Transactional Memory for the Masses
(ASTM, dual data structures)

Michael L. Scott
University of Rochester
(joint work with Bill Scherer & Virendra Marathe)
8 April 2005

Practical *Nonblocking* STM Systems

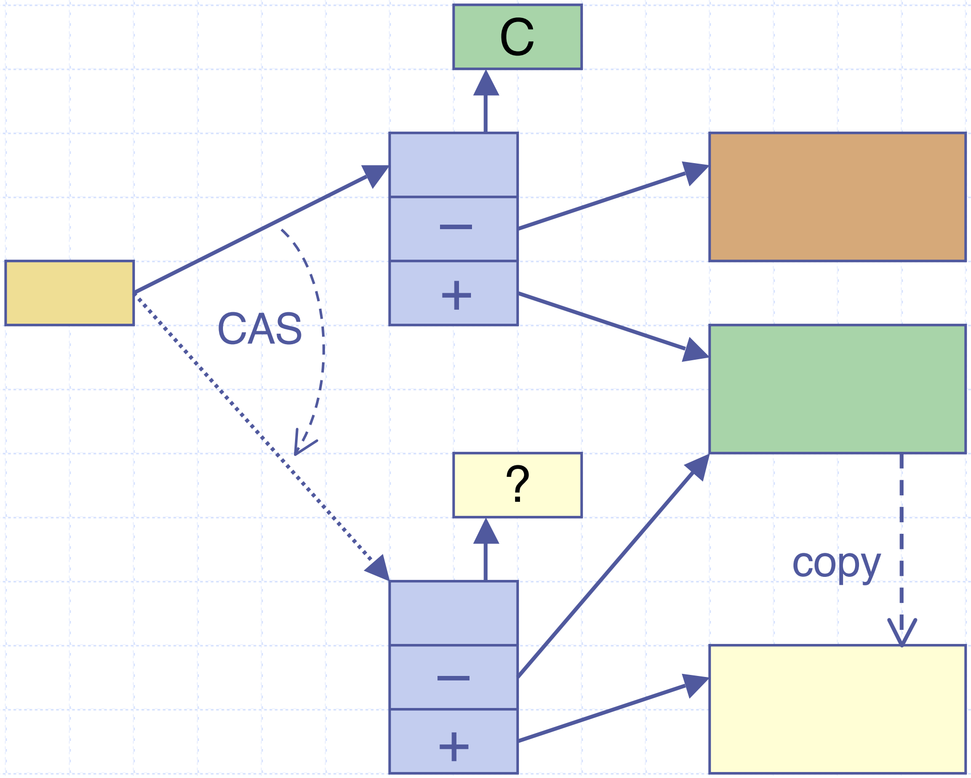
- Sun DSTM, Cambridge OSTM, . . .
  - Space linear in total #objects or #objects currently in use, w/ very small constant
  - Time w/in a modest constant factor of fine-grain locks in the absence of contention
  - Faster than coarse-grain locks in the presence of contention, with comparable programmer effort
  - (not to mention failure/preemption/page fault tolerance, deadlock/inversion freedom, . . .)

Systems vary in:

- Granularity – word-based, object-based
- Progress model – lock-free, obstruction-free
- Indirection overhead – in-place mutation,
  pointer to object, pointer to pointer to object
- Acquire semantics – eager v. lazy, read v. write,
  (validation overhead)

➢ These are not independent!

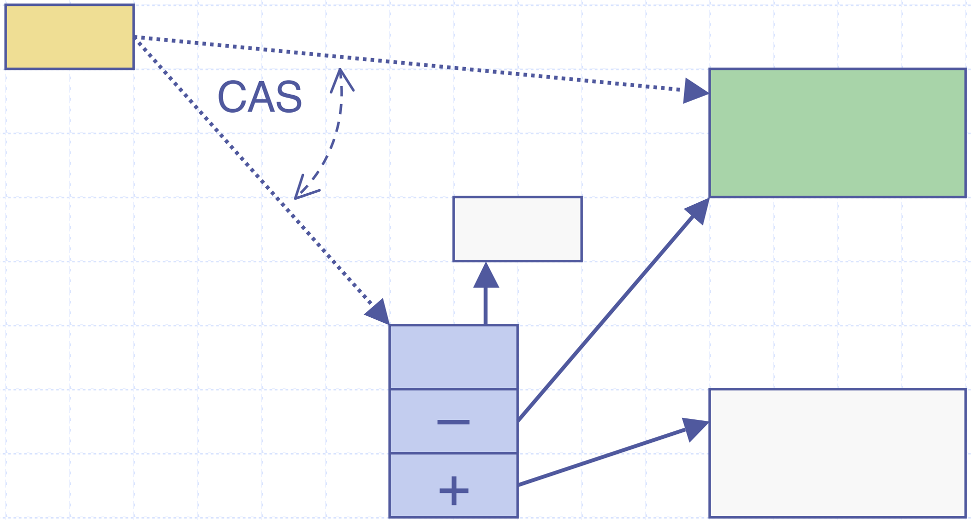
Sun DSTM [Herlihy, Luchangco, Moir, Scherer]



- Object-based
- Obstruction-free (w/ contention management)
- Double indirection
- Eager acquire for write; optional reader list

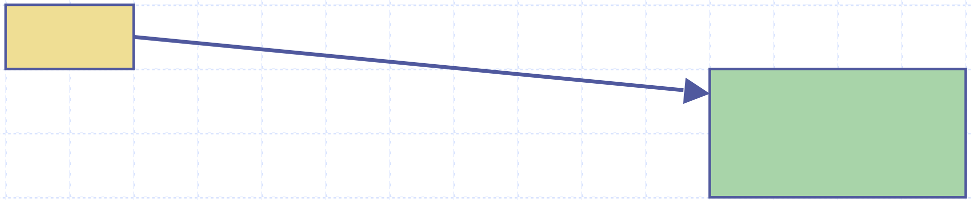
Rochester ASTM

- Optional locators; both eager and lazy acquire



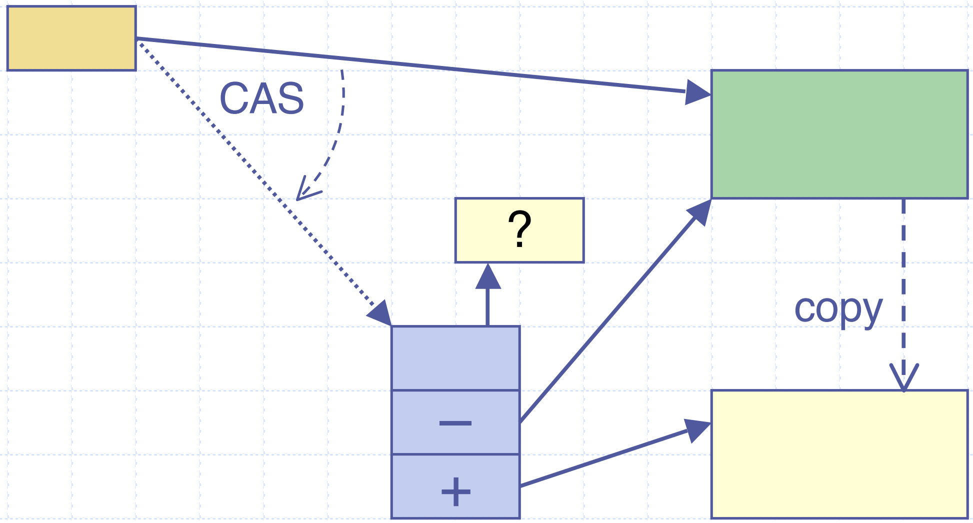
Rochester ASTM (2)

- Single-indirection for mostly-read object



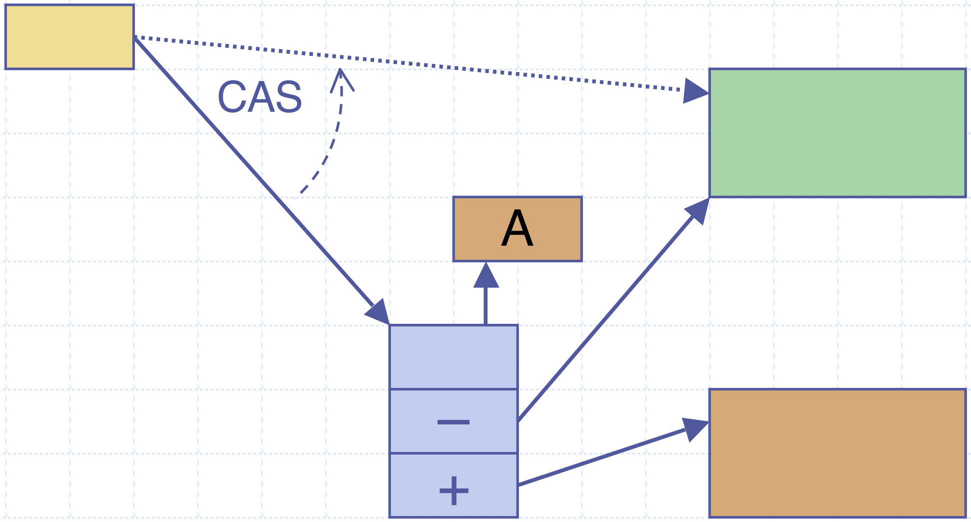
Rochester ASTM (3)

- Writer installs DSTM-style locator; retained by subsequent writers



Rochester ASTM (4)

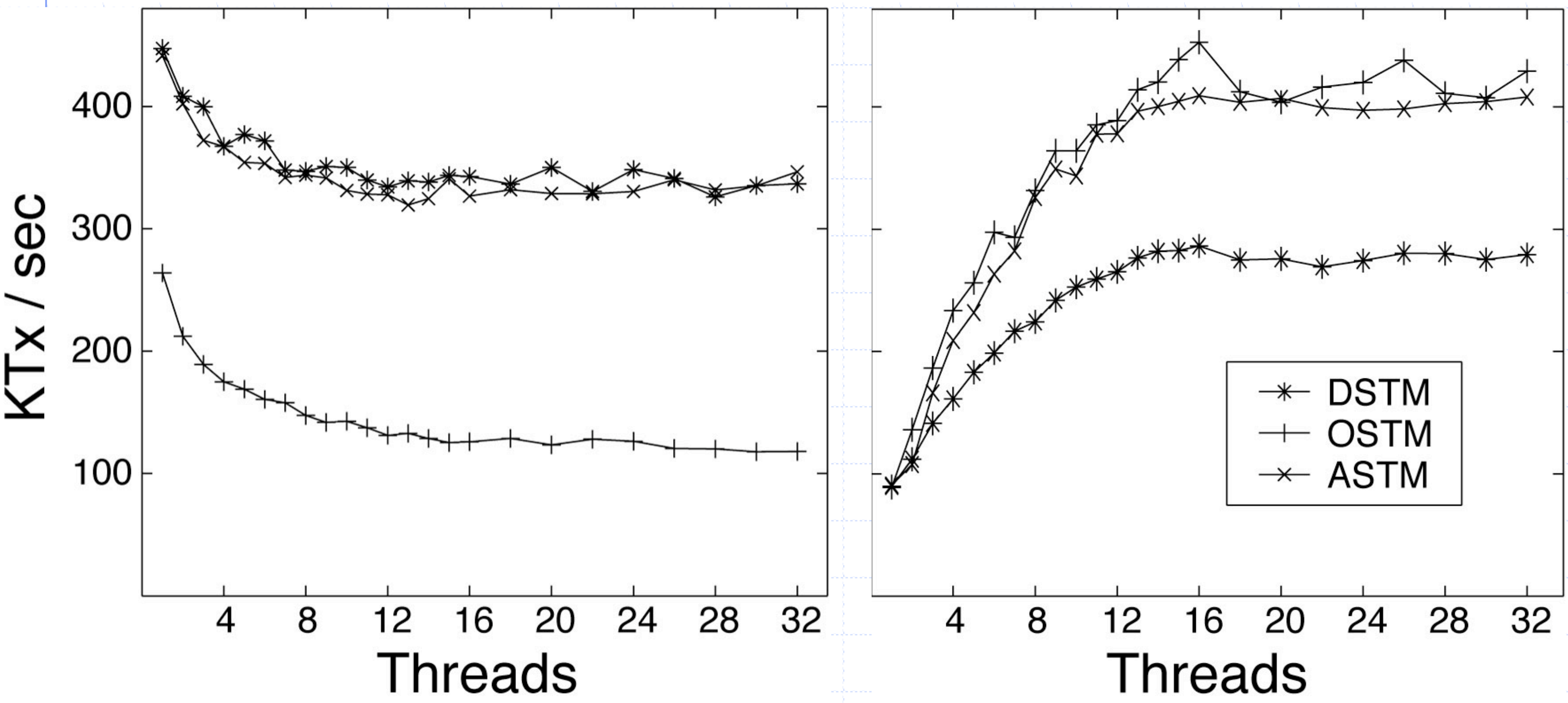
- Reader reverts to single indirection



- Avoid indirection when reading
- Detect conflicts early when writing
- Lazy acquire also an option

> Contention management

ASTM Performance



LFUCache

RBTree

STM Challenges

- Finding the right programming model
- Integrating with existing models, languages, compilers
- Overhead reduction
- Hardware accelleration/hybrids; portability across platforms

- Condition synchronization

Dual Data Structures [DISC'04]

- Don't fail; insert request instead
  - explicit (fair) control of request ordering
- Request and *successful* follow-up are nonblocking
- Contention-free waiting: *unsuccessful* follow-ups perform no remote references
  - spinning or scheduler-based
- Compatible with Java, C#, etc.
  - avoid "covering conditions," repeated testing
- SynchronousQueue, Exchanger for JSR166

www.cs.rochester.edu/~scott/synchronization/