Online Data-Race Detection via Coherency Guarantees

Dejan Perkovic & Pete Keleher

University of Maryland

Overview

- On-the-fly data-race detection
- Lazy Release Consistent Distributed Shared Memory(LRC DSM)
- Low overhead
- Detected data-races in Water and TSP

Motivation

Parallel programming is hard due to:

- Explicit synchronization
- Nondeterminism => hard debugging

Data-Race detection NP-hard in general

What is a Data-Race?

- Unsynchronized accesses to the same shared memory location
- At least one is write

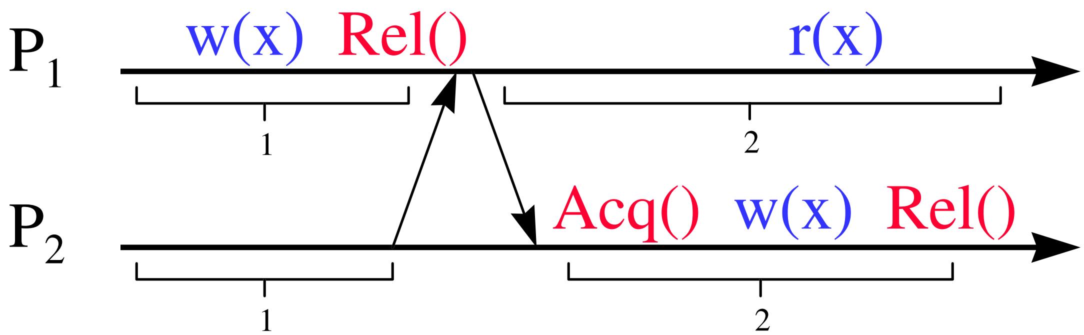
P_1	P_2
Lock(L)	
x+=1	
Unlock(L)	
	Lock(L)
	x+=1
	Unlock(L)
Result: x+=2	

P_1	P_2
Lock(L)	
x+=1	x+=1
Unlock(L)	
Result: x+=1 or x+=2	

Proposed Method

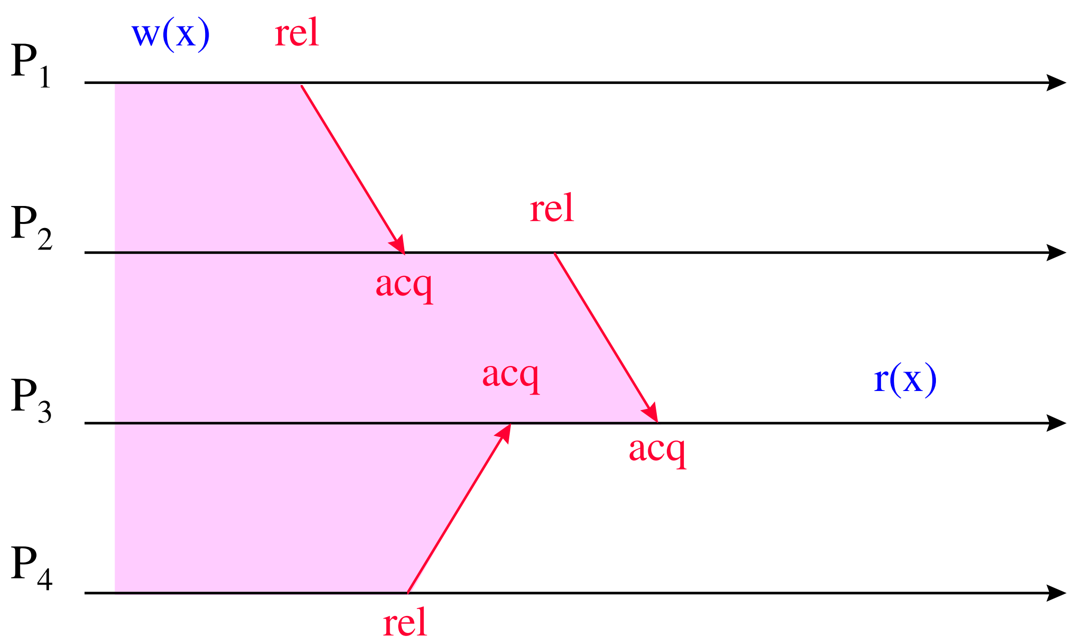
- On-the-fly data-race checks
- Consistency information from CVM
- Instrument shared references via ATOM
- *Read* page notices along with *write* page notices(invalidations)

Data-Races in LRC DSM

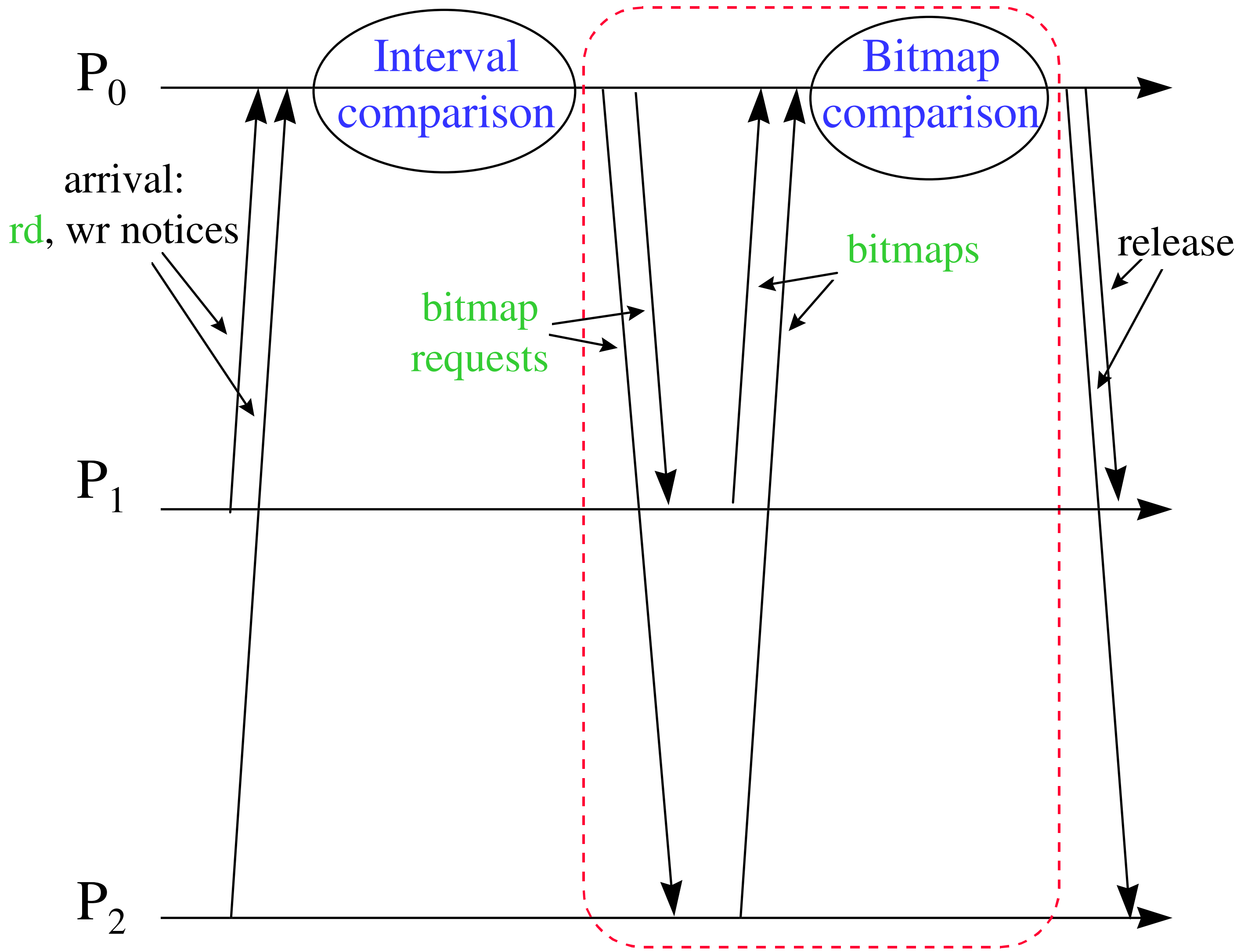


- P_1's write and P_2's write: no data-race
- P_1's read and P_2's write: data-race

Lazy Release Consistency

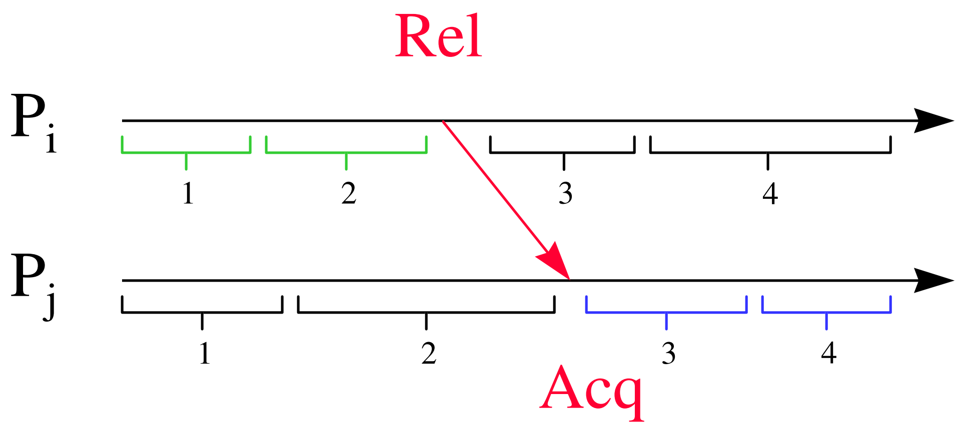


Barrier



False sharing or Data Races

Interval Concurrency Check



Many intervals

=> Many synchronizations

=> Many ordering

Test Environment

- DEC Alpha workstations
- ATM network connection
- CVM: LRC single writer protocol

Applications Tested

- Four common applications
- Standard problem sizes

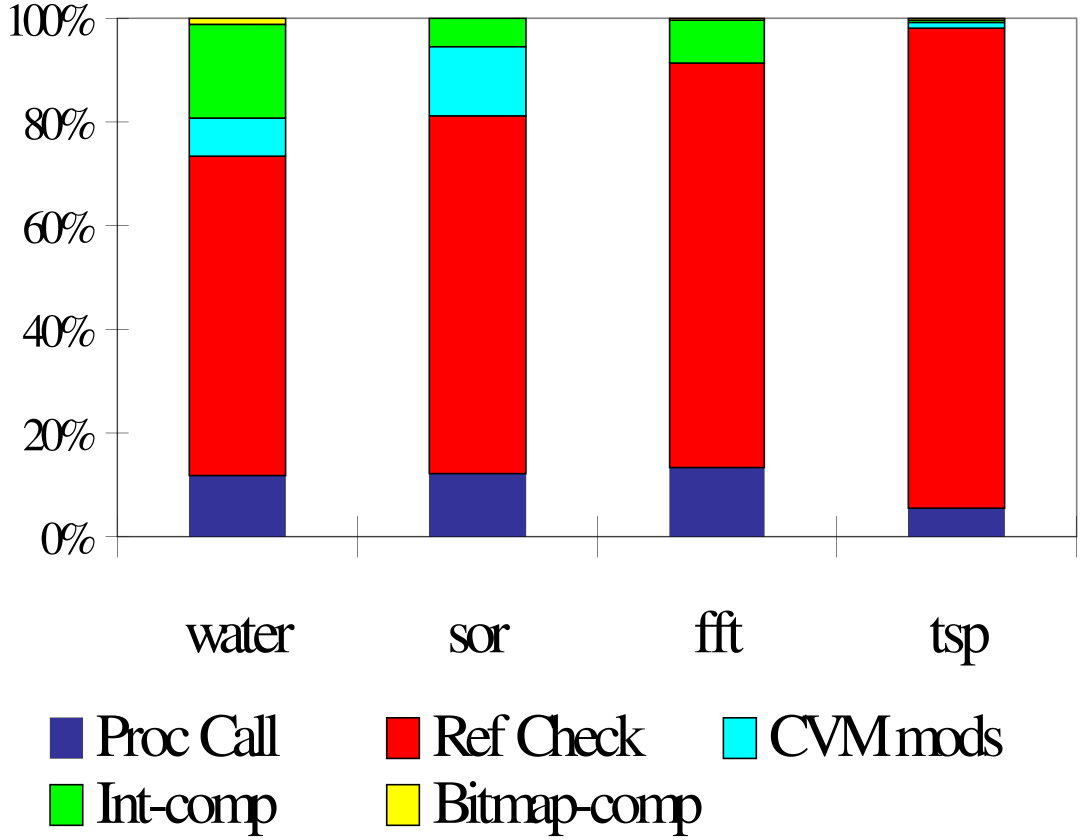
	Input set	Sync used	Slowdown (n=8)
FFT	64x64x64	barrier	2.61
SOR	1024x1024	barrier	4.25
TSP	19 cities	lock	2.51
Water	512 mols	lock, barrier	2.84

Reference Instrumentation

- Gather all shared addresses referenced during each interval
- Instrument only potentially shared references

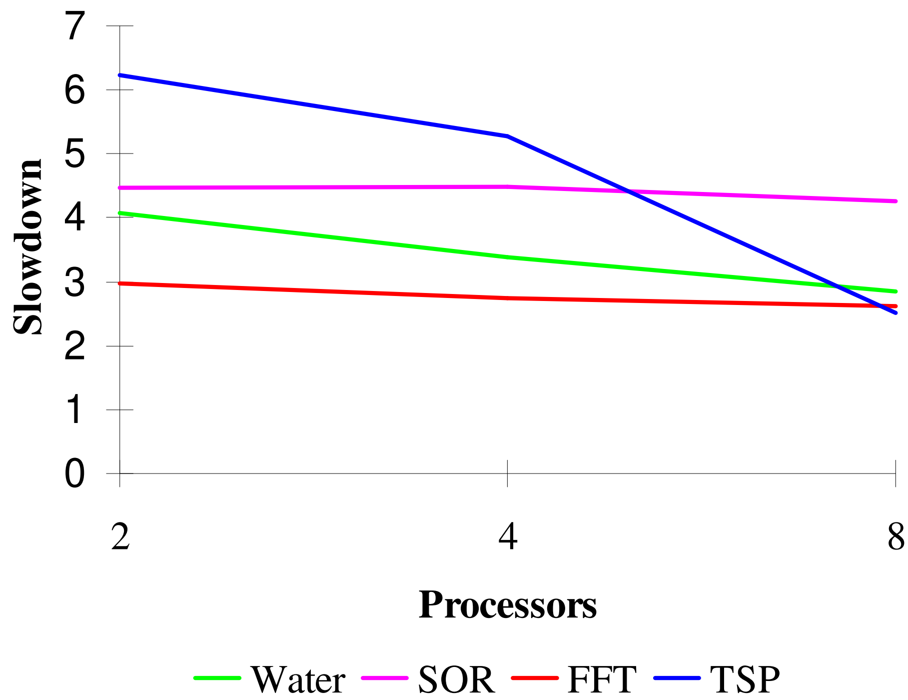
	Instrumented (static)	Shared (Dynamic)
Water	17%	12%
SOR	7%	63%
FFT	8%	24%
TSP	20%	25%

Overhead



100%
80%
60%
40%
20%
0%
water
sor
fft
tsp
Proc Call
Ref Check
CVM mods
Int-comp
Bitmap-comp

Scalability



Accuracy

- Actual vs. feasible data-races
- Detecting all feasible data-races is NP hard

P_1	P_2
write x	
Lock(L)	
…	
Unlock(L)	
	Lock(L)
	write x
	Unlock(L)

Instruction Identification

- Basic version: only the addresses involved in the data-race
- Address combined with symbol table => variable involved
- Brute force: saving PCs with every shared reference
- Enforcing the same synchronization ordering

Improvements

- Data-flow analysis to instrument fewer references
- On-the-fly code modification
- Inlining of reference instrumentation

Conclusion

- Implemented on-the-fly
  actual data-race detection
- Leverage existing DSM information
- Good performance
- Detected data-races in Water and TSP