Partial Incoherence (GPU) Case Study

- Counterruine outcome for mp with membar [Aglave et al. ASPLOS15]
- If load-load fence does not ensure InvCache ordering, no cycle

Lazy Coherence Case Study (TSO-CC)

- No eager invalidation of sharers, but self-invalidates on L1 miss
- InvCache edges model self-invalidates and complete cycle for mp

Coherence-Consistency Interface (CCI)

CCI = guarantees that coherence protocol provides to rest of microarchitecture + memory ordering
 guarantees that rest of microarchitecture expects from coherence protocol

CCI Match $\Rightarrow$ Consistency Maintained!

CCI Mismatch $\Rightarrow$ Consistency Violation!

Microarchitectural happens-before (hbp) graphs with ViCLs

- Executions modelled by hbp graphs
- Node $\rightarrow$ microarchitectural event or pipeline stage
- Edge $\rightarrow$ local happens-before relation between nodes

Microarchitectural happens-before (hbp) graphs with ViCLs

VCIL Create and Expire map to nodes

Traditional Cache Line States for co-mp

ViCLs for co-mp, including requests and downgrades

Coherence-Casiness Example (“Peekaboo”)

- Prefetch + Liveloop Avoidance Mechanism + Inv Before Use $\Rightarrow$ Consistency Violation!

<table>
<thead>
<tr>
<th>Core 0</th>
<th>Core 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i1) St $x \leftarrow 0$</td>
<td>(i2) St $x \leftarrow 1$</td>
</tr>
<tr>
<td>(i3) Ld $y \leftarrow x$</td>
<td>(i4) Ld $y \leftarrow x$</td>
</tr>
<tr>
<td>Under TSC: Forget $y = 0$, $2 = 0$</td>
<td></td>
</tr>
</tbody>
</table>

- Counterintuitive outcome for et al. ASPLOS15
- CCICheck can handle ordering, no cycle
- Avoidance Mechanism $+$ sll graph for value $= Core 1$
- Independent verification of coherence and consistency
- No Maps onto period to S, send [x = 0] to core 1 and publicly available at Models Before Use = ensure that coherence request for instr. that uses

Conclusions

- CCI verification is critical to the correct operation of large or complex parallel systems
- CCICheck’s static CCI-aware microarchitectural consistency verification is a first step in this direction
- CCICheck uses hbp graphs and exhaustive enumeration of all possible litmus test executions to verify a microarchitecture
- The Value in Cache Lifetime (ViCL) abstraction, constraint-based enumeration, and intelligent pruning allow comprehensive yet tractable analysis
- CCICheck can handle partial incoherence, lazy coherence, and a variety of coherence protocol transitions
- CCICheck is open-source and publicly available at github.com/yamanerka/ccicheck

Results

- CCICheck was run on a variety of microarchitectures and coherence protocols across 85 litmus tests
- Geometric test case execution time < 10 seconds on all architectures
- Subsequent research used SMT solver-based methods to run most tests in just a few seconds [ASPLOS 2016]

CCICheck Toolflow

Path Enum.

Constraint Satisfaction

Pruning (Cycle Checking)

Compare $\rightarrow$ Pass/Fail

Inputs are parse spec. and litmus test(s)

2 high-level enumeration steps: Path Enumeration & Constraint Satisfaction

Intelligent pruning and unsatisfiable constraint detection keep runtimes scalable

Coherence-Consistency Interface (CCI)

CCI = ensures that coherence protocol provides to rest of microarchitecture + memory ordering
 guarantees that rest of microarchitecture expects from coherence protocol

CCI Match $\Rightarrow$ Consistency Maintained!

CCI Mismatch $\Rightarrow$ Consistency Violation!

Constraint-Based Enumeration

- Multiple solns – each further enumerated independently
- No solns → invalid scenario
- Cyclic graphs → pruned (can’t become acyclic)

Coherence-Consistency Interface (CCI)

CCI = guarantees that coherence protocol provides to rest of microarchitecture + memory ordering
 guarantees that rest of microarchitecture expects from coherence protocol

CCI Match $\Rightarrow$ Consistency Maintained!

CCI Mismatch $\Rightarrow$ Consistency Violation!

Constraint-Based Enumeration

- Multiple solns – each further enumerated independently
- No solns → invalid scenario
- Cyclic graphs → pruned (can’t become acyclic)

Coherence-Consistency Interface (CCI)

CCI = guarantees that coherence protocol provides to rest of microarchitecture + memory ordering
 guarantees that rest of microarchitecture expects from coherence protocol

CCI Match $\Rightarrow$ Consistency Maintained!

CCI Mismatch $\Rightarrow$ Consistency Violation!

Constraint-Based Enumeration

- Multiple solns – each further enumerated independently
- No solns → invalid scenario
- Cyclic graphs → pruned (can’t become acyclic)