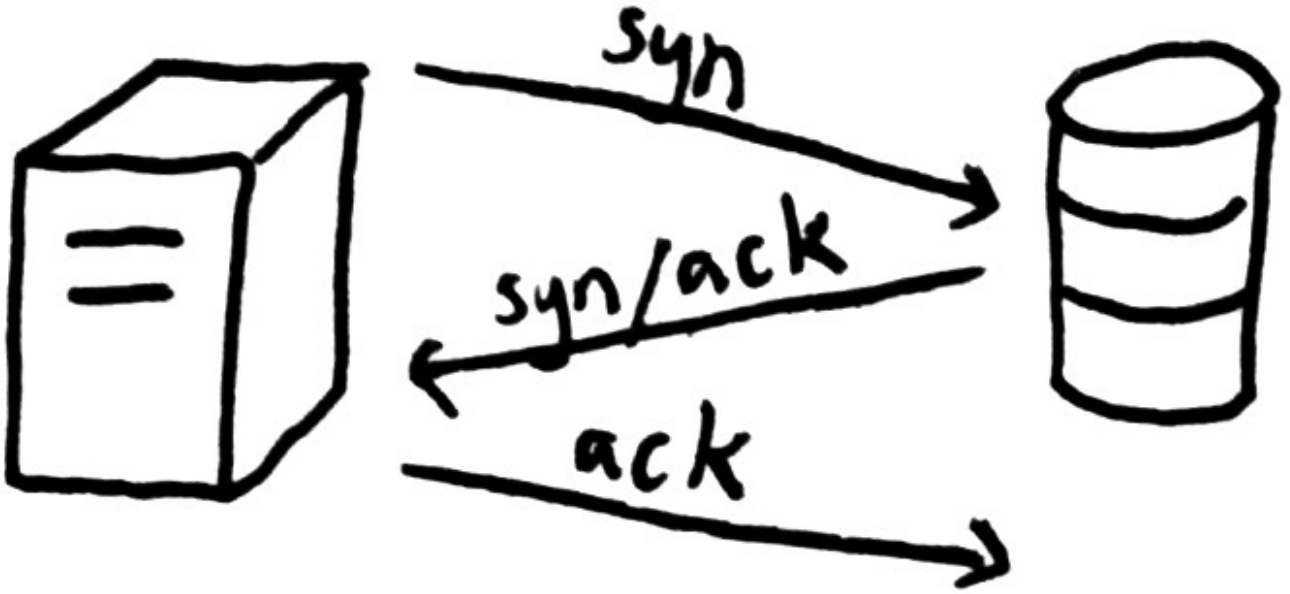
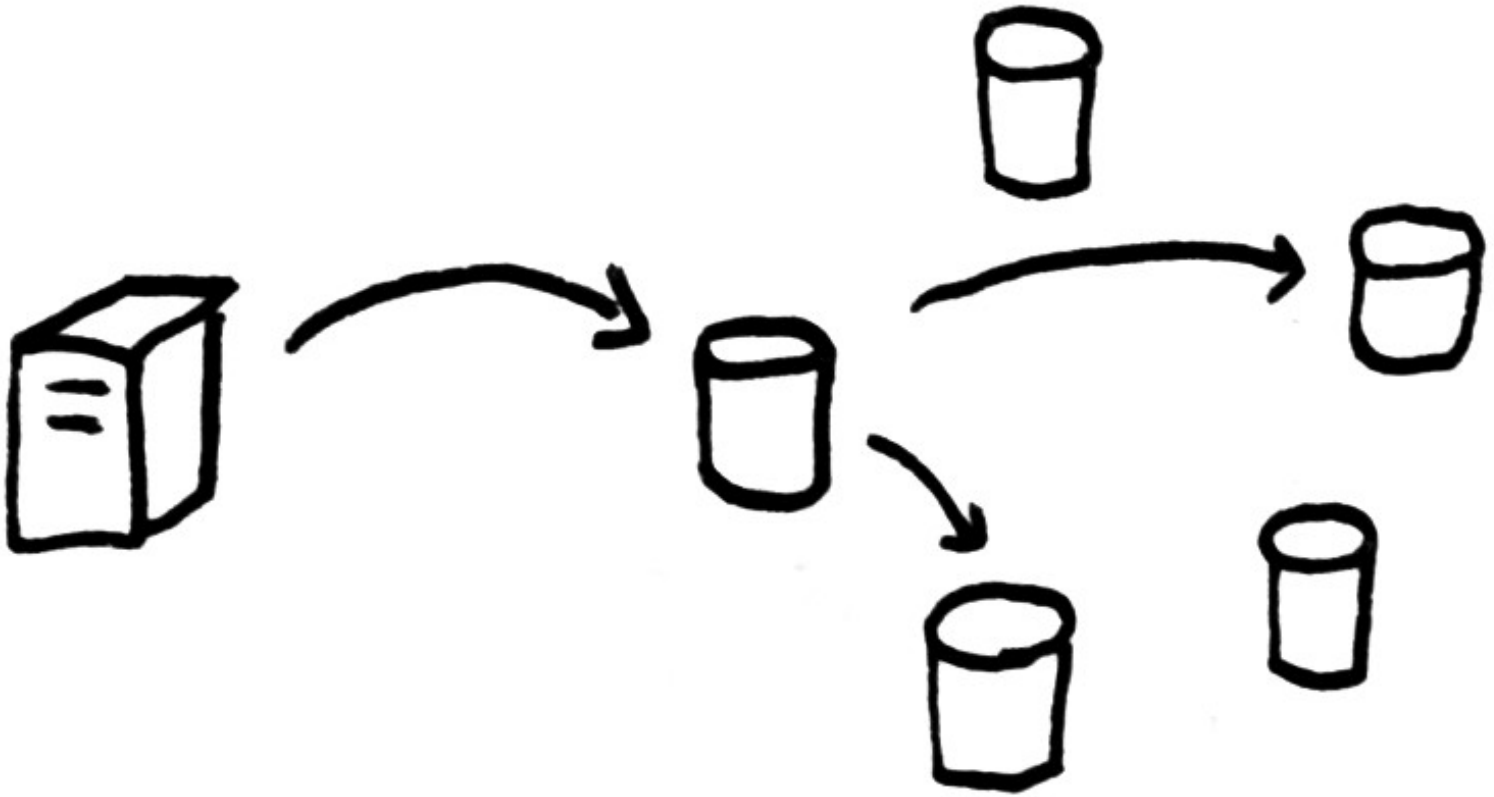
HEY,

I just met you

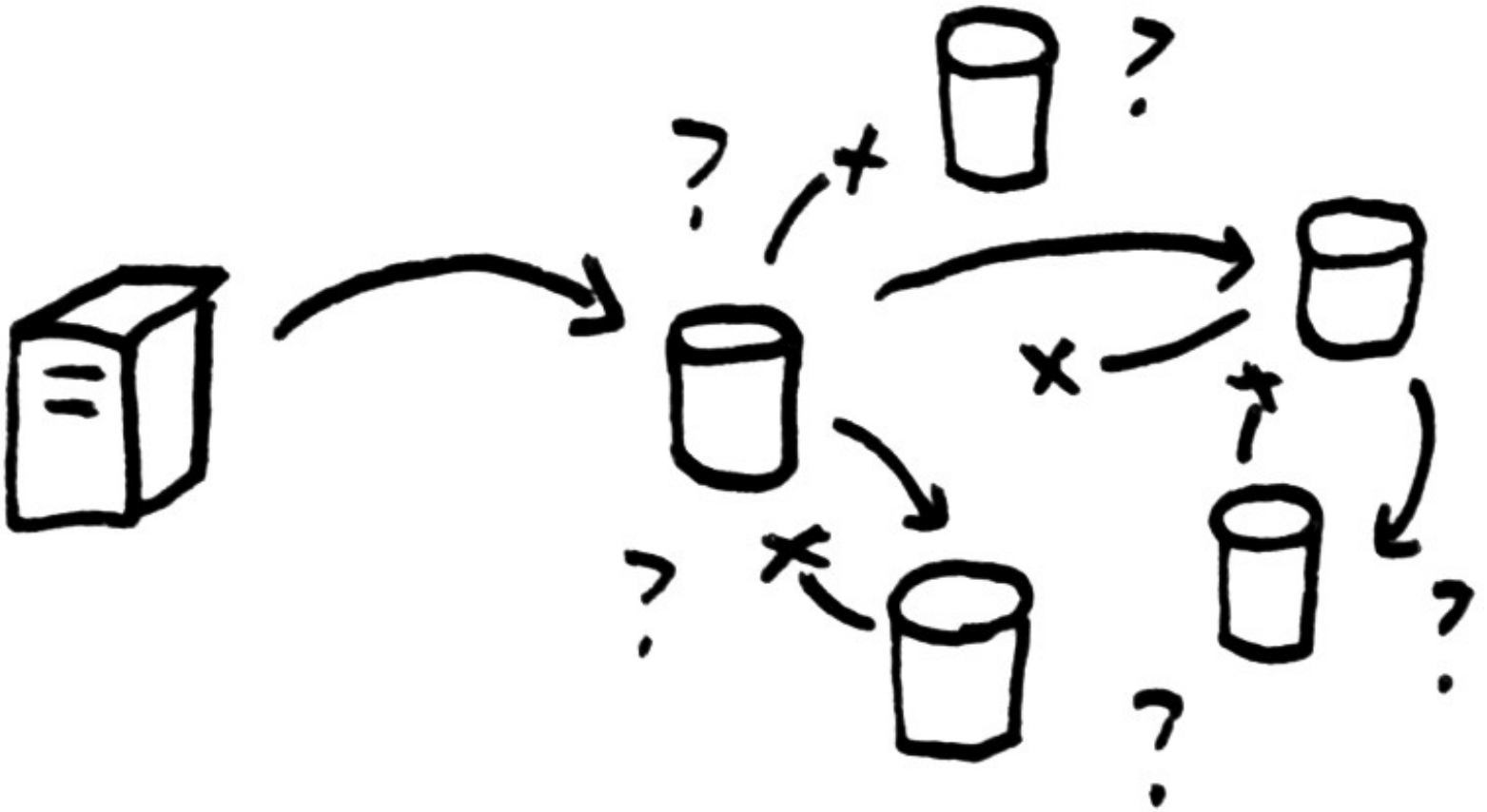


Our network's flaky

But here's my data



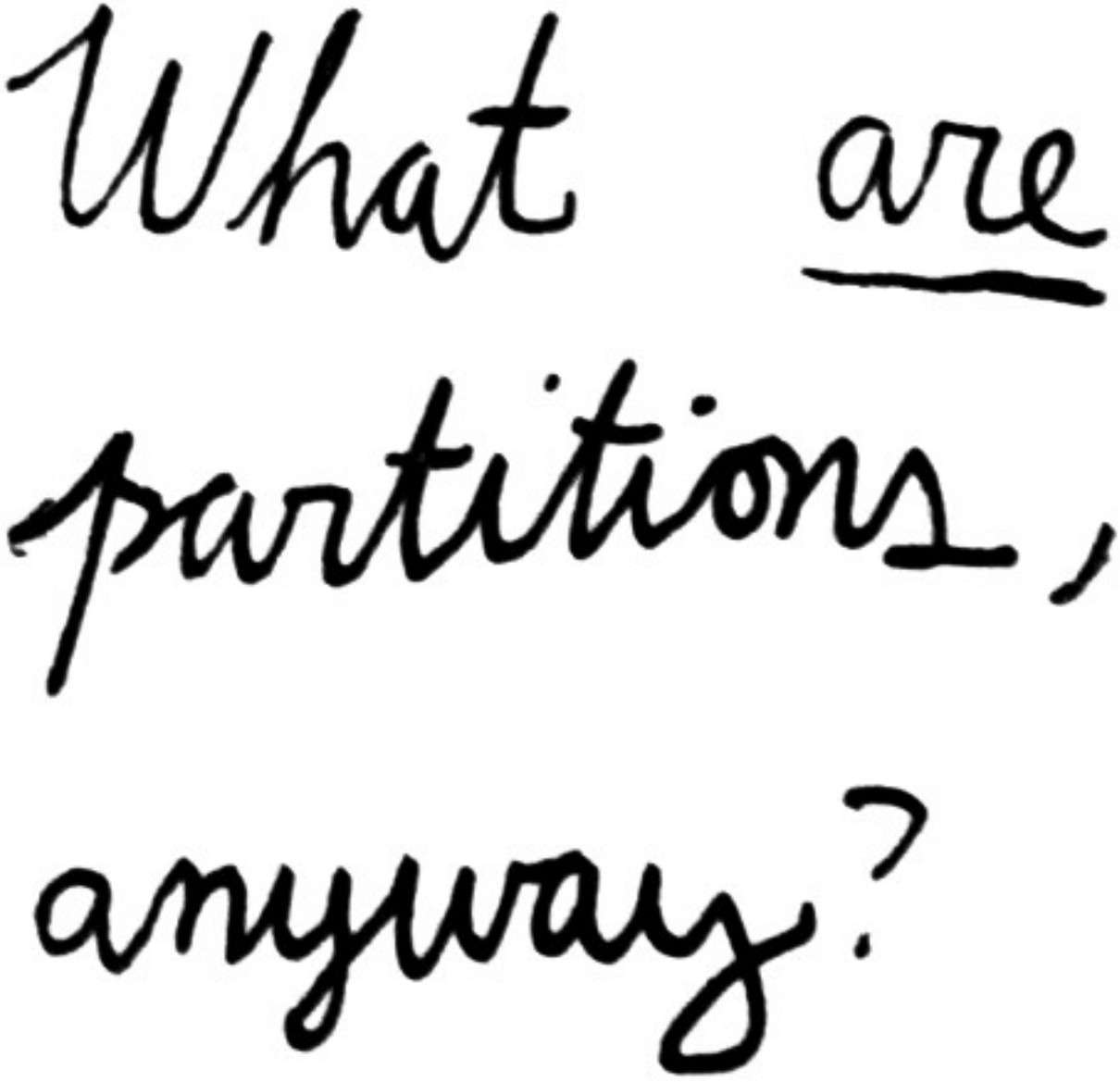
So store it maybe?



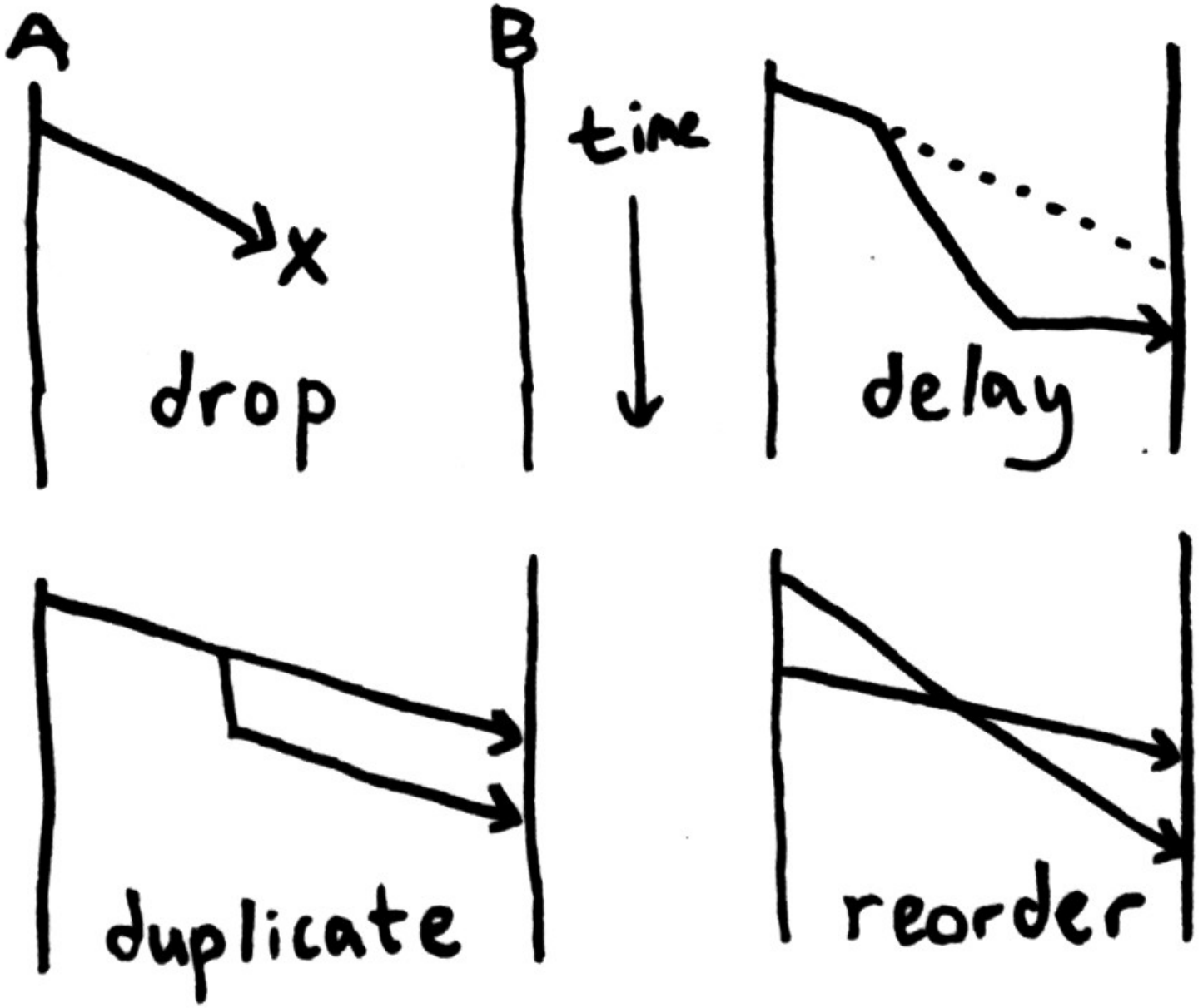
@aphyr

(Kyle Kingsbury)


PARTITIONS
★for★
EVERYONE!


What are
partitions,
anyway?

Formally:

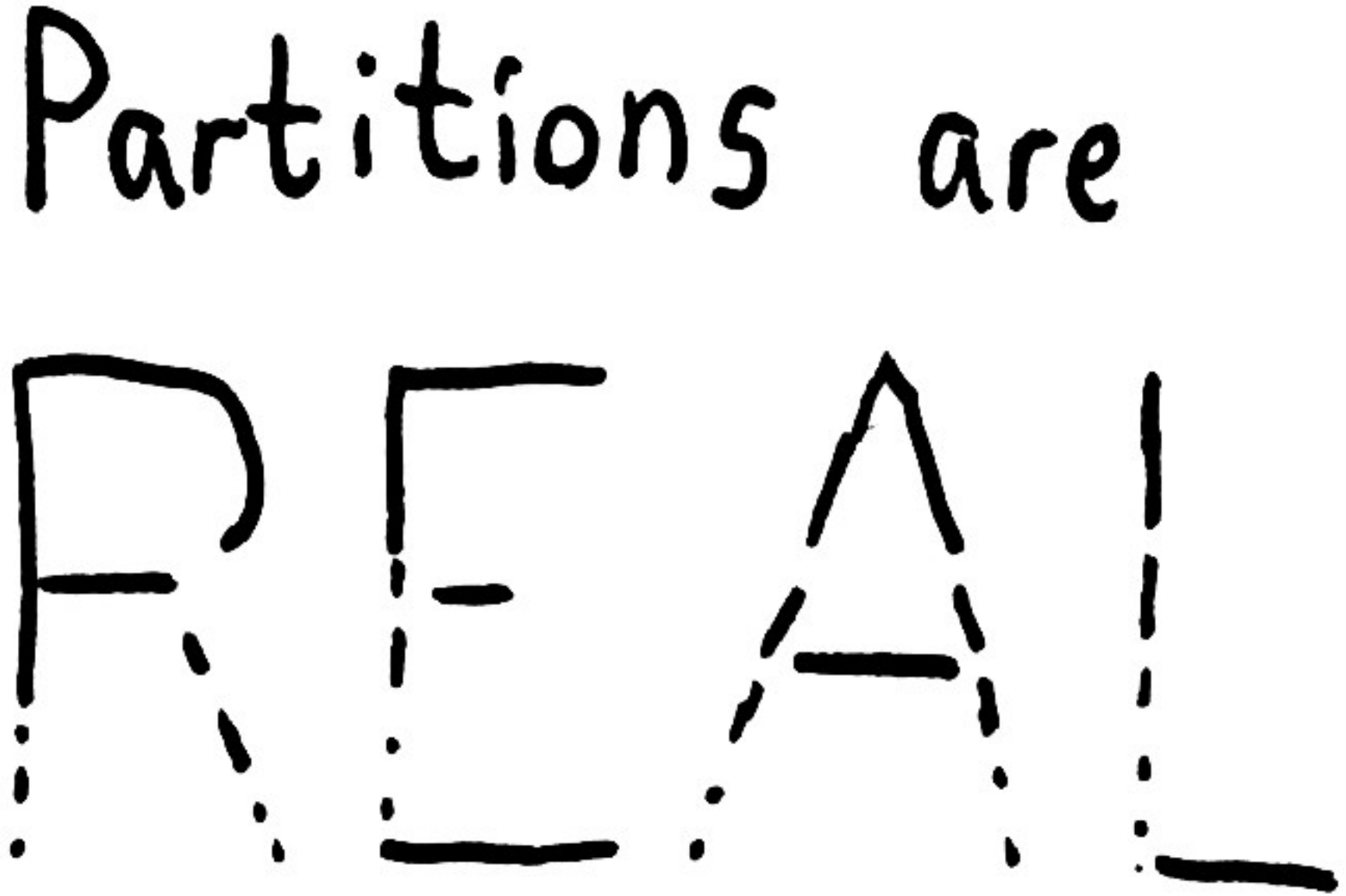


IP networks do all four.

TCP means no dupes, reorders

- Unless you retry!

Delays are indistinguishible from drops, after timeout


Partitions are
REAL

Microsoft: 40.8 link failures/day

- 5 minutes — one week
- Redundancy improves packet loss by only 43%

Google: in 1 cluster...

- 5 racks see 50% pkt loss
- 8 net maint/yr, 30 min loss in 4
- 3 router failures/yr

Yahoo Sherpa

Amazon Dynamo

Google Chubby

Causes

- GC pauses
- Network maint
- Segfaults & crashes
- Faulty NICs
- Bridge loops

... continved

- Spanning tree
- Hosted networks
- The cloud
- WAN links & Backhoes

Cloudflare

Pagerduty

Fog Creek

Netflix

Github

City Cloud

Searchbox.io

AWS

Freistel IT

Twilio

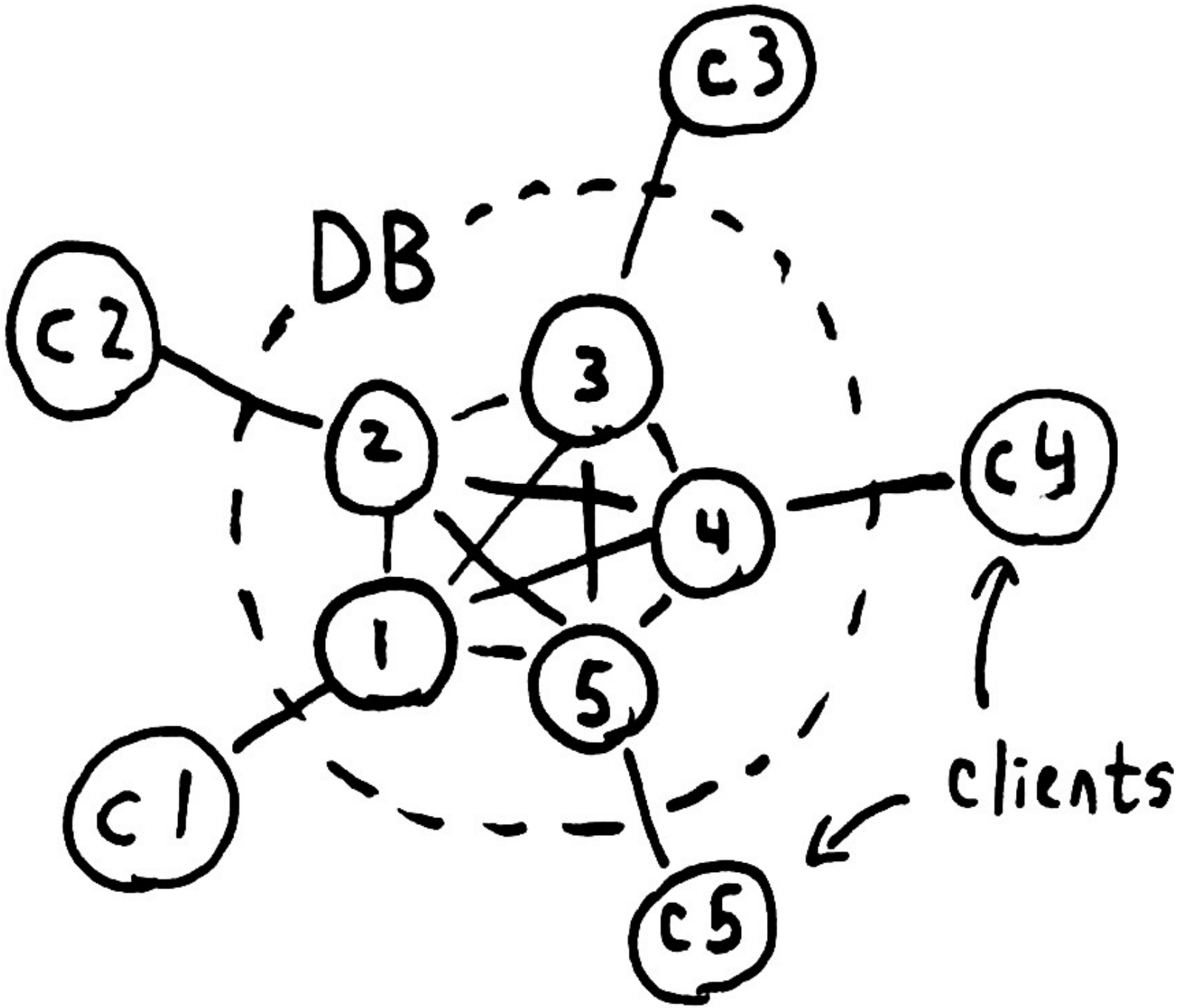
PLANET

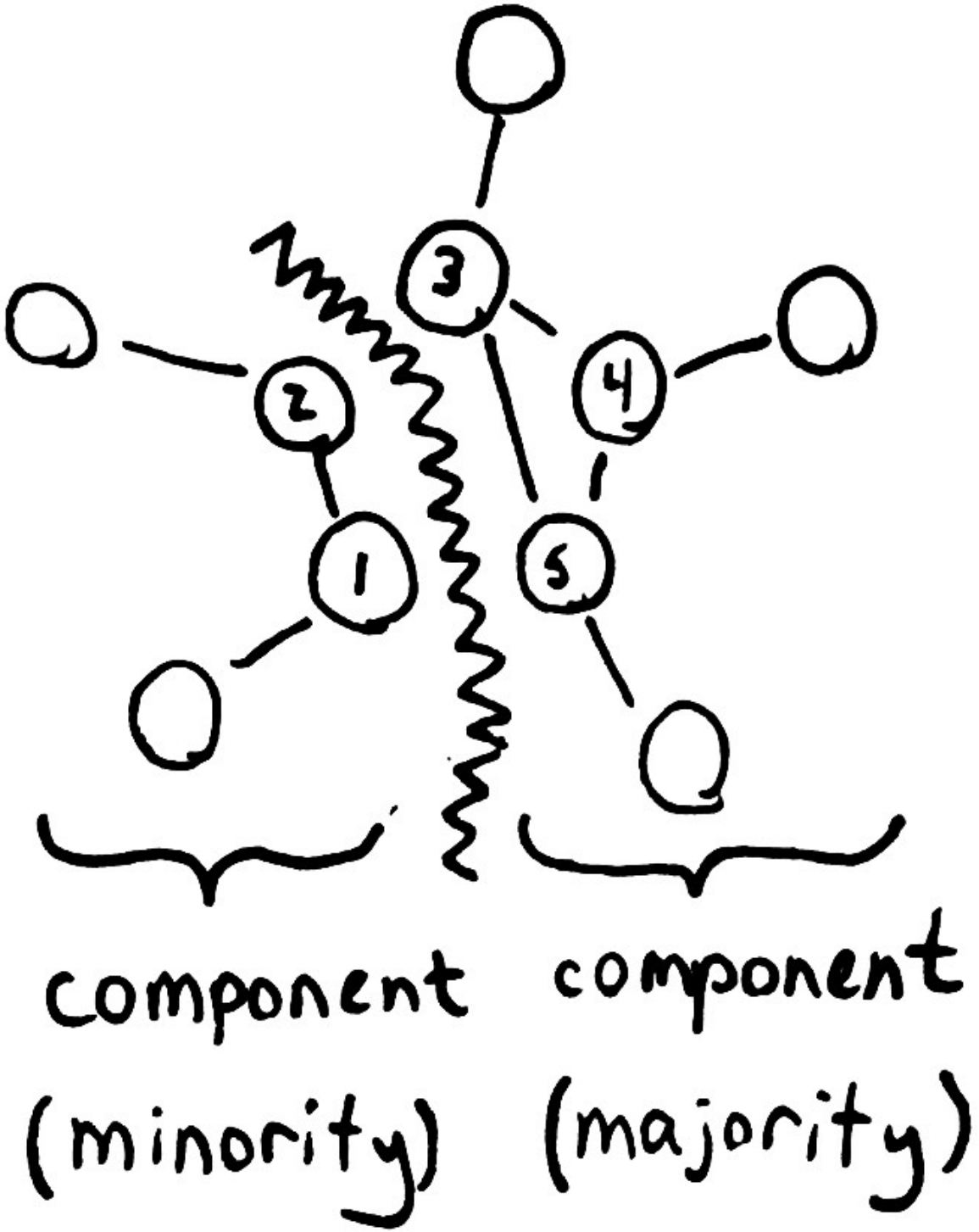
TERAH

(you are not here)

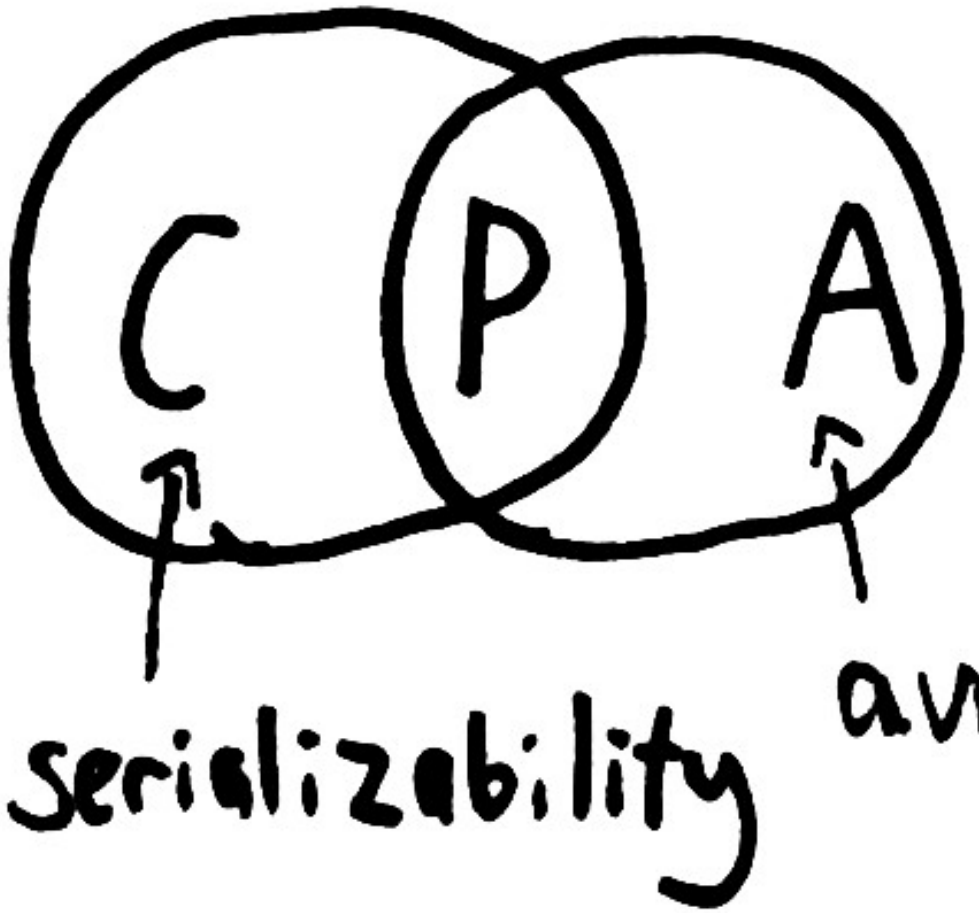
Distributed Systems tend to fail part... rtially

Jepsen is an exploration of those failure modes


c3
DB
c2
3
2
c4
4
1
5
clients
c1
c5


3
2
4
1
5
component
(minority)
component
(majority)

In broad terms



Both AP and CP systems can remain available in this case

needs a majority or primary node - forces re-election

doesn't care about partitions — business as normal

Single-server Postgres

- CP
- SPOF
- consistent
- clients may not agree

Redis (w/failover)

- Split-brain
- Drops all data on one component
- Not even close to consistent.

MongoDB

- All consistency levels led to lost data
- Weak defaults
- Majority was a bug, fixed

Rick

- LWW means dropping writes
- Even with PR/PW = ALL
- Even with lock service
- Use CRDTs

That was Then,

This is 

MEOW

That was Then,

This is 

now


Zookeeper
NuoDB
Kafka
Cassandra


Cassandra!

Dynamo system

- Partitioned hash ring
- Quorums & handoff
- LWW only, not vclocks

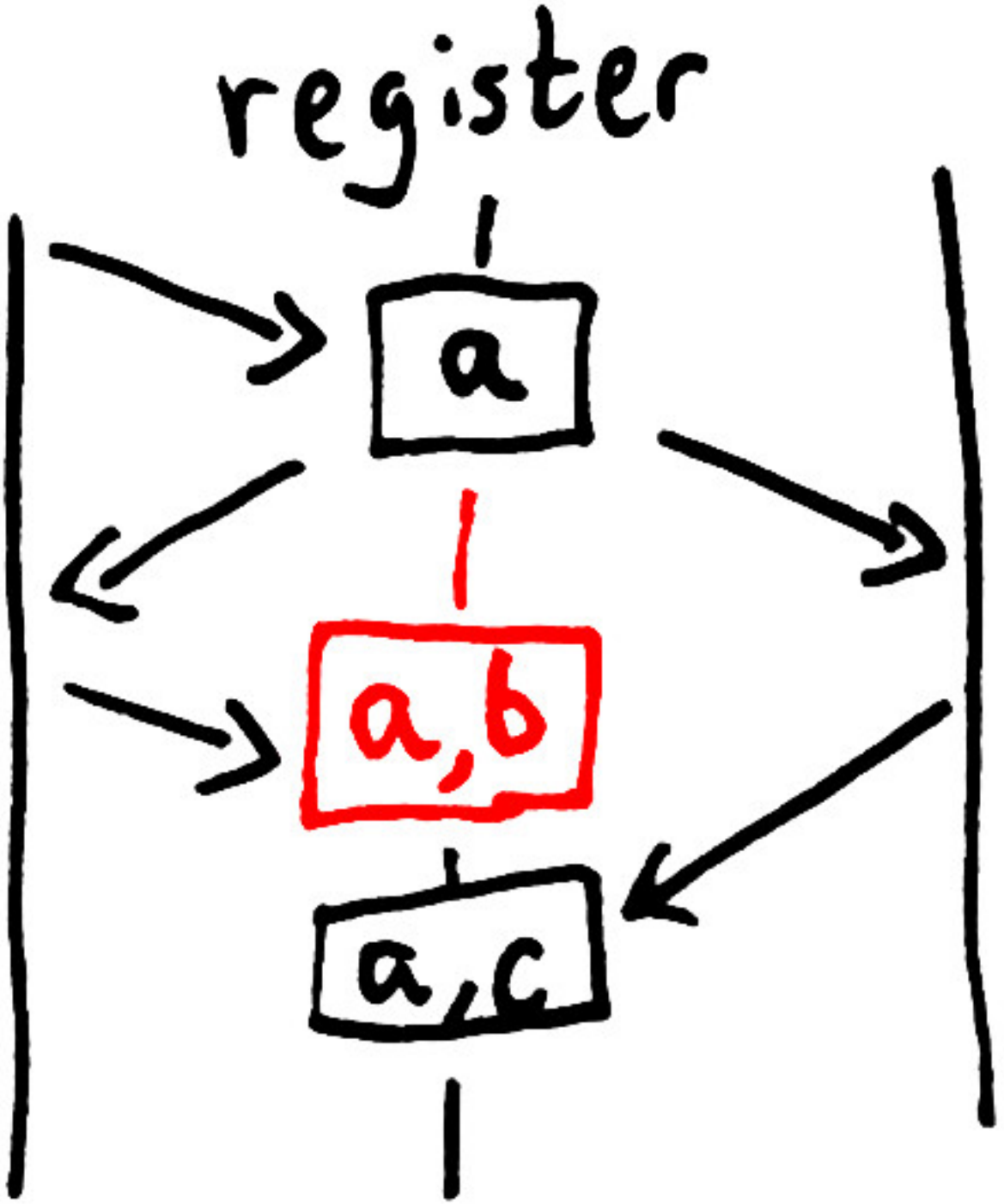
"But Last Write Wins is bad, right?"

Well Sally, it depends!

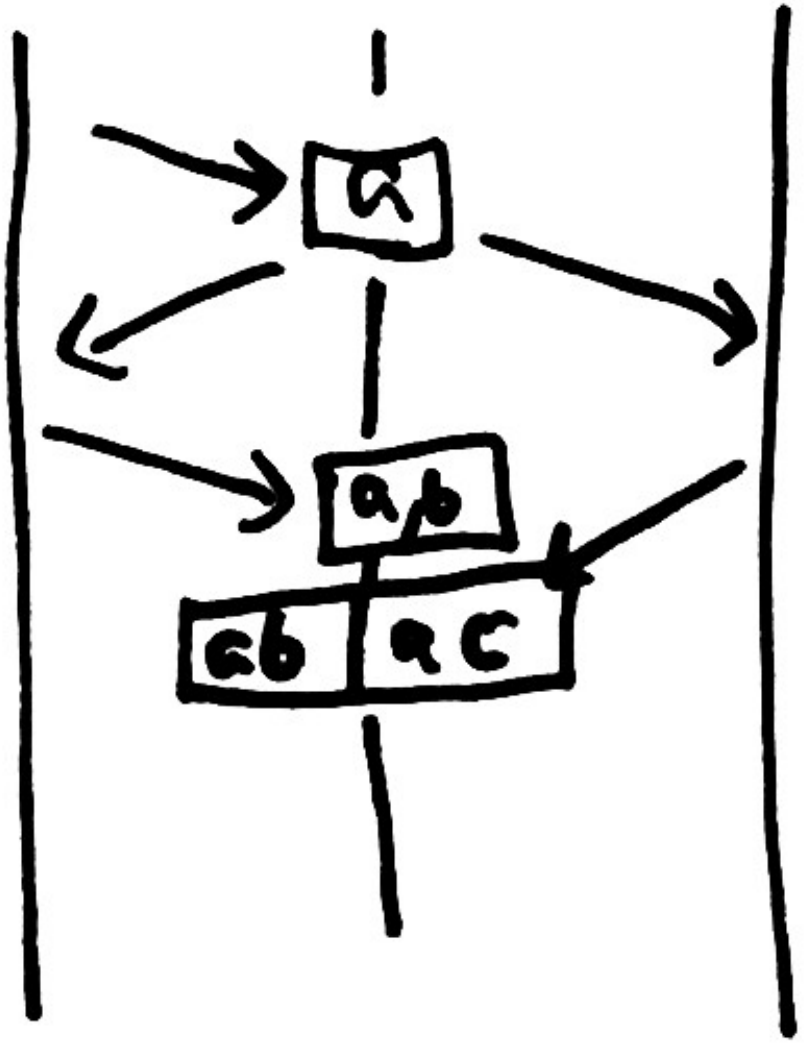
You can't safely change an LWW register.

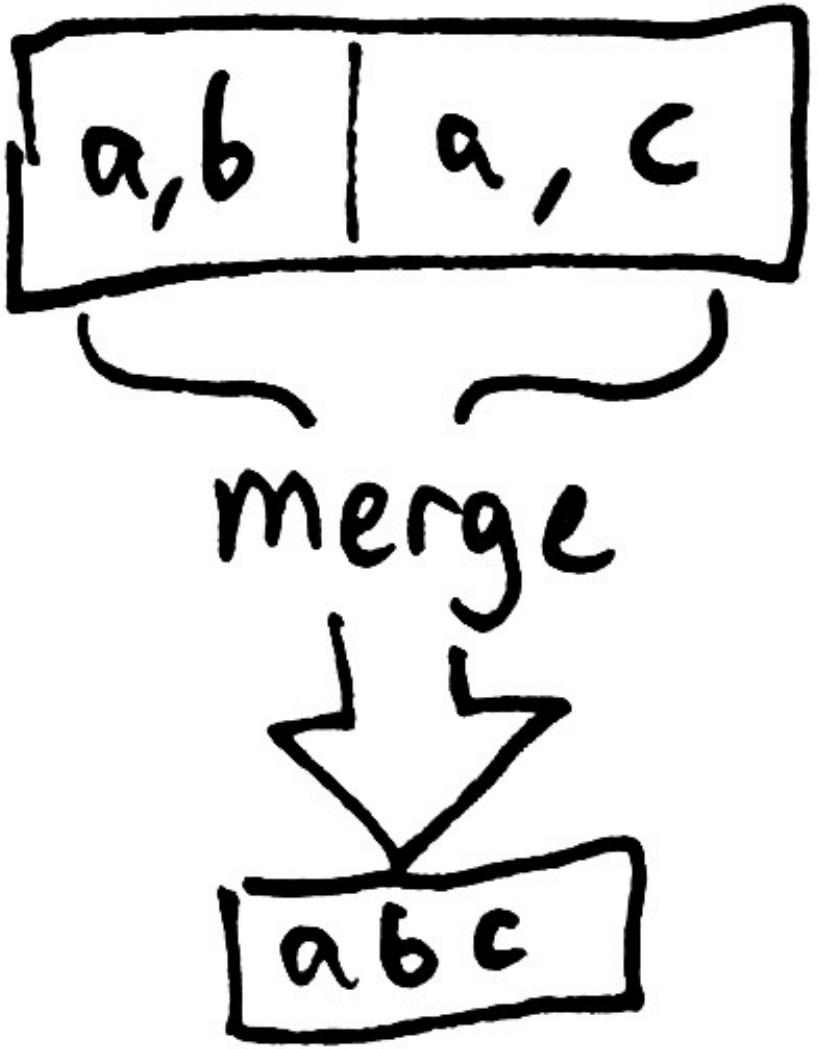
Safe?

- Guarantee that your write is causally connected to a future read.


register
a
a,b
a,c

Vector clocks identify these causally isolated writes, and give you both copies.


a
ab
ab
ac


a,b
a, c
merge
abc

Merge fn must be

- Associative
- Commutative
- Idempotent

CRDTs!

Early in Cassandra history, chose not to use vclocks for performance.

Commensurately, no safe way to modify a cell.

Cassandra Cell Change
(quorum r+w, locks)

44% acknowledged

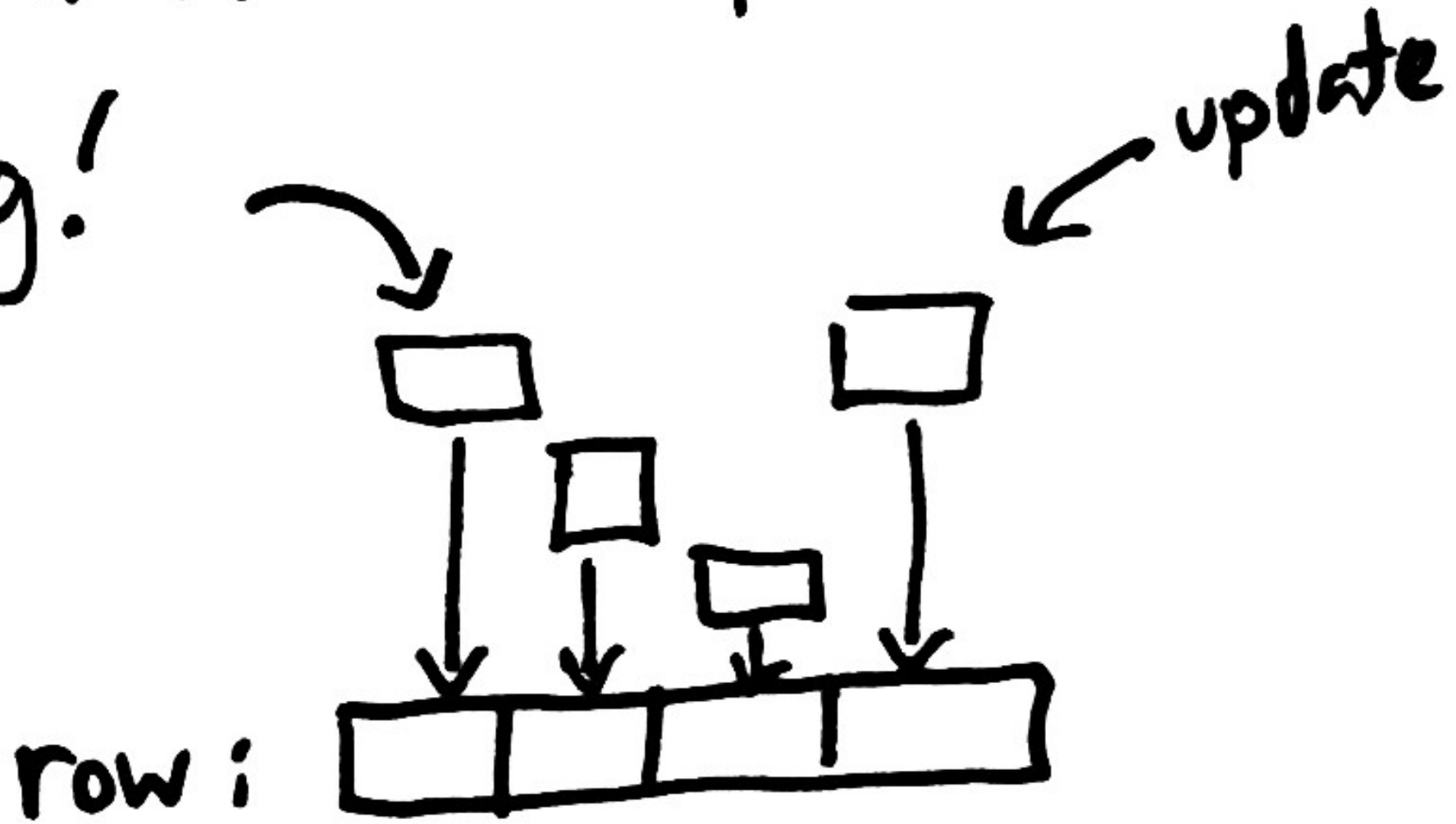
27% false positives

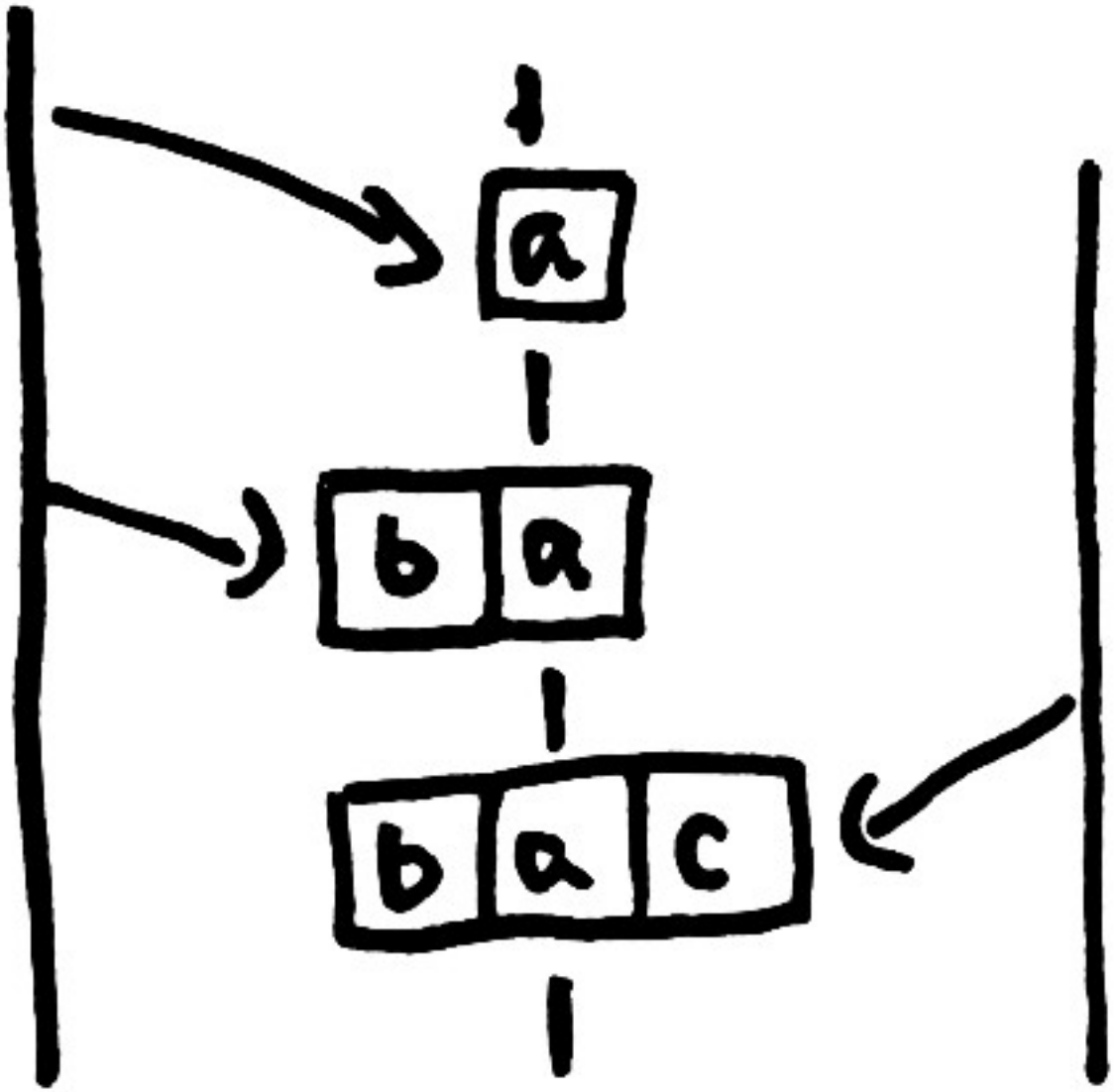
0% false negatives

or tables. Cassandra supports tuning availability and consistency, and always gives you partition tolerance. Cassandra can be tuned to give you strong consistency in the CAP sense where data is made consistent across all the nodes in a distributed database cluster. A user

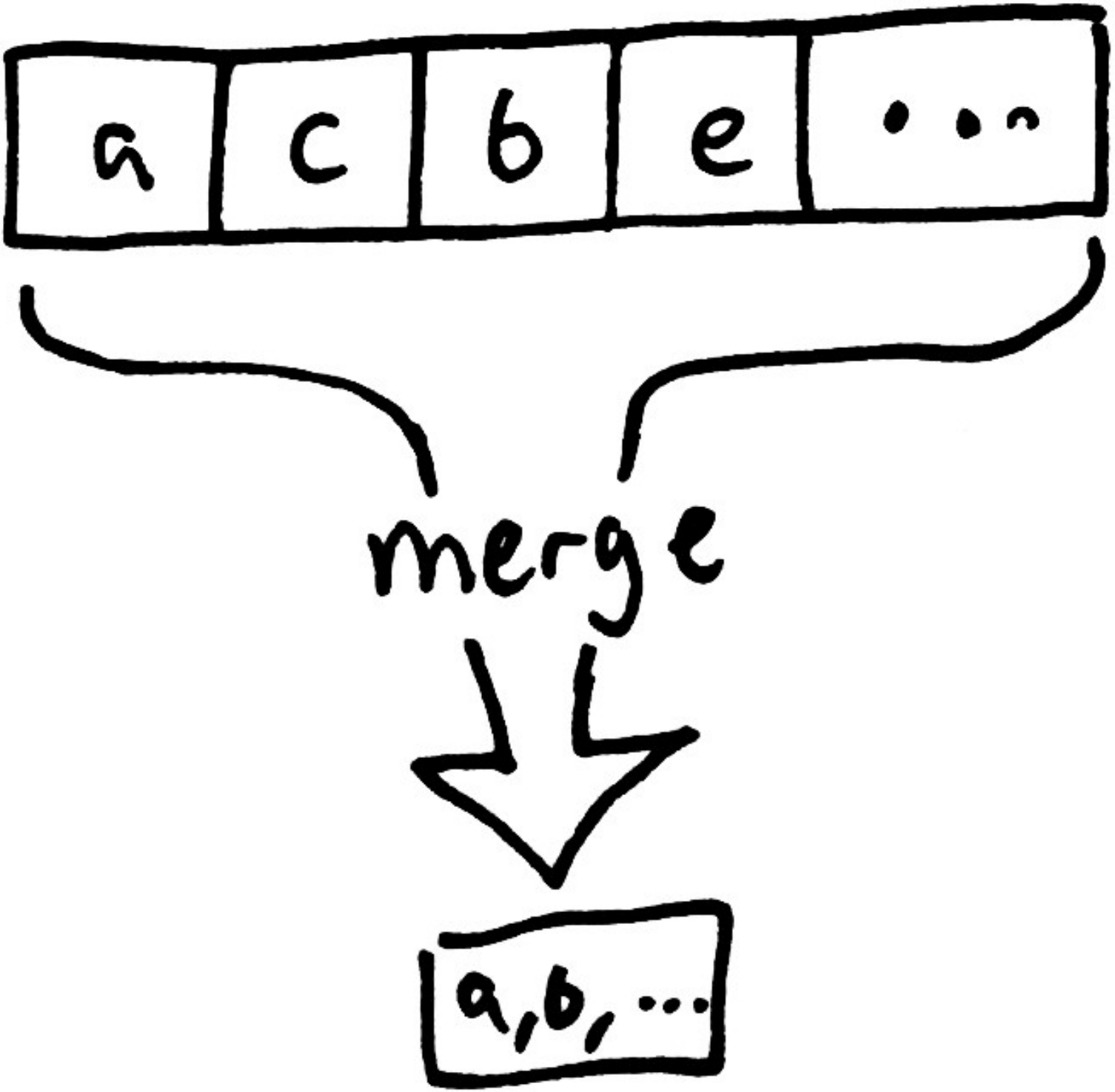
Instead, Cass. has evolved efficient ways to write every change to a distinct cell.

Immutable operation log!

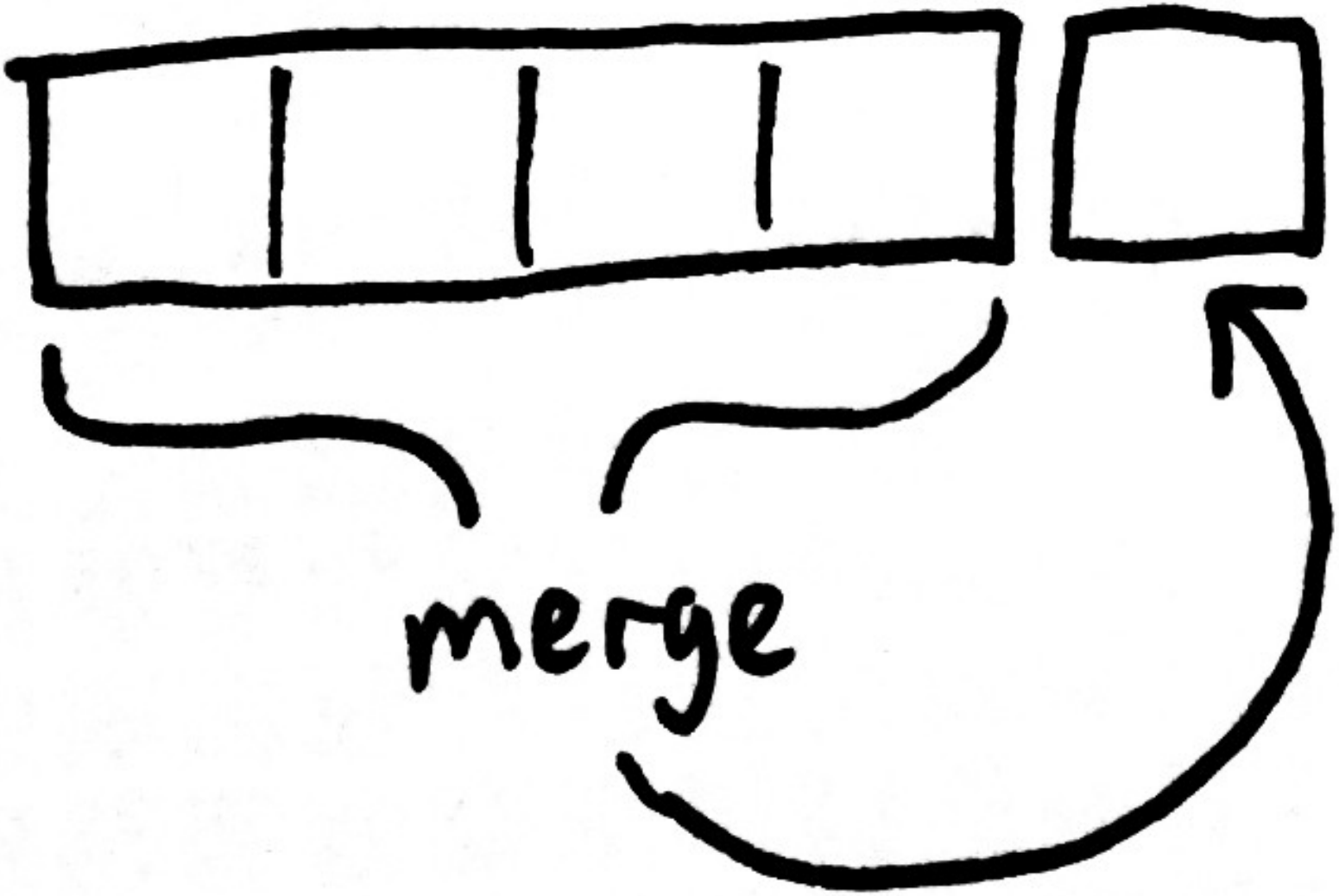



a
b
a
b
a
c

Reads require merge functions

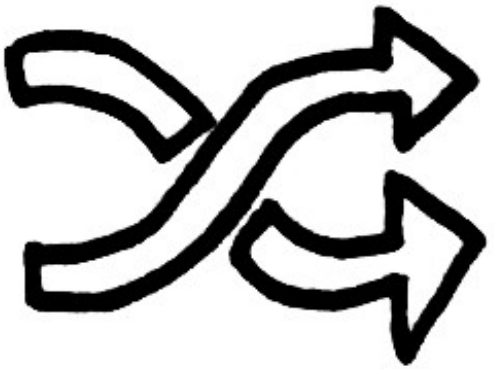

a
c
b
e
merge
a,b,...

Garbage Collection?

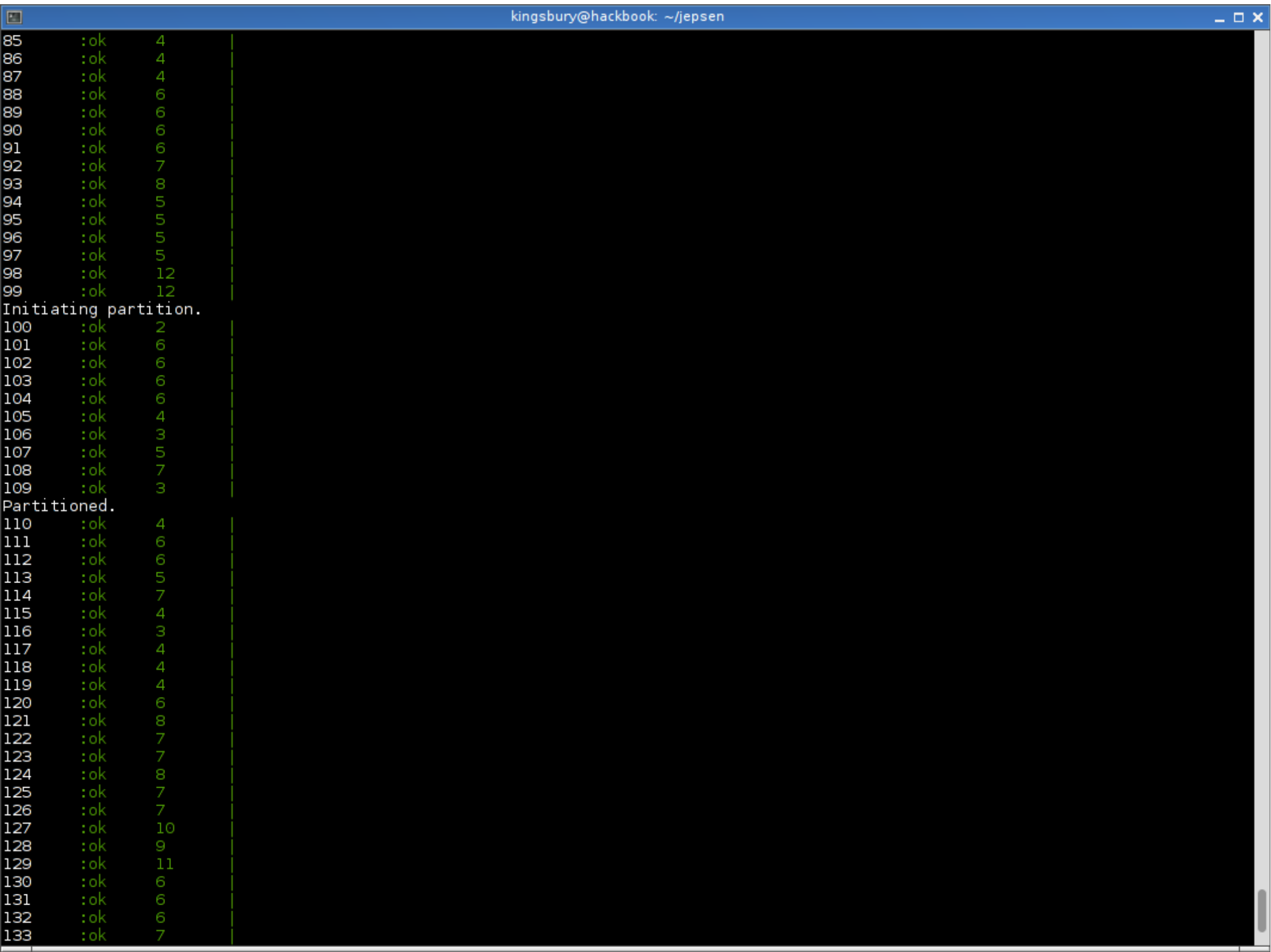


Vector clocks handle identifying old, unneeded data.

Since there is no global ordering for updates, still need commutative merges.



So! Cassandra and Riak have similar consistency semantics. CQL provides limited CRDTs.


kingsbury@hackbook: ~/jepsen
85 :ok 4
86 :ok 4
87 :ok 4
88 :ok 6
89 :ok 6
90 :ok 6
91 :ok 6
92 :ok 7
93 :ok 8
94 :ok 5
95 :ok 5
96 :ok 5
97 :ok 5
98 :ok 12
99 :ok 12
Initiating partition.
100 :ok 2
101 :ok 6
102 :ok 6
103 :ok 6
104 :ok 6
105 :ok 4
106 :ok 3
107 :ok 5
108 :ok 7
109 :ok 3
Partitioned.
110 :ok 4
111 :ok 6
112 :ok 6
113 :ok 5
114 :ok 7
115 :ok 4
116 :ok 3
117 :ok 4
118 :ok 4
119 :ok 4
120 :ok 6
121 :ok 8
122 :ok 7
123 :ok 7
124 :ok 8
125 :ok 7
126 :ok 7
127 :ok 10
128 :ok 9
129 :ok 11
130 :ok 6
131 :ok 6
132 :ok 6
133 :ok 7

Cassandra CQL sets

100% acknowledged

0% false positives

0% false negatives

CQL collections are partition tolerant!

- But semantics are subtle...

BEWARE DELETES

Cassandra counters

at consistency = quorum

+/- 50% actual value

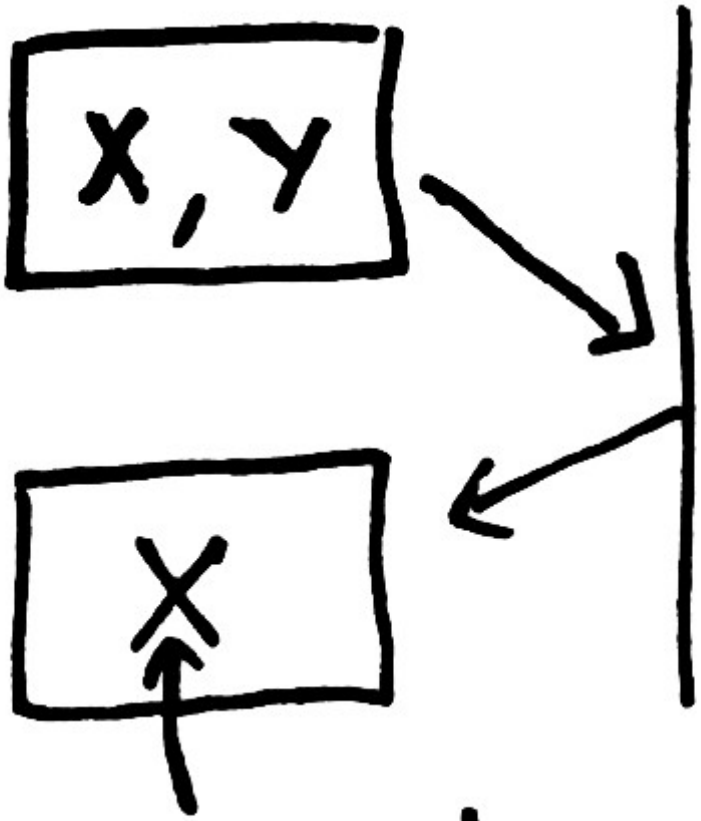
What about ISOLATION?

"From a transactional ACID... standpoint, [atomic row ops] give Cassandra transactional AID support"

—Datastax "About Writes"

"Atomic"

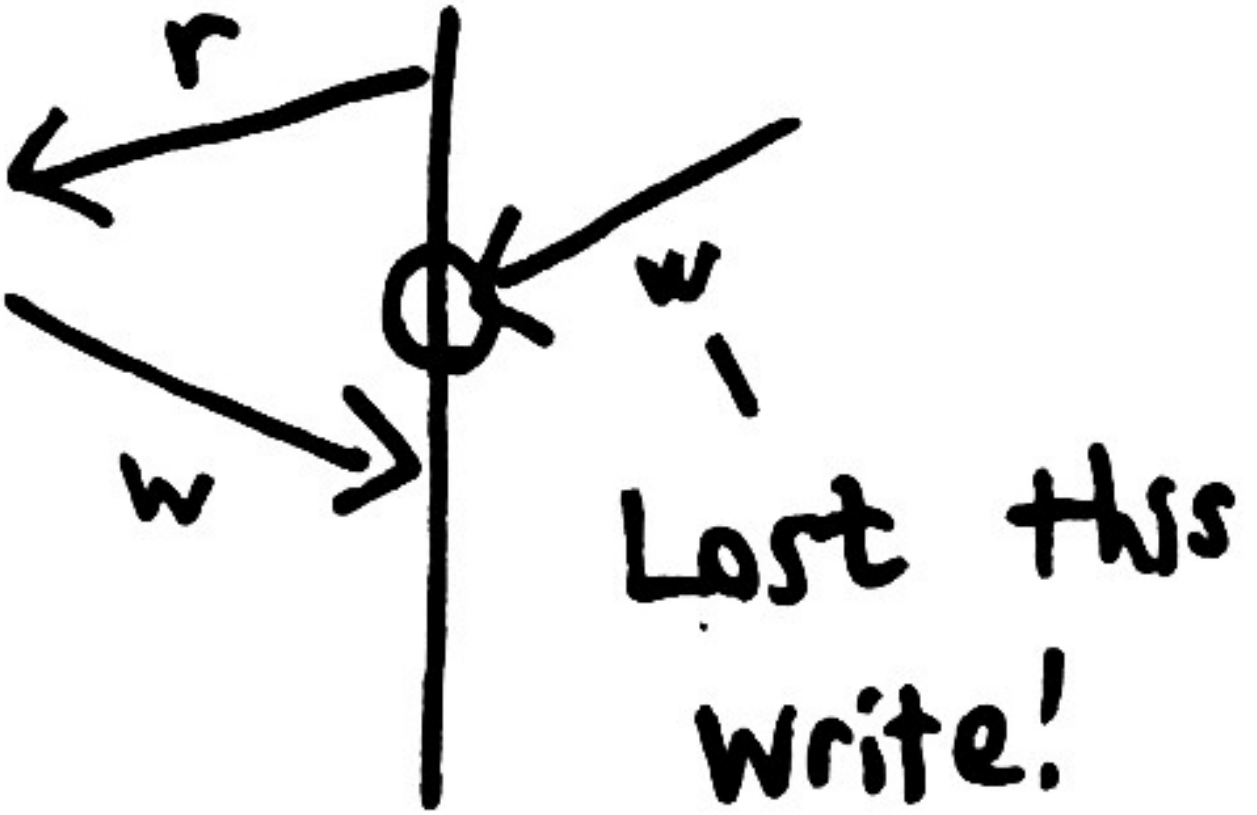
"Isolated"



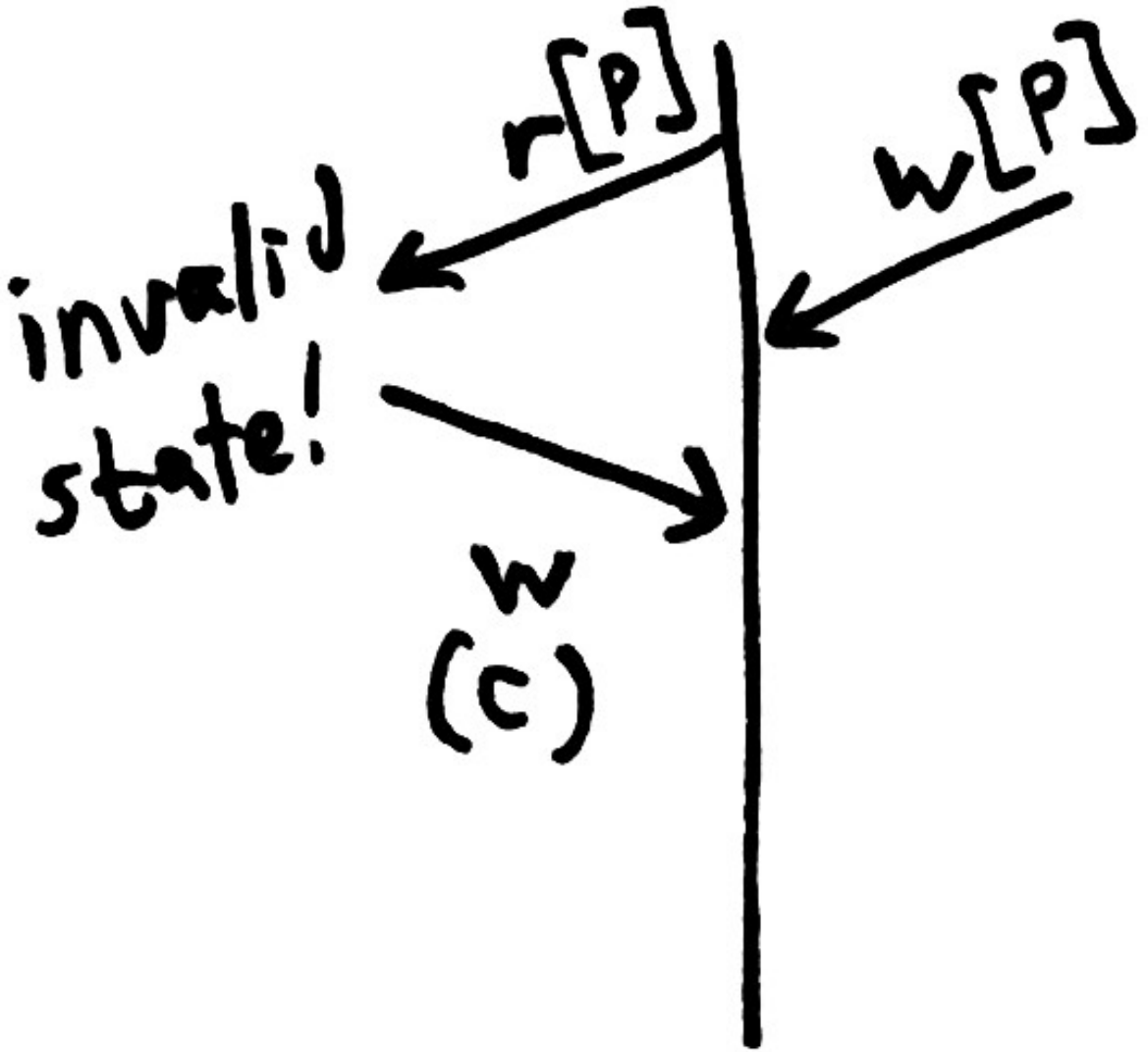
Wrong!

... um...

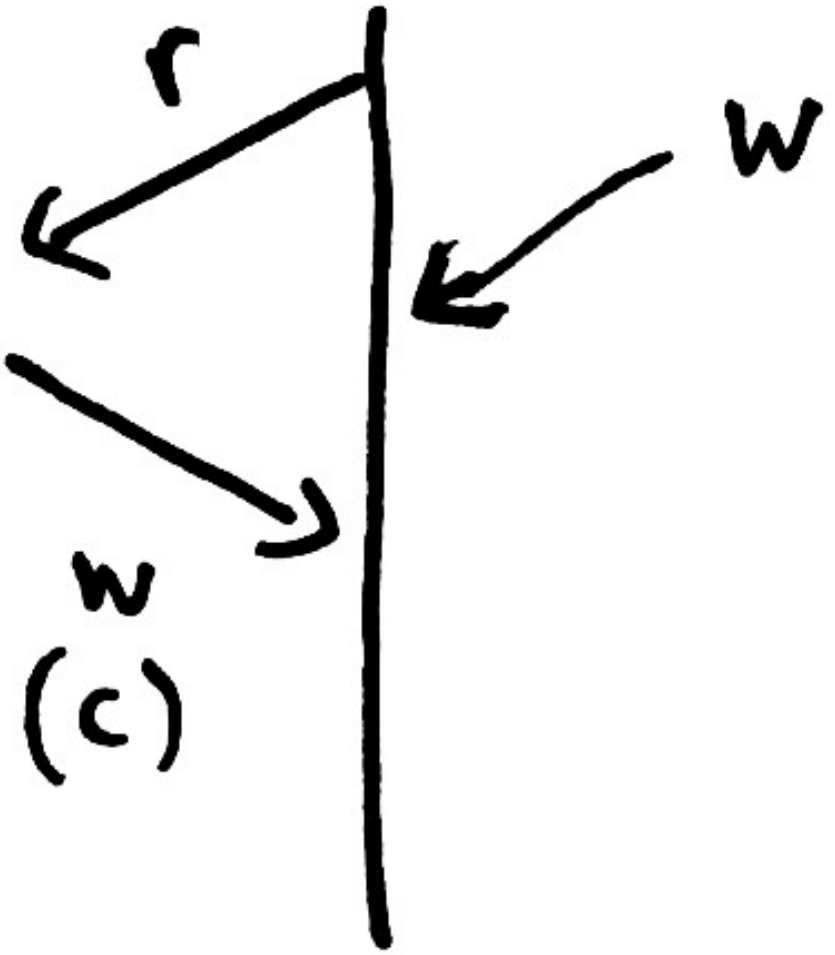
P4: Lost update (the LWW problem!)



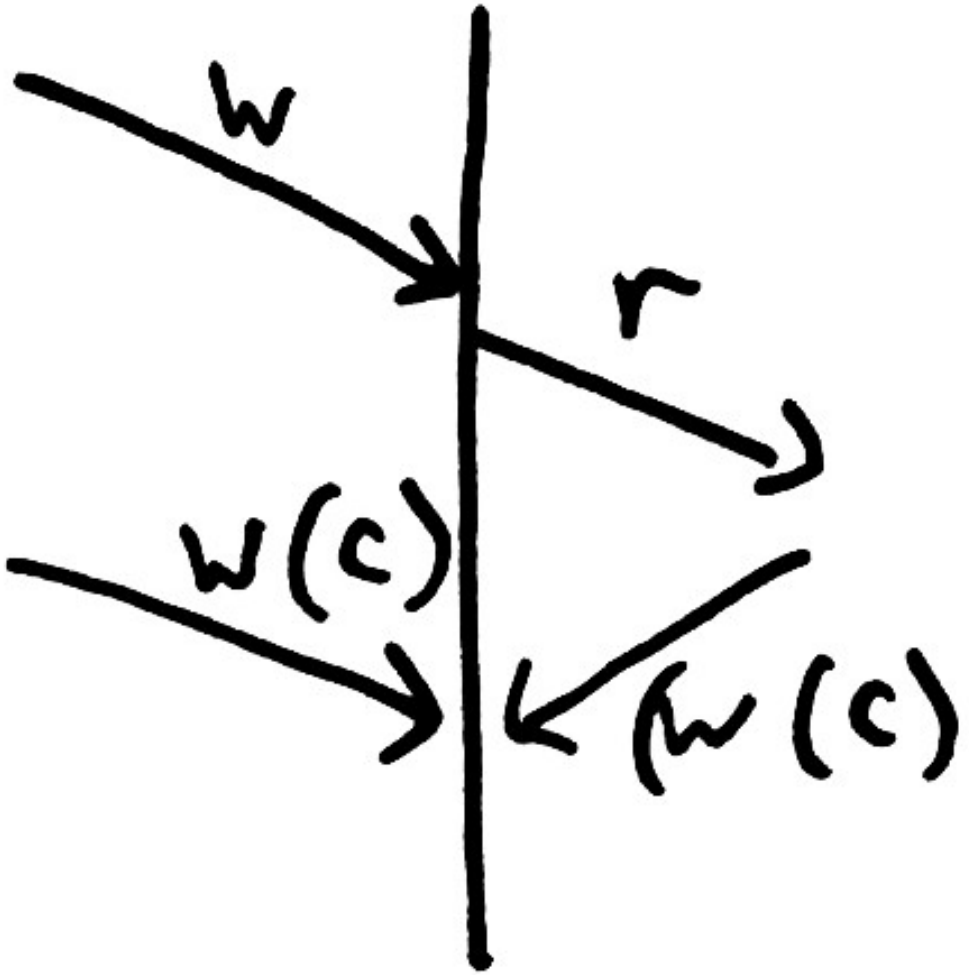
P3: Phantom



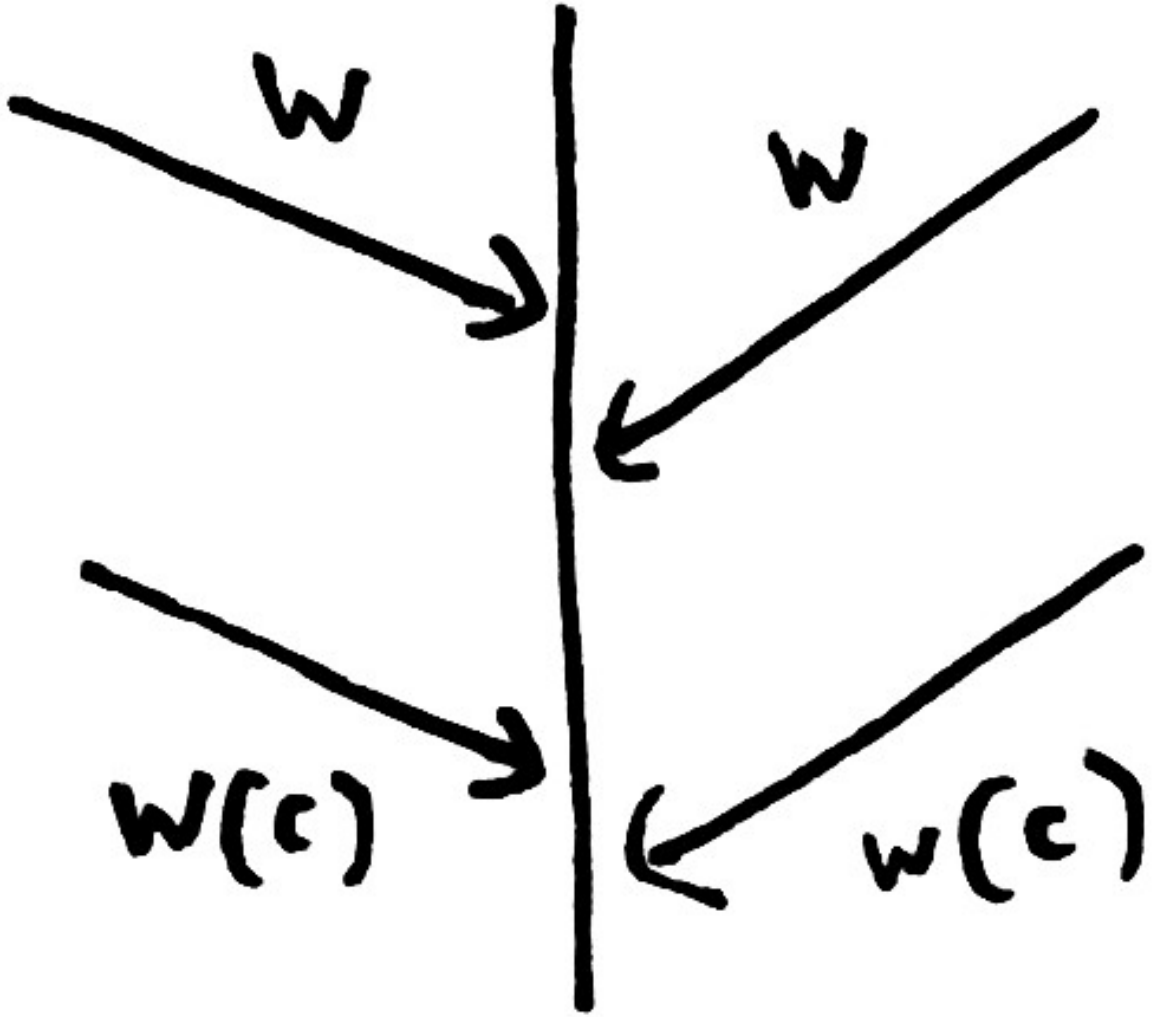
P2: Fuzzy Read

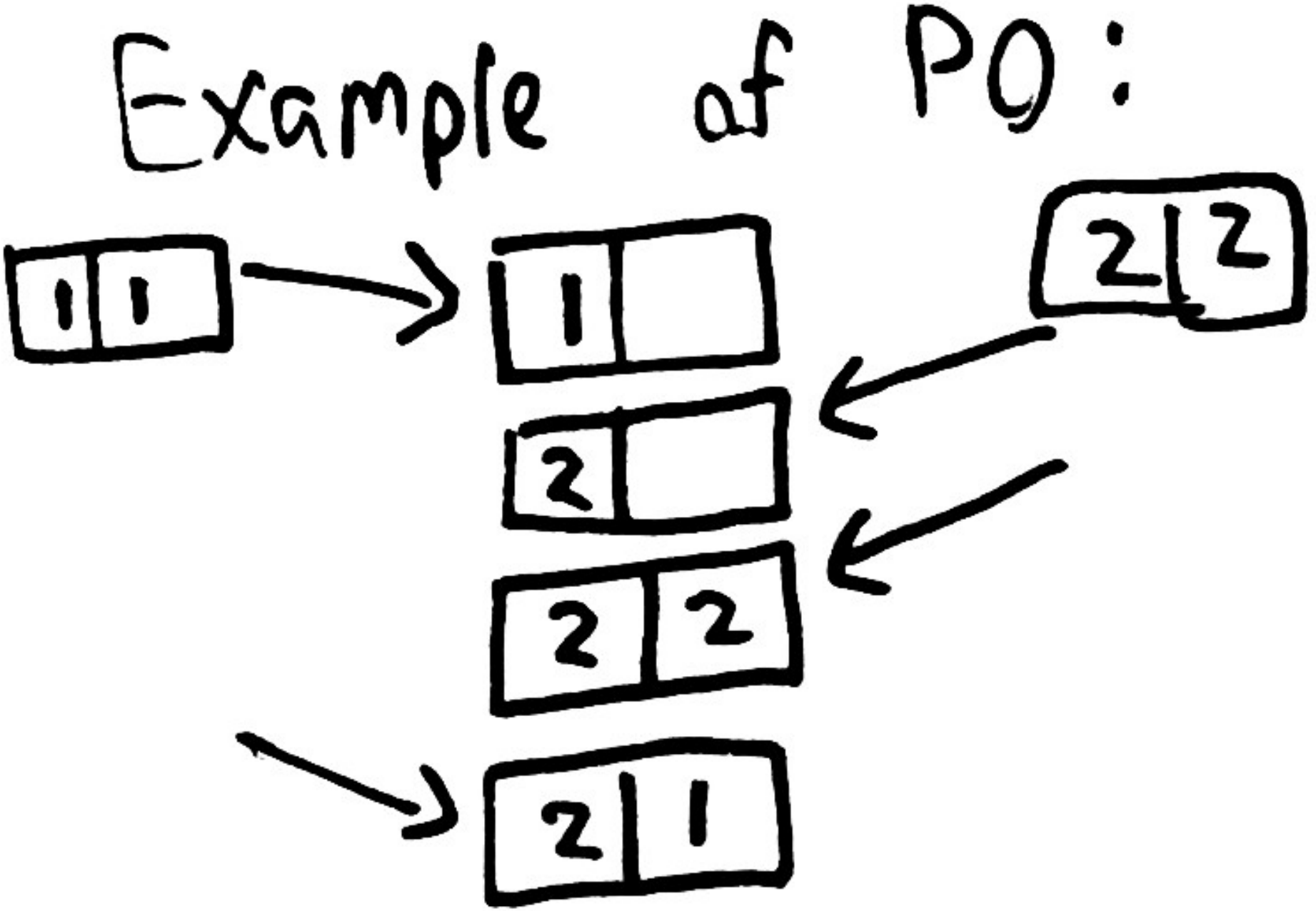


P2: Dirty Read



P0: Dirty write




Example of PO:
1 1
1
2 2
2
2 2
2 1

"ANSI SQL Isolation should ... require PO for all isolation levels"

-Berenson, et al

MSR-TR-95-51

Cassandra 1.1 as row-level updates are now made in isolation. Cassandra 1.1 guarantees that if you update both the login and password in the same update (for the same row key) then no concurrent read may see only a partial update.

From a transactional ACID (atomic, consistent, isolated, durable) standpoint, this enhancement now gives Cassandra transactional AID support. A write is isolated at the row-level in the storage engine.

Actually, they are isolated in the sense of ACID,

Certainly there is nothing like ACID, but a single multi-cell update to a given record is guaranteed to be _atomic_, and if you have two concurrent multi-cell updates to a single record, they are guaranteed to eventually resolve to a consistent ordering of those operations

The difference is that all the cells with the same `id` would belong to the same partition and stored together, so you'd be able to write them atomically and read them together cheaply in a single operation.

It wasn't really mentioned in the article but if you do both {X1,Y1} and {X2,Y2} as full row updates then no you can't end up with a mixed row.

http://www.datastax.com/dev/blog/row-level-isolation

In Cassandra the second update agreed upon by the cluster would "win", so either (X1,Y1) or (X2,Y2).

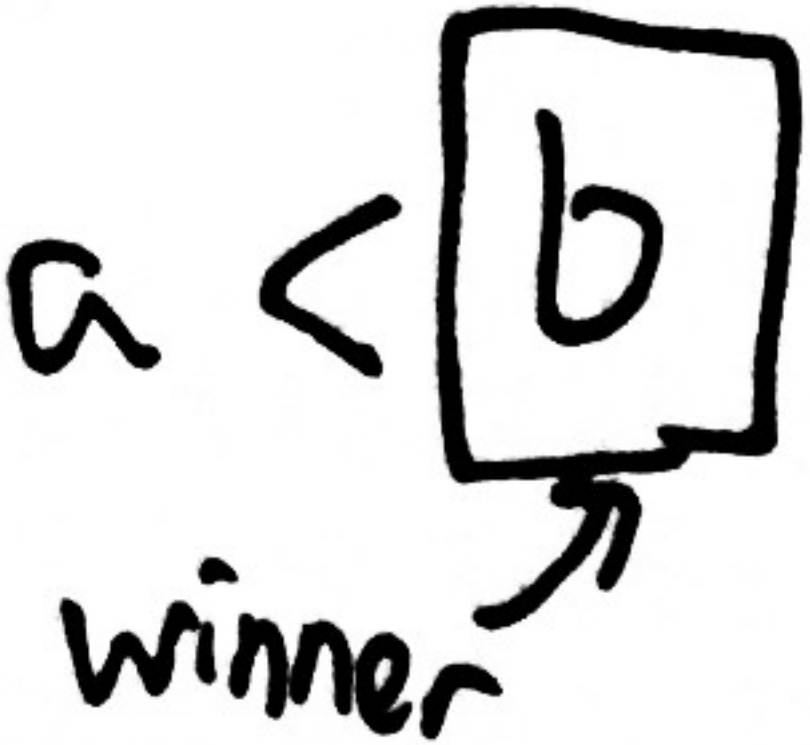
Cassandra allows PO (and more!)

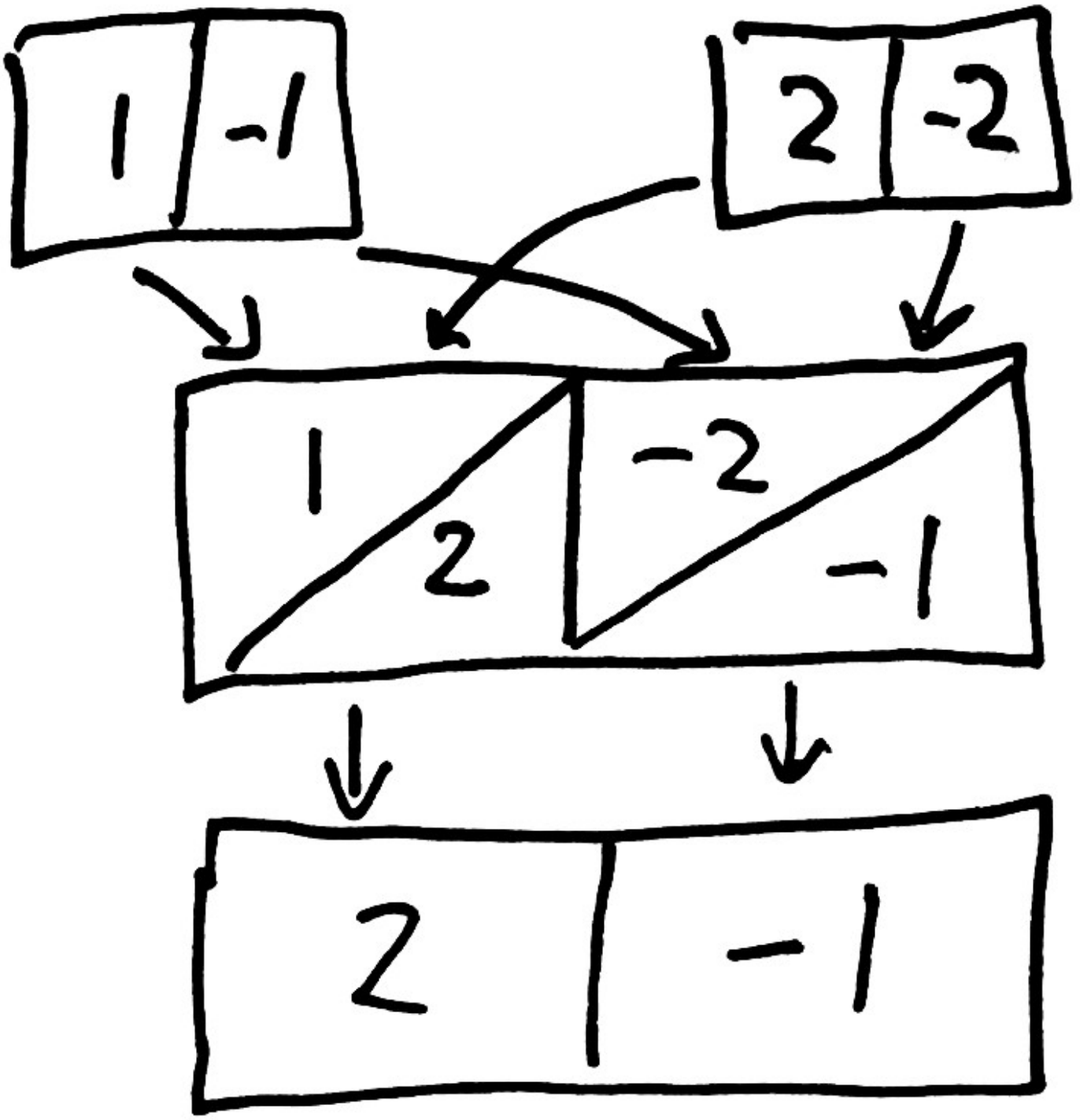
In Cassandra, write order doesn't matter. LWW always takes prio.

What if timestamps are equal?

$$t_a = t_b$$

Compare values



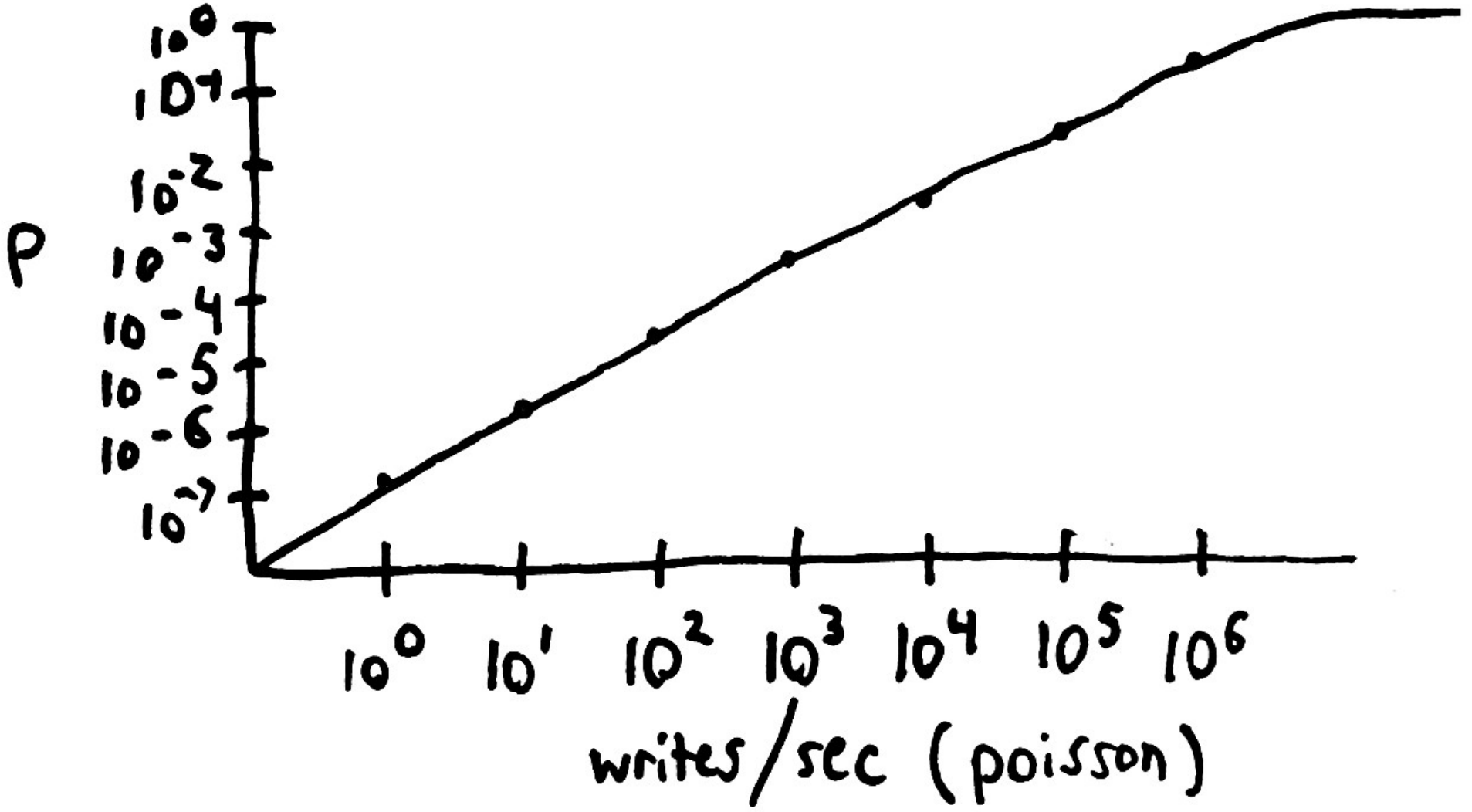

1
-1
2
-2
1
2
-2
-1
2
-1

"But timestamps never conflict!"

"But timestamps never conflict!"

It Depends!™

Probability of a microsecond conflict visible in a read:



10 writes/sec

1 read/sec

34% chance of collision in a day

2 writes per row,
separated by mean
latency 100 ms
(poisson processes)

$$p(\text{corrupt}) = 5 \times 10^{-6}$$

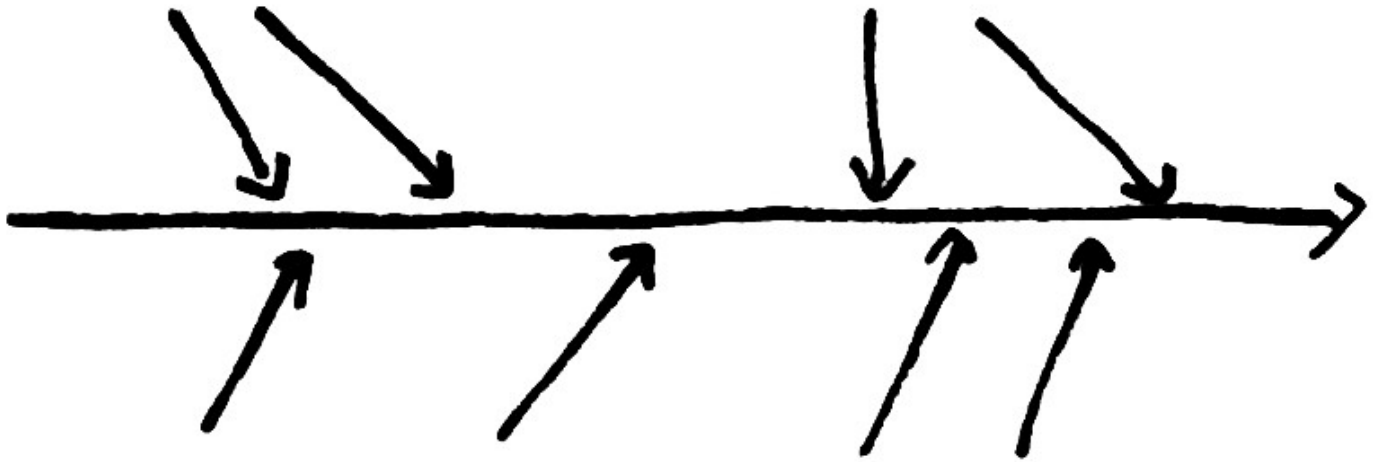
Theoretical Limit!

. With Cassandra
1.2.8 & 2.0.0,
Java driver 1.0.3,
2 writes / row
mean latency 100ms

1 in 250
rows found
corrupt!

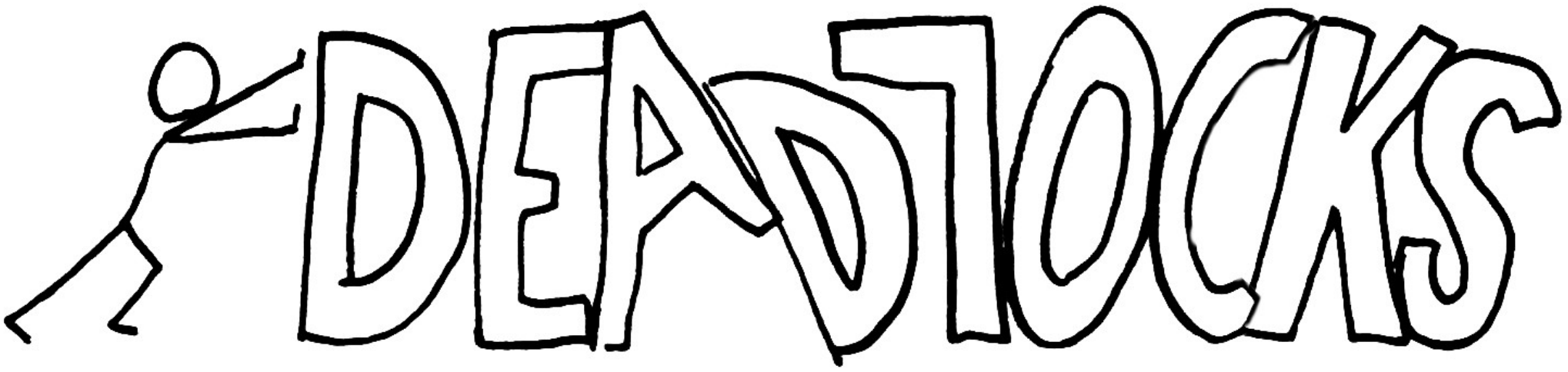

USE
lightweight
TRANSACTIONS

"Linearizable Consistency"



- DataStax, "Lightweight transactions in Cassandra 2.0"

Java driver does not support Paxos transactions.


DEADLOCKS

Paxos txns lock up the cluster permanently.

-clear system.paxos

```
$ git checkout paxos-fixed-hopefully
```

Paxos transactions are not linearizable.

#6012 #6013

Can accept two values for same paxos round.

Failed proposals can still succeed in later rounds.

Cassandra Transactions

5% acknowledged

4% false positives

9% false negatives

"Paxos made without stress tests."

Open source has had the reputation of producing good imitations, but not innovation. Perhaps Cassandra's origins as a hybrid of Dynamo and Bigtable did not disprove this, but Apache Cassandra's development of lightweight transactions and CQL are true industry firsts.

Cassandra strategies

- Use as a strict AP store
- No isolation/atomicity guarantees for writes
- Avoid transactions

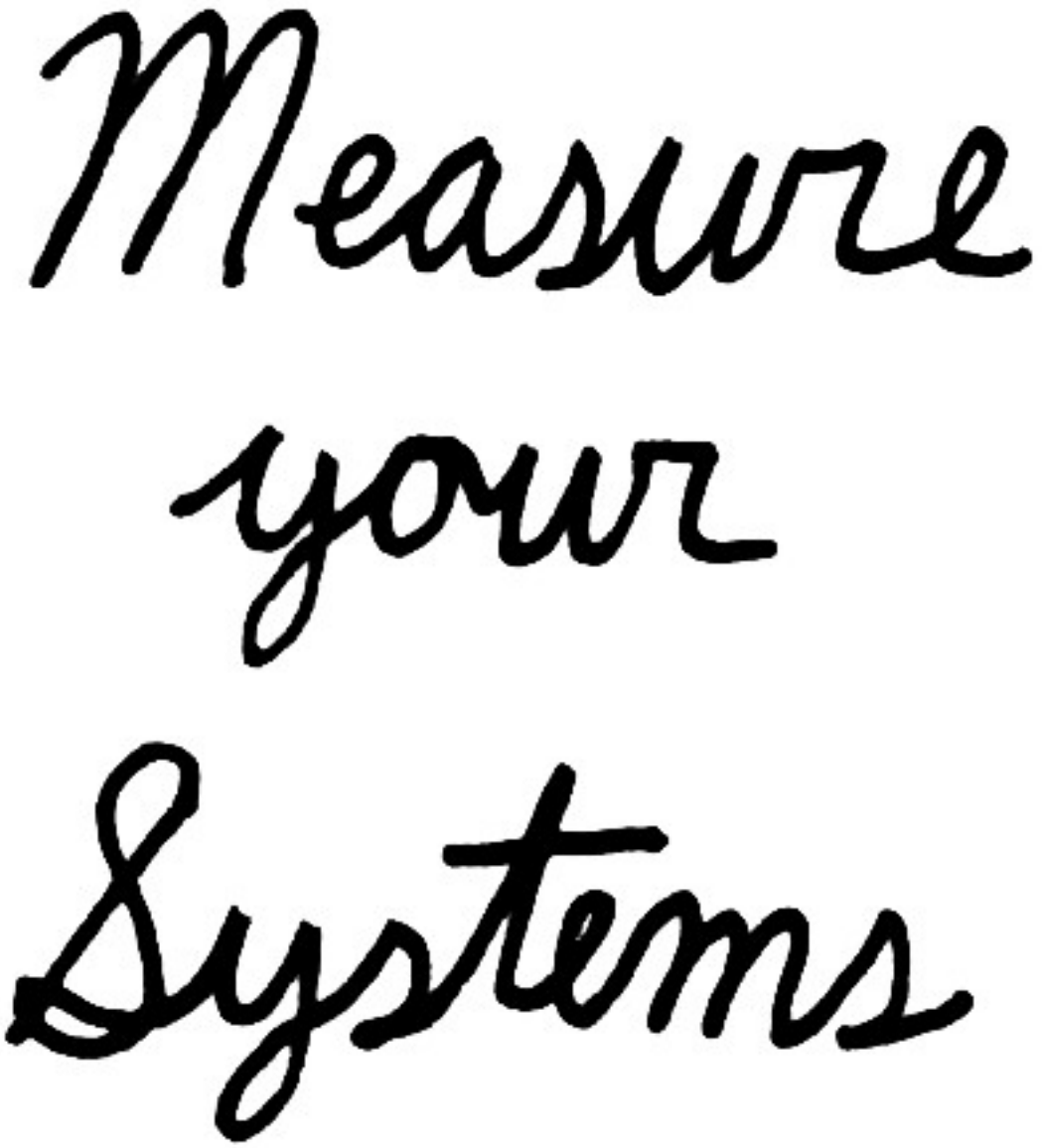
to

ReCAP

Jepsen only tells you about one system

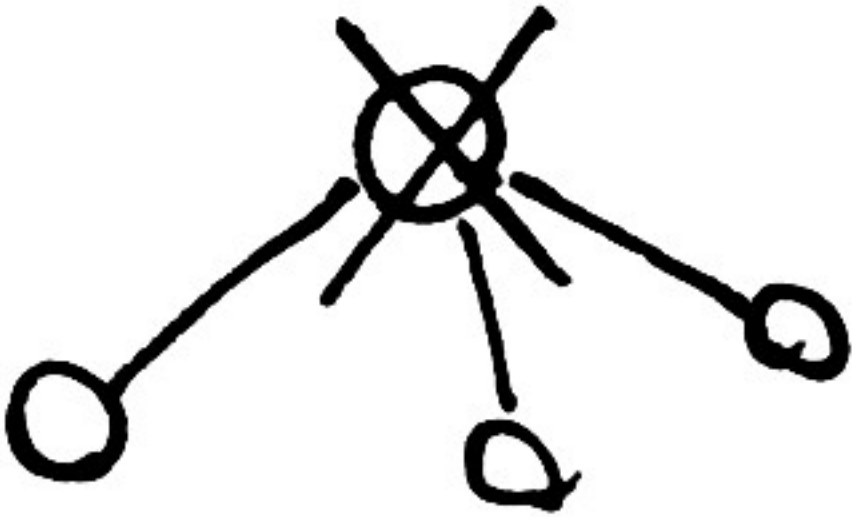
Clients are
a part of the
distributed system!

- Clients
- Servers
- Network
- Semantics


Measure
your
Systems

ASK:

"What if we lose this node?"

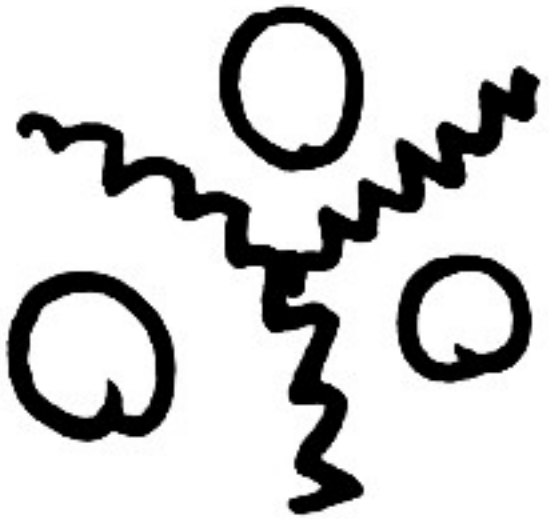


"What if these nodes paure for 3 minutes?"

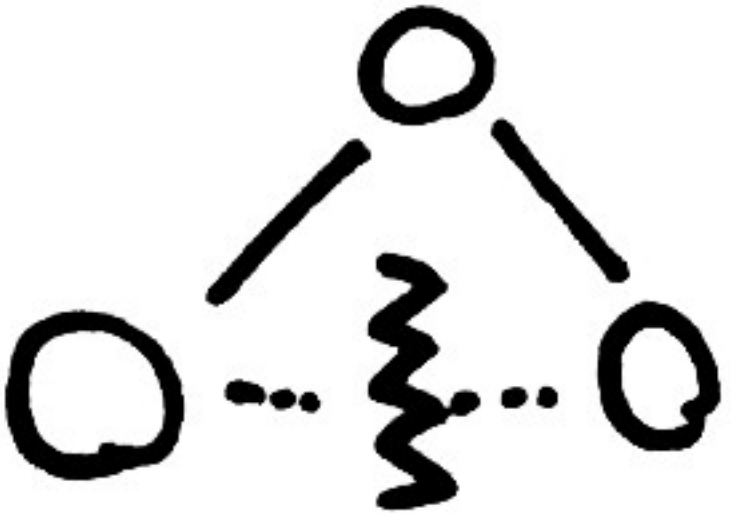
"How do we rely on clocks?"

"What if the network delivers this message 5 minutes late?"

"What if there is no majority?"



"What about partial visibility?"



"What safety and liveness guarantees does this system really need?"

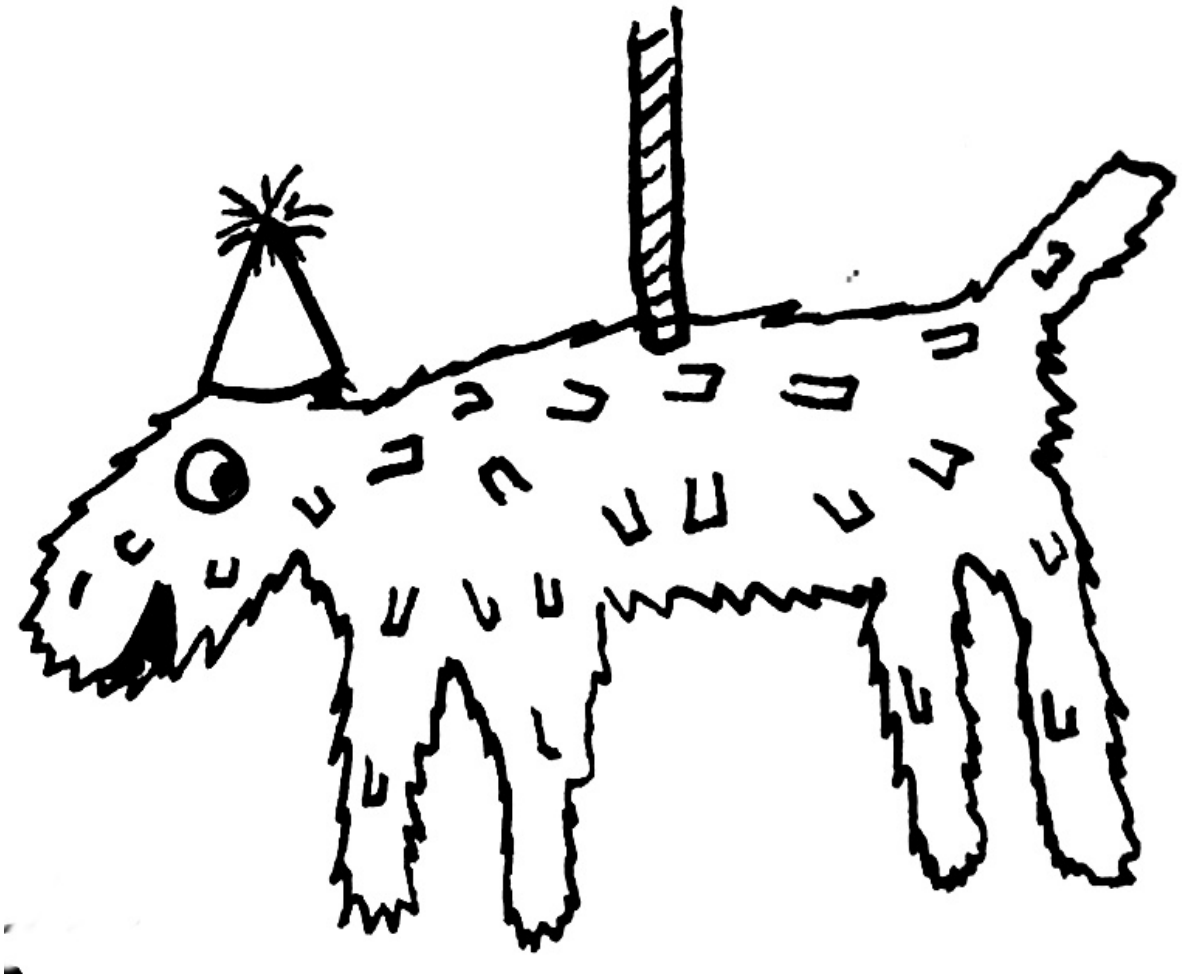
Data loss or inconsistency can be OK!

Balance

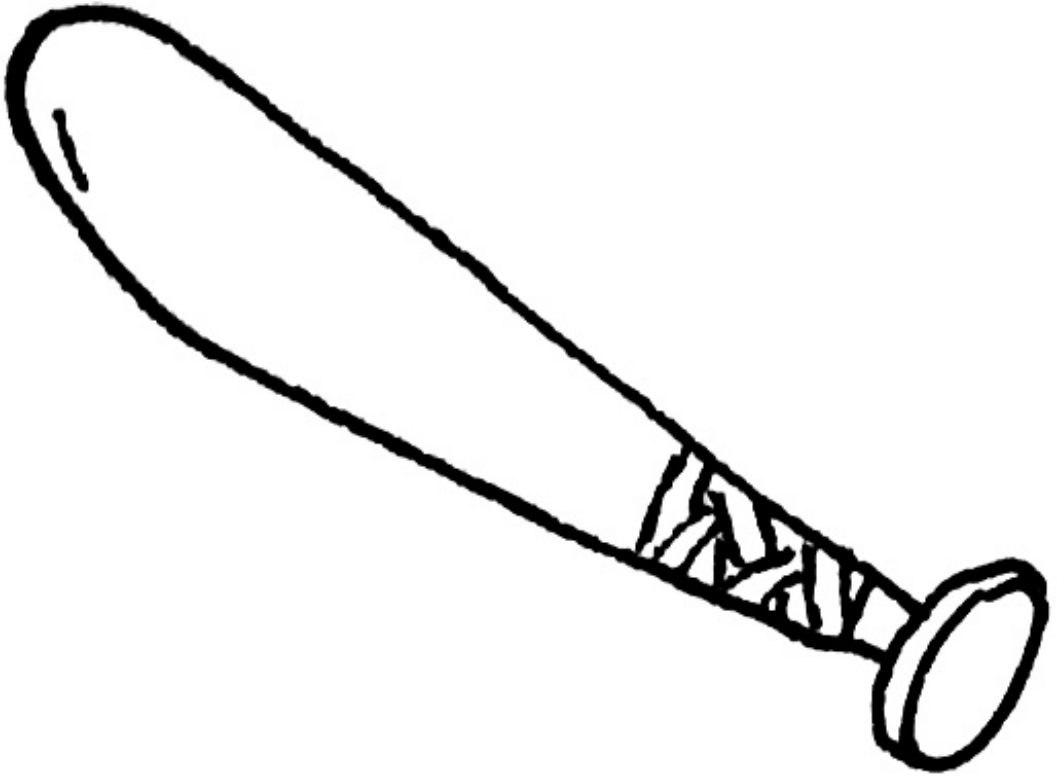
Complexity

— with —

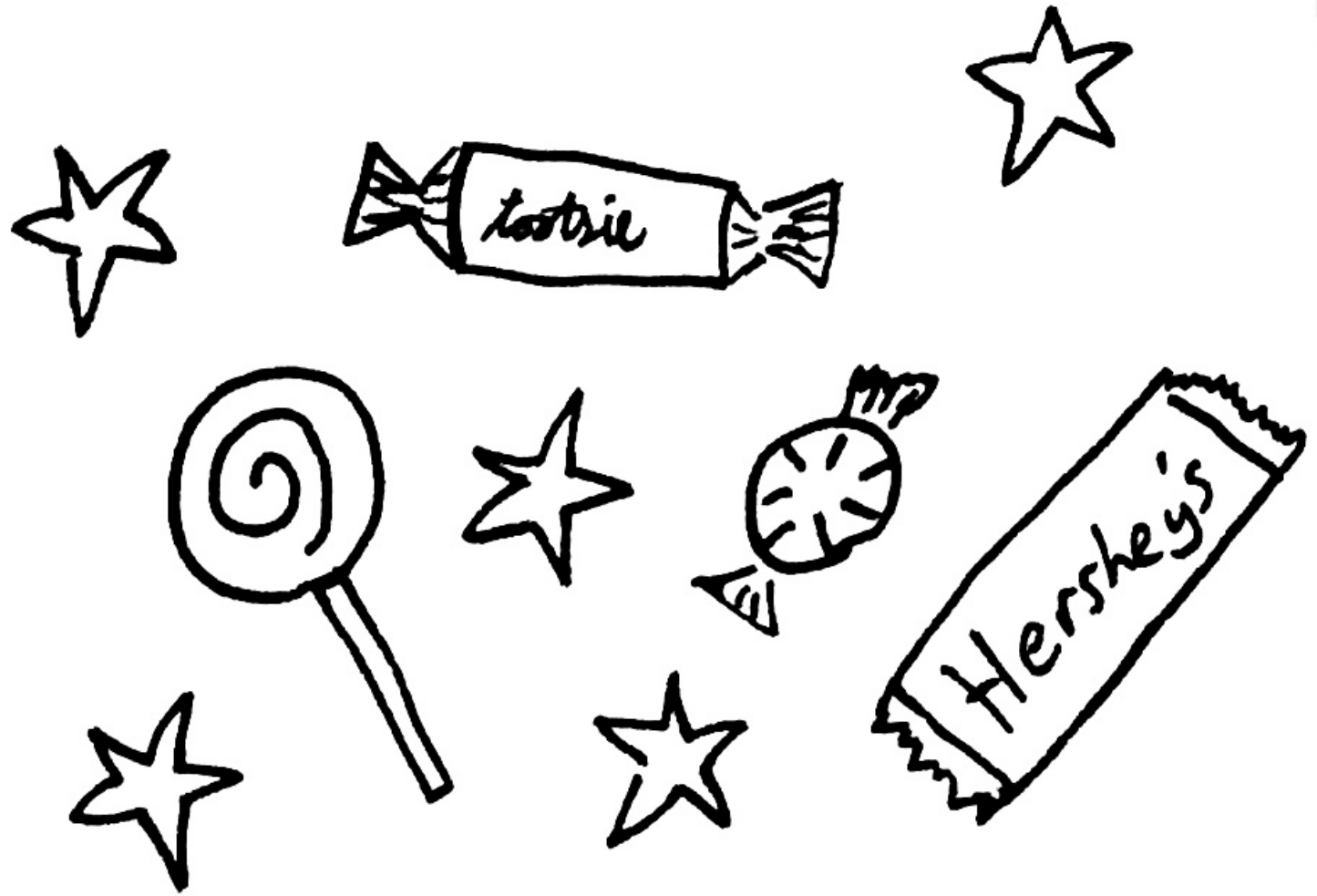
Correctness



Write a
test program!



Cause
Failures



SEE WHAT HAPPENS!

Thank You!

Jonathan Ellis

Rick Branson

Eric Lindvall

leftside

Patrick McFadin

Aleksey Yeschenko

Seth Proctor

Duke Adamonis

Neha Narkhede

Jun Rao

Kelly Sommers

Drew Blas

Michael Klishin

Camille Fournier

Jay Kreps

Joeseph Blomstedt

Peter Bailis

Thanks!

github.com/aphyr/jepsen