The Perils of Not Always Coordinating

Marc Brooker

Amazon Web Services

mbrooker@amazon.com

@marcjbrooker

“Can we avoid coordination more generally…? When?”

Hellerstein and Alvaro, “Keeping CALM: When Distributed Consistency is Easy”, 2019

Let’s build a block storage system!

- Highly Available
- Strongly consistent (linearizable reads and writes)
  - Compatible with NVMe
- <1ms latency
- Durability like a hard drive
  - ~0.5% AFR
- Scalable to any number of nodes

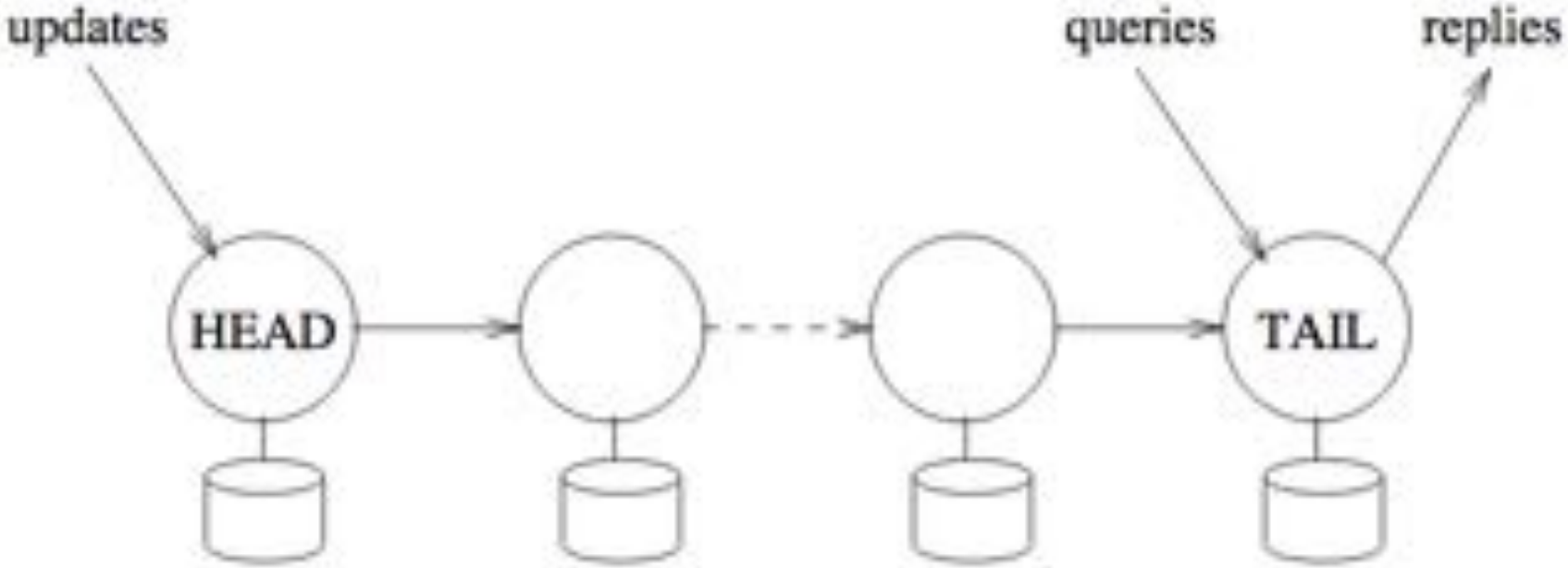
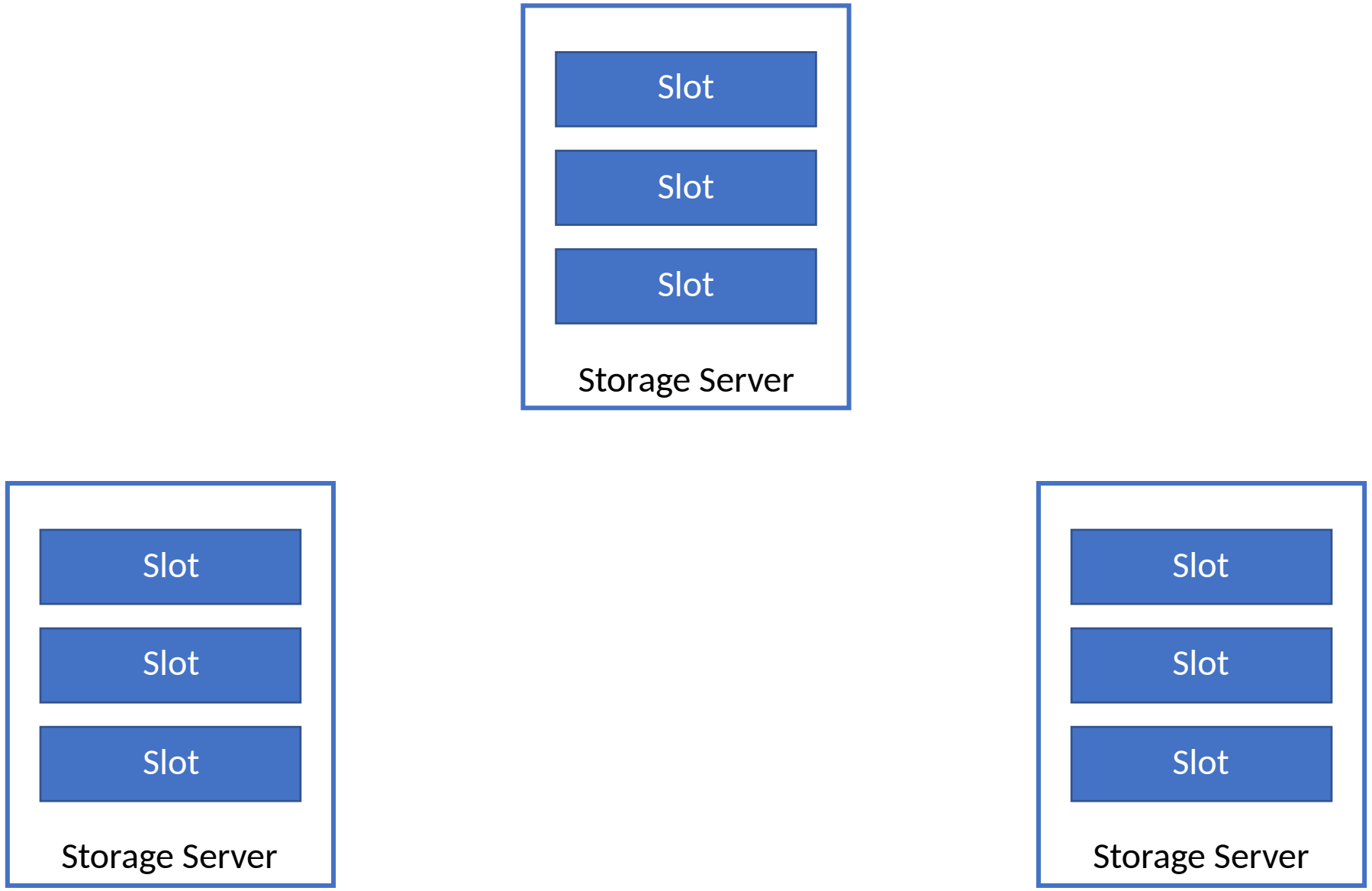
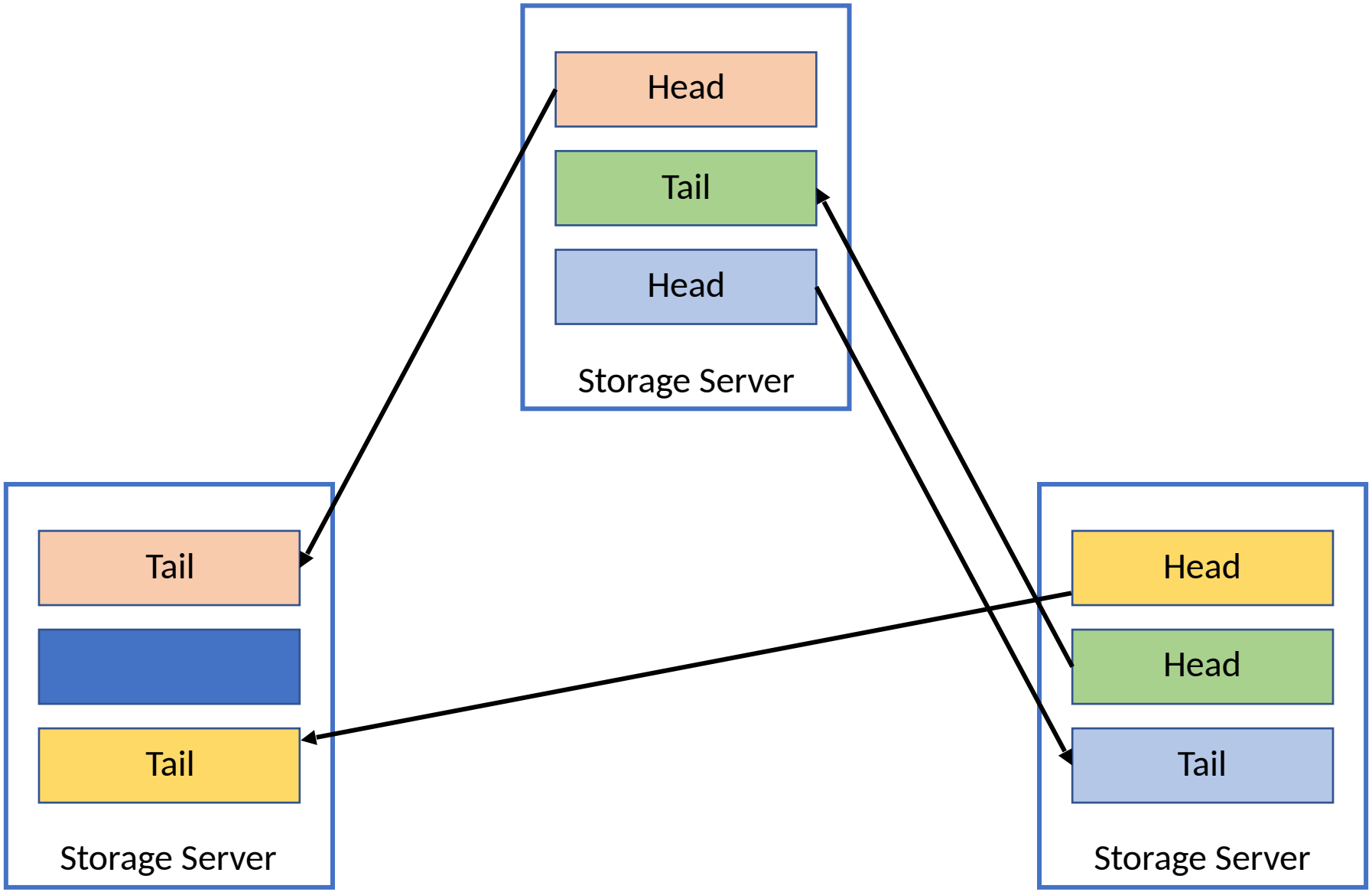


Figure 2: A chain.

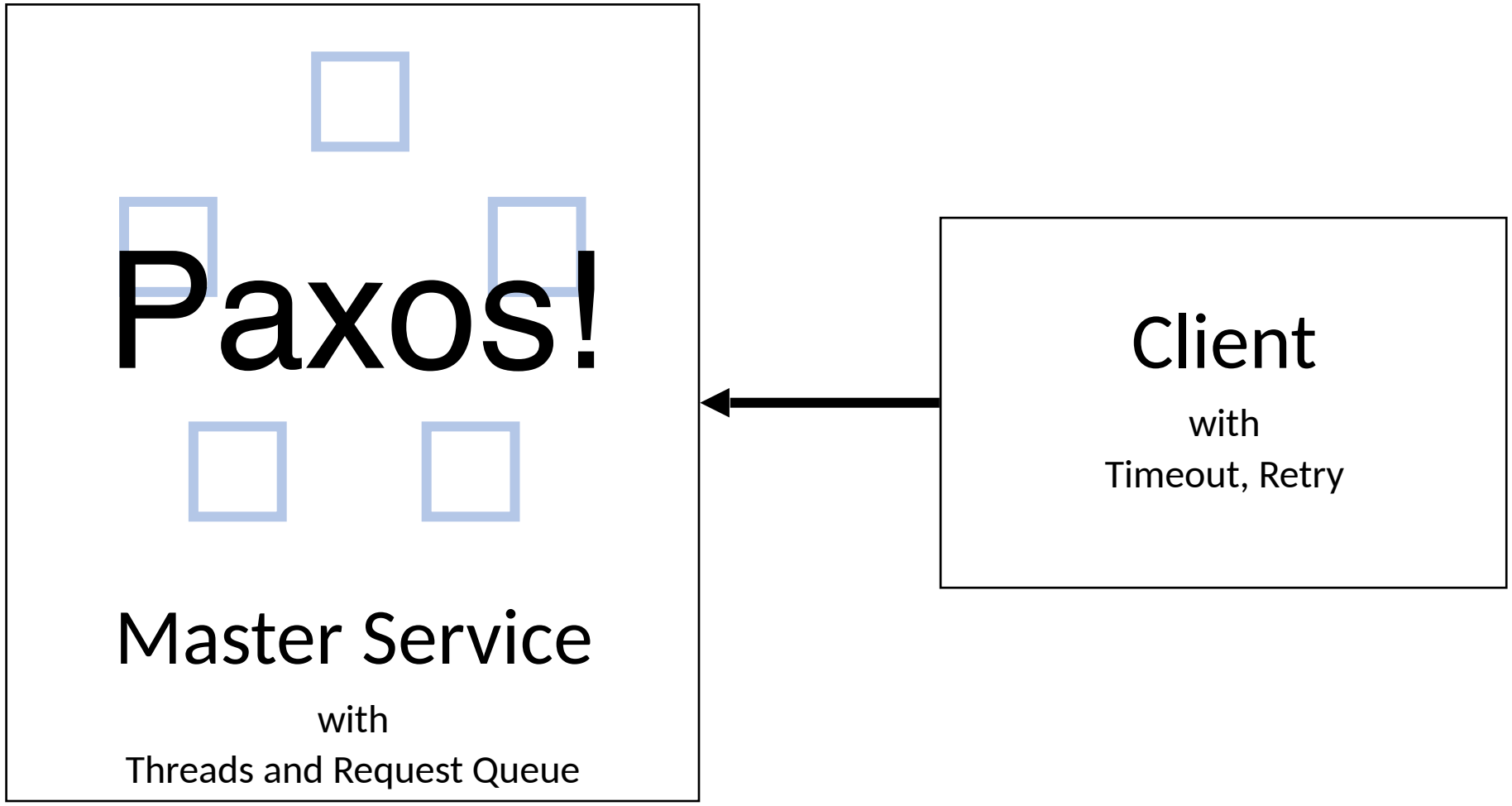
Van Renesse and Schneider, “Chain Replication for Supporting High Throughput and Availability”, OSDI’04

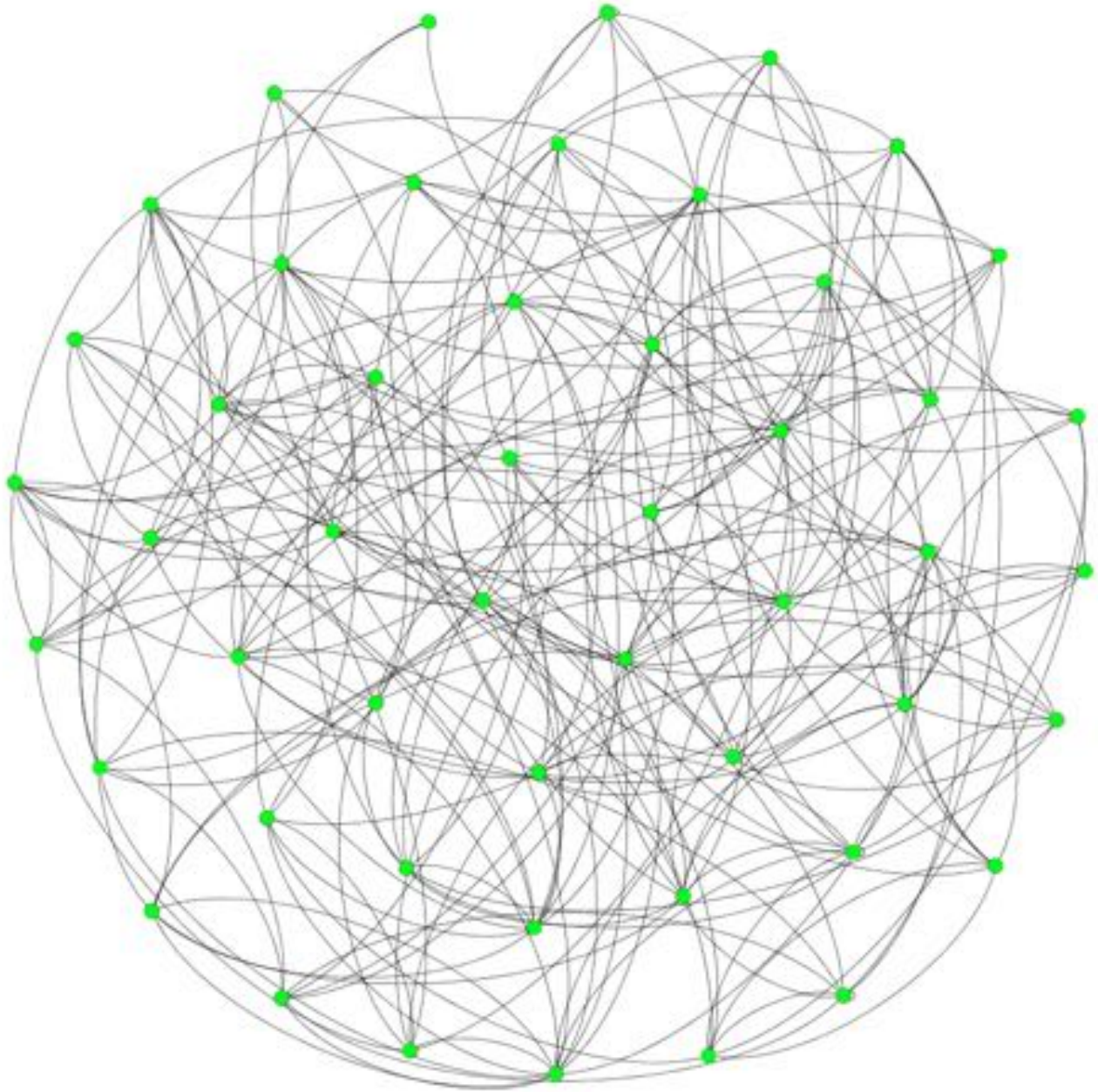

Slot
Slot
Slot
Storage Server
Slot
Slot
Slot
Storage Server
Slot
Slot
Slot
Storage Server

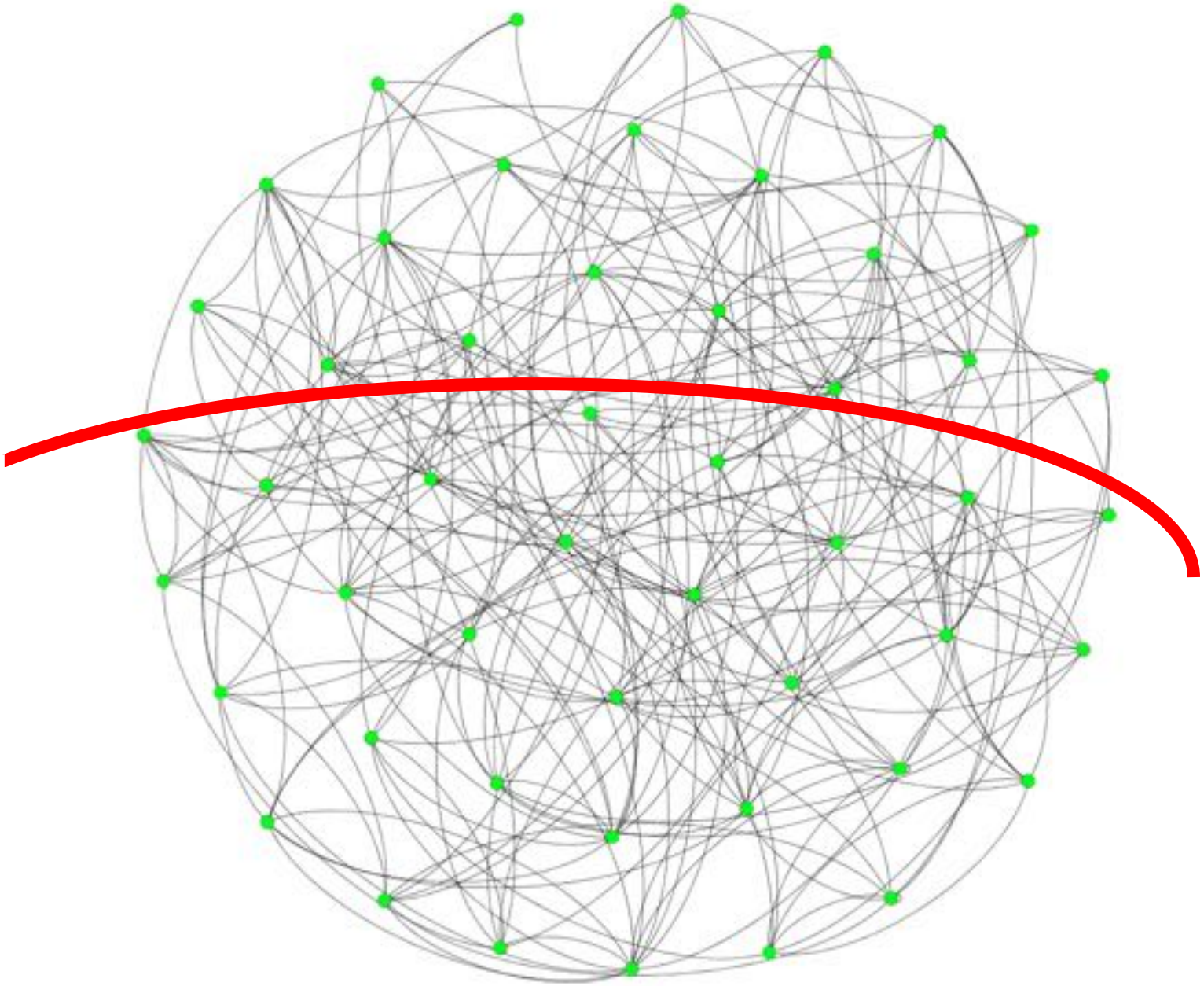

Head
Tail
Head
Storage Server
Tail
Tail
Storage Server
Head
Head
Tail
Storage Server

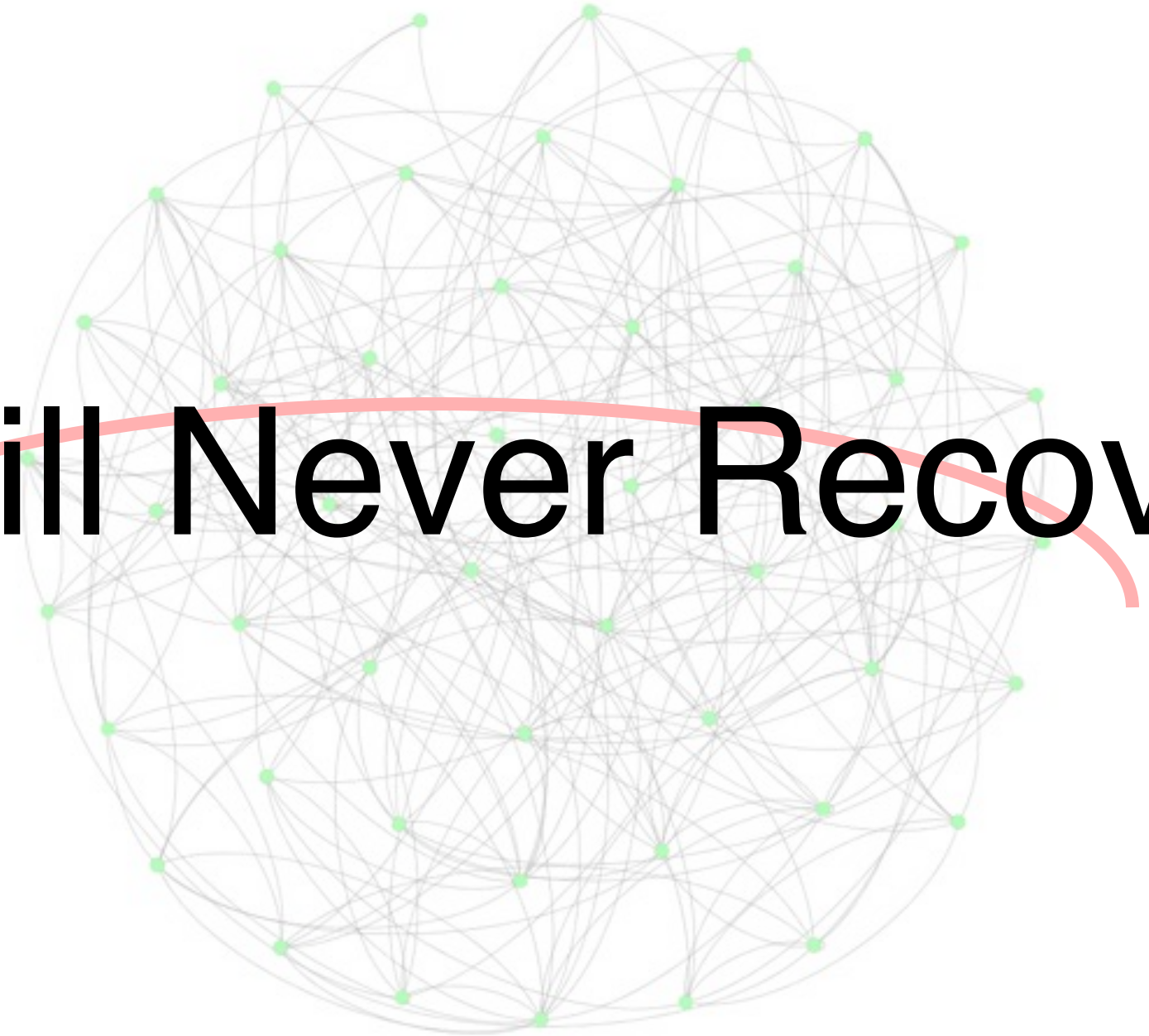
In response to detecting the failure of a server that is part of a chain…, the chain is reconfigured to eliminate the failed server. For this purpose, we employ a service, called the *master*.

Van Renesse and Schneider, “Chain Replication for Supporting High Throughput and Availability”, OSDI’04


Paxos!
Master Service
with
Threads and Request Queue
Client
with
Timeout, Retry

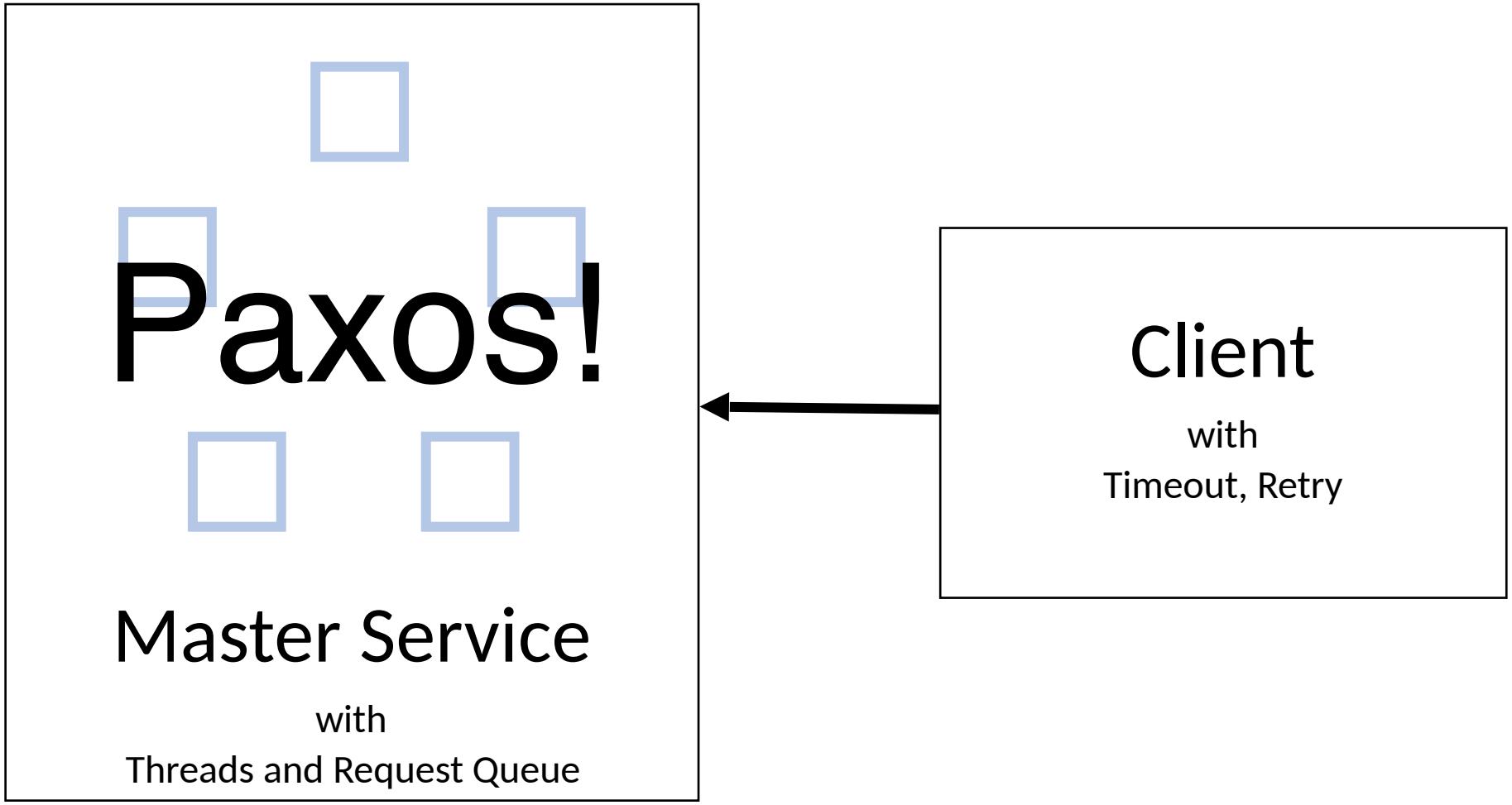


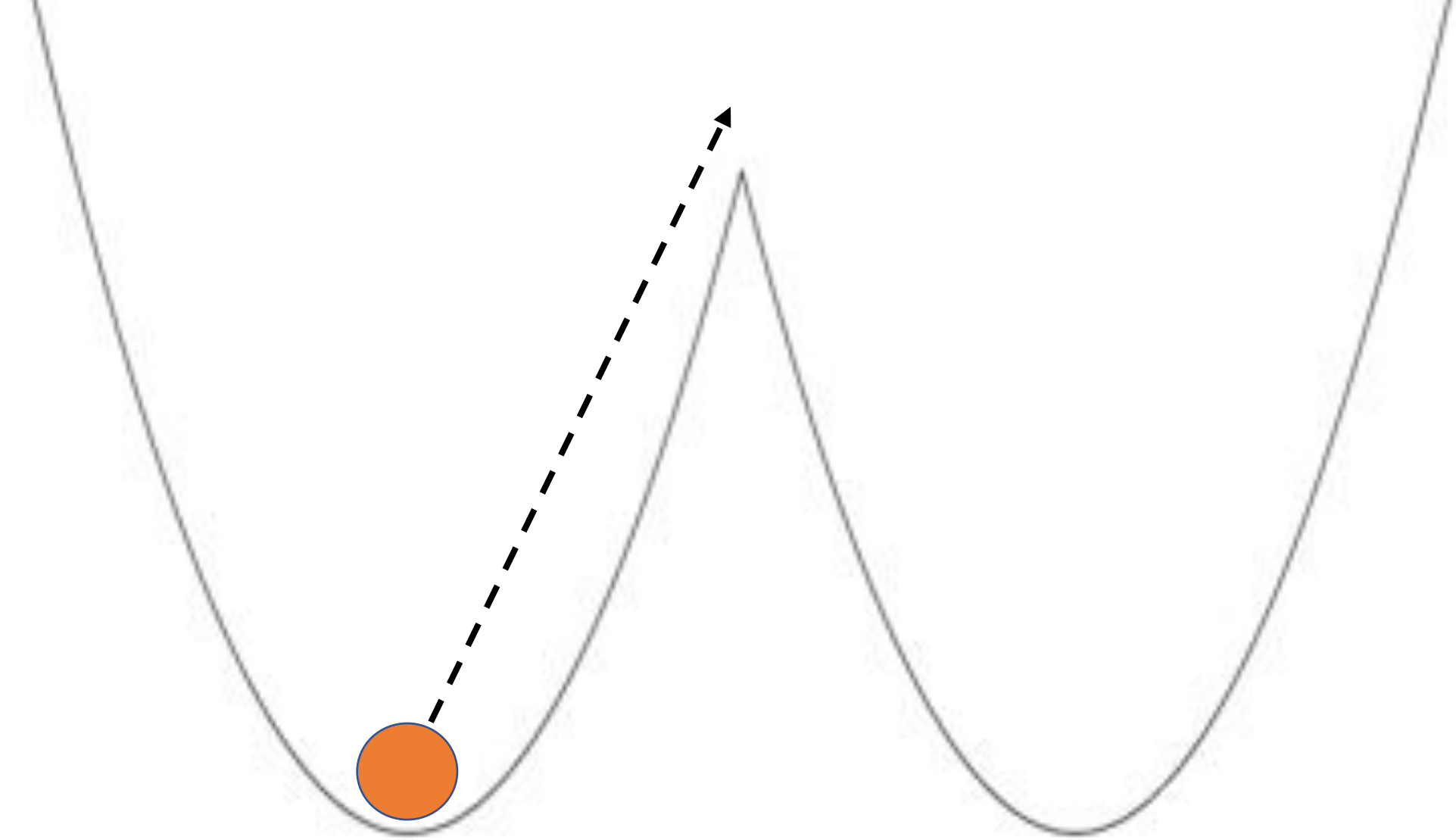




It Will Never Recover!

Scale Inversion





Available
Stable
State

Unavailable
Stable
State

“A liveness property is one which states that something must happen.”

– Leslie Lamport, from “Proving the Correctness of Multiprocess Programs”

Systems that *sometimes* coordinate.

When?

Can a whole lot of coordination happen at once?

Who?

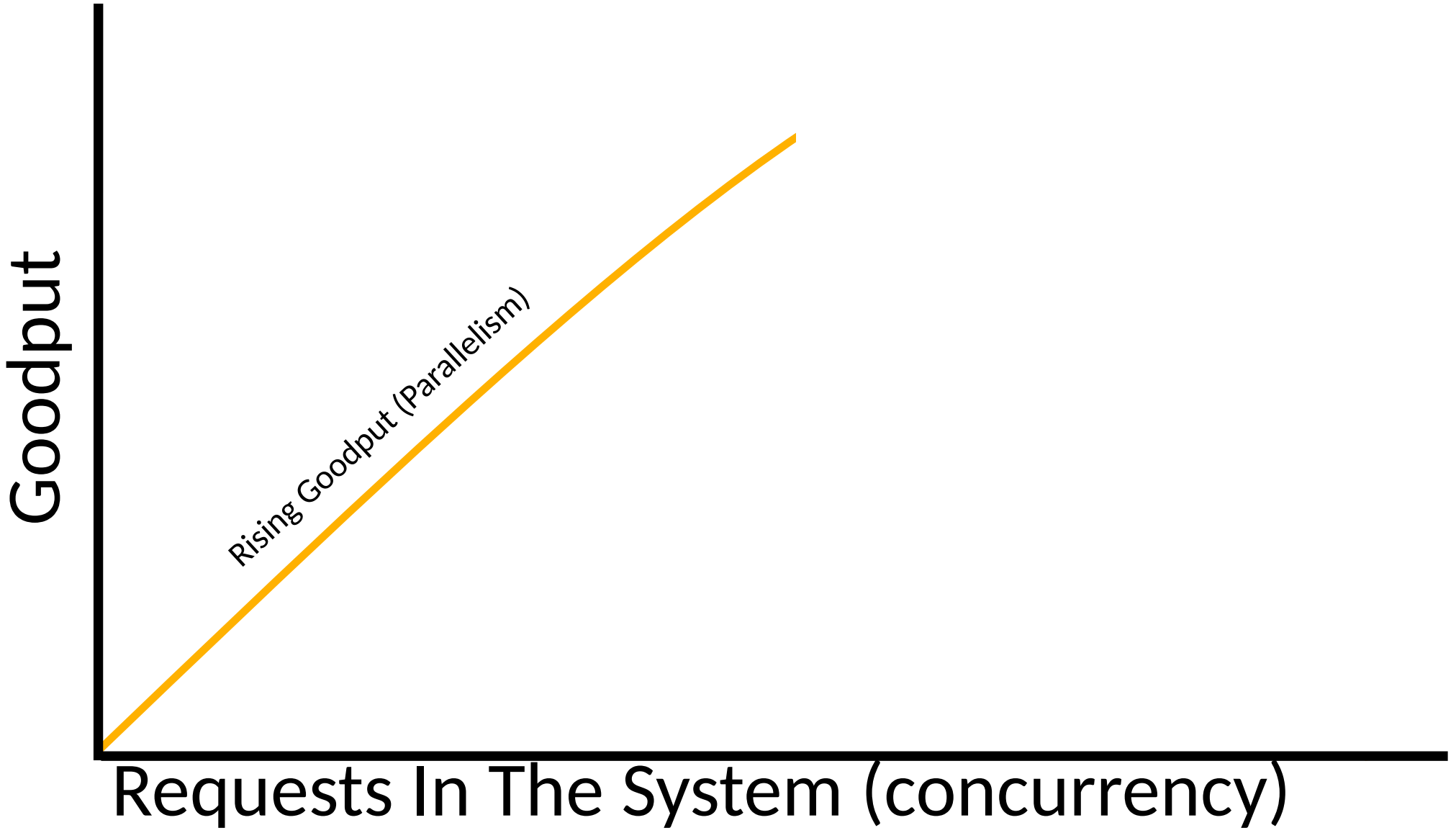
Can some workloads affect the performance of the whole system?

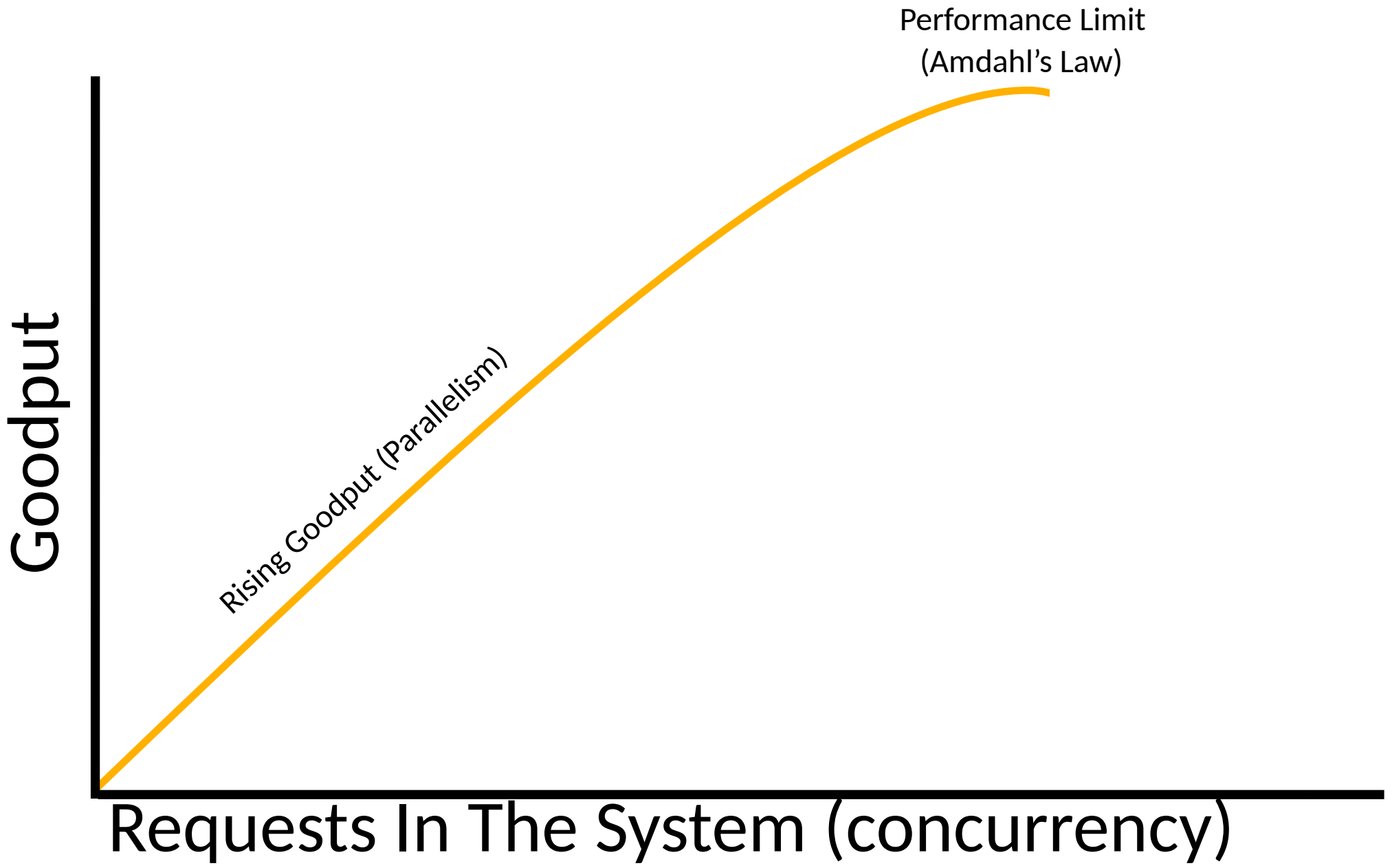
How Much?

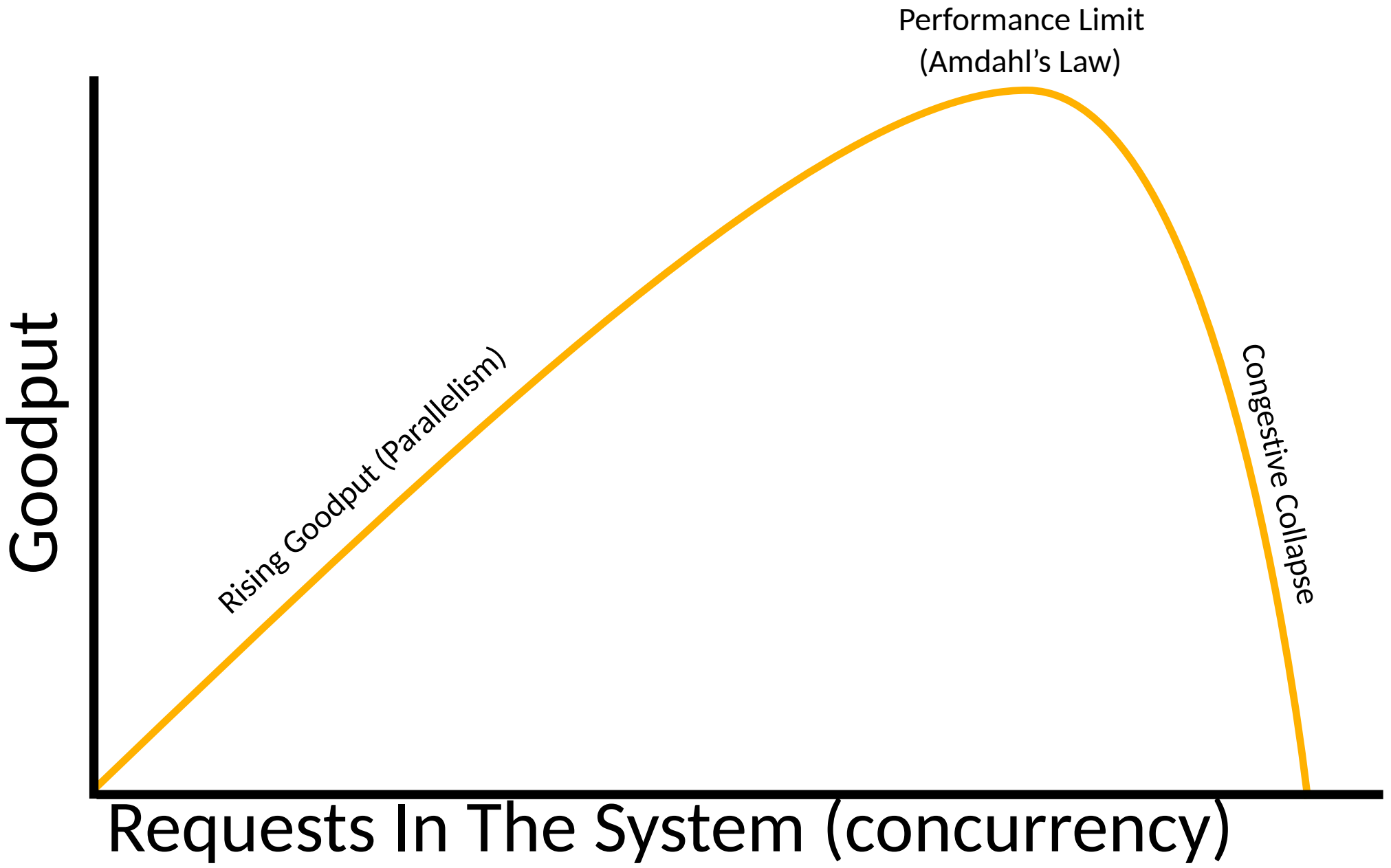
What's the worst-case amount of coordination that can happen?

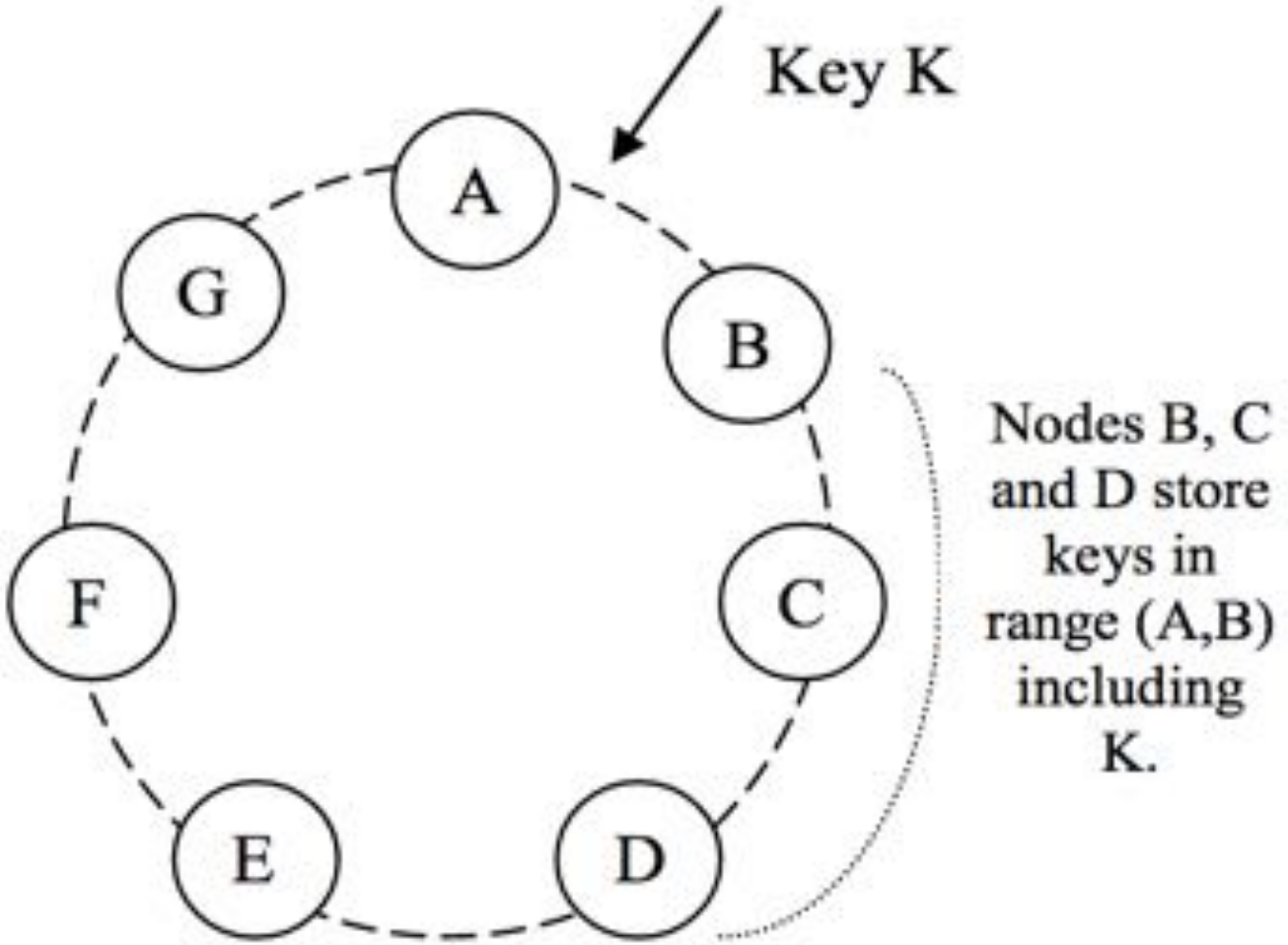
Instability?

Is there positive feedback?


Goodput
Rising Goodput (Parallelism)
Requests In The System (concurrency)

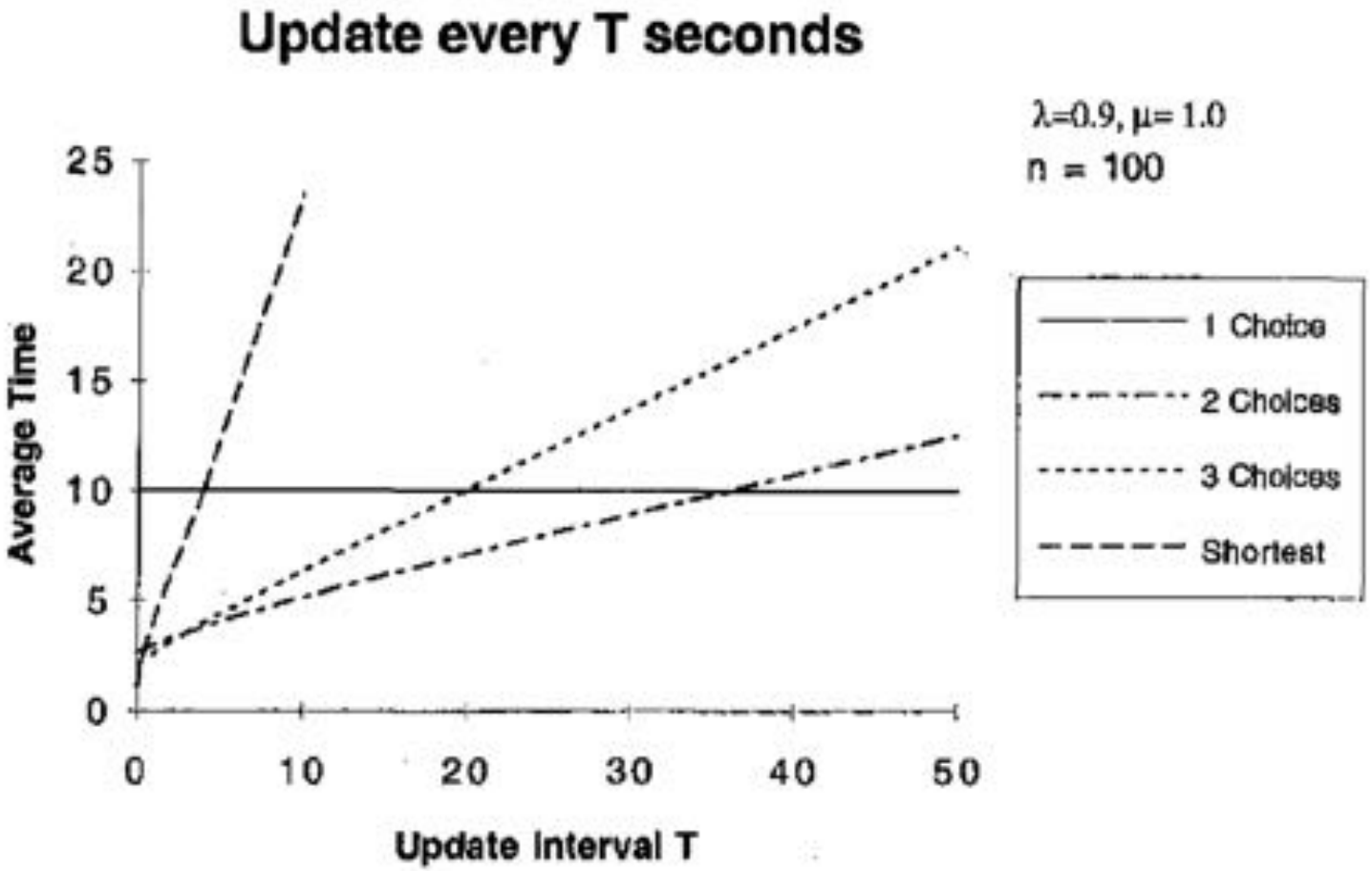

Performance Limit
(Amdahl's Law)
Goodput
Rising Goodput (Parallelism)
Requests In The System (concurrency)


Performance Limit
(Amdahl's Law)
Goodput
Rising Goodput (Parallelism)
Congestive Collapse
Requests In The System (concurrency)



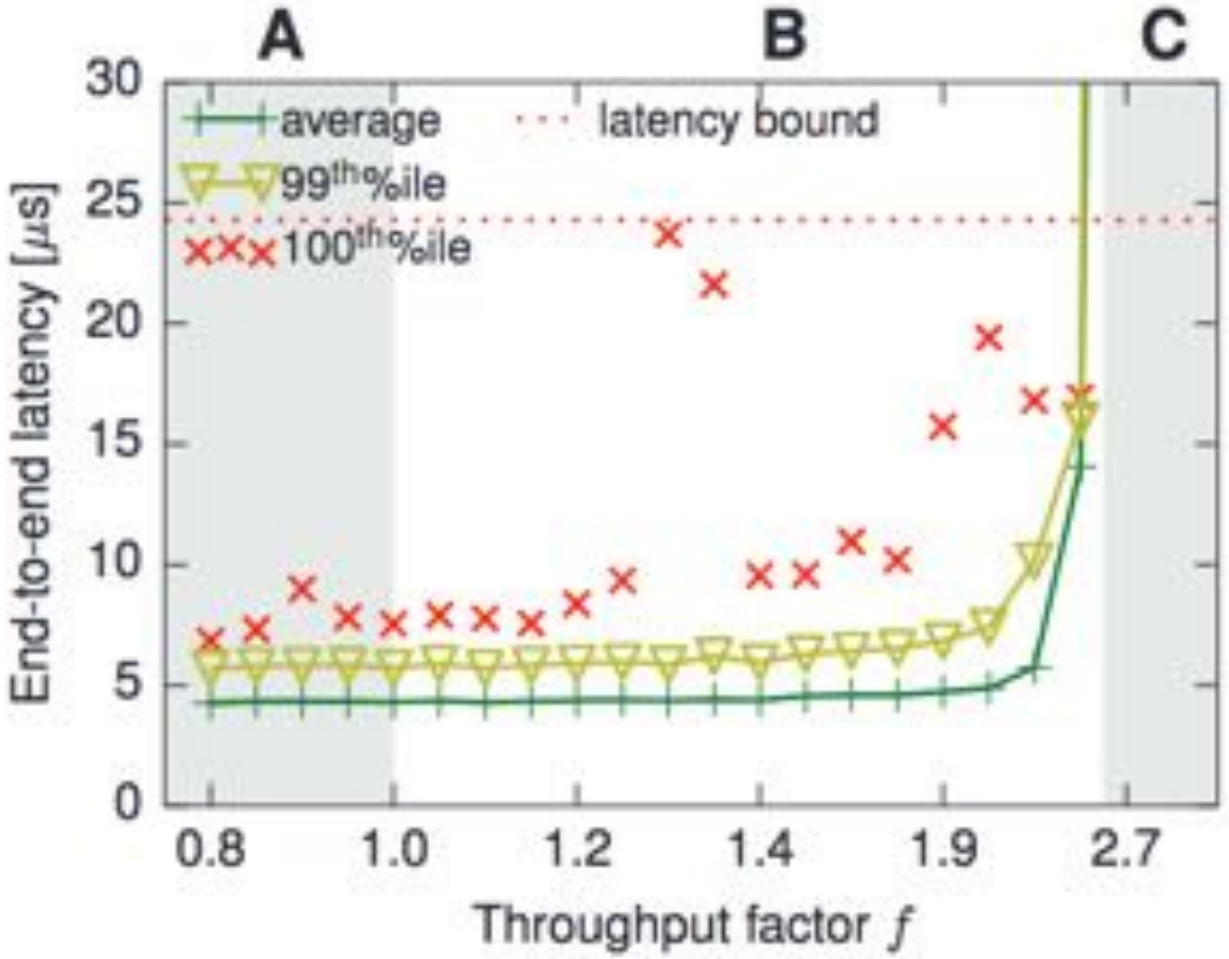
Recovering from overload adds load.

De Candia et al, “Dynamo: Amazon’s Highly Available Key-value Store”, SOSP’07



What if T depends on load?

Mitzenmacher, “How useful is old information?”, 2000



QoS delays the inevitable

Grosvenor et al, “Queues Don’t Matter When You Can JUMP Them!”, NSDI’15

A Partial Zoo of Partial Solutions

Congestion Control (e.g. TCP)

- Track record of success
- Strong theory
- Decades of development

But,

- Local, not global
- Load is a graph, not a line
- Tuning is complex
- What is fairness?

Queue Theory and Teletraffic Engineering

- Track record of success
- Strong theory
- Decades of development

But,

- Local, not global
- Load is a graph, not a line
- Tuning is complex
- Not all systems have a single owner

http://www.polytechphotos.dk [Public domain]

Backpressure, Throttling, Quotas, etc.

- Simple
- End-to-end (per client)

but

- “somebody else’s problem”
- What is fairness?
- Where is the optimal place to reject load?
- # clients could be large

Static Stability

- Simplify stability reasoning
- Very effective in practice

but,

- Not always achievable
- Move fast vs. stability

Constant Work

- Simple

but,

- Slow
- Inefficient
- May not always converge

Blast Radius Reduction

- Test to destruction
- Limit non-linear scale effects

but

- Increased cost?
- Limited maximum transaction scope

Chaos Testing

- Theoretically simple
- End-to-end

but

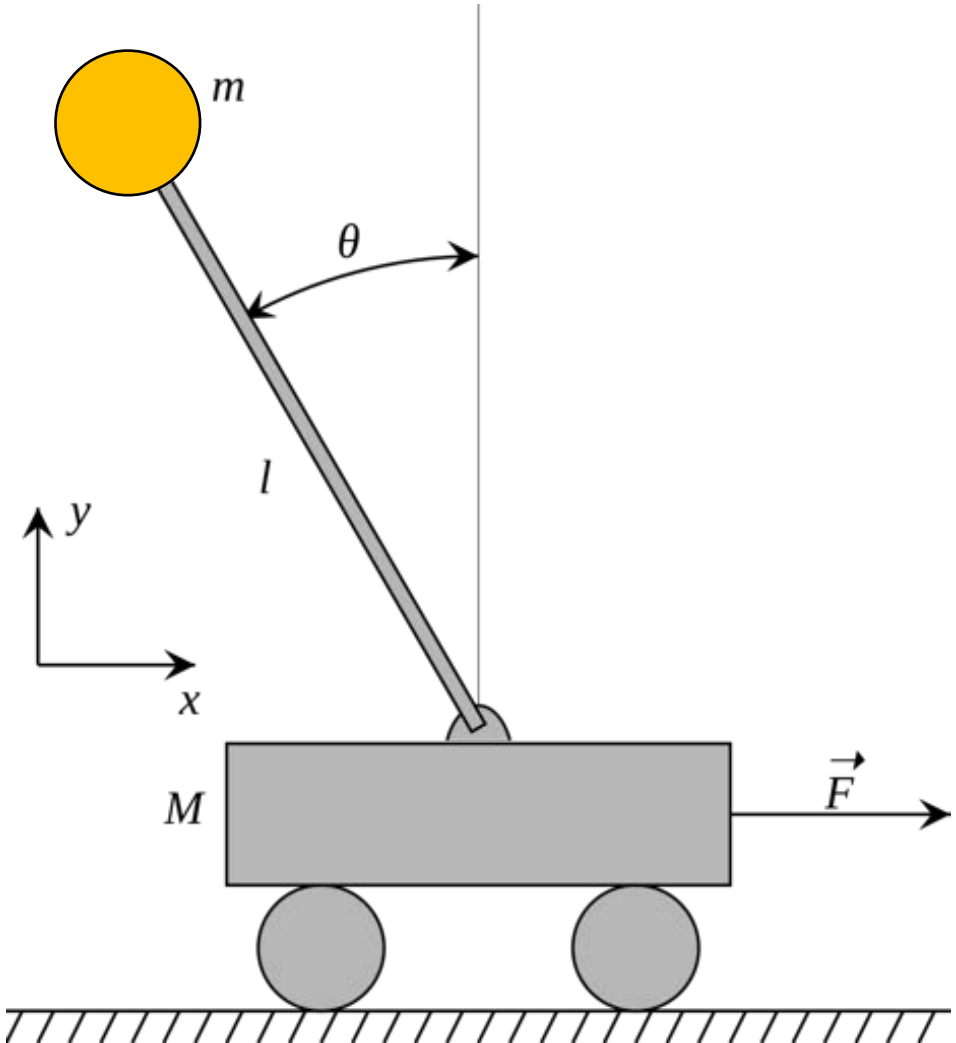
- Incomplete
- Infeasible at largest scales?

We’re doing OK, but not winning.

What do I want?

Possible Real Solutions

Control and Stability Theory



Simulation & Numerical Methods

Created with https://github.com/narimiran/double_pendulum

Stability is just as important as consistency and performance, but gets way less attention.