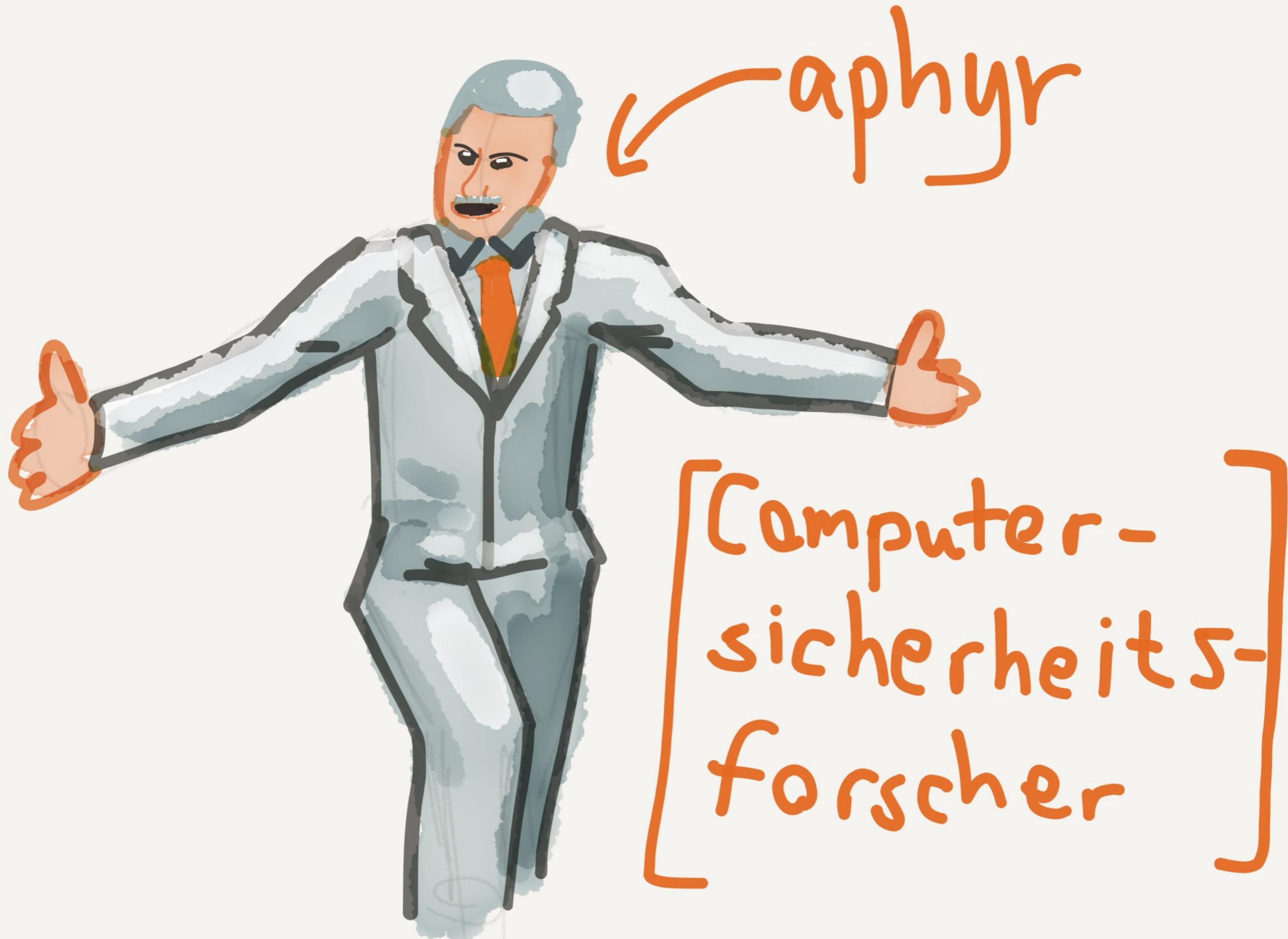

Jepsen

5

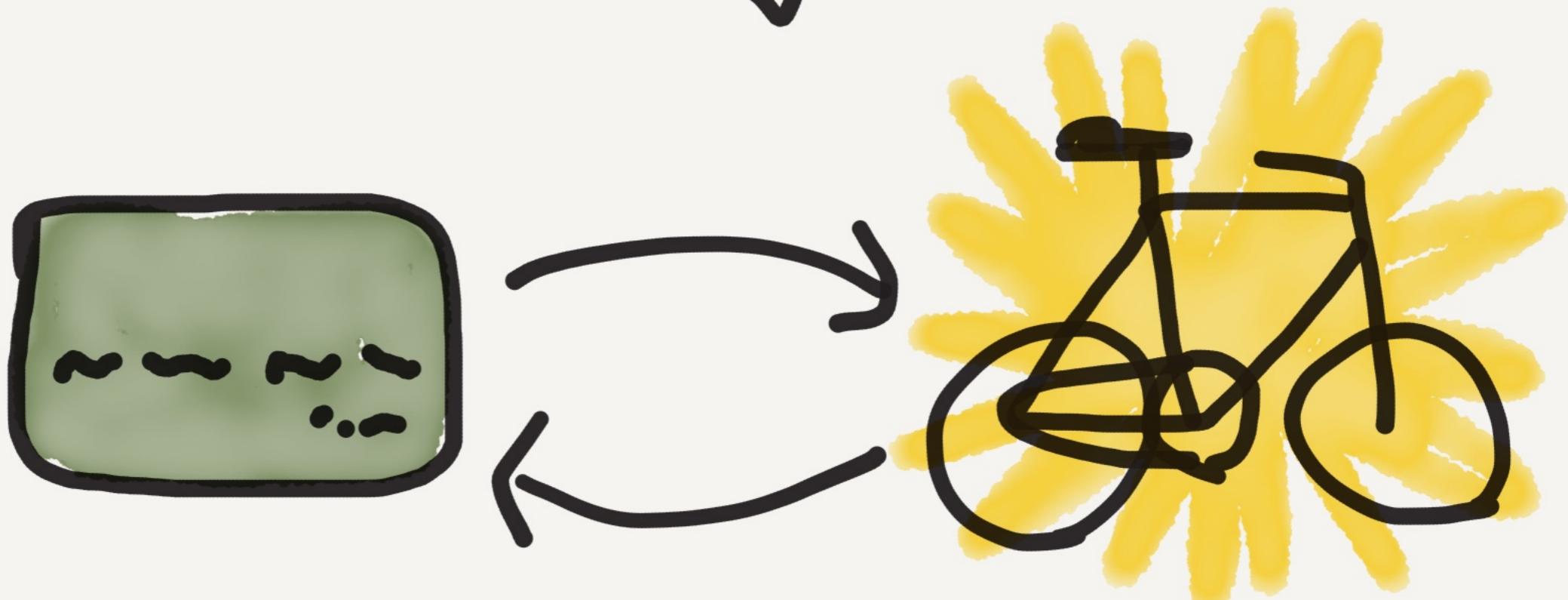
WHAT
EVEN
ARE
COMPUTERS





[Computer-
sicherheits-
forscher]

style



Public
API {



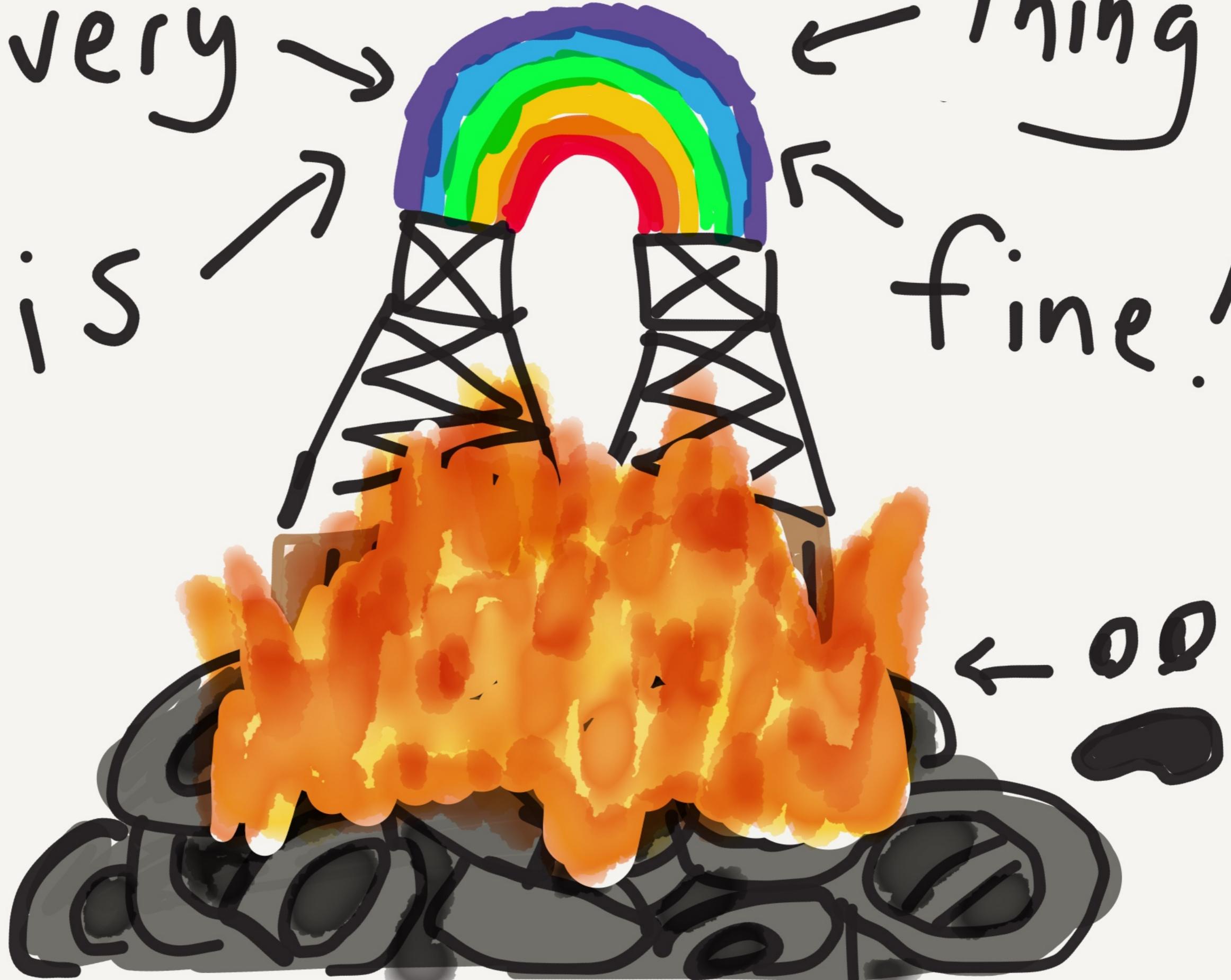
} API code

Ruby {

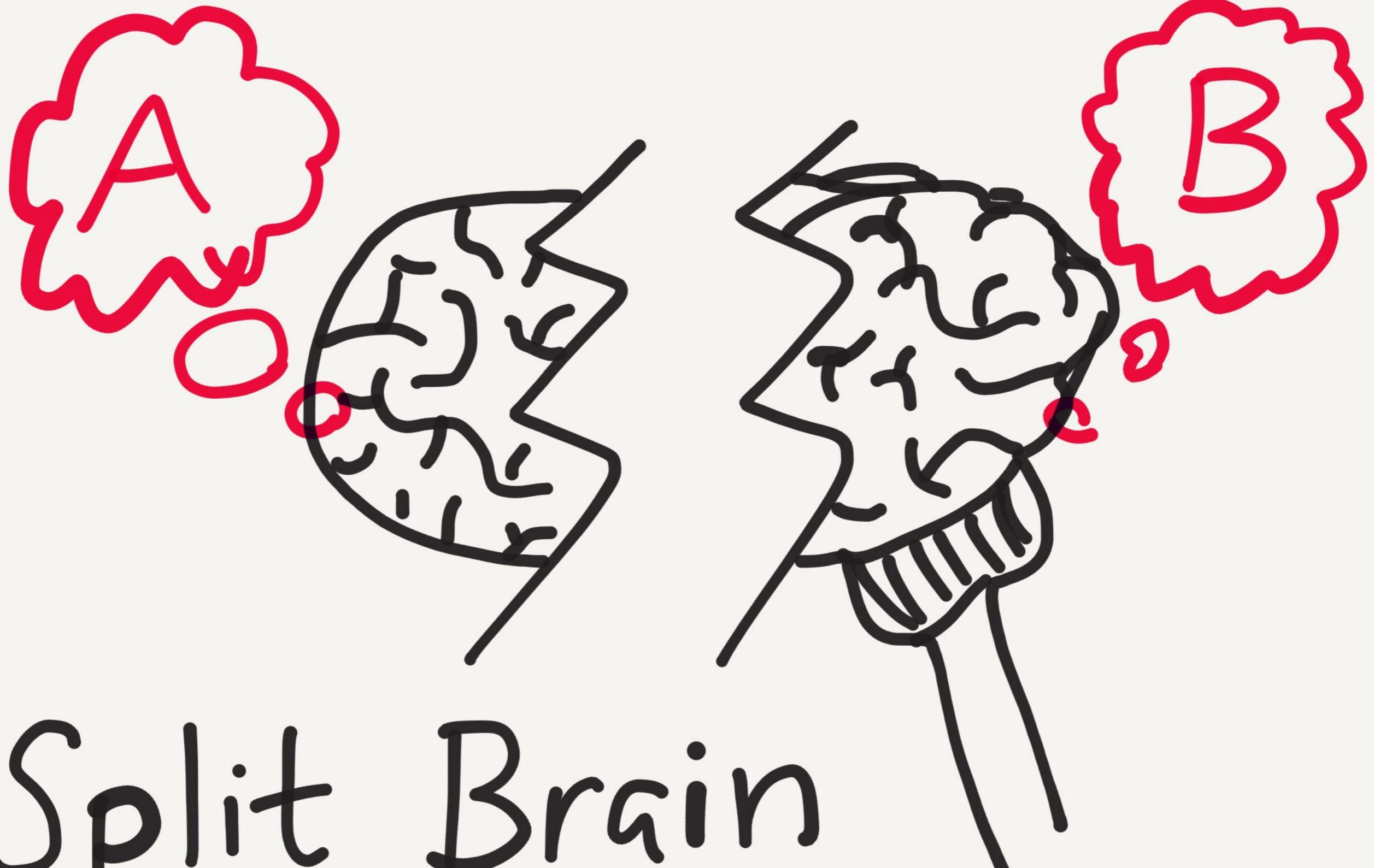


DBs

Every →
is →
← Thing
fine !

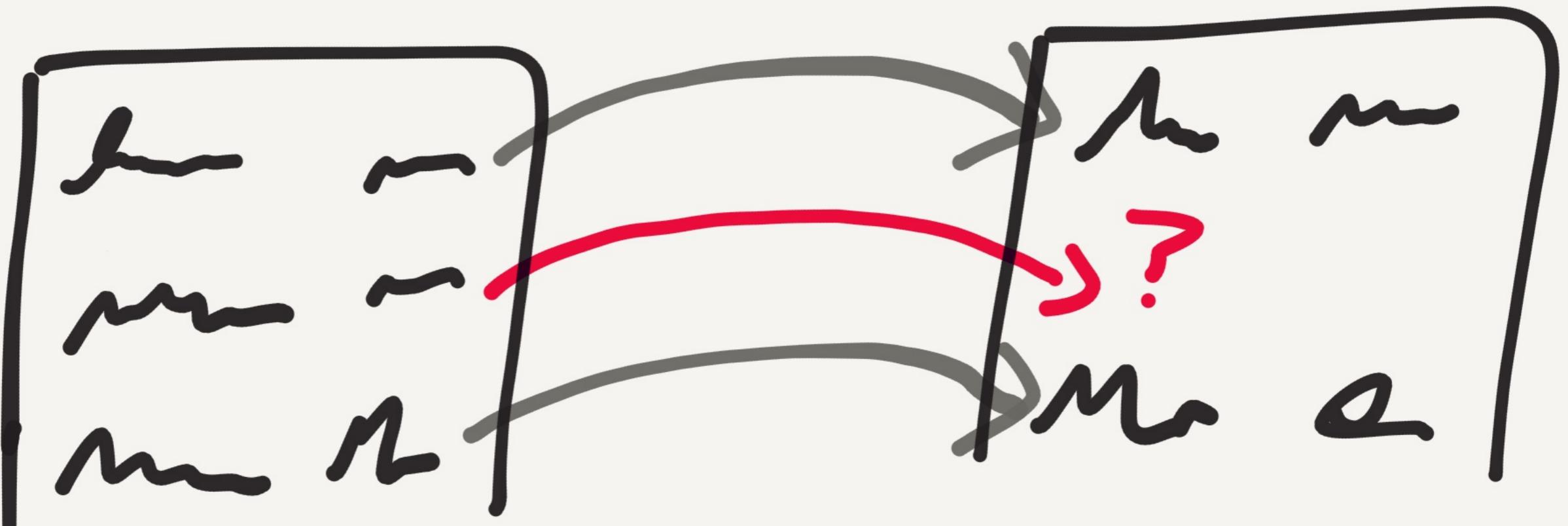


Databases! /
Queues! /
Discovery! /
THE HORROR

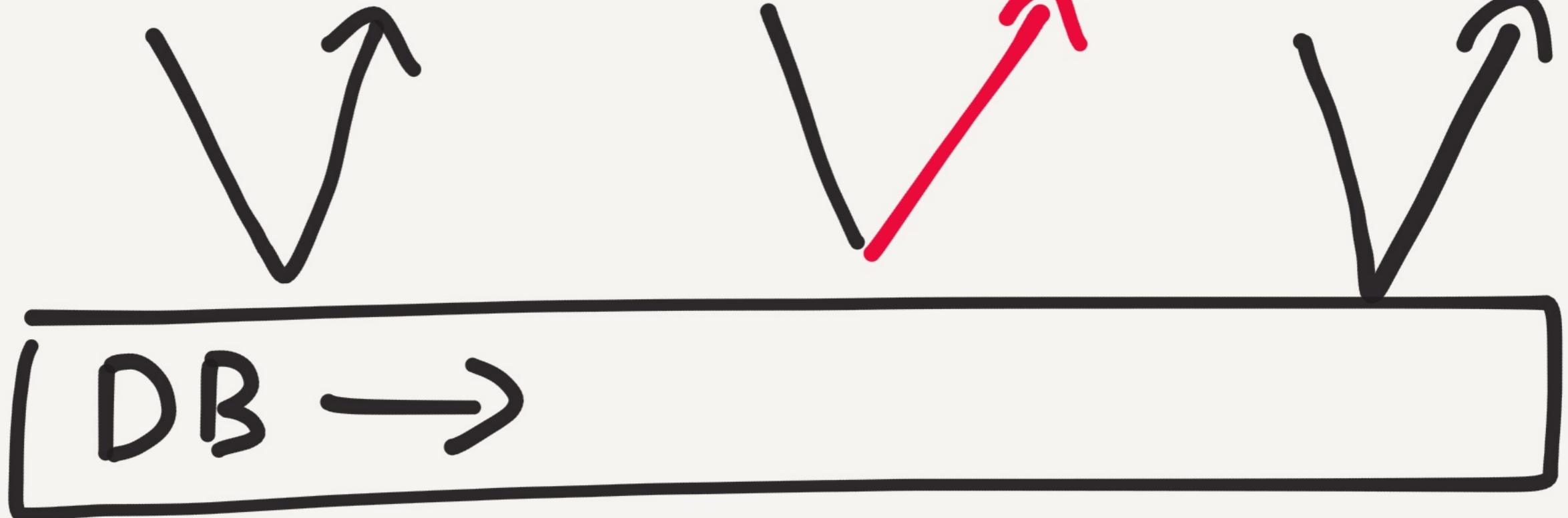


Split Brain

Broken Foreign Keys



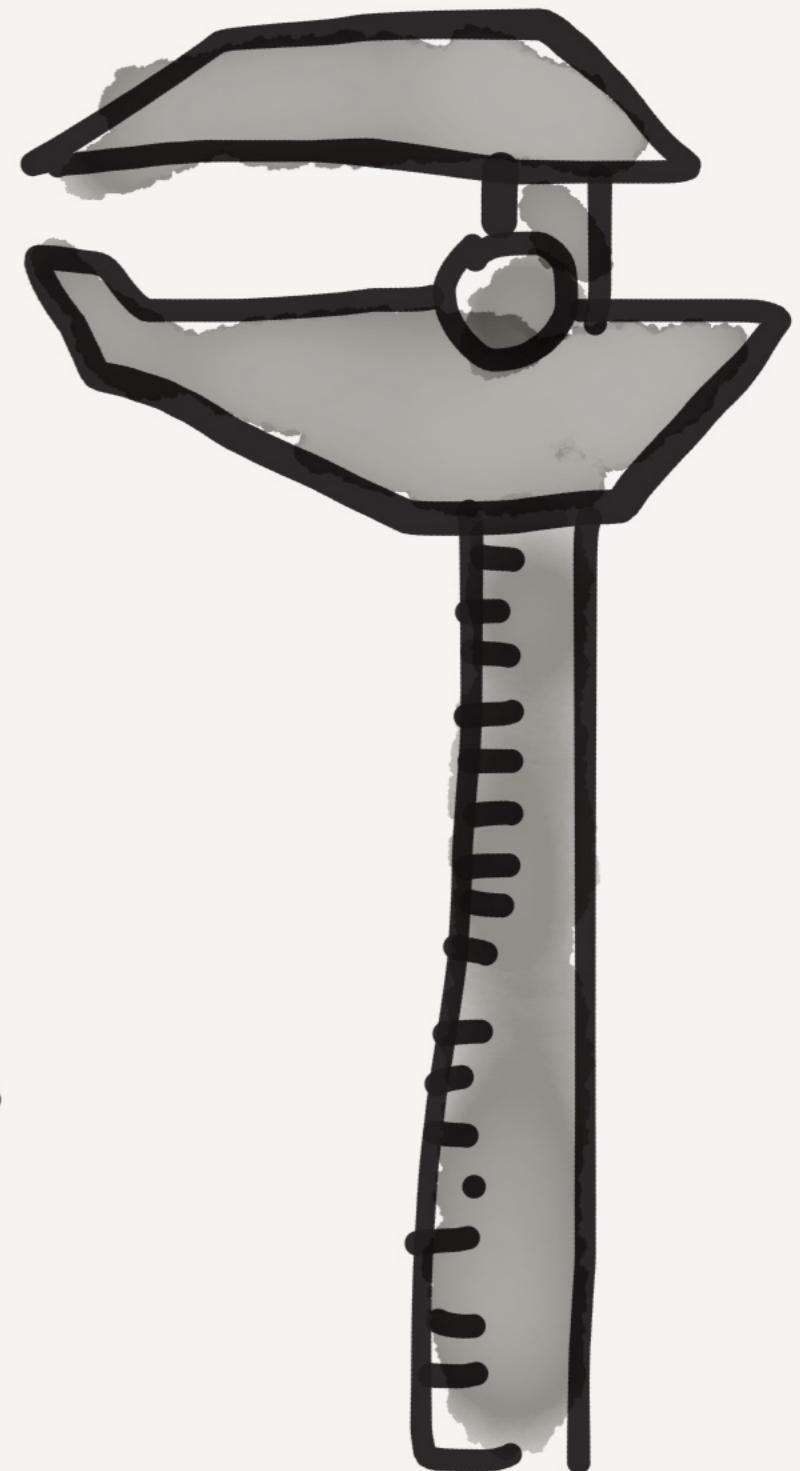
write ok read \emptyset read ok



Anomalies

How do you
know if a
system is safe?

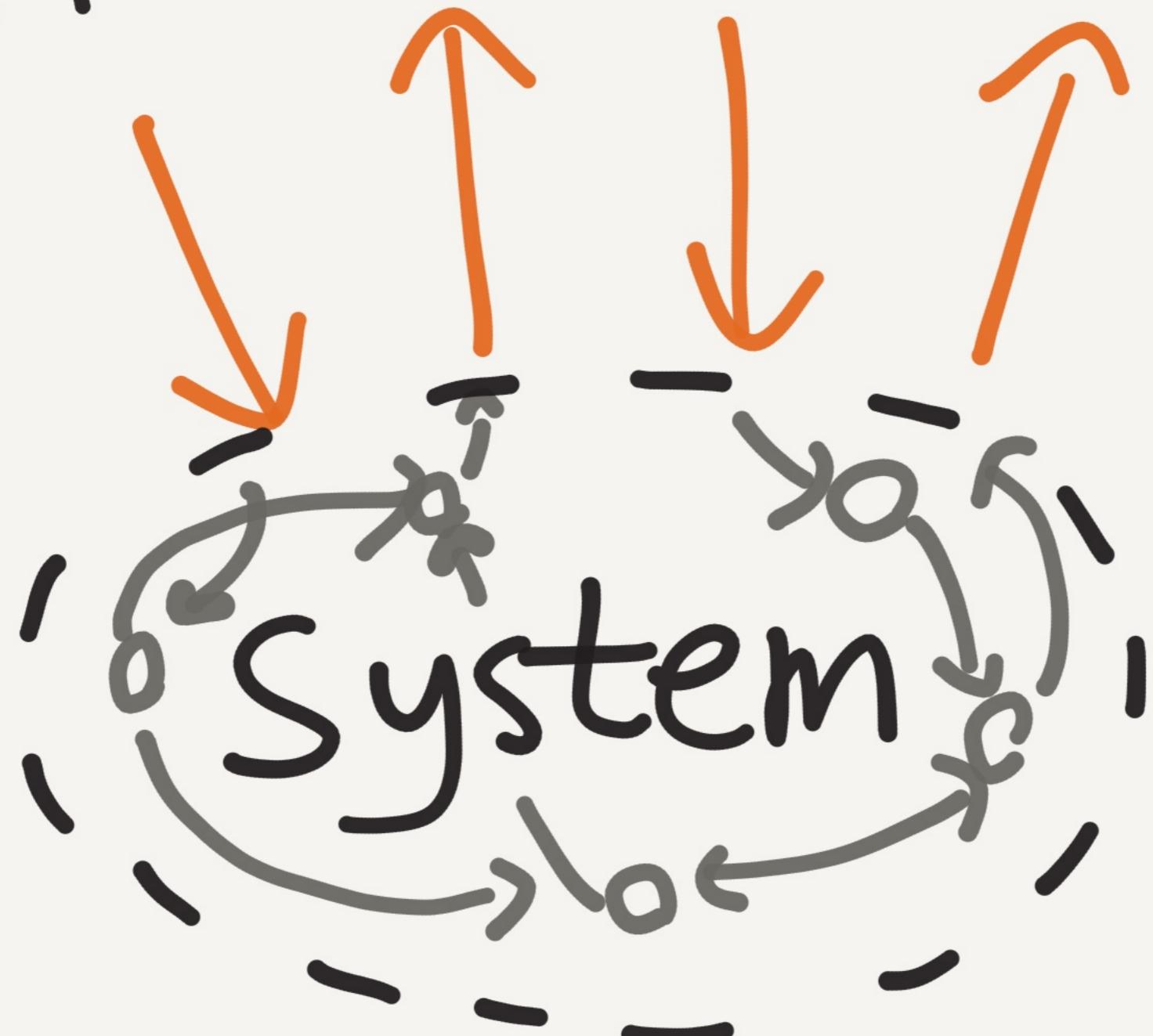
Measure
your
Systems



Jepsen

github.com/aphyr/jepsen

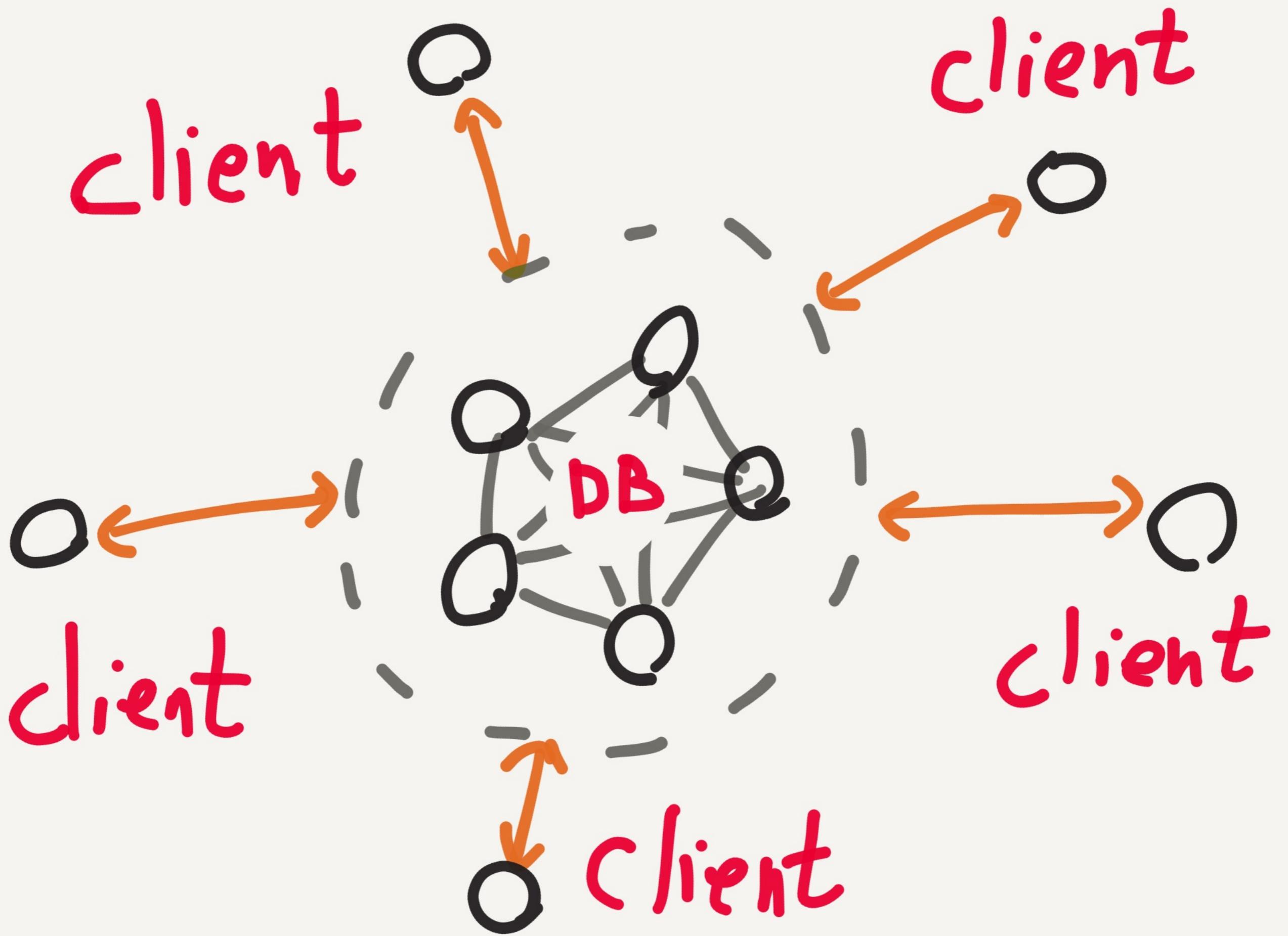
Environment





— INVARIANTS —





client: $w-w'$ $r-r'$ $w-w'$

The diagram illustrates three client requests to a central database system. Each request is shown as a horizontal line segment with arrows at both ends, indicating a transaction or query. The first request is labeled $w-w'$, the second $r-r'$, and the third $w-w'$. All three requests converge on a central, blurred area representing the database.

DB :

client : w — w' w — ... -

The diagram shows the internal state of the database after the client requests have been processed. The database is represented by a central cloud-like shape. Inside, there are two nodes labeled w and w' . A horizontal line connects them. A vertical arrow points downwards from the center of this line to a dashed line below, representing a log or history of operations.

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



invoke

ok

invoke

fail

invoke ?

? info

?

?

?

invoke ok

invoke fail

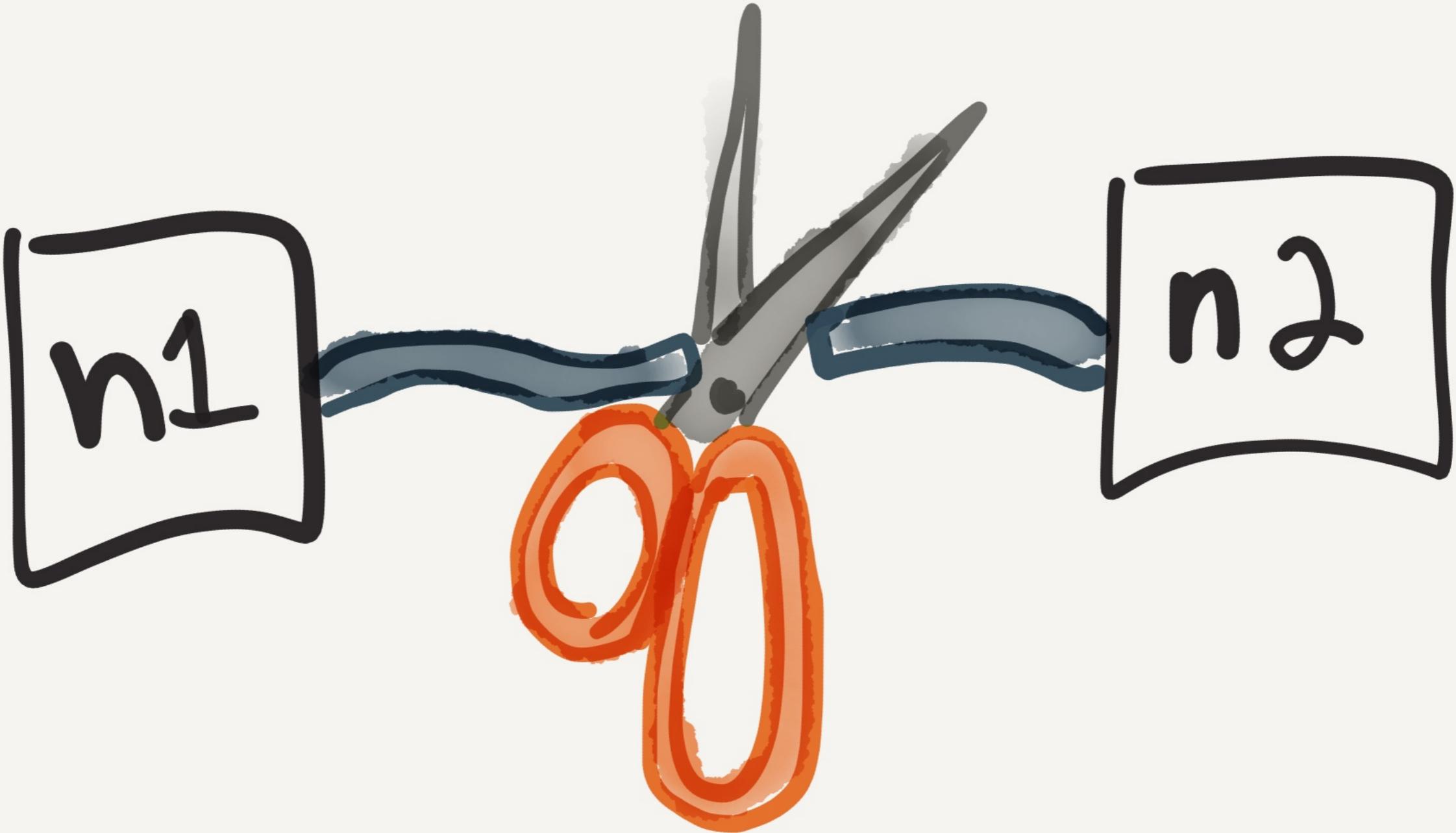
invoke ok

invoke info

Bollersdorf

700%

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



Partitions!

So, what
have you
found?



Riak

LWW \rightarrow lost writes

CRDTs \rightarrow safe

5/13

Mongo

Data loss at all

Write Concerns

Redis Sentinel

5/13

Split brain,
massive write loss

Cassandra

9/13

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

NuoDB

Beat CAP by buffering all requests in IRAM during partition

9/13

Kafka

In-sync Replica Set

Could shrink to 0

nodes, causing msg

loss.

9/13

Zookeeper

Works.

etcd / Consul

6/14

Stale reads

Elastic search

Loses documents in
every class of partition
tested.

6/14

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

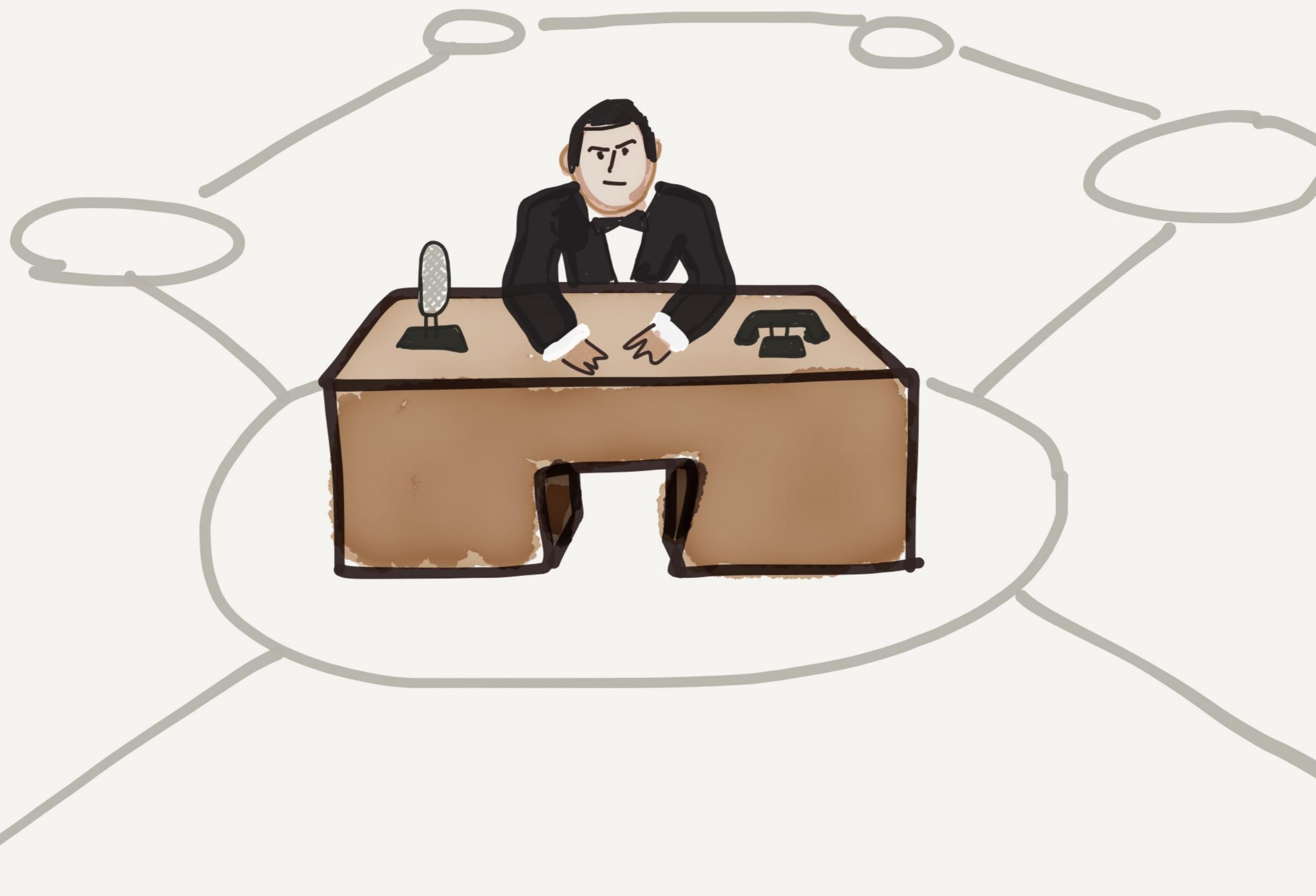
dirty reads

As an industry, we are

woefully underprepared

to handle distributed

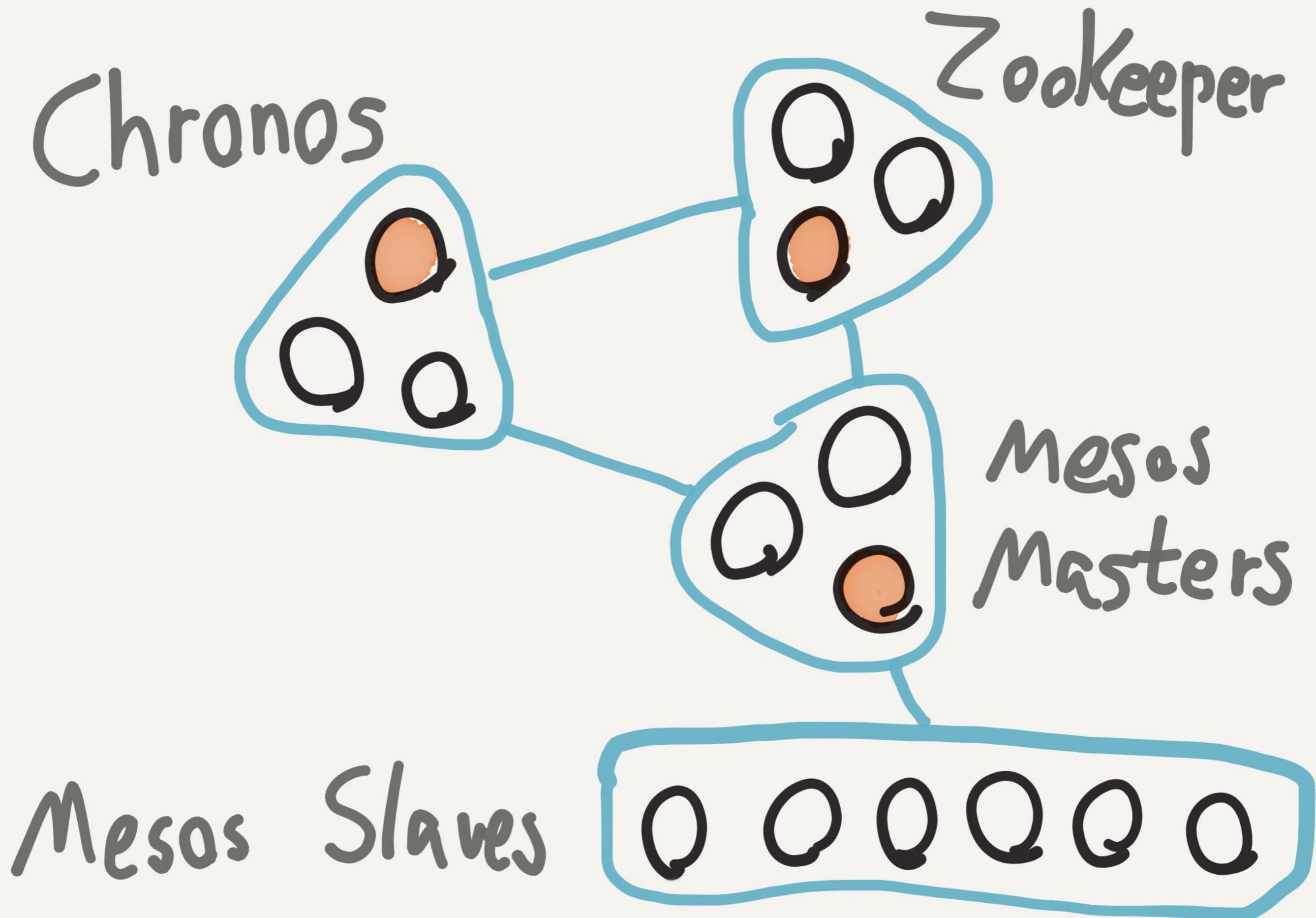
systems failure modes



Chronos

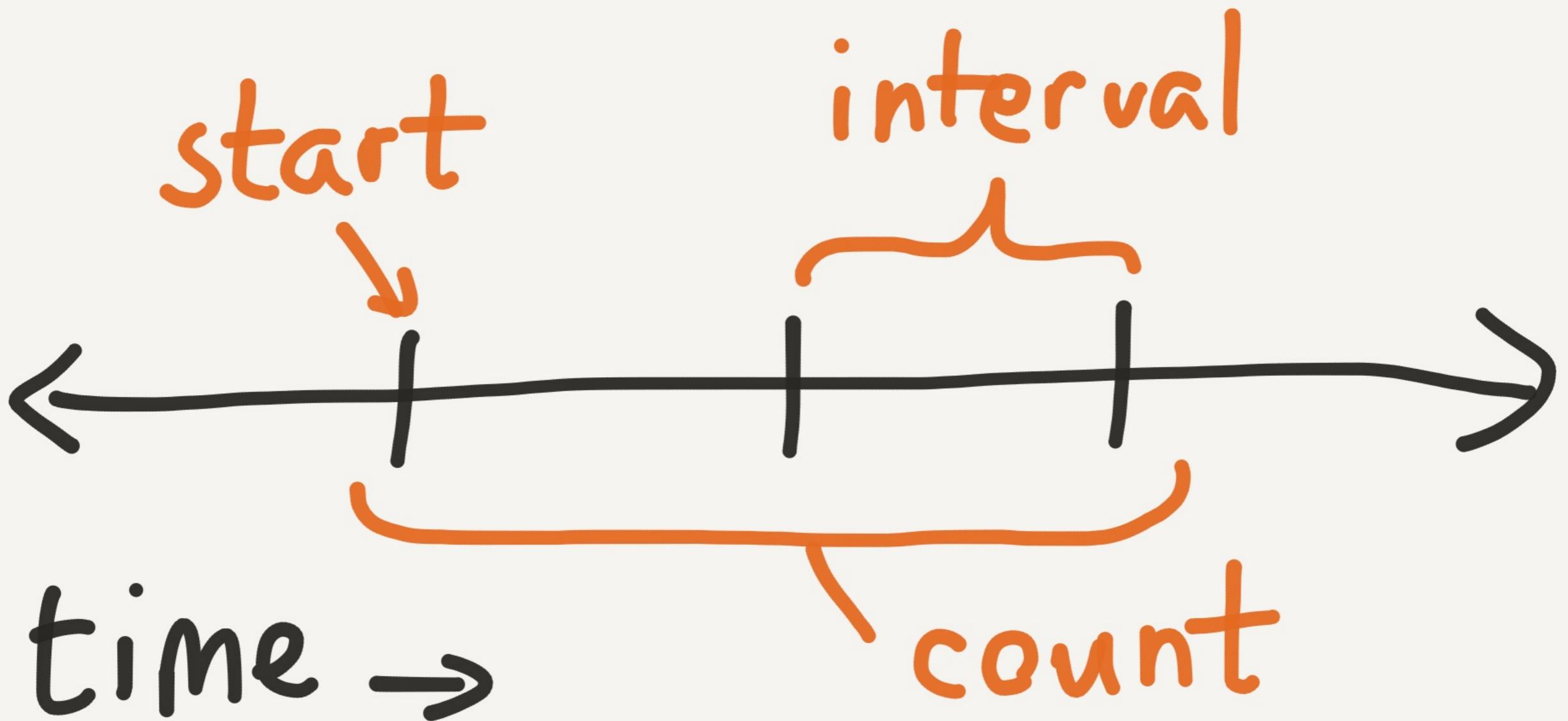
Like Cran, but

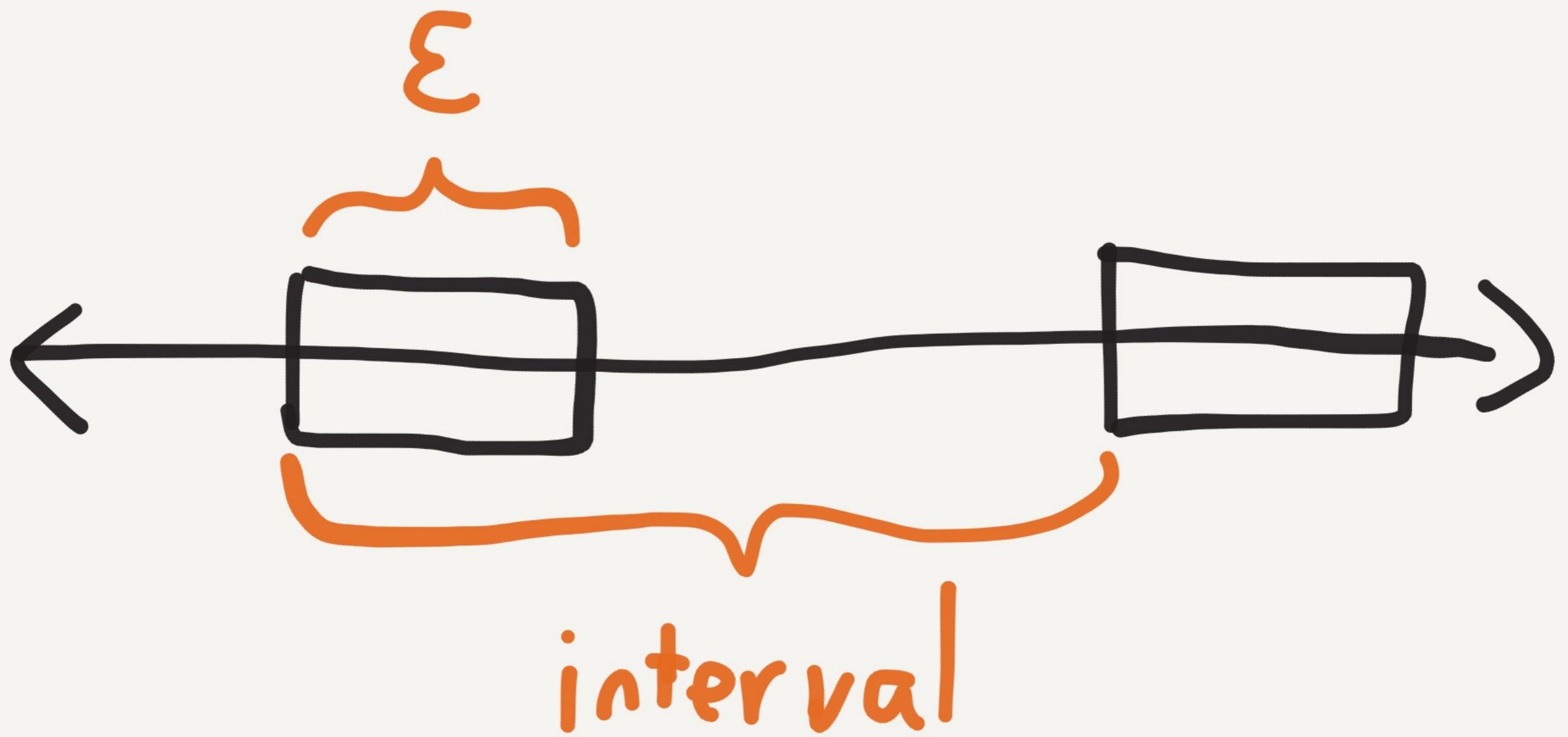
DISTRIBUTED

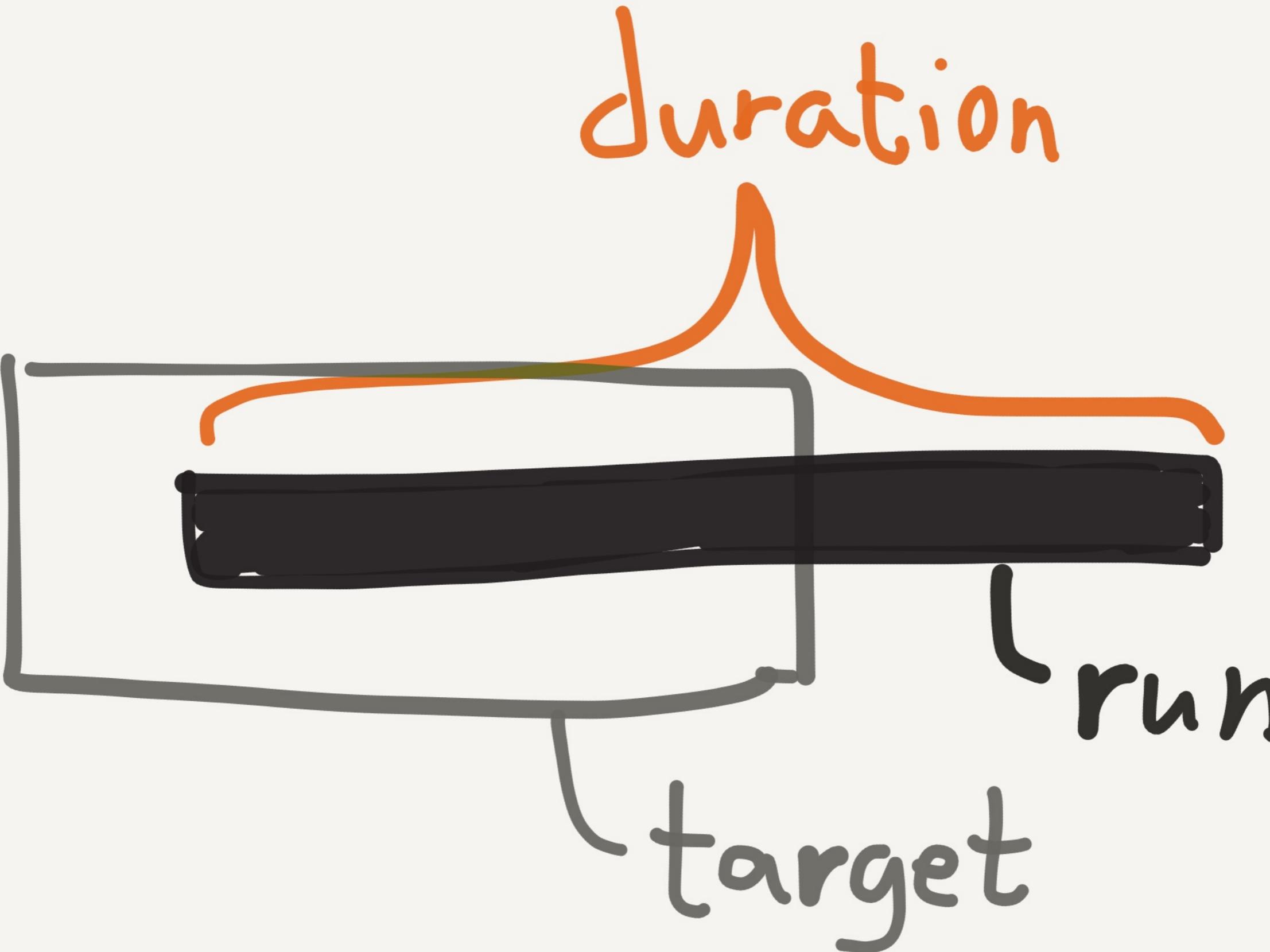


n1	ZK	Chr	MM
n2	ZK	Chr	MM
n3	ZK	Chr	MM
n4	ZK	Chr	MS
n5	ZK	Chr	MS

Job Spec



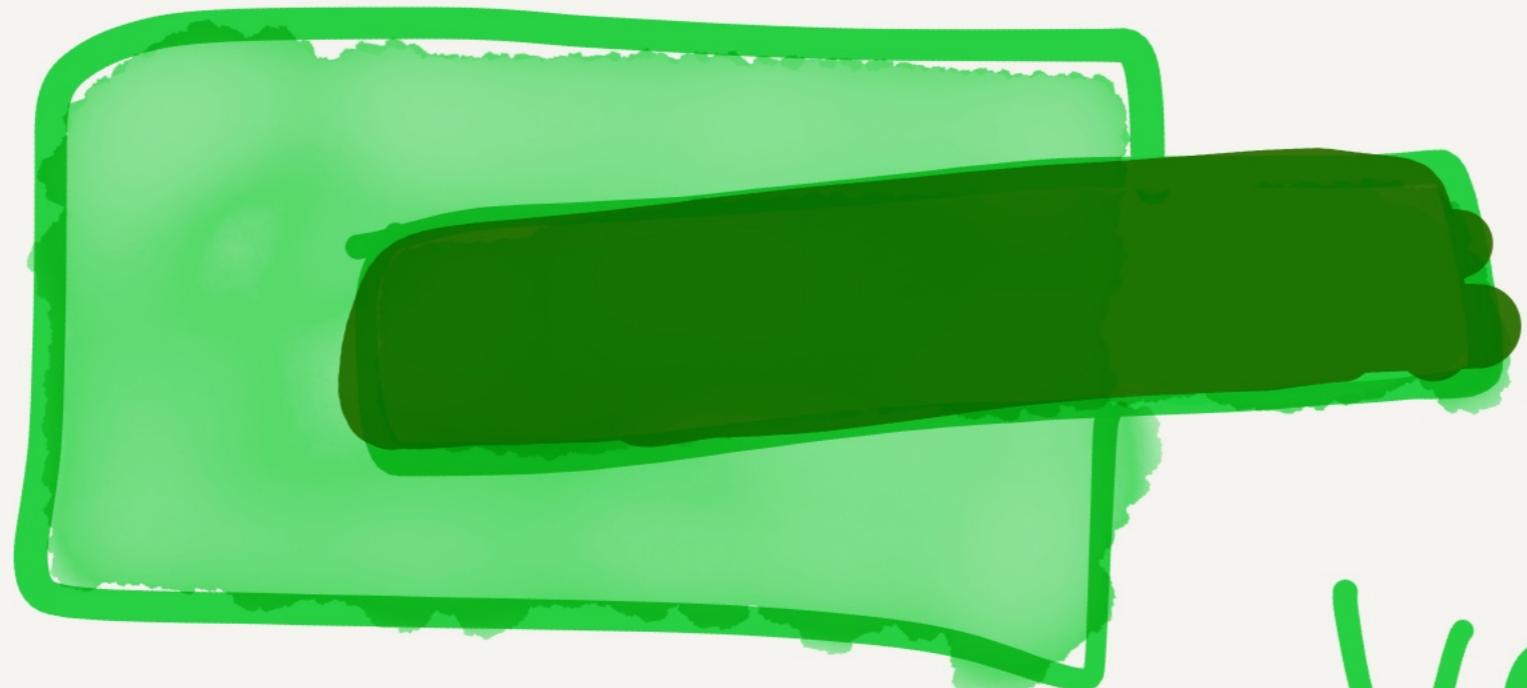




duration

run

target



Valid



invalid

valid



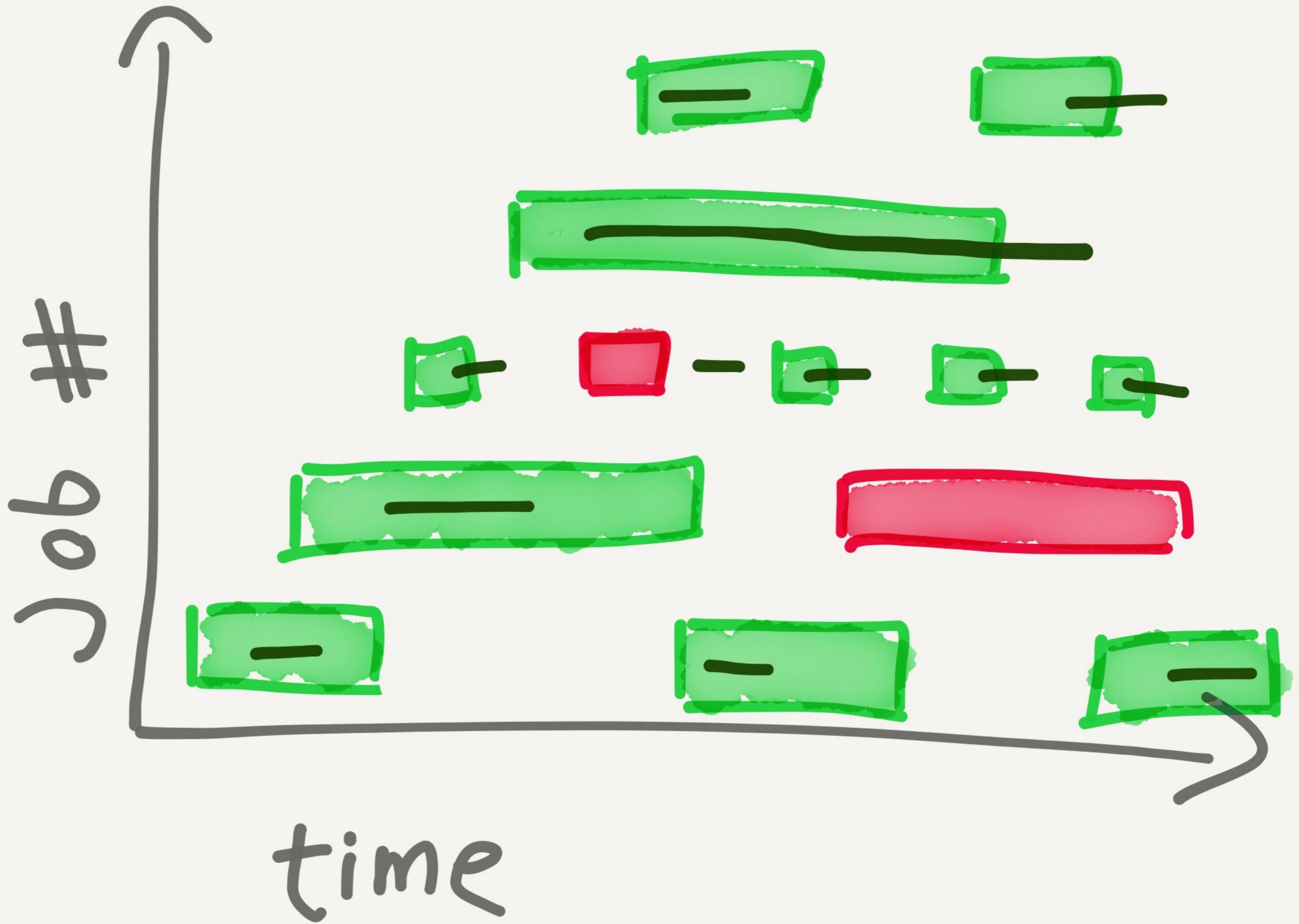
invalid



valid



time →



Fun Things

- Empty/invalid HTTP responses
- Connections refused
- Blank 400 responses

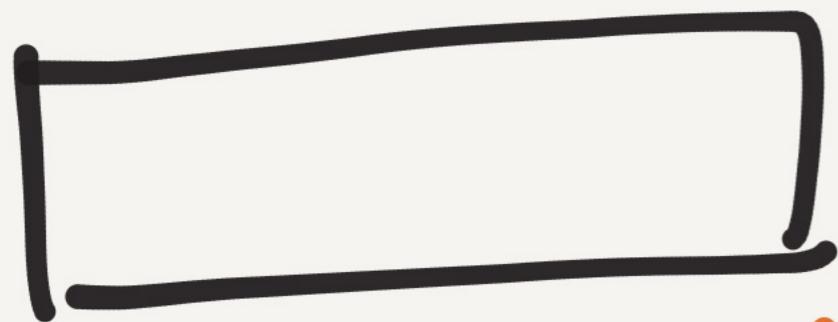
Chronos & Mesos

processes will crash
on losing zk conn.

(By design)

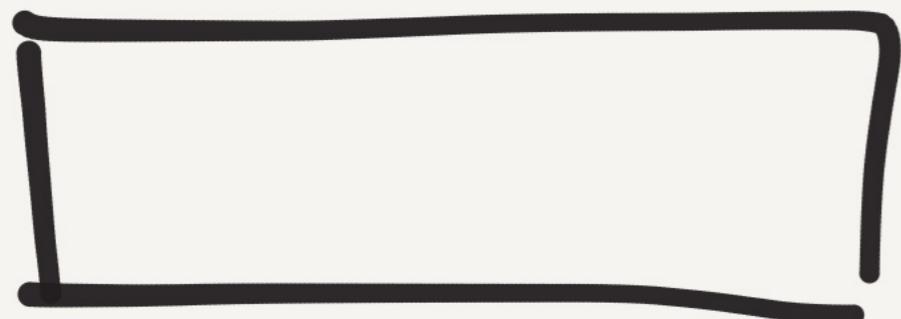
(Except when they don't)

--schedule_horizon=60s



horizon

Epsilon is a fuzzy thing

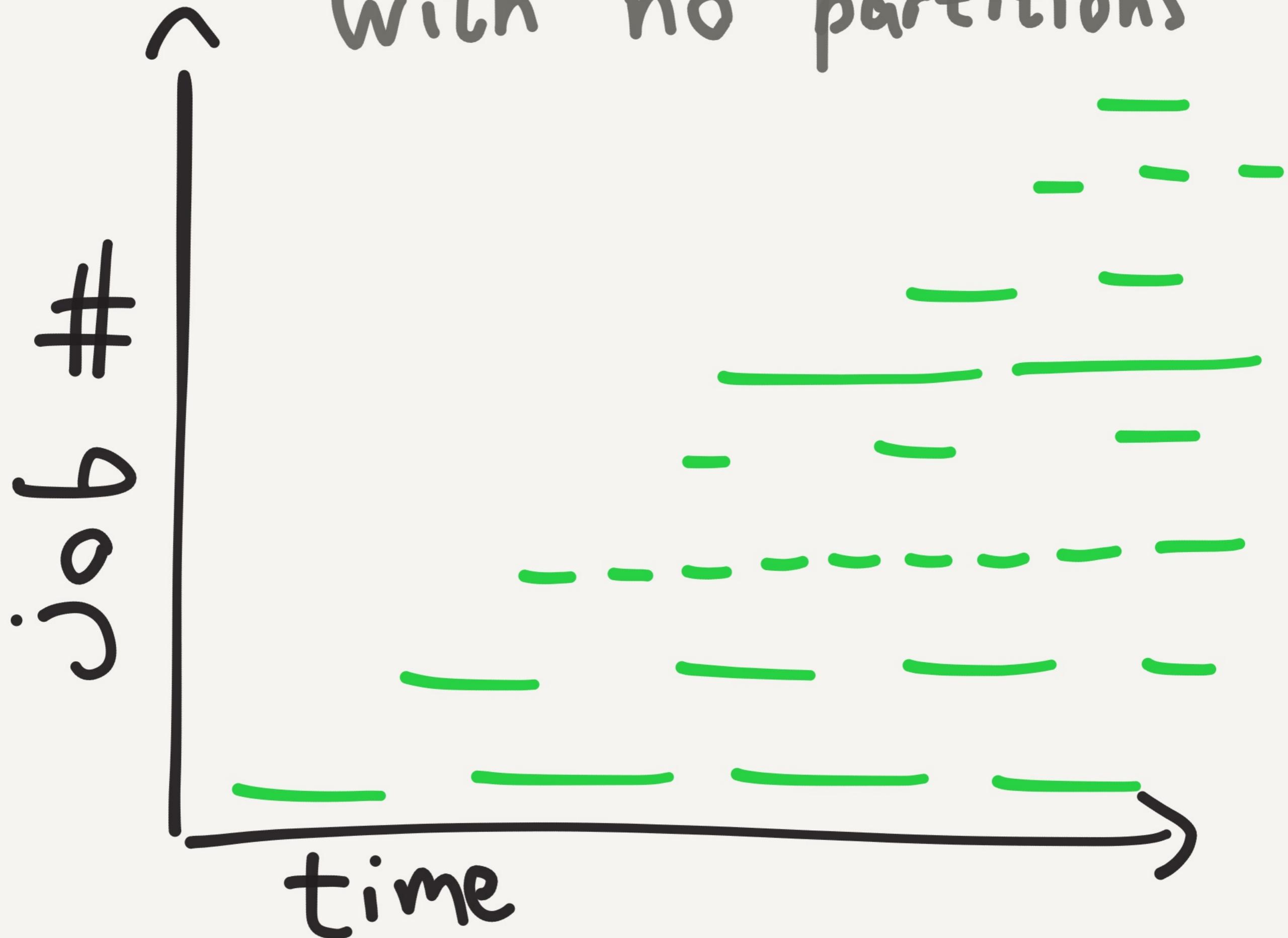


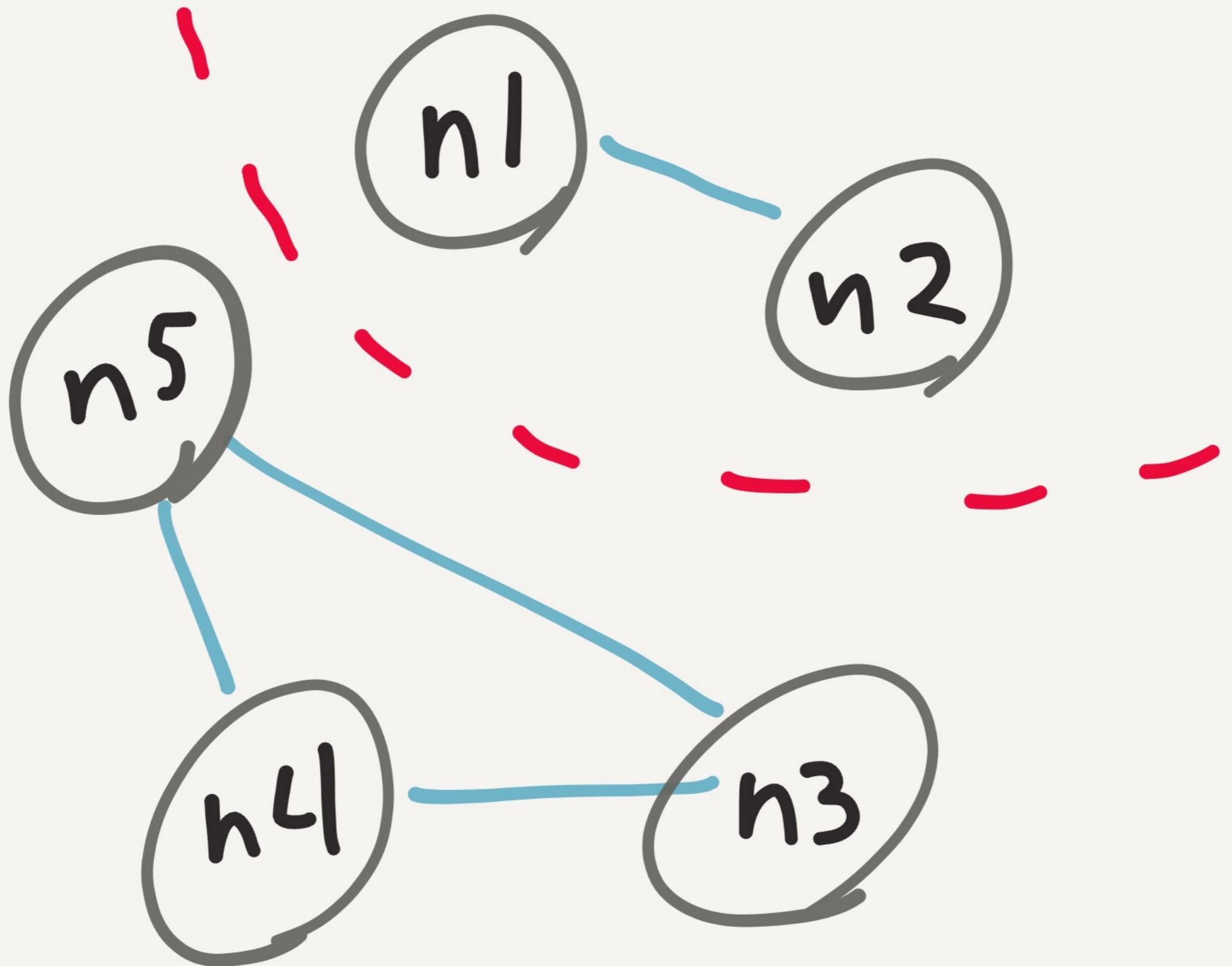
ϵ

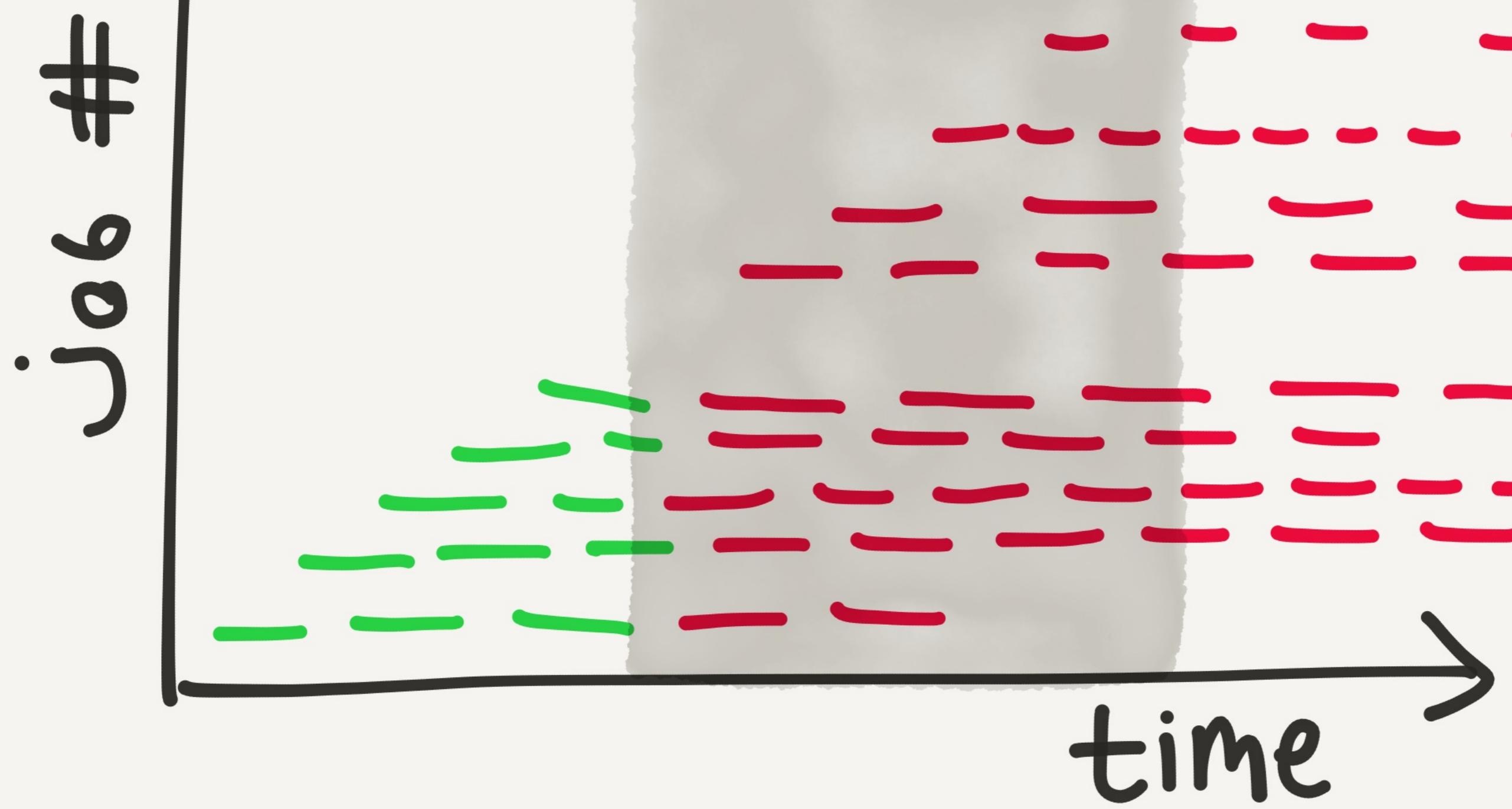
A large orange wavy line with a small orange circle at its end, resembling a hand-drawn letter 'e'.

"It's mostly
on time..."

with no partitions







Partition

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 Chronas

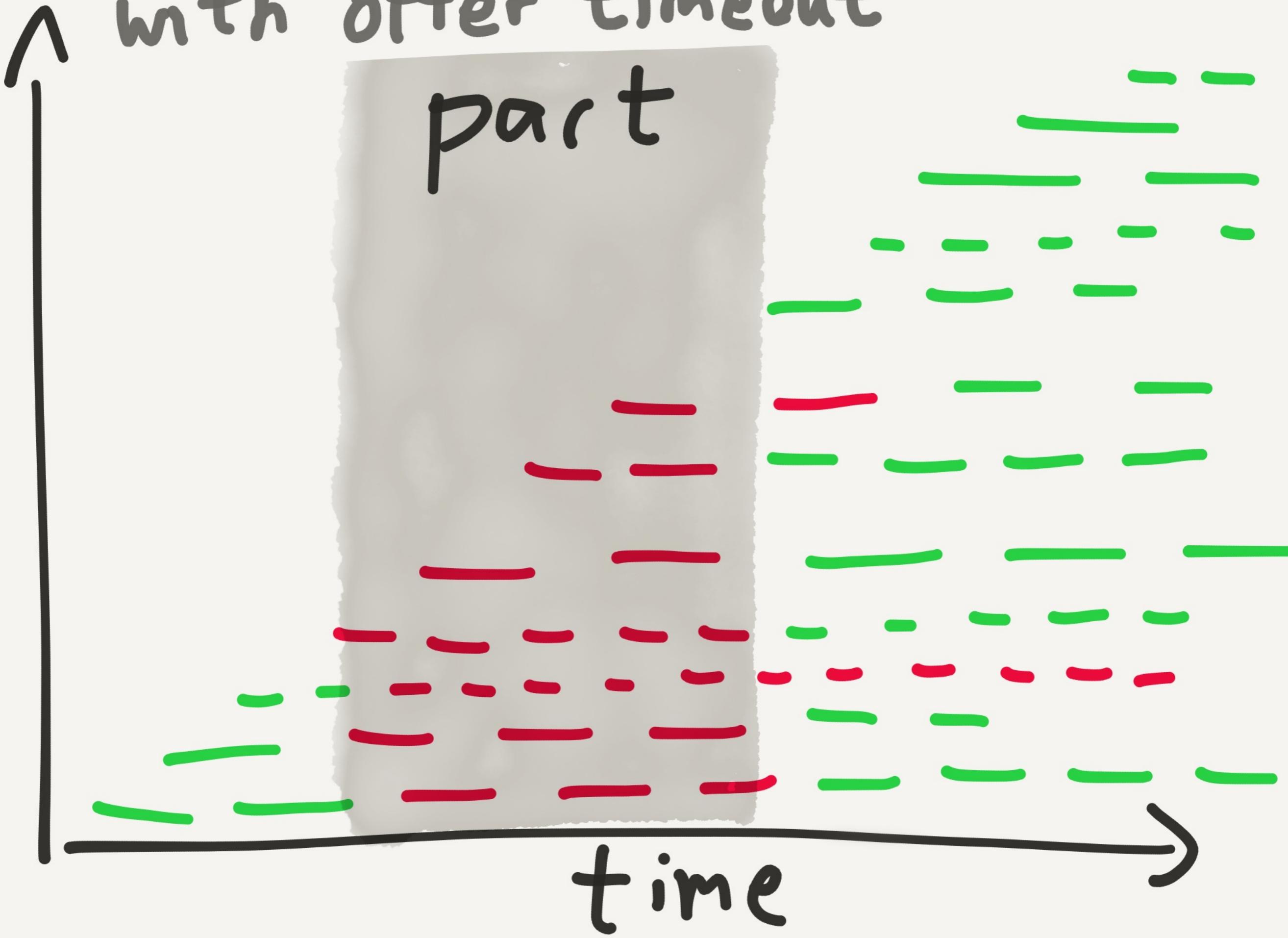
a c o Q a o o e e Q
c c a a a Q Q Q e e Q Q

New Chronas

--offer_timeout=30s

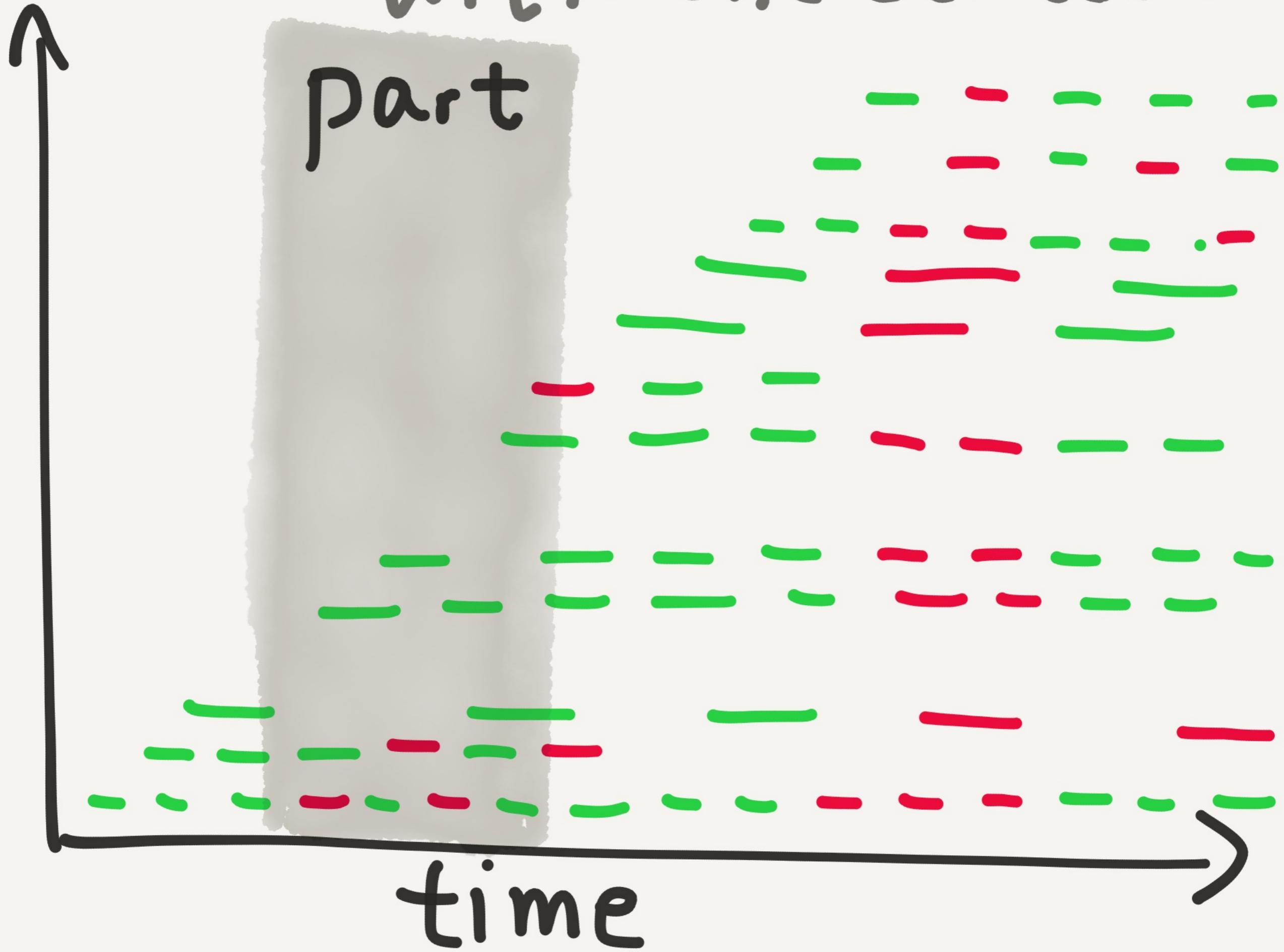
with offer timeout

part



with offer_timeout

Part



Chronos

Recommendations

It's not gonna

run your jobs

oh time.



Colacate quora!

ZK

MM

Chrina

ZK

MM

Chronas

ZK

MM

Chmnas

- Wrappers for Chronos +

Mesos daemons

- schedule_horizon

- offer_timeout

- Monitor your jobs

- Monitor daemon restarts

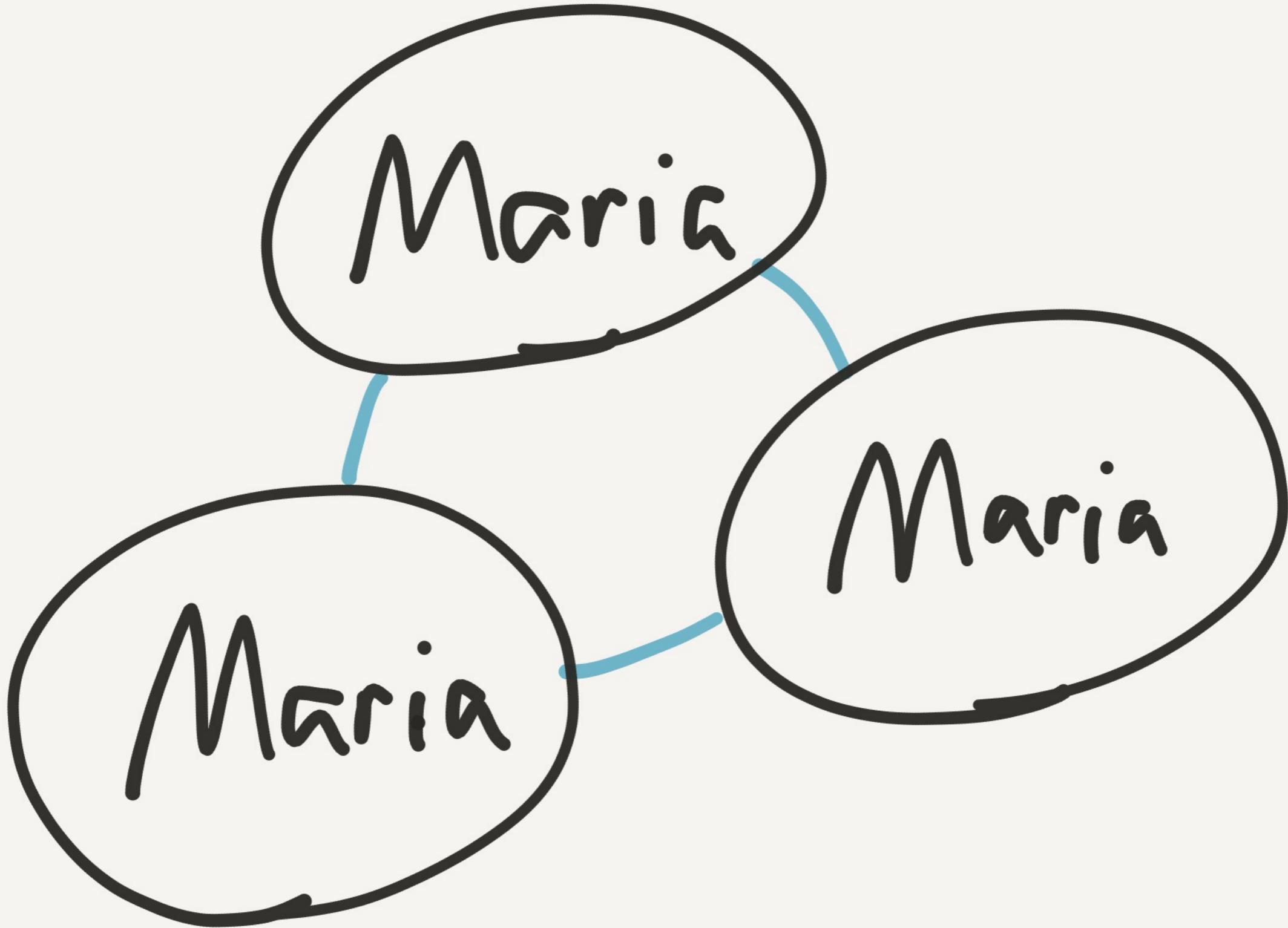


Symmetric replication

for MySQL,

MariaDB,

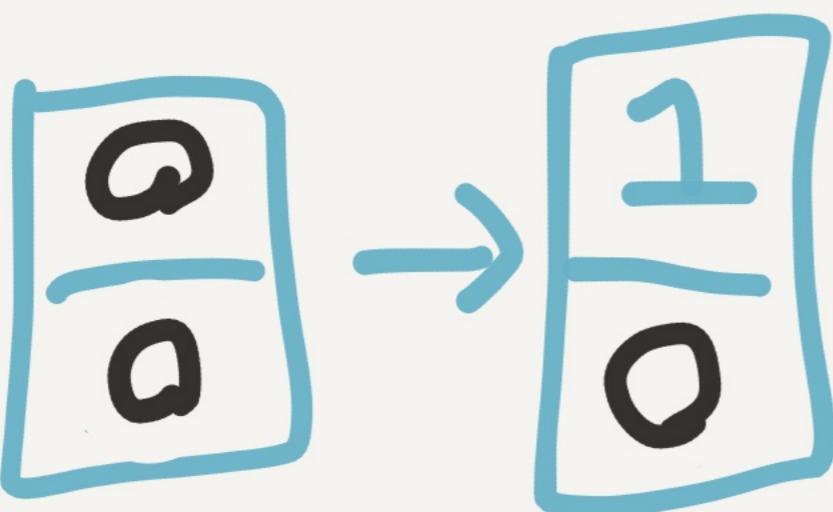
Percona XtraDB, ...



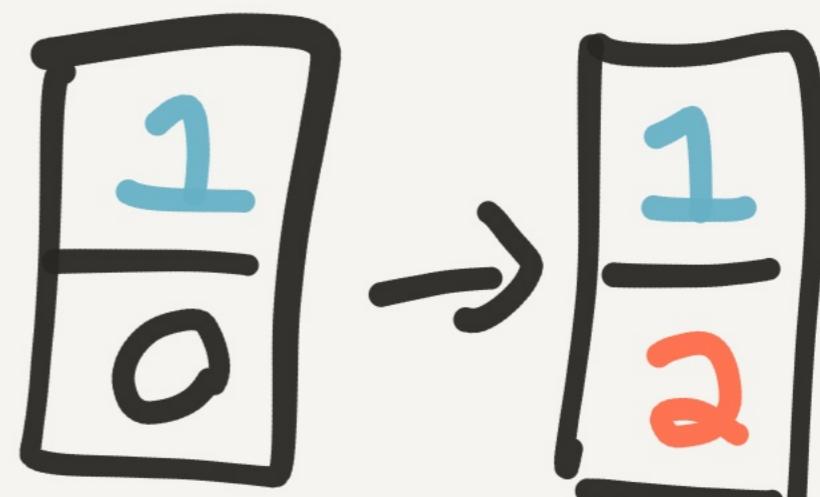
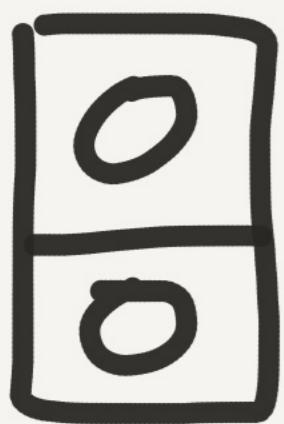
Between nodes,
claims to provide

Snapshot Isolation

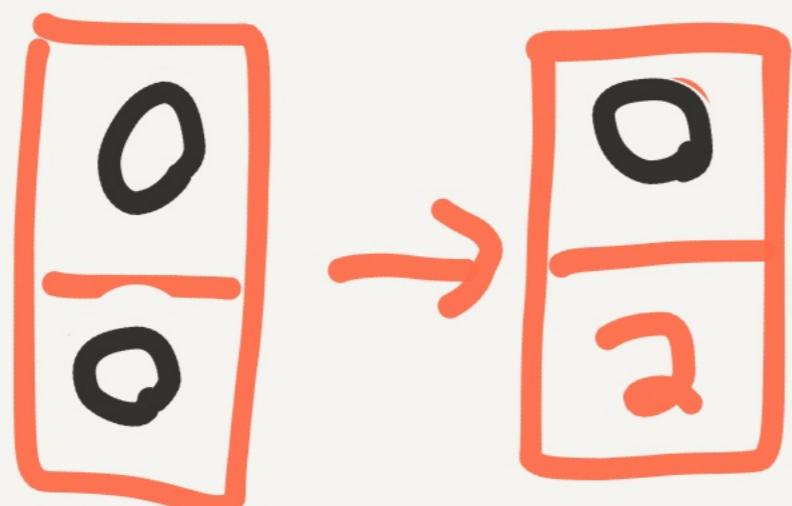
write



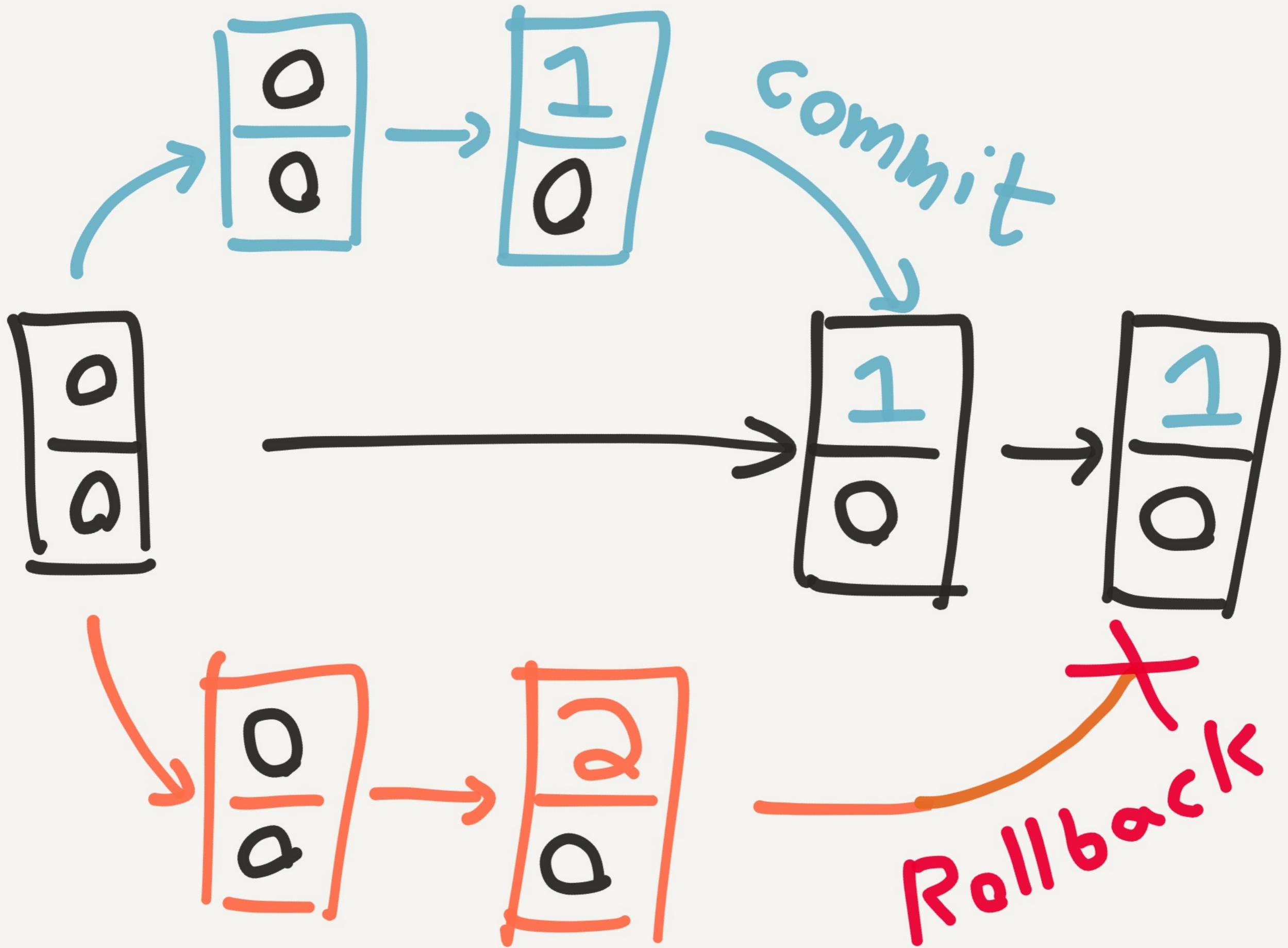
Commit



write



commit



“SI level occurs
between Repeatable
Read and Serializable”

(so, use 1SR, get SI)

serializability

1SR

snapshot isolation

SI

(wrong!)

repeatable read

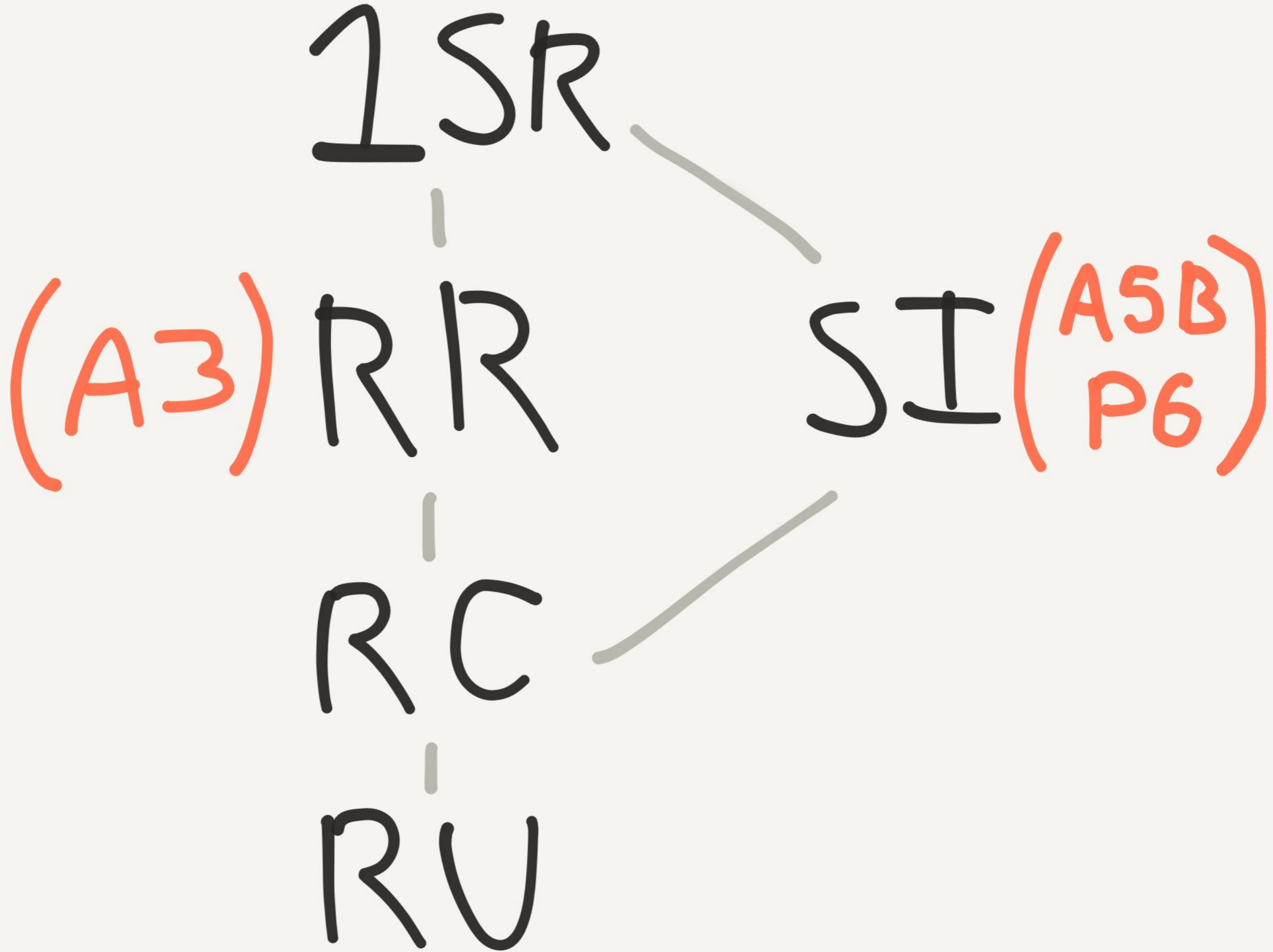
RR

read committed

RC

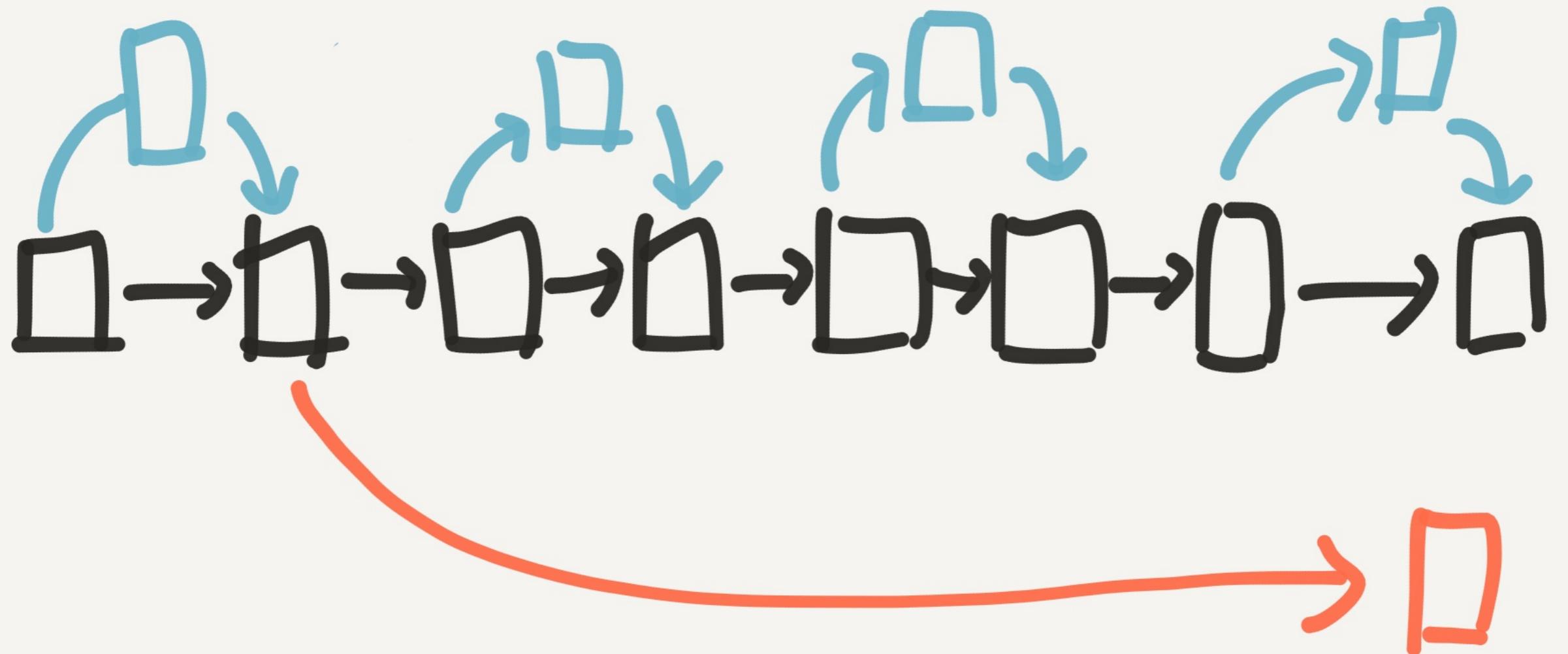
read uncommitted

RU



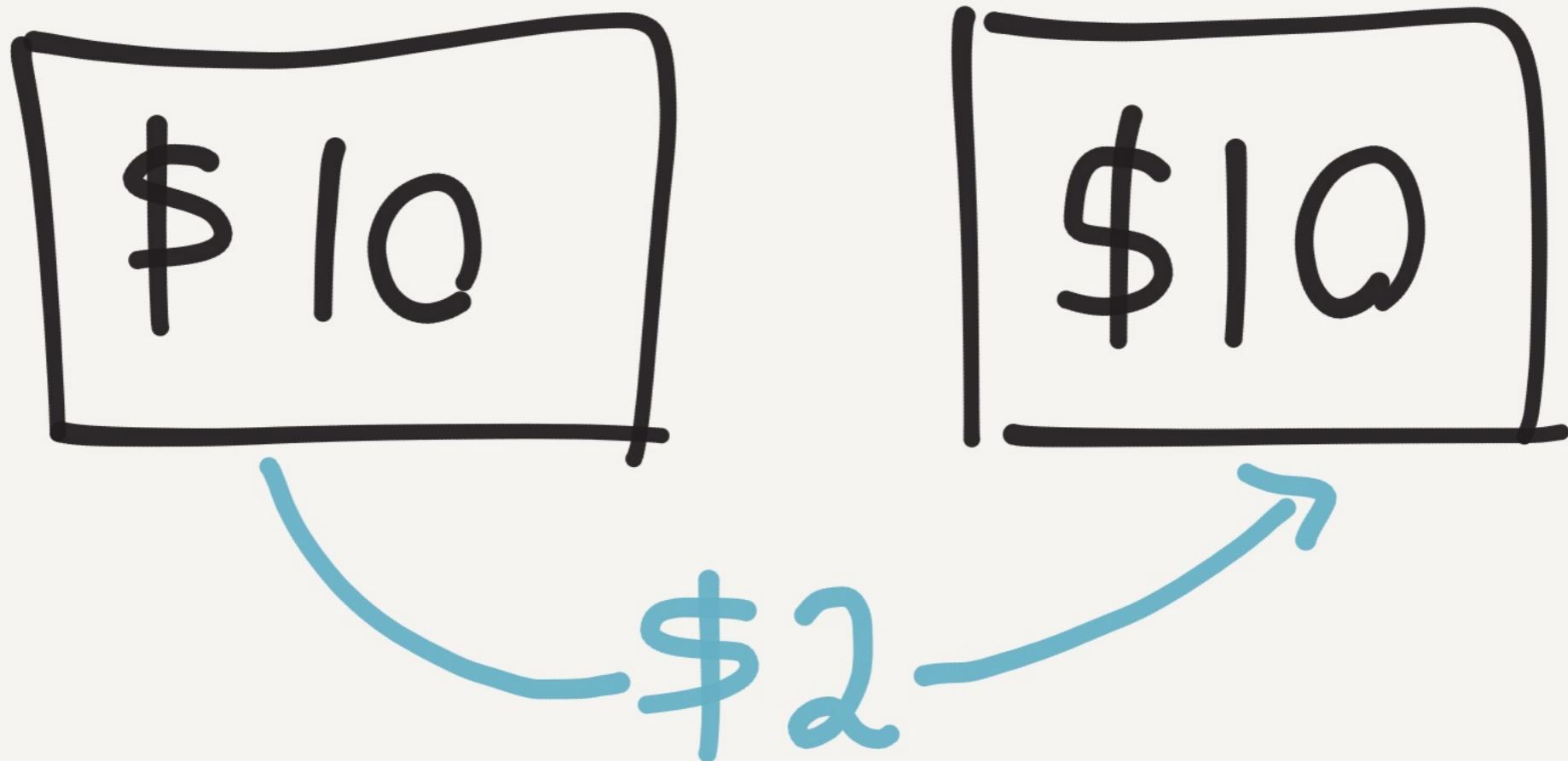
OK...So how
do we test it?

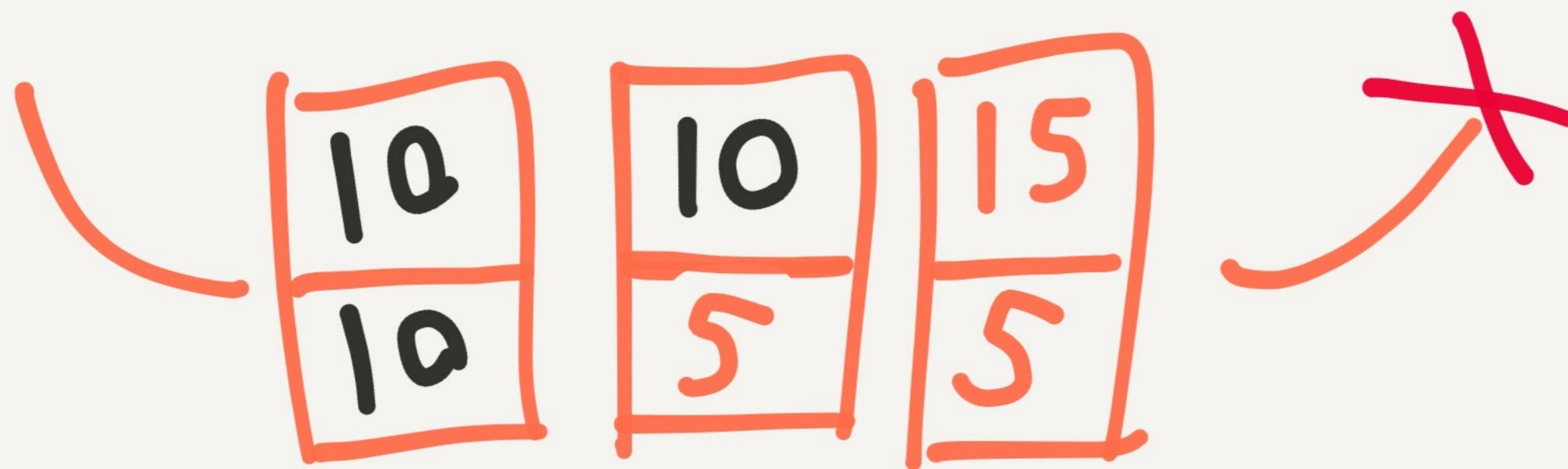
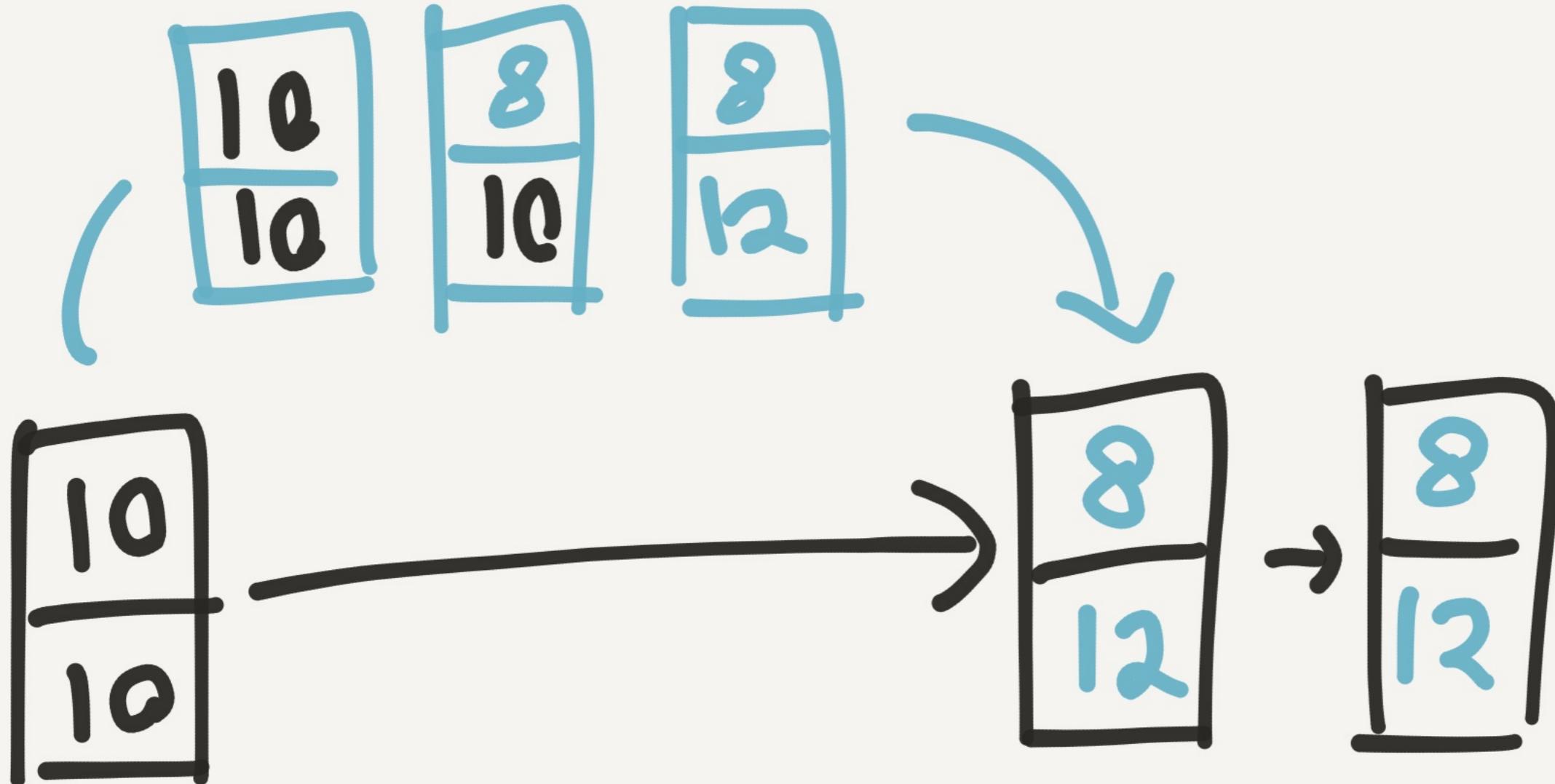
Can't use our
linearizability
checker...

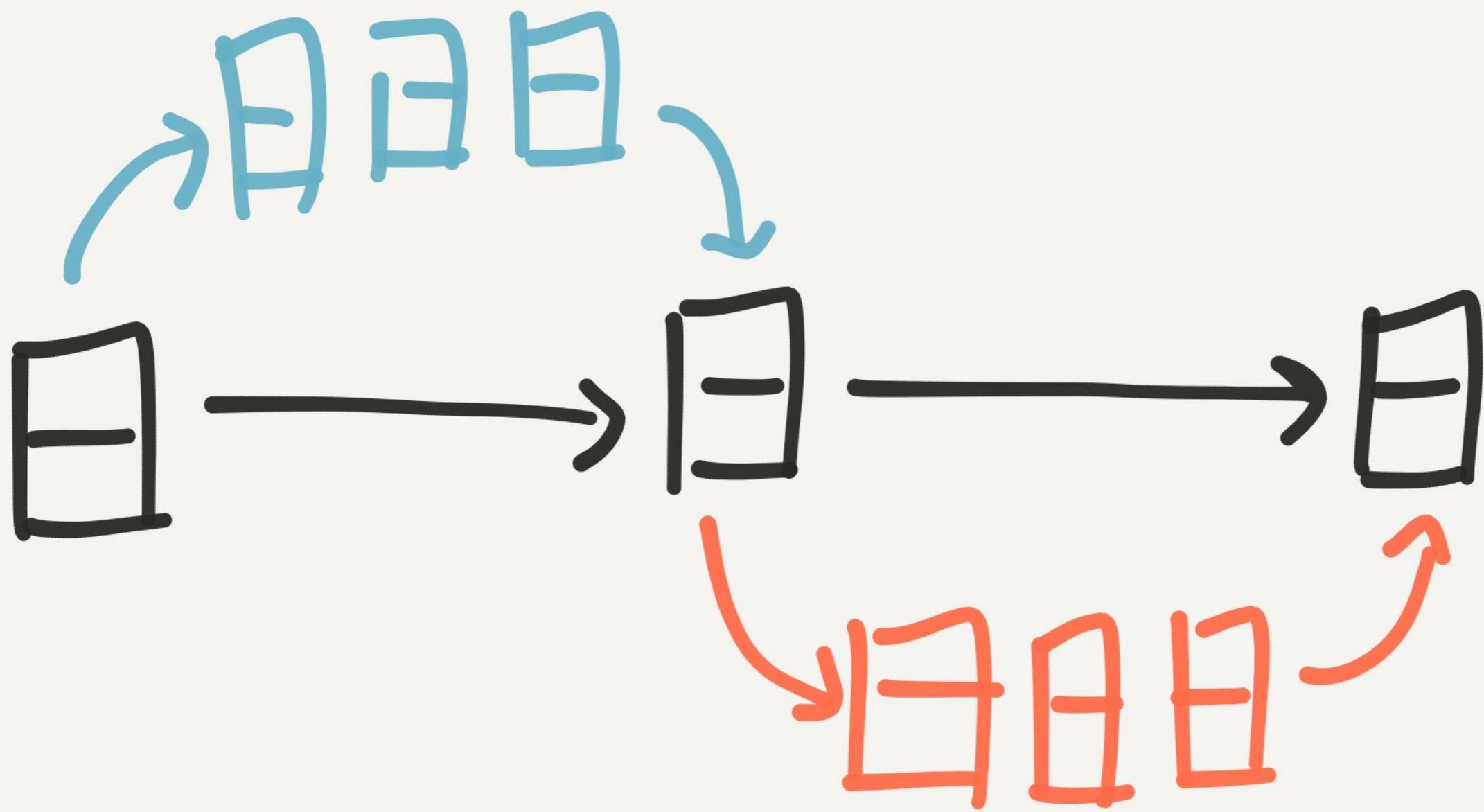


stale read

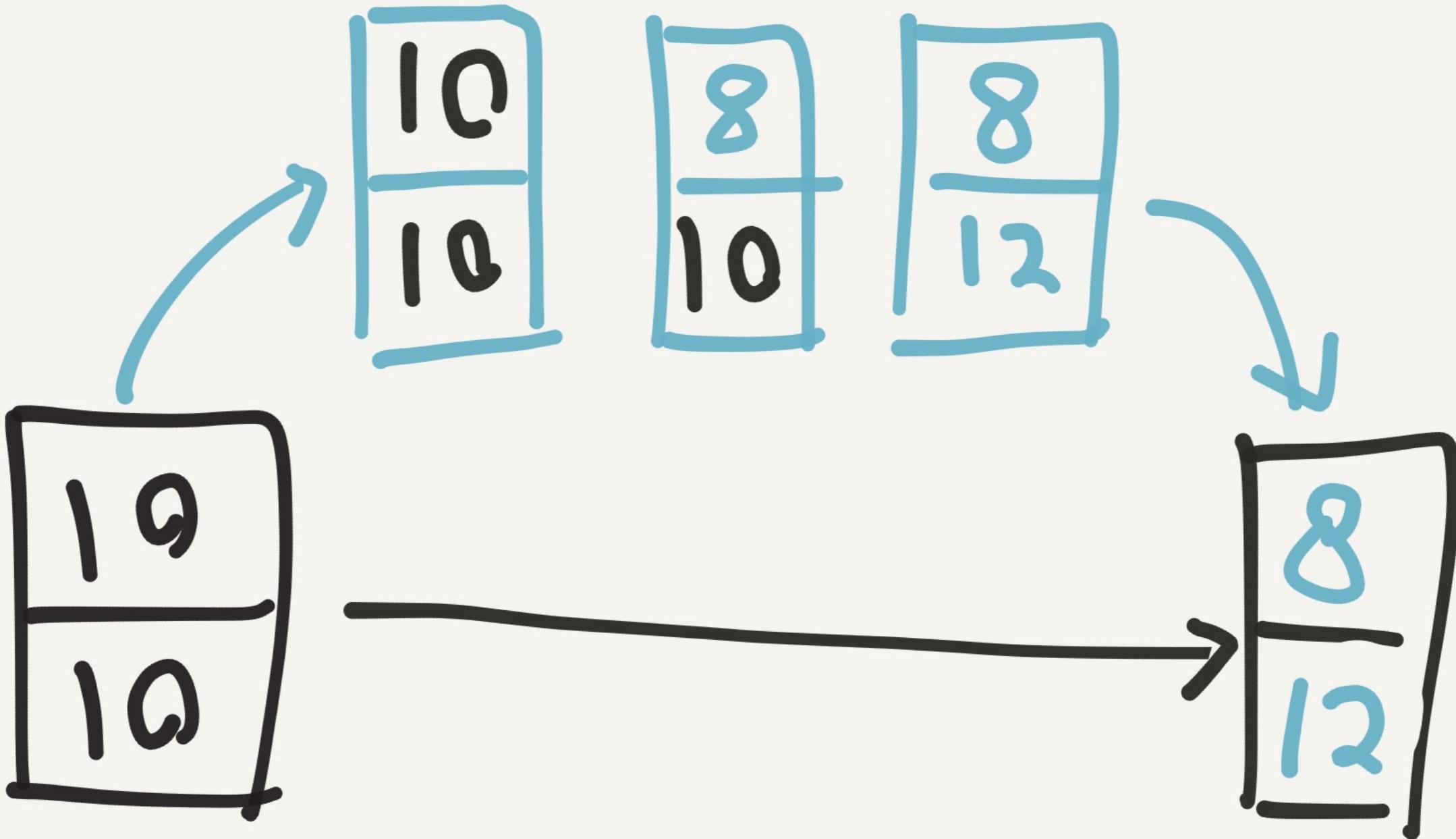
Consider 2 bank accounts...



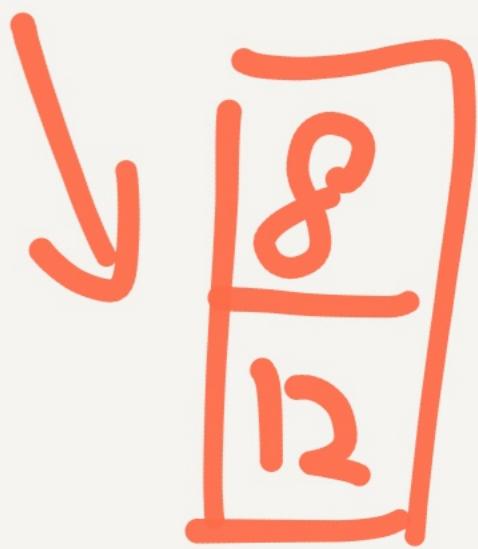




Transfers must serialize



consistent
reads



→ Sum of accs

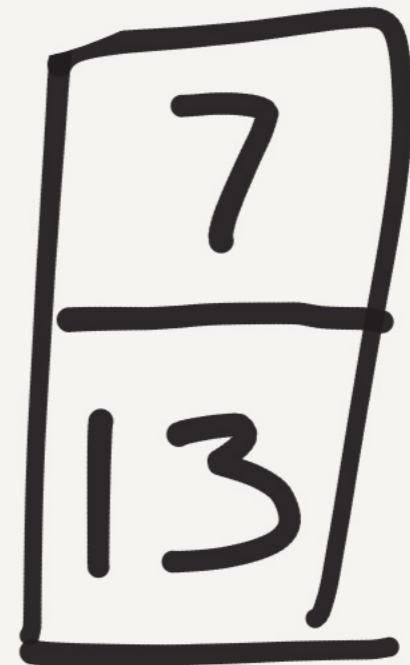
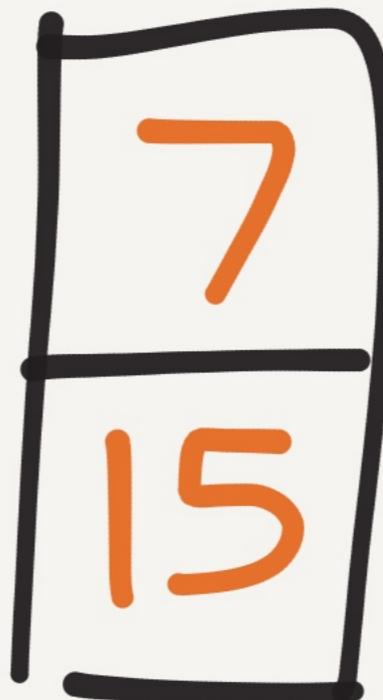
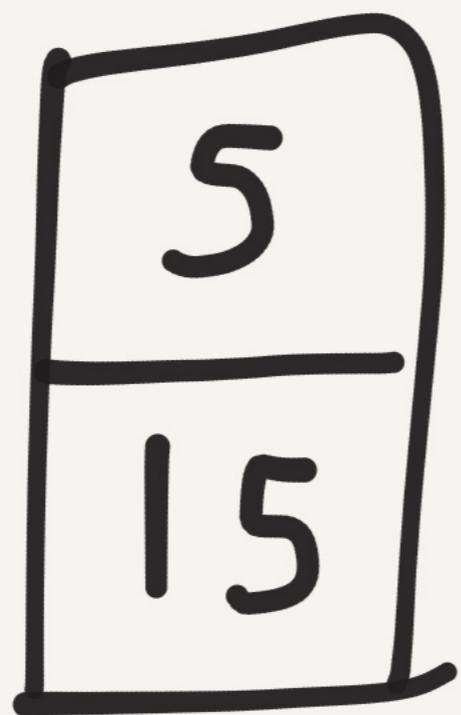
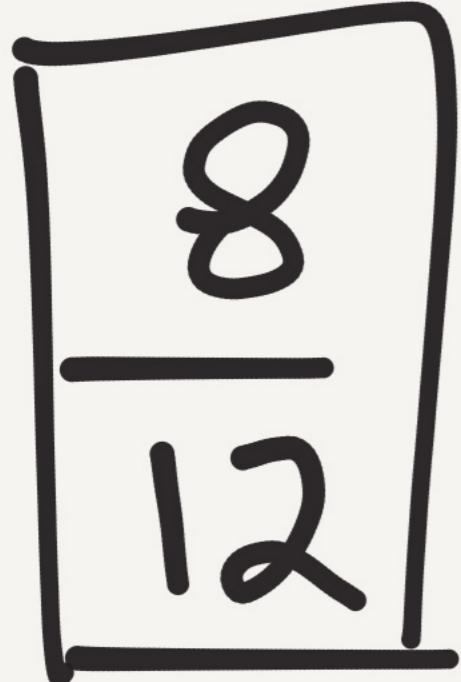
is always a

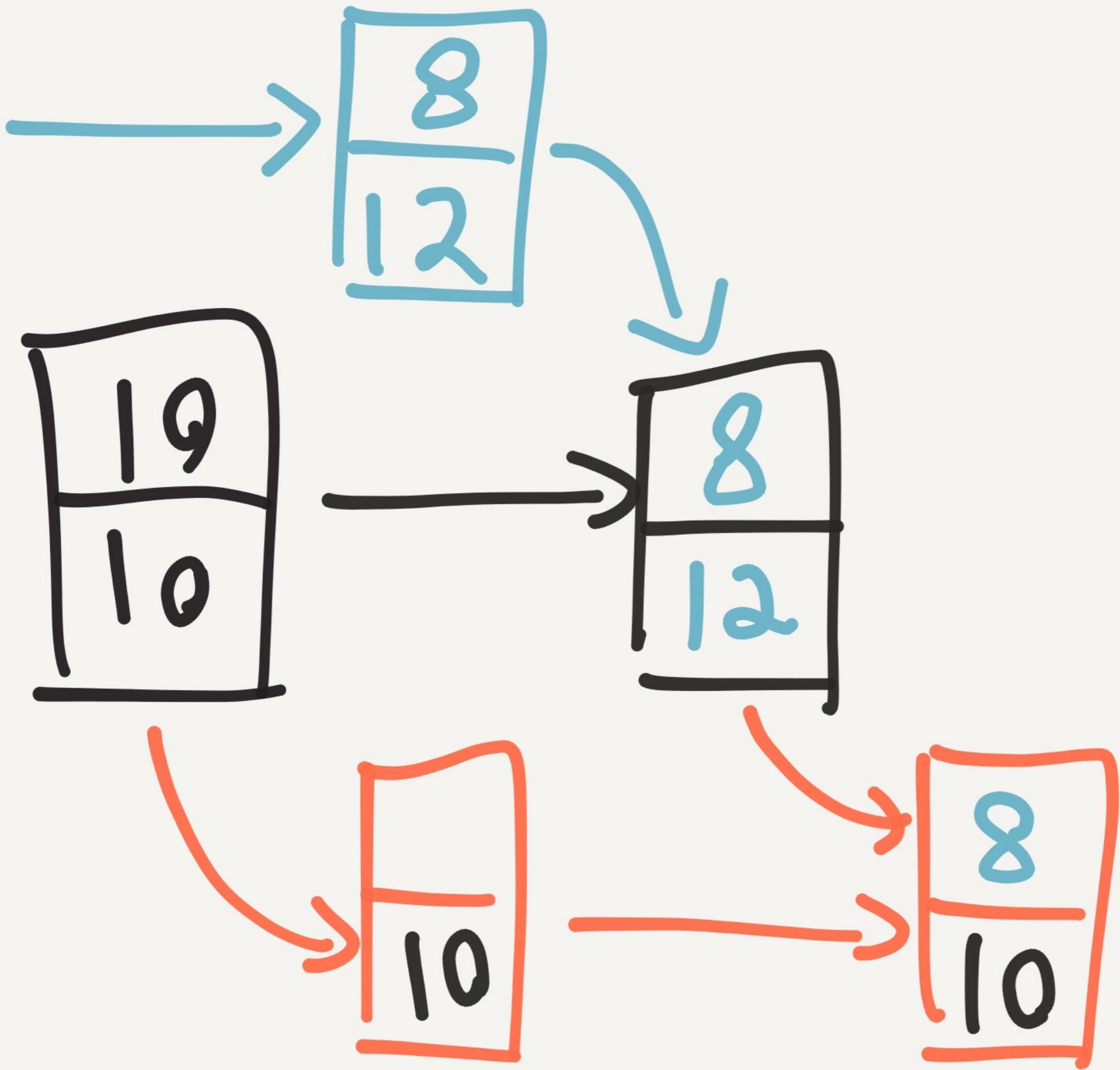
CONSTANT

ORTS

TIT?

Inconsistent Reads





A5A

READ
SKREW

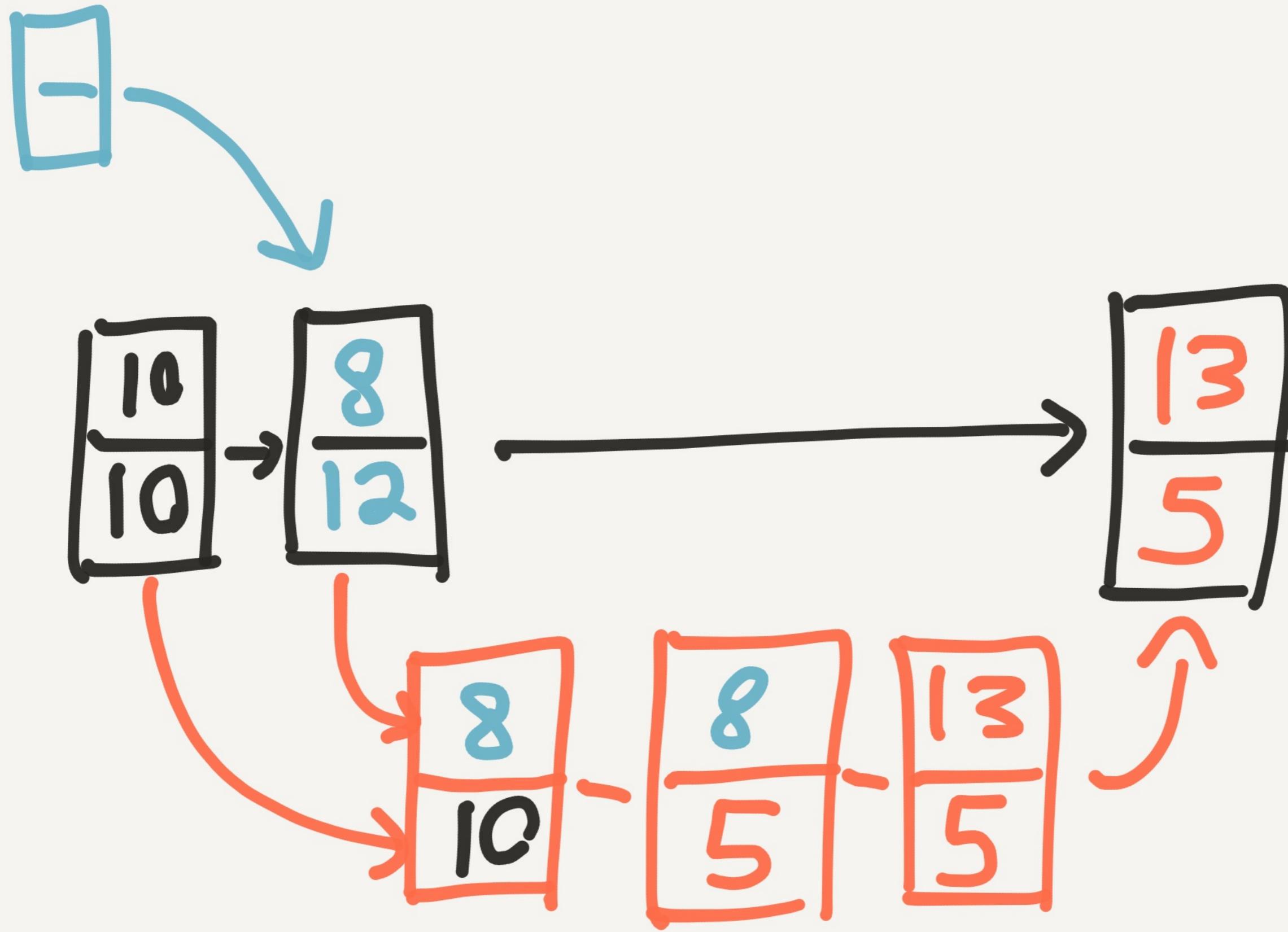
Read Skew

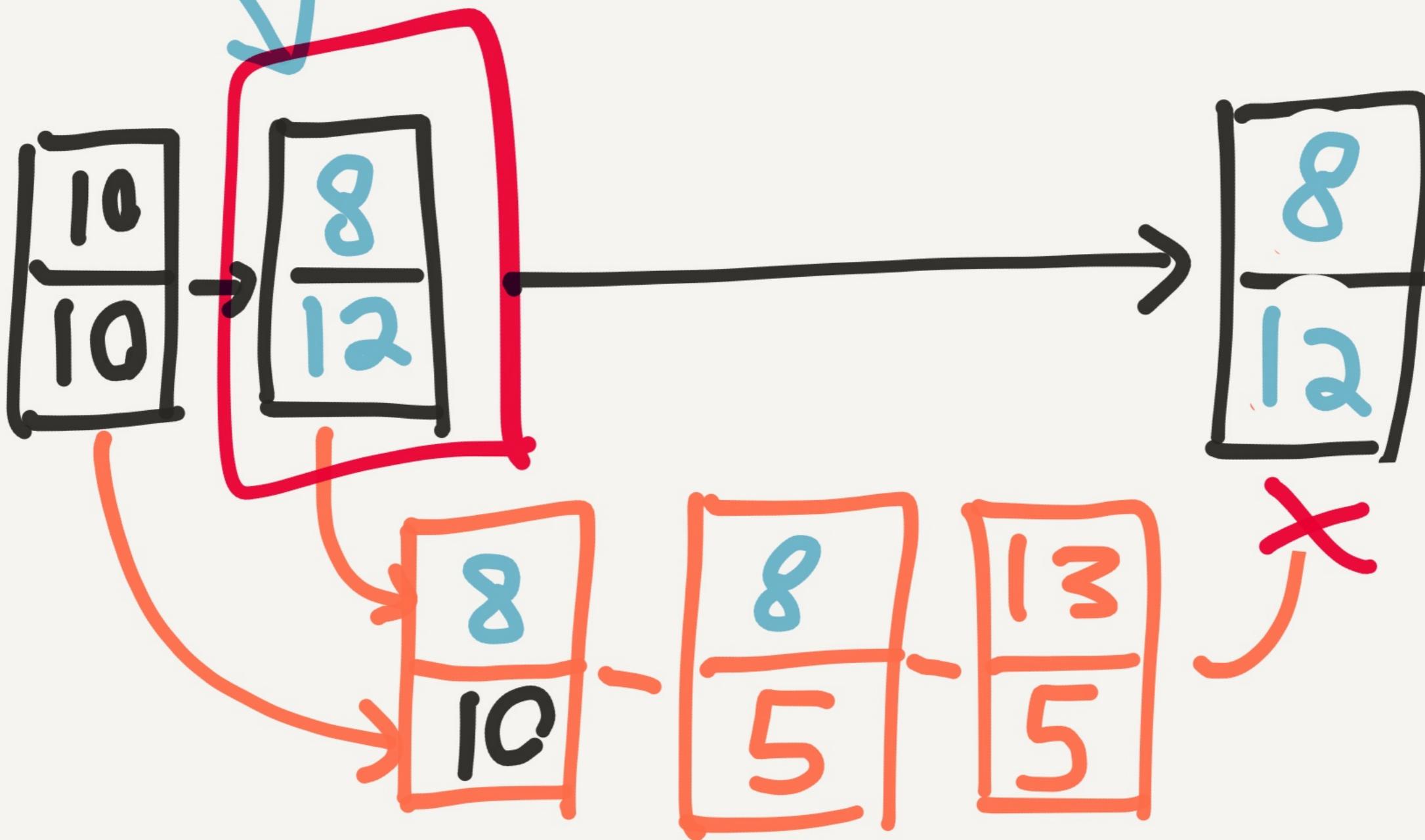
breaks SI

invariants

IT GETS

WORSE

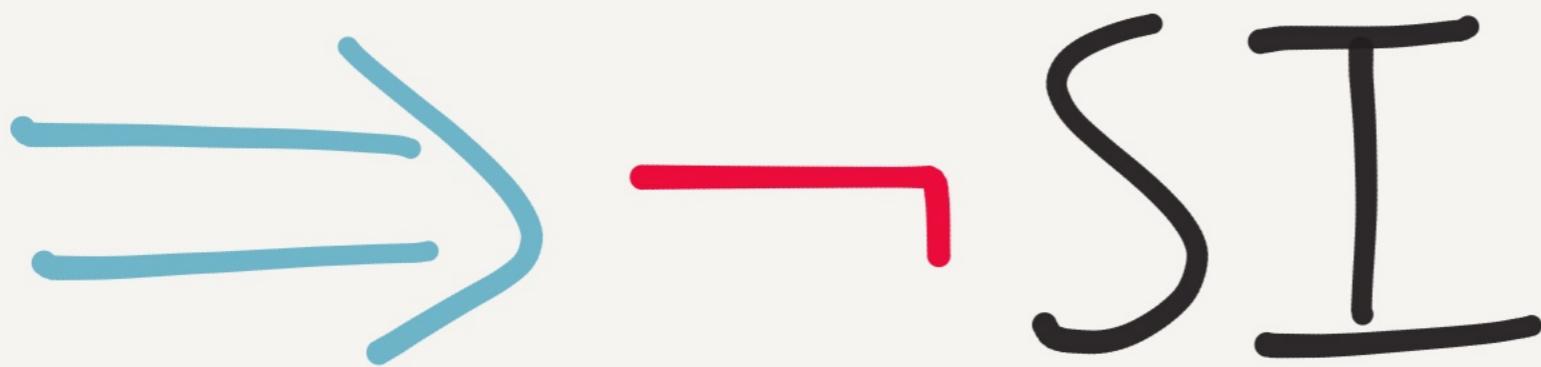




Galera engineers:

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

→ first-committer-wins



Docs are lying

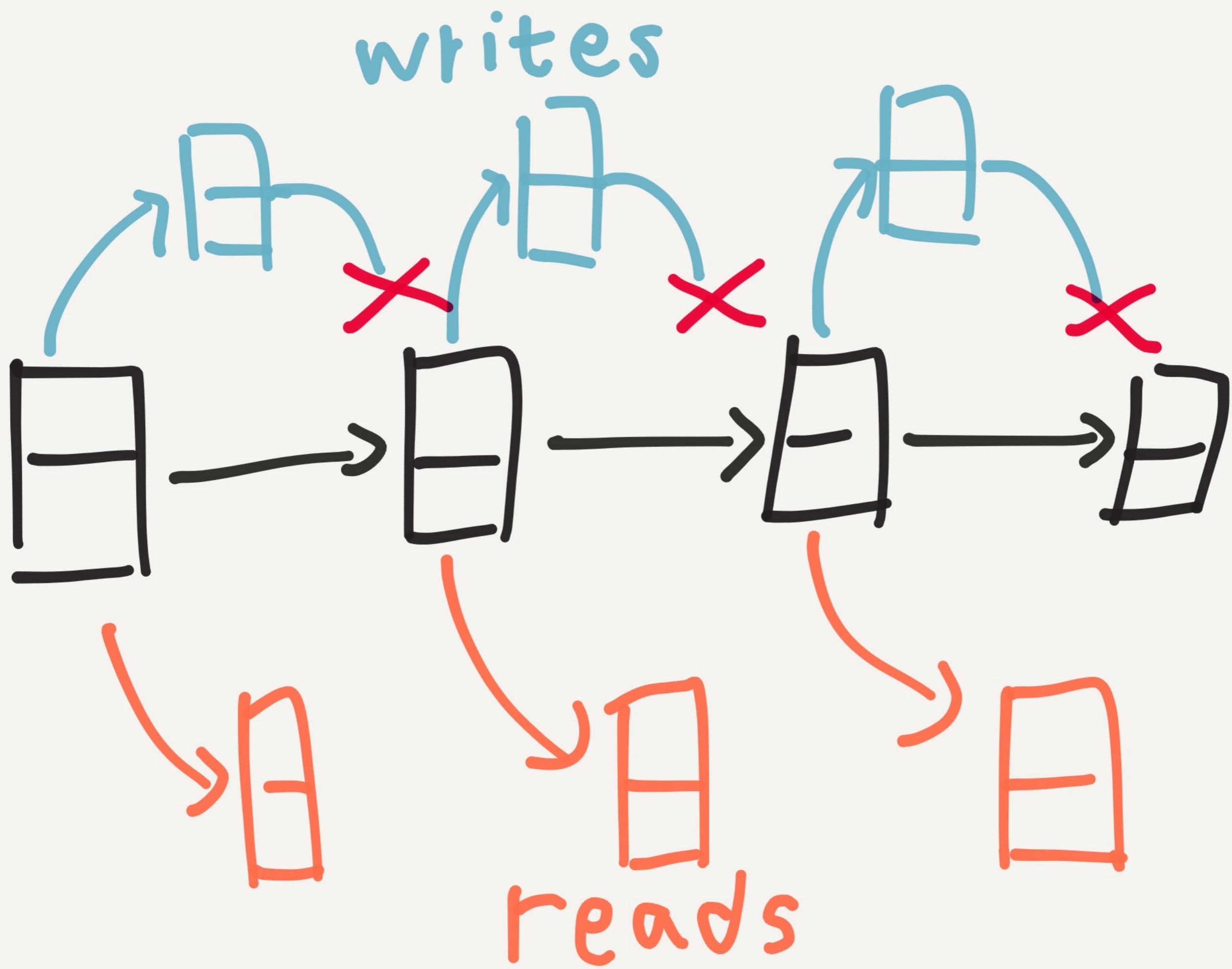
Not all is

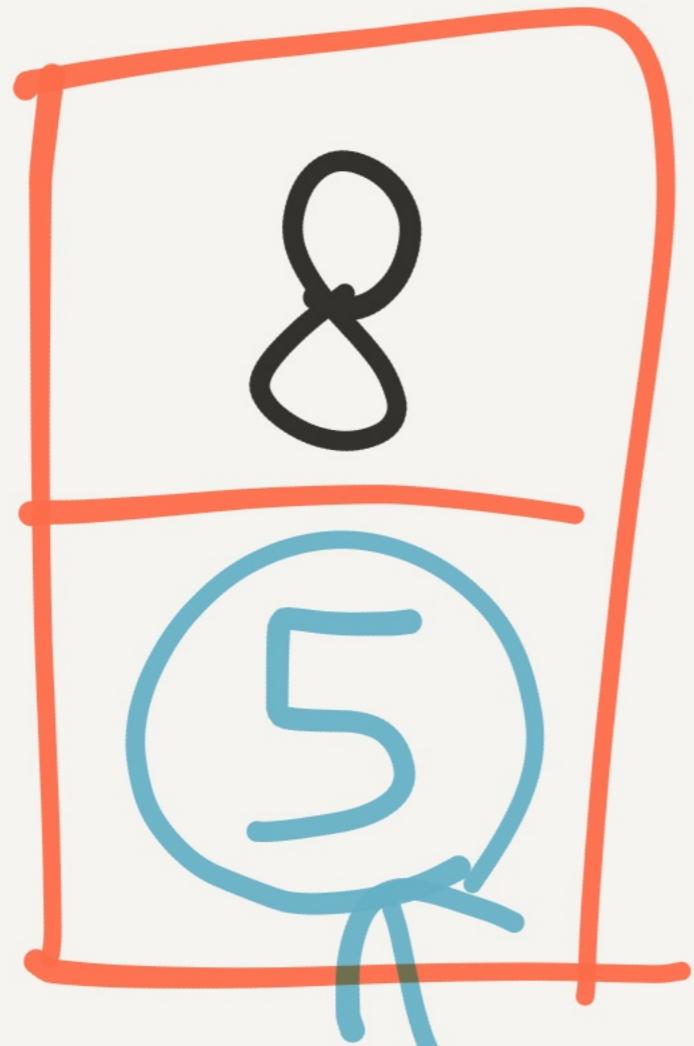
LOST

Galera does

not allow

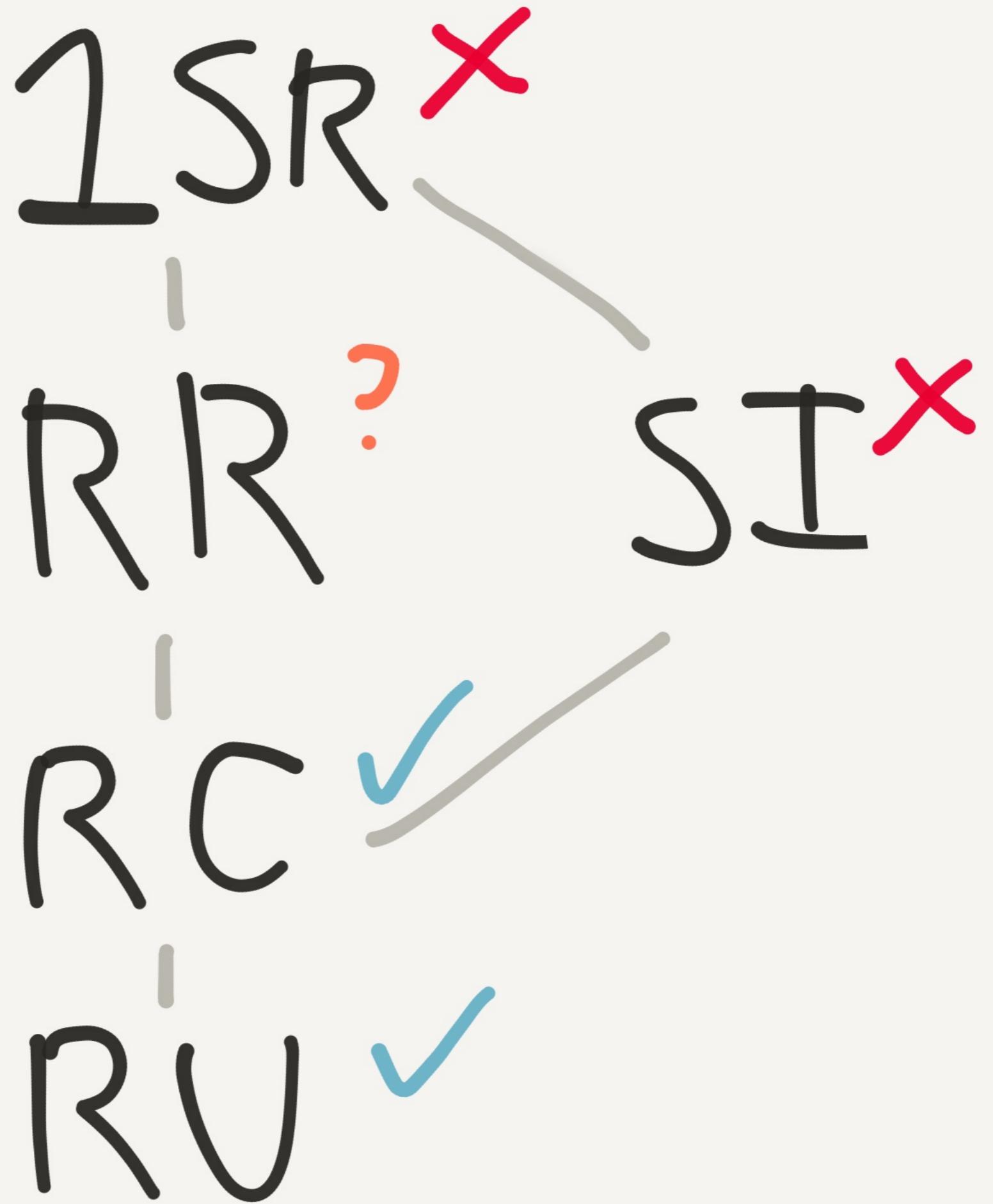
Dirty Read





(we never)
see this

from a aborted
transaction



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

RC is a pretty
weak property.

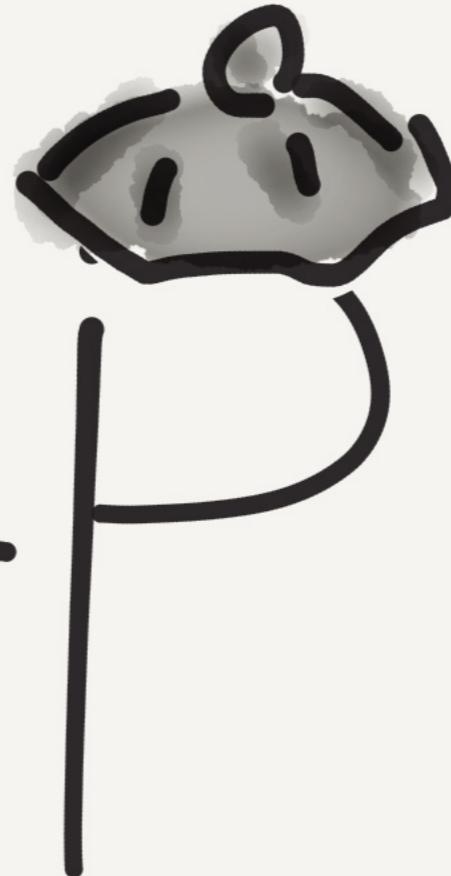
Allows those
inconsistent reads!

Galera might

support SI

someday....

Recap



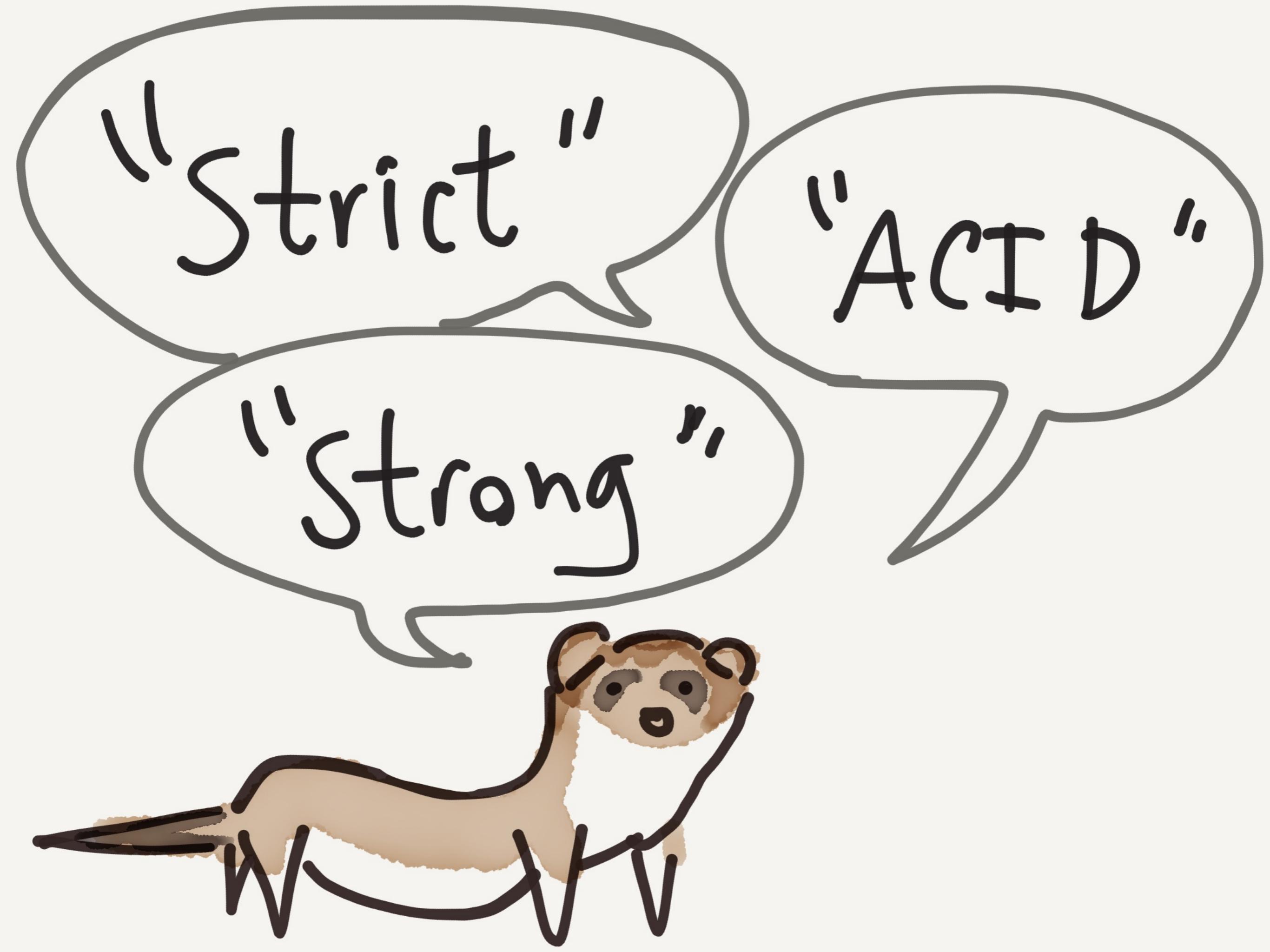
Read the docs!



Then test it

— for —

YOURSELF



"Strict"

"ACID"

"Strong"

Be Formal

Be Specific.

*Figure out the
invariants
your system needs*

How much



can you tolerate?

Consider
your
failure modes

Process Crash

#Kill -9 1234

Node failure

- AWS terminate
- Physical power switch

Clock Skew

date 1028000

fake time ...

GC/TQ Pause

killall -s STOP foo

killall -s CONT foo

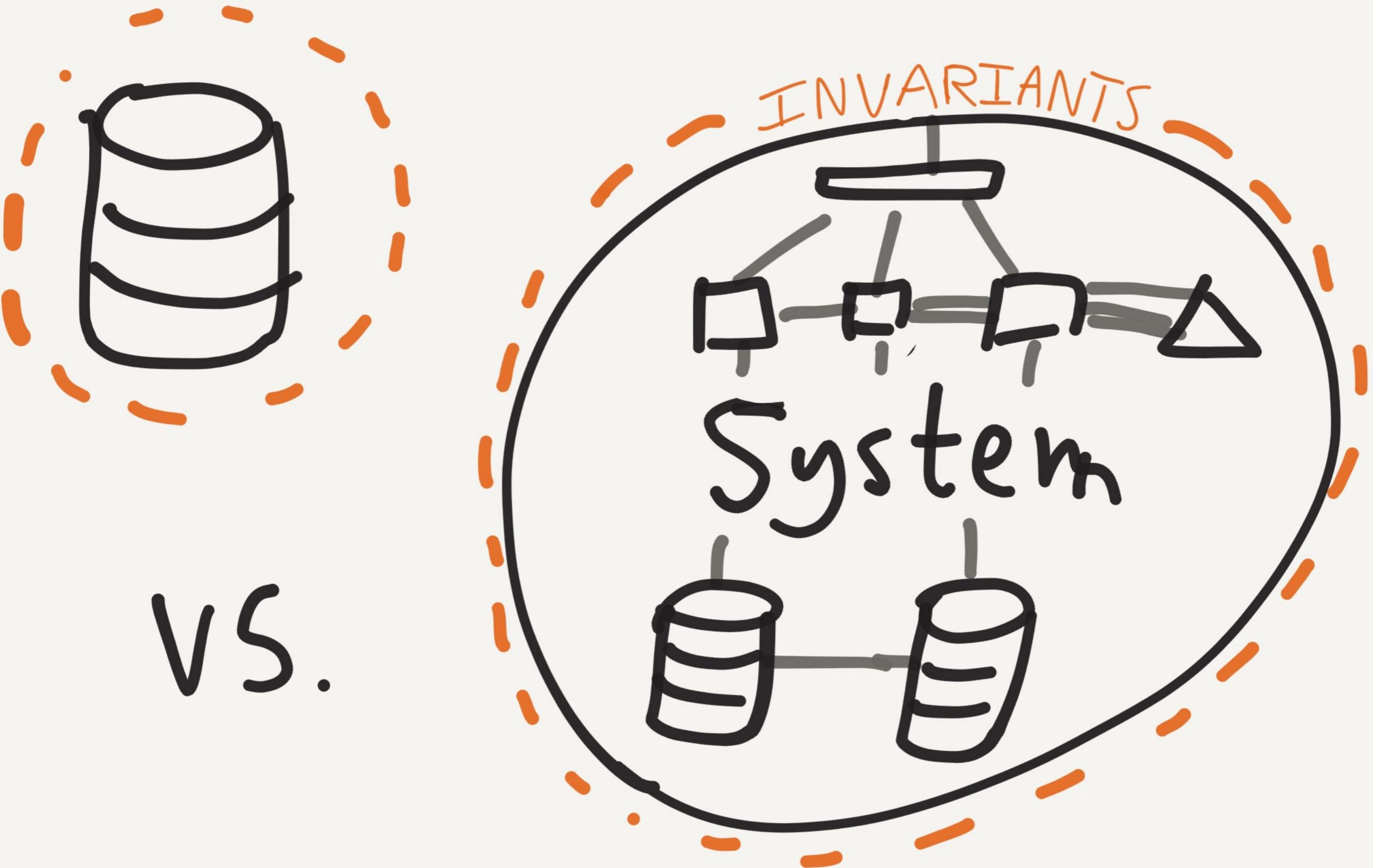
Network Partition

```
# iptables -j DROP  
# tc qdisc ... delay...  
      drop...
```

Test your

Systems

end-to-end



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

Thanks

Sjaakala

Siddarth Chandrasekaran

Brendan Taylor

Cosmin Nicolaeescu

Thanks

Brenden Matthews

Timothy Chen

Aarch Bell

Kyle Conroy

Thanks!

Codership

Mesos (sphere)?

Stripe

<http://github.com/aphyr/jepsen>

Questions?

<http://aphyr.com/tags/jepsen>