

## Motivation

- ▶ Eventual consistency not (always) enough
- ▶ State Machine Replication protocols like Paxos are not scalable

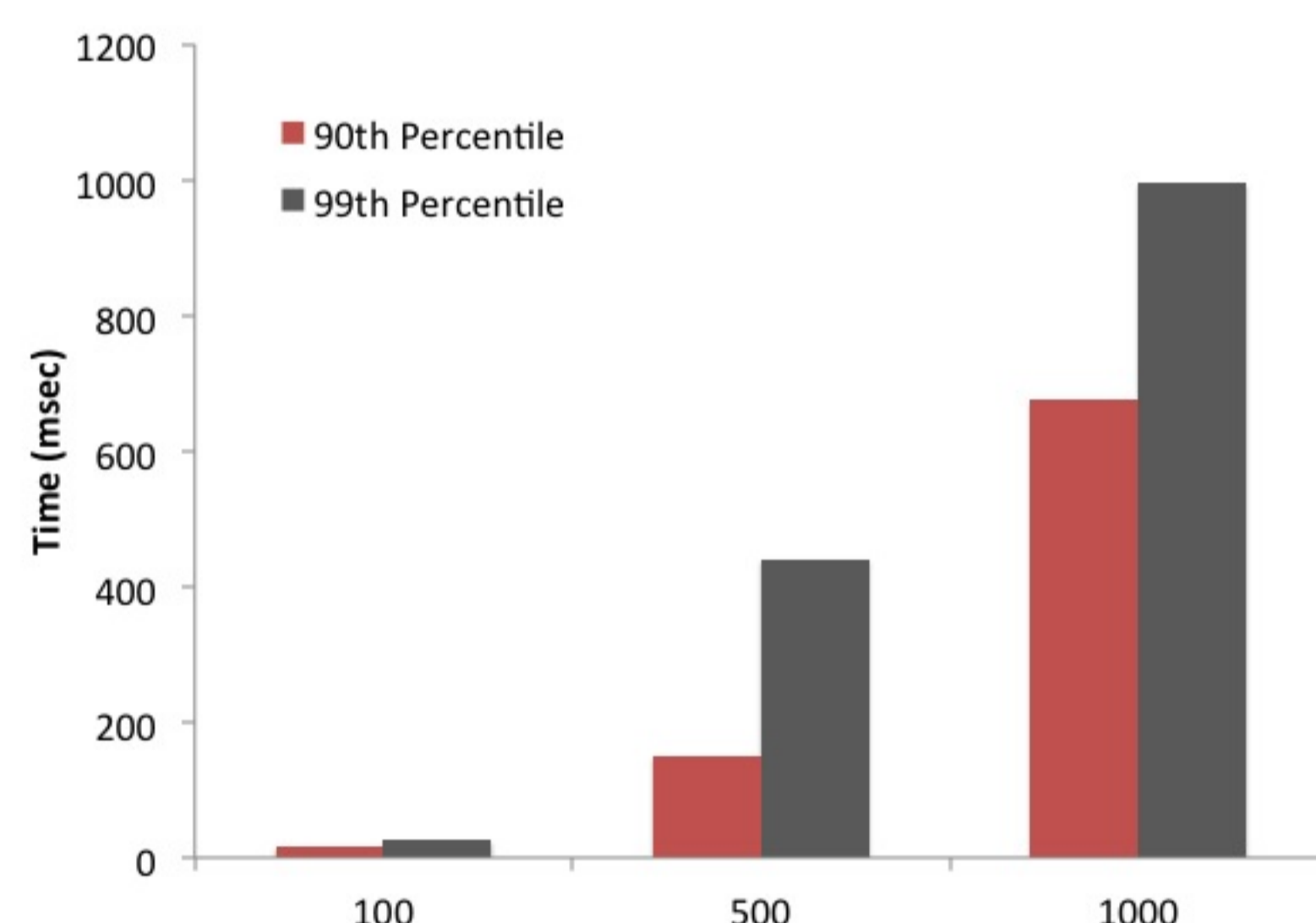
## Can we build scalable and strongly consistent services?

Salient properties:

- ▶ Do not assume accurate failure detection (crash-failure)
- ▶ Minimize replica count (no majority voting)
- ▶ Fully decentralized implementation

## Possible Solution

- ▶ Fail-stop protocols + centralized configuration manager.
- ▶ Manager notifies clients when configuration changes.



Time difference for clients to learn of configuration change using ZooKeeper

## Stale configurations lead to inconsistencies!

- ▶ Not all clients learn about new configuration at the same time.
- ▶ Some might continue to talk to defunct replicas.

## Elastic Replication Internals

- Replicate state to a *configuration* of replicas.
- One replica designated *orderer*.
- All replicas must see an update before responding.
- Stop replicas on fault suspicion.

(each configuration has at least one correct/non-suspected-faulty replica)

## Tolerating Failures

Suspect a replica is faulty?

- Broadcast a *wedge* command to all replicas.
- A wedged replica does not modify its state.
- At least one replica will be wedged.

## Wedging Guarantees Safety!

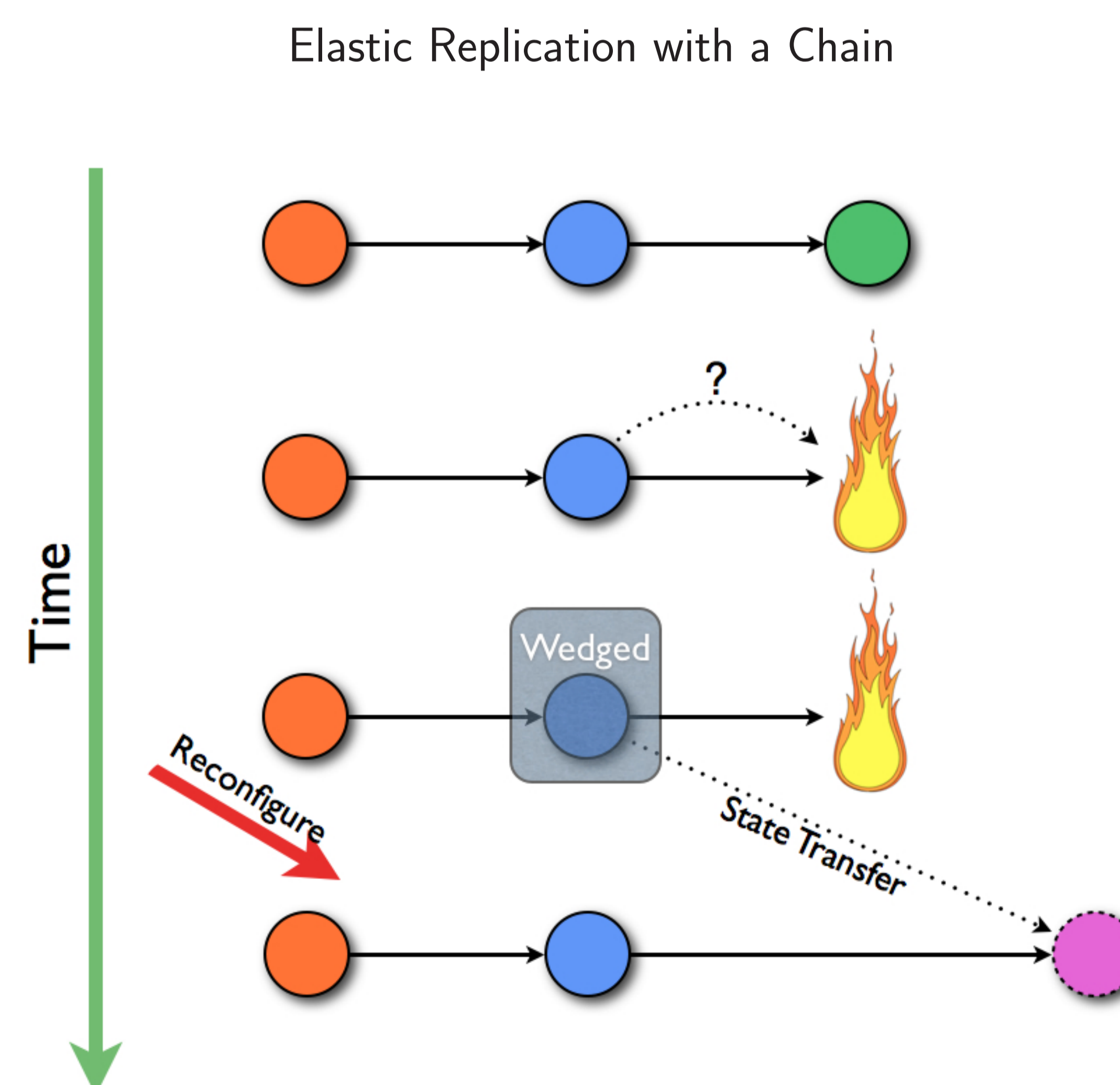
State cannot diverge and clients cannot see inconsistency due to network partitioning or inaccurate failure detection.

## Reconfigure for Liveness

New configuration's replicas *inherit* state from any wedged replica of prior configuration.

## Novelty:

Wedging eliminates need to vote on new configuration's state!



## Scaling Out

### Horizontal partitioning to cut costs

- Each partition is an elastically replicated object.
- A partition's state is separate from its configuration.
- Partitions act as *configuration sequencers* of one another.
- When a partition needs to be reconfigured, its sequencer wedges the old configuration and issues a new one.



- ▶ No need for a centralized configuration management service.

- ▶ Each partition only needs  $f + 1$  replicas to tolerate  $f$  crash failures.

(as long as there always exists at least one partition with no faulty replicas)

