OptiQL: Robust Optimistic Locking for Memory-Optimized Indexes

[SIGMOD 2024]

Ge Shi, Ziyi Yan, Tianzheng Wang
Simon Fraser University
https://github.com/sfu-dis/optiql

SFU SIMON FRASER UNIVERSITY

What? Optimistic locks are fast for read-mostly workloads, but not robust, i.e., can collapse under contention.

Why? Centralized design that necessitates writers to spin on one memory word.

How? Queue up writers and let them spin on a local memory location, while supporting optimistic reads.

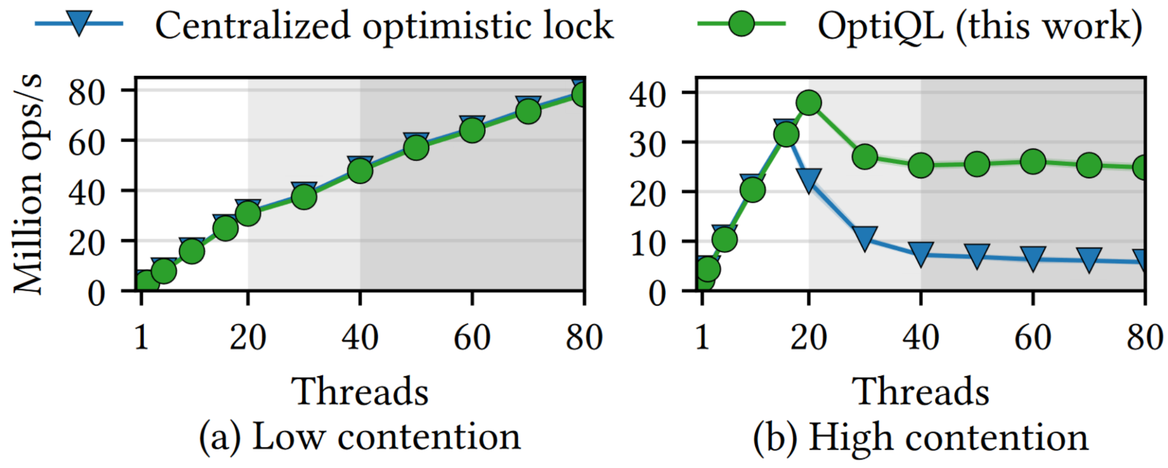
Memory-Optimized Indexes

- Indexes (e.g., B+-trees) need to be **thread-safe**
- **Fine-grained reader-writer locking*** (one lock per tree node)
- Need **optimistic lock coupling** during traversal and SMOs

* "Lock" == latch here in DBMS literature

Desirable:
(1) Fast Read, (2) Robust Against Contention, (3) Fair,
(4) Compact, (5) Easy-to-Adopt to Existing Index Locking

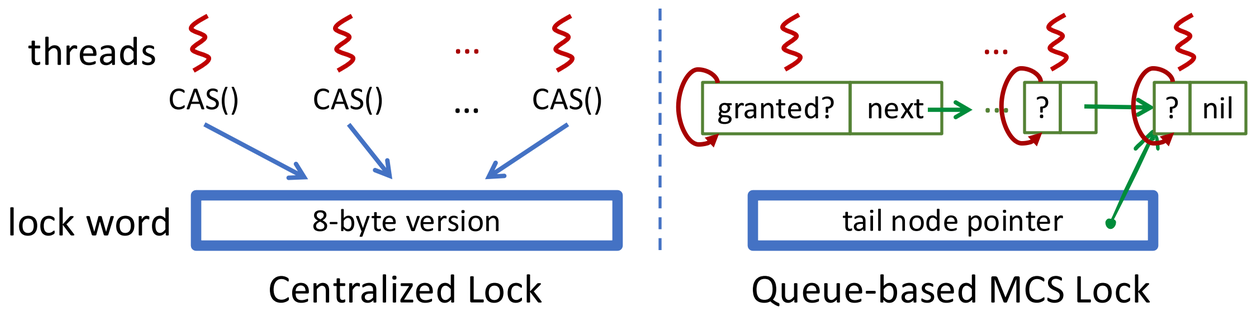
Prior Optimistic Locks: Fast, not Robust



Culprit: Centralized Spinning

Writers issue compare-and-swap (CAS) and spin on the lock

- CAS doesn't guarantee fairness or latency
- Lots of cycles are wasted on spinning
- Cache-coherency traffic floods interconnect



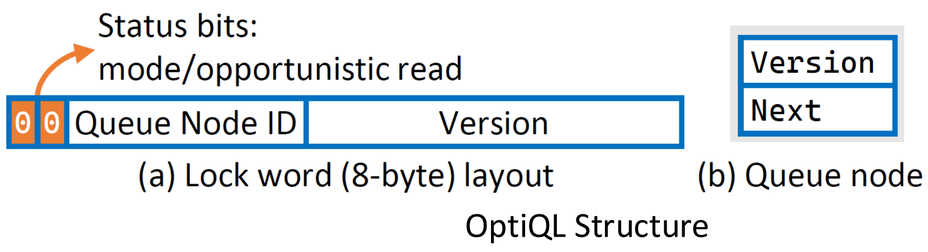
Lock Design Tradeoffs

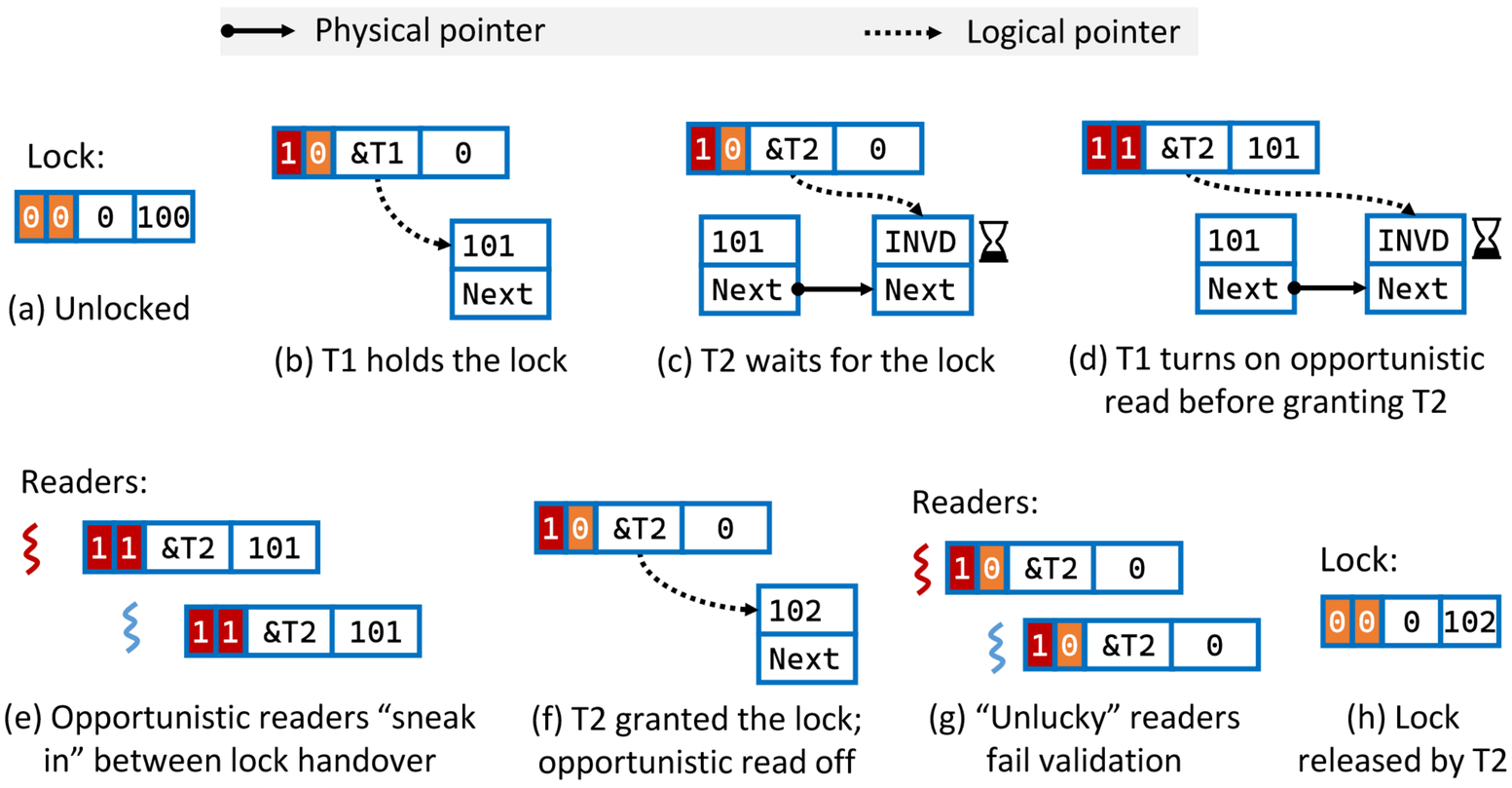
- Optimistic over Pessimistic for **cheap read**
- Queue-based over Centralized for **local spinning** and **First-Come-First-Serve fairness (for writers)**

	Pessimistic	**Optimistic**
Centralized	(Test-and-)test-and-set, traditional r/w locks	Optimistic locks
Queue-based	MCS locks	***OptiQL*** *Best of both*

OptiQL = Queue-based (MCS-like) Writers + Optimistic Readers

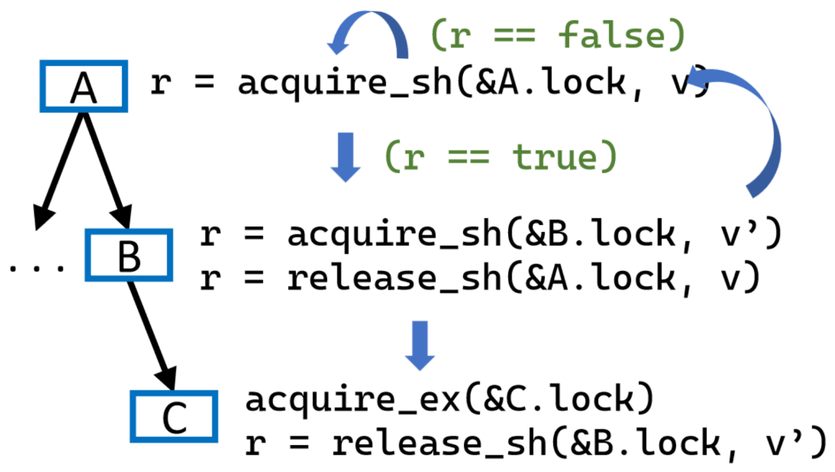
- Writer:
  - XCHG to add itself to the queue (a,b)
  - Pass along version number to next node
- Reader:
  - Lock: only read lock word
  - Unlock: check version number
  - Opportunistic read (d,e,f) to alleviate starvation





Optimistic Latch Coupling with OptiQL

- Reader: No changes in interface
- Writer: Slight changes to accommodate lock queue



Lock Coupling with OptiQL in B+-Tree Insertion

Performance

Dual-socket server:
- 2 x 20-core (80 HT) Intel Xeon Gold 6242R, 3.1GHz
- 387GB DRAM

Microbenchmarks:
- B+-tree and ART (not shown here)
- 80% updates, 20% reads
- Self-similar distribution (skew factor of 0.2)