Aria

A Fast and Practical Deterministic OLTP Database

Yi Lu, Xiangyao Yu, Lei Cao, Samuel Madden
Background

- **Deterministic** = transactions across replicas without coordination
  - Every node gets the same result individually

- **Traditional methods** require reads and writes to be known
  - All locks acquired before execution
  - Works well for simple transactions
  - Multiple executions for more complex transactions
Solution

- Two phases: Execution and commit
  1. Execute deterministically with no dependency analysis
  2. Find conflicts during execution
  3. Commit transactions that have no conflicts
  4. Move conflicts to the next batch of transactions
Execution phase

- Execution is done one batch at a time
- Each transaction reads from the same snapshot of the database
- Transactions make reservations for their writes
  - Aborting if a reservation with a lower ID exists

- T1: $x = x + 1$, T2: $y = x - y$, and T3: $x = x + y$

- T1 makes a reservation for $x$
- T2 makes a reservation for $y$
- T3 tries to make a reservation for $x$, but because of T1 it must abort
- T3 is pushed into the next batch
Commit phase

- There are two dependencies that can make a transaction abort
  - Write-after-write (WAW)
  - Read-after-write (RAW) (since the second transaction should have seen the write)

- Write-after-read (WAR) avhengigheter er ikke noe problem

- Aborted transactions will be pushed to the front of the next batch
  - Since the first transaction of a batch always commits, this ensures that all transactions eventually commits
Deterministic reordering

- Reduce conflicts in commit phase
- Transforming RAW-dependencies to WAR-dependencies

- T1: y = x, T2: z = y, and T3: Print y + z
- By reordering the transactions, they can all commit
The fallback phase

- Another phase is used for batches that suffers from WAW-conflicts
- Read/write sets are known after the commit phase
- Approach similar to Calvin
- Moving average of abort rate used to activate the phase
Resultater
Throughput

Figure 9: Performance on YCSB-A and YCSB-B
Scheduling overhead

(c) Scheduling vs. Execution
Effectiveness of reordering
Distributed transactions

Figure 13: Performance on YCSB and TPC-C in the multi-node setting
Conclusions

“Aria outperforms systems with conventional nondeterministic concurrency control algorithms and state-of-the-art deterministic databases by a large margin on a single node and up to a factor of two on a cluster of eight nodes.”