Hardware Acceleration of Transactional Memory on Commodity Systems

Jared Casper, Tayo Oguntebi,
Sungpack Hong, Nathan Bronson,
Christos Kozyrakis, Kunle Olukotun

Pervasive Parallelism Laboratory
Stanford University

TM Design Alternatives

- Software (STM)
  - "Barriers" on each shared load and store update data structures
- Hardware (HTM)
  - Tap hardware data paths to learn of loads and stores for conflict detection
  - Buffer speculative state or maintain undo log in hardware, usually at the L1 level
- Hybrid
  - Best effort HTM falls back to STM
  - Generally target small transactions
- Hardware accelerated
  - Software runtime is always used, but accelerated
  - Existing proposals still tap the hardware data path

TMACC: TM Acceleration on Commodity Cores

- Challenges facing adoption of TM
  - Software TM requires 4-8 cores just to break even
  - Hardware TM is expensive and risky
    - Sun’s Rock provides limited HTM for small transactions
    - Support for large transactions requires changes to core
    - Optimal semantics for HTM is still under debate
  - Hybrid schemes look attractive, but still modify the core
  - No systems available to attract software developers

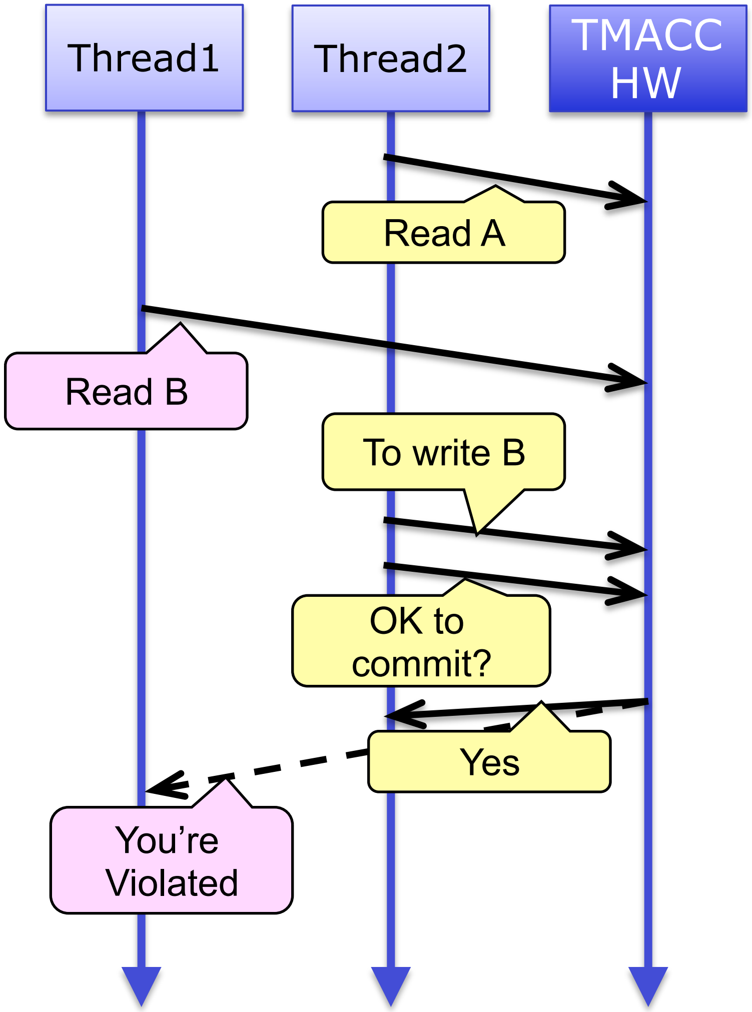
- Accelerate STM without changing the processor
  - Leverage much of the work on STMs
  - Much less risky and expensive
  - Use existing memory system for communication

TMACC: TM Acceleration on Commodity Cores

- Conflict detection ✓
  - Can happen after the fact
  - Can nearly eliminate expensive read barriers
- Checkpointing ✗
  - Needs access to core internals
- Version management ✗
  - Latency critical operations
  - Common case when load is not in store buffer must take less than ~10 cycles
- Commit ✗
  - Could be done off-chip, but would require removing everything from the processor’s cache

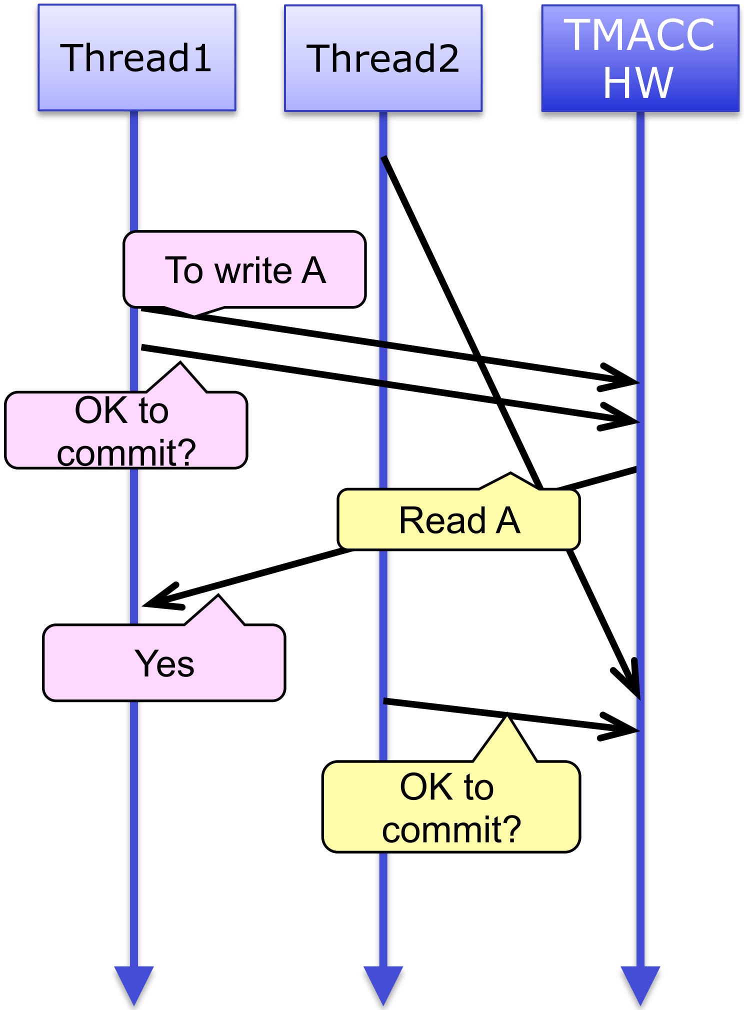
Protocol Overview

- Reads
  - Send address to HW
  - Check for value in write buffer
- Writes
  - Add to the write buffer
  - Same as STM
- Commit
  - Send HW each address in write set
  - Ask permission to commit
  - Apply write buffer
- Violation notification
  - Must be fast to check for violation in software

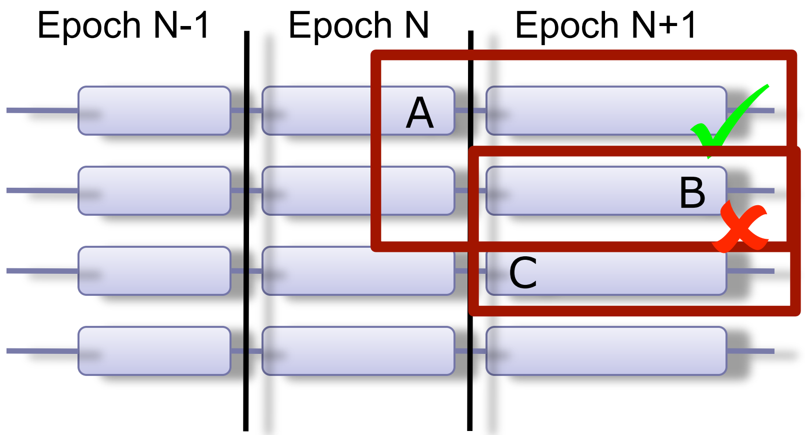


Problem of Being Off-Core

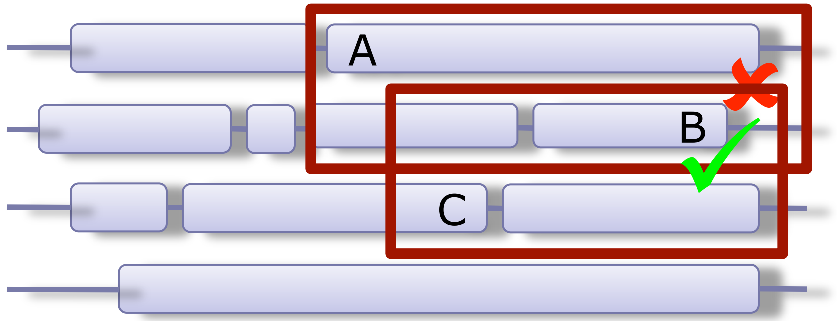
- Variable latency to reach the HW
  - Network latency
  - Amount of time in the store buffer
- How can we determine correct ordering?



Global and Local Epochs



Global Epochs



Local Epochs

- Global Epochs
  - Each command embeds *epoch number (a global variable)*.
  - Finer grain but requires global state
  - Know A < B,C but nothing about B and C
- Local Epochs
  - Each thread declares start of new epoch
  - Cheaper, but coarser grain (non-overlapping epochs)
  - Know C < B, but nothing about A and B or A and C

Two TMACC Schemes

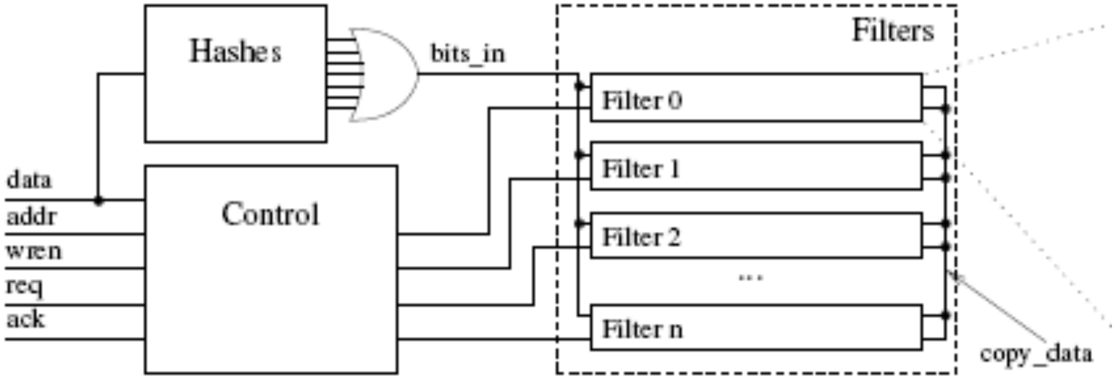
- We proposed two TM schemes.
  - TMACC-GE uses global epochs
  - TMACC-LE uses local epochs
- Trade-Offs

TMACC-GE	TMACC-LE
More accurate conflict detection → less false positives ✓	No global data in software → less SW overhead ✓
Global epoch management → more SW overhead ✗	Less information for ordering → more false positives ✗

- Details in the paper

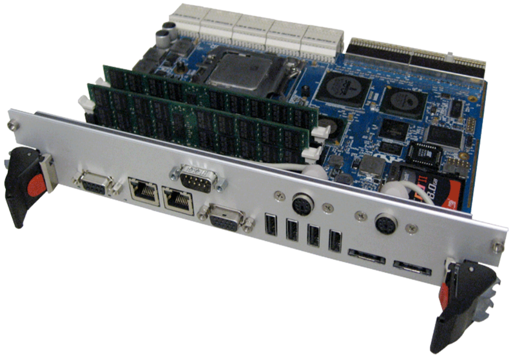
TMACC Hardware

- A set of generic BloomFilters + control logic
  - BloomFilter: a condensed way to store ‘set’ information
  - Read-set: Addresses that a thread has read
  - Write-set: Addresses that other threads have written
- Conflict detection
  - Compare read-address against write-set
  - Compare write-address against read-set

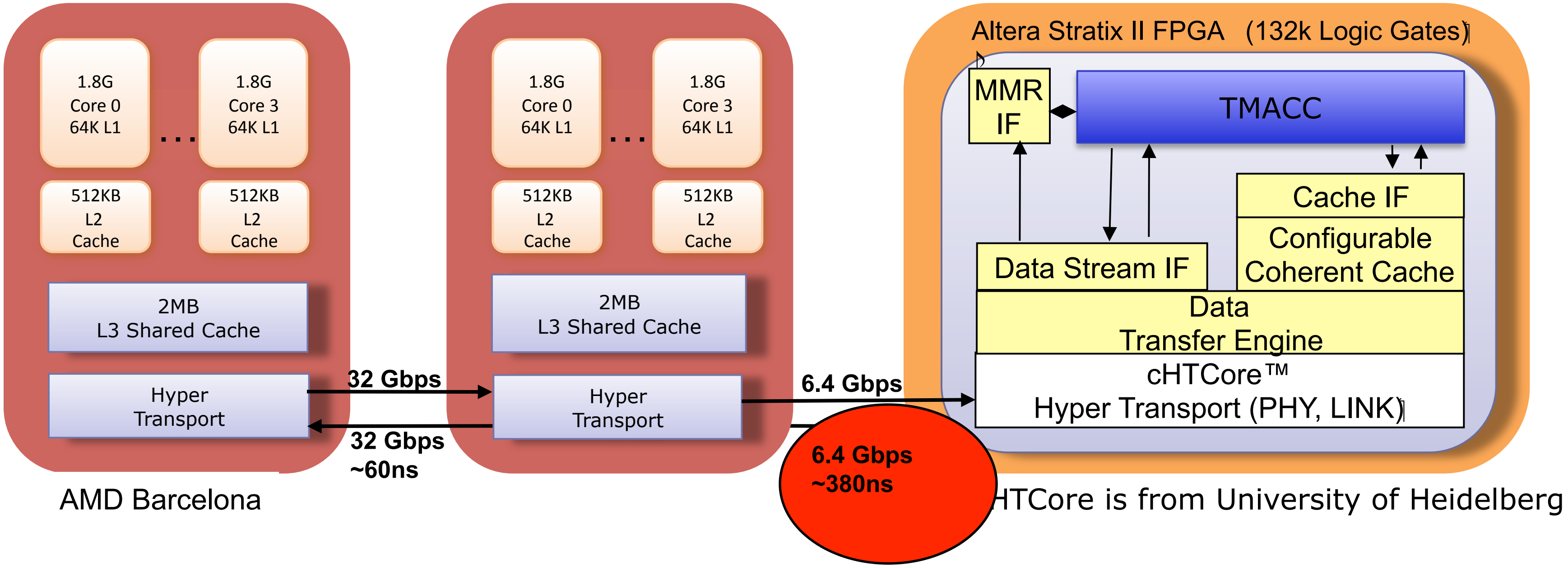


Procyon System

- First implementation of FARM single node configuration
- From A&D Technology, Inc.
- CPU Unit (x2)
  - AMD Opteron Socket F (Barcelona)
  - DDR2 DIMMs x 2
- FPGA Unit (x1)
  - Altera Stratix II, SRAM, DDR
- Each unit is a board
- All units connected via cHT backplane
  - Coherent HyperTransport (ver 2)
  - We implemented cHT compatibility for FPGA unit (next slide)



Base FARM Components

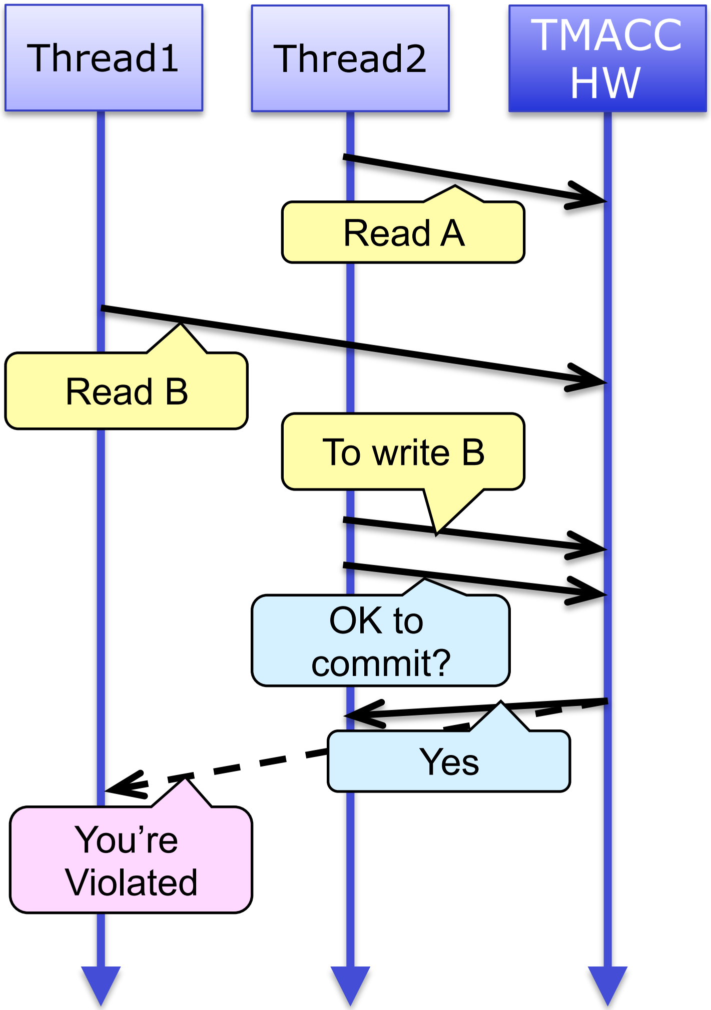


- Block diagram of Procyon system
- FPGA Unit = communication logics + user application
- Three interfaces for user application
  - Coherent cache interface
  - Data stream interface
  - Memory mapped register interface

FARM: A Prototyping Environment for Tightly-Coupled, Heterogeneous Architectures. Tayo Oguntebi et. al. FCCM 2010.

Communication

- Sending addresses
  - FARM's streaming interface
  - Address range marked as "write-combing" causes non-temporal store
  - As close to "fire-and-forget" as is available
  - 630MB/s
- Commit request
  - Read from memory mapped register
  - Approx. 700ns, 1000s of cycles!
- Violation notification
  - FPGA writes to cacheable address
  - Common case of no violation is fast, just as cache hit for the processor



Implementation Result

- Full prototype of both TMACC schemes on FARM
- HW Resource Usage

	Common	TMACC-GE	TMACC-LE
4Kb BRAM	144 (24%)	256 (42%)	296 (49%)
Registers	16K (15%)	24K (22%)	24K (22%)
LUTs	20K	30K	35K
FPGA	Altera Stratix II EPS130 (-3)		
Max Freq.	100 MHz		

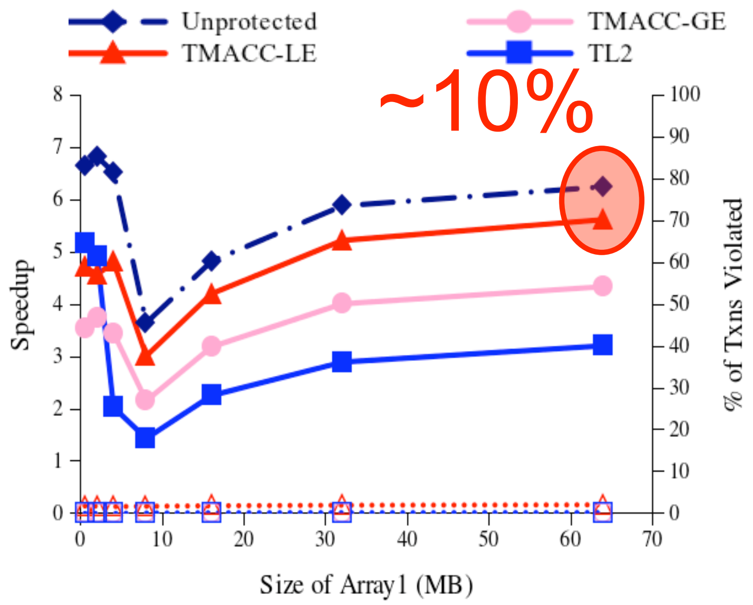
Microbenchmark Analysis

- Two random array accesses
  - Partitioned (non-conflicting)
  - Fully-shared (possible conflicts)
- Free from pathologies and 2nd-order effects
- Decouple effects of parameters
  - Size of Working Set (A1)
  - Number of Read/Writes (R,W)
  - Degree of Conflicts (C, A2)

```
Parameters: A1, A2, R, W, C

TM_BEGIN
 for I = 1 to (R + W) {
     p = (R / R + W)

      /* Non-conflicting Access */
      a1 = rand(0, A1 / N) + tid * A1/N;
      if (rand_f(0,1) < p))
             TM_READ( Array1[a1] )
       else
             TM_WRITE( Array1[a1] )

       /* Conflicting Access */
       if (C) {
            a2 = rand(0, A2);
            if (rand_f(0,1) < p))
                   TM_READ( Array2 [a2] )
            else
                   TM_WRITE( Array2[a2] )
       }
 }
TM_END
```

EigenBench: A Simple Exploration Tool for Orthogonal TM Characteristics. Sungpack Hong et. al. IISWC 2010

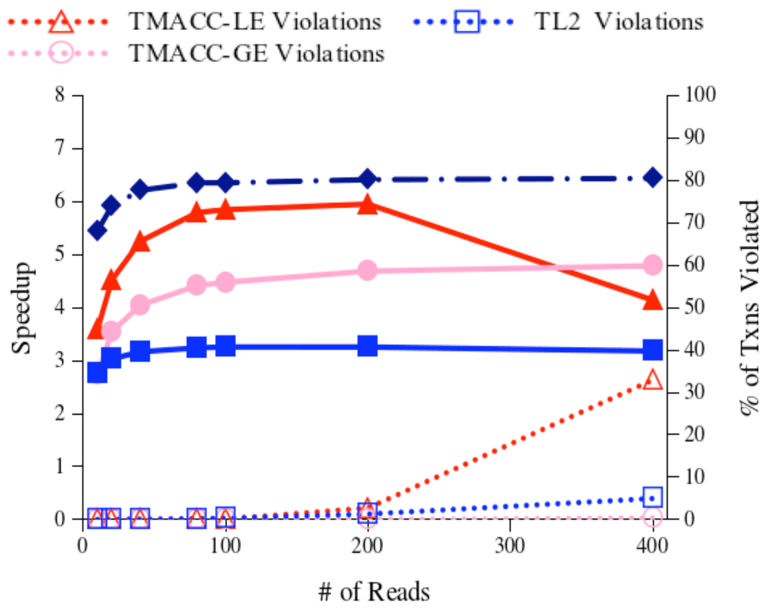
Microbenchmark Results



~10%

Working set size

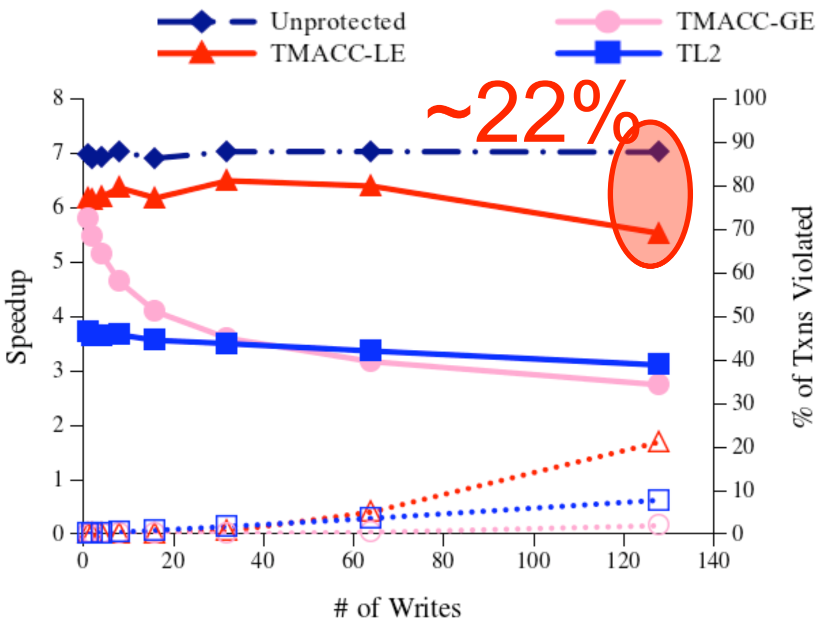
- The knee is overflowing the cache
- Constant spread out of speedup



Transaction size

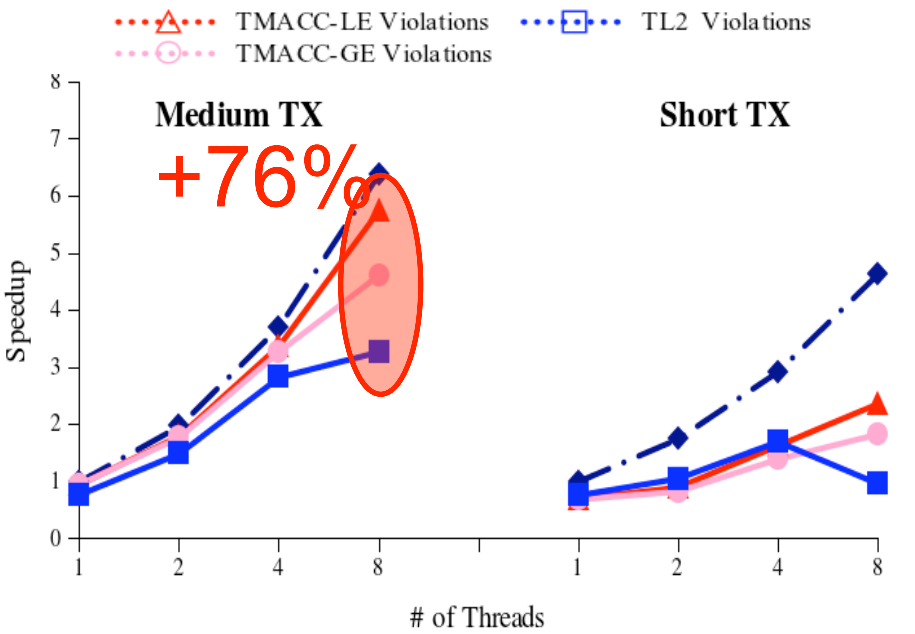
- All violations are false positives
- Sharp decrease in performance for small transactions
- TMACC-LE begins to suffer from false positives

Microbenchmark Results



Write set size

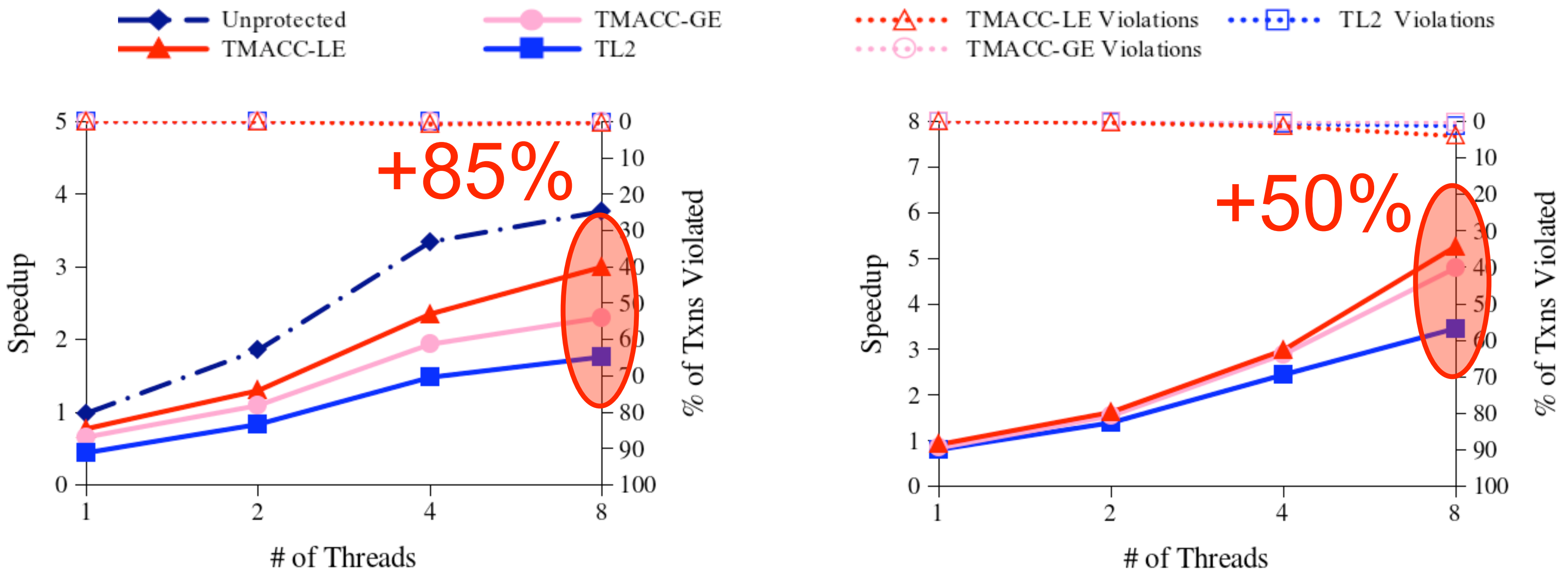
- TMACC-GE suffers from lock migration as the number of writes goes up



Number of threads

- Medium sized transactions scale well
- Small transactions are not accelerated
- TL2 suffers across chip boundary

STAMP Benchmark Results

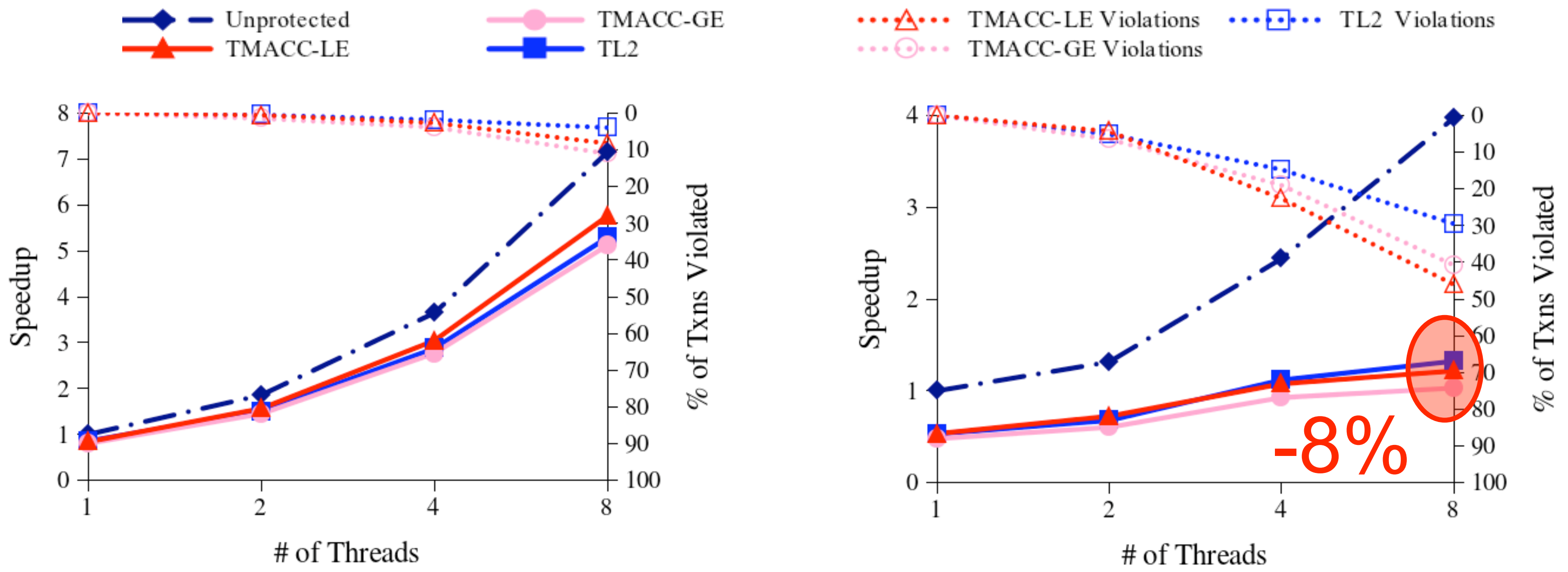


Vacation

Genome

- Transactions with few conflicts, a lot of reads, and few writes
- Bread and butter of transactional memory apps
- Barrier overhead primary cause of slowdown in TL2

STAMP Benchmark Results



K-means low

- Few reads per transaction
  - Not much room for acceleration
- Large number of writes
  - Hurts TMACC-GE

K-means high

- Violations dominating factor
- Still not many reads to accelerate

Prototype vs. Simulation

- Simulated processor greatly exaggerated penalty from extra instructions
  - Modern processors much more tolerant of extra instructions in the read barriers
- Simulated interconnect did not model variable latency and command reordering
  - No need for epochs, etc.
- Real hardware doesn't have "fire-and-forget" stores
  - We didn't model the write-combining buffer
- Smaller data sets looked very different
  - Bandwidth consumption, TLB pressure, etc.

Summary: TMACC

- A hardware accelerated TM scheme
  - Offloads conflict detection to external HW
  - Accelerates TM without core modifications
  - Requires careful thinking about handling latency and ordering of commands
- Prototyped on FARM
  - Prototyping gave far more insight than simulation.
- Very effective for medium-to-large sized transactions
  - Small transaction performance gets better with ASIC or on-chip implementation.
  - Possible future combination with best-effort HTM

Questions