Transactional Storage Class Memory

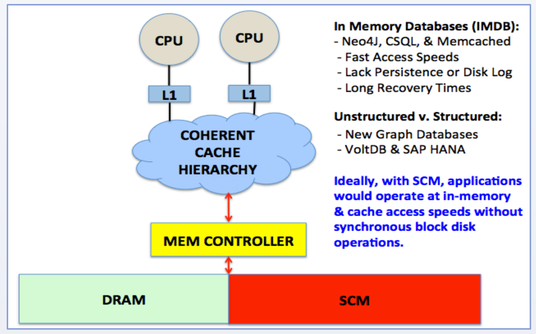
Ellis Giles

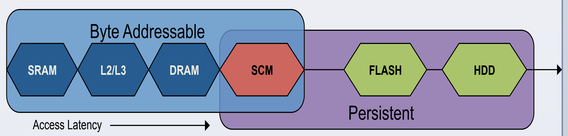
Rice University

Peter Varman (Advisor)

MOTIVATION

Emerging Storage Class Memory (SCM) promises to be a fast, byte-addressable, persistent memory near DRAM on the main memory bus.



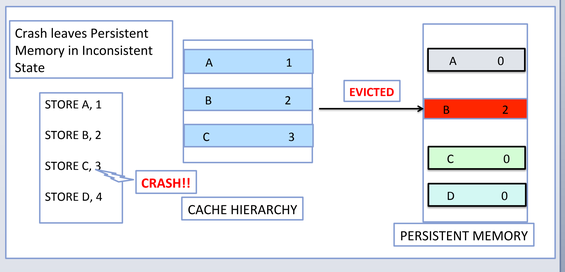


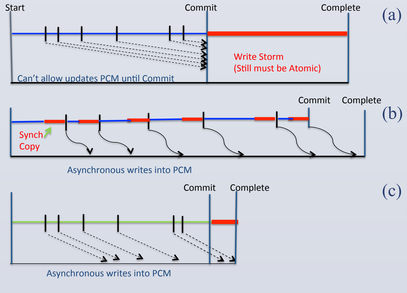
Technology	Density um²/bit	Read / Write Latency (ns)		Read / Write Energy pJ/bit		Write Endurance
HDD	0.00006	3,000,000	3,000,000	2,500	2,500	Near Inf
Flash SSD	0.00210	25,000	200,000	250	250	10^5
DRAM	0.00380	55	55	24	24	10^{18}
PCM	0.00580	48	150	2	20	10^8
Memristor	0.00580	100	100	2	2	10^8

*S. Venkataraman , N. Tolia , P. Ranganathan , R. Campbell, "Consistent and durable data structures for non-volatile byte-addressable memory," FAST 2011, San Jose, California.

PROBLEM

Hardware and software crashes can leave the system in an inconsistent state.





(a) All writes must be atomic

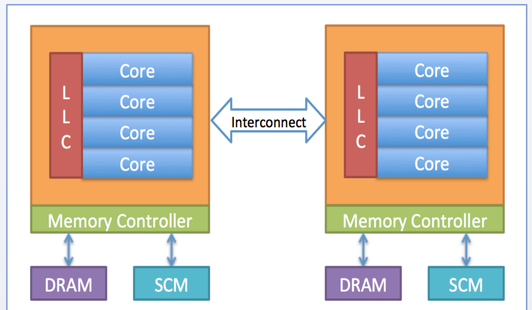
(b) Copy-on-Write

(c) Ideal – asynchronous updates to SCM

Random cache evictions and system crashes cause the programmer to create a slow logging solution to maintain SCM consistency.

PARALLEL PERSISTENCE CHALLENGES

With both Hardware and Software Transactional Memory, committing parallel transactions doesn't guarantee persistence to SCM.



Listing 1: Using Locks

```
Txfer(&x, &y, amt)
{
  Lock(x);
  Lock(y);
  x -= amt;
  y += amt;
  Unlock(y);
  Unlock(x);
}
```

Listing 2: Using Atomic

```
Txfer(&x, &y, amt)
{
  atomic
  {
    x -= amt;
    y += amt;
  }
  // end
}
```

Listing 3: Using HTM

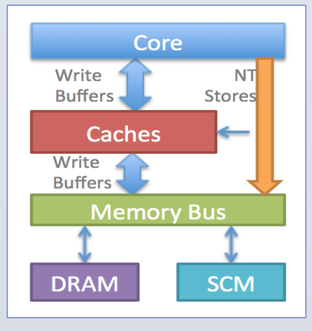
```
Txfer(&x, &y, amt)
{
  HTMBegin();
  x -= amt;
  y += amt;
  HTMEnd();
}
```

Listing 4: Using STM

```
Txfer(&x, &y, amt)
{
  STMBegin();
  t=STMRead(&x);
  STMWrite(&x, tmp-amt);
  t=STMRead(&y);
  STMWrite(&y, tmp+amt);
  STMEnd();
}
```

Locks, Non-locking methods, Hardware and Software Transactional Memory guarantee isolation but not persistence.

Values may be caught anywhere in the memory hierarchy.

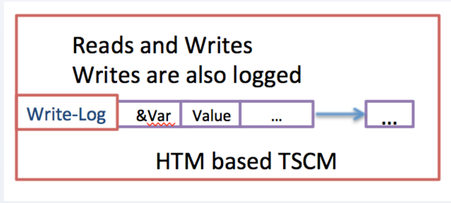


Non-temporal or streaming stores and cache line flushes and write-backs place values in the write buffers but may abort transactions.

TRANSACTIONAL STORAGE CLASS MEMORY

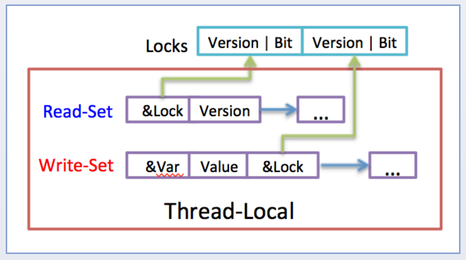
TSCM modifies both HTM and STM to combine parallelism with persistence for High Performance Applications.

TSCM With Hardware TM Support



On Transaction commit, Log is committed then may commit values to home locations in the background.

TSCM With Software TM Support



Values may be persisted to SCM using persistent memory fences (sfence, pcommit, sfence) in the write set using an Eager or Lazy method.

RELATED WORK

Intel is adding support from ISA for persistent memory:

- Failure atomicity for writes of 64-bits.
- More efficient cache flushing w/new CLWB instruction, which places a cache line into the write buffer without cache eviction.
- An additional new instruction, PCOMMIT, which flushes a volatile write buffer to memory locations.

Software Transactional Memory (STM) with Transactional Locking
Hardware Transactional Memory (HTM)

Battery Backup Based Work

Up-Front Cache Line Changes
- BPFS (epoch barriers on cache eviction; uses copy-on-write)
- Consistent and Durable Data Structures (requires a *flush* primitive)
- Kiln (non-volatile victim cache provides transaction buffering)

Software Solutions
- REWIND (In-place updates; atomic doubly linked list)
- NV-Heaps (Logging in persistent heaps)

Software Control Layer and Compiler Additions
- Mnemosyne (STM based interception of all reads and writes)
- ATLAS (Automatically generates atomic regions; uses Undo Log)

ACKNOWLEDGEMENTS

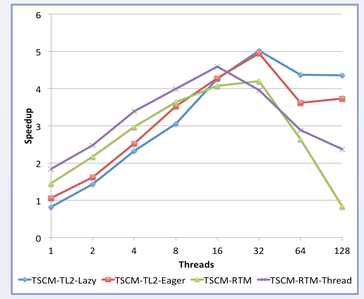
The student author would like to thank Dr. John Mellor-Crummey for inspiring this idea during his course on Multicore Computing Rice University.

RESULTS

The TSCM techniques can achieve high performance by combining fast persistence techniques with HTM and STM.

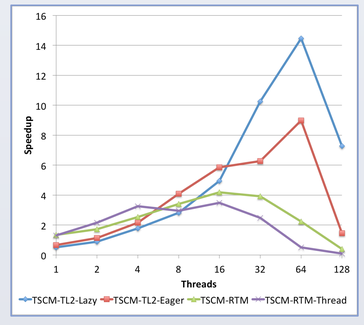
Stanford Transaction Applications for Multi-Processing (STAMP) Benchmarks
SSCA – Scalable Synthetic Compact Applications tests, short, small.

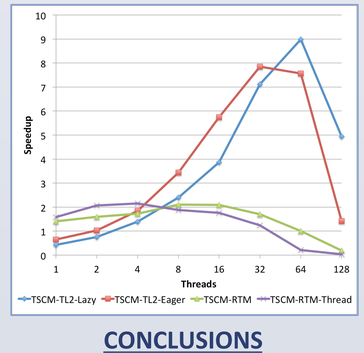
TSCM For Persistence with Parallelism Using TL2 and RTM:



Vacation Benchmarks from STAMP Simulate OLTP System with larger write sets and much longer transaction times.

TSCM with TL2 and RTM for Vacation:





CONCLUSIONS

- TSCM is a high performance library that combines TM techniques with persistence.
- It achieves high performance, close to TM applications without persistence.
- TL2-Lazy approaches in TSCM perform better than eager due to write-combining and reduced numbers of commit operations.
- TSCM with HTM or RTM performs best with lower contention.

Supported in part by Intel Corp SSG grant and NSF grant CCF-1439075.