Invyswell: A HyTM for Haswell RTM

Irina Calciu, Justin Gottschlich, Tatiana Shpeisman, Gilles Pokam, Maurice Herlihy
Multicore Performance Scaling

- Problem: Locking
- Solution: HTM?
  - IBM BG/Q, zEC12, POWER
  - Intel Haswell TSX

Source: embedded.com
Restricted Transactional Memory (RTM)

\[
\text{xbegin()}
\]

Atomic region called transaction

\[
\text{xend()}
\]

Execute optimistically, without any locks

Read and Write Sets

Abort on memory conflict: programmer defined behavior
if (xbegin() == XBEGIN_STARTED)

Execute Transaction
xend()

else

Execute Fallback Path
Lock Elision

Source: Anand Tech
Why Lock Elision Is Not Enough
InvalSTM (prior work)

- [Gottschlich et al., CGO 2010]
- Scalable
- Good for large transactions
- Conflict detection using bloom filters
InvalSTM Software Transaction (prior work)

Main body of SW txn

On read:
Add to read Bfilter

On write:
Add to write Bfilter
Add writes to hash table

If can_commit() Invalidation
Else restart

update memory
InvalSTM Invalidation (prior work)

Inflight Transactions

Conflicts? (using bloom filters)

Can I commit?

Committing Transaction

Contention Manager
InvalSTM Invalidation (prior work)

Inflight Transactions

Committing Transaction

Contention Manager

no

yes

yes

no

ABORT
InvalSTM Invalidation (prior work)

Inflight Transactions

Contention Manager

Aborted
InvalSTM Invalidation (prior work)
InvalSTM Invalidation (prior work)
Software Transaction (InvalSTM)
Hardware Transaction + Invalidation

- HW Txn
- Invalidation
- Commit

Time

ABORT
Hardware Transaction + Invalidation

- **Commit (Check BF)**
- **Invalidation**

COMMIT

Already committed, can’t abort
Software Transaction (Modified InvalSTM)
\[ x = 2; \; y = 1; \]

SW Transaction 1
(commit)

\[ x++; \]
\[ y++; \]

SW Transaction 2
(execution)

Read \( x \);

Read \( y \);
\[ z = \frac{1}{x - y}; \]

Time

ABORT

\[ z = 1/0!!! \]
Read Validation

SW Transaction 1
(commit)

SW Transaction 2
(execution)

Check BF

ABORT

Read x;

Read y;

z = 1/(x - y);

x++;

y++;
**SPEC SW (Speculative Software)**

- **Begin SW txn, increment sw_cnt**
- **Main body of SW txn**
  - **On read:** Validate and add to read Bfilter
  - **On write:** Add to write Bfilter
    - Add writes to hash table
  - Acquire commit_lock
  - **Validate**
    - If can_commit() **update memory**
    - Else release lock and restart
  - Invalidation
    - Decrement sw_cnt, release lock
BFHW (Bloom Filters Hardware)

Main body of HW txn.

**On Read:** add to read Bfilter

**On Write:** add to write Bfilter

if (commit_lock)
    if (BF conflict()) xabort()

xend()
\[ x = 2; \, y = 1; \]

**HW Transaction 1** (commit)
- \( x++; \)
- \( y++; \)

**SW Transaction 2** (execution)
- Read \( x; \)
- Read \( y; \)
- \( z = \frac{1}{(x - y)}; \)

**ABORT**

**ABORT**

**z = 1/0!!!**
Read Validation

HW Transaction 1
(commit)

x++;
y++;

Check BF

SW Transaction 2
(execution)

Read x;

Check BF

Read y;

z = 1/(x - y);
Main body of HW txn.

**On Read**: add to read Bfilter
**On Write**: add to write Bfilter

if (commit_lock)
    if (BF_conflict()) xabort()
++hw_post_commit;
xend()

Invalidation
--hw_post_commit (fetch_and_sub)
Read Validation

HW Transaction 1
(commit)

x++;
y++;

SW Transaction 2
(execution)

Wait for hw_post_commit == 0

Read x;

Wait for hw_post_commit == 0

Read y;
z = 1/(x - y);
**SPEC SW**

- **Commit**
- **Post-Commit**

**BFHW**

- **Commit**
- **Post-Commit**

- **On Read:** add to read Bfilter
- **On Write:** add to write Bfilter

_Expensive!_
LITEHW (Light Hardware)

Main body of HW txn.

Time

HW

Commit

if (sw_cnt) xabort();
else xend()
Ensuring Progress

Inflight Transactions

Committing SW Transaction

Can I commit?

Contention Manager
Ensuring Progress

Inflight Transactions

Committing HW Transaction

Committing SW Transaction

Contention Manager

Can I commit?
IRREVOCSW (Irrevocable Software)

Does not abort – Guarantees Progress

Main body of SW txn.

**On Read:** add to read Bfilter
**On Write:** add to write Bfilter
Use direct updates

Time

Acquire commit lock, increment sw_cnt

Do nothing
(Changes are already committed)

Invalidation **Expensive**!
Decrement sw_cnt, release lock
SGLSW (Single-Global-Lock Software)

- **Acquire commit lock, increment sw_cnt + ++commit_sequence**
- **Main body of SW txn.**
- **Use direct updates**
- **Do nothing**
  (Changes are already committed)
- **Post-Commit**
  - Decrement sw_cnt, release lock
Invyswell State Diagram

- **Start**
  - **SW txns running?**
    - no
      - retry
      - **LiteHW**
        - retry threshold exceeded
          - large txns with unsupported HTM instructions / overflow
            - retry
        - retry threshold exceeded
    - yes
      - retry
      - **BFHW**
        - retry threshold exceeded
      - **SglSW**
        - retry threshold exceeded
      - **SpecSW**
        - retry threshold exceeded
          - conflict
      - **IrrevocSW**
**Invyswell State Diagram**

Start -> Fail-fast? (no) -> SW txns running? (yes) -> SglSW

Start -> Fail-fast? (yes) -> retry

Fail-fast? (no) -> SW txns running? (no) -> retry

SW txns running? (yes) -> small txns with unsupported HTM instructions / fail-fast

SW txns running? (no) -> LiteHW

LiteHW -> retry threshold exceeded -> large txns with unsupported HTM instructions / overflow

LiteHW -> retry

SglSW -> retry threshold exceeded

SglSW -> retry

BFHW -> retry threshold exceeded

BFHW -> retry
**Concurrent Execution Matrix**

<table>
<thead>
<tr>
<th>Types</th>
<th>BFHW</th>
<th>LiteHW</th>
<th>SpecSW</th>
<th>IrrevocSW</th>
<th>SglSW</th>
</tr>
</thead>
<tbody>
<tr>
<td>BFHW</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>LiteHW</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>SpecSW</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>IrrevocSW</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>SglSW</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>
Speedup
kmeans low.
Speedup

Labyrinth.
Speedup

Yada.
Transaction Types - 1 Thread

The chart shows the percentage of transactions for various benchmarks. The benchmarks are: bayes, genome, intruder, kmeans low, kmeans high, labyrinth, ssc2, vacation low, vacation high, and yada. The chart uses different colors to represent different types of transactions: SGL, SPEC, BFHW, and LITE.
Conclusions

- HLE and RTM w/ SGL fallback are not enough

- Invyswell is 35% faster than NOrec, 18% faster than Hybrid NOrec and 25% faster than HLE across all STAMP benchmarks
Thank you!

- http://cs.brown.edu/~irina
- irina@cs.brown.edu