RTR: 1 Byte/Kilo-Instruction Race Recording

Min Xu  Rastislav Bodik  Mark D. Hill

Why Do You Need a Recorder?

Ideally ...

Low cost  Low runtime overhead  Long recording: small log

Applicability:
Programs – data race
Systems – non-SC
Better and Better Recorders

<table>
<thead>
<tr>
<th>Low Cost</th>
<th>Low Overhead</th>
<th>Small Log Size</th>
<th>Applicability</th>
</tr>
</thead>
<tbody>
<tr>
<td>InstRefy 87</td>
<td>8 &amp; 6 '91</td>
<td>Netzer '93</td>
<td>Dajia Yu '98</td>
</tr>
<tr>
<td>RePlay '00</td>
<td>JaRec '04</td>
<td>PDB '03</td>
<td>BugNet '05</td>
</tr>
<tr>
<td>ReNevertheless '03</td>
<td>CORD '06</td>
<td>Strata '06</td>
<td></td>
</tr>
</tbody>
</table>

Hardware Acceleration [ISCA'03]
Byte/Kilo-Instruction [ASPLOS'06]

This talk covers only RTR
- Regulated Transitive Reduction algorithm

Outline

Race Recording

RTR Algorithm
Compress log during recording \(\Rightarrow\) replay more "regularly"

Results with Commercial Workloads

Conclusion
Technically, what's race recording?

Race Recording

X = 6

Terminologies and Assumptions

Goal: Reproduce some conflicts with minimum log data.
Regulated Transitive Reduction (RTR)

Log All Conflicts

Dependence Log

Log J: 2→3
3→5
4→6

Log I: 2→3

Log Size: 5*16=80 bytes
(10 integers)

But too many conflicts

Netzer’s Transitive Reduction (TR)

TR Reduced Log

Log J: 2→3
3→5
4→6

Log I: 2→3

Log Size: 64 bytes
(8 integers)

How to further reduce log size?
The Intuition of the RTR Algorithm

After Reduction

“Regulate” Replay

Vectors

From I to J

Vectors

Stricter Dependences to Aid Vectorization

Log J: x=3, 5, \( \Delta = 1 \)

New Reduced Log

Log I: 2\( \rightarrow \)3

4\( \rightarrow \)5

Log I: 2\( \rightarrow \)3

Log Size: 48 bytes
(6 integers)

Fewer dependencies to log

Compress Vectorized Dependencies

Vectorized Log

Log J: x=3, 5, \( \Delta = 1 \)

Log I: x=3, \( \Delta = 1 \)

Log Size: 40 bytes
(5 integers)

TR\( \rightarrow \)RTR: fewer deps + fewer byte/dep
Deadlock Avoidance of RTR

Replay Cycle

\(c_4 \rightarrow c_3 \rightarrow c_2 \rightarrow c_1\)

Limit the strict dependencies (see paper)

Results with Commercial Workloads

Full-system Simulation Method

Commercial server hardware
- GEMS: http://www.cs.wisc.edu/gems
- Full-system (OS + application) executions
- 4-core CMP (Sequential Consistent)
  - 1-way in-order issue, 2 GHz,
  - 64KB I/D L1, 4MB L2, 64byte lines, MOSI directory

Commercial server software
- Apache - static web serving
- SpecJBB - middleware
- OLTP - TPC-C like
- Zeus - static web serving
Log Size: 1 byte/KI

Less buffer, longer recording, smaller logs

RTR vs. Netzer's TR

28% smaller log
- TR was "optimal"

Why Does RTR Work Well?

- Instructions execute at similar speed
- Dependencies are often "vectorizable"
"Less hardware" & "TSO" not covered
- Equally important
- More details in the paper

Result: One more step toward practical

Conclusion

Race recording → Counter nondeterminism

RTR → 1 byte/kilo-instruction
- Based on Netzer's transitive reduction
- Create stricter dependencies
- Vectorize dependencies to compress log
- Avoid overly-strict hence no deadlock

Future work
- Support snooping, SMT, replayer