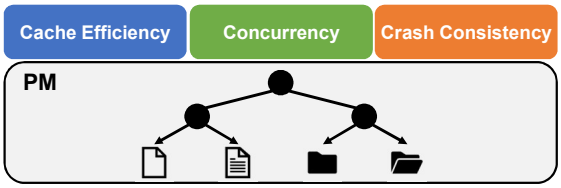
RECIPE : Converting Concurrent DRAM Indexes to Persistent-Memory Indexes

Se Kwon Lee[1], Jayashree Mohan[1], Sanidhya Kashyap[2], Taesoo Kim[2], Vijay Chidambaram[1,3]

Introduction and Motivation

- **Persistent Memory (PM)**: low-latency, persistent, high-capacity storage with load-store interface
- **PM Index Structures**: Crucial for efficient data access (100 billion 64-byte key-value pairs on a single node)

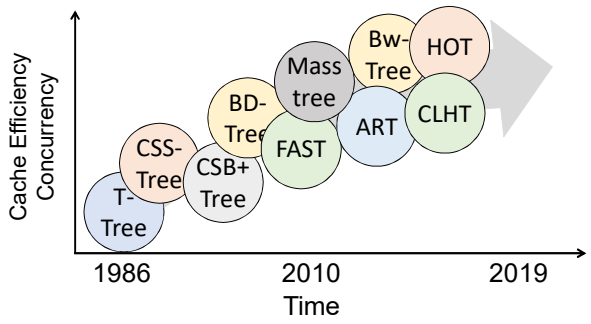


Concurrent Crash-Consistent PM Index

- **Correct Concurrency**: Return consistent data regardless of the conflicts between multiple threads
- **Correct Crash Consistency:** Return consistent data irrespective of an unexpected crash
- **Challenge**: Concurrency and crash consistency interact with each other, a bug in either can lead to data loss
- **5 Bugs** in two state-of-art PM Indexes
  3 bugs in FAST&FAIR (Concurrent PM B+Tree)
  2 bugs in CCEH (Concurrent PM hash table)

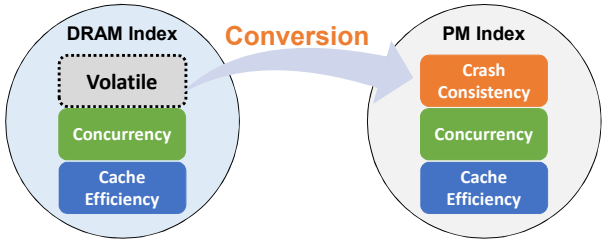
DRAM Index

- **DRAM Index**: Already designed for cache efficiency and concurrency in the last three decades
- DRAM Index on PM: Cache efficient (✓) Concurrent (✓) Crash Consistent (X) → **Crash Vulnerable**



The RECIPE Approach

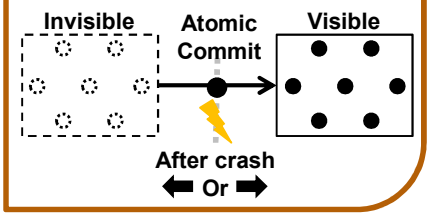
- Convert concurrent DRAM indexes into PM indexes, instead of building PM indexes from scratch
- **Challenge**: Minimal changes to DRAM indexes without modifying original design principles
- **Insight**: Isolation and crash consistency are similar
- **Non-blocking algorithm**: Detect (✓) Tolerate (✓) Fix (✓) Helping mechanism → **Inherently crash consistent**



Three Conversion Conditions

Condition #1

- Update via Single Atomic Store
- **Conversion**: Adding flushes after each store and bind final atomic store using fences

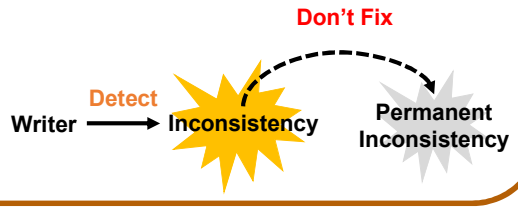


Condition #2

- Writers fix inconsistencies
  Detect (✓) Tolerate (✓) Fix(✓)
- Non-blocking readers and writers
- **Conversion**: Adding flushes & fences after each store and specific loads

Condition #3

- Writers don't fix inconsistencies
  Detect (✓) Tolerate (✓) Fix(X)
- Non-blocking readers and Blocking writers
- **Conversion**: Adding helping mechanism

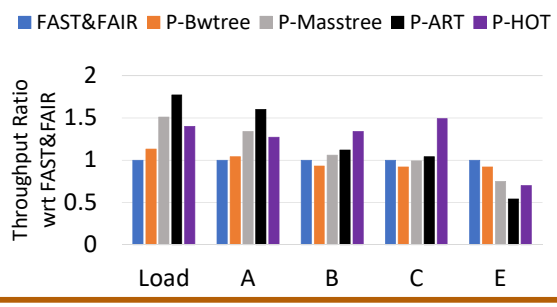


Conversion Results

- Convert five popular DRAM indexes

Indexes	DS Types	LoC	
		Core	Modified
P-CLHT	Hash Table	2.8K	30 (1%)
P-HOT	Trie	2K	38 (2%)
P-BwTree	B+Tree	5.2K	85 (1.6%)
P-ART	Radix Tree	1.5K	58 (3.4%)
P-Masstree	Hybrid (B+Tree, Trie)	2.2K	200 (9%)

YCSB
Random Integer Keys, Ordered Indexes



Summary

- Cache-efficient designs of P-Indexes has become a major source of high performance
- Up-to 1.6X better with random integer keys
- Up-to 5.2X better with string keys

https://github.com/utsaslab/RECIPE