Centiman : Elastic, High Performance Optimistic Concurrency Control by Watermarking

Bailu Ding, Lucja Kot, Alan Demers
Cornell University

Johannes Gehrke
Microsoft Corp.

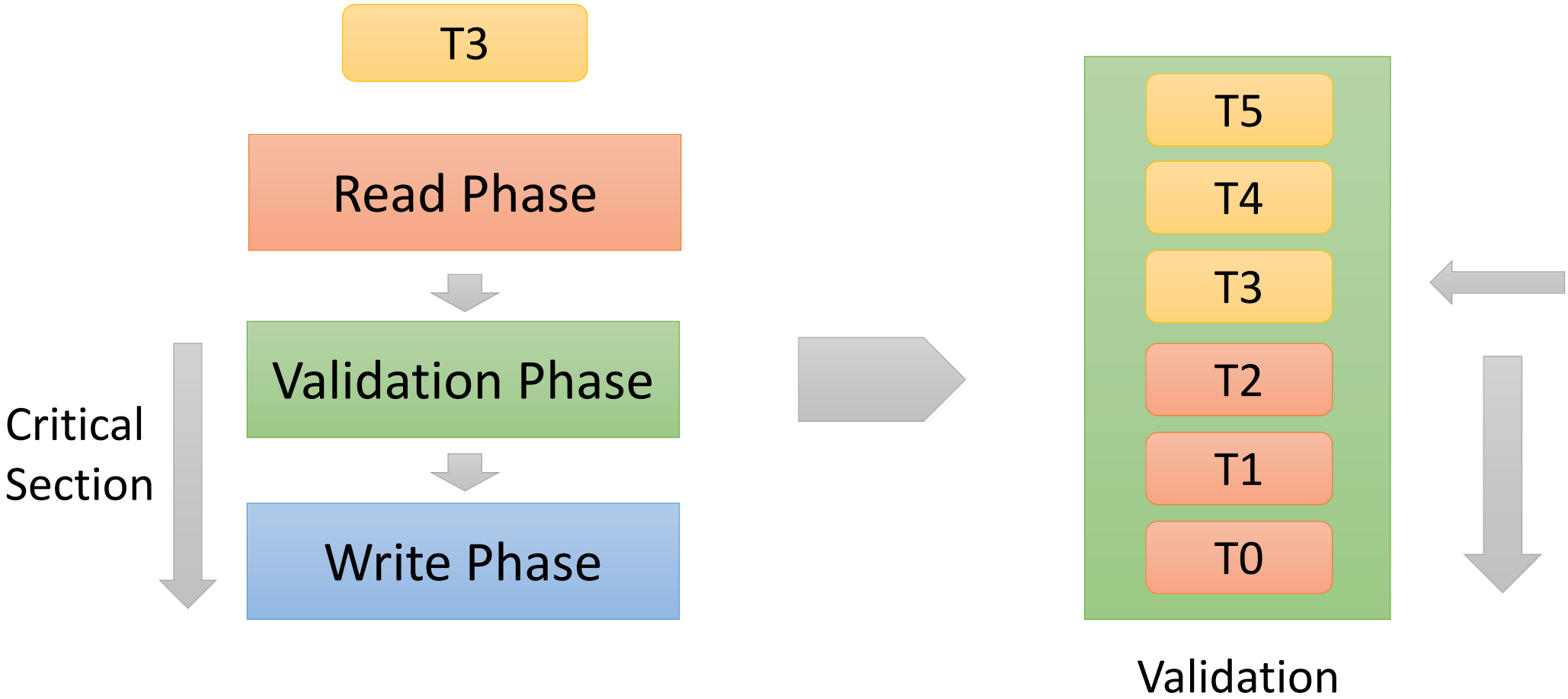
Motivation

- Transaction processing systems face new challenges in the cloud
- Elasticity: want to scale up or down on the fly based on workload
- Modularity: need to decouple storage and processing, must be easy to deploy on commodity infrastructure
- Design Principle: simplicity and loose coupling

Outline

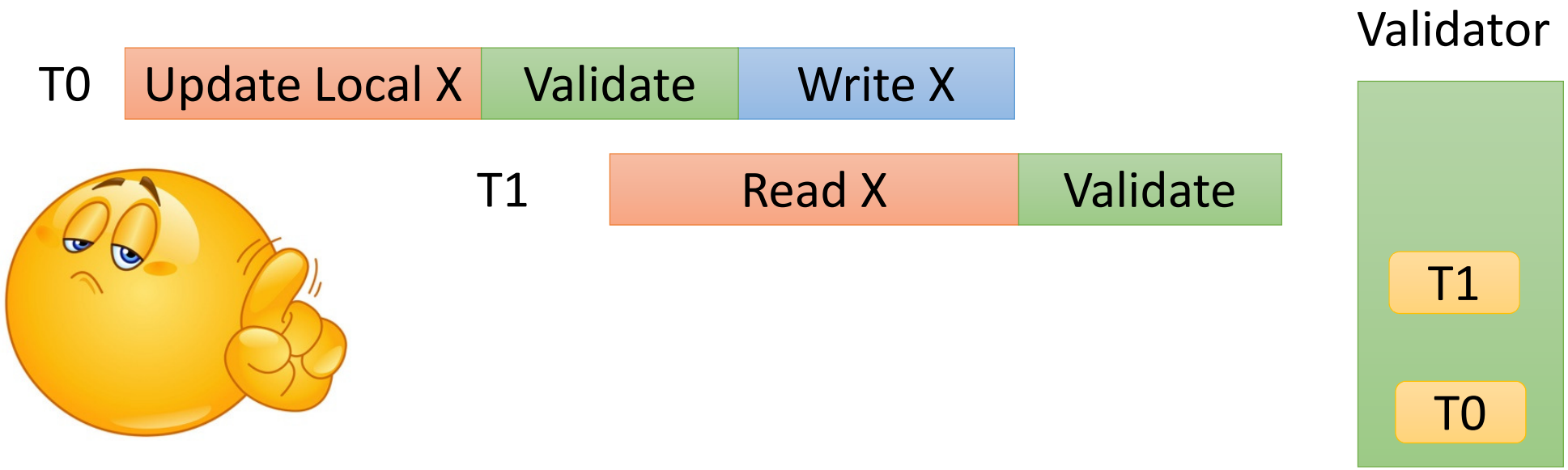
- Motivation
- OCC Background
- Challenges in Distributed OCC
- Sharded Validation with Watermarks
- Elasticity
- Conclusion

Optimistic Concurrency Control



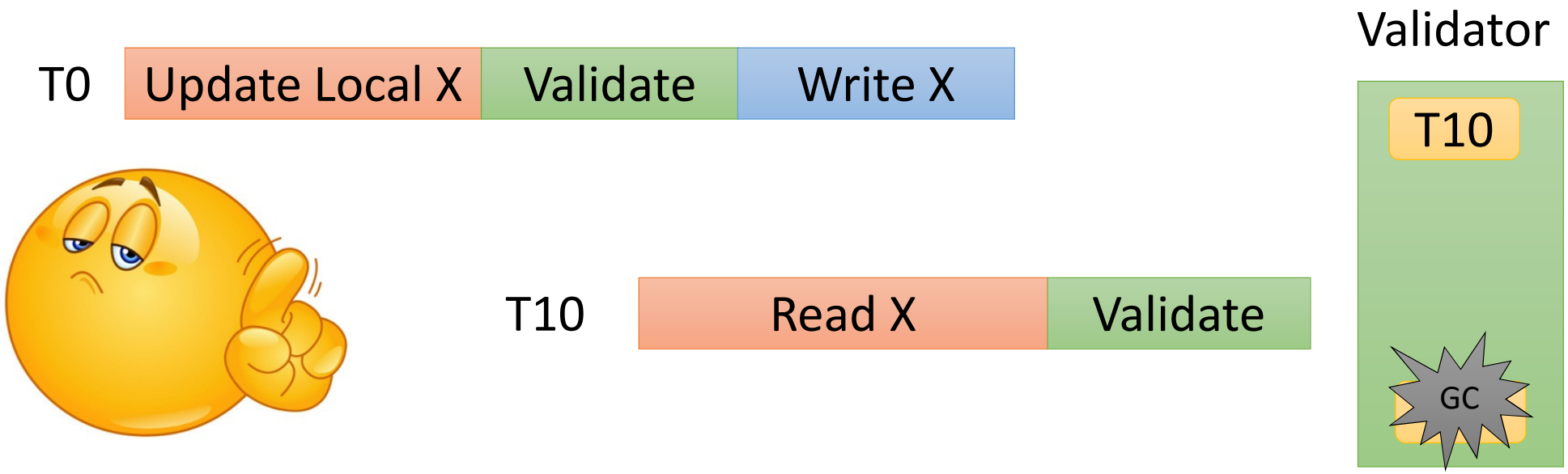
Validation in OCC

- Compare read set with all previously committed write sets



Validation in OCC: State in the Validator

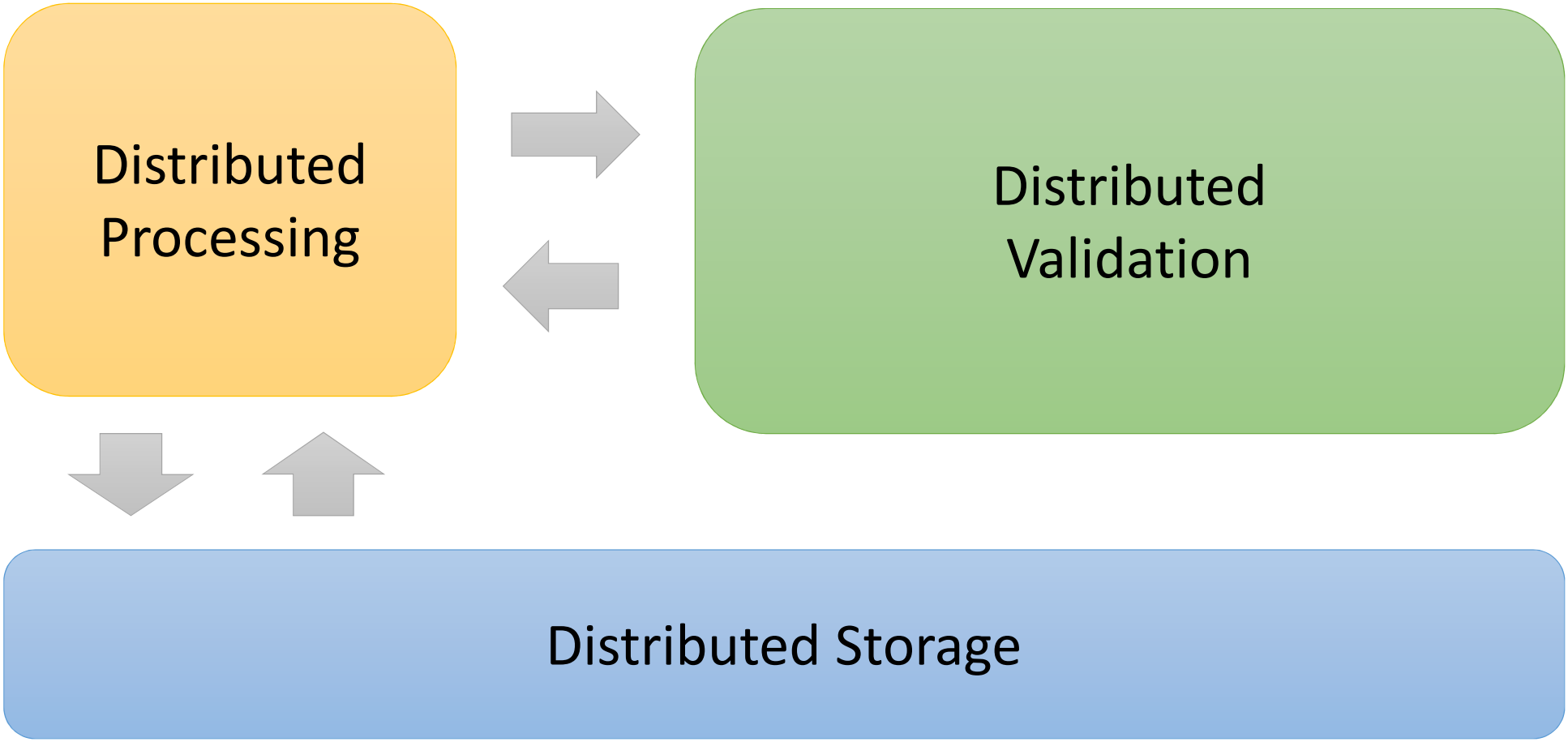
- Need to garbage collect write sets eventually
- If garbage collect too early, may need to abort due to insufficient information



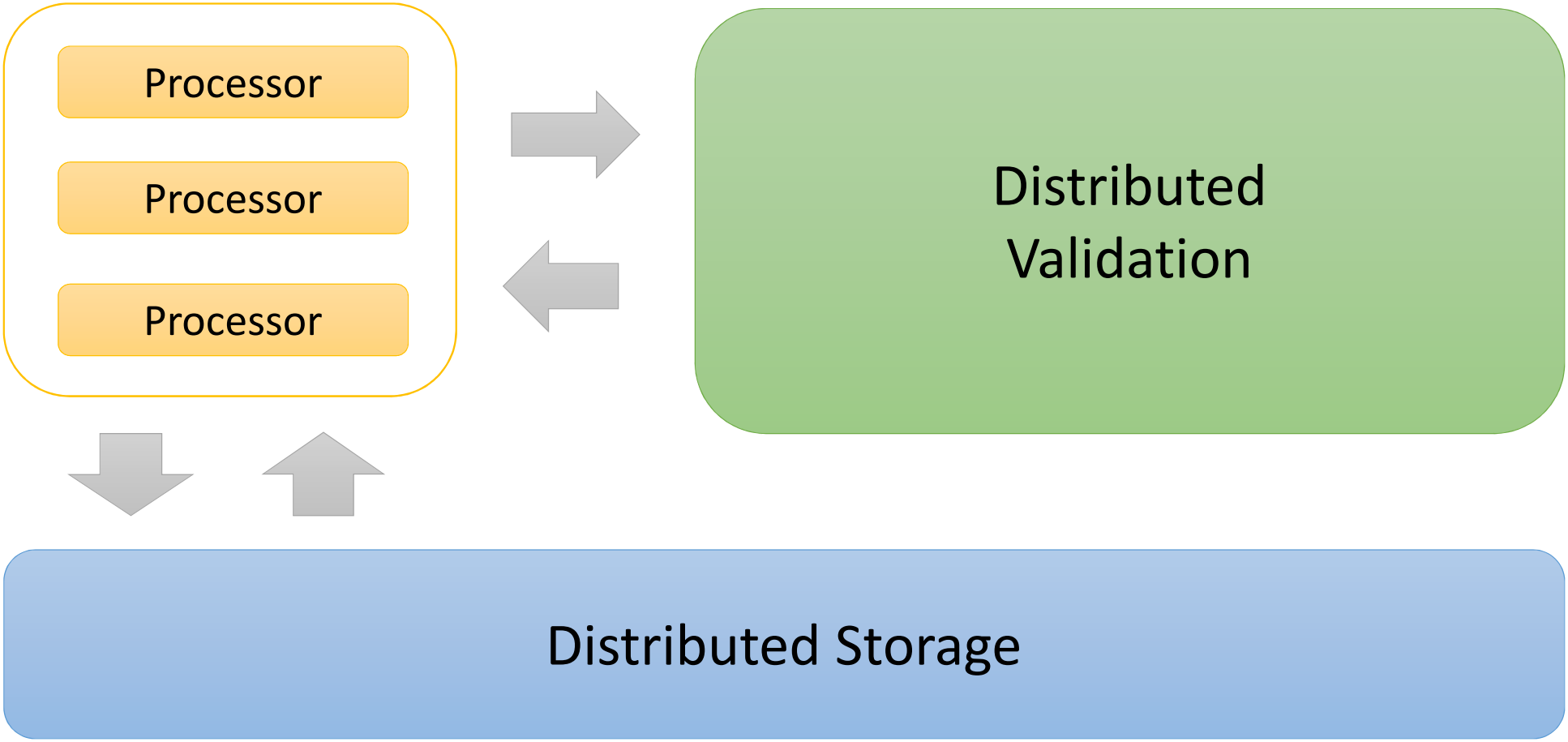
Outline

- Motivation
- OCC Background
- Challenges in Distributed OCC
- Sharded Validation with Watermarks
- Elasticity
- Conclusion

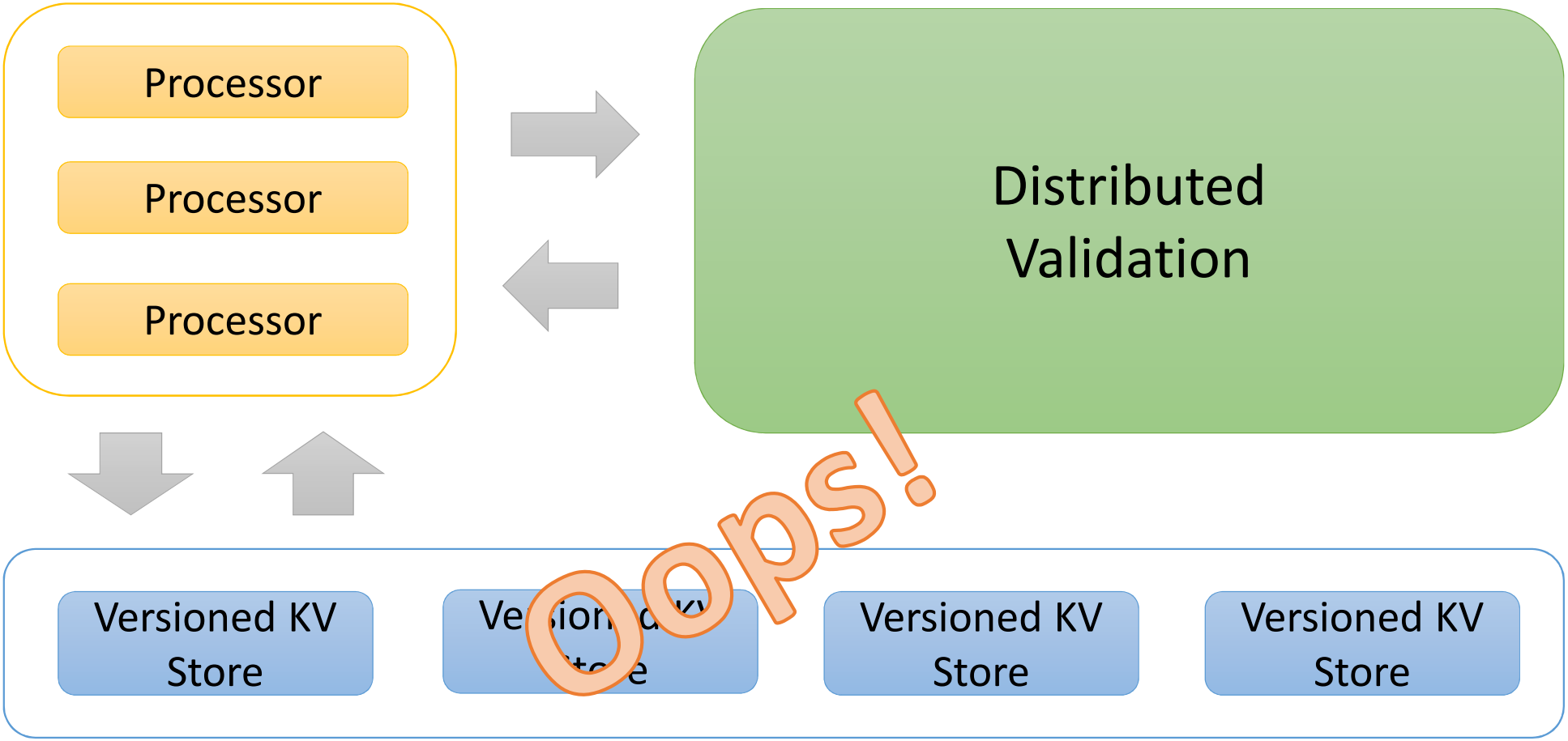
Go Distributed!



Go Distributed!

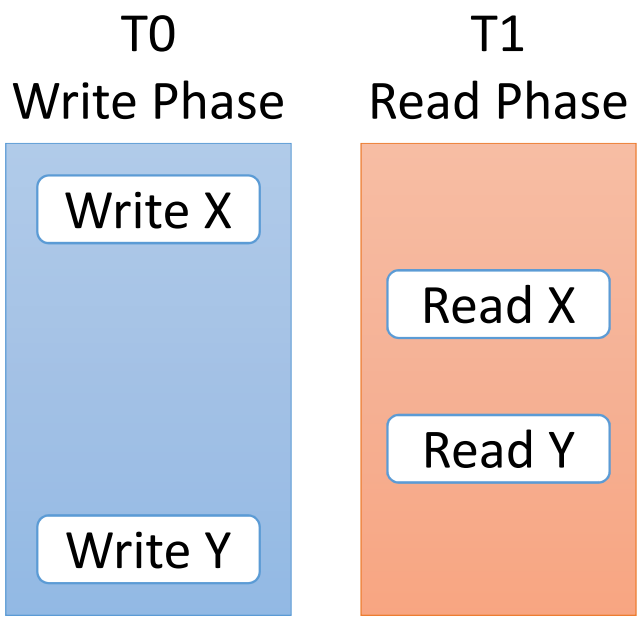


Go Distributed!

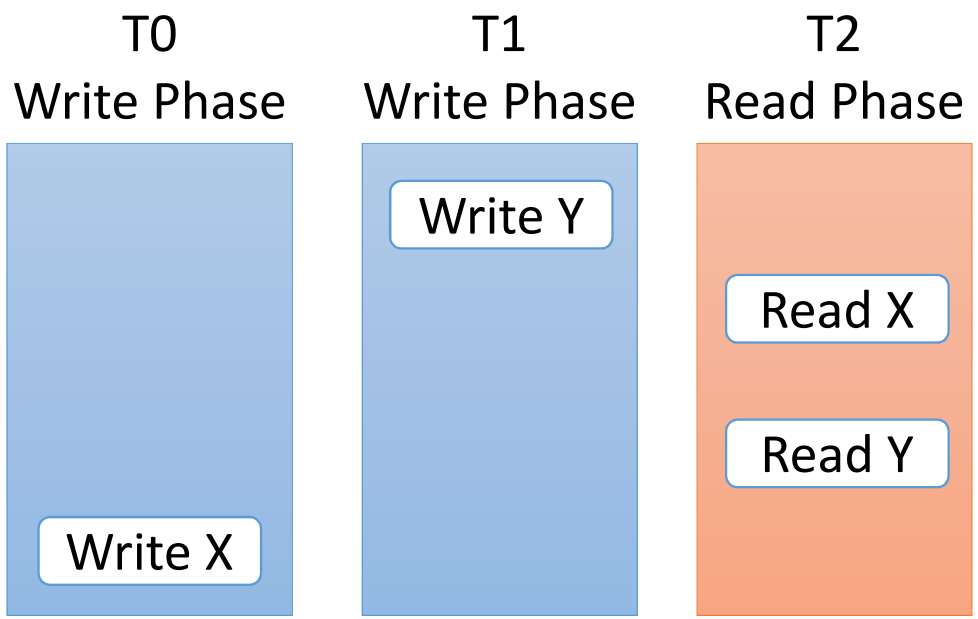


Problem 1 : Reading Inconsistent Snapshots

- Updates not installed in storage atomically
- Check against inconsistent reads at validation

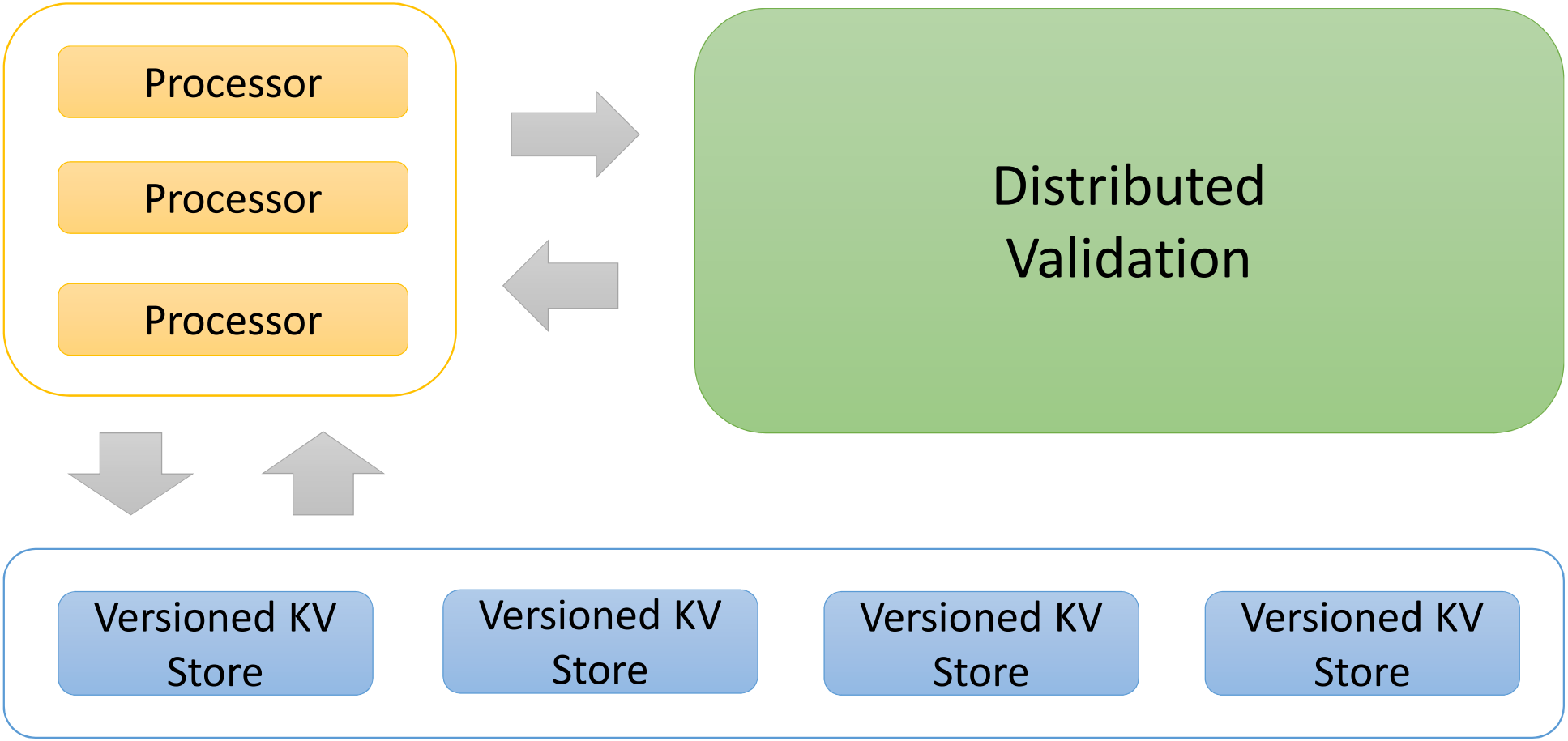


Partial Updates

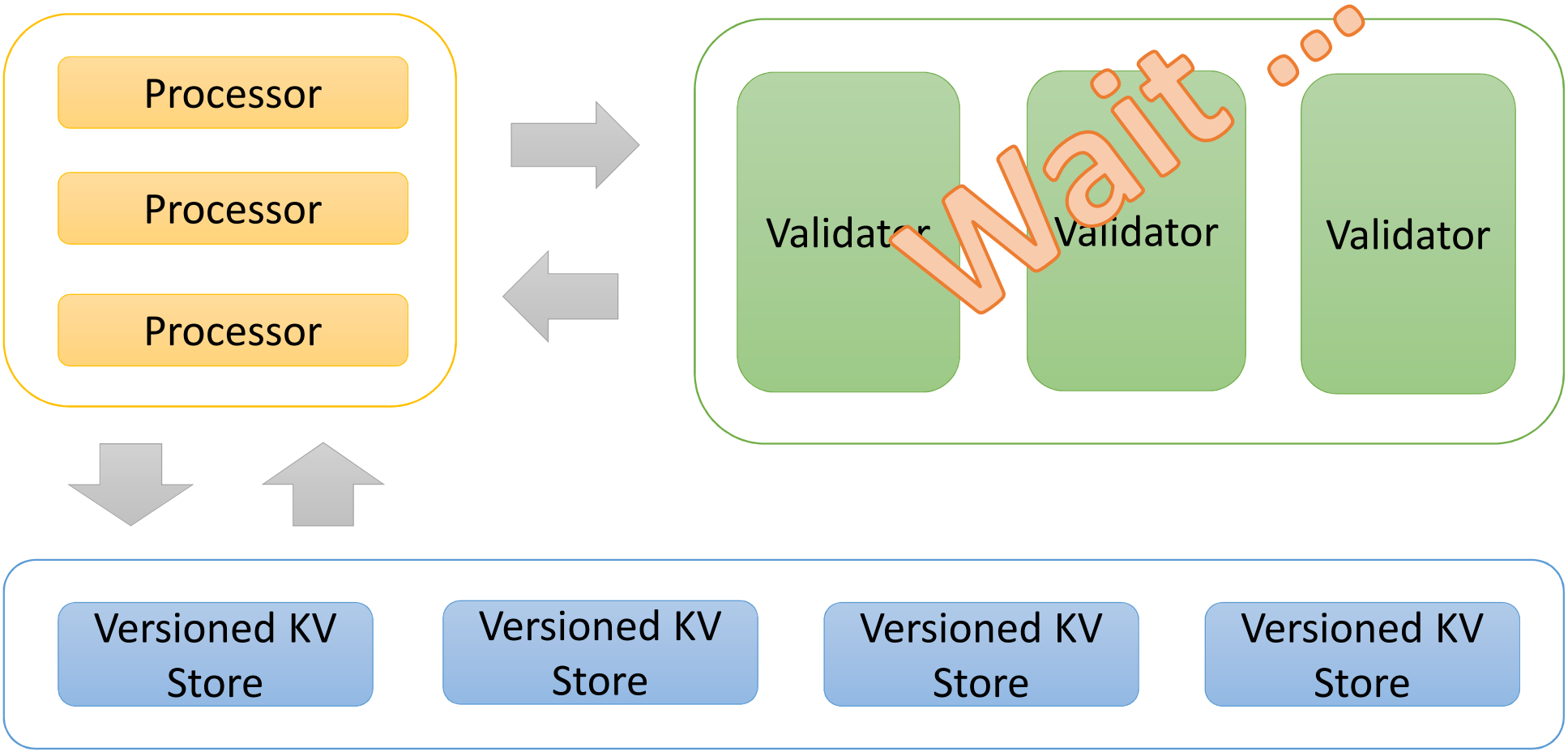


Inconsistent Snapshot

Go Distributed!



Go Distributed!



Problem 2 : Divergent Decisions

T3 R(…), W(X), W(Y)

Abort!

T4 R(X), W(X)

Validator A
X

Validator B
Y

Validator A
X

No conflict!

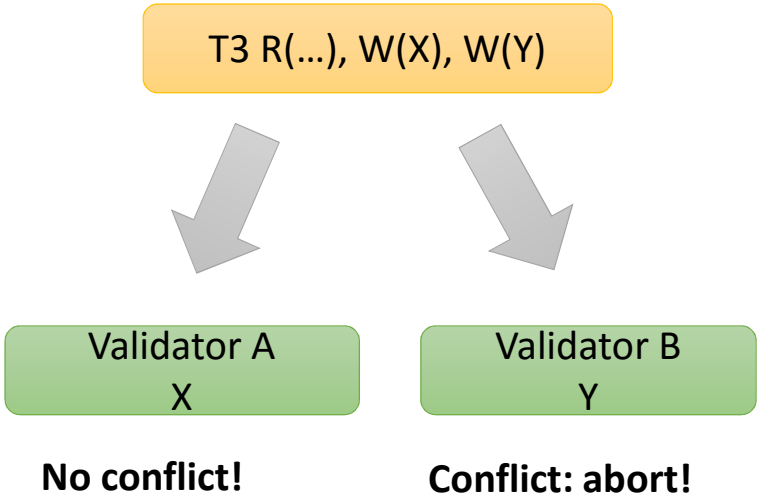
Conflict: abort!

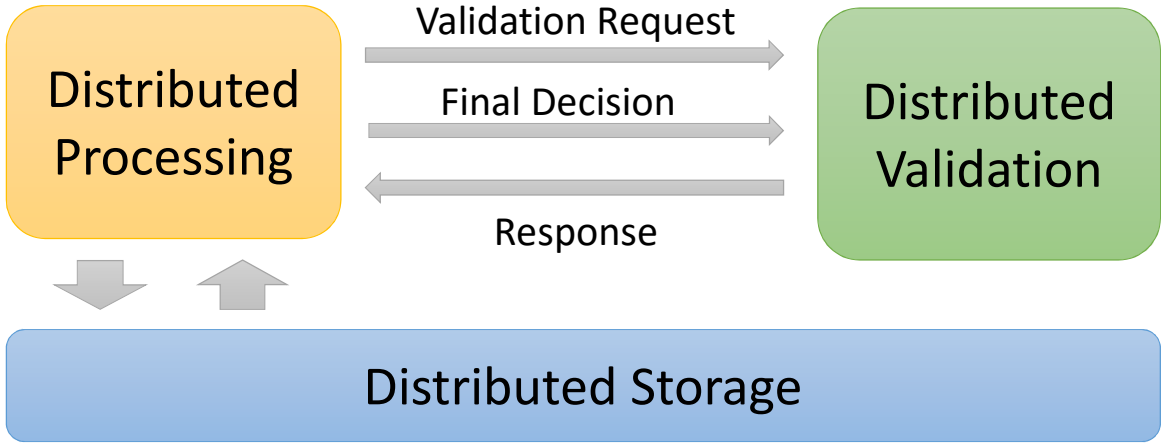
Thinks T3 committed, detects conflict between T3 and T4

Spurious Abort Due to Spurious Update!

Eliminate Spurious Aborts: Proactive

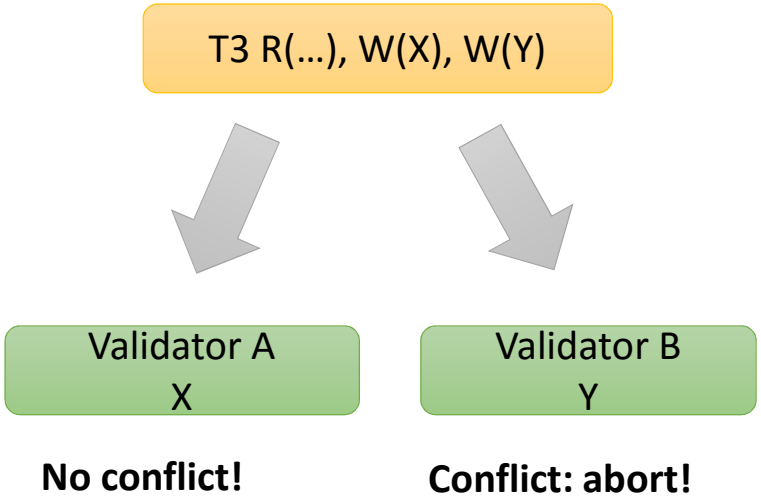
- Processor informs validators about the final decision
- Synchronous: slow down the system
- Asynchronous: additional complexity
  - Revoke updates, handle loss of messages

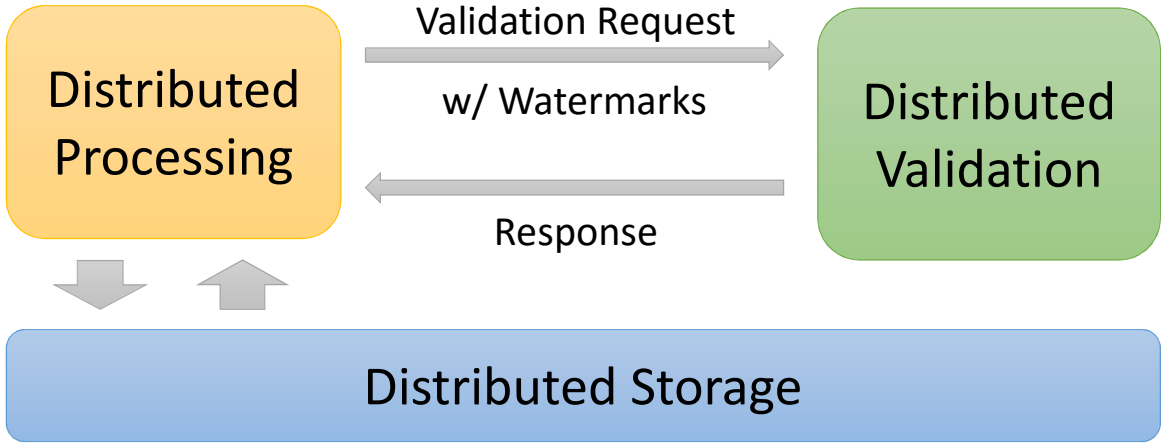




Eliminate Spurious Aborts: Reactive

- Spurious updates are garbage collected after a while
  - Loose coordination between processors and validators
- Tradeoff: aborts due to insufficient info vs. spurious aborts
- Our approach: watermarks



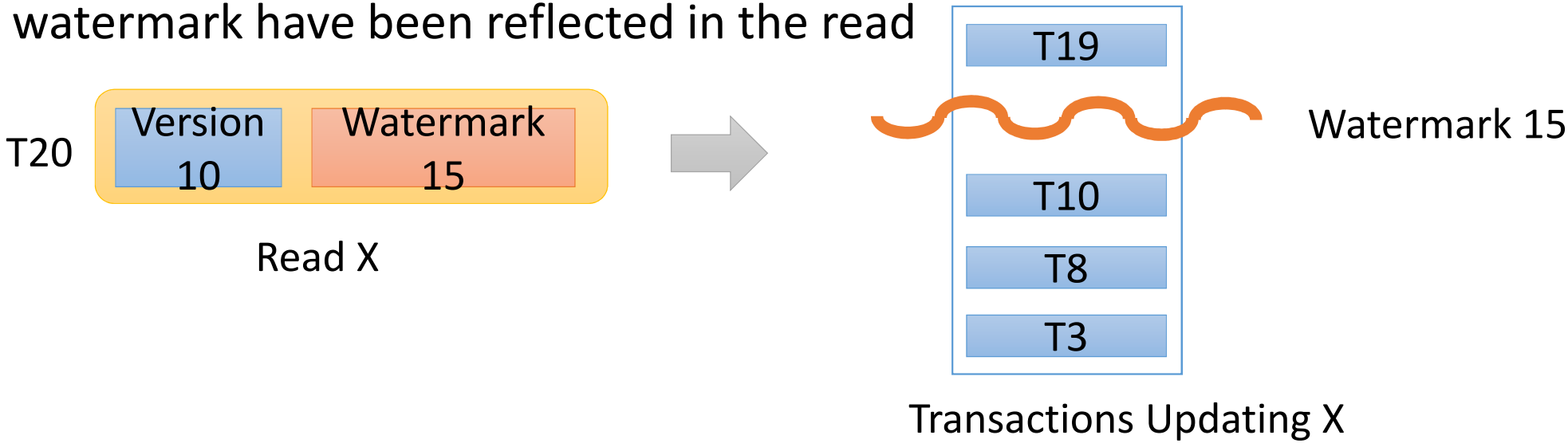


Outline

- Motivation
- OCC Background
- Challenges in Distributed OCC
- Sharded Validation with Watermarks
- Elasticity
- Conclusion

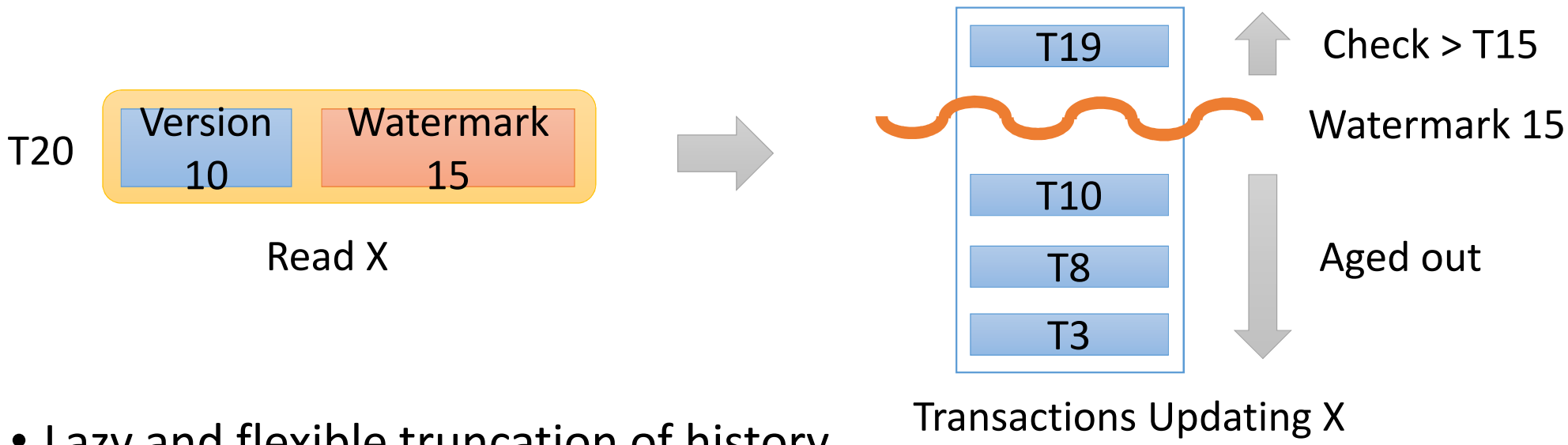
Watermark Abstraction

- Transactions are validated in timestamp order
- The watermark is of the same type as the timestamp
- System associates a read on a record with a watermark
- Guarantee: all updates made by transactions to the record before the watermark have been reflected in the read



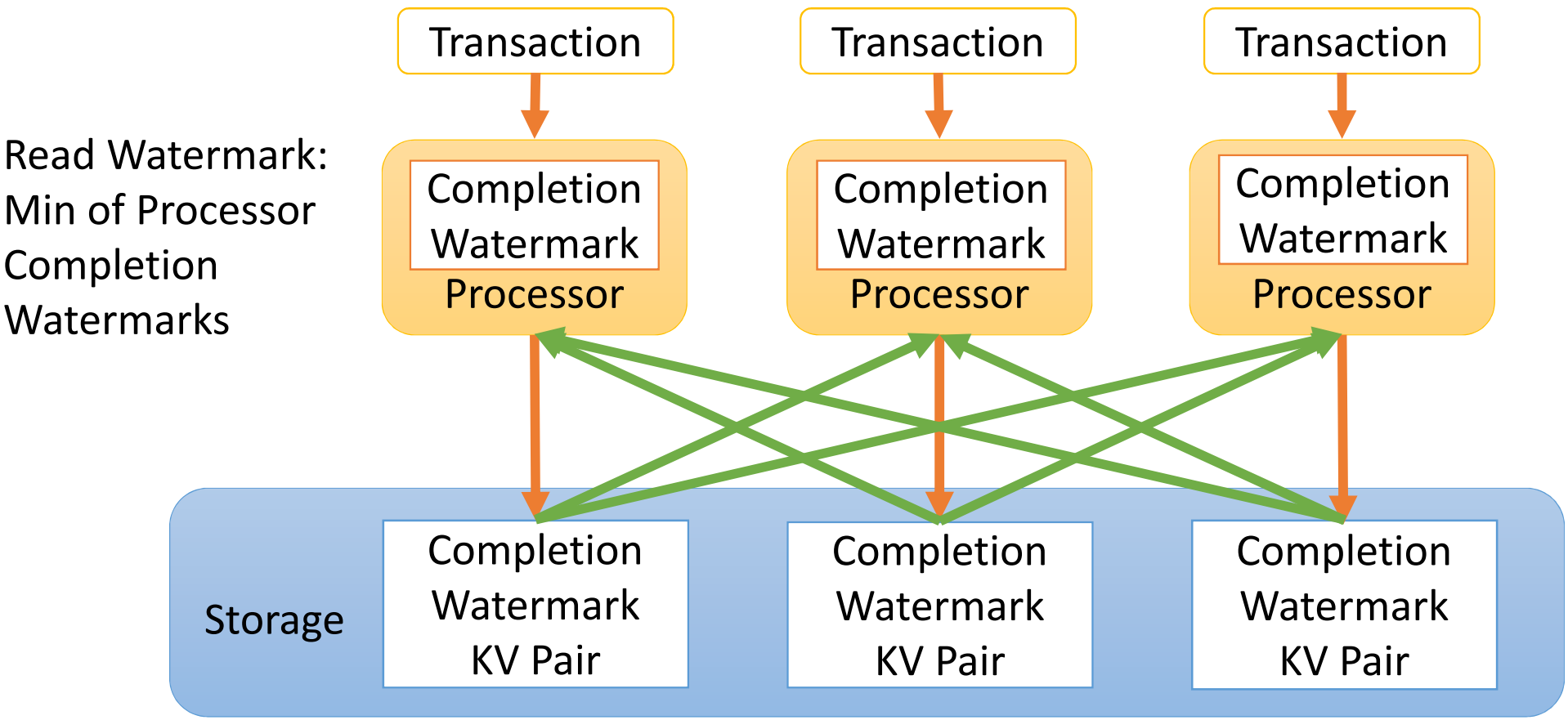
Watermarks Reduce Spurious Aborts

- Spurious updates aged out when watermark advances beyond the updating transactions

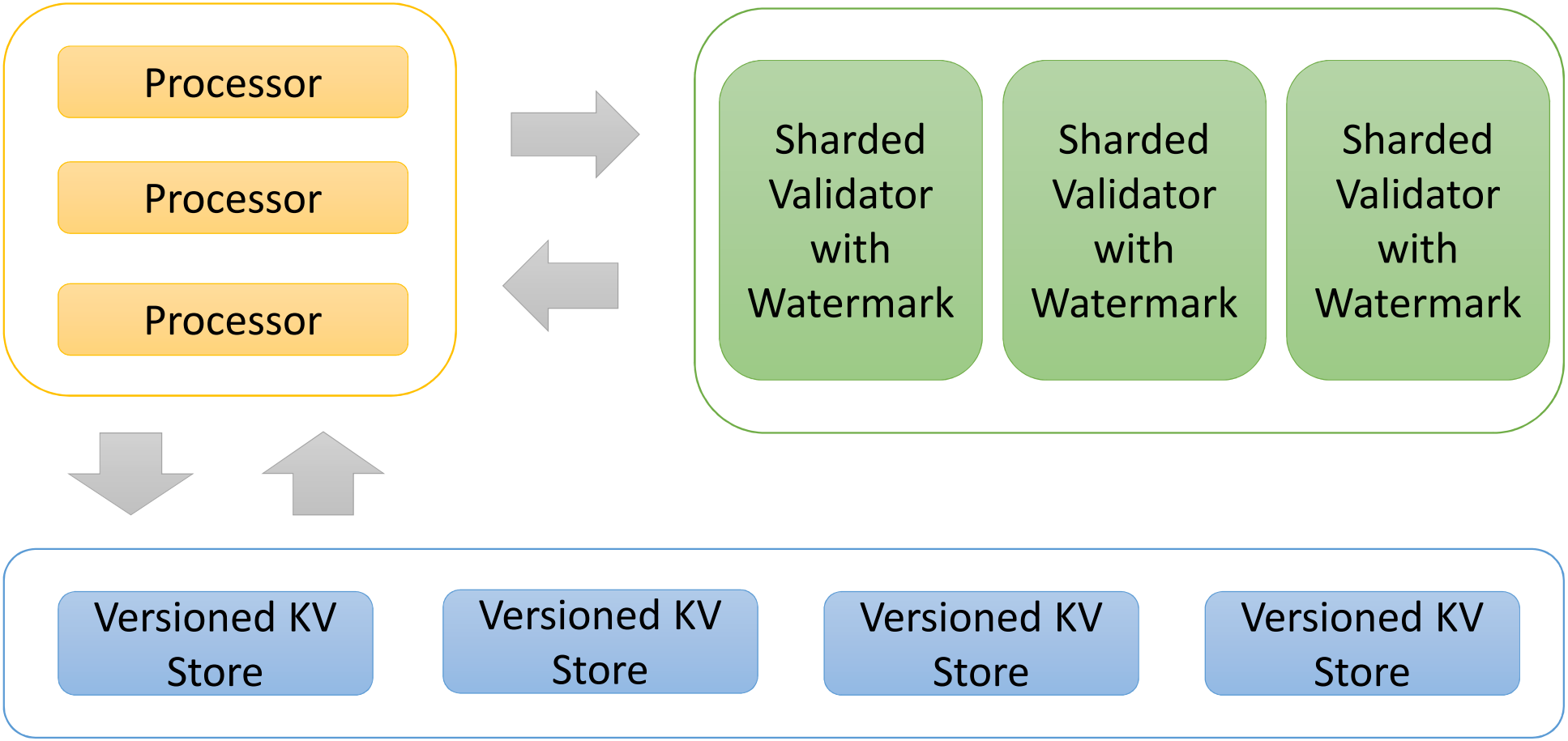


- Lazy and flexible truncation of history

Implement Watermarks



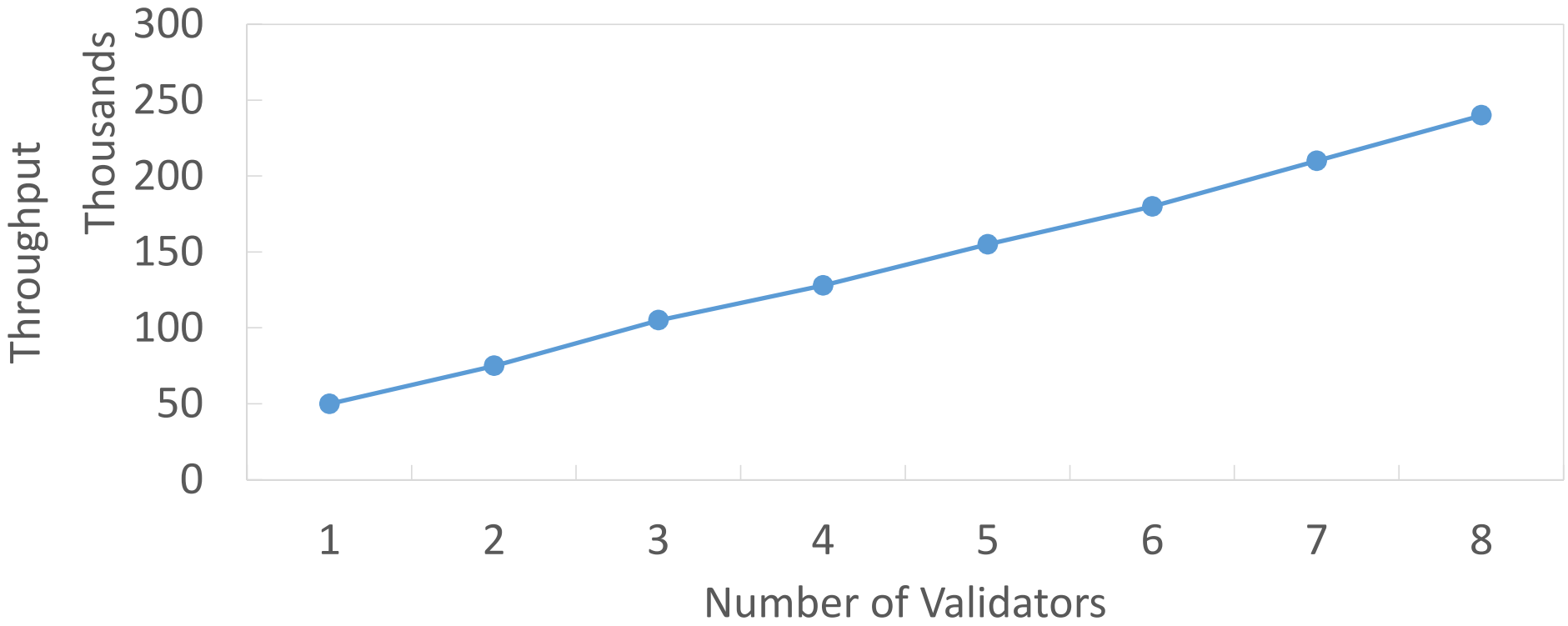
Go Distributed! → Centiman Architecture



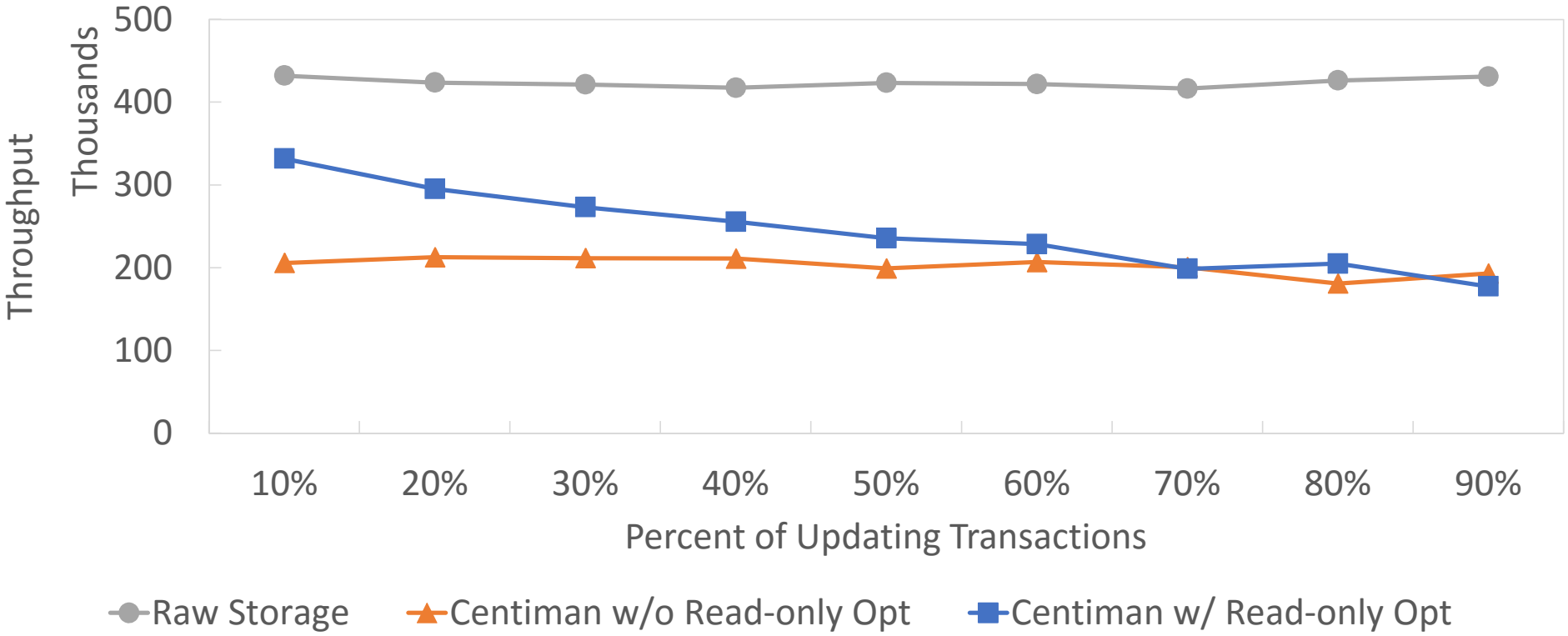
Experiment Settings for TPC-C

- Each machine: 20 EC2 compute units, 7GB RAM
- 50 processor and 50 storage instances

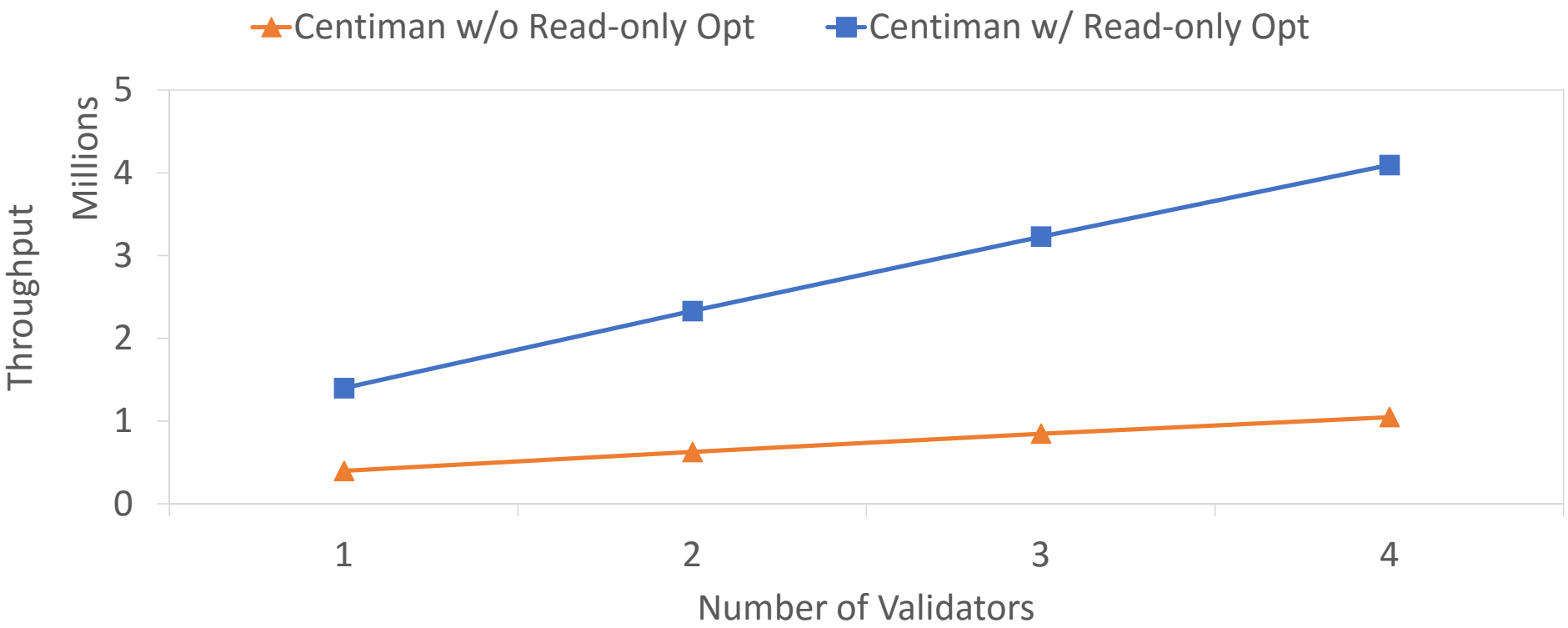
Experiment Result on TPC-C



Watermarks Optimize Read-Only Transactions



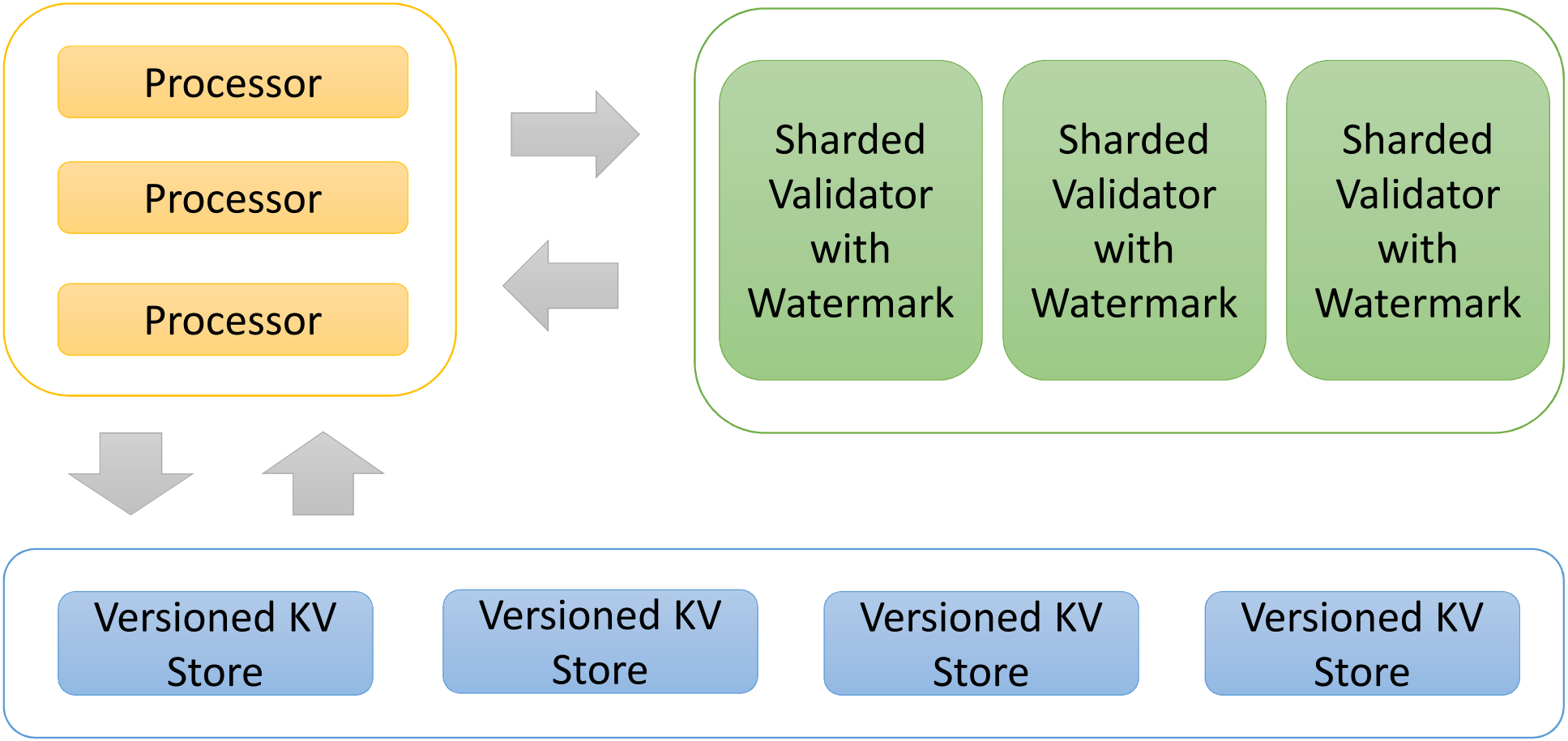
Experiment Result on TATP



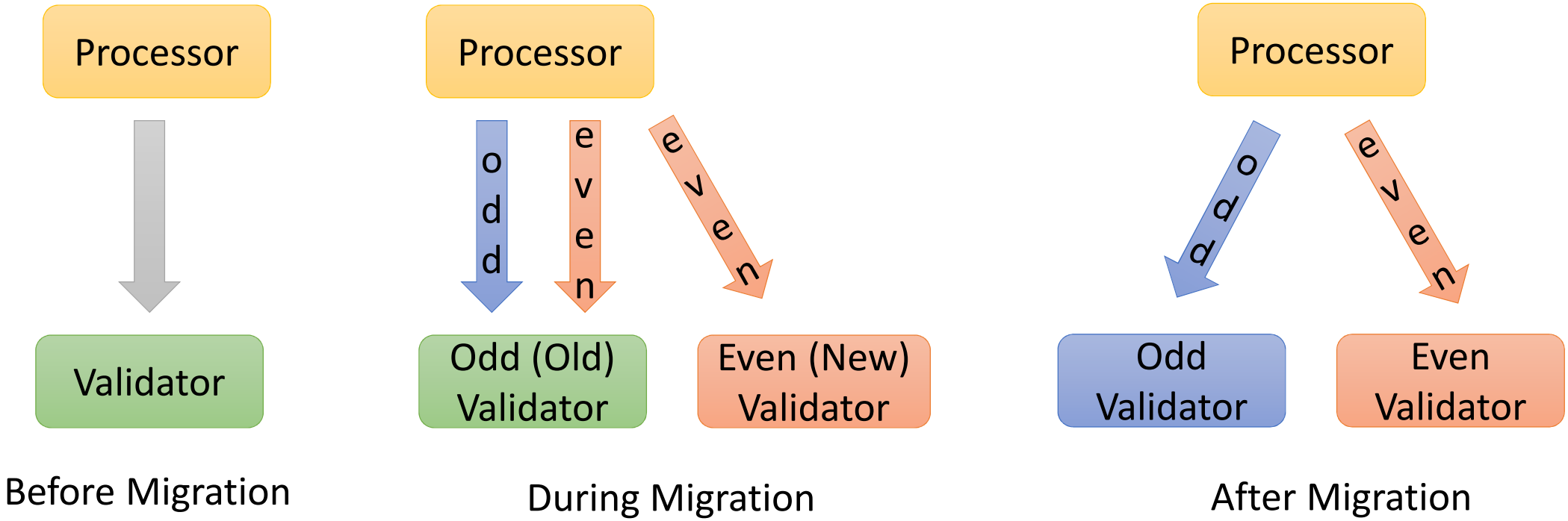
Outline

- Motivation
- OCC Background
- Challenges in Distributed OCC
- Sharded Validation with Watermarks
- Elasticity
- Conclusion

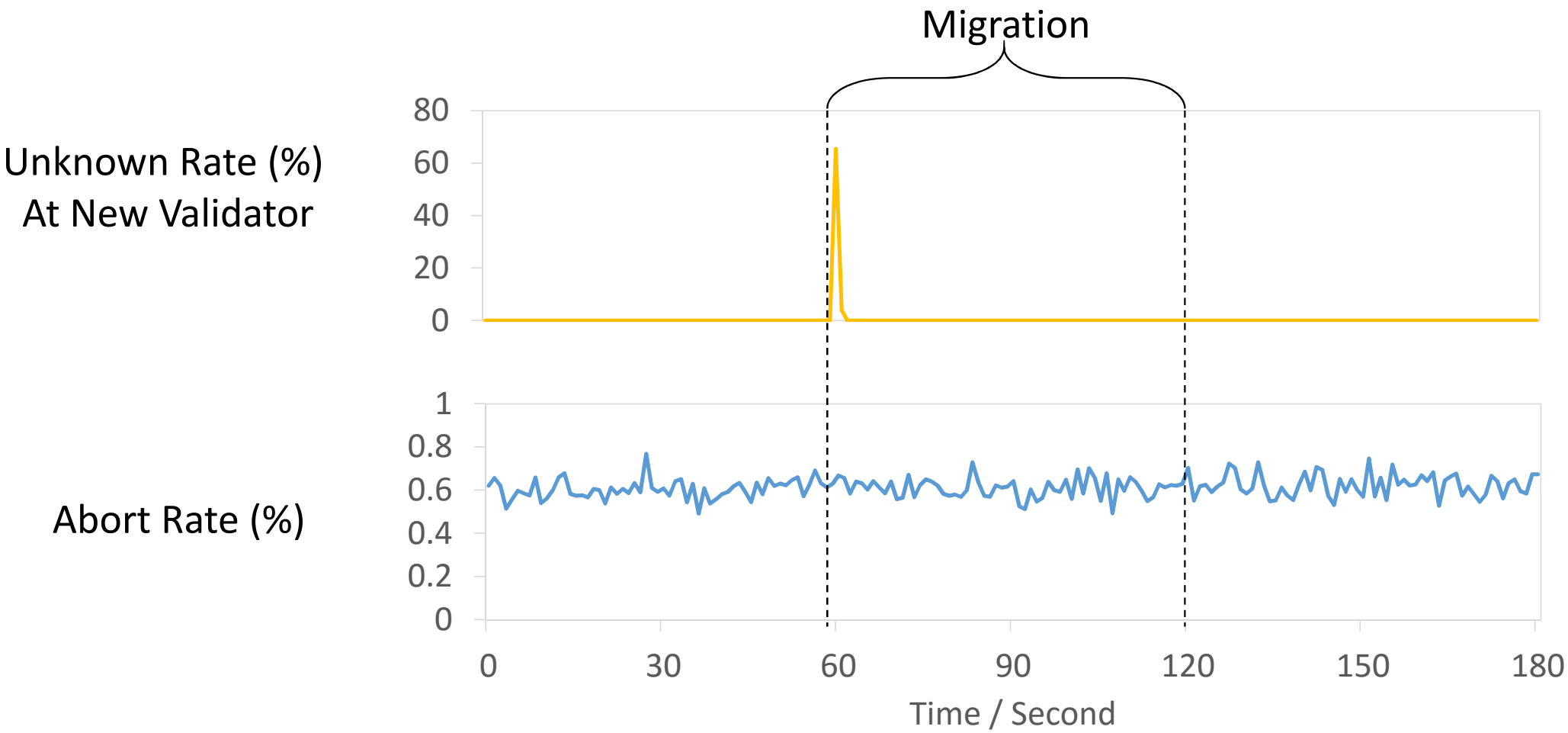
Elasticity


Processor
Processor
Processor
Sharded Validator with Watermark
Sharded Validator with Watermark
Sharded Validator with Watermark
Versioned KV Store
Versioned KV Store
Versioned KV Store
Versioned KV Store

Elasticity of the Validators



Elastic Validation : 3 Validators to 4 Validators



Conclusions

- Centiman is a decoupled and elastic distributed transaction processing system based on optimistic concurrency control
- Watermark abstraction reduces spurious aborts, optimizes read-only transactions, enables elastic validation
- Extensive experiments in the paper: high performance

Question?