Centiman: Elastic, High Performance Optimistic Concurrency Control by Watermarking

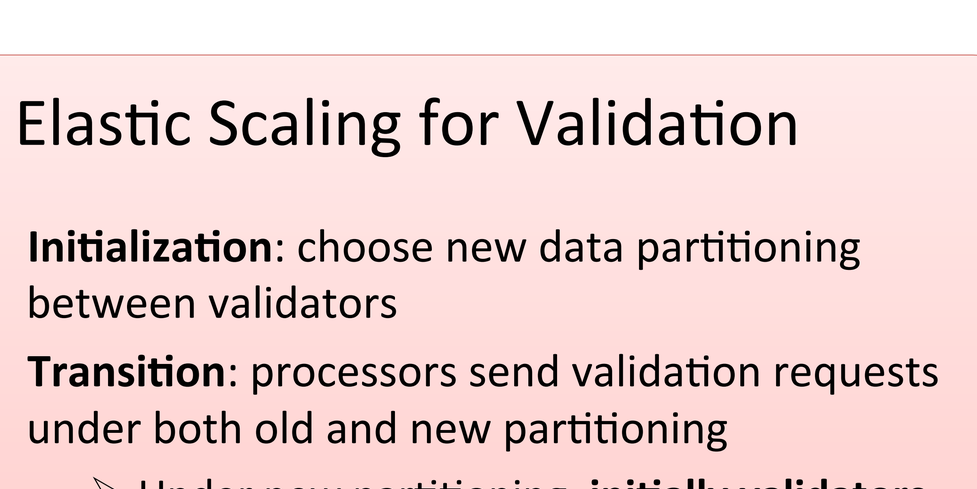
Database Group

Bailu Ding, Lucja Kot, Alan Demers, Johannes Gehrke
Cornell University and Microsoft Corp.

Transactions in the Cloud

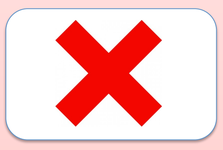
- Goal: high performance, strong consistency, elasticity, and loose coupling
- Centiman: **OCC-based transaction engine** on top of a key-value store
- **Sharded validation**: # of validators changes elastically, minimal points of synchronization, no shared state
- **Watermarks:** lazily propagate info about global state, optimize read-only transactions

Centiman System Architecture



Sharded Validation & Watermarks

- **Partitioned OCC validators**: commit only if all agree
- Problem: **spurious aborts**



Validator 1 abort

Validator 2 commit

Txn aborts but Validator 2 stores write set and uses it to validate future txns; may abort unnecessarily

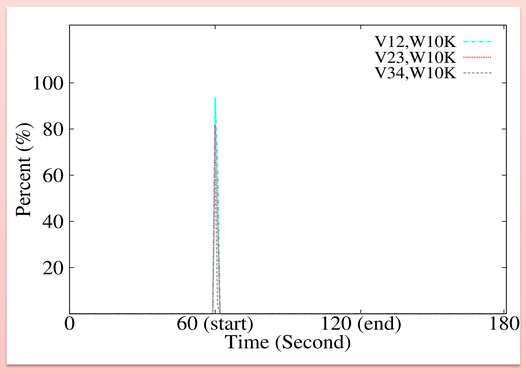
- Solution: **watermarks** lazily propagate through system info about completed transactions
  - Each read associated with a read watermark
  - Bad write sets "*age out*"
  - **Local check optimization** for read-only txns

Elastic Scaling for Validation

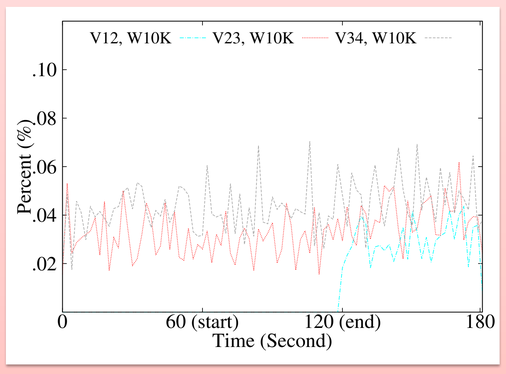
- **Initialization**: choose new data partitioning between validators
- **Transition**: processors send validation requests under both old and new partitioning
  - Under new partitioning, **initially validators have insufficient state;** abort conservatively
  - Authoritative decisions made under old partitioning
- **Switch**: when validators have sufficient state, switch over to new partitioning

Evaluation: Elastic Scaling

- Vxy - start with x validators, scale to y
- Watermarks updated every 10K transactions
- Scaling starts at 60s, ends at 120s



% of transactions for which new validator has insufficient state to validate

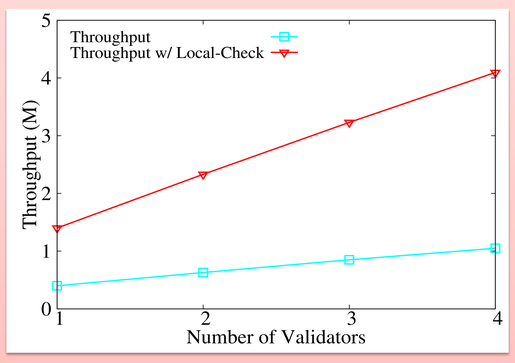


% of spurious aborts in system

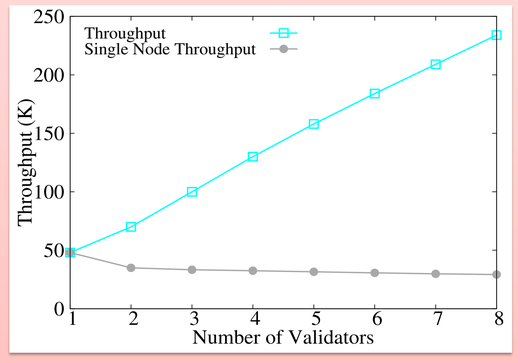
- no spurious aborts possible with one validator

Evaluation: Benchmarks

- Stress validators with up to 50 storage nodes and 50 processors



TATP



TPC-C