Exploiting Semantic Commutativity in Hardware Speculation

Guowei Zhang, Virginia Chiu, Daniel Sanchez
{zhanggw, vchiu, sanchez}@csail.mit.edu

Motivation
Exploiting commutativity helps update-heavy apps by reducing communication and synchronization
- Commutative ops produce equivalent results when reordered → no true data dependence

Software techniques that exploit commutativity incur high run-time overheads
- e.g., STM is 2-6x slower than HTM

Prior hardware limited to simple commutative ops

Semantic commutativity

Strict commutativity
- Reordering produces exactly the same final states

Semantic commutativity
- Reordering produces semantically equivalent results

CommTM overview
- Leverages HTM to support multi-instruction updates
- Extends coherence protocol to perform commutative updates locally and concurrently
- Benefits speculative execution by reducing conflicts

Example: Transactions X0–X4 increment a shared counter, and X5 reads it

Baseline HTM
- Eager conflict detection
- Lazy versioning (buffer speculative data in L1s)
- Timestamp-based conflict resolution

CommTM can be applied to other HTMs and hardware speculation techniques

Coherence protocol

Speculative value management
- Manage U-state data similarly to M-state data
- Reuse L1 speculation status bits

Conflict detection scheme
- Conflict scenarios
- Non-speculative reductions
- Use shadow hardware threads or interrupts

Evaluation

Programmed interface

Transaction update
void add(int* counter, int delta) {
    tx_begin();
    int v = load[ADD](counter);
    nv = v + delta;
    store[ADD](counter, nv);
    tx_end();
}

Non-transactional reduction
void add_reduce(int* counterLine, int* deltas) {
    for (int i = 0; i < intPerCacheLine; i++) {
        int v = load[ADD](counterLine[i]);
        int nv = v + deltas[i];
        store[ADD](counterLine[i], nv);
    }
}

Gather requests

Motivation: Frequent reductions triggered by condition checks limit concurrency

Gather requests allow updates to move across caches without leaving the reducible state, achieving higher concurrency (e.g., for reference counting)

Conventional HTM

CommTM

Less traffic
Higher concurrency
Less wasted transactional work

Microbenchmark results

Full application results

Boruvka
Kmeans
Ssca2
Genome
Vacation

CommTM
Baseline