Scalable Speculative Replicated Data Store

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

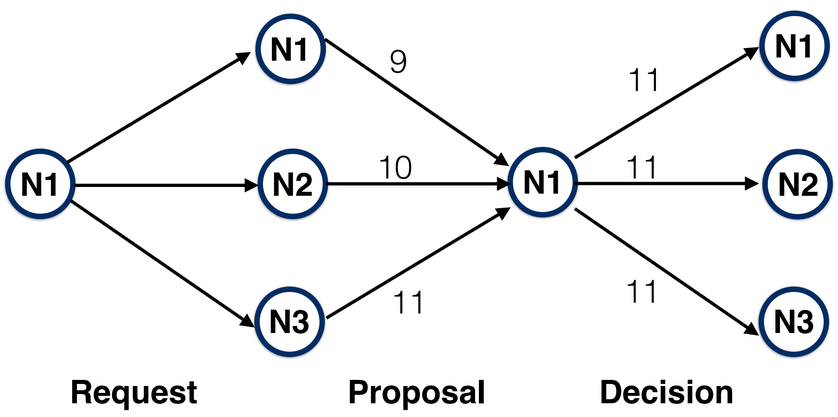
Background

- State-Machine Replication (SMR) implements fault-tolerant services with linearizability; it is a natural approach to build replicated data stores.
- SMR relies on a non-scalable to approach to ensure the determinism of replication:
  - Total order execution: all replicas execute transactions by the given order
- State-partitioning can be used to scale up distributed services, which requires:
  - Parallelism: transactions should be executed as concurrently as possible
  - Genuineness: the reception or execution of operations should only involve accessed partitions
- Achieving these conflicting requirements is not trivial!

Proposing Solution

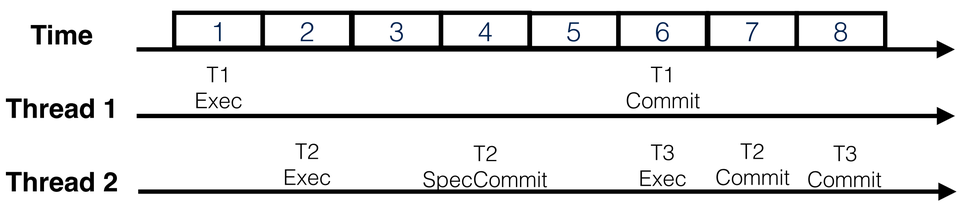
- A scalable, massively-parallel partitioned data-store built upon the SMR approach.

- Scalable transaction dispatching
  - Inspired by Skeen's algorithm
  - Partitions agree on the batch to include multi-partition transactions
  - Transactions are deterministically order by timestamp within a batch



- Partitions can not proceed beyond a proposed batch before hearing the decision
- Dispatching is executed in each DC before replication, due to its non-determinism

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



Issues

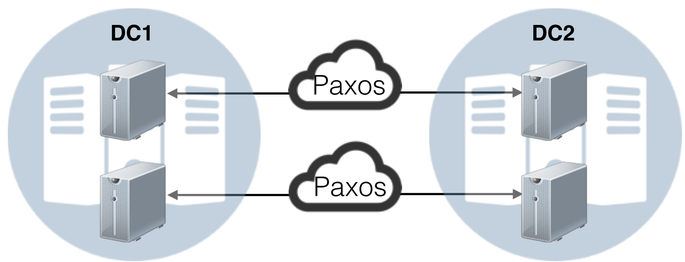
1. Missing versions: T2 missed updates from T1
   - When a transaction reads, it leaves a 'read dependency'
2. Minimize the chance of 'missing versions'
   - Write operation locks the key, avoiding later transactions to read it prematurely
3. Multi-partition transaction
   - Each involved partition executes 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

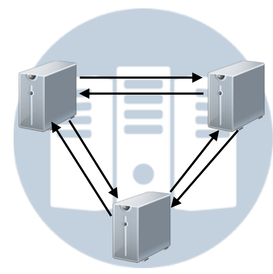
1 Replication

Replicas of the same partition exchange batched transactions.



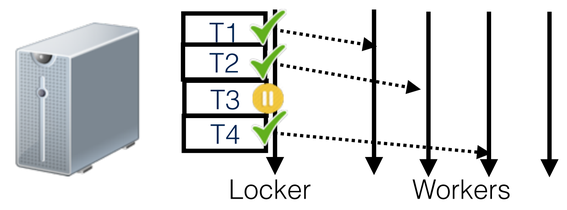
2 Dispatching

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



3 Execution

Each partition executes transactions by the given order.
- S-SMR: a single thread to execute operations.
- Calvin: assume knowing transaction's read/write sets and rely on a single thread to serialize lock requests

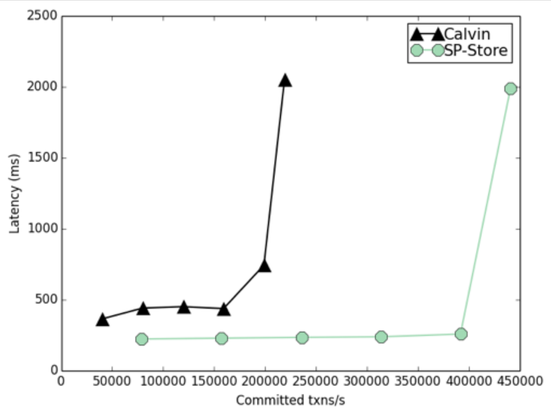


Preliminary Results

Deployed on Grid'5000.
Data store consists of 8 nodes, each with 8 cores.
200ms delay is injected to simulate geo-replication.

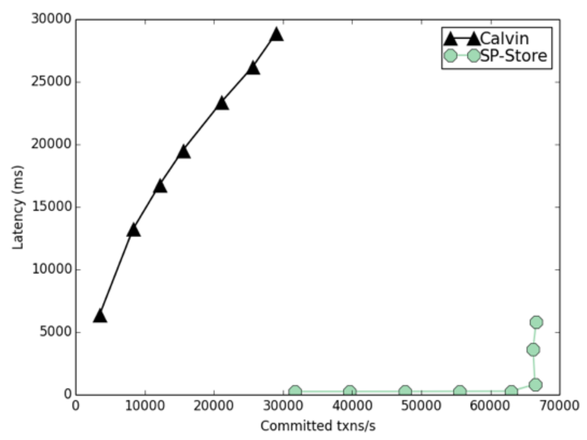
Micro benchmark

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



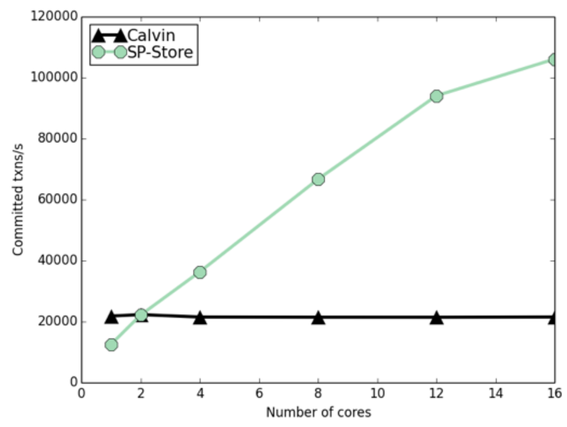
TPC-C

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



Scalability

- Micro-benchmark on a single machine.
- The throughput of SP-Store scales linearly up to 16 cores



Summary

- This work proposes a scalable transactional protocol designed for partitioned, replicated data store based on the SMR-approach.
- The next stage is to evaluate the scalability of our multicast protocol.