RTR: 1 Byte/Kilo-Instruction Race Recording

Min Xu Rastislav Bodik Mark D. Hill

vmware

Why Do You Need a Recorder?

```
% gcc sim.c
% a.out
Segmentation fault
%
```

```
% gdb a.out
gdb> run
Program received SIGSEGV.
In get() at hash.c:45
45          a = bucket->d;
```

```
% gcc para-sim.c
% a.out
Segmentation fault
%
```

```
% gdb a.out
gdb> run
Program exited normally.
gdb>
```

```
% gcc para-sim.c
% a.out
Segmentation fault
Race recorded in "log"
%
```

```
% gdb a.out log
gdb> run
Program received SIGSEGV.
In get() at para-hash.c:67
67          a = bucket->d;
```

Ideally …

Low cost

Low runtime overhead

Long recording: small log

```
% gcc para-sim.c
% a.out
Segmentation fault
Race recorded in "log"
%
```

```
% gdb a.out log
gdb> run
Program received SIGSEGV.
In get() at para-hash.c:67
67          a = bucket->d;
```

Applicability:
Programs – data race
Systems – non-SC

Better and Better Recorders

3

	Low Cost	Low Overhead	Small Log Size	Applicability	
				Data Race	Non-SC
InstRply '87					
B. & G. '91					
Netzer'93					
Déjà Vu '98					
RecPlay '00					
JaRec '04					
FDR '03					
BugNet '05					
ReEnact '03					
CORD '06					
Strata '06					

Hardware Acceleration [ISCA'03]

1 Byte/Kilo-Instruction [ASPLOS'06]

This talk covers only RTR

- Regulated Transitive Reduction algorithm

Outline

5

Race Recording

⇩

RTR Algorithm

Compress log during recording → replay more "regularly"

⇩

Results with Commercial Workloads

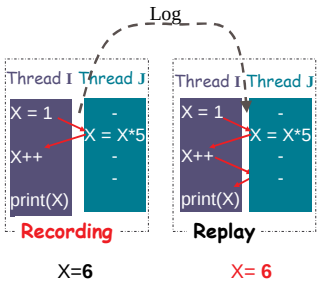
⇩

Conclusion

Technically, what's race recording?

Race Recording

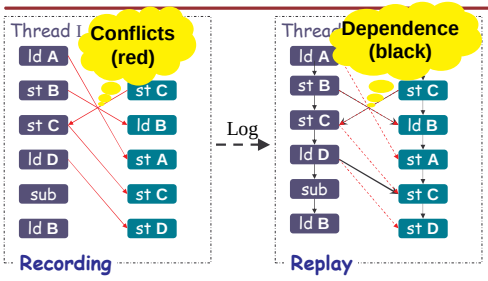
7



X=6 X= 6

Terminologies and Assumptions

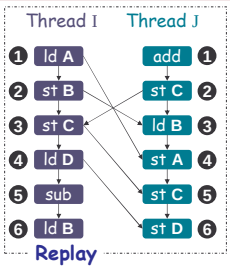
8



Goal: Reproduce same conflicts with minimum log data

Regulated Transitive Reduction (RTR)

Log All Conflicts



Dependence Log

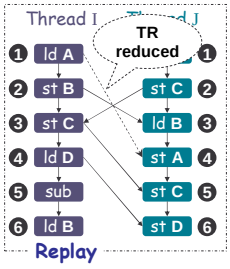
Log J: 2→3, 1→4, 3→5, 4→6 (16 bytes)

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

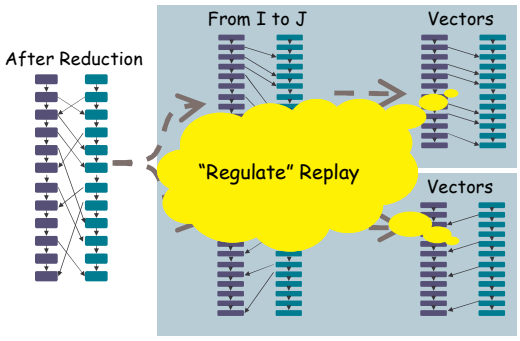
Log I: 2→3

Log Size: 64 bytes
(8 integers)

How to further reduce log size?

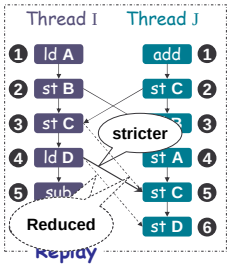
The Intuition of the RTR Algorithm

12



Stricter Dependences to Aid Vectorization

13



New Reduced Log

Log J: 2→3
4→5

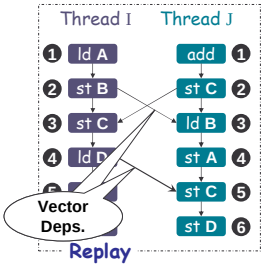
Log I: 2→3

Log Size: 48 bytes
(6 integers)

Fewer dependencies to log

Compress Vectorized Dependencies

14



Vectorized Log

Log J: x=3,5, Δ=1

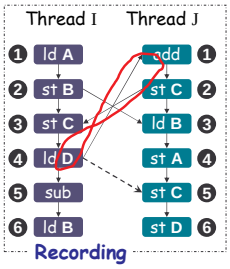
Log I: x=3, Δ=1

Log Size: 40 bytes
(5 integers)

TR→RTR: fewer deps + fewer byte/dep

Deadlock Avoidance of RTR

15



Replay Cycle

i:4→j:1 →j:2 →i:3 →i:4

Limit the strict dependencies (see paper)

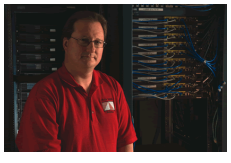
Results with Commercial Workloads

Full-system Simulation Method

17

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

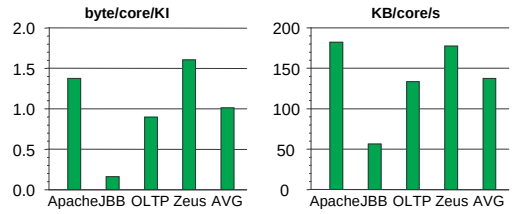


Commercial server software

- Apache – static web serving
- SpecJBB – middleware
- OLTP – TPC-C like
- Zeus – static web serving

Log Size: 1 byte/KI

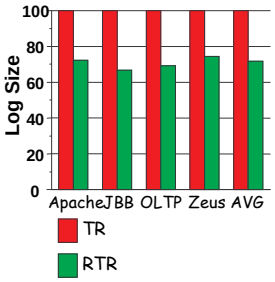
18



Less buffer, longer recording, smaller logs

RTR vs. Netzer's TR

19



28% smaller log

- TR was "optimal"

Why Does RTR Work Well?

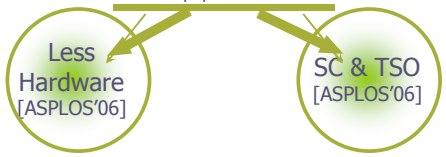
20

RTR

- 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

22

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