# High Performance Replicated Transactional Data Store

**Zhongmiao Li** • Peter Van Roy • Paolo Romano zhongmiao.li@uclouvain.be

UCL Université catholique de Louvain — TÉCNICO LISBOA

## Background

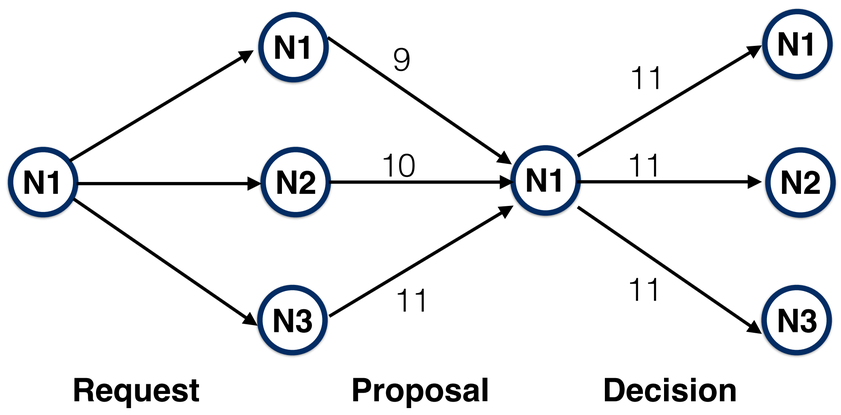
- State-Machine Replication (SMR) is a classical technique to implement fault-tolerant services providing linearizability.
- The SMR-approach has been leveraged to build replicated data store.
- SMR assumes full-replication and total-order execution among all replicas, which is not scalable.

- State-partitioning can be applied to scale up SMR-based data store, with the following requirements:
  - Determinism: each replica commits transactions by the given order
  - Genuineness: partitions should not be involved in the recipient or execution of transactions that do not involve them.

## Proposing Solution

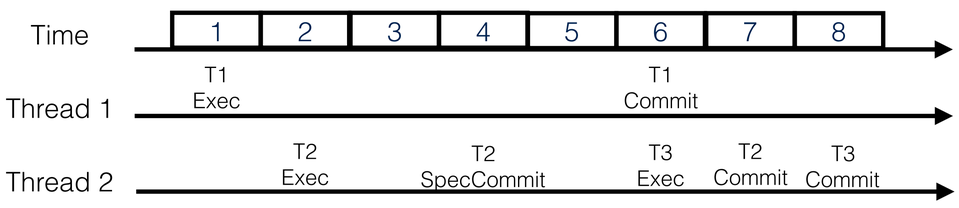
- A scalable, massively-parallel partitioned data-store relying on the SMR approach.

- Scalable transaction dispatching protocol
  - Inspired by Skeen's algorithm
  - Partitions agree on the batch to include multi-partition transactions




- A partition can not proceed beyond a proposed batch before hearing the decision
- Dispatching is executed in each DC before replication, due to its non-determinism
- Transactions are deterministically order by timestamp with a batch

- Highly-parallel speculative execution
  - Multiple worker threads execute transactions concurrently, but commit transactions by the given order



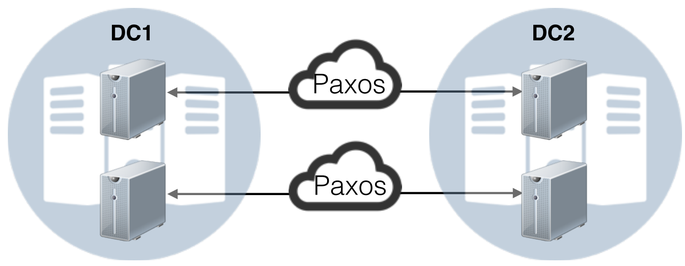

- **Issues**
- Missing versions: T2 missed updates from T1
  - When a transaction reads, it leaves a 'read dependency'
- Minimize the chance of 'missing versions'
  - Writes to a key results in locking the key, avoiding later transactions to read it
- Multi-partition transaction
  - Each involved partition executes the transaction and sends its local read-set to other involved partitions

## Existing Approaches

Existing work: Calvin: Fast Distributed Transactions for Partitioned Database Systems
Scalable State-Machine Replication

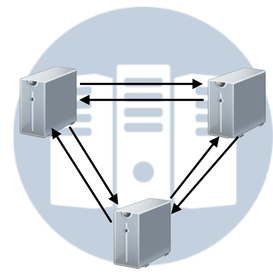
**1 Replication**

Replicas of the same partition exchange batched transactions.



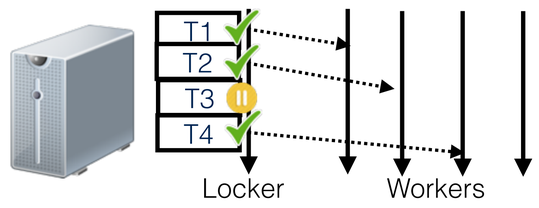

**2 Dispatching**

Transactions are sent to involved partitions.
Calvin: all-to-all broadcast
S-SMR: a `global` Paxos instance to total-order multi-partition operations



**3 Execution**

Transactions are executed in a given order.
Calvin: a single thread to serialize lock requests ( assumes transactions' read/write sets are known).
S-SMR: a single thread to execute operations.

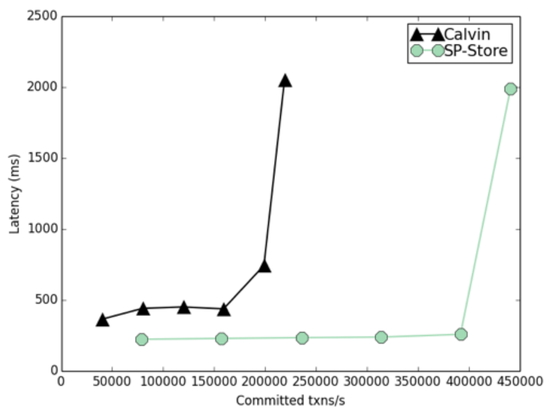



## Preliminary Results

Evaluated on Grid'5000 to compare with Calvin.
Deployed on 8 nodes, each with 8 cores.
200ms delay is injected to simulate geo-replication.

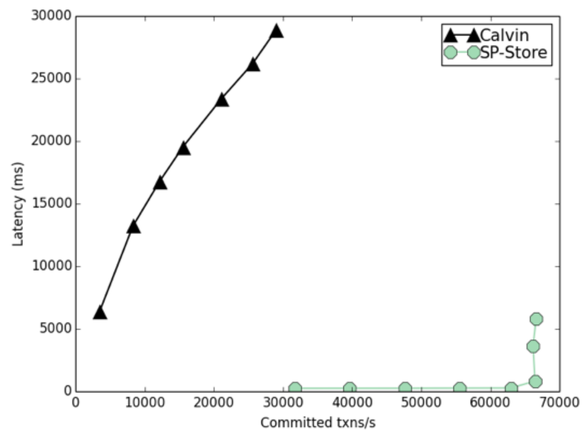
### Micro benchmark

- Transactions read and update 10 keys
- Calvin's throughput is bottlenecked by its single locking thread



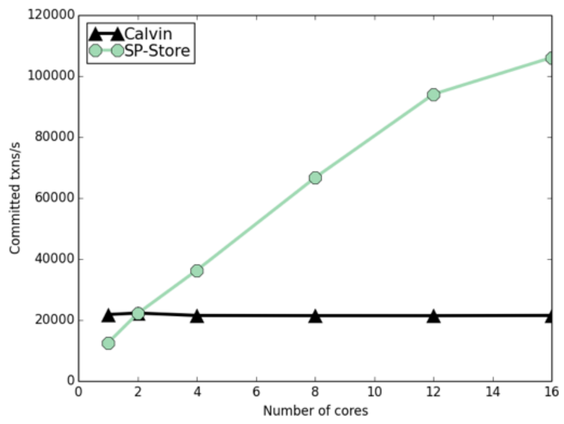

### TPC-C

- In Calvin, transactions whose rw sets are not know in advance should be executed preliminarily to predict their rw sets
- Three out of five transactions type need to be run preliminarily
- This results in the high latency of Calvin




### Scalability

- Micro-benchmark, deployed on a single machine with large number of cores.
- The throughput of SP-Store scales linearly up to 16 cores
- Calvin's throughput is bottlenecked by its single locking thread




## More information