Persian: A Fast Checkpointing Method Based on Concurrent Prefix Recovery

Lijia Jiang Hideyuki Kawashima
Keio University, Japan
rika.shou@keio.jp, river@sfc.keio.ac.jp

Abstract

Modern key-value stores and in-memory databases achieve extremely high throughput on multi-core machines but still face a major bottleneck in durability. Concurrent Prefix Recovery (CPR) provides asynchronous checkpointing to ensure durability without blocking ongoing transactions, but it still suffers from redundant data copying and synchronization costs. We propose **Persian**, a lightweight and parallelized extension of CPR that removes the stable-copy phase and introduces concurrent flushing. Persian reduces both memory and CPU overhead while preserving CPR's consistency guarantees. Experiments show that Persian achieves up to 2.3× higher throughput under read-dominant workloads, demonstrating a better balance between performance and reliability.

1 Motivation

Modern data-intensive systems must guarantee both high performance and reliability even in the presence of hardware or software failures. To achieve this, database engines implement mechanisms that ensure the *durability* aspect of the ACID properties, allowing committed data to survive crashes or power losses. Traditional approaches such as Write-Ahead Logging (WAL) ensure correctness by persisting every update before commit, but this design inherently requires serial log writes, which form a central bottleneck and limit scalability on multi-core architectures.

To address this limitation, *Concurrent Prefix Recovery (CPR)* was proposed as part of the FASTER key-value store. CPR allows transactions to continue execution while consistent checkpoints are asynchronously generated in the background, providing durability without global coordination barriers. However, as workloads scale to dozens of cores and the size of in-memory logs increases, even background checkpointing can introduce noticeable performance degradation—particularly during the record-copy and synchronization phases, where threads compete for shared resources.

2 Problem

Although CPR enables durability without halting transaction processing, it still incurs significant overhead in practice. Two major limitations reduce its scalability:

- **Memory Overhead:** CPR maintains both live and stable copies of records. Duplicating large memory regions during every checkpoint leads to excessive memory consumption and increased garbage collection pressure.
- **Lock Contention:** During the copy phase, threads synchronize to copy stable versions of records, creating contention between active worker threads and checkpoint threads. This synchronization cost becomes dominant under high concurrency.

3 Proposal

We propose **Persian**, a fast and lightweight checkpointing method that builds upon CPR but eliminates its most expensive operations. Instead of generating a stable copy for each record, Persian performs an in-memory scan of valid records during the *IN_PROGRESS* phase and directly flushes them to persistent storage in the *WAIT_FLUSH* phase.

Key Design Principles

- **No stable copy:** Persian skips the stable-version creation step, directly scanning the live log in memory. This reduces memory footprint and avoids redundant writes.
- **Lock-free pipeline:** By decoupling checkpoint collection from transaction updates, Persian eliminates the need for fine-grained synchronization.
- **Parallel flushing:** Checkpoint data is flushed in parallel
- **Consistency verification:** Each checkpoint includes version metadata to validate that recovered states preserve prefix consistency without replay conflicts.

Workflow

Persian retains CPR's four-phase design but optimizes internal behavior for concurrency:

1. **PREPARE** – Initialize metadata, record the current version, and determine checkpoint boundaries.
2. **IN_PROGRESS** – Scan in-memory data to collect valid (live) records. Skip invalid or overwritten entries.
3. **WAIT_FLUSH** – Flush the collected records to persistent storage asynchronously and in parallel. Verify consistency.

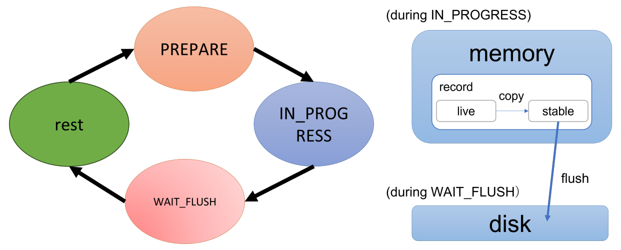

Figure 1: CPR checkpoint cycle and Persian optimization. Persian eliminates the stable-copy step and performs parallel flushing during *WAIT_FLUSH*, reducing synchronization and memory overhead.

4. **REST** – Finalize metadata and return to normal execution once all threads complete flushing.

Persian's checkpointing model preserves CPR's correctness while significantly improving efficiency by skipping unnecessary data duplication. The approach also enables fine-grained control over flush granularity, allowing a balance between consistency and I/O cost.

4 Evaluation

Workload: We evaluated Persian and CPR on a read-dominant workload (R90/W10) across 1–64 threads. Each configuration performed periodic checkpoints every 2 seconds to simulate sustained durability requirements.

Metrics: Throughput (operations per second) and recovery correctness were measured.

Results:

- **Scalability:** Persian maintains near-linear throughput growth as the number of threads increases, demonstrating strong scalability under read-dominant workloads.
- **Performance gain:** Persian achieves more than double the throughput of CPR, showing clear advantages in high-concurrency environments.
- **Consistency:** Verification confirmed that Persian preserves prefix-consistent recovery identical to CPR.

5 Conclusion

We presented **Persian**, a fast checkpointing technique derived from Concurrent Prefix Recovery. By removing the stable-copy phase and introducing multi-threaded flushing, Persian reduces memory footprint and synchronization overhead while maintaining prefix consistency. Our evaluation demonstrates significant throughput improvements under read-heavy workloads. Persian bridges the gap between performance and consistency, and we plan to extend it to distributed and hybrid memory systems in future work.

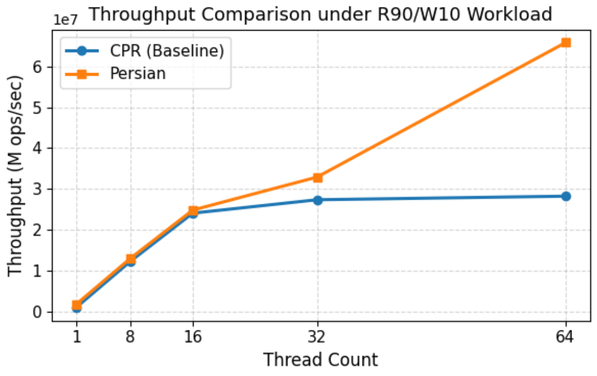

Figure 2: Throughput scalability under R90/W10 workload. Persian achieves over 2× higher throughput compared to CPR while maintaining consistent performance growth.

Acknowledgments

This paper is based on results obtained from the project "Research and Development Project of the Enhanced Infrastructures for Post-5G Information and Communication Systems (JPNP20017) " and JPNP16007 commissioned by the New Energy and Industrial Technology Development Organization (NEDO), and from JSPS KAKENHI Grant Number 25H00446, and from JST CREST Grant Number JPMJCR24R4 and from SECOM Science and Technology Foundation and JST COI-NEXT SQAI (JPMJPF2221), JST Moonshot RD Grant Number JPMJMS2215.

References

[1] G. Prasaad, B. Chandramouli, and D. Kossmann, "Concurrent Prefix Recovery: Performing CPR on a Database," in *Proc. SIGMOD*, 2019.