Reasoning about Transactional Memory

Transitioning ideas from academia to industry

Nathan Chong, John Wickerson and Tyler Sorensen

FMATS-5
21 September 2017
Architecture: fundamental contract between hardware/software
Multiprocessor program

\[ T_0 \ T_1 \ \ldots \ T_{n-1} \]

a sequence of assembly instructions (ADD, CMP, B, LDR, STR, …)

Essential Question: What values can this load return?

Memory Models

<table>
<thead>
<tr>
<th>Stronger</th>
<th>Sequential Consistency</th>
<th>Weaker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpha</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power</td>
<td>ARM</td>
<td></td>
</tr>
<tr>
<td>ARM v8.0</td>
<td>RISC-V</td>
<td></td>
</tr>
<tr>
<td>x86 (TSO)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Transactional Memory: Optimistic Concurrency
Transactional Memory: Optimistic Concurrency

Shared Memory

Tx₀

Tx₁

Conflict!
Transactional Memory: Optimistic Concurrency
Transactional Memory: Optimistic Concurrency

Shared Memory

Replay Tx₀
ISA Changes: TXSTART, TXCOMMIT, TXABORT, TXTEST

Impact on Memory Model

Interaction with exceptions, virtualisation, vector-extensions, debug, …

Unintended consequences
- KASLR attack [JLK16]
- Prime+Abort [DKP+17]
This Talk

- Principled method for refining TM models
  - x86, Power, Armv8, C++
  - Automatic generation of minimal conformance testsuites
  - Transferring this technique to engineers

- The tricky case of aborting transactions
\textbf{Final values of }r_0, r_1\textbf{ }

\begin{center}
\begin{tabular}{c|cccc}
 & 1,1 & 1,0 & 0,1 & 0,0 \\
\hline
\textsc{sc} & ✓ & ✓ & ✓ & x \\
\textsc{x86} & ✓ & ✓ & ✓ & ✓ \\
\end{tabular}
\end{center}

\begin{align*}
x &= 1; \\
r_0 &= y; \\
y &= 1; \\
r_1 &= x;
\end{align*}
Final values of r0, r1

<table>
<thead>
<tr>
<th></th>
<th>1,1</th>
<th>1,0</th>
<th>0,1</th>
<th>0,0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>x86</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Store Buffer
tx { x=1; r0=y; }  

tx { y=1; r1=x; }

<table>
<thead>
<tr>
<th>Final values of r0, r1</th>
<th>1,1</th>
<th>1,0</th>
<th>0,1</th>
<th>0,0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>x86</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>x86+TM</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
</tbody>
</table>
\[ x = 1; \quad r_0 = y; \]

\[ y = 1; \quad r_1 = x; \]
The set of all executions
The set of all executions

Finding minimal disallowed executions [LWP+17]
The set of all executions

Finding minimal disallowed executions [LWP+17]
The set of all executions

Finding *minimal* disallowed executions [LWP+17]

N is stronger-than M

M is weaker-than N
The set of all executions

Finding minimal disallowed executions [LWP+17]

N is stronger-than M

M is weaker-than N

Finding a distinguishing execution in this set M\N [WBS+17]
The set of all executions

Finding **minimal** disallowed executions \([LWP+17]\)

\(N\) is stronger-than \(M\)

\(M\) is weaker-than \(N\)

Finding a **distinguishing** execution in this set \(M \setminus N\) \([WBS+17]\)

E.g., Let \(M=x86\) and \(N=SC\)
Then \(M \setminus N\) includes the store-buffering execution
Combine [LWP+17, WBS+17]

- Base (e.g., x86)
- Base+TM

SC

- Find minimal distinguishing executions
- Forbid tests
- Allow tests

TSC

Weaker

Stronger

Run on HW and use results to refine model
Table 2. Empirical testing of our transactional x86 model on Intel Haswell and Broadwell machines

| $|E|$ | Solve (Sec) | T | S | ¬S | Solve (Sec) | T | S | ¬S |
|-----|-------------|---|---|----|-------------|---|---|----|
| 2   | 1           | 2 | 0 | 2  | 1           | 0 | 0 | 0  |
| 3   | 2           | 6 | 0 | 6  | 1           | 0 | 0 | 0  |
| 4   | 6           | 26| 0 | 26 | 3           | 6 | 2 | 4  |
| 5   | 191         | 45| 0 | 45 | 47          | 10| 4 | 6  |
| 6   | 3600*       | 167| 0 | 167| 3600*       | 38| 16| 22 |
| 7   | 3600*       | 372| 0 | 372| 3600*       | 79| 4 | 75 |
| 8   | 3600*       | 514| 0 | 514| 3600*       | 124| 4 | 120|
| 9   | 3600*       | 37 | 0 | 37 | 3600*       | 21| 0 | 21 |
| 10  | 3600*       | 9  | 0 | 9  | 3600*       | 4 | 0 | 4  |
| Sum | 1178        | 0  | 1178 | 282 | 30 | 252 |

Note: * indicates floating point numbers.
Results

- Experimentally validated x86 TSX and Power TM models
- Proposals for Armv8 and C++ TM extensions
- Small additions to each model
  - Strong isolation
  - Transaction ordering (including implicit barriers)
  - Transaction propagation (Power-only)
- Methodology transferred to architecture-validation team in Arm
Failing Transactions

successful  

failing
TXSTART fail
// ...
// Body
// ...
TXCOMMIT

fail:
// ...
// Fail handler
// ...
Abort causes state rollback: how do we get visibility inside a failing tx?
TXSTART fail
// ... // Body // ...
TXCOMMIT

fail:
// ...
// Fail handler // ...

fail => rollback state and branch to handler
TXSTART fail
// ...
// Body
// ...
TXABORT #VAL

fail:
// ...
// Fail handler (TXSTATUS.reason==VAL)
// ...

fail => rollback state
and branch to handler
Failing Transactions

a: \( W[x] = 1 \)
b: \( W[y] = 1 \)
c: \( R[y] = 0 \)
d: \( R[x] = 1 \)

TXSTART fail
LDR W0, [X1] // c
LDR W2, [X3] // d
// if W0==0 && W2==1
TXABORT #1
// else
TXABORT #0
fail:
// TXSTATUS.reason==1
Future Work

- Transactional lock elision correctness
- Specifying TM operationally
- Fairness and forward-progress
- Interaction with PTW, exceptions, ... 
- What about Opacity?
Reflection

- Automatic generation of minimal conformance testsuites
- Minimality (close to the boundary)
- Distinguishing
- Automated
- Output is very understandable

- Value of not-observing a forbidden test?
- Value of not-observing an allowed test?
We’re hiring!

The security group is interested in the design, implementation and application of testing and verification at all levels of the system stack.

Senior Formal Verification Researcher
Specifying and verifying real-world systems
www.arm.com/careers (search: 10720)
References

[AMT14] Herding cats: Modelling, Simulation, Testing, and Data-mining for Weak Memory (Alglave, Maranget, Tautschnig)
[CMF+13] Robust Architectural Support for Transactional Memory in the Power Architecture (Cain et al.)
[Deacon17] ARMv8.0 Application Level Memory Model (Deacon)
[DS09] Strong Isolation is a Weak Idea (Dalessandro, Scott)
[HM93] Transactional Memory: Architectural Support for Lock-Free Data Structures (Herlihy, Moss)
[LWP+17] Automatic Synthesis of Comprehensive Memory Model Litmus Test Suites (Lustig et al.)
[MBL06] Subtleties of Transactional Memory Atomicity Semantics (Martin, Blundell, Lewis)
[WBS+17] Automatically Comparing Memory Consistency Models (Wickerson et al.)
Reasoning about Transactional Memory

Transitioning ideas from academia to industry

Nathan Chong, John Wickerson and Tyler Sorensen

FMATS-5
21 September 2017