HIGH-PERFORMANCE CONCURRENCY CONTROL MECHANISMS FOR MAIN-MEMORY DATABASES

7/6/22

Boren Zang,
David R. Cheriton School of Computer Science

UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS

Overview

- Background of MVCC
- Optimistic MVCC
- Pessimistic MVCC
- Evaluation
- Personal reflection

Main Memory Database

- Data reside in memory.
- Support high transaction rates.
- Current concurrency control methods (exp. Single-version locking) do not always scale.

Multiversion Concurrency Control (MVCC)

- Serialization of transactions
  - Read stability
    - The readability should not change when a transaction tries to read a version of the record.
  - Phantom avoidance
    - Scans do not return new transactions

Lower isolation level than serialization

- Repeatable read: No phantom avoidance.
- Read committed: Only guarantee reads are committed. No validation is required.
- Snapshot isolation: Read as beginning of versions. No validation is required.

Example

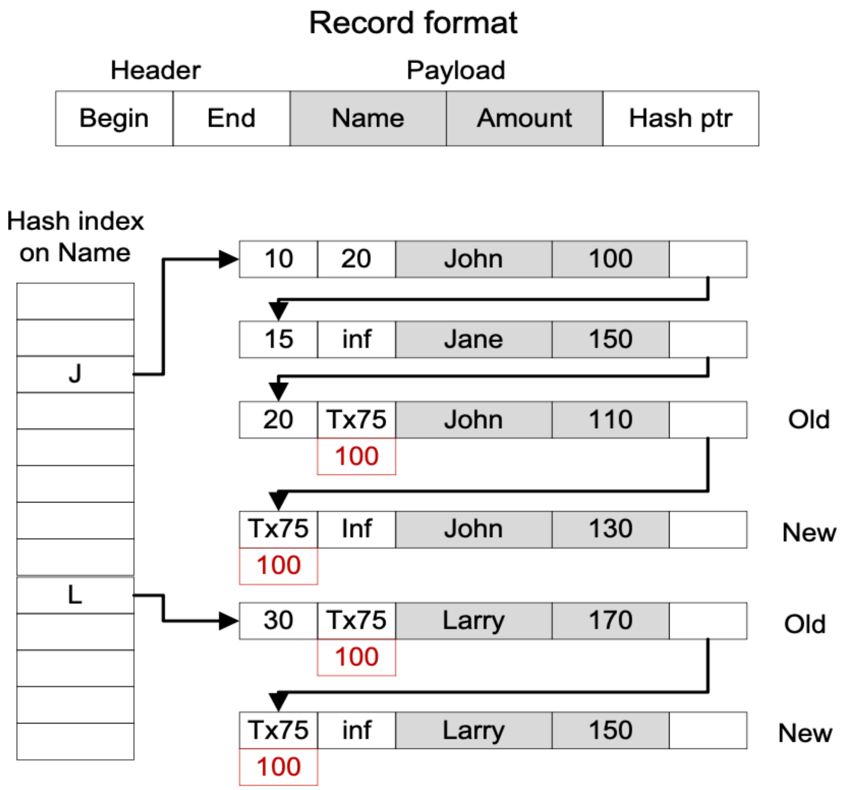


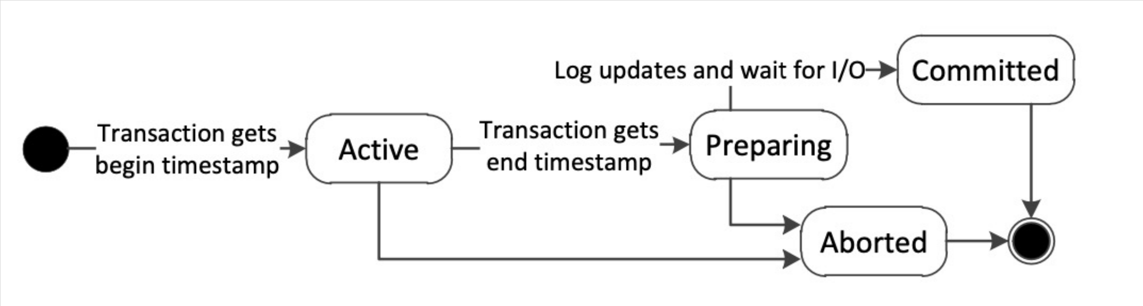
Figure 1: Example account table with one hash index. Transaction 75 has transferred $20 from Larry's account to John's account but has not yet committed.

TRANSACTION PHASES

Normal processing phase

Preparation phase

Postprocessing phase.



OPTIMISTIC TRANSACTIONS

Validation-based

Normal Processing Phase

- Start scan
  - Record information about indexes and predicates
- Check predicate
- Check visibility
  - May need commit dependency check
- Read version
  - Store versions into a ReadSet for further validation
- Check updatability
  - Updatable: End field equals infinity or it contains a transaction ID and the referenced transaction has aborted

Normal Processing Phase (cont.)

- Update version
  - The transaction creates a new version
  - Set Transaction ID to End Field
- Delete version
  - Update without creating new version

UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS

Preparing Phase

- Read validation
  - Read visibility check
  - Check for phantoms

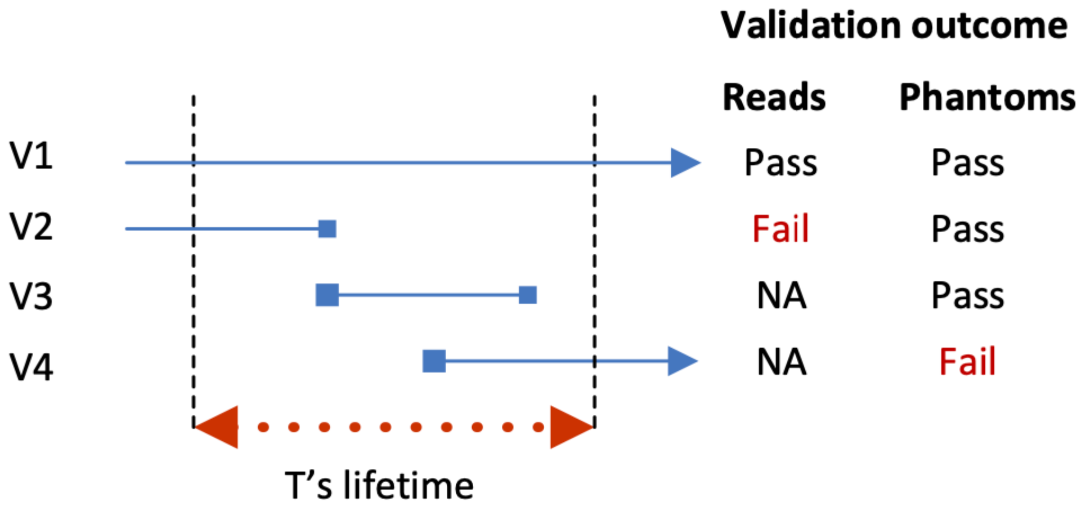


Figure 3: Possible validation outcomes.

PESSIMISTIC TRANSACTIONS

Lock-based

Lock Types

- Record Locks
  - Locks on versions
  - Ensure version readability
- Bucket Locks
  - Locks on Buckets
  - Check for phantoms

Wait-for dependencies

- Eagerly update
- Incoming dependency: Wait on other transactions
- Outcoming dependency: Waited by other transactions
- Wait-for graph: Directed graph for deadlock detection

Normal Processing Phase

- Start Scan: A bucketlock is taken out to prevent phantom
- Check predicate
- Check Visibility: Record lock checking
- Read Version: Acquire locks
- Check updatability
- Update Version: Take out wait-for dependencies if the current version is locked
- Delete Version: Same as updating version
- Release locks.

Experiment Setup

- two-socket Intel Xeon X5650 @ 2.67 GHz (Nehalem) that has six cores per socket. Hyper- Threading was enabled. The system has 48 GB of memory, 12 MB L3 cache per socket, 256 KB L2 cache per core, and two separate 32 KB L1-I and L1-D caches per core.

Experimental Results

R=10 and W=2 in each transaction
Table with 10 million rows

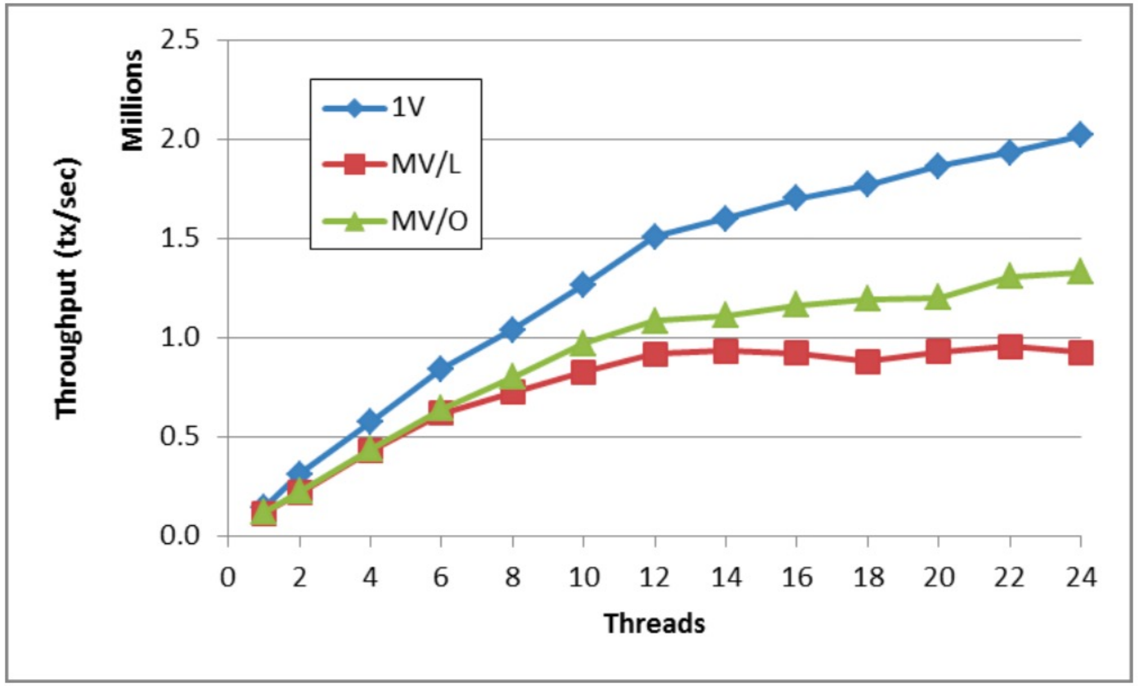


Figure 4: Scalability under low contention

R=10 and W=2 in each transaction
Table with 1000 rows

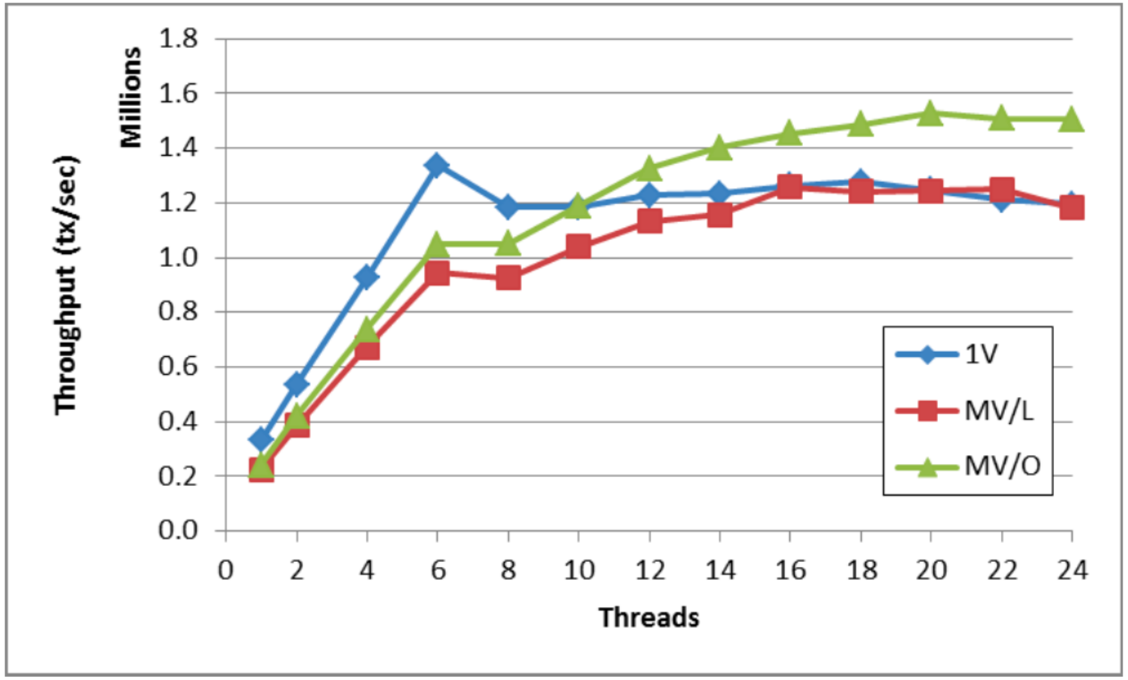


Figure 5: Scalability under high contention

Different isolation level

	Read Committed	Repeatable Read		Serializable	
	tx/sec	tx/sec	% drop vs RC	tx/sec	% drop vs RC
1V	2,080,492	2,042,540	1.8%	2,042,571	1.8%
MV/L	974,512	963,042	1.2%	877,338	10.0%
MV/O	1,387,140	1,272,289	8.3%	1,120,722	19.2%

Impact of Short Read Transactions

Fixed threads #: 24

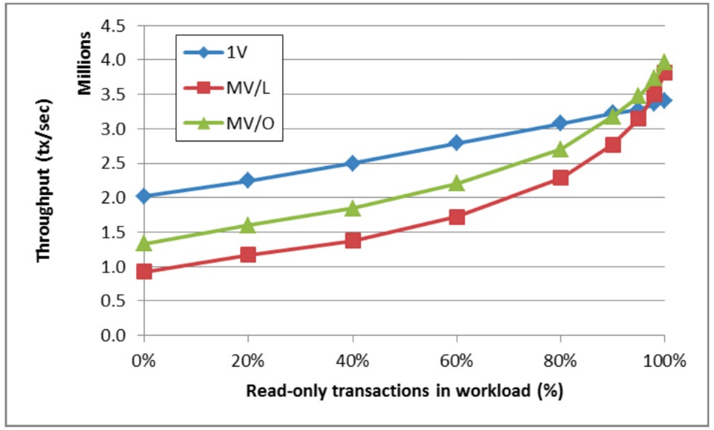


Figure 6: Impact of read-only transactions (low contention)

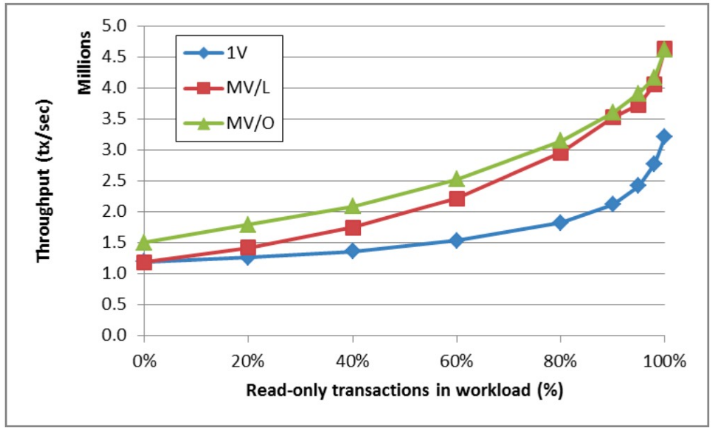


Figure 7: Impact of read-only transactions (high contention)

Impact of Long Read Transactions

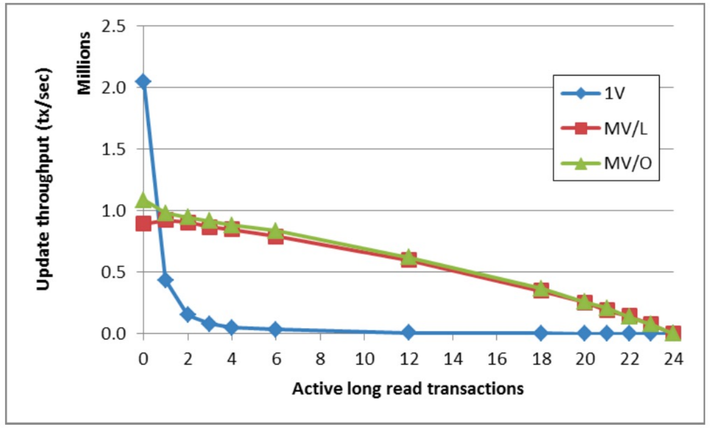


Figure 8: Update throughput with long read transactions

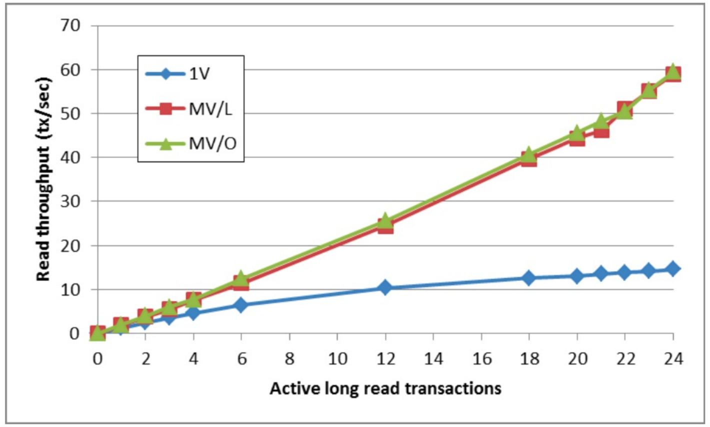


Figure 9: Read throughput with long read transactions

UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS

TATP Results

	1V	MV/L	MV/O
Transactions per second	4,220,119	3,129,816	3,121,494

Personal reflection

- No real-world evaluation of serialized level.
- Garbage collection can be a future direction

UNIVERSITY OF WATERLOO

FACULTY OF MATHEMATICS