Persian: A fast checkpointing based on concurrent prefix recovery

Lijia Jiang, Hideyuki Kawashima / Keio University

1. Introduction

Recovery and ACID Properties

In modern key-value stores, systems must recover quickly after a crash to avoid data loss and ensure reliability.

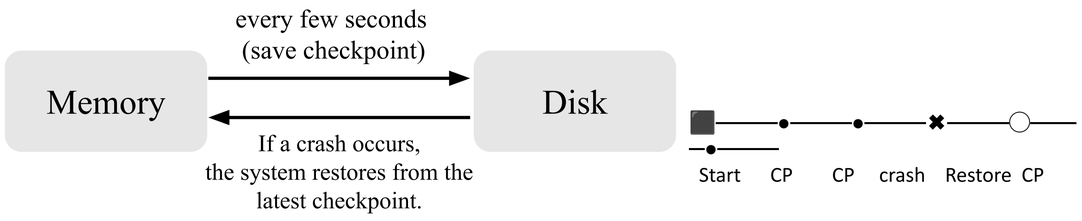
To achieve this, transactions follow the ACID properties. A transaction is a group of operations that must succeed or fail together.

Atomicity (all-or-nothing)	Consistency (data integrity)	Isolation (independent from others)	Durability (permanent once completed)

Checkpointing (CP)

Checkpointing is the main mechanism used to achieve Durability. It periodically saves in-memory data to persistent storage (disk).

When a crash occurs, the system restores from the latest checkpoint, reducing recovery time.



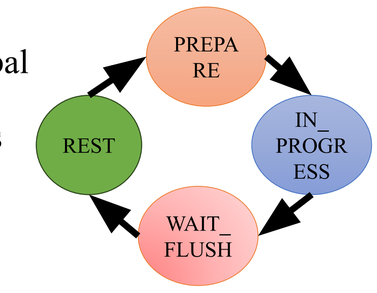
2. Conventional Method: CPR

Design

Concurrent Prefix Recovery (CPR) is a checkpointing technique designed to ensure Durability with minimal transaction interruption.

It runs concurrently with transactions and follows four phases :

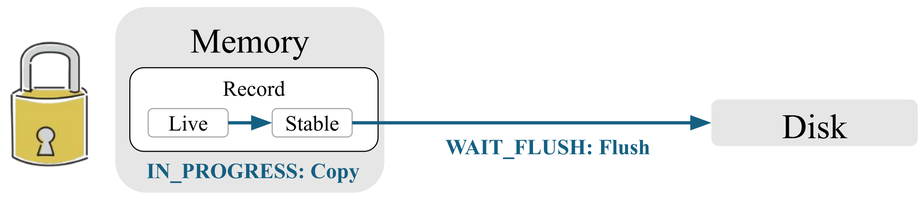
1. *PREPARE*: Initializes metadata and assigns a global checkpoint version.
2. *IN_PROGRESS*: Scans the in-memory log, copies valid records to a stable version.
3. *WAIT_FLUSH*: Flushes the stable version to disk asynchronously.
4. *REST*: Finalizes metadata and cleans up.



CPR enables asynchronous checkpointing, allowing transactions to proceed without pausing for the entire duration.

Challenge: Stable Copy Overhead

- Creating a stable copy incurs memory overhead.
- The process of scanning and copying records also introduces locking cost.



Research Question

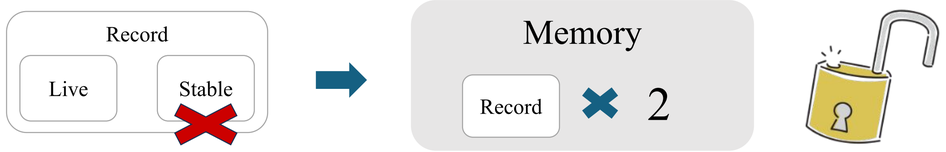
How to reduce checkpointing overhead?

3. Proposed Method: Persian

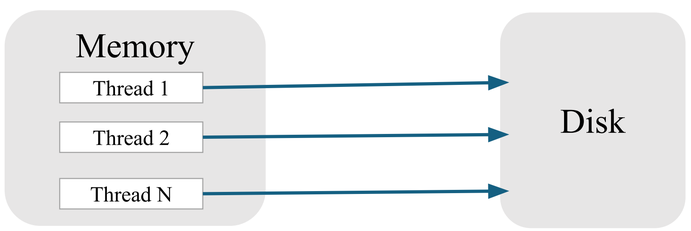
Persian removes the stable copy phase of CPR. It directly scans the in-memory log in the IN_PROGRESS phase, collects valid records, and flushes them to storage in parallel during WAIT_FLUSH.

This approach avoids locking and reduces memory usage.

Phase	CPR	Persian
IN_PROGRESS	Stable Copy	Eliminate Stable Copy ⇨ Save Memory ⇨ Remove Locks (Less Contention) ⇨Version Filter *(v-only scan)* ⇨ Prepare ensures consistency
WAIT_FLUSH	Single-thread Execution	Multi-thread Execution ⇨ Fast Flush & Scalable



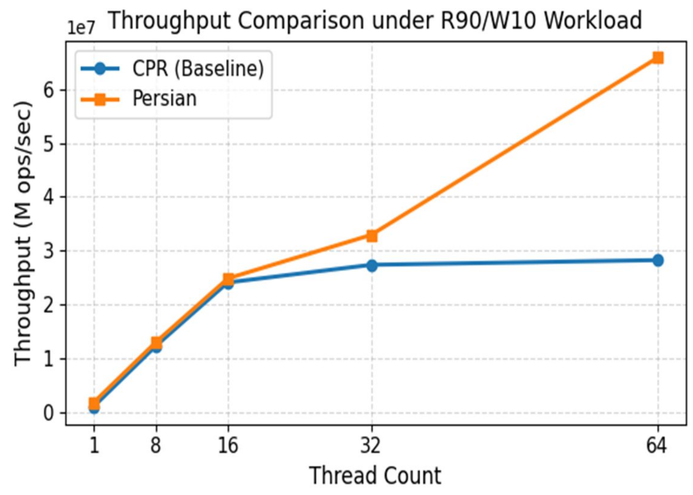
Persian reconstructs the version-v by scanning the log prefix and filtering records by version.



Hypothesis

Persian performs well under read-heavy workloads.

4. Results (YCSB-like Read Heavy)



Workloads:
90% Read /
10% Write

As the thread gets higher…

Throughput improves by up to 2× higher

Persian scales better under read-heavy loads, while CPR is slowed down by copying.

5. Conclusion

We proposed Persian to reduce CPR's overhead in read-heavy, high-concurrency workloads.

By eliminating stable copies and flushing in parallel, Persian avoids locking, achieving up to 2x higher throughput under read-dominant scenarios.

Acknowledgment

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 R&D Grant Number JPMJMS2215.