Chasing Away RAts: Semantics and Evaluation for Relaxed Atomics on Heterogeneous Systems

Matthew D. Sinclair, Johnathan Alsop, Sarita V. Adve
University of Illinois @ Urbana-Champaign
hetero@cs.illinois.edu

Matt’s Future: AMD Research, University of Wisconsin
“Everyone (thinks they) can cook” use relaxed atomics (RAts)

Correctness Health code violations:
- Incorrect usage
- No formal definition
- Not portable
- Hard to debug
- Out-of-thin-air values
“If you think you understand quantum computers, it’s because you don’t. Quantum computing is actually \textit{harder} than memory consistency models.”

- \textit{Luis Ceze, video in ISCA ’16 Keynote}

\textbf{Memory consistency: gold standard for complexity}

\textbf{Relaxed atomics add even more complexity}
Formal specification for relaxed atomics is a longstanding problem.
But generally use simple, SW-based coherence

– Cost of staying away from relaxed atomics too high!
Our Approach

• Previous work
  – Goal: formal semantics for all possible relaxed atomics uses
  – Unsuccessful despite ~15 years of effort

• Insight: analyze how real codes use relaxed atomics
  – What are common uses of relaxed atomics?
  – Why do they work?
  – Can we formalize semantics for them?
Contributions

• Identified common uses of relaxed atomics
  – Work queues, event counters, ref counters, seqlocks, …

• Data-race-free-relaxed (DRFrlx) memory model:
  – Sequentially consistent (SC) centric semantics + efficiency

• Evaluated benefits of using relaxed atomics
  – Up to 53% less cycles (33% avg), 40% less energy (20% avg)

Everyone can safely use RAts
Outline

• Motivation
• Background
• Data-race-free-relaxed
• Results
• Conclusion
**Atomics Background**

- Default: Data-race-free-0 (DRF0) [ISCA ‘90]
  - Identify all races as synchronization accesses (C++: atomics)

```c
// each thread
for i = 0:n
...
ADD R4, A[i], R1   synch (atomic)
ADD R5, B[i], R1   synch (atomic)
...
```

- All atomics order data accesses
- Atomics order other atomics
  - Ensures SC semantics if no data races
Atomics Background (Cont.)

• Default: Data-race-free-0 (DRF0) [ISCA ‘90]
  – All atomics order data accesses
  – Atomics order other atomics
  ⇒ Ensures SC semantics if no data races

• Data-race-free-1 (DRF1): unpaired atomics [TPDS ‘93]
  + Unpaired atomics do not order data accesses
  – Atomics order other atomics
  ⇒ Ensures SC semantics if no data races

• Relaxed atomics [PLDI ‘08]
  + Do not order data or other atomics
  ⇒ But can violate SC and no formal specification
• Motivation
• Background
• **Data-race-free-relaxed**
• Results
• Conclusion
Identifying Relaxed Atomic Use Cases

• Our Approach
  – What are common uses of relaxed atomics?
  – Why do they work?
  – Can we formalize semantics for them?

• Contacted vendors, developers, and researchers

How do relaxed atomics work in Event Counters?
Event Counter

- Threads concurrently update counters
  - Read part of a data array, updates its counter
• Threads concurrently update counters
  – Read part of a data array, updates its counter
  – Increments race, so have to use atomics
• Threads concurrently update counters
  – Read part of a data array, updates its counter
  – Increments race, so have to use atomics

Commulative increments: order does not affect final result

How to formalize?
Incorporating Commutativity Into DRFrlx

• New relaxed atomic category: commutative
• Formalism:
  – Accesses are commutative
  – Intermediate values must not be observed

⇒ Final result is always SC

What about the other use cases?
Incorporating Other Use Cases Into DRFrlx

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Category</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work Queues</td>
<td>Unpaired</td>
<td>SC</td>
</tr>
<tr>
<td>Flags</td>
<td>Non-Ordering</td>
<td></td>
</tr>
<tr>
<td>Event Counters</td>
<td>Commutative</td>
<td>Final result always SC</td>
</tr>
<tr>
<td>Seqlocks</td>
<td>Speculative</td>
<td></td>
</tr>
<tr>
<td>Ref Counters</td>
<td>Quantum</td>
<td>SC-centric: non-SC parts isolated</td>
</tr>
<tr>
<td>Split Counters</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
• Motivation
• Background
• Data-race-free-relaxed
• Results
• Conclusion
Evaluation Methodology

• 1 CPU core + 15 GPU compute units (CU)
  – Each node has private L1, scratchpad, tile of shared L2

• Simulation Environment
  – GEMS, Simics, Garnet, GPGPU-Sim, GPUWattch, McPAT

• Study DRF0, DRF1, DRFrlx w/ GPU & DeNovo coherence

• Workloads
  – Microbenchmarks for each use case
    • Relaxed atomics help a little (Avg: 10% cycles, 5% energy)
  – Benchmarks with biggest RAts speedups on discrete GPU
    • UTS, PageRank (PR), Betweenness Centrality (BC)
Relaxed Atomics Applications – Execution Time

- G0 = GPU coherence + DRF0
- G1 = GPU coherence + DRF1
- GRlx = GPU coherence + DRFrlx
- D0 = DeNovo coherence + DRF0
- D1 = DeNovo coherence + DRF1
- DRlx = DeNovo coherence + DRFrlx
Relaxed atomics reduce cycles up to ~50%

DeNovo increases reuse over GPU: 10% avg. for DRFrlx
Energy similar to execution time trends

DeNovo’s reuse reduces energy over GPU: 29% avg. for DRFrlx
Conclusion

• Cost of avoiding relaxed atomics too high
• Difficult to use correctly: no formal specification
• Insight: Analyze how real codes use relaxed atomics

DRFrlx: SC-centric semantics + efficiency

Everyone can safely use RAts