MRVs: Enforcing Numeric Invariants in Parallel Updates to Hotspots with Randomized Splitting

Nuno Faria, José Pereira | nuno.f.faria@inesctec.pt, jop@di.uminho.pt | INESCTEC & University of Minho

SIGMOD 2023

Motivation

- Transactional conflicts greatly impact the performance of operational database systems, especially distributed ones.
- Numeric hotspots are one of the most common causes of such conflicts.
- Existing solutions are limited in the concurrency allowed, adaptability to dynamic workloads, and/or ensuring bound invariants (e.g., $x \geq 0$).

Multi-Record Values (MRVs)

- MRVs alleviate this problem by converting a value into multiple physical records through **randomized splitting**, allowing updates to execute concurrently without conflict.
- They are **dynamically adjusted** to load to improve write performance while optimizing for read and storage overheads. They also ensure **bound invariants**, often needed in financial and logistical workloads.

Operations

- ***lookup(pk, rk')*** – looks up the record of MRV pk with the minimum rk such that $rk \geq rk'$, or the record with the minimum rk if none exists.
- ***add(pk, δ)*** – adds δ to MRV pk ($lookup$ + $update$).
- ***sub(pk, δ)*** – removes δ to MRV pk. ($lookup$ + $update$; if the value in the record is not enough, carries the remaining to the next and repeats).
- ***read(pk)*** – materializes the value of MRV pk.
- ***write(pk, v)*** – sets the value of MRV pk to v.

Implementation Strategies

- Directly in the **storage engine**, using low-level code.
- In the **query engine**, using views for reads and rules/triggers for writes.
- At the **application-level**, e.g., in the database driver library.

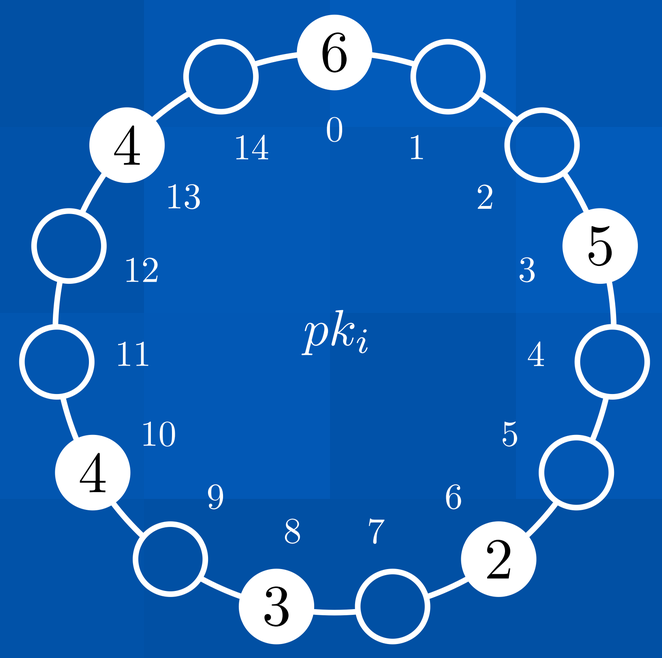

Figure 1: Structure of an MRV pk_i with 24 units.

Structure

- Logically represented by a ring of size N.
- Each MRV contains at most N records.
- Each record in an MRV is assigned a unique integer $rk \in [0, N]$.

Background Workers

- **Adjust** the number of records based on the workload.
- **Balance** the amounts to keep the number of subtract lookups low.

These strategies make MRVs feasible in SQL, NoSQL, centralized, distributed, and even closed-source database systems.

Selected Results

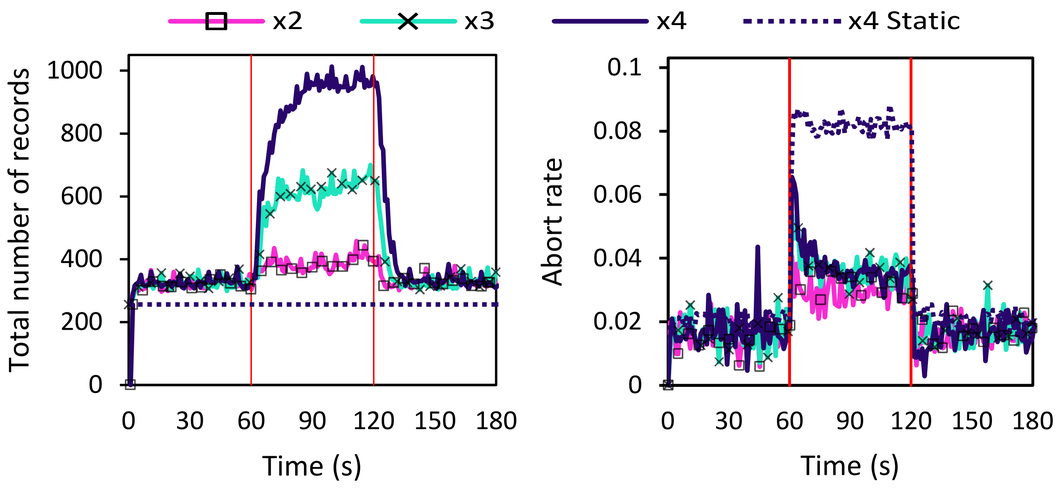

Figure 2: MRVs adaptability with variable load increases between 60 and 120 seconds (2x, 3x, and 4x). MRVs without dynamic adjusting are also depicted (Static). 256 total MRVs. Target abort rate = 5%.

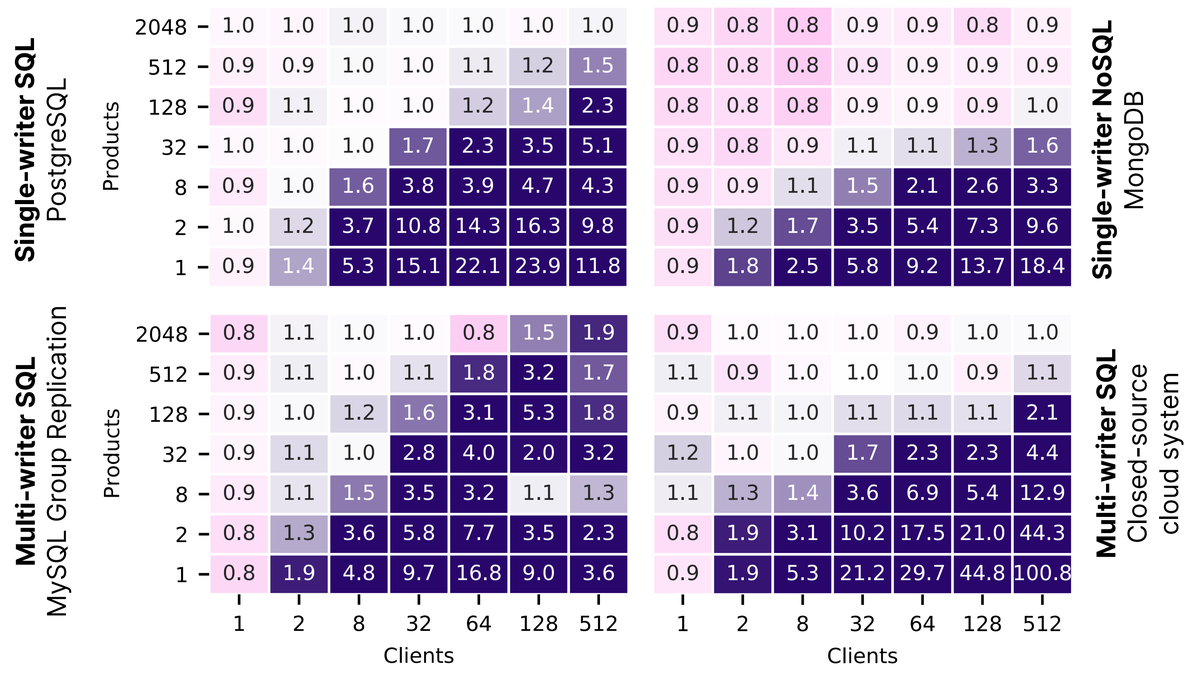

Figure 3: Scale up in throughput of MRVs vs native single-record solution in various database system architectures.

Partially funded by project AIDA -- Adaptive, Intelligent and Distributed Assurance Platform (POCI-01-0247-FEDER-045907) co-financed by the European Regional Development Fund (ERDF) through the Operational Program for Competitiveness and Internationalization (COMPETE 2020) and by the Portuguese Foundation for Science and Technology (FCT) under CMU Portugal.

Scan to read the full paper.