Beyond Transactions

The Evolution of Transactional Memory

Christos Kozyrakis

Computer Systems Lab
Stanford University

http://csl.stanford.edu/~christos

Need Some HW Support for TM?

- IMHO: Yes!
- Reason #1: parallel programming is about performance
  - Can we afford a 2x overhead for SW TM?
  - Short-term: reward for extra coding effort
  - Long-term: energy efficiency
- Reason #2: TM hardware has good uses beyond concurrency
  - These uses put significant stress on TM system
  - If they cause slowdowns, users will turn them off

TM Uses Beyond Concurrency

- TM = mechanisms for ACI
  - Atomicity, isolation, and consistency
- ACI are the building blocks for
  - Debugging: deterministic replay, watchpoints, bookmarks
  - Tuning: contention & locality monitoring
  - Security: isolated execution, canaries
  - Fault-tolerance: undo-on-error, checkpointing
  - Dynamic binary translation: fix metadata races
- What is different with these “transactions”?
  - They vary significantly in duration, read/write-set size
  - Some exhibit no locality; other difficult to analyze
  - May want to have transactions on all the time

The HW/SW Interface for TM
(see McDonald et.al at ISCA'06)

- The interface mechanisms
  - SW handlers on all TM events
    - Abort, conflict, two-phase commit
  - Multiple (potentially nested) transactions
    - Allow concurrent uses (parallelism, security, …)
    - Support for system & PL tools
  - Ways to turn versioning/conflict detection on/off
    - Fine-grain & coarse-grain
- Use them to build higher levels of abstraction
  - E.g. TM-safe collection classes [PPoPP'07]
- Many HW systems match this interface
  - "Hardware TMs" and "hybrid TMs"

Are HW Signatures Sufficient?

- Issue #1: performance guarantees
  - We already deal with some unpredictability
    - Caches, predictors, schedulers, routing schemes, etc
  - Can alleviate with SW and new HW
- Issue #2: loss of information
  - Uncertainty about address, false conflicts …
    - Maybe OK for parallelism but not for other uses
  - Solution: HW signatures as a first-level filter
    - Retrieve more accurate info with SW techniques
    - Further work needed, but I believe it will work
- My conclusion: thumbs up for signatures
  - Difficult to beat their cost-effectiveness
  - Can always combine with other SW/HW schemes

Beyond Memory Transactions

- TM is good but not good enough
  - Users want generalized transactions
  - Atomic{} that access memory, files, network, …
- System-level transactions
  - Use transactional mechanisms with all resources
    - Mem ⇒ TM, FS ⇒ LFS, Net ⇒ message queues, DBMS
  - Coordinate transactions across multiple resources
  - General transaction model with few(er) restrictions (?)
- Historical example: IBM's Quicksilver ('80s)
  - Coordinated transactions across files, network, …
  - Pre-TM ☺