Concurrency bugs

and tools to find them

CSE 333
James Wilcox
Hi, I’m James

PL/Systems

Ask questions!
Eraser: A Dynamic Data Race Detector for Multithreaded Programs

STEFAN SAVAGE
University of Washington

MICHAEL BURROWS, GREG NELSON, and PATRICK SOBALVARRO
Digital Equipment Corporation

and

THOMAS ANDERSON
University of California at Berkeley

Multithreaded programming is difficult and error prone. It is easy to make a mistake in synchronization that produces a data race, yet it can be extremely hard to locate this mistake in execution. Eraser is a dynamic data race detector for programs written in C or C++.
Eraser: A Dynamic Data Race Detector for Multithreaded Programs

STEFAN SAVAGE
University of Washington
MICHAEL BURROWS, GREG NELSON, and PATRICK SOBALVARRO
Digital Equipment Corporation
and
THOMAS ANDERSON
University of California at Berkeley

Multithreaded programming is difficult and error prone. It is easy to make a mistake in synchronization that produces a data race, yet it can be extremely hard to locate this mistake in typical debugging sessions. With Eraser, we have incorporated techniques for fast, precise and automatic data race detection.
How multicore programs actually run

Thread 1

tmp1 = bal
bal = tmp1 + 10

Thread 2

tmp2 = bal
bal = tmp2 + 10
How multicore programs actually run

Thread 1
	tmp1 = bal
	bal = tmp1 + 10
What we probably meant

**Thread 1**

tmp1 = bal
bal = tmp1 + 10

**Thread 2**

tmp2 = bal
bal = tmp2 + 10
Interleaving model

The execution behaves as if steps of each thread were interleaved.
Reasoning in Interleaving model

Thread 1

tmp1 = bal
bal = tmp1 + 10

Thread 2

tmp2 = bal
bal = tmp2 + 10
Reasoning in Interleaving model

Thread 1

tmp1 = bal

bal = tmp1 + 10

Thread 2

tmp2 = bal

bal = tmp2 + 10
If the program is data race free, then:

The execution behaves as if steps of each thread were interleaved.
Eraser: A Dynamic Data Race Detector for Multithreaded Programs

STEFAN SAVAGE
University of Washington

MICHAEL BURROWS, GREG NELSON, and PATRICK SOBALVARRO
Digital Equipment Corporation

and

THOMAS ANDERSON
University of California at Berkeley

Multithreaded programming is difficult and error prone. It is easy to make a mistake in synchronization that produces a data race, yet it can be extremely hard to locate this mistake during debugging. While existing tools can detect data races, they are not as effective as they could be.
Data races

Two threads access:
the same location
at the same time
at least one of them writes
Happens Before

Thread 1

tmp1 = bal
bal = tmp1 + 10

Thread 2

tmp2 = bal
bal = tmp2 + 10
Thread 1
lock m
tmp1 = bal
bal = tmp1 + 10
unlock m

Thread 2
lock m
tmp2 = bal
bal = tmp2 + 10
unlock m
Thread 1

lock m

tmp1 = bal

bal = tmp1 + 10

unlock m

Thread 2

lock m

tmp2 = bal

bal = tmp2 + 10

unlock m
Eraser: A Dynamic Data Race Detector for Multithreaded Programs

STEFAN SAVAGE
University of Washington
MICHAEL BURROWS, GREG NELSON, and PATRICK SOBALVARRO
Digital Equipment Corporation
and
THOMAS ANDERSON
University of California at Berkeley

Multithreaded programming is difficult and error prone. It is easy to make a mistake in synchronization that produces a data race, yet it can be extremely hard to locate this mistake in the program that produces the race. Eraser takes an approach to help locate such bugs.
How to find races

Track every memory location

Track happens before

Check every access is ordered
How to find races in practice (Eraser)

Enforce locking discipline

Can be implemented more efficiently
How to find races in practice (Eraser)

Enforce locking discipline

Can be implemented more efficiently

Reports races when no guarding lock reflects engineering practice

False positives: other sync, “benign” races
Safe languages

segfault  ->  ArrayOutOfBoundsExceptions

segfault  ->  NullPointerException
Safe concurrent languages

segfault  ->  ArrayOutOfBoundsException

segfault  ->  NullPointerException

data race  ->  DataRaceException
Thread 1
lock m
   tmp1 = bal
unlock m
lock m
   bal = tmp1 + 10
unlock m

Thread 2
lock m
   tmp2 = bal
unlock m
lock m
   bal = tmp2 + 10
unlock m
Other ways of finding races

Dynamic
  Efficient HB detectors

Static
  Static lockset
  HB
  Symbolic execution
  Verification
Weak memory models

Ensuring DRF may be prohibitively expensive

Interact directly with hardware memory model

Exposed through, e.g., volatile

Lock-free data structures/algorithms