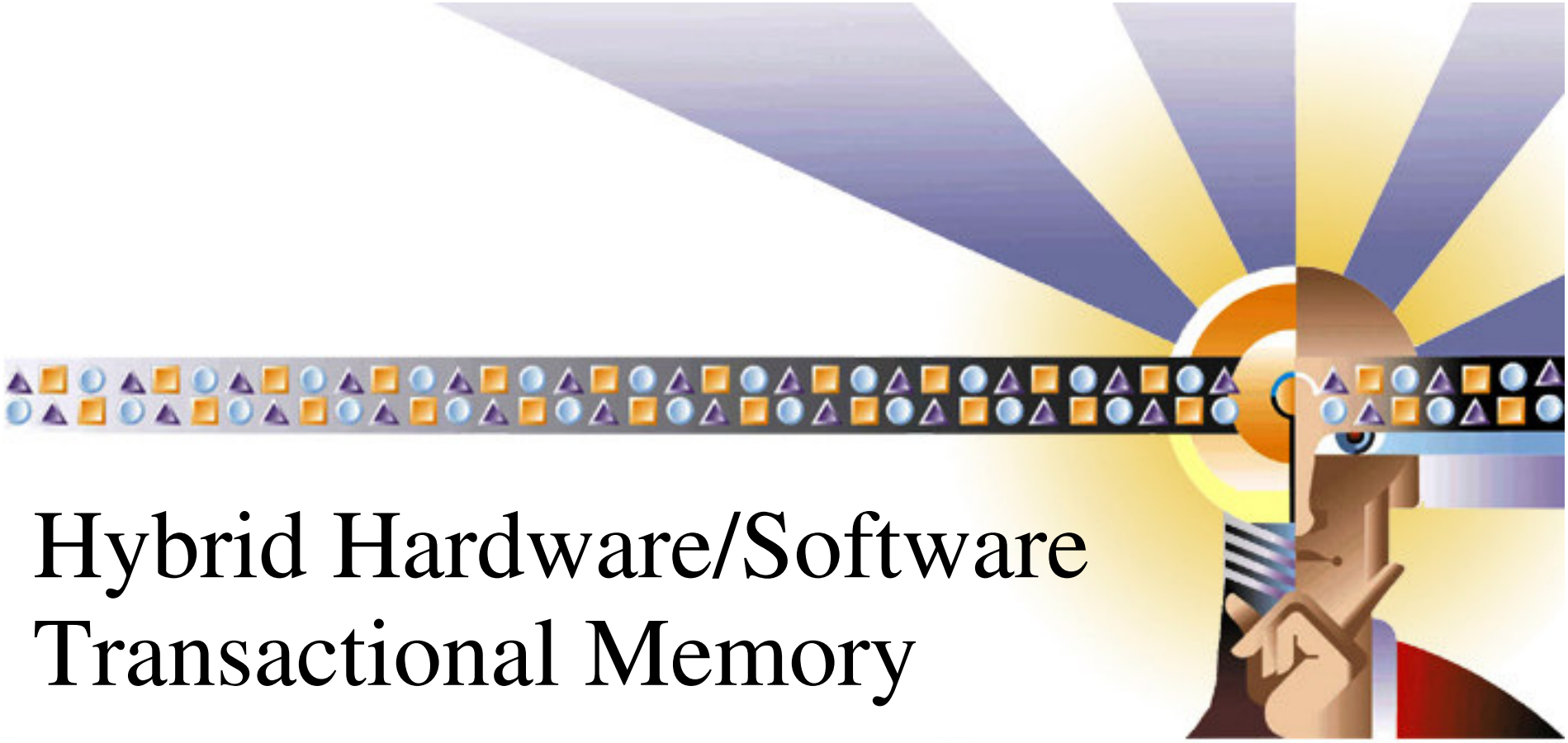


Hybrid Hardware/Software Transactional Memory

Mark Moir

Principal Investigator

Scalable Synchronization Research Group

Joint work with: Peter Damron, Yossi Lev, Victor Luchangco, Dan Nussbaum, Nir Shavit

Sun microsystems

We make the net work.

Outline

- Progress in software TM
- Developments in hardware TM
- Hybrid Hardware/Software TM
  - Key idea in abstract
  - Applied to DSTM
  - Applied to word-based STM
- Concluding remarks

Software Transactional Memory

- Lots of progress in:
  - Space overhead
  - Data transparency
  - Parallelism
  - Performance
- Work beginning on programming language/model/interface
- Still requires multiple expensive atomic instructions per transaction

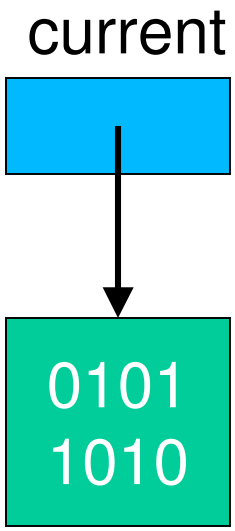
Hardware Transactional Memory…

- (…including related things like SLE).
- Renewed interest.
- Two flavours of design:
  - Best effort: simple, but no guarantees
  - Unbounded: can commit any transaction, but more complicated

Hybrid Transactional Memory

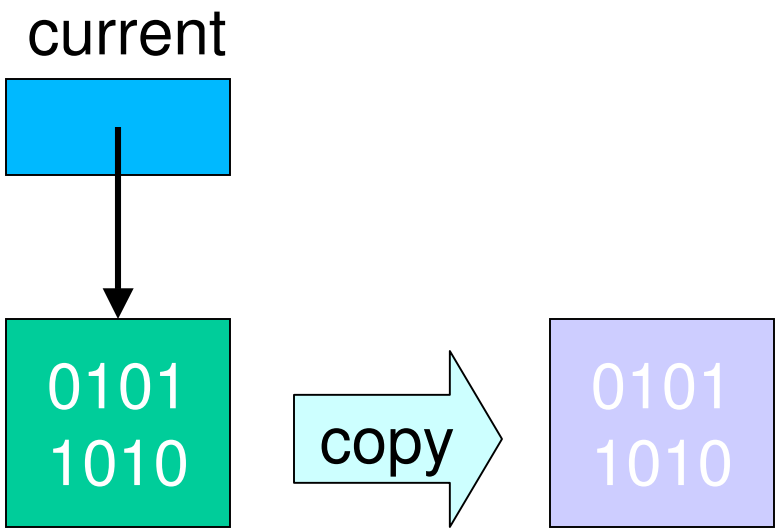
- Make HTM and STM txns "play together nicely"; try hardware first, resort to software when it fails
- Programmer writes txn code once; hybrid TM does the rest
- Hardware can be "best effort", without impacting programmers
- Transactional code can be developed and tested without hardware support

Key idea illustrated



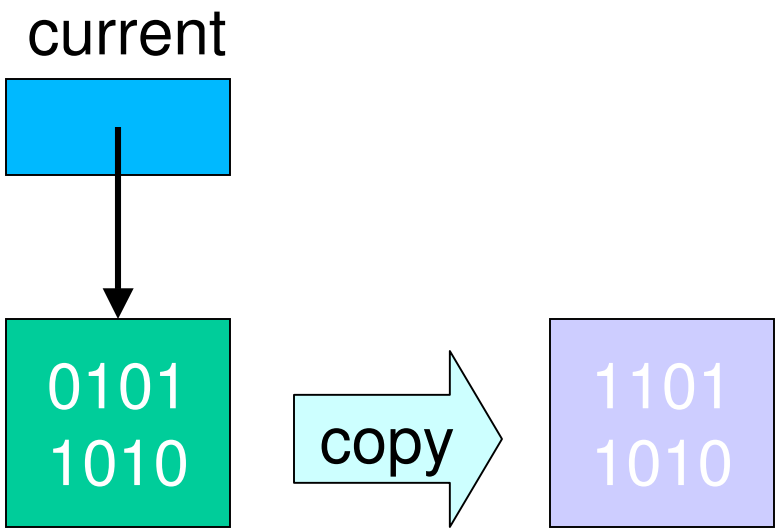
- Well-known software synchronization technique
- Copies of data and indication of which is "current".

Key idea illustrated



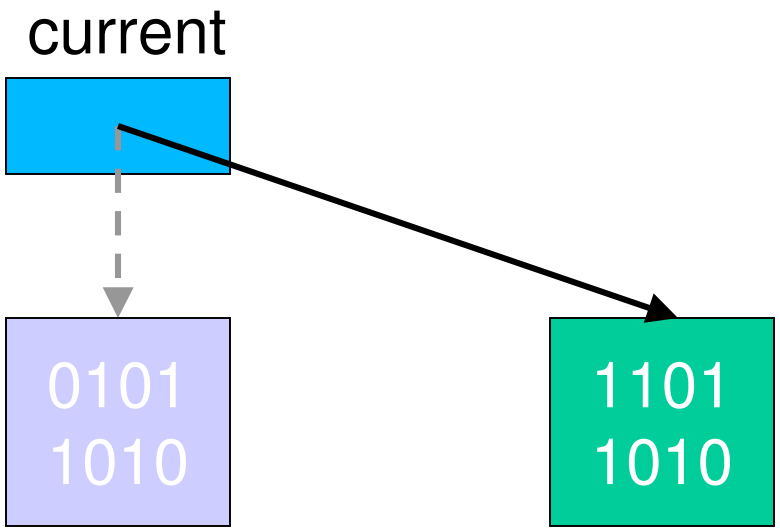
- Applying an operation:
  - make private copy

Key idea illustrated



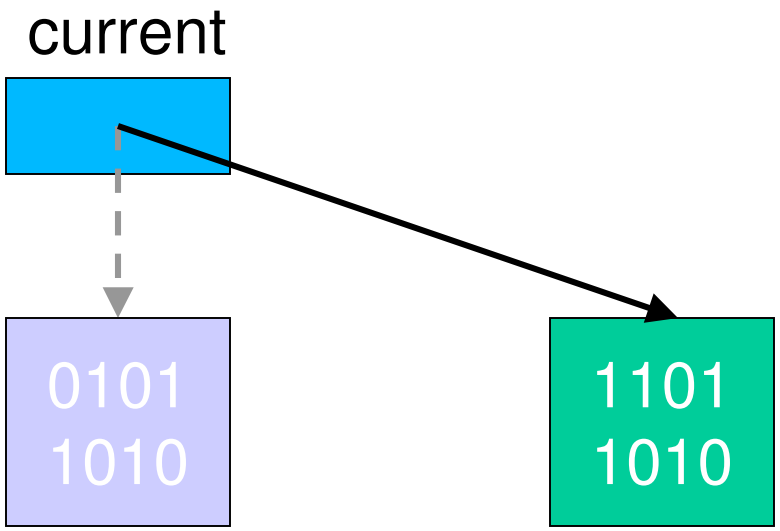
- Applying an operation:
  - make private copy
  - modify private copy (sequential code)

Key idea illustrated



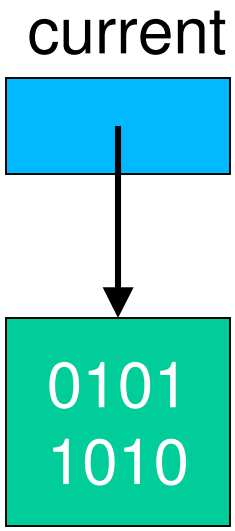
- Applying an operation:
  - make private copy
  - modify private copy (sequential code)
  - atomically make copy become current

Key idea illustrated



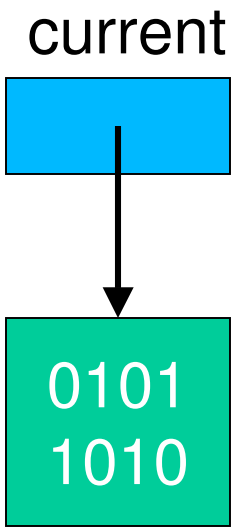
- Disadvantages:
  - no parallelism
  - copying overhead

Key idea illustrated



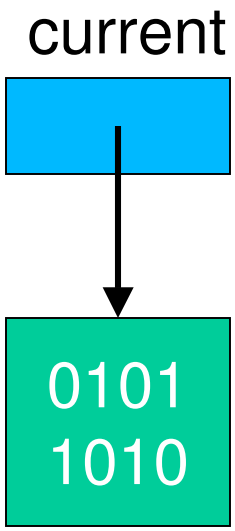
- Simplified (and wrong!) hybrid optimization approach:
  - use hardware TM to modify copy in-place, and
  - check current pointer doesn't change

Key idea illustrated



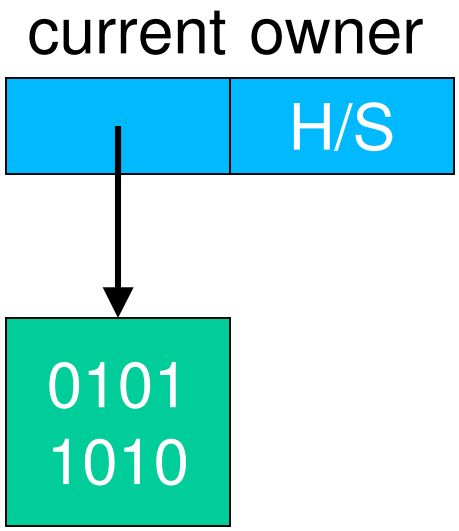
- Advantages:
  - Nonconflicting operations can succeed in parallel
  - no copying

Key idea illustrated



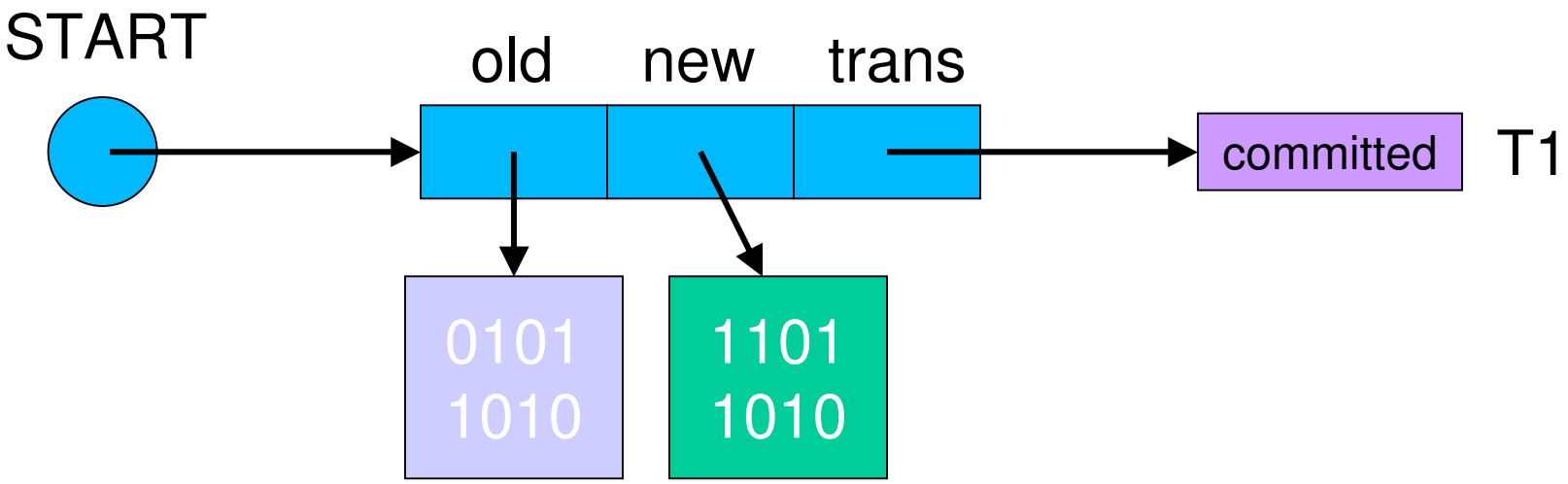
- Advantages:
  - Nonconflicting operations can succeed in parallel
  - no copying
- Disadvantage:
  - not correct ☺
  - software operations might copy inconsistent data

Key idea illustrated

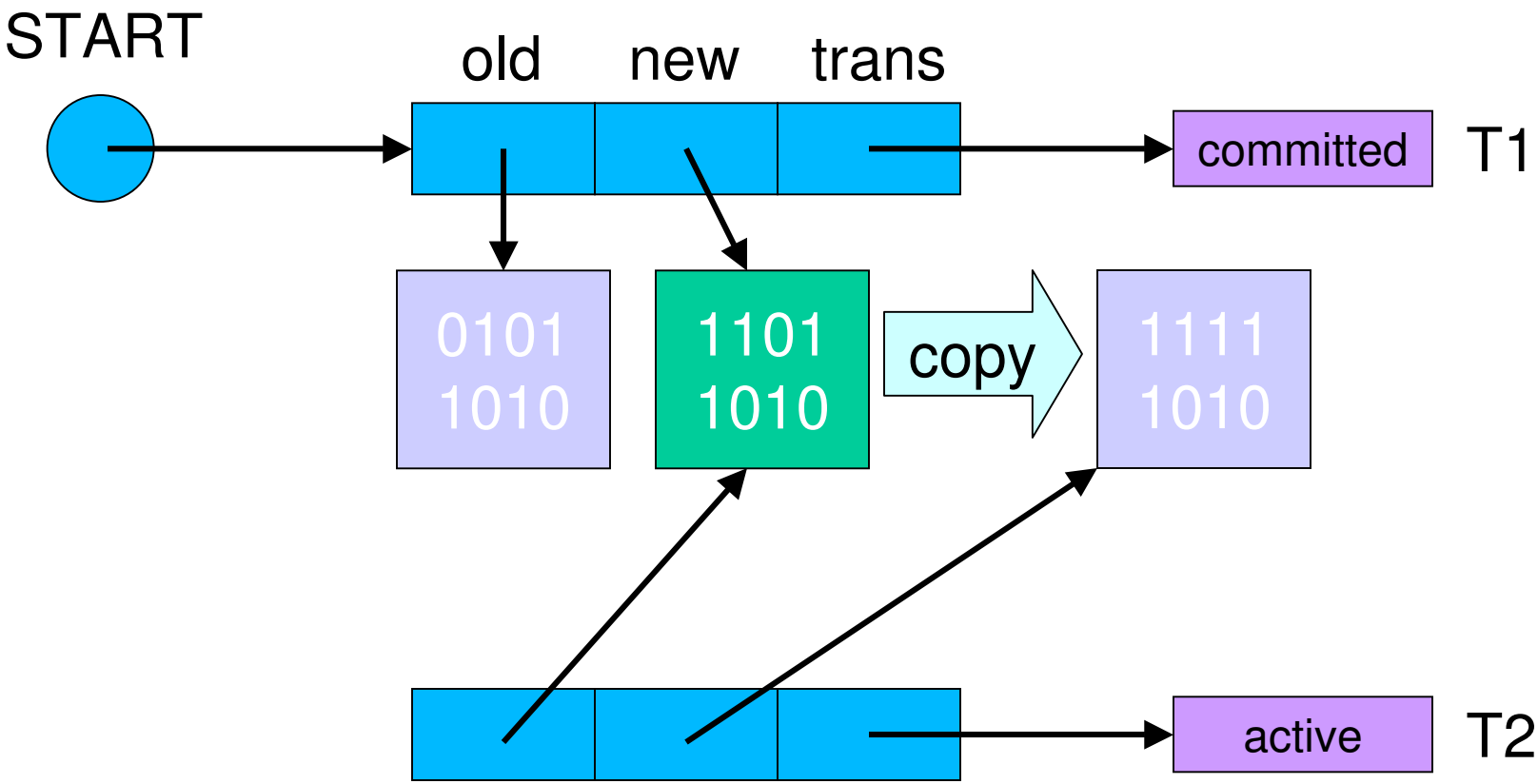


- Solution:
  - indicate “ownership” by software operation(s)
  - software ops acquire ownership before copying
  - hardware ops abort if object owned by software

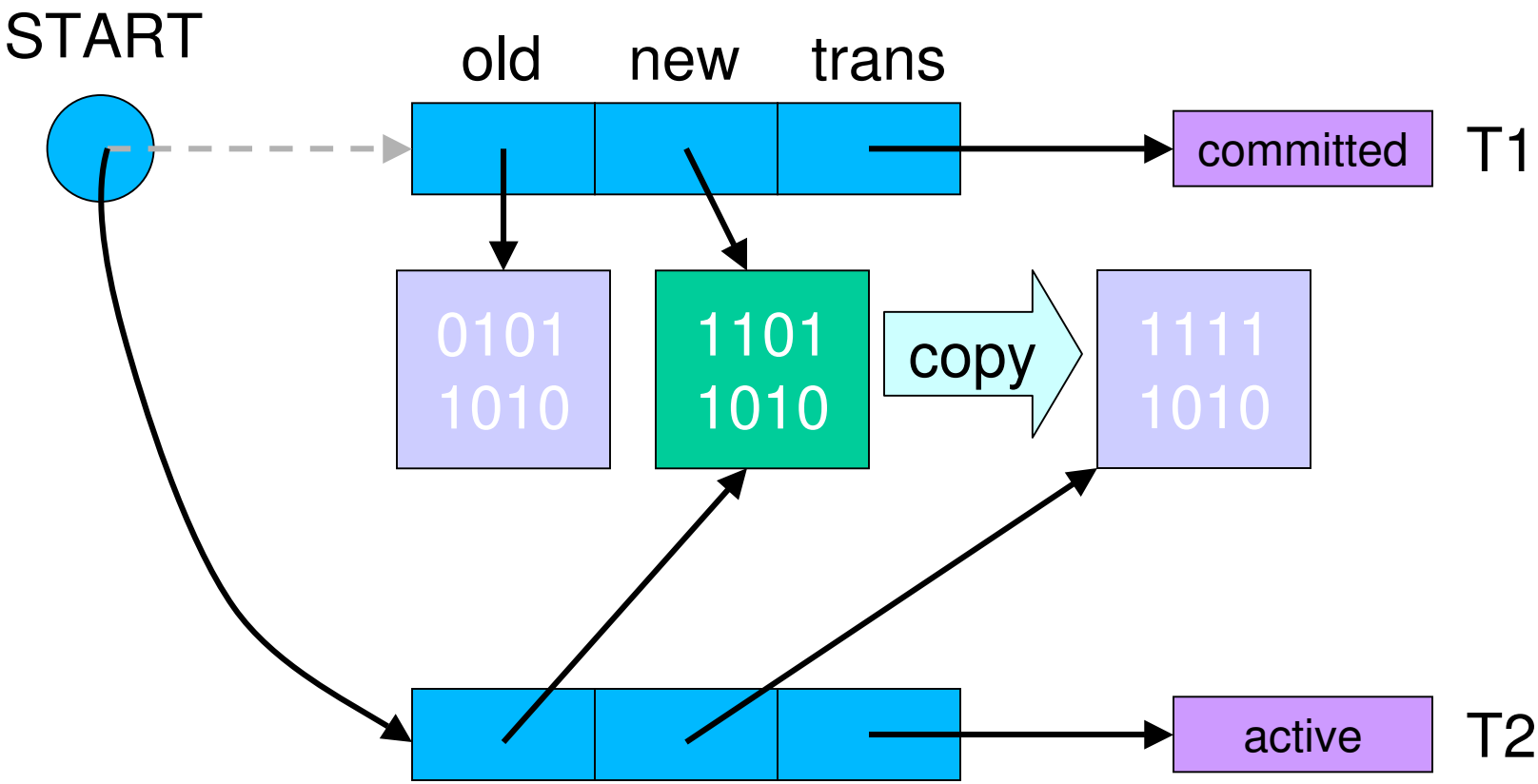
Example: DSTM



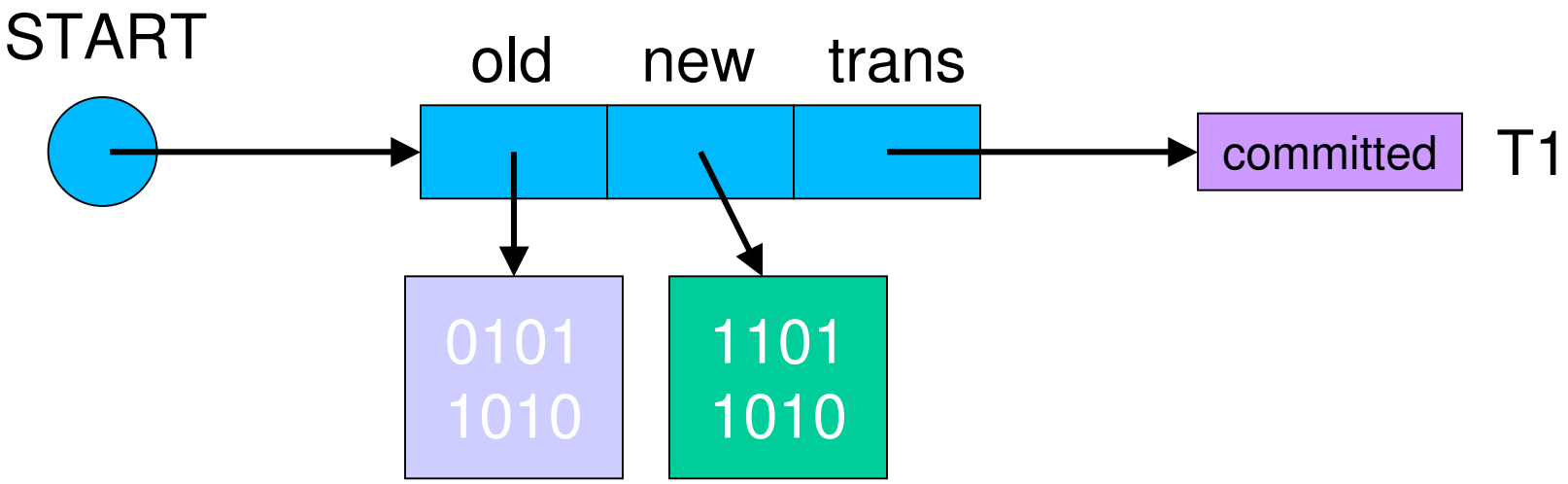
Example: DSTM



Example: DSTM



Hybrid DSTM



- Use hardware TM to check object not owned by active software txn, and if not modify current object in-place
- Avoids copying and expensive synchronization, allows intra-object parallelism

An Aside

- Level of indirection because of single-word CAS, multi-word locator
- H/W TM that guarantees (eventual) success of single-cache-line txn would remove indirection simply.
- Other simple guarantees useful too: e.g., txns that read one cache line and write one cache line

Word-based Hybrid TM

- TM useful at lower levels too, e.g. JVM, GC, etc.
- No object infrastructure, want to modify data in-place

Word-based Hybrid TM Prototype

- Similar in structure to Harris & Fraser STM (OOPSLA 2003)
- Ownership table; data transparency
- Compiler emits code for hardware txn and software txn (library calls), and retry/contention mgmt code
- Hardware transactions augmented to check ownership table for conflicting software txns (multiple levels of granularity possible)

Word-based Hybrid TM Prototype

- Shoe-horned into existing languages to allow experimentation with existing code, e.g. GC, JVM, etc.
- Programming model primitive, but workable
- Remaining challenges include:
  - better language integration
  - validation of hybrid approach

Concluding Remarks

- Hybrid Transactional Memory
  - Use best-effort hardware TM to boost performance of self-contained STM
  - Eases constraints on hardware designers
    - Best effort ok, but simple guarantees desirable
  - Eases path to adoption of transactional programming