ERMIA: Fast Memory-Optimized Database System for Heterogeneous Workloads

Kangnyeon Kim
University of Toronto

Tianzheng Wang

Ryan Johnson
LogicBlox

Ippokratis Pandis
Amazon Web Services

What? Heterogeneous workloads are coming, but existing MMDBMS isn’t good at them

Why? “Wrong” concurrency control in use: aborts too many read-mostly transactions

How? Fair and robust CC (snapshot isolation + certifier) + scalable physical layer

ERMIA = Snapshot Isolation + Serial Safety Net + Scalable Physical Layer

Fair and robust logical layer

1. Read-friendly snapshot isolation
   - Abort on write-write conflicts, preserves reads

2. Serializability with Serial Safety Net *
   - Cheap certifier on top of any CC >= RC (e.g. SI)
   - Maintains fairness and robustness


Scalable physical layer

1. Minimal global communication
   - One atomic-fetch-add per tx for global ordering
   - Eases implementation of snapshot isolation
   - Simplifies logging/recovery

2. Easy maintenance via indirection
   - Fast recovery, single-hop index update, etc.

Serial Safety Net

\[ \text{def: } c(T) = T’s \text{ commit time}, \pi(T) = \text{earliest successor} \]

- Forbid any pred P with \( \pi(T) \leq c(P) \leq c(T) \)

Commit order

\[ \text{Exclusion window} \]

Dependency order

- Private footprint until commit
- Carve out space via atomic-fetch-add
- Fill in log space asynchronously
- Abort \( \rightarrow \log \) discarded
- LSN offset = global order

Commit LSN (cLSN) = XADD(current LSN, log size)

Scalable centralized redo logging

- Central log (buffer):
  - Free space
  - T1 cLSN = T3 begin LSN
  - Log records
  - Install new version: CAS V1 \( \rightarrow \) V2
  - CAS failure \( \rightarrow \) WW conflict

Indirection array

- Versions in durable log

Robust to “convenient” & real workloads

- Object IDs (instead of pointers) at leaf level
- Updates: no index update needed

HW: 4-socket 6-core Intel E7-4807, 64GB RAM