Concurrency bugs

and tools to find them

CSE 333

James Wilcox

Hi, I'm James

PL/Systems

Ask questions!

Hi ... James

...tions!

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

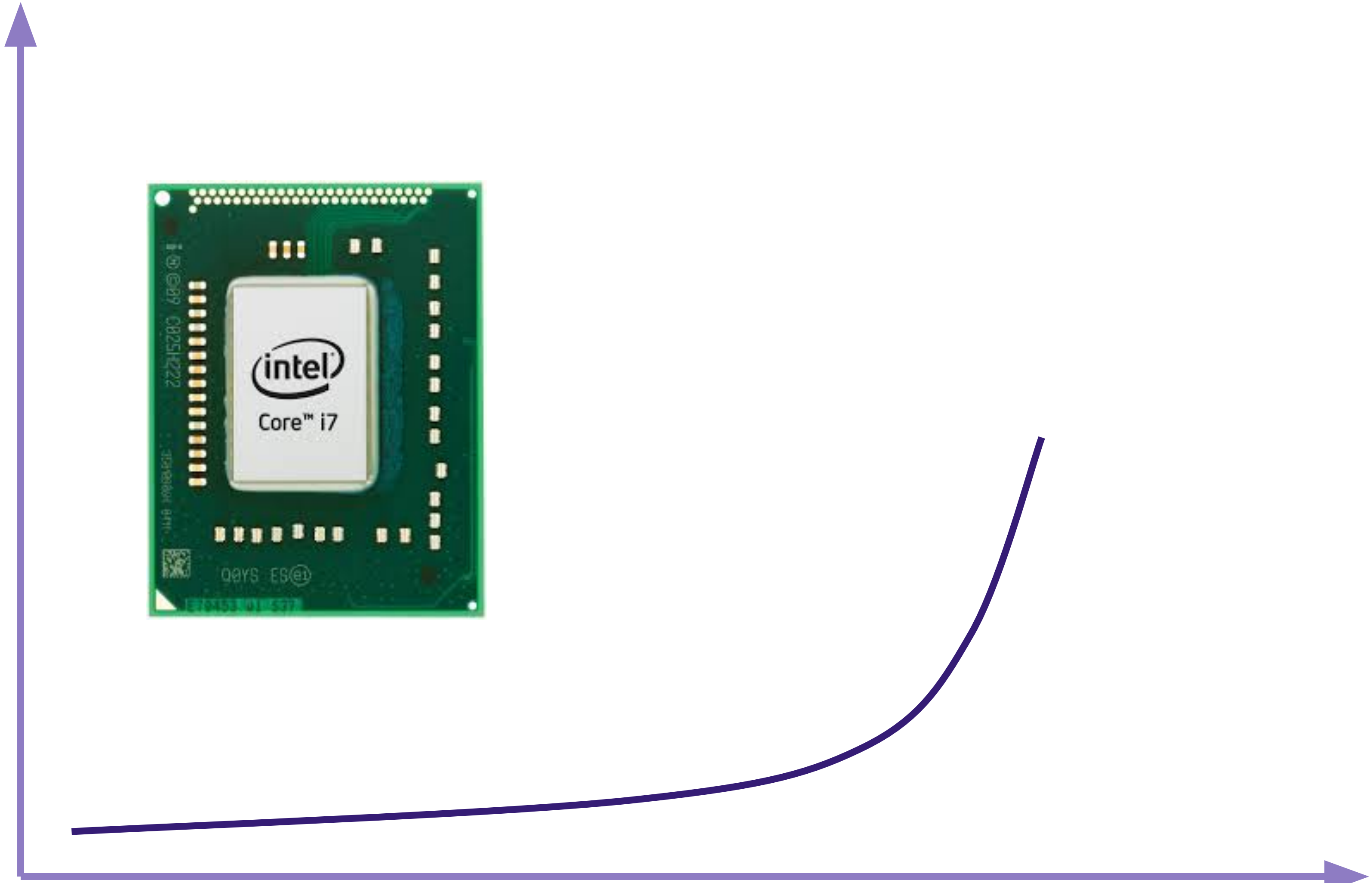
Multithreaded programming is difficult and error prone. It is easy to make a mistake in synchronization that produces a data race, yet it can be extremely hard to locate this mistake

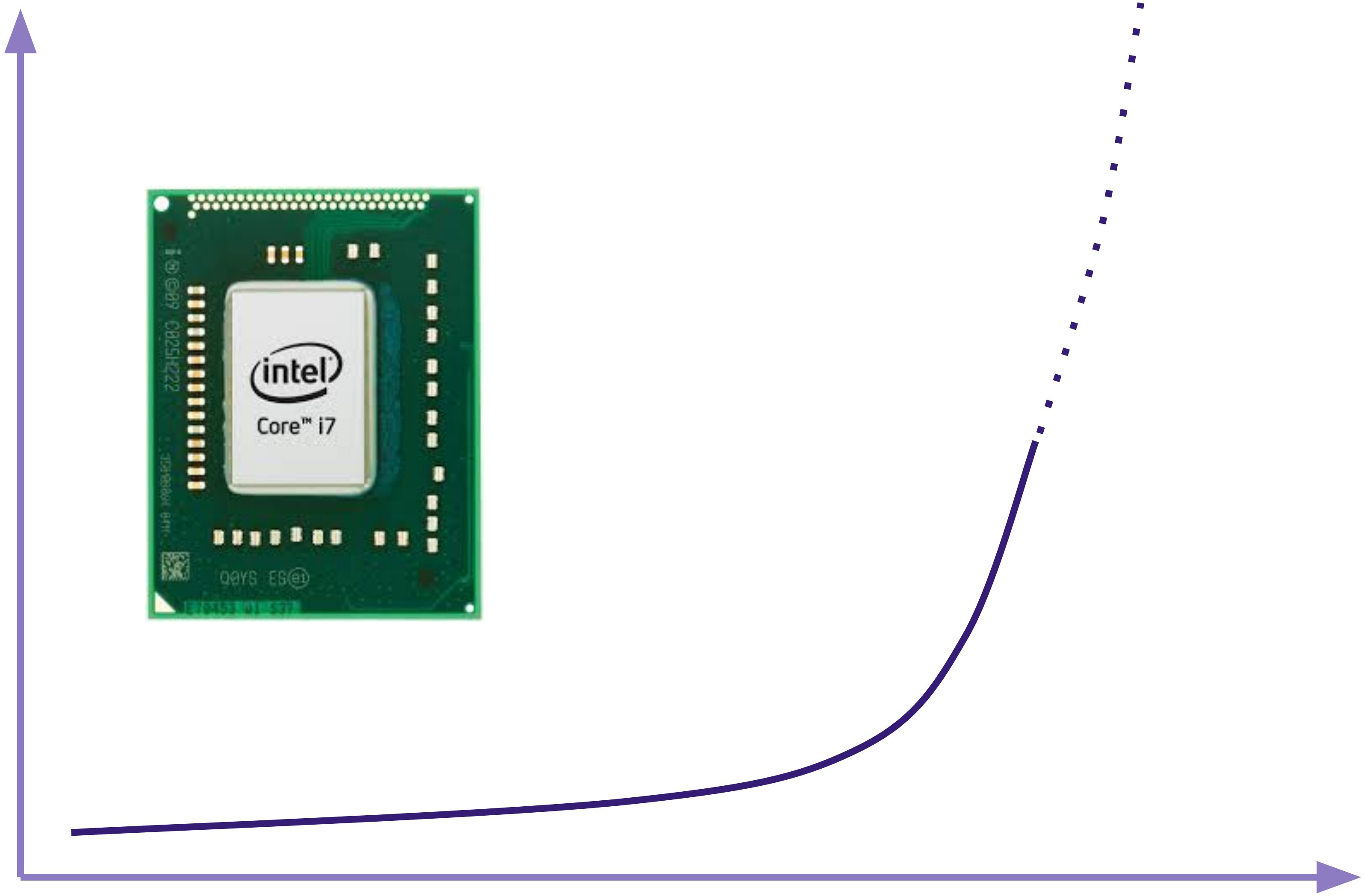
Demo

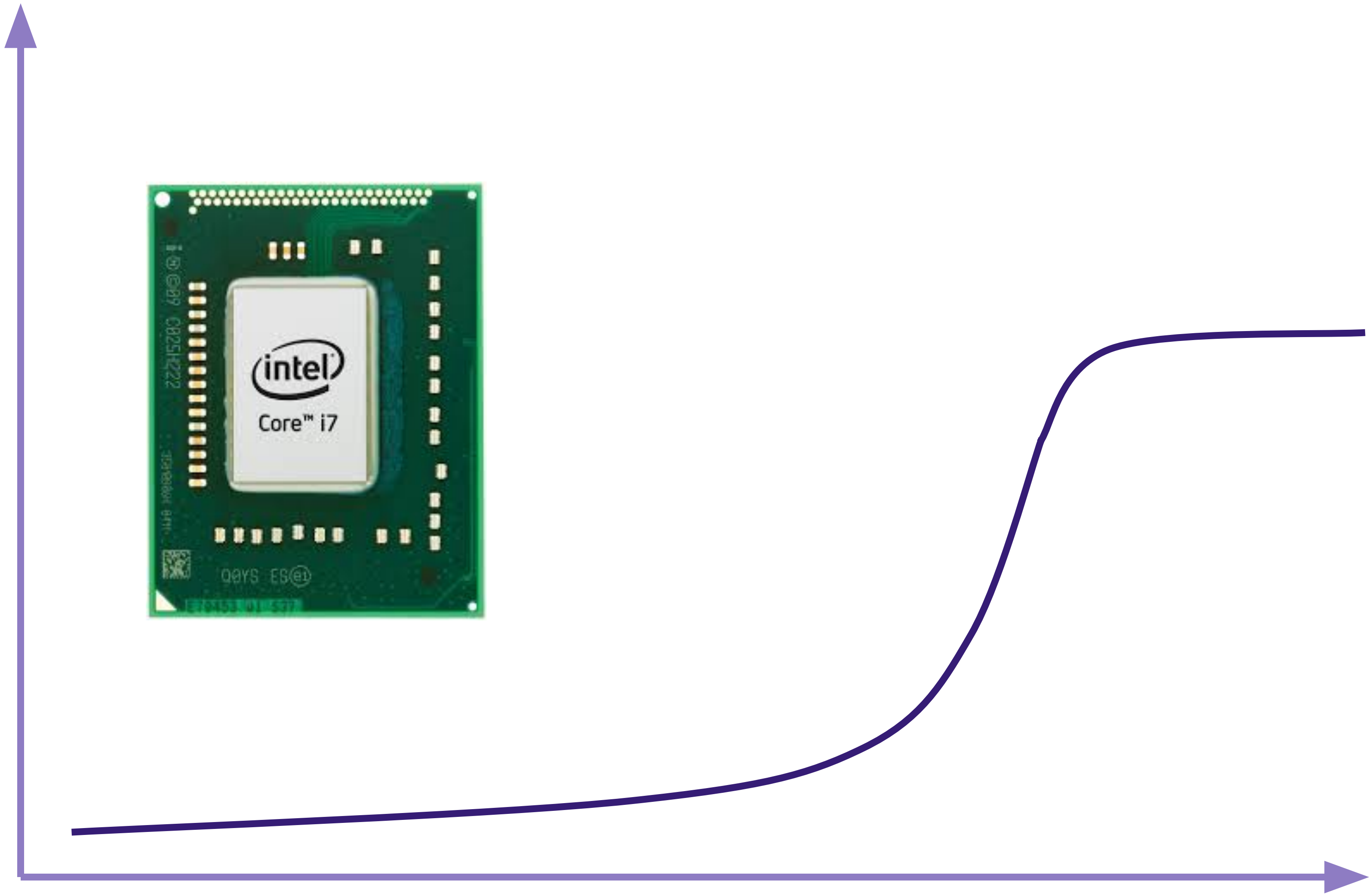
Eraser: A Dynamic Data Race Detector for Multithreaded Programs

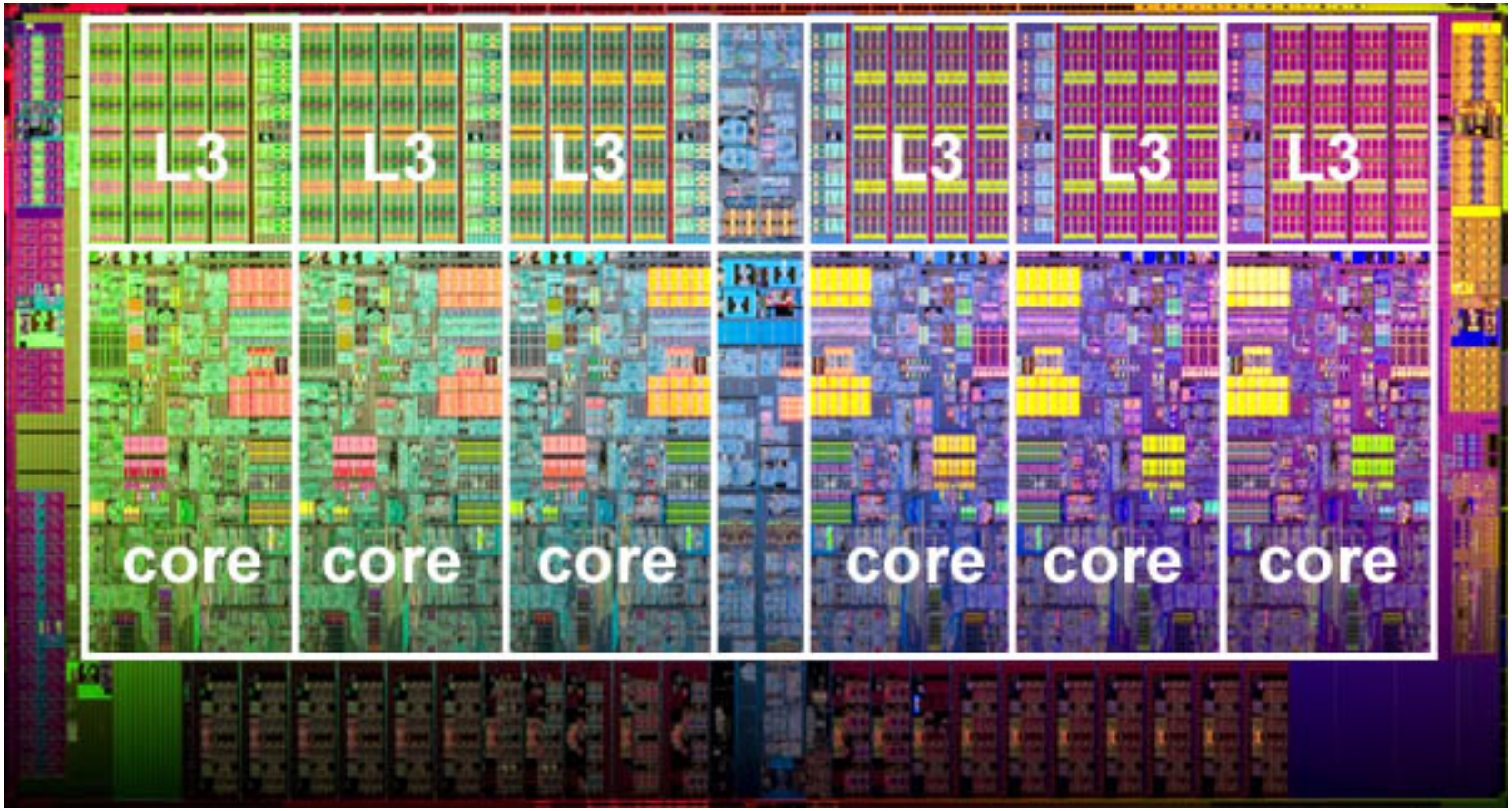
STEFAN SAVAGE
University of Washington
MICHAEL BURROWS, GREG NELSON, and PATRICK SOBALVARRO
Digital Equipment Corporation
and
THOMAS ANDERSON
University of California at Berkeley

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


intel
Core™ i7


intel
Core™ i7


intel
Core™ i7


L3
L3
L3
L3
L3
L3
core
core
core
core
core
core

How multicore programs actually run

Thread 1

```
tmp1 = bal
bal = tmp1 + 10
```

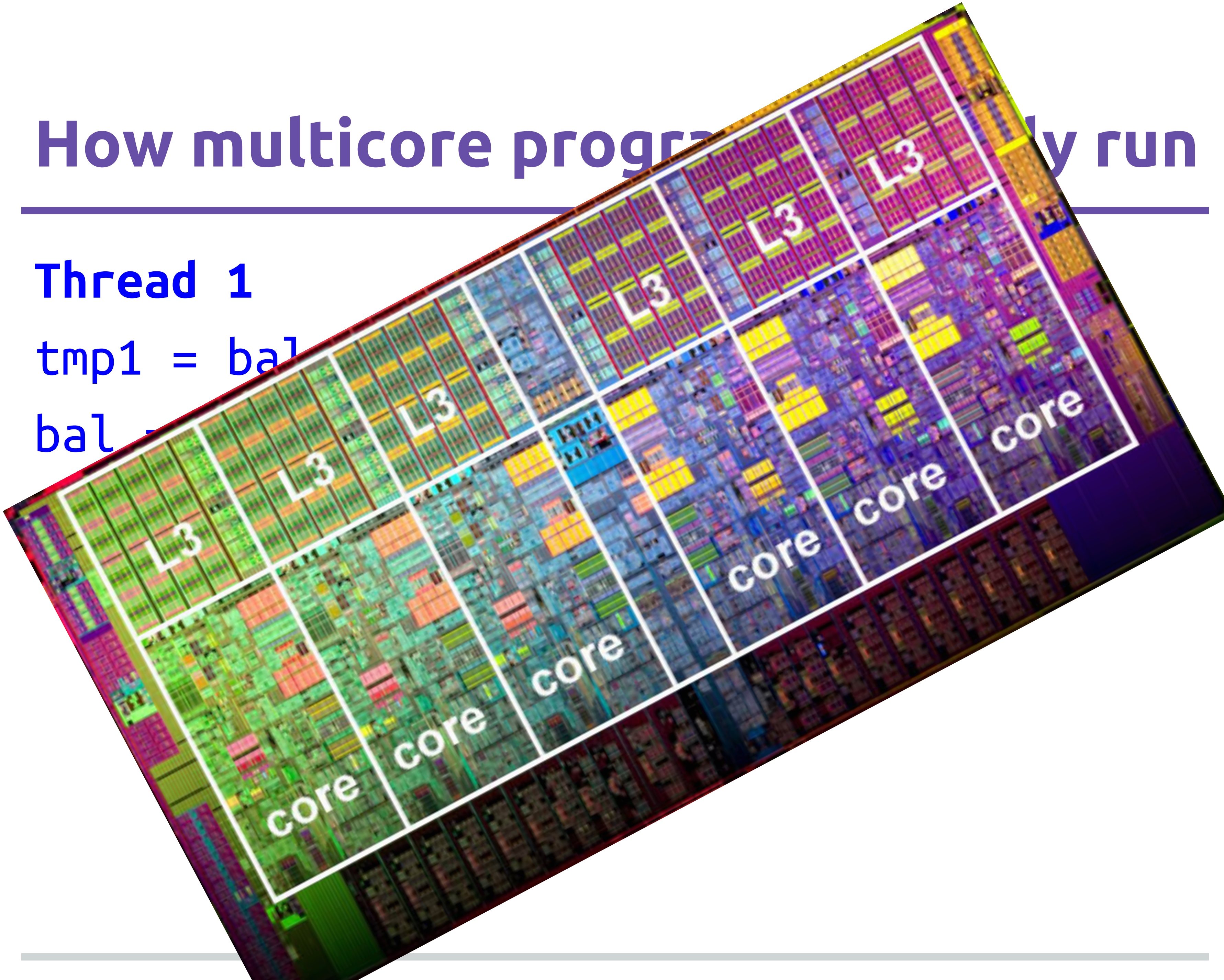
Thread 2

```
tmp2 = bal
bal = tmp2 + 10
```

How multicore progra... ...ly run

Thread 1

```
tmp1 = bal...
bal ...
```



What we probably meant

Thread 1	**Thread 2**
`tmp1 = bal`	
`bal = tmp1 + 10`	
	`tmp2 = bal`
	`bal = tmp2 + 10`

Interleaving model

*The execution behaves as if
steps of each thread were interleaved.*

Reasoning in Interleaving model

Thread 1	**Thread 2**
`tmp1 = bal`	
`bal = tmp1 + 10`	
	`tmp2 = bal`
	`bal = tmp2 + 10`

Reasoning in Interleaving model

Thread 1	**Thread 2**
`tmp1 = bal`	
	`tmp2 = bal`
`bal = tmp1 + 10`	
	`bal = tmp2 + 10`

Reaso… …odel

“Undefined behavior!!!”

Thread 1

```
tmp1 = bal
                    tmp2
bal = tmp1 + 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

Data races

Two threads access:

- the same location
- at the same time
- at least one of them writes

Happens Before

Lamport 1978. “Time, Clocks, and the Ordering of Events in a Distributed System”

Thread 1

```
tmp1 = bal
bal = tmp1 + 10
```

Thread 2

```
tmp2 = bal
bal = tmp2 + 10
```

Thread 1	**Thread 2**
lock m	lock m
tmp1 = bal	tmp2 = bal
bal = tmp1 + 10	bal = tmp2 + 10
unlock m	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

How to find races

Track every memory location

Track happens before

Check every access is ordered

How to find r

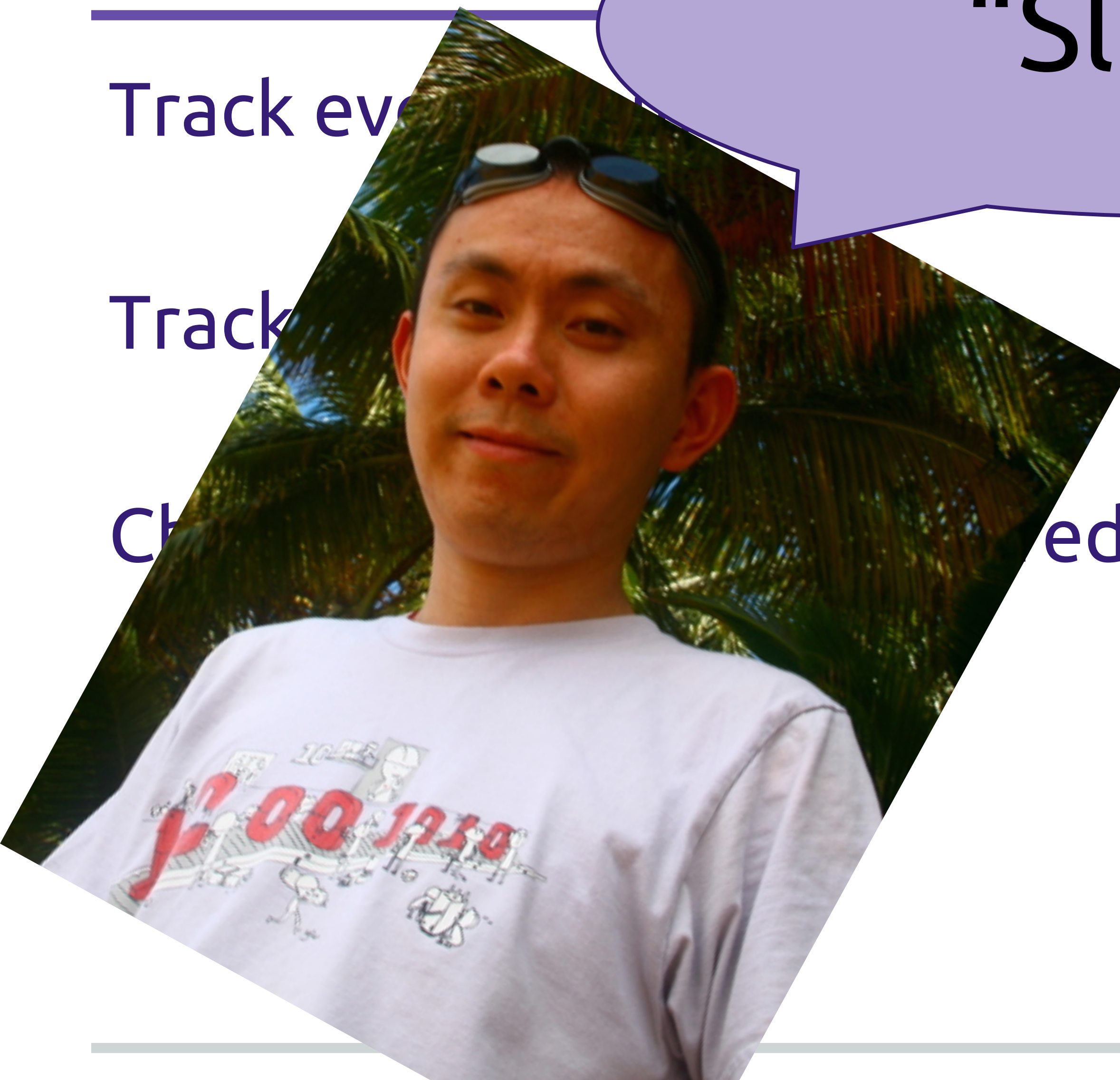
“Slow!”

Track ev

Track

Ch

ed



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 -> ArrayOutOfBoundsExceptions

segfault -> NullPointerException

data race -> DataRaceException

FTFY

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
```

FTFY

“Still wrong!”

```
Thread 1
lock m
  tm
un
unlo
```

```
mp2 = bal
m
tmp2 + 10
k m
```


DRF ⇒ SC

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, eg, `volatile`

Lock-free data structures/algorithms