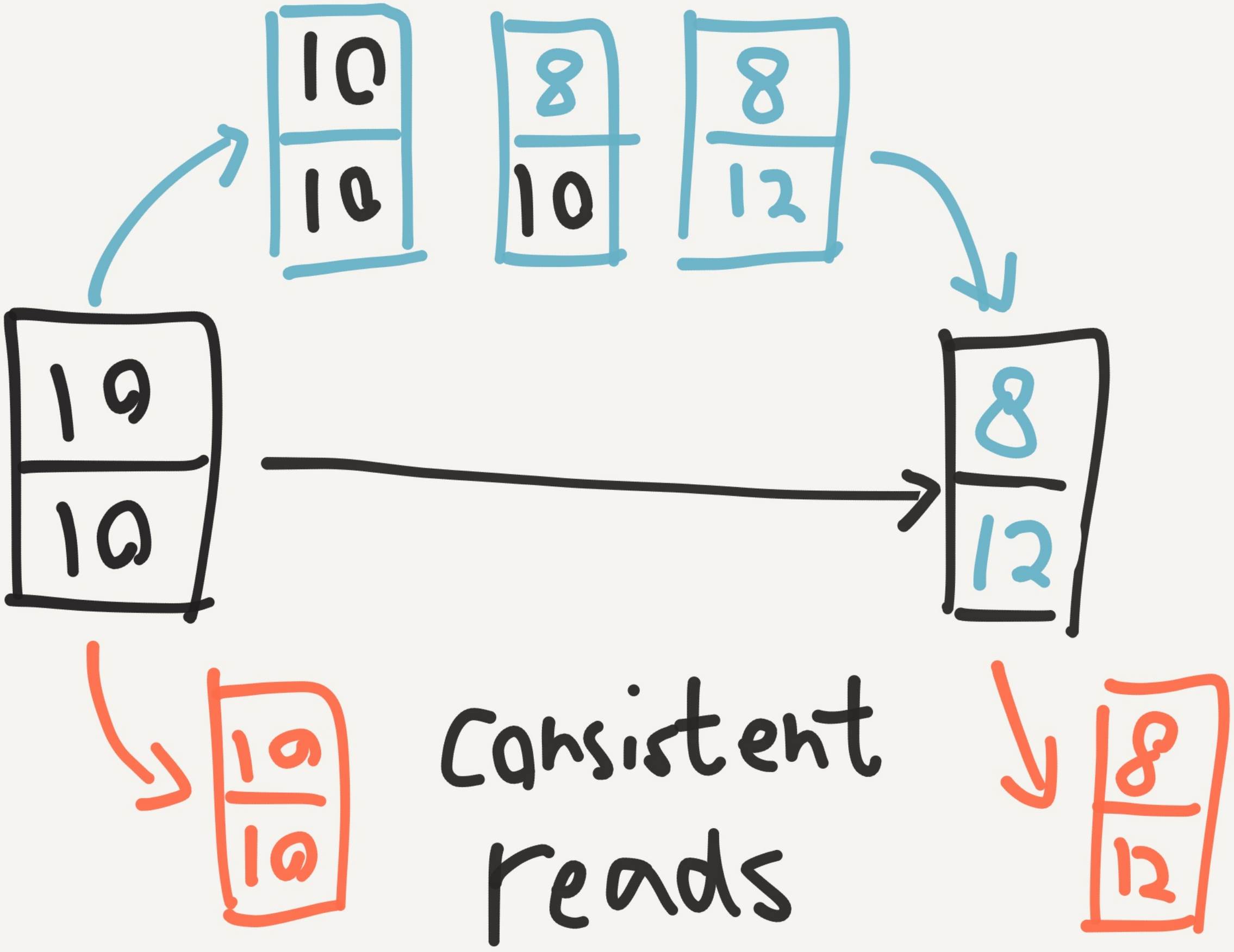
Consider 2 bank accounts...




10
10
8
10
8
12
10
10
8
12
8
12
10
10
10
5
15
5



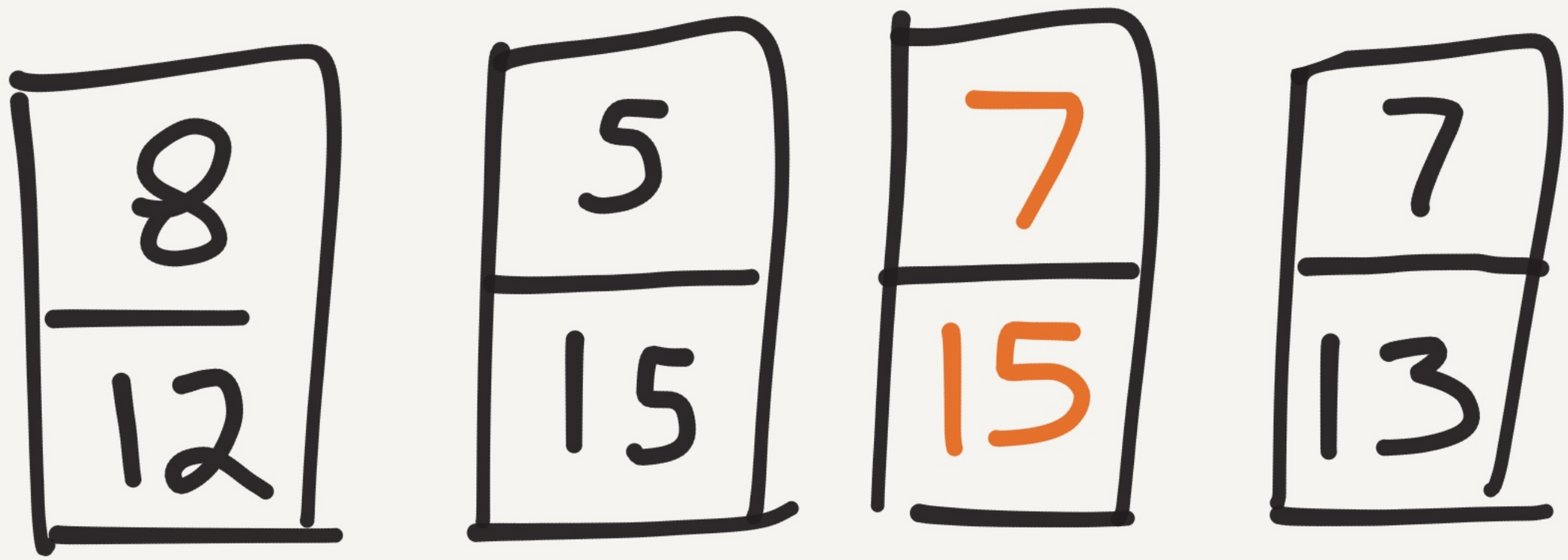
Transfers must serialize

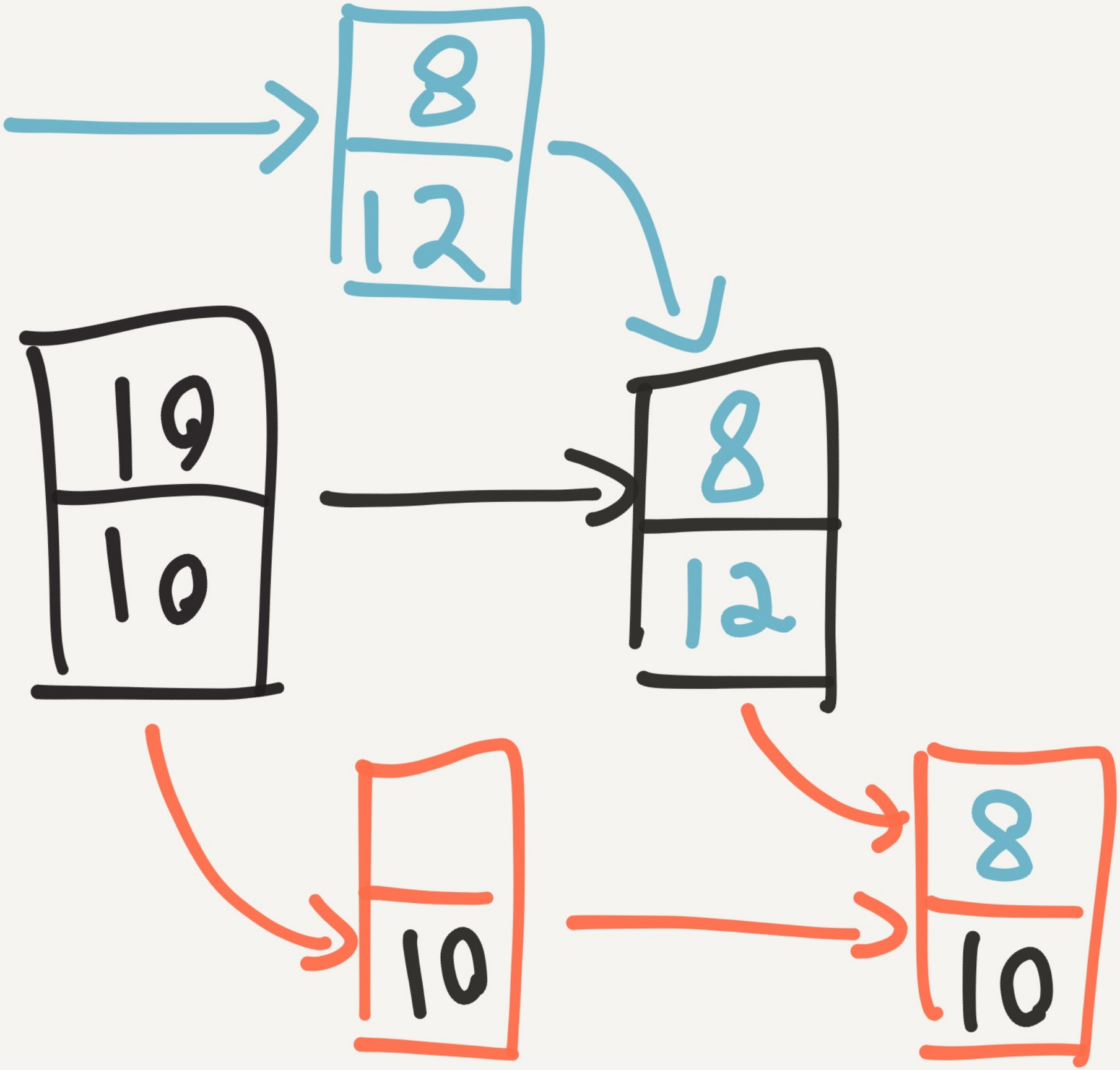

10
10
8
10
8
12
10
10
8
12
10
10
8
12
Consistent
reads

⇒ Sum of accts is always a CONSTANT

ORIS IT?

Inconsistent Reads



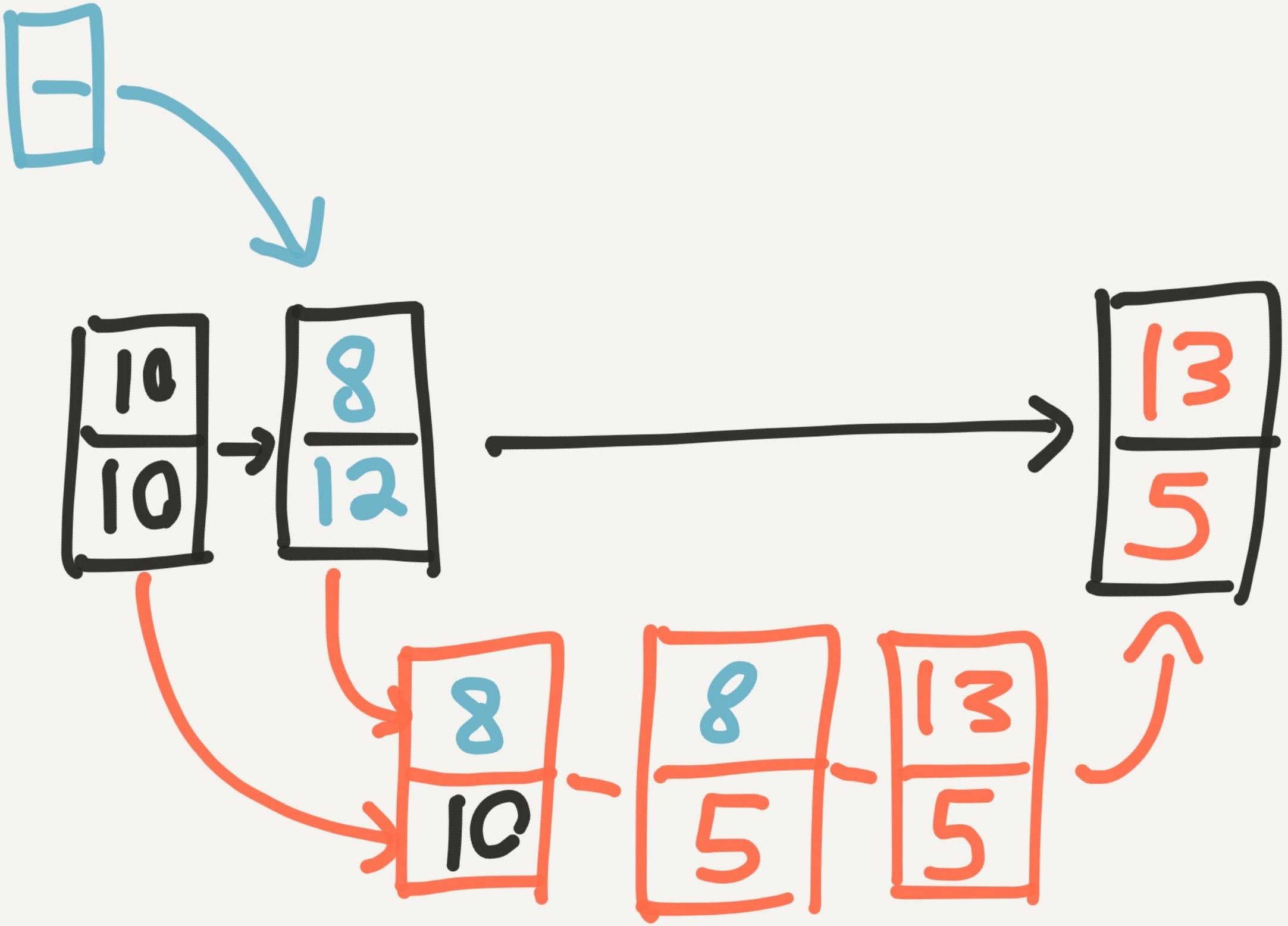

8
12
19
10
8
12
10
8
10

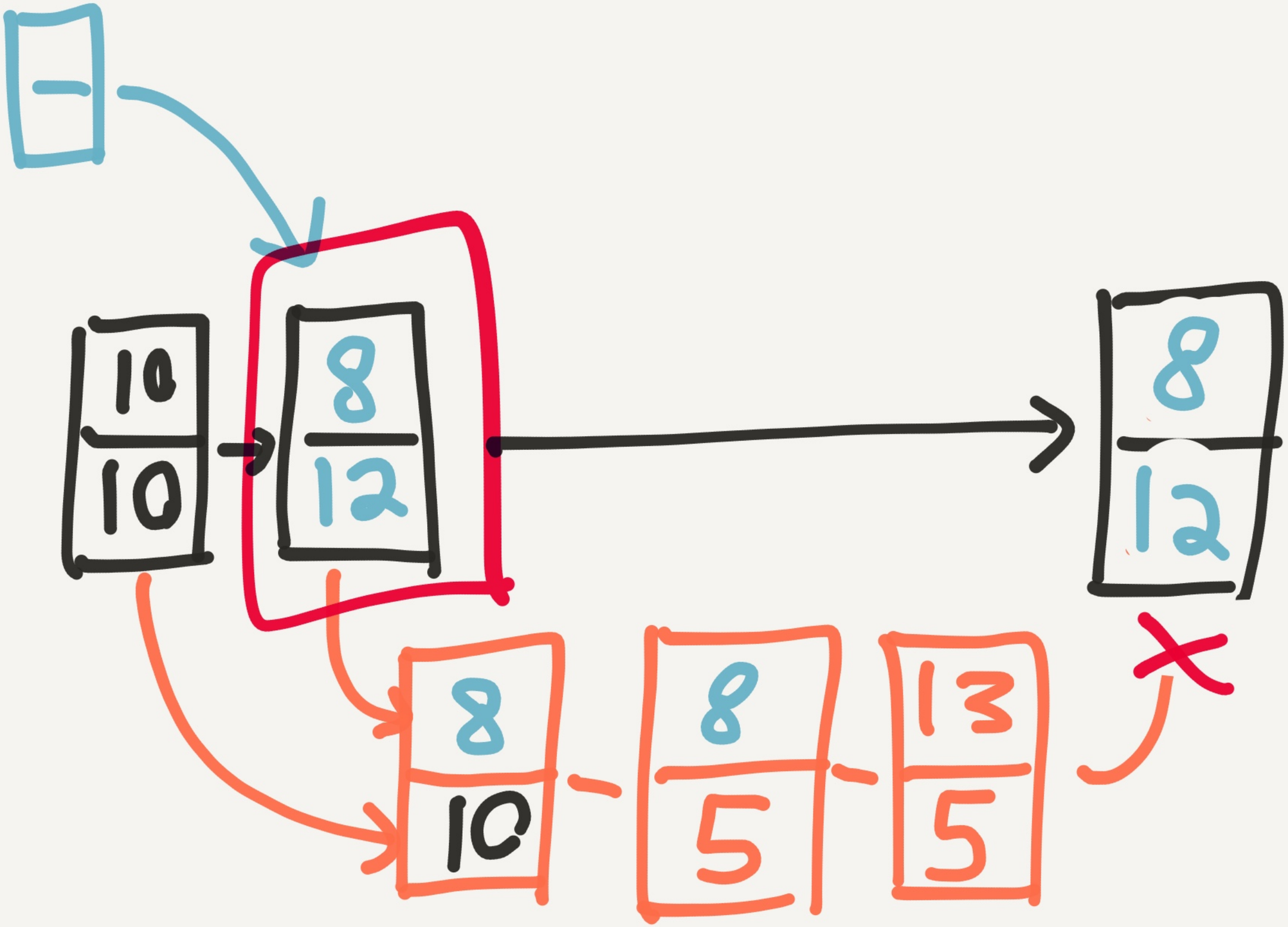
A5A READ SKEW

Read Skew breaks SI invariants

IT GETS

WORSE


1
10
10
8
12
13
5
8
10
8
5
13
5


1
10
10
8
12
8
12
8
10
8
5
13
5
X

Galera engineers:

"first committer wins rule is not honored by galera certification"

Inno DB's Serializable isolation relies on shared read locks which Galera doesn't enforce.

¬ first-committer-wins

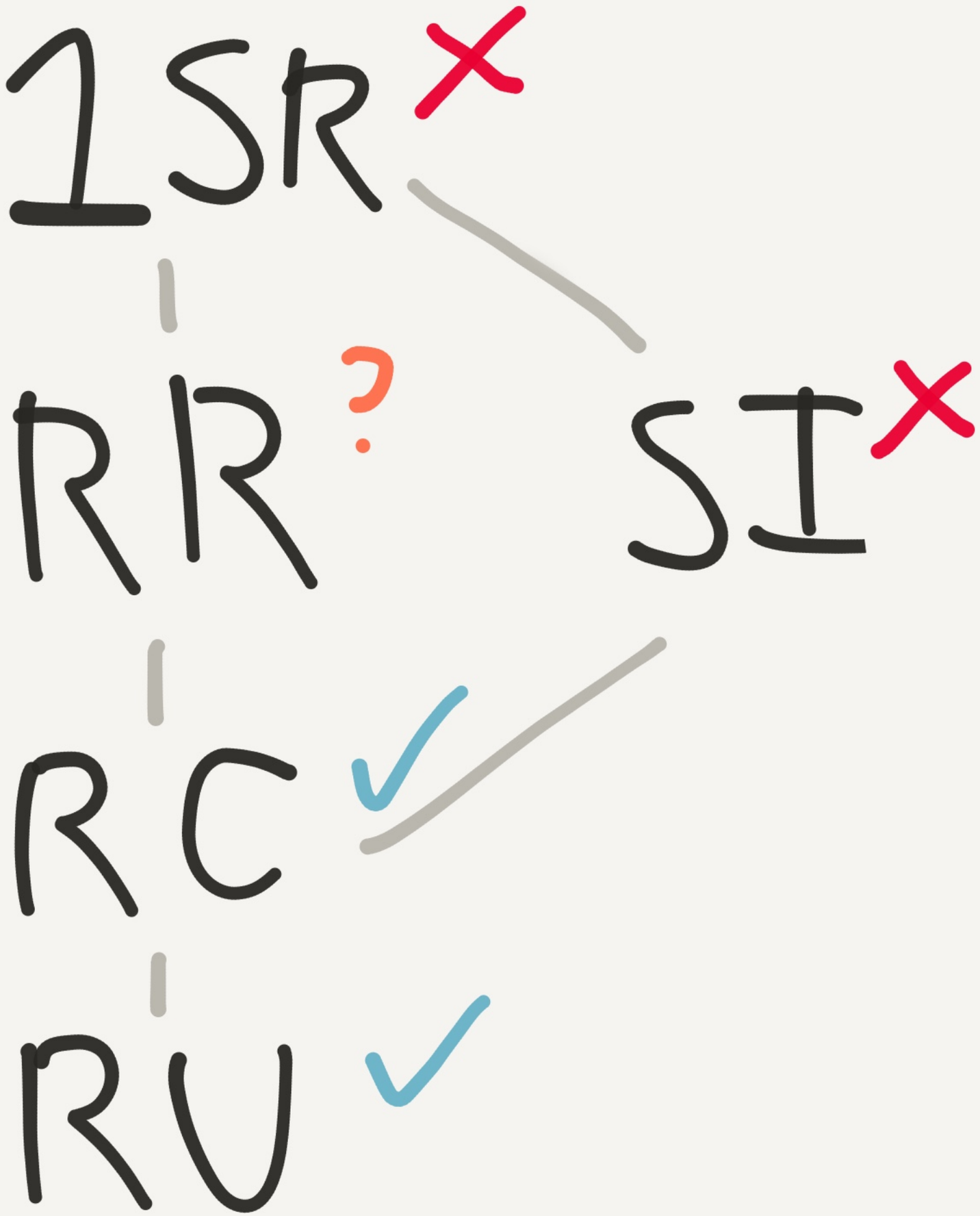
⇒ ¬ SI

Docs are lying

Not all is

LOST

Galera does not allow Dirty Read


1SR
RR
SI
RC
RU

If your app works with RC, it might be ok.

RC is a pretty weak property.

Allows those inconsistent reads!

Still waiting on a fix

RethinkDB

- Document Store
- Like MongoDB
  - Linearizable on 1 Key
  - No multi-doc txns

Now uses

to offer

consensus!

reads

writes	majority	single	outdated
majority	Linear	Dirty	Reads
single	LOST UPDATES	Stale Reads	

During Partitions,

- brief latency spike
- brief downtime
- Recovery in ~10s

Outdated reads always available

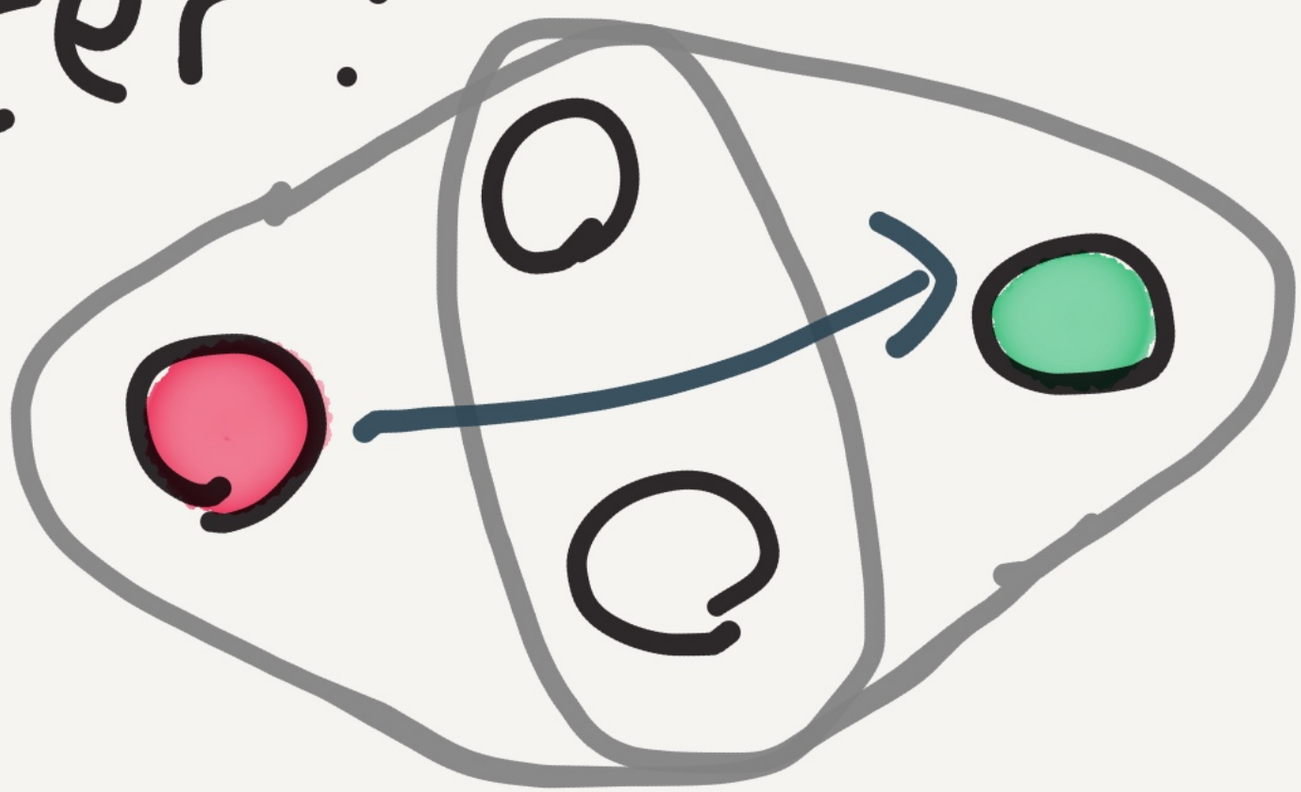
All other ops require a majority connected

Majority/Majority

PASSED

We have to go Deeper

What if... we rebalanced the cluster?



still passes. Ok.

Let's rebalance

during partitions

lost update!

3

write 0

17

cas 3→0

time ⟶

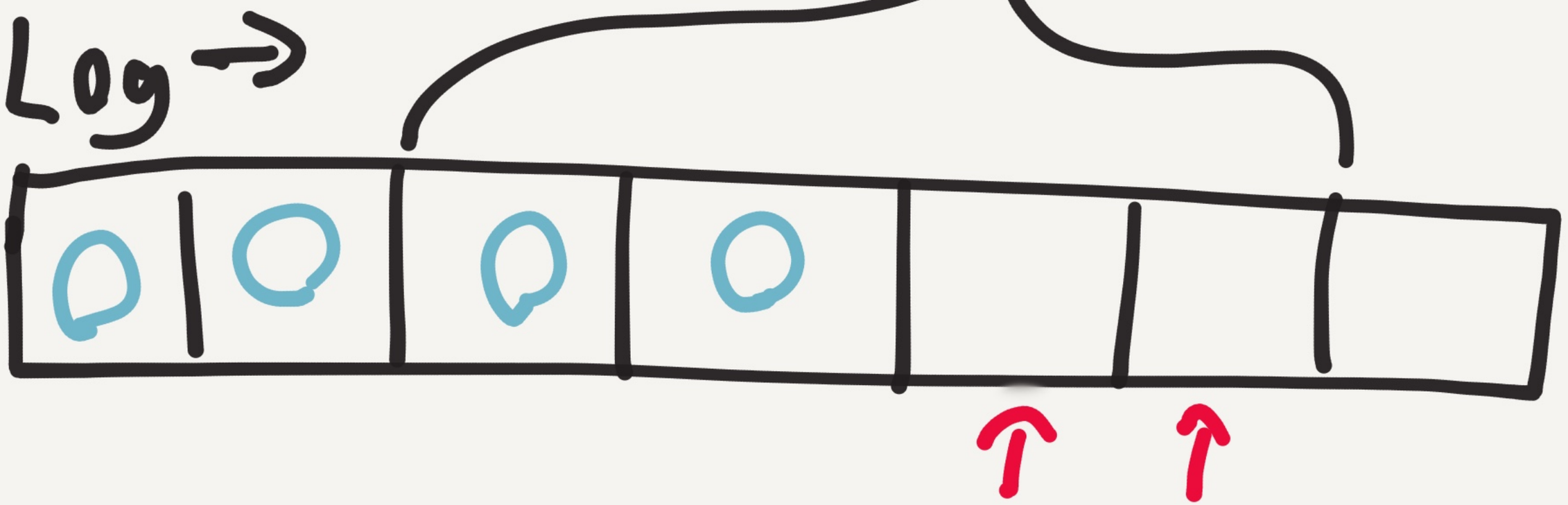
"Guarantee failed:

[index <= get_latest_index]

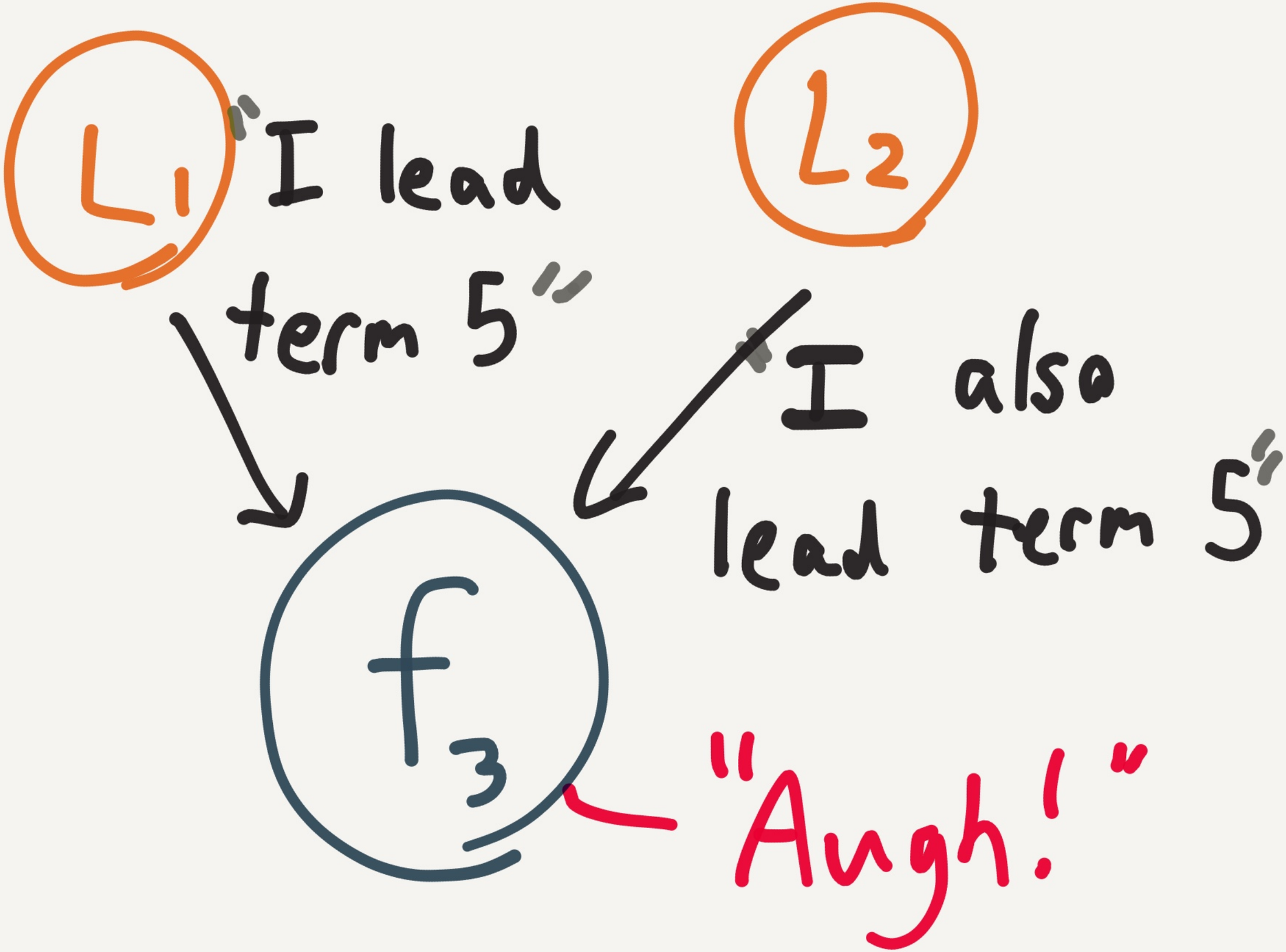
the log doesn't go

forward this far"

Leader: "please apply this range to your SM"



"Guarantee failed:
[current_term_leader_id
== request_leader_id]"


L1
"I lead term 5"
L2
"I also lead term 5"
f3
"Augh!"

seq #

raft ID

START	1	foo
STOP	2	foo

START | 1 | foo

STOP | 2 | foo

dup

START | 1 | foo

START | 1 | foo

STOP | 2 | foo

dup

START | 1 | foo

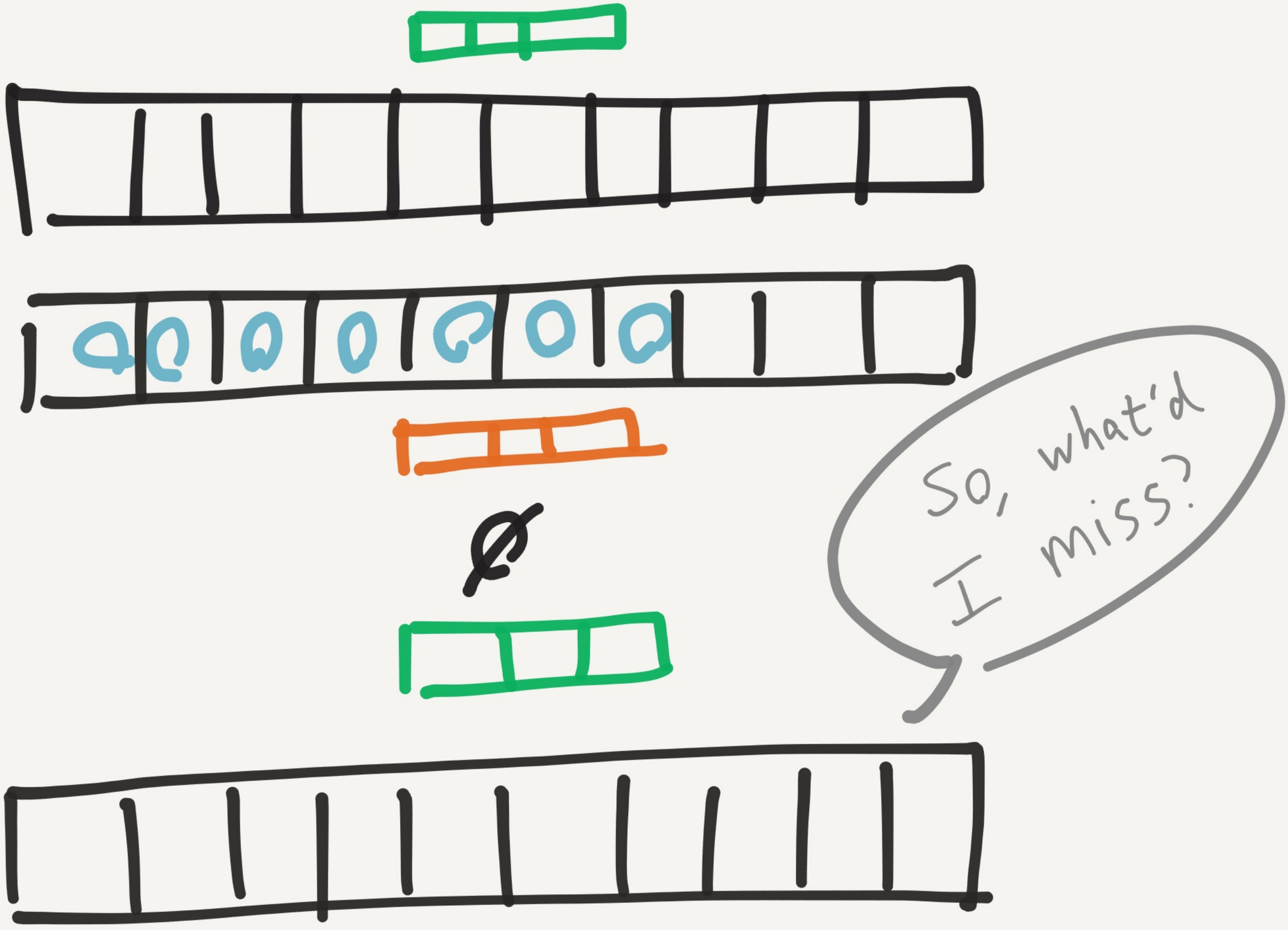
. . . mostly .

#4668

In version 2.1.0,
seq #s set to
MAX_INT.

Workaraund:

Always process ACTIVE messages


So, what'd I miss?

Patches in

2.2.4 &

2.1.6

Recap

Read the docs!

CAREFULLY

Then test it

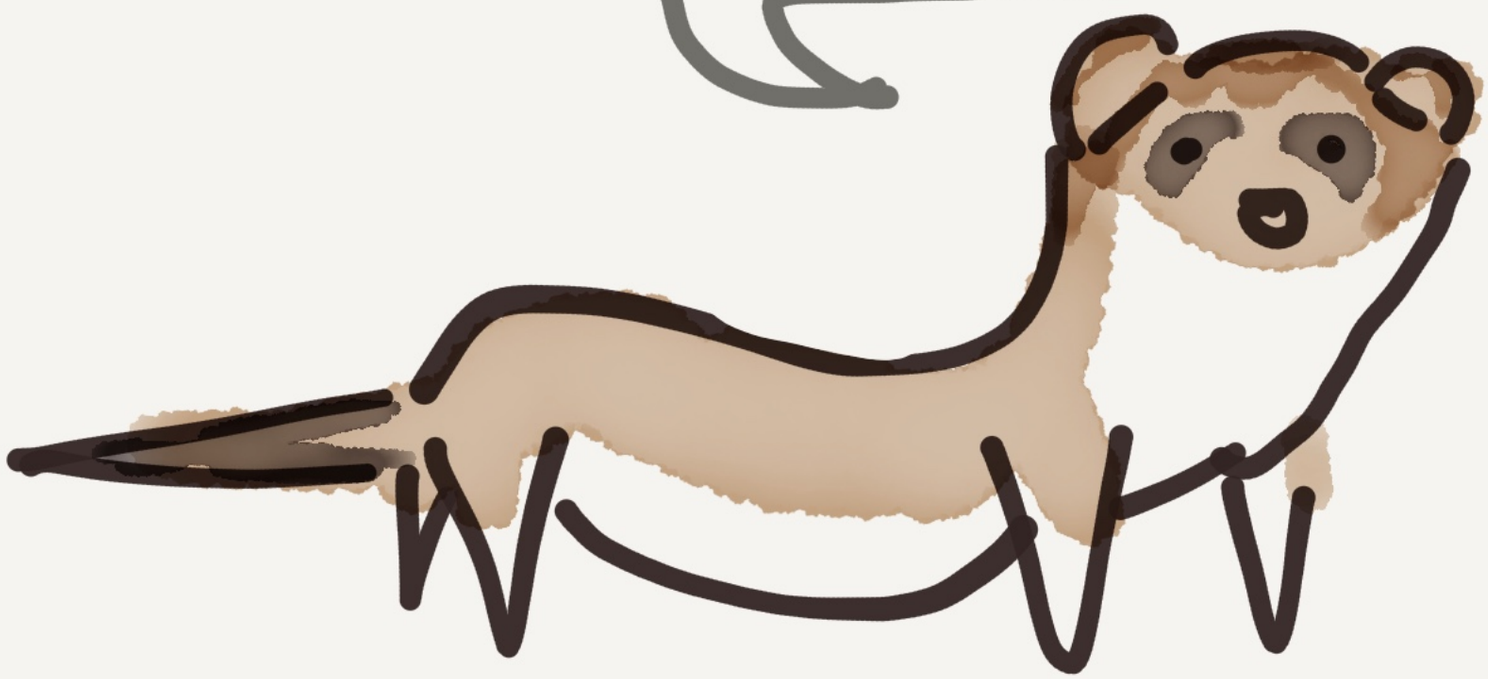
— for —

YOURSELF

"Strict"

"Strong"

"ACID"



Be Formal

&

Be Specific!

Figure out the invariants your system needs

How much

can yan tolerate?

Consider your failure modes

Process Crash

#kill -9 1234

Node failure

- AWS terminate
- Physical power switch

Clock Skew

```
# date 10280000
# faketime ...
```

SC/IO Pause

```
# killall -s STOP foo
# killall -s CONT foo
```

Network Partition

```
# iptables -j DROP
```

```
# tc qdisc ... delay ...
```

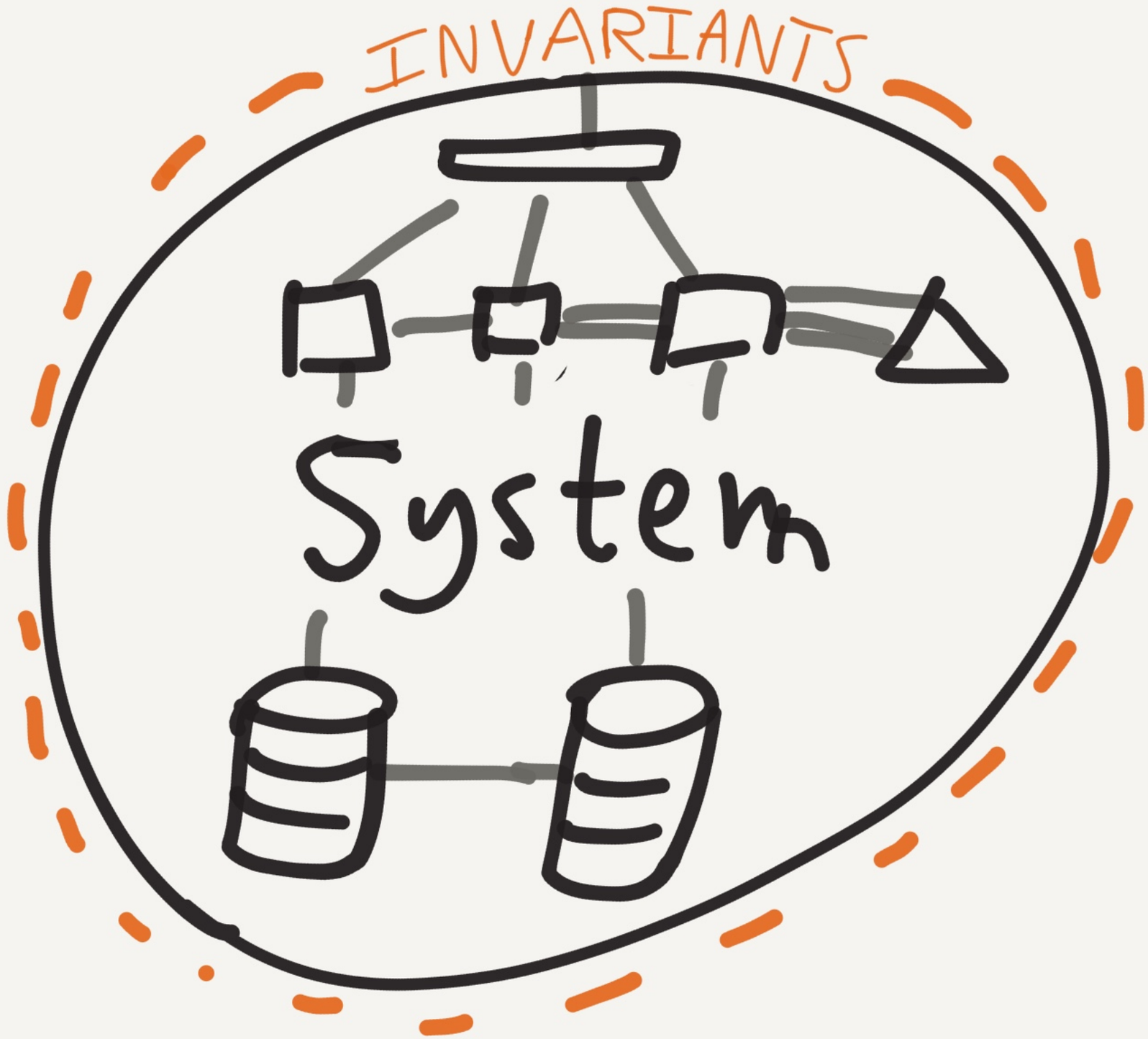
```
drop ...
```

Test your

Systems

end ——— to ——→ end

vs.



- Property testing
- High-level invariants
- With distsys failure modes

Thanks

sjaakala

Siddarth Chandrasekaran

Brendan Taylor

Cosmin Nicolaescu

Thanks

Brenden Matthews

Timothy Chen

Aaron Bell

Kyle Conroy

Thanks!

Daniel Mewes

Michael Lucy

Slava Alchmechet

Thanks!

Codership

Mesos (sphere)?

Thanks!

Stripe

RethinkDB

} funded work

http://jepsen.io