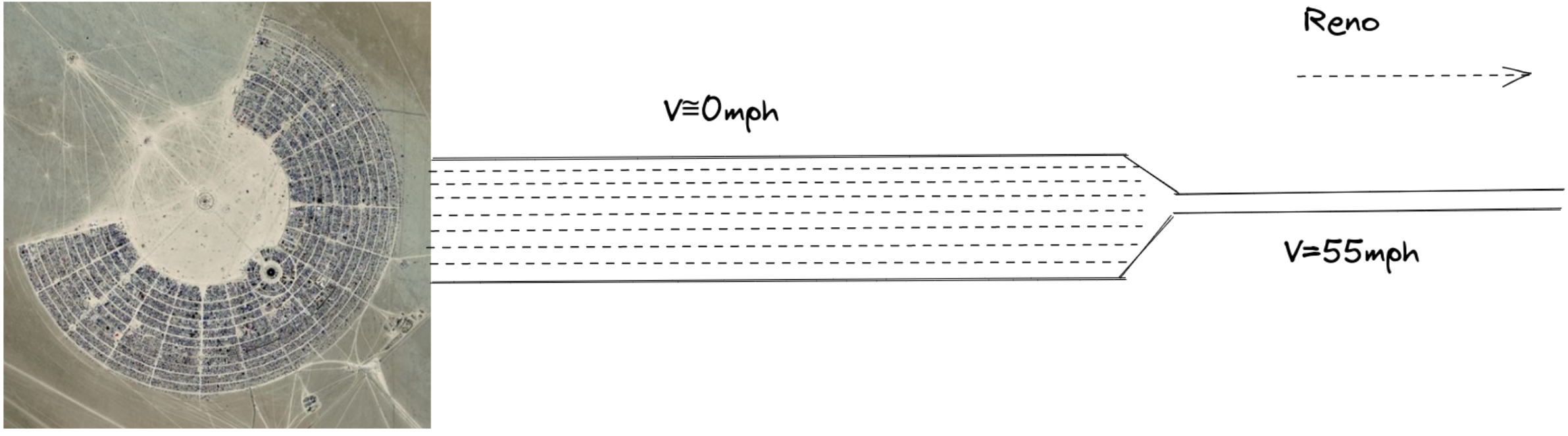
The Math of Scalability

Avishai Ish-Shalom (@nukemberg)

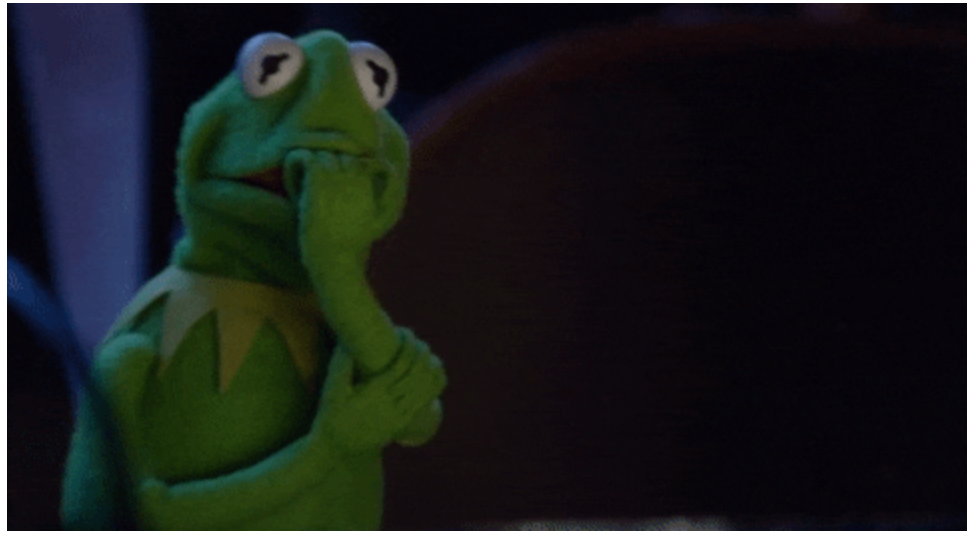

Reno
V≅0mph
V=55mph

Reno

V≅55mph

V=55mph

Math???



Define ”scalability”

Define ”scalability”

The relation between

- Resources
- Processing time
- Problem size / Work

$$S(R, T, W)$$

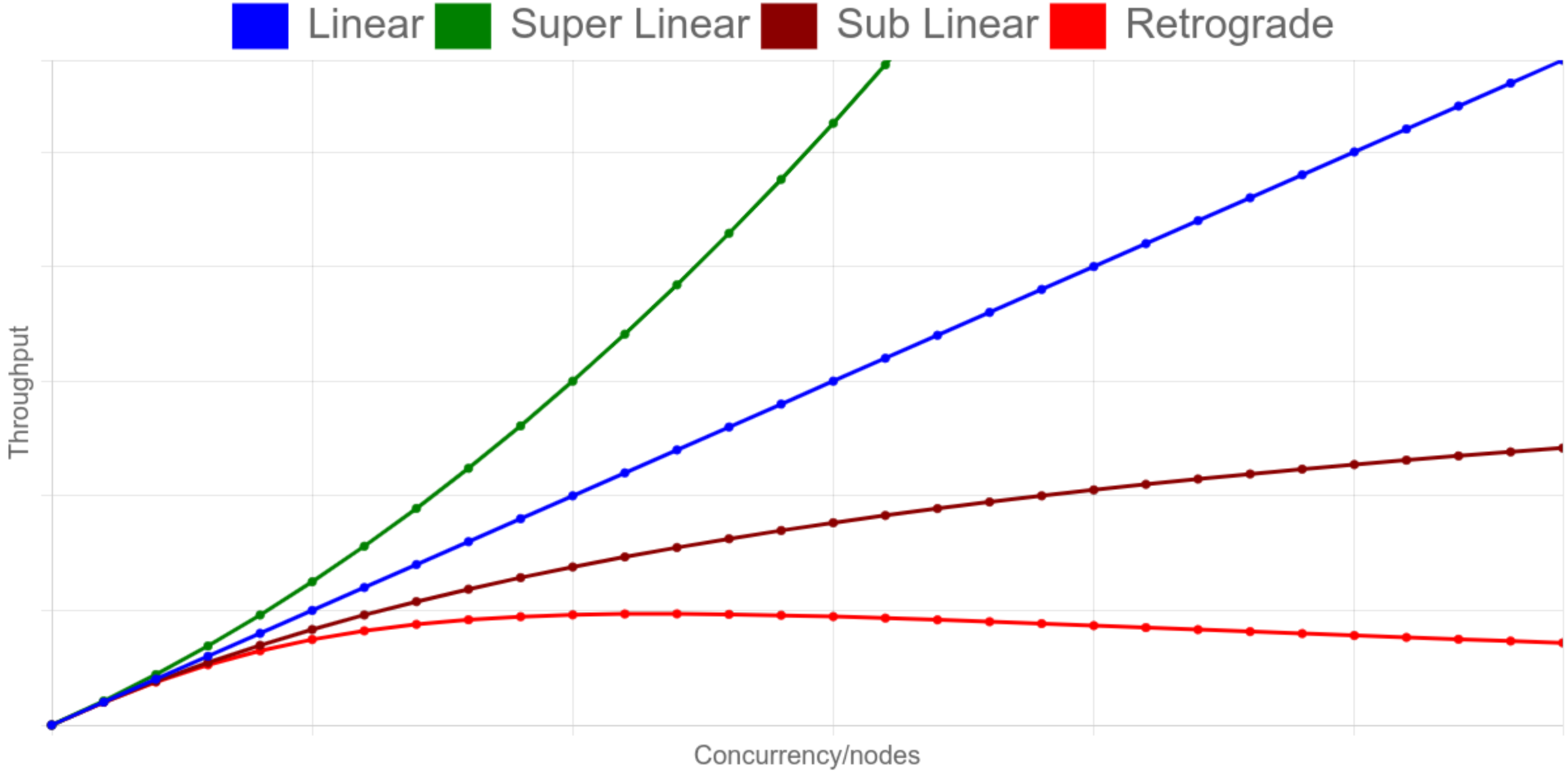
Batch

$$T = S(R) \mid W = const$$

Interactive

$$W = S(R) \mid T = const$$

Scalability chart



Lies, damn lies and statistics

Someone will win the lottery
but
it won't be you

The law of truly large numbers

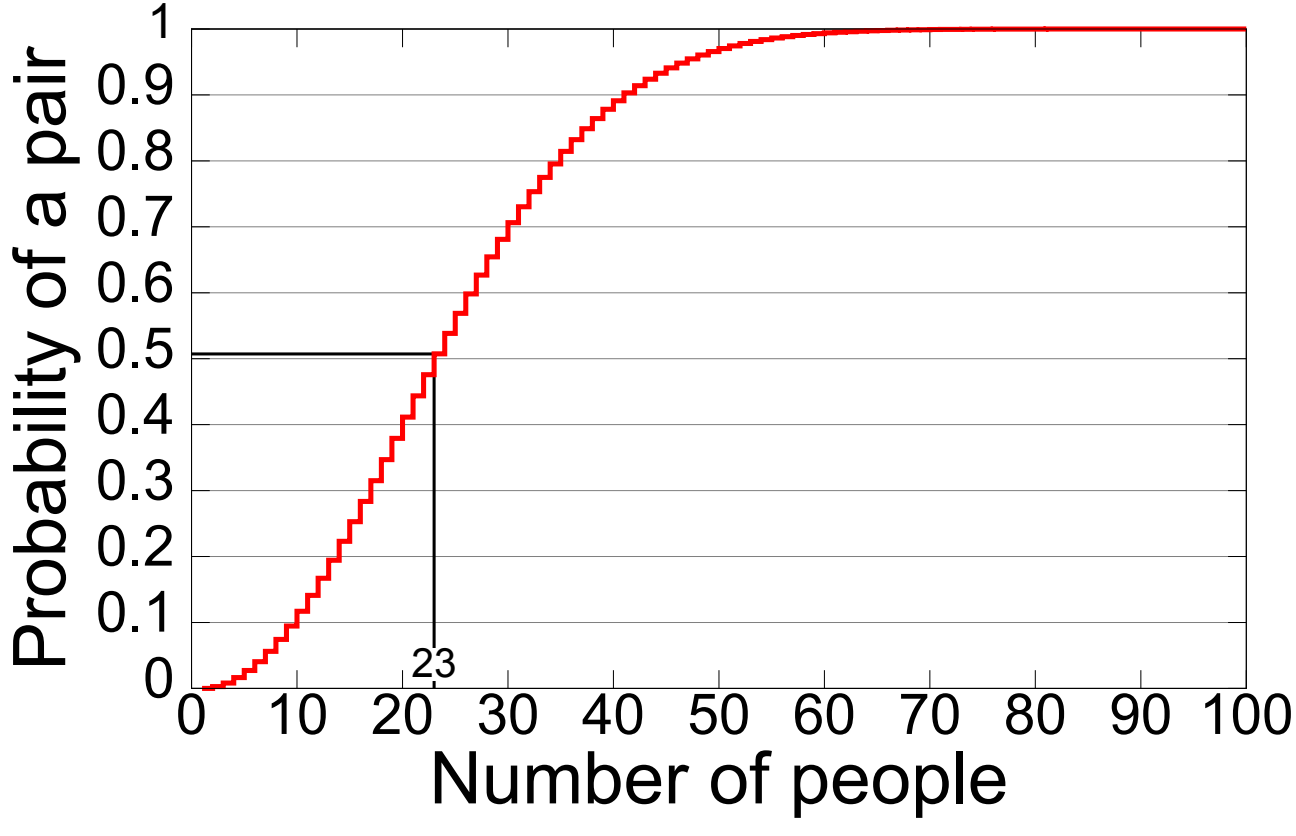
Once in a million events happen all the time

The birthday paradox

How many people should be in a room for P[shared birthday] > 0.5?

The birthday paradox

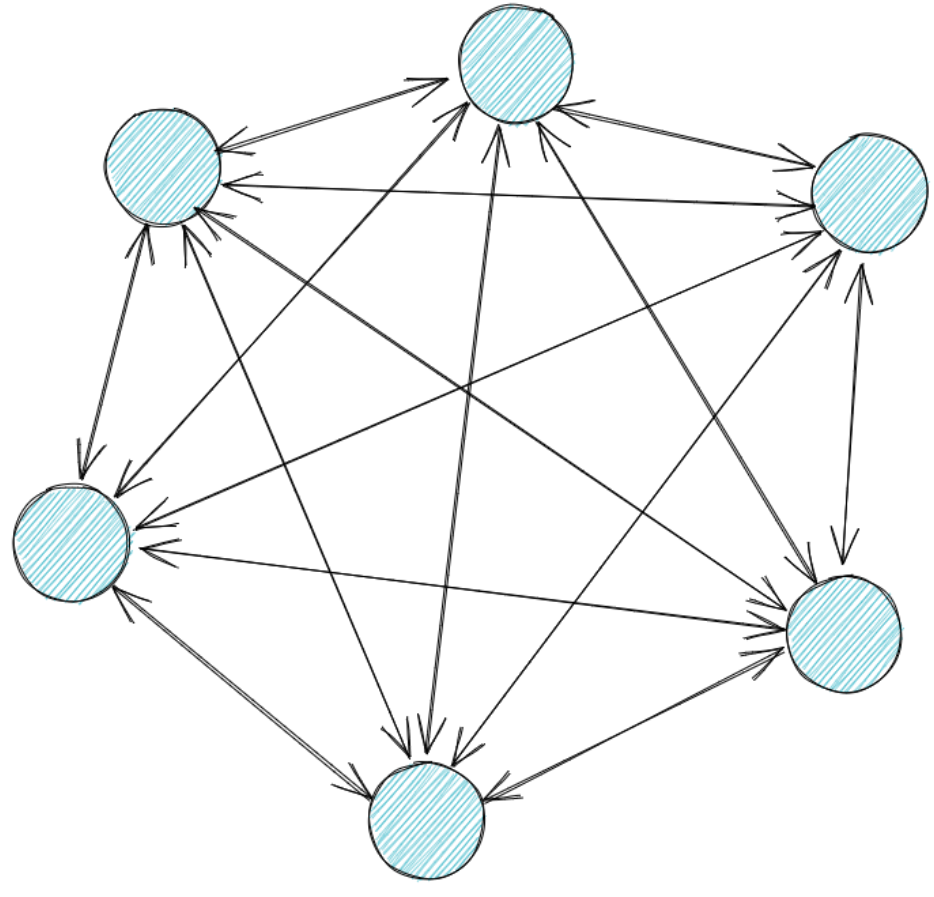
How many people should be in a room for P[shared birthday] > 0.5?



Volume scales faster than surface

Connections $\propto \mathcal{O}(n^2)$

Subgroups $\propto \mathcal{O}(2^n)$



Emergent behavior

When do grains of sand become a heap?

Let’s play a game

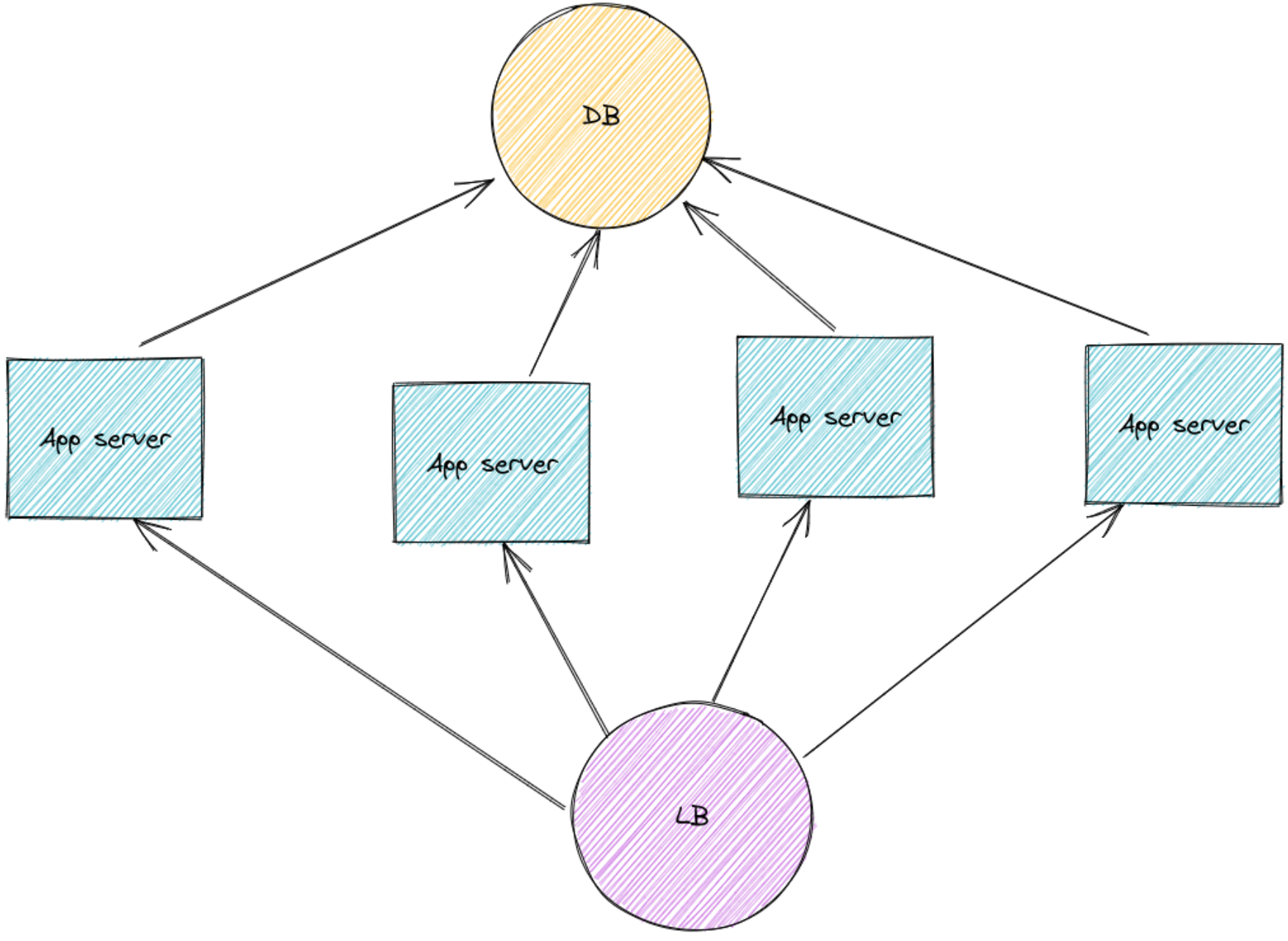
1. Choose a number between 1 and 5, call that X
2. Wait until you hear hand clapping
3. Clap your hands X times
4. Wait X seconds
5. Go back to #2

When do re-mirrors become a storm?

Emergent behavior

- Aggregate impact
- Interactions of elements dominate
- Non-linear emergence


STATE? IN A
STATELESS SYSTEM??
INCONCEIVABLE!
imgflip.com

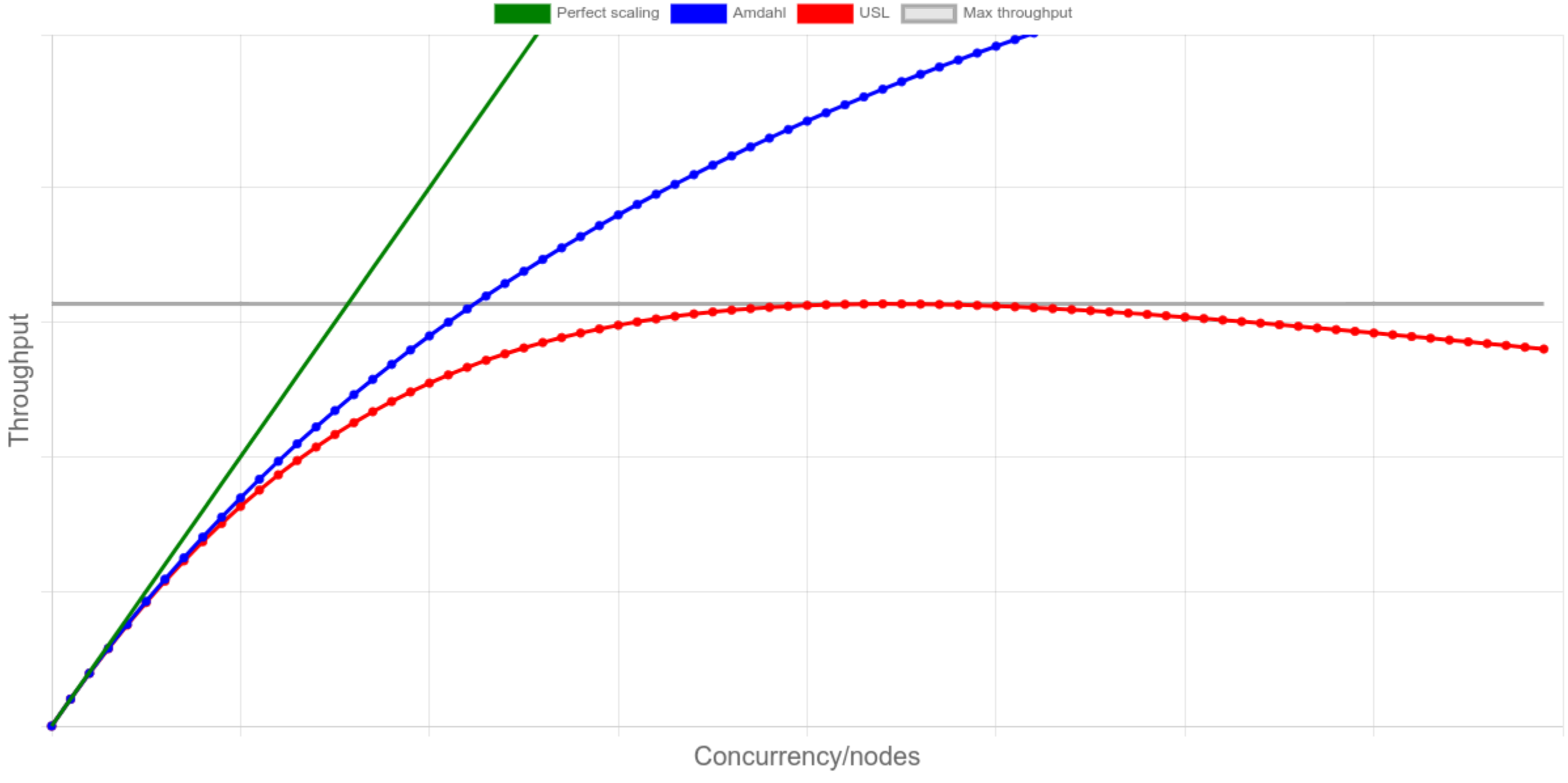

DB
App server
App server
App server
App server
LB

Emergence of state

- Interactions *are* state
- Super linear scaling
- Propagation time increases with scale

All large systems are essentially stateful

The Universal Scalability Law



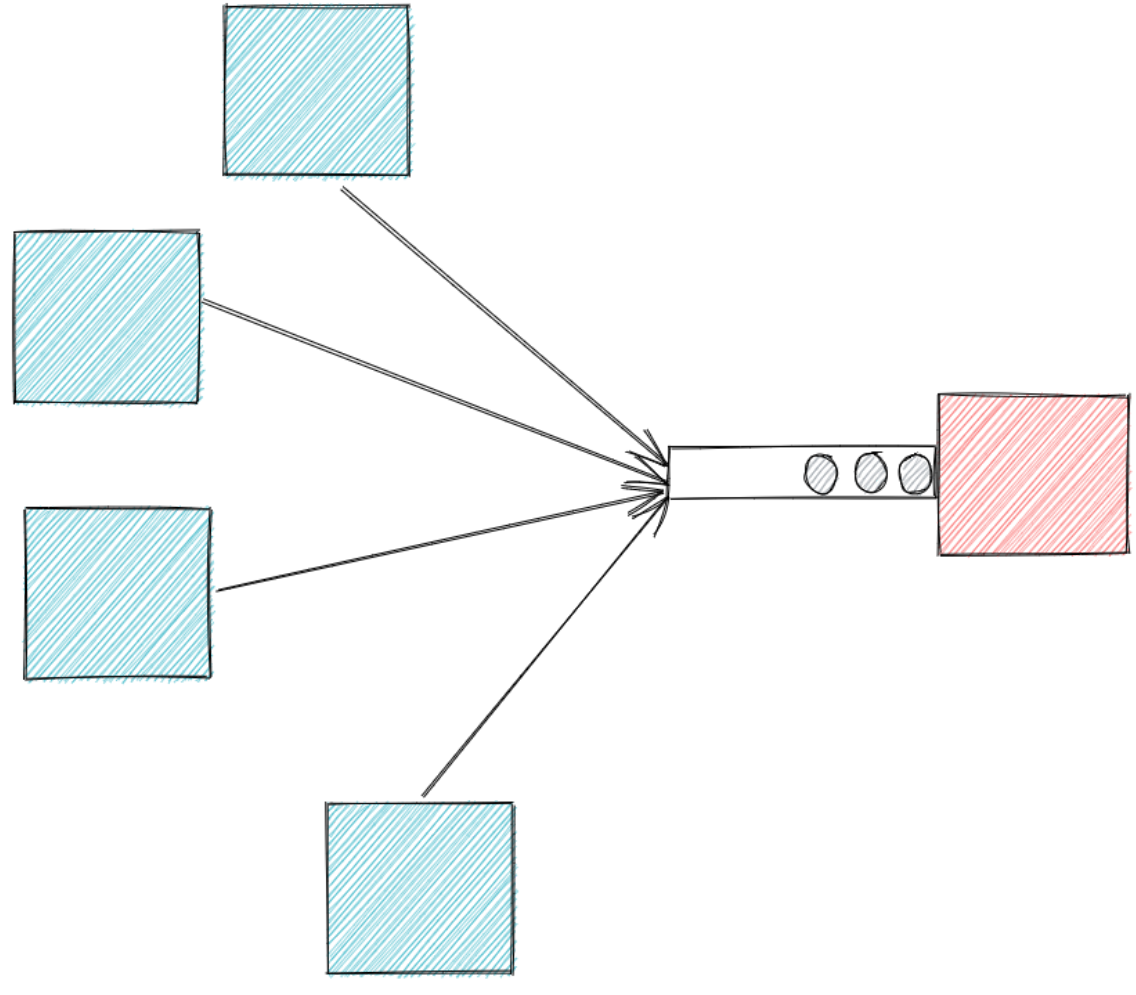
The Universal Scalability Law

$$X(N) = \frac{\gamma N}{1 + \alpha(N-1) + \beta N(N-1)}$$

- α - Contention; queueing for shared resource
- β - Consistency; Coordination between processes
- γ - Relative scale parameter

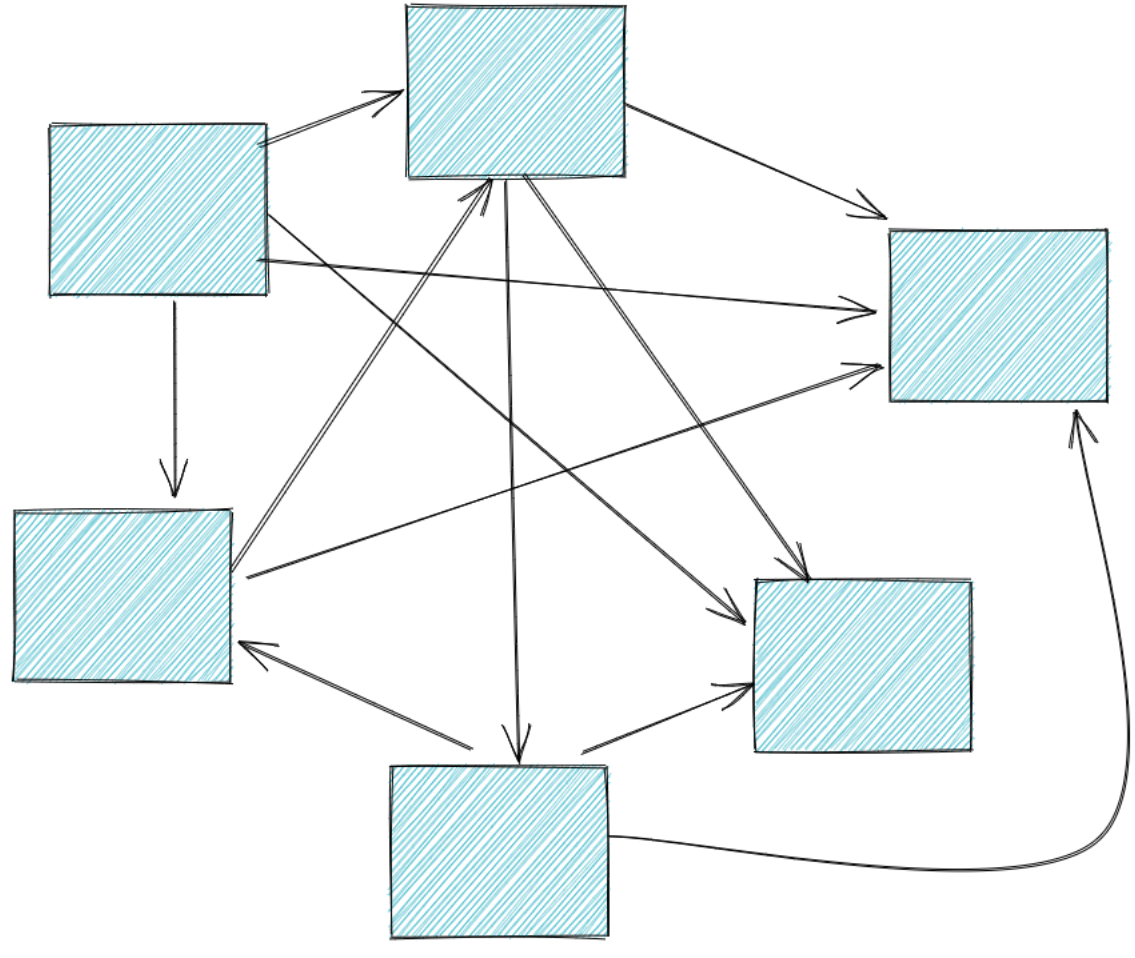
α - Contention

- Waiting for shared resource
- Queueing
- Limited by shared resource

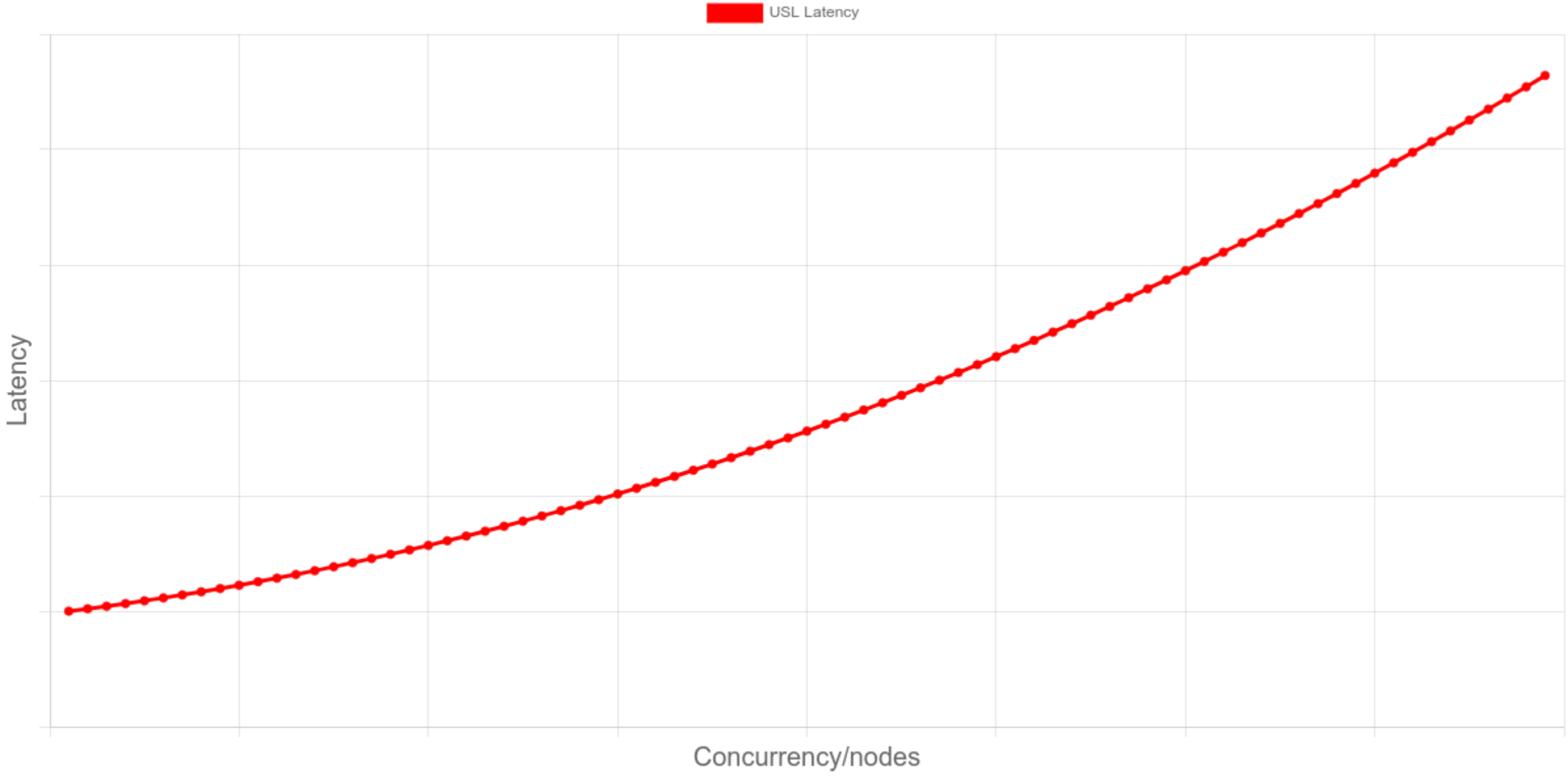


β - Consistency

- Coordination between processes
- Processes wait for each other
- Limited by any process



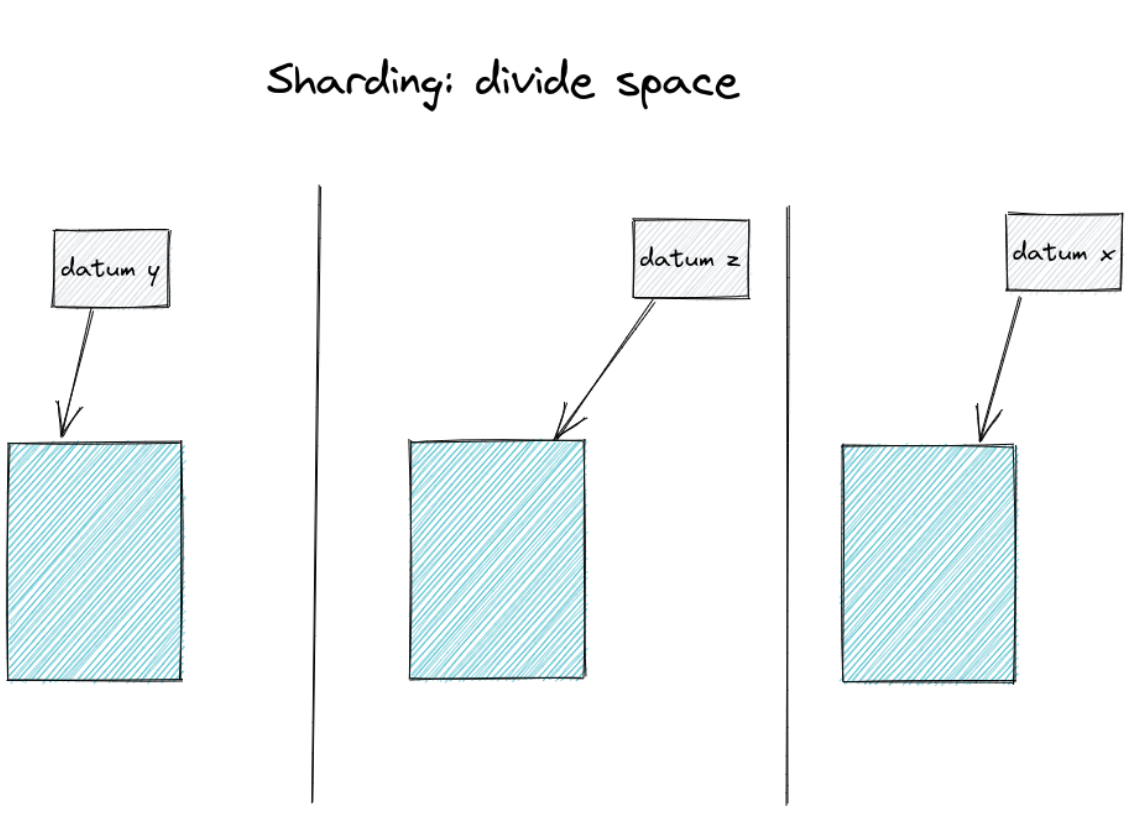
What about latency?

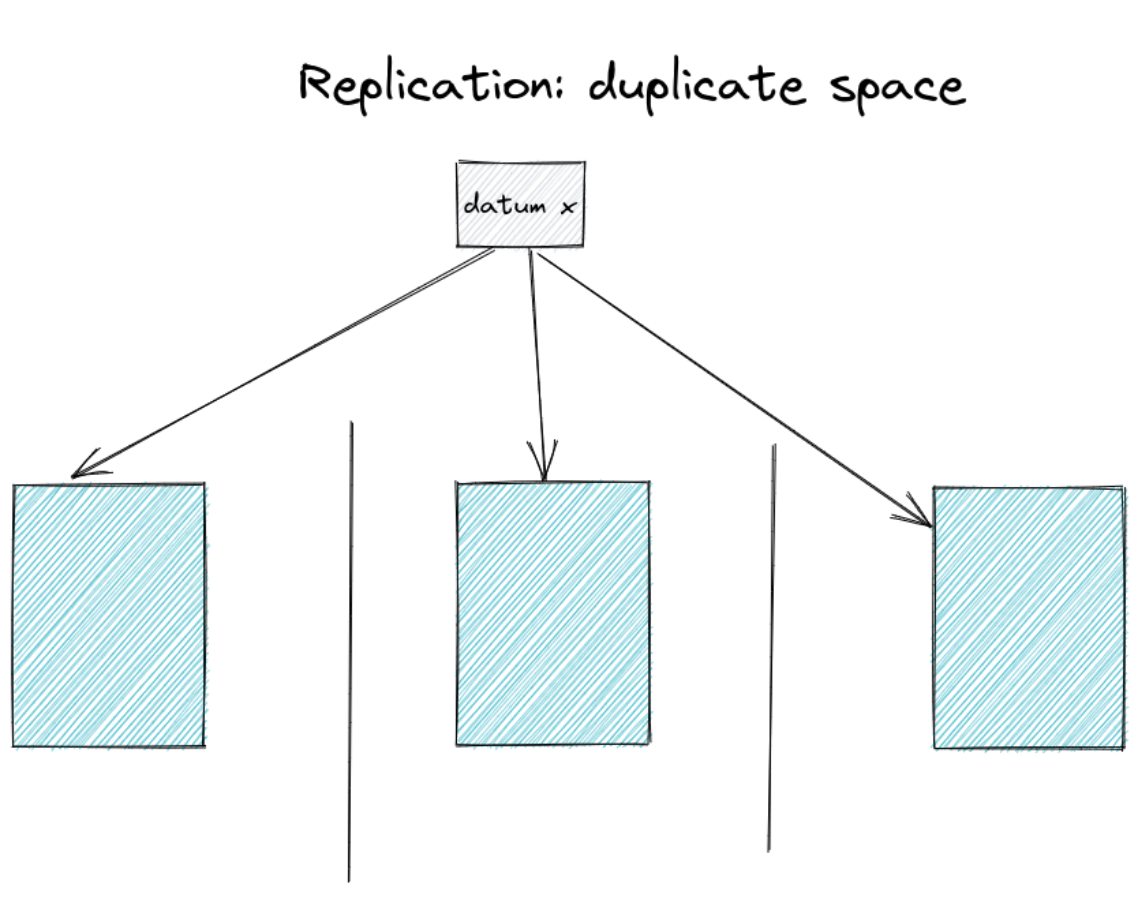


How do we scale things?

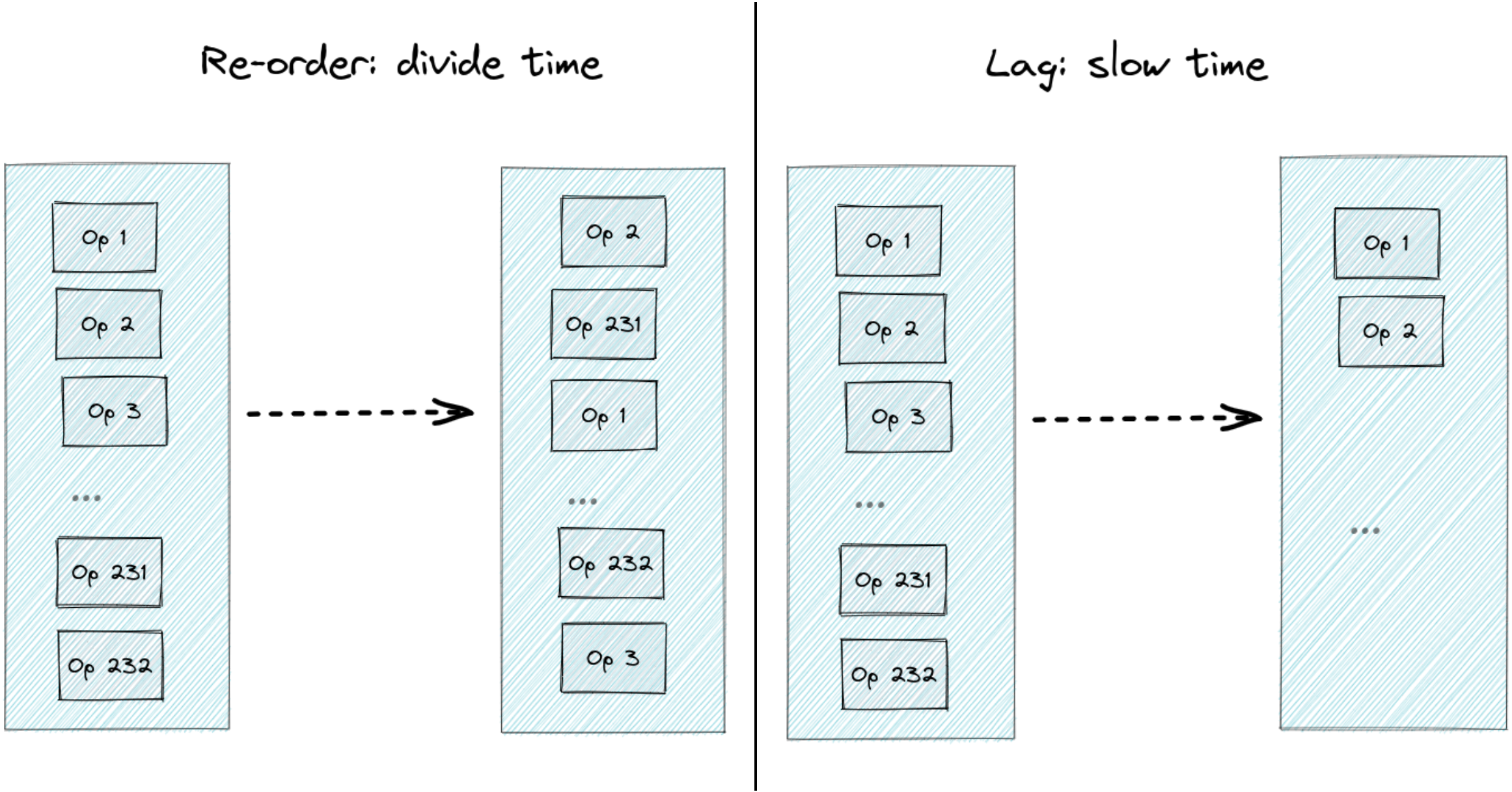
By warping space and time!

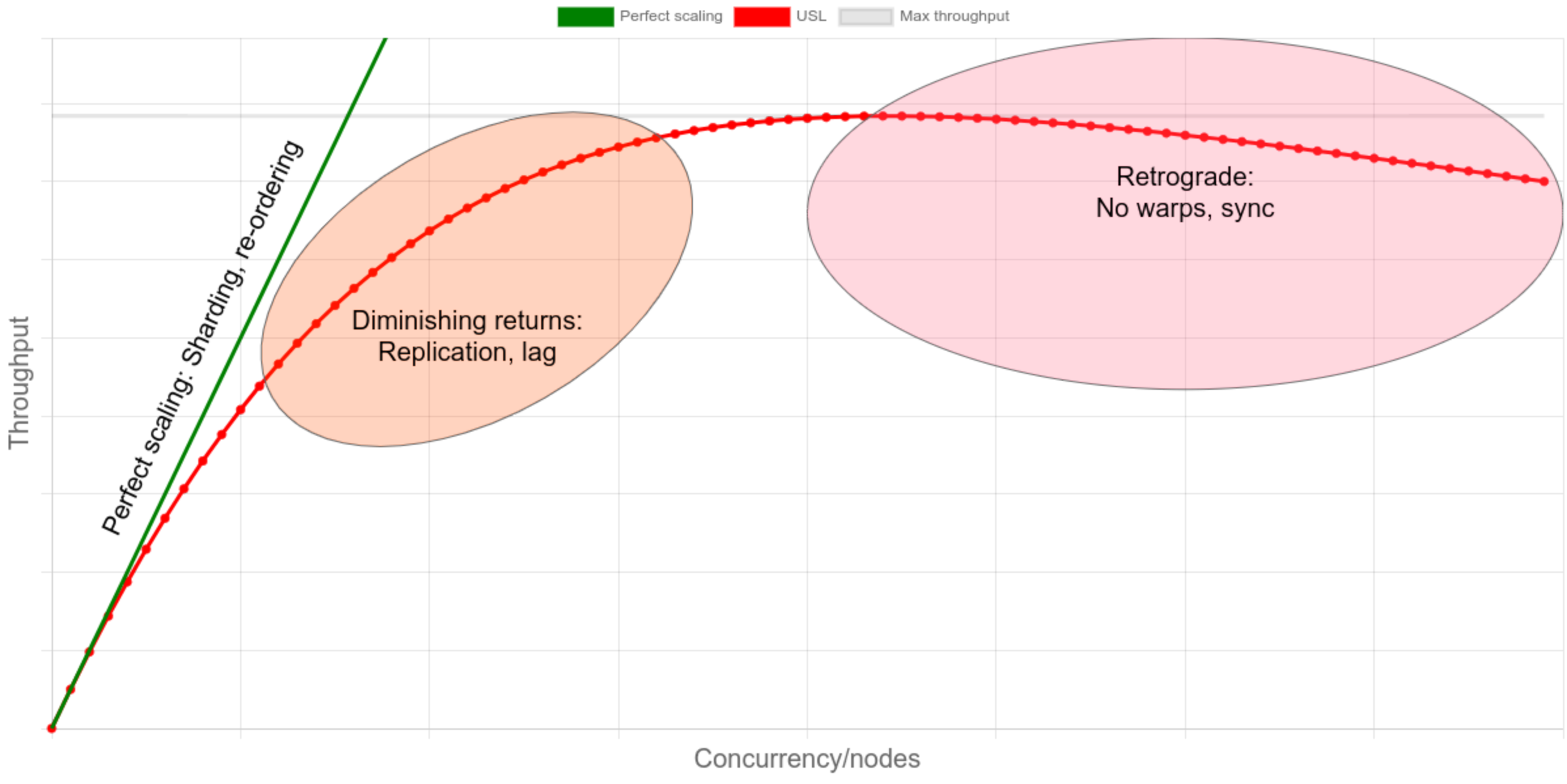
Space warp


Sharding: divide space
datum y
datum z
datum x

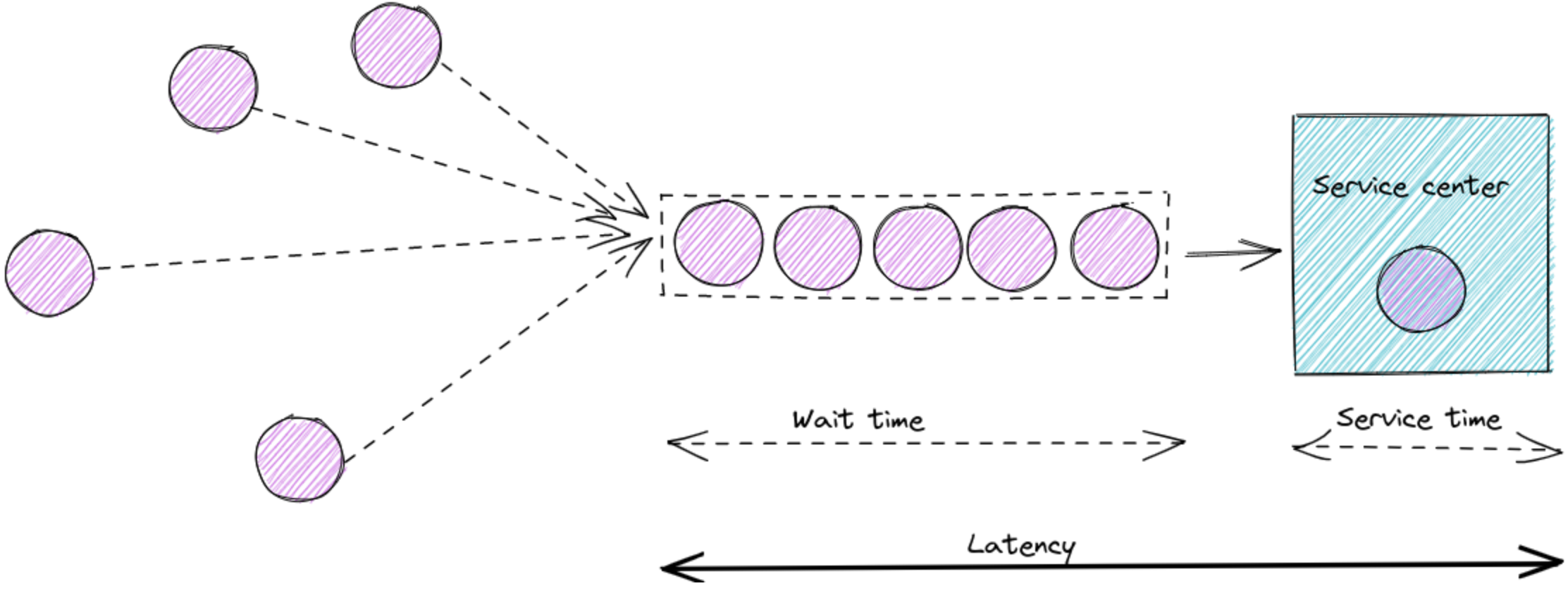

Replication: duplicate space
datum x

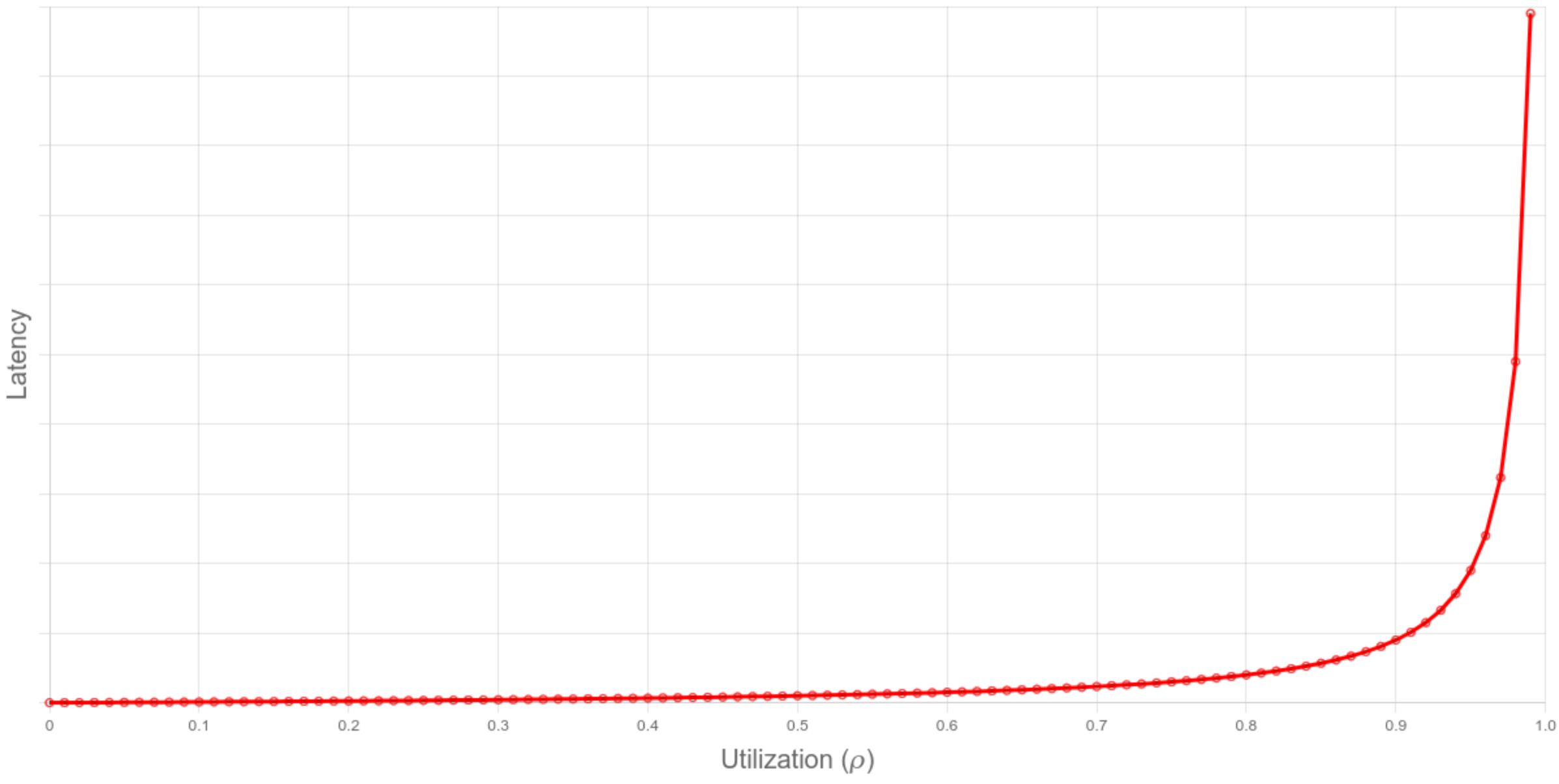
Time warp


Re-order: divide time
Op 1
Op 2
Op 3
...
Op 231
Op 232
Op 2
Op 231
Op 1
...
Op 232
Op 3
Lag: slow time
Op 1
Op 2
Op 3
...
Op 231
Op 232
Op 1
Op 2
...


Perfect scaling
USL
Max throughput
Perfect scaling: Sharding, re-ordering
Diminishing returns:
Replication, lag
Retrograde:
No warps, sync
Throughput
Concurrency/nodes

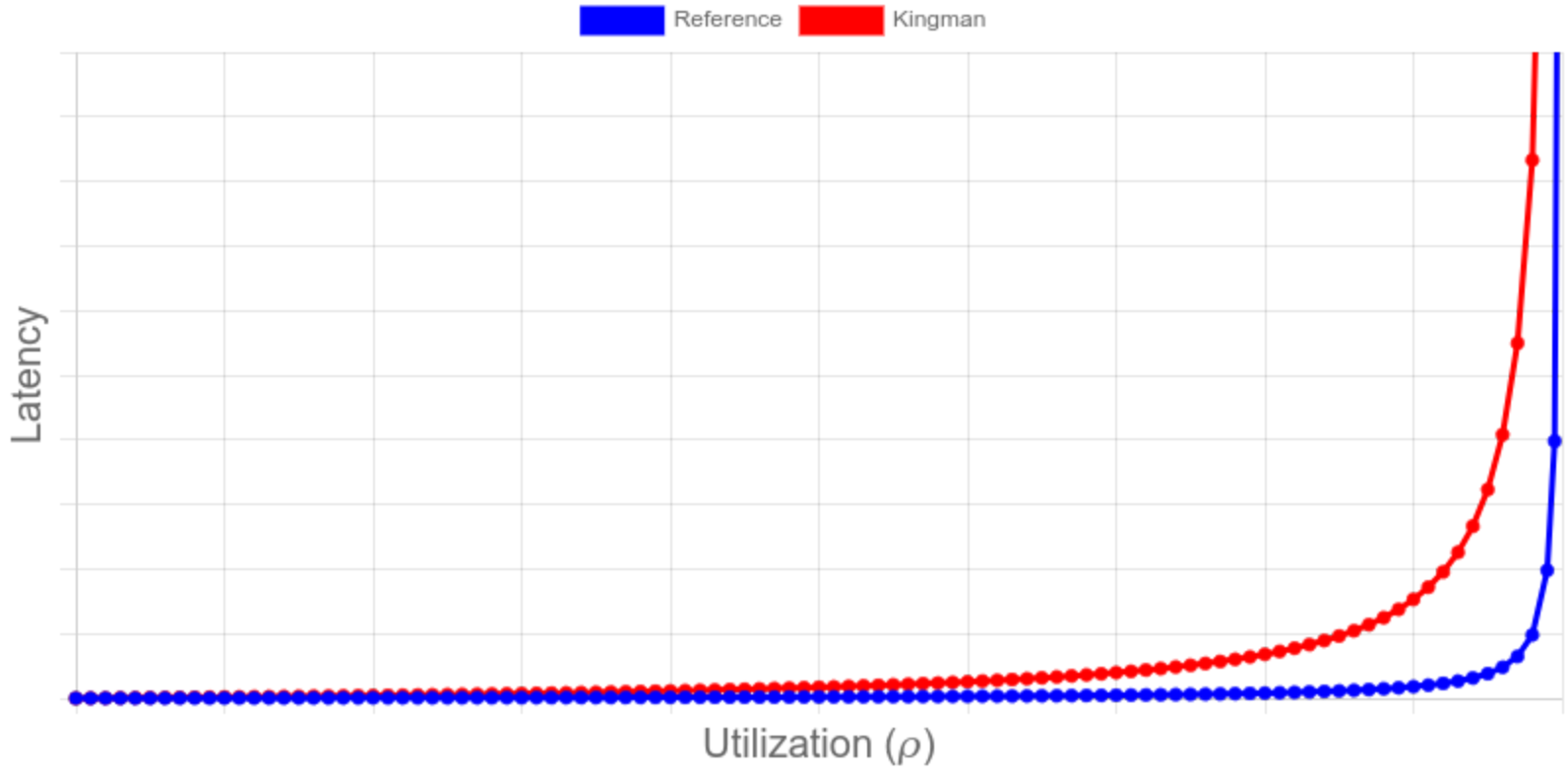
Queue theory crash course





$$W \propto \frac{\rho}{1-\rho}$$

Variance



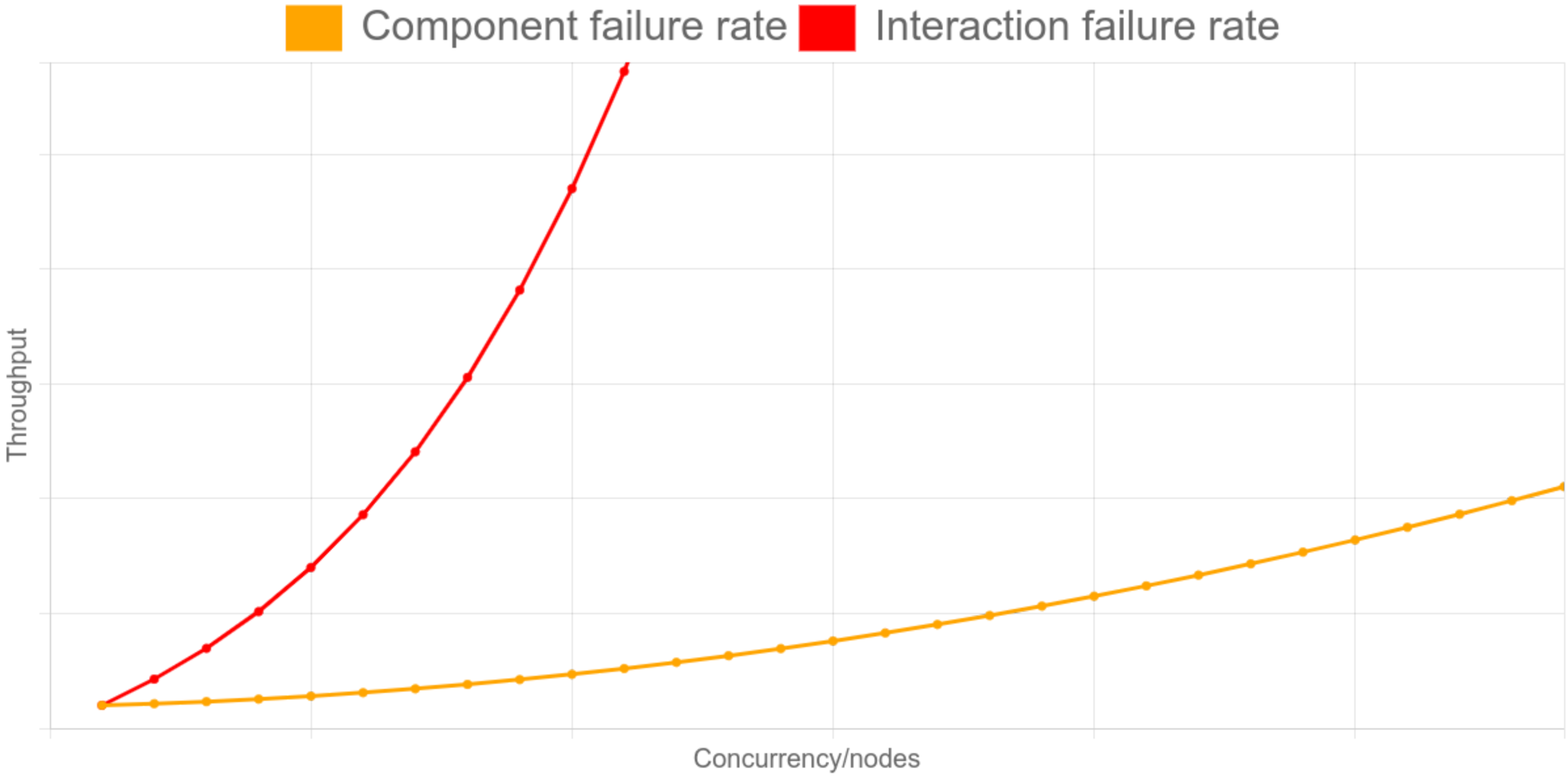
$$W \propto \frac{\rho}{1-\rho} \frac{C_s^2 + C_a^2}{2}$$

#FailAtScale

Component failure | Interaction failure

#FailAtScale

Component failure	Interaction failure
independent → linear scaling	dependent → super linear scaling


Component failure rate
Interaction failure rate
Throughput
Concurrency/nodes

#FailAtScale

- Statistical failures
- Latency grows → timeouts
- Failure demand (retries)

Go forth and scale

- Lower the variance, raise the mean
- Avoid coordination
- Warp time and space
- Reduce statistical failures

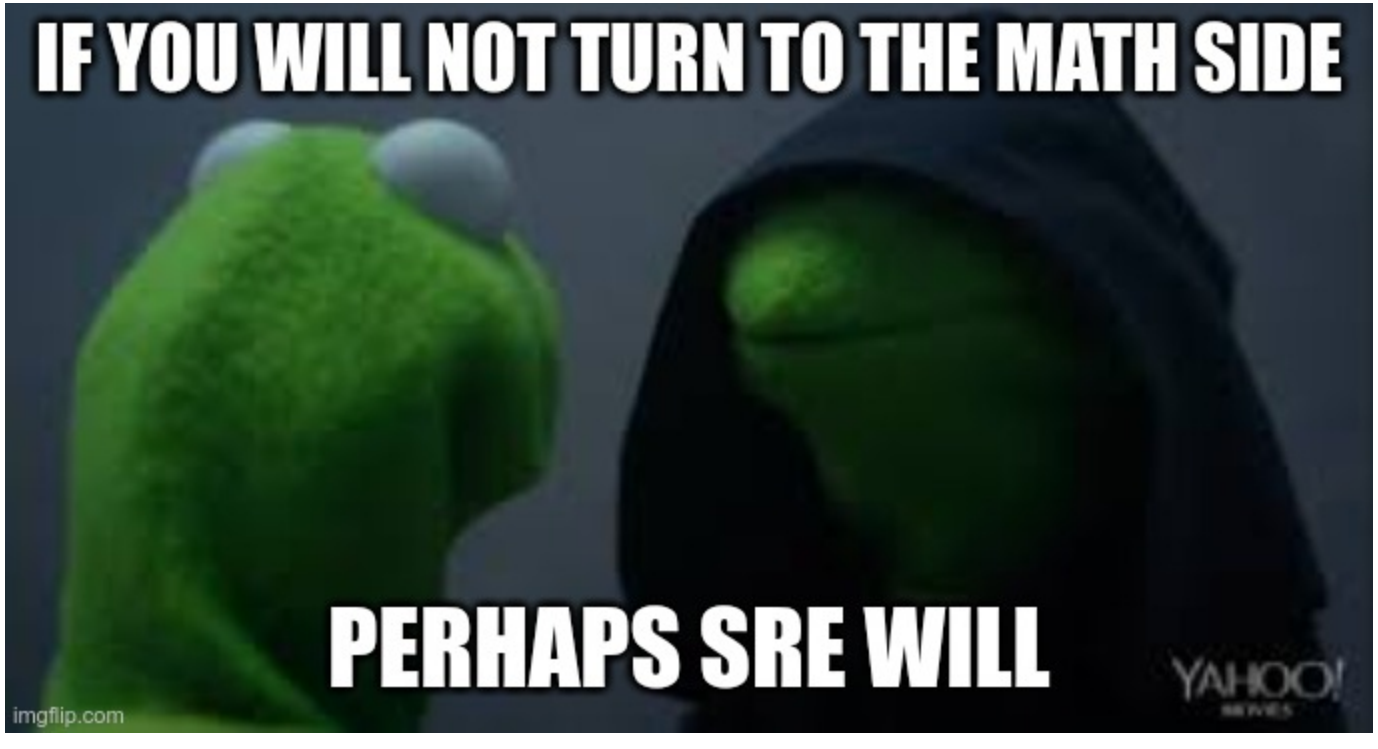
Quality is key to Scaling

”Quality” → less rework, uniformity

What have we learned?

- Math helps us think
- Models reveal scaling challenges

QED


IF YOU WILL NOT TURN TO THE MATH SIDE
PERHAPS SRE WILL
imgflip.com
YAHOO!