RICE

SoftWrAP: A Lightweight Framework for Transactional Support of Storage Class Memory

Ellis Giles
Rice University
Houston, Texas
erg@rice.edu

Kshitij Doshi
Intel Corp.
Portland, OR
kshitij.a.doshi@intel.com

Peter Varman
Rice University
Houston, Texas
pjv@rice.edu

Massive Storage Systems and Technologies Conference, 2015

RICE

Motivation

- In Memory Databases (IMDB) & Caches
  - Main Memory Databases – MMDB
  - solidDB, Oracle TimesTen, CSQL, Memcached
  - Fast Access Speeds
  - Lack Durability (ACID) or must log to disk with long recovery times
- New Graph Databases, Neo4j, Graph 500
- Ideally, with byte-addressable persistent storage, applications could operate at in-memory access speeds without having to log to disk.

Storage Technology

Memory
- Fast
- Byte Addressable
- Volatile
- DRAM –Refreshed

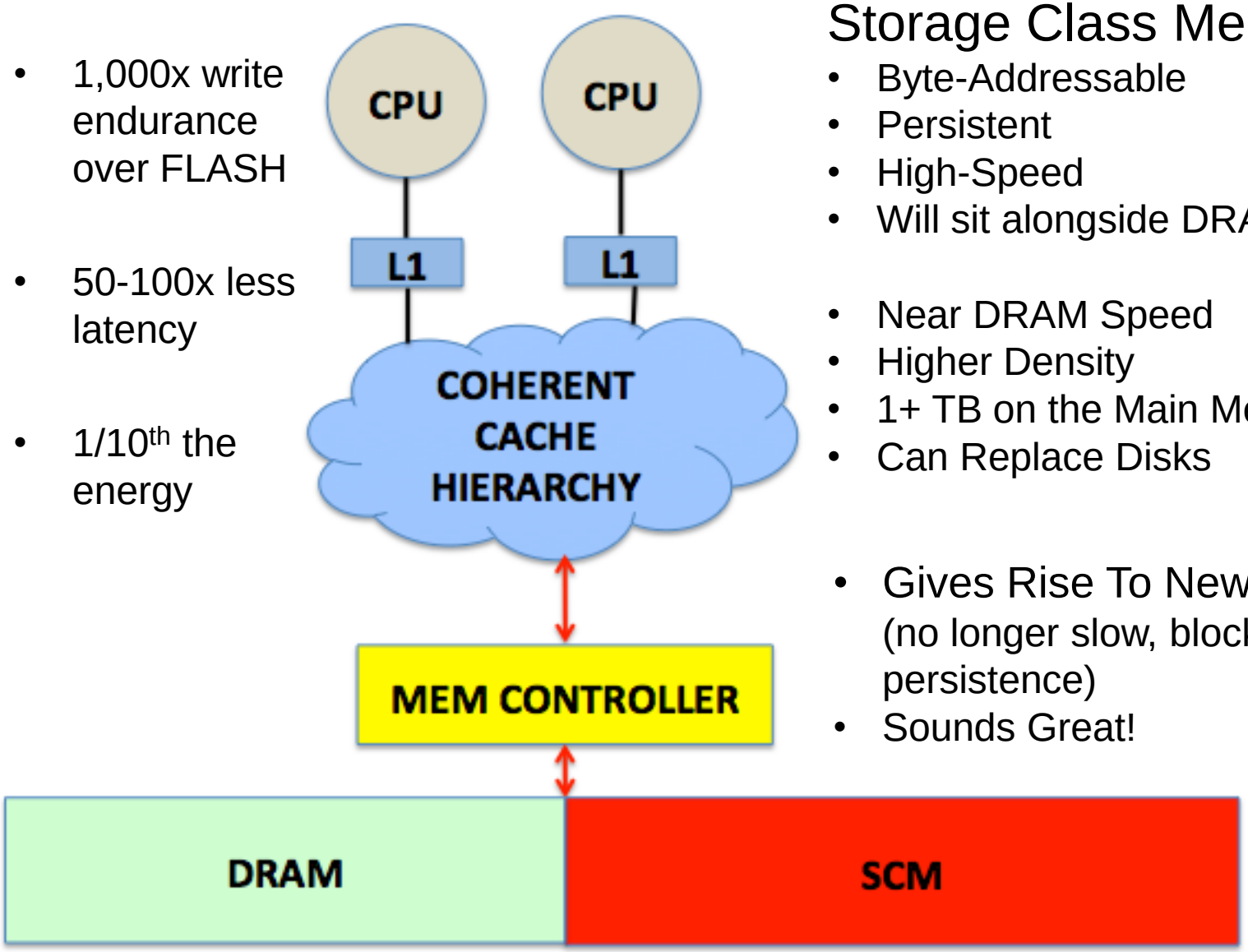
Storage Class Memory
- Fast
- Byte Addressable
- Non-Volatile
- Examples:
  - PCM
  - ReRAM
  - ST-MRAM

Storage
- Non-Volatile
- Slow
- Block-Based

**Figure Adapted from: M. K. Qureshi, V. Srinivasa, and J. A. Rivers, "Scalable high performance main memory system using phase-change memory technology," ISCA'09.*

Storage Class Memory

- 1,000x write endurance over FLASH
- 50-100x less latency
- 1/10th the energy



Storage Class Memory (SCM):

- Byte-Addressable
- Persistent
- High-Speed
- Will sit alongside DRAM

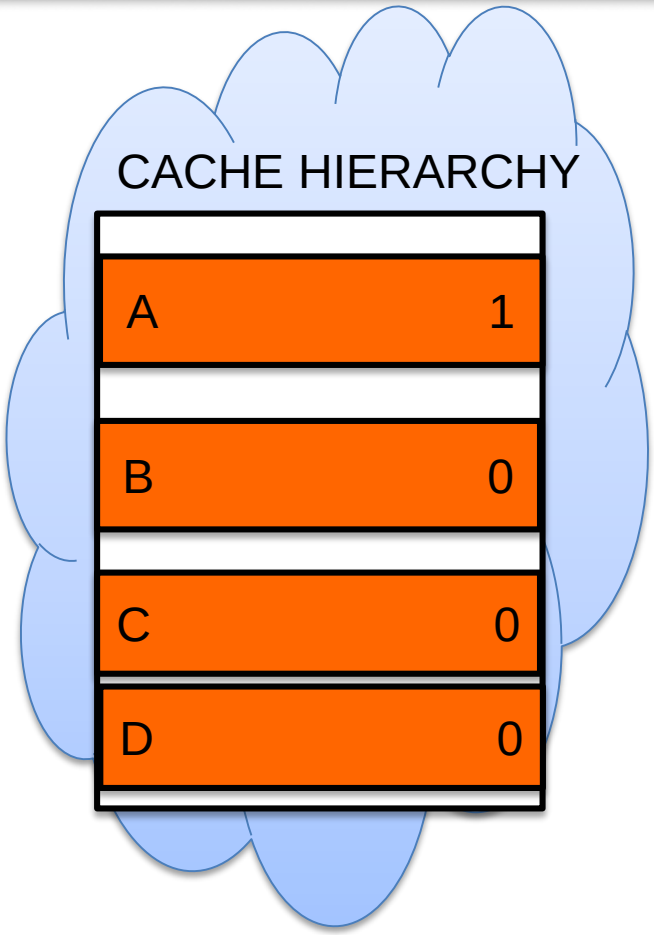
- Near DRAM Speed
- Higher Density
- 1+ TB on the Main Memory Bus
- Can Replace Disks

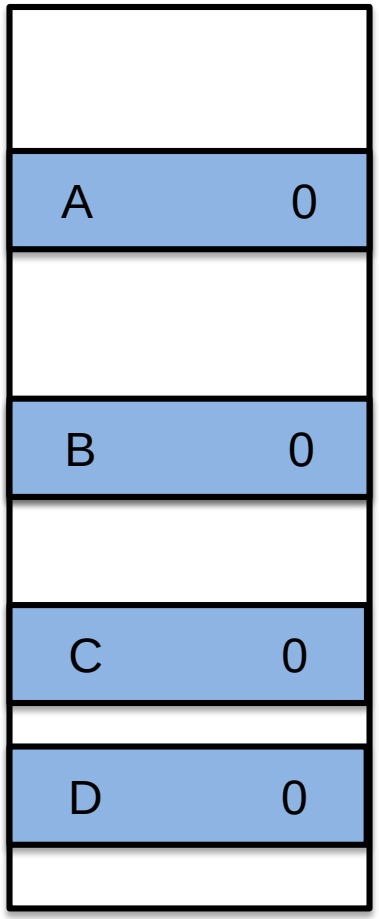
- Gives Rise To New Applications (no longer slow, block based persistence)
- Sounds Great!

Problem

Transaction

STORE A, 1

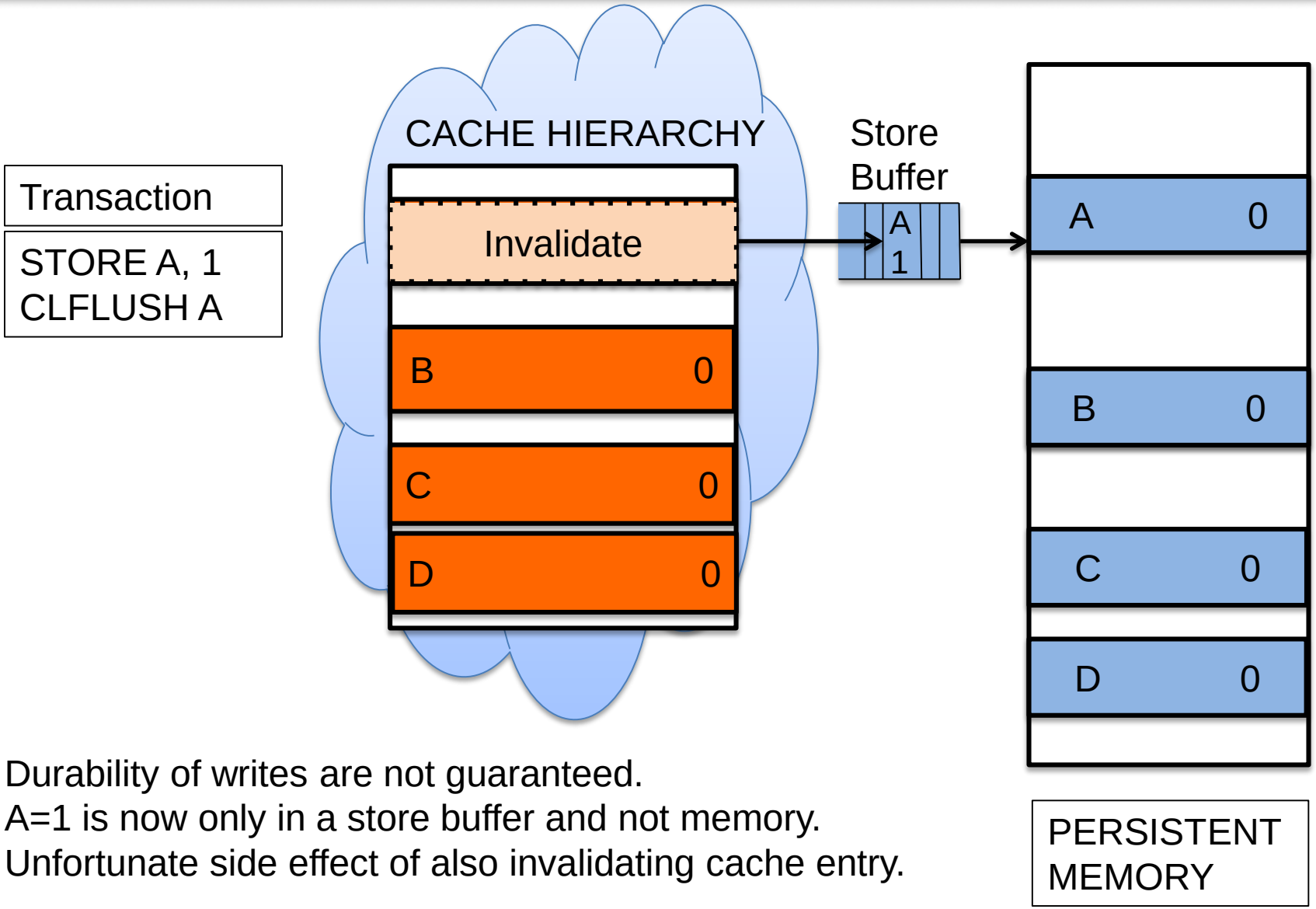




PERSISTENT MEMORY

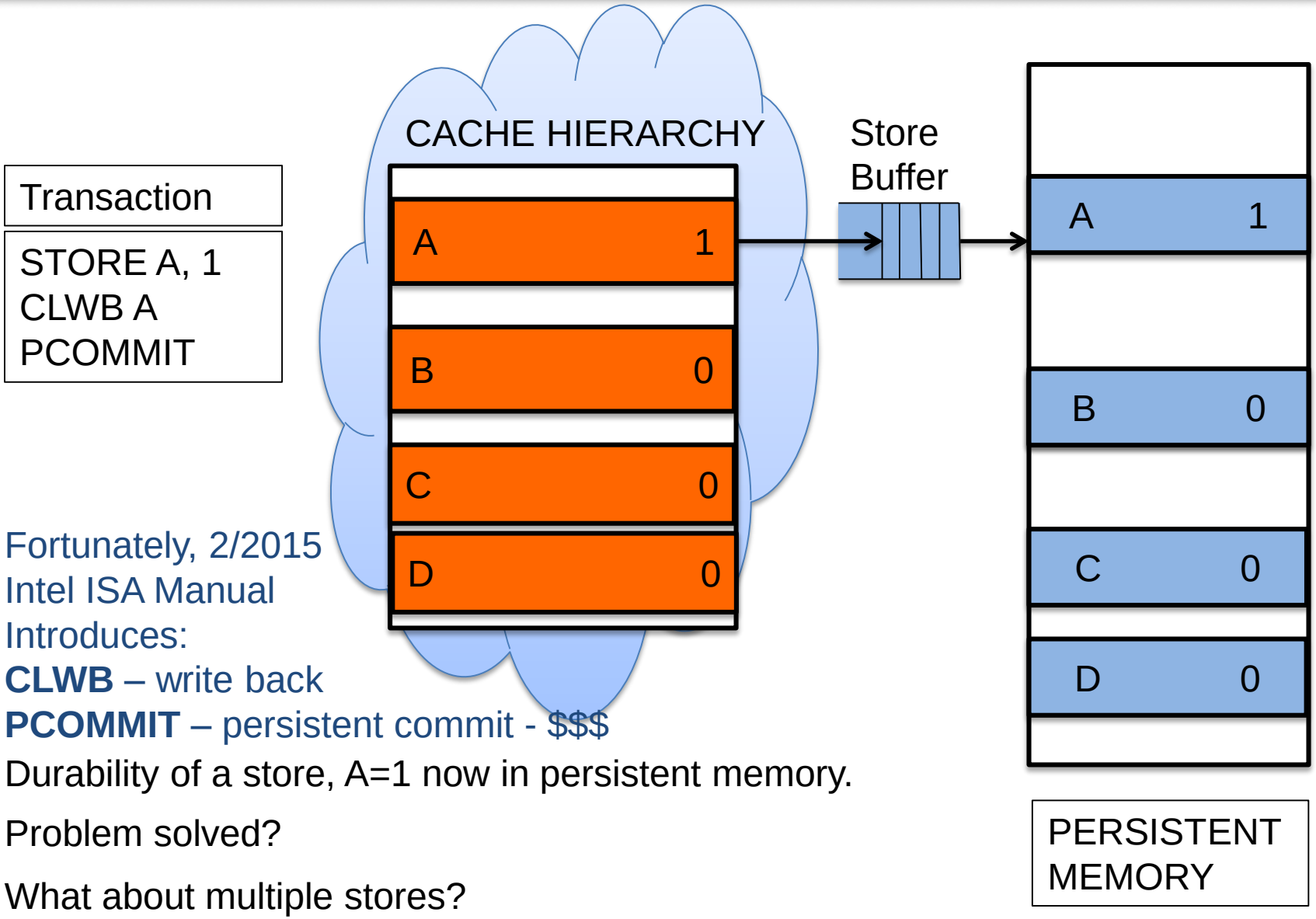
Durability of writes (even a single write) are not guaranteed.
A=1 is only in cache hierarchy and not in memory.

Problem



Durability of writes are not guaranteed.
A=1 is now only in a store buffer and not memory.
Unfortunate side effect of also invalidating cache entry.

Problem



Fortunately, 2/2015 Intel ISA Manual Introduces:

CLWB – write back

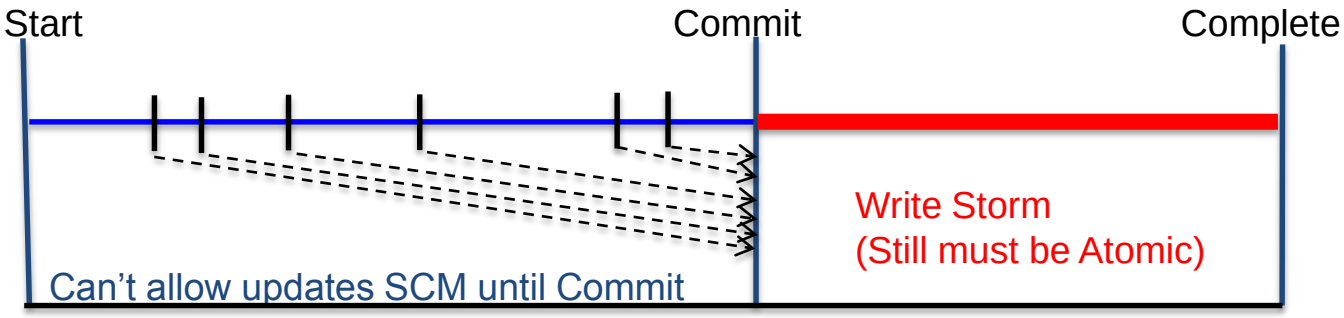
PCOMMIT – persistent commit - $$$

Durability of a store, A=1 now in persistent memory.

Problem solved?

What about multiple stores?

Problem



Writes Must be Atomic Or crash during Write Storm would leaved data inconsistent.

Reduces window of vulnerability.

Option 1:

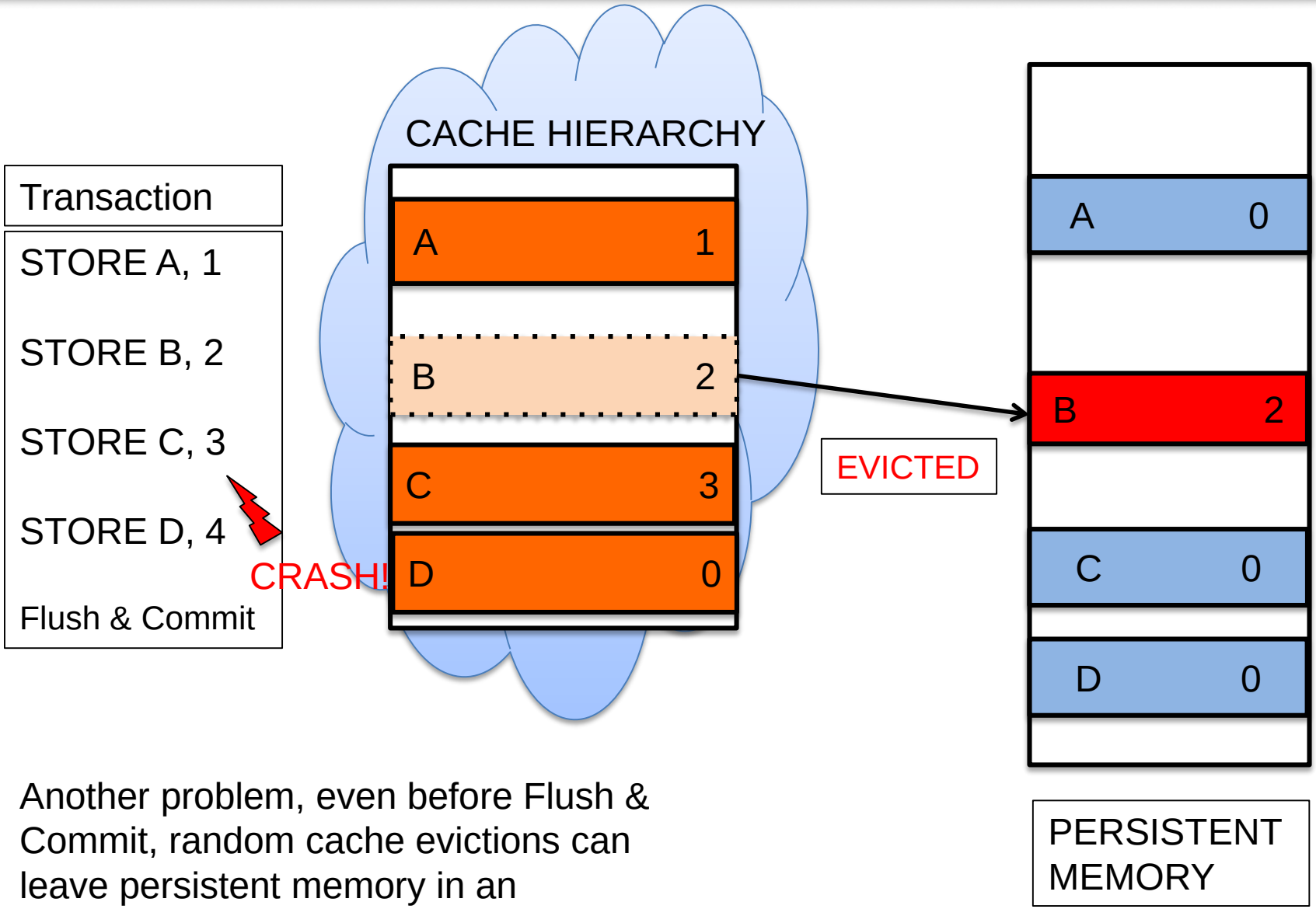
Transaction
STORE A, 1 CLWB PCOMMIT STORE B, 2 CLWB PCOMMIT STORE C, 3 CLWB PCOMMIT STORE D, 4 CLWB PCOMMIT

CRASH!!
Data Is Inconsistent

Option 2:

Transaction
STORE A, 1 STORE B, 2 STORE C, 3 STORE D, 4 Flush & Commit

Problem



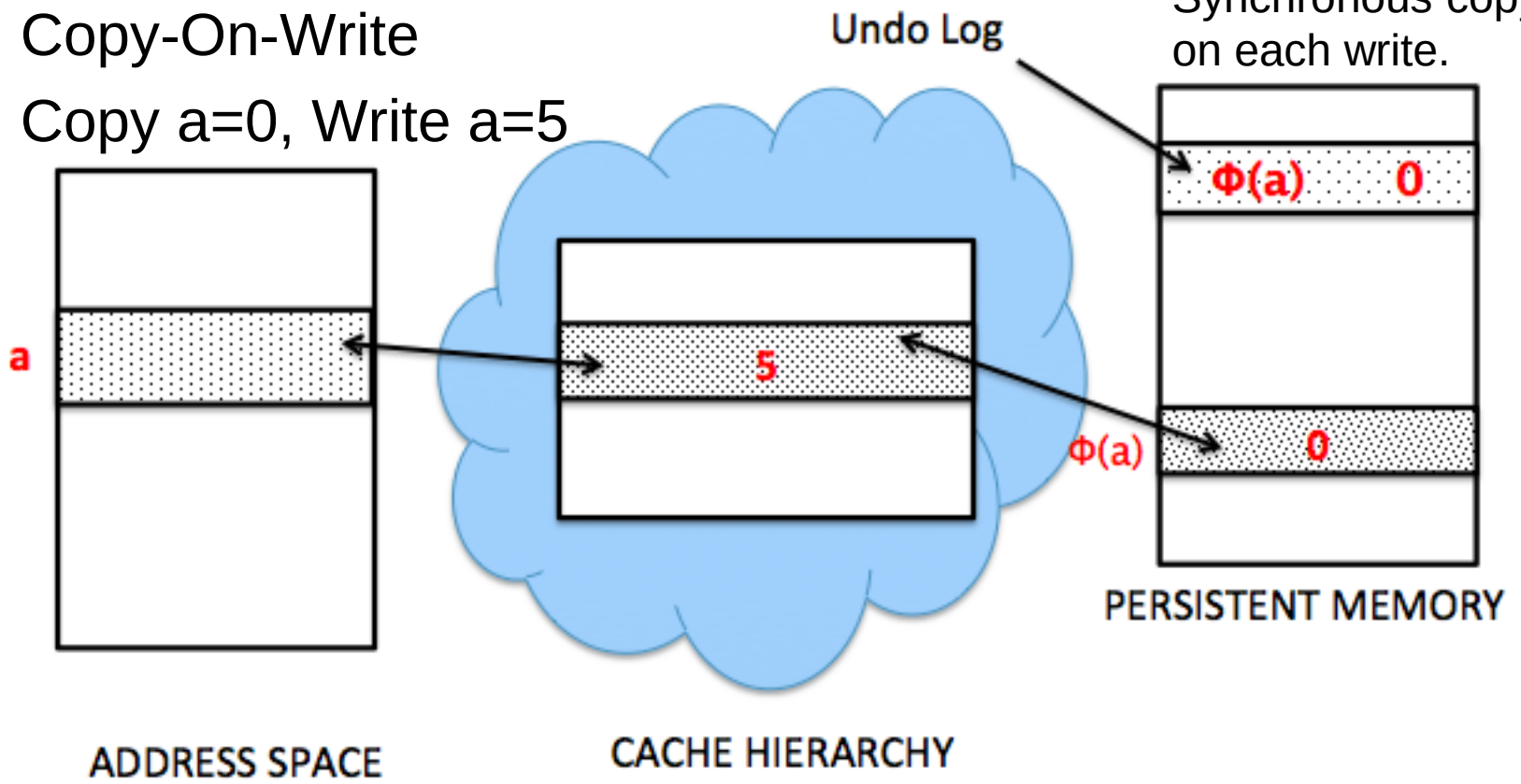
Another problem, even before Flush & Commit, random cache evictions can leave persistent memory in an inconsistent state.

Undo Log

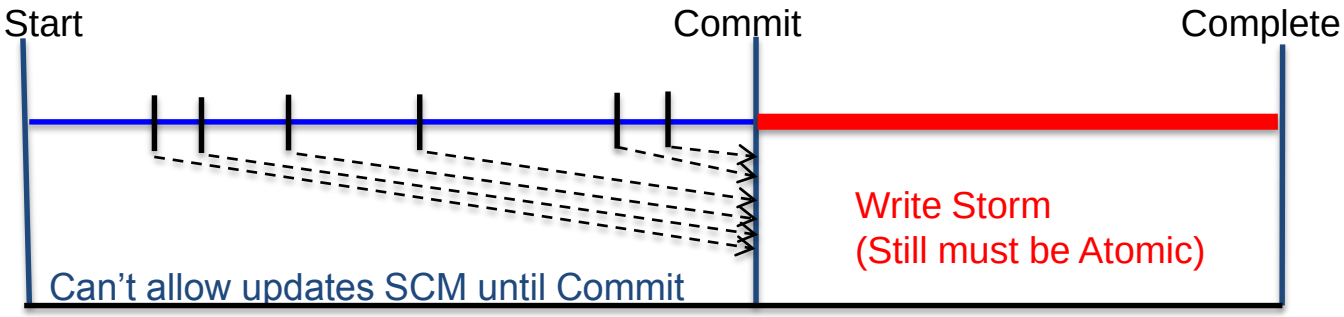
- One Solution:
  - Copy old value before writing new value
- Recovery from failure use undo log
- Copy-On-Write
- Copy a=0, Write a=5

Must flush and commit copy before writing new value.

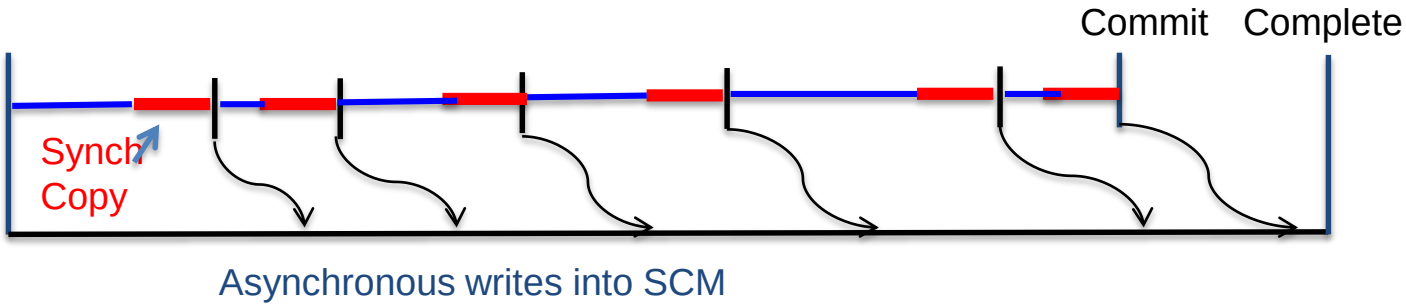
Synchronous copy on each write.



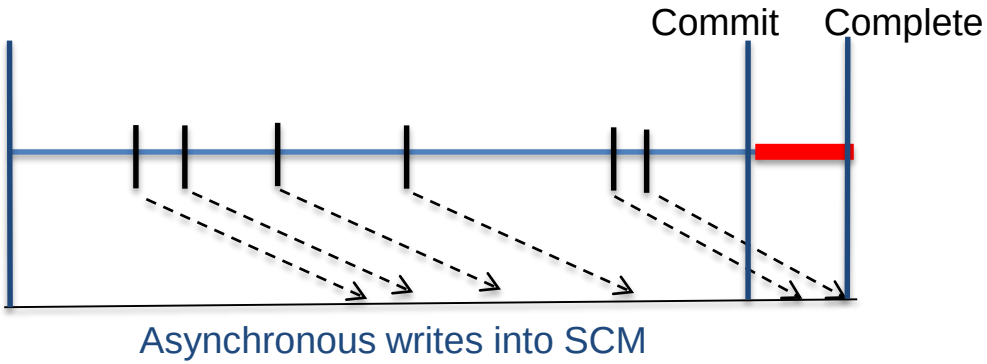
Atomic Writes to SCM



Writes Must be Atomic
Re-Do Log



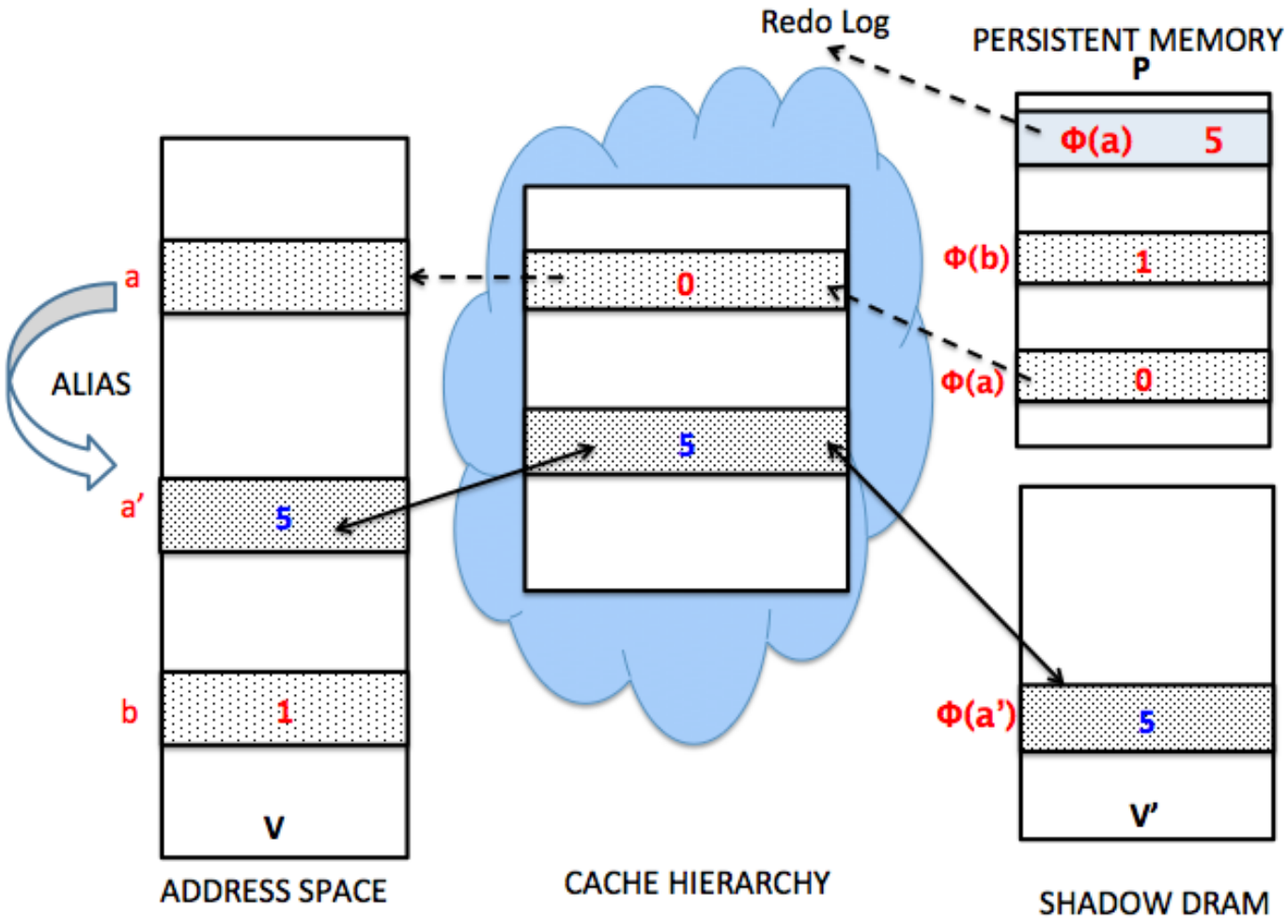
Copy-On-Write
Or
Undo Log



Ideal Method:

- Asynchronous Writes to SCM
- Speed
- Only small delay at end to ensure all are written.
- No front-end changes

SoftWrAP Approach



Software-based Write-Aside Persistence

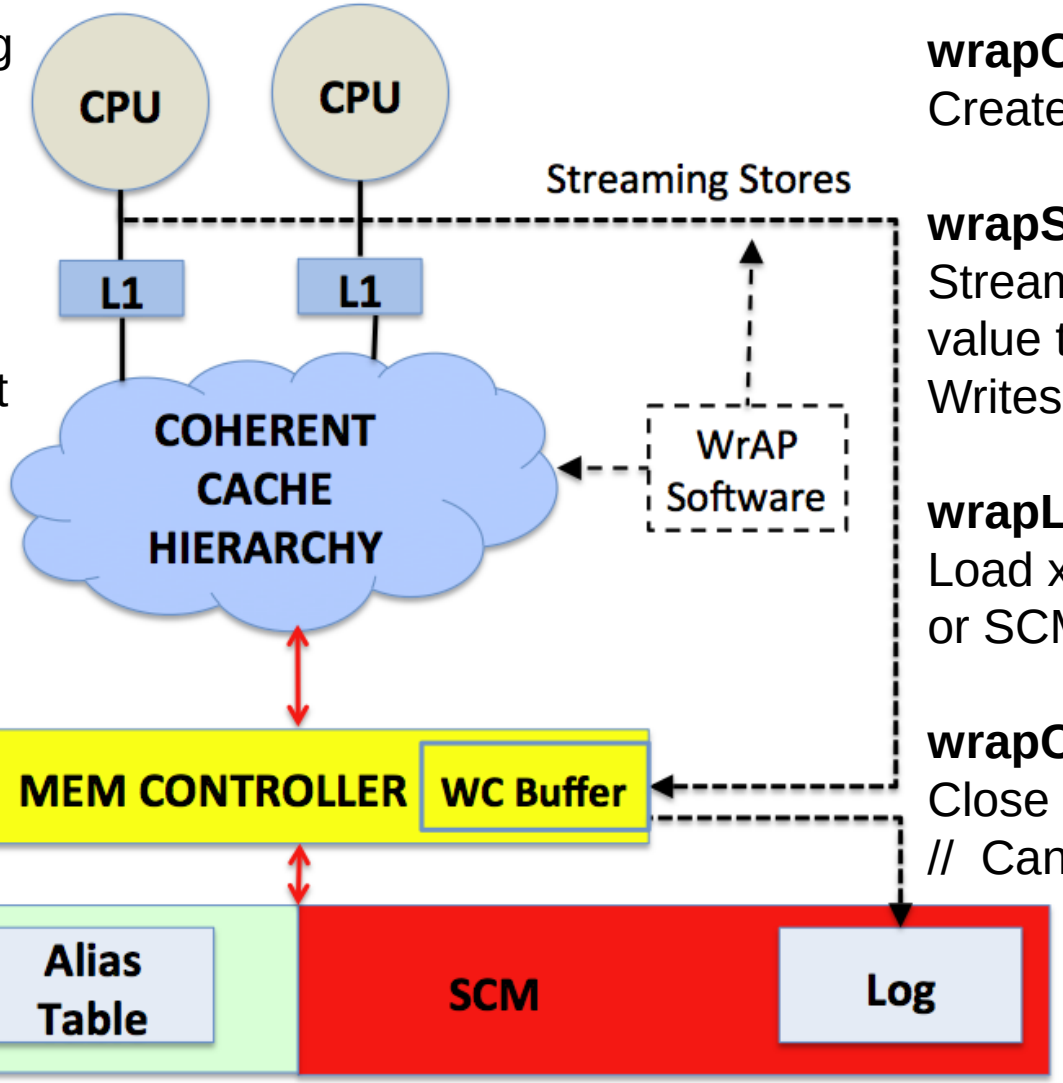
- Aliasing catches cache evictions.
- Fast path through cache hierarchy
- Re-Do Log for atomicity

Transaction

wrapOpen();
wrapStore(a, 5);
….
c = wrapLoad(b);
wrapClose();

SoftWrAP Approach

- Aliasing & Redo Log
- Fast foreground path through the cache hierarchy using alias location.
- Asynchronous conduit to persistent memory log using streaming stores.
  - mm_stream_si32
  - MOVNTI
  - Bypass cache
- Approach decouples concurrency control from persistence.



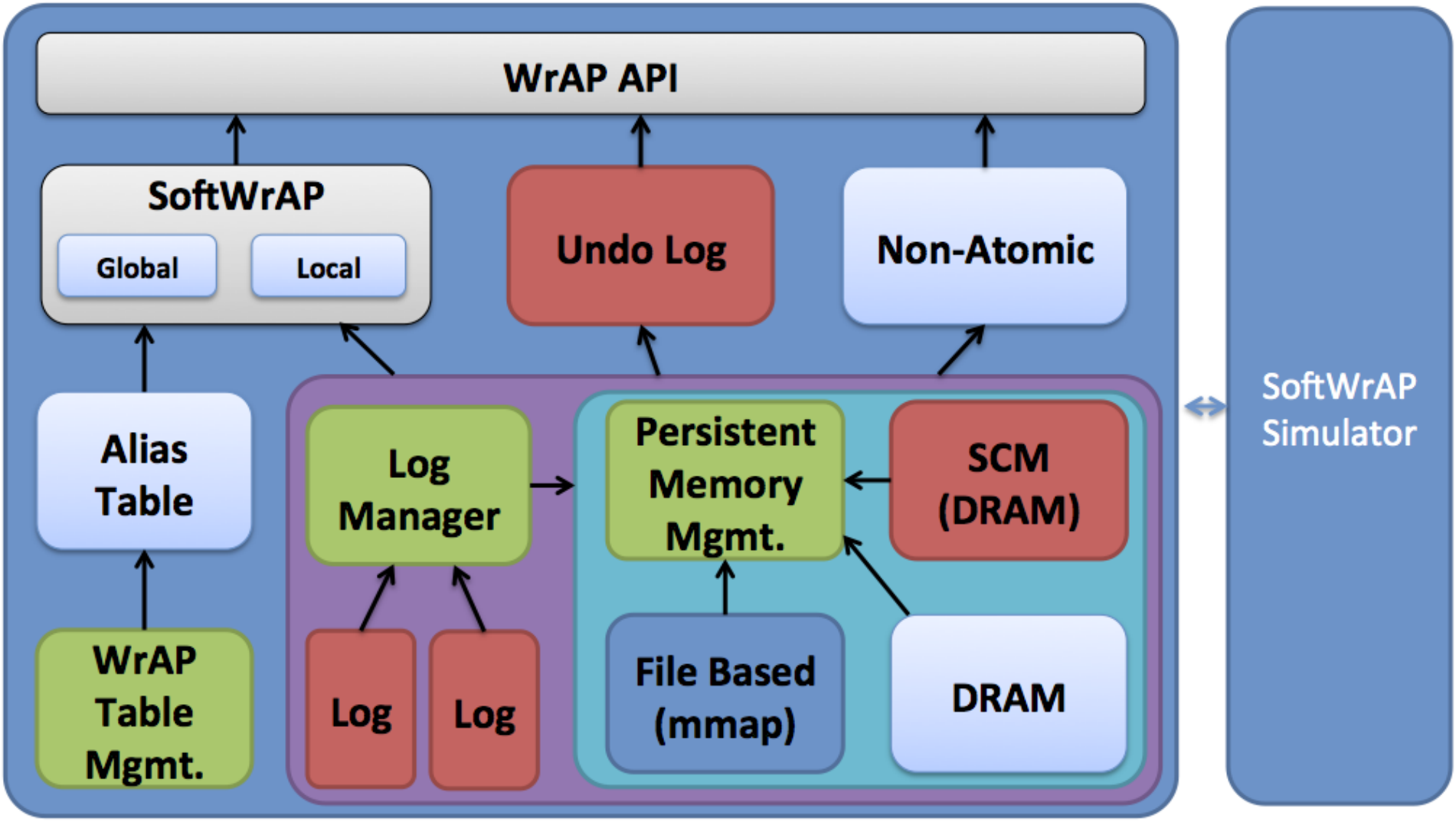
wrapOpen()
Creates Log

wrapStore(x, val)
Streams location x and value to log
Writes x to Alias Table

wrapLoad(x)
Load x from alias table or SCM if not present

wrapClose()
Close log & PCOMMIT
// Can process table.

SoftWrAP Architecture



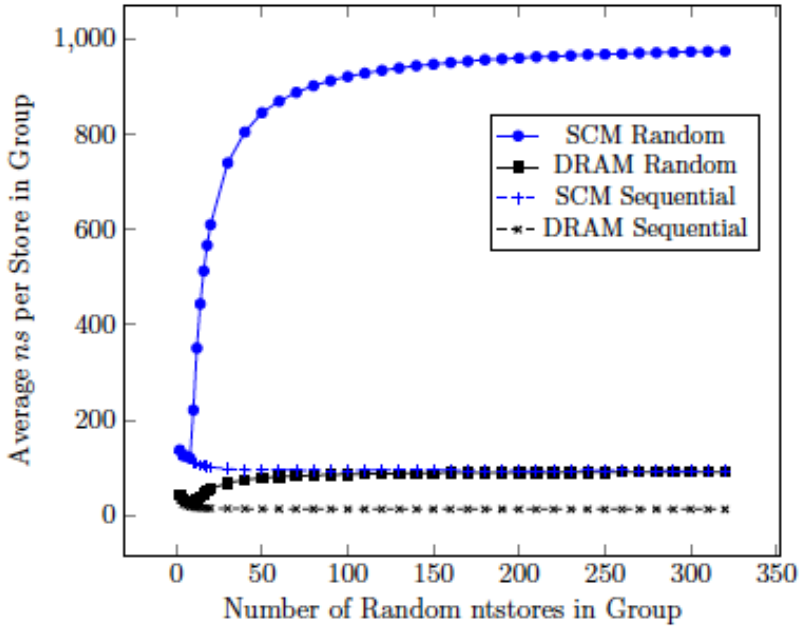
SCM Emulation

- Simulation:
  - Provided only In-Order execution.
  - Single instruction issue
  - Results depend on model of the cache and memory subsystem
  - Multi-threading scheduling
- SoftWrAP Could benefit from Out-Of-Order execution.
- Tested SoftWrAP on HW DRAM Interposer/Tracer at Intel.
- Analysis showed additional features: DRAM based provided better speedup, pre-fetching, etc..
- Validated writes proceeding to memory.

SCM Emulation:

- Streaming stores go into a software emulated write buffer and to DRAM.
- Running software on HW and DRAM

Tunable SCM Model:



Global Alias Table

Hash Table A
State: Retiring

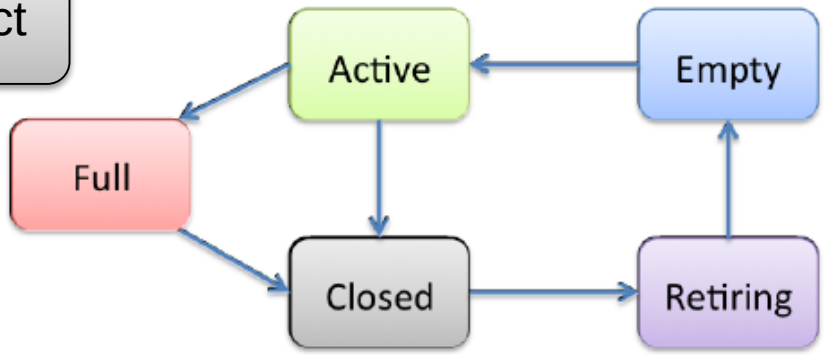
Address	Value	Size
…	…	…
…	…	…
M	1	8
Z	3	8
N		1024
…	…	…

Hash(W)

Hash Table B
State: Active

Address	Value	Size
W	5	4
X	-1	4
Y	7	4
Z	8	8
A	5	4
…	…	…

Data Object



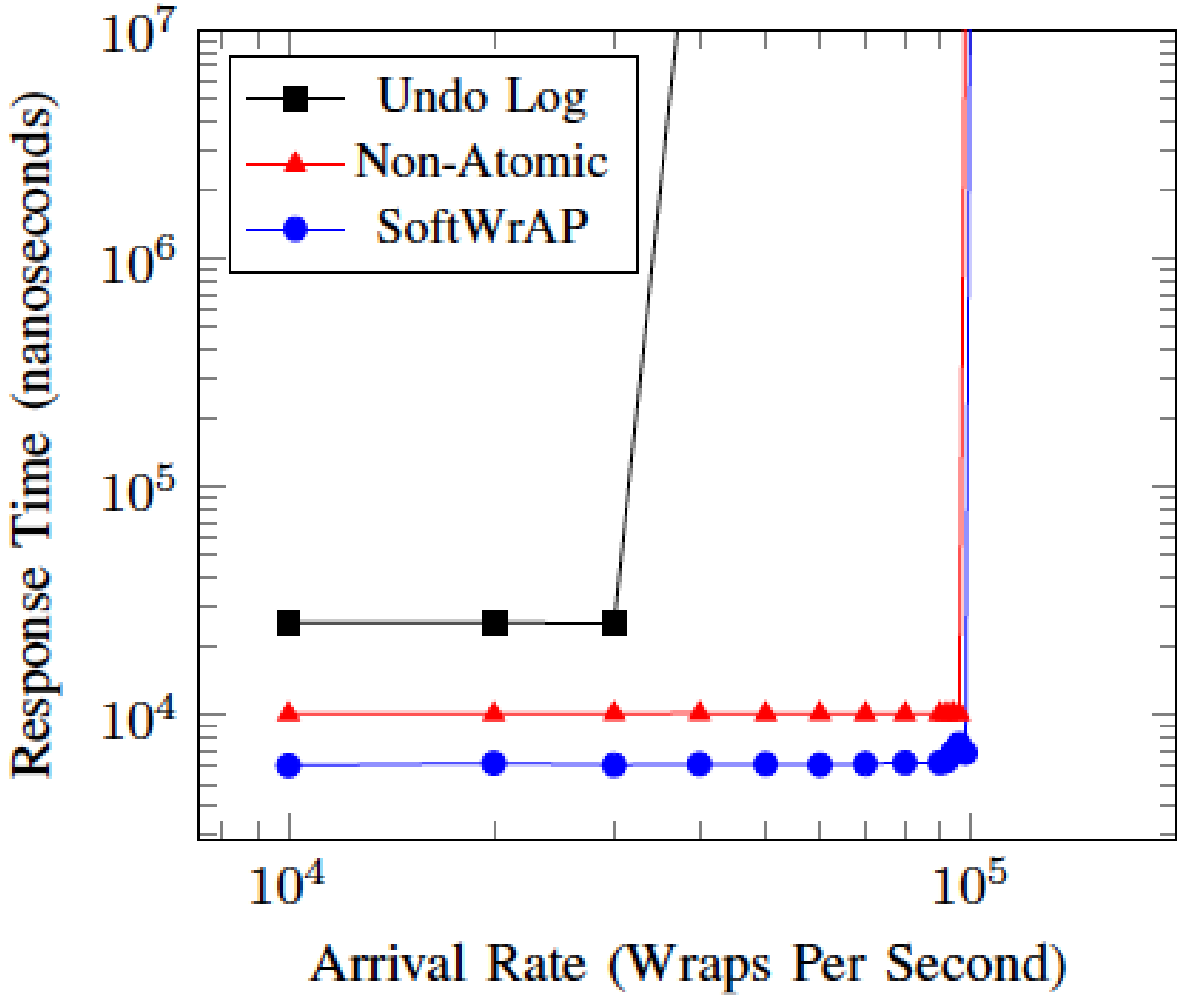
- Double Buffered (2 Hash Tables)
- Concurrent Retirement
- Supports primitives and object types
- Reads check both tables (if non-empty)
- 5 States for the Alias Table.
- Locks on state change for retirement and open/closeWrap.

Performance Model

- Tw is SCM write time and Talias is hash or alias time
- Ts is overhead to perform persistent memory sync (CPUID) and PCOMMIT
- One log entry for a 4-byte integer requires 4-bytes + 8-byte address = 12 bytes
- Write combining cache lines of 64 bytes of contiguous log entries and 1 write for log management.
- Commit a group of *n* writes to SCM:

	SCM Writes	*pcommits*	Estimated Time
Non-Atomic	n	1	$nT_w + T_s$
UndoLog	$2n + 1 + \lceil 12n/64 \rceil$	$n + 1$	$n(2T_w + T_s) + \lceil 12n/64 + 1 \rceil T_w + T_s$
SoftWrAP	$1 + \lceil 12n/64 \rceil$	1	$T_w + max(n * T_{alias}, \lceil 12n/64 \rceil T_w) + T_s$

Response Time



n=10 writes

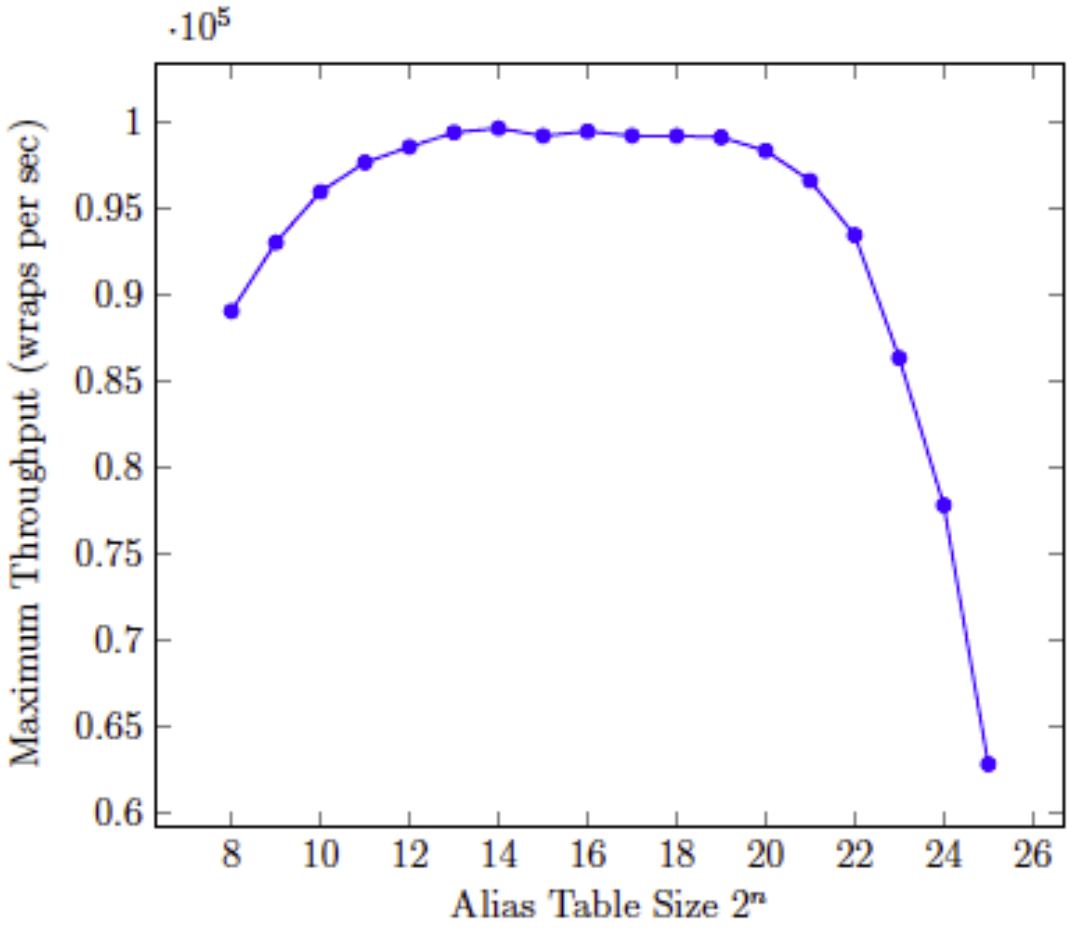
Tw = 1µs

Response Time

Foreground 12n/64 writes and aliasing is faster than n direct writes to SCM.

Large array, groups of 10 stores to random locations in the array, varying arrival rate of incoming transactions. Processing of alias table, size 16k, in background.

Alias Table Analysis



N=10 writes

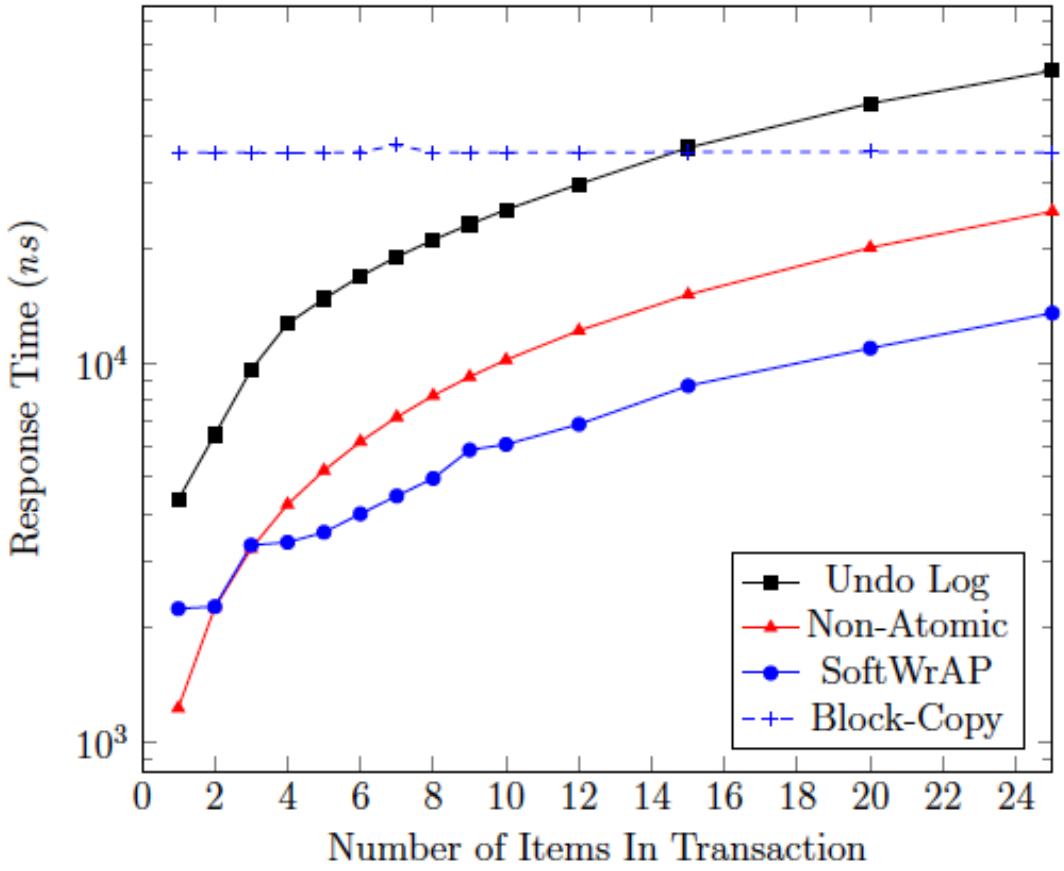
Tw = 1μs

Arrival rate = 50k wraps per second

Small tables that fit in L1 cache perform best. Too small causes too many table switches.

Maximum throughput (wraps per second) for various of alias table sizes in double buffered implementation.

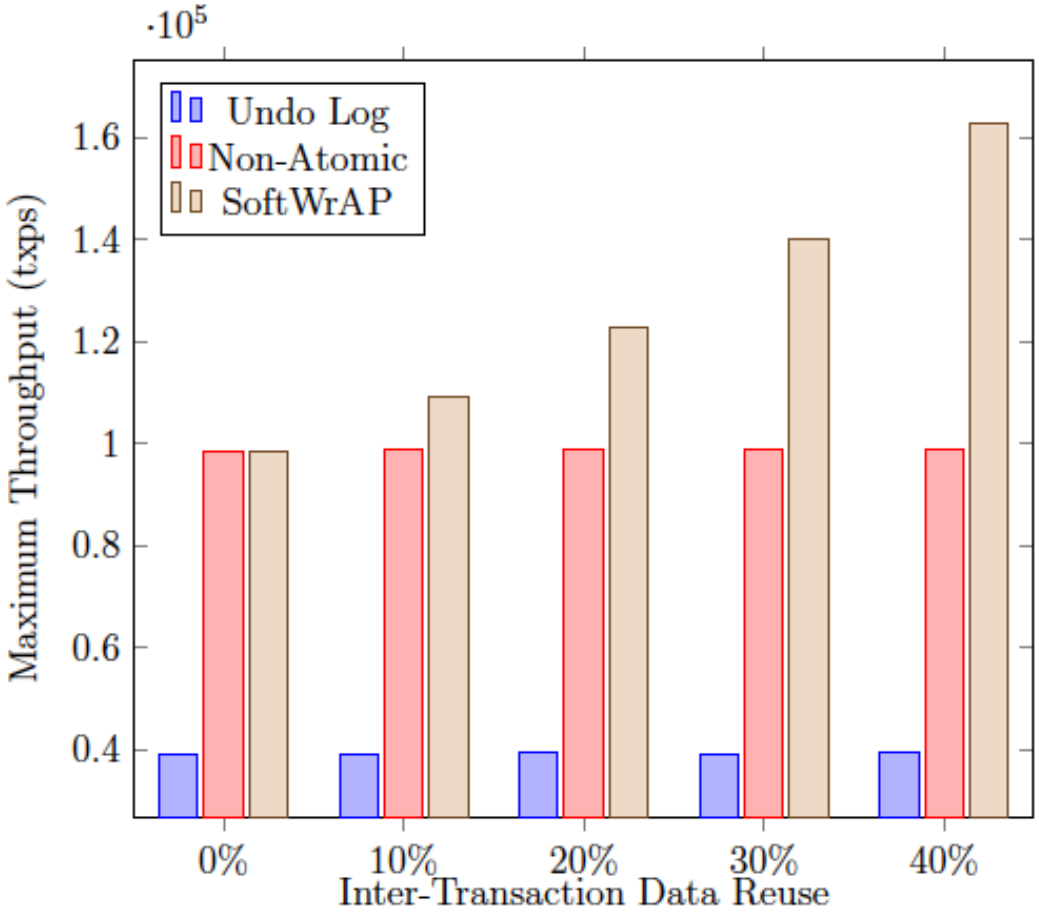
Transaction Size



Block Copy:
- All N items in 1k cont. block
- Data structure allows for a pointer flip to new block.

Response time for various transaction sizes with arrival rate of 1,000 wraps per second, alias table size of max 8k entries, and SCM Tw=1µs.

Data Reuse



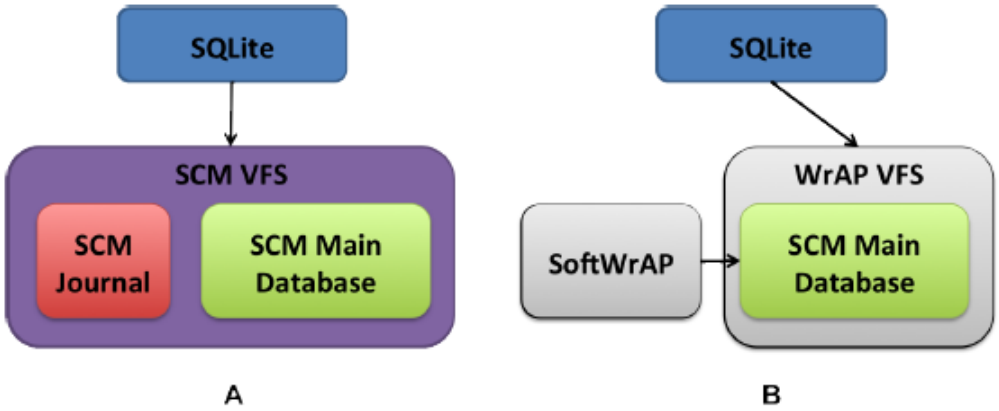
- Cache hits in alias table results in faster loads and stores.

- *Retirement of only one copy of reused variable required.

Maximum throughput for various percentage of data reuse across transactions with size n=10, alias table size of max 8k entries, and SCM Tw=1µs

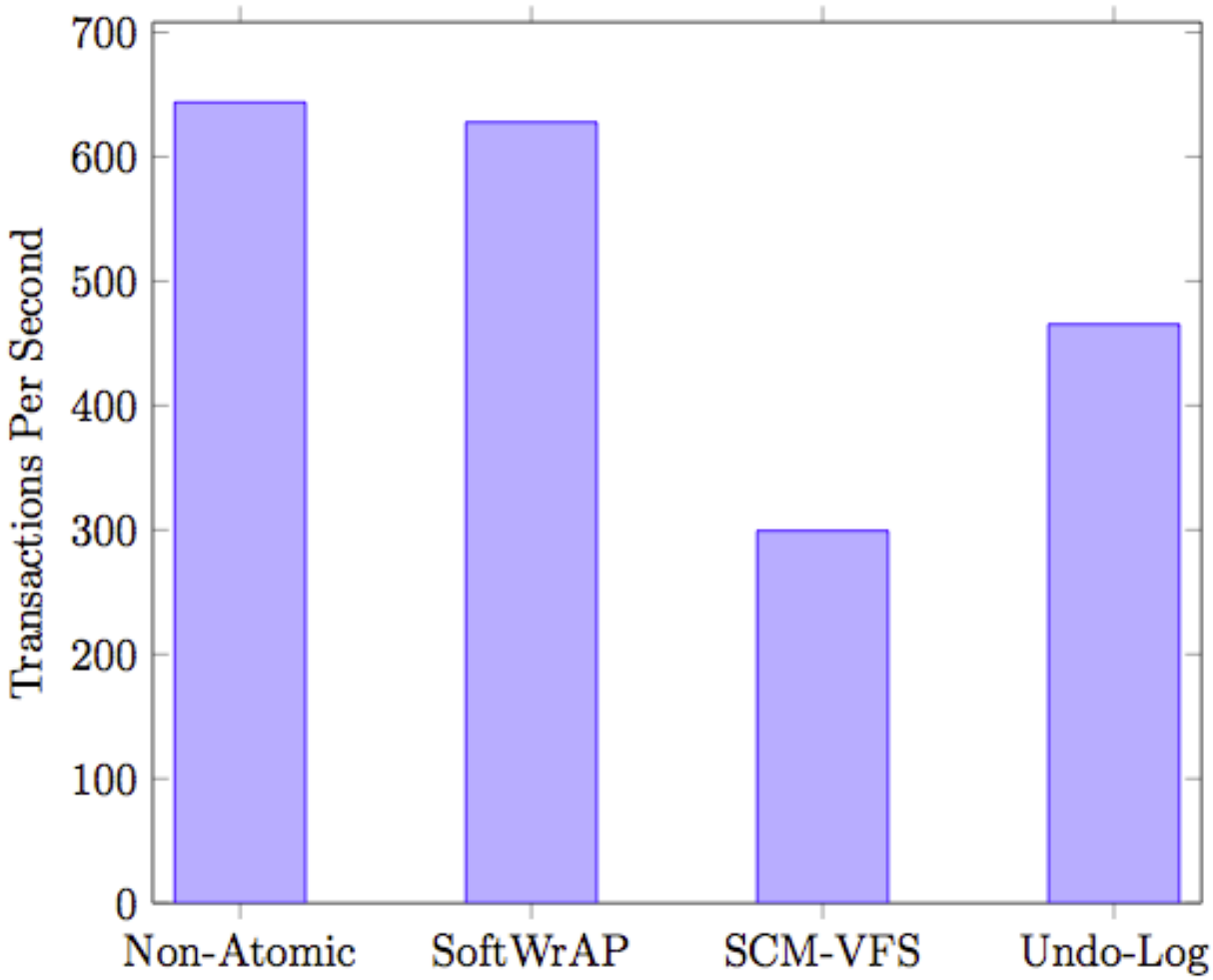
SQLite / TPC-C

- Created two VFS. One native to SCM and one using SoftWrAP (can also model Undo-Log, Non-Atomic).
- A: SCM VFS uses SCM journal (r/w bytes)
- B: SoftWrAP handles consistency



- TPC-C is an Online Transaction Processing Benchmark. Comprised of 9 tables and a number of transactions
- PY-TPCC is modified and executed to save SQL statements for TPC-C benchmark to file.
- SQLite is executed with VFS under test and generated TPC-C SQL statement file.

SQLite SCM TPC-C



Tw = 1us

Throughput in Transactions Per Second for the TPC-C Benchmark with SQLite. SoftWrAP has similar performance to Non-Atomic.

RICE

Conclusions and Forward Work

- Looking at additional aliasing mechanisms and enhancements such as compiler integrations.
- Evaluation on hardware when available.
- SoftWrAP is a fast, straightforward approach to ensuring transactional support for writing byte-addressed persistent data without any hardware changes.
- It provides a fast path through the cache hierarchy while utilizing a background path to persist groups of stores to SCM atomically.
- SoftWrAP decouples concurrency control from persistence.
- Being released as open source software.
- SoftWrAP has promising results that approach the performance of persistence methods that don’t guarantee consistency and outperform Undo-Log approaches.

RICE

Questions?

Thank You!

Supported by NSF Grant CCF 1439075 and Intel SSG.