COATCheck, Future Possibilities, and Wrap-up
COATCheck: Verifying Memory Ordering at the Hardware-OS Interface

Daniel Lustig, Geet Sethi⁺, Michael Pellauer*, Margaret Martonosi, Abhishek Bhattacharjee⁺

Princeton University  ⁺Rutgers University  ⁺NVIDIA

ASPLOS 2016

http://check.cs.princeton.edu/
## Simple Motivating Example

Initially: \([x]=0, [y]=0\)

<table>
<thead>
<tr>
<th>Thread 0</th>
<th>Thread 1</th>
</tr>
</thead>
</table>
| St \([x]\) \(
\)\(\rightarrow 1\) | St \([y]\) \(
\)\(\rightarrow 2\) |
| Ld \([y]\) \(\rightarrow r1\) | Ld \([x]\) \(\rightarrow r2\) |

Proposed outcome: \(r1=2, r2=1\)

---

Permitted if \(x\) and \(y\) are different addresses

<table>
<thead>
<tr>
<th>Initially: ([x]=0, [y]=0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread 0</td>
</tr>
<tr>
<td>St PA1(\rightarrow 1)</td>
</tr>
<tr>
<td>Ld PA2(\rightarrow r1)</td>
</tr>
</tbody>
</table>

Outcome \(r1=2, r2=1\) permitted

---

Forbidden if \(x\) and \(y\) are synonyms

<table>
<thead>
<tr>
<th>Initially: ([x]=0, [y]=0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread 0</td>
</tr>
<tr>
<td>St PA1(\rightarrow 1)</td>
</tr>
<tr>
<td>Ld PA1(\rightarrow r1)</td>
</tr>
</tbody>
</table>

Outcome \(r1=2, r2=1\) forbidden
“Transistency Model”

- Memory ordering verification is fundamentally incomplete unless it explicitly accounts for address translation
- Superset of consistency which captures all address translation-aware sets of ordering rules
- Most prior techniques ignore the implications of virtual-to-physical address translation on memory ordering
  - E.g., synonyms, and page permission updates
- Microarchitectural events and OS behavior can affect memory ordering in ways for which standard memory model analysis can be fundamentally insufficient
Ongoing Work

- We’ve seen how memory model bugs can result in incorrect program outcomes that are intermittent/unpredictable.

- Currently, we are applying our techniques of exhaustive enumeration and checking of event orderings to other domains:
  - Security – does an underlying implementation provide certain security guarantees?
  - IoT – how do we reason about many concurrently acting IoT devices?
Takeaways

- Memory consistency modes matter
  - Reliability, correctness, and portability
  - Performance
  - Security

- Intuitive “checking” through automated verification

- Move memory model verification earlier in the design processes

- Evaluate across interfaces and design boundaries
  - If interfaces are often source of bugs

- Speed of approach enables new opportunities
  - Comprehensive and fast verification for iterative design
http://check.cs.princeton.edu/