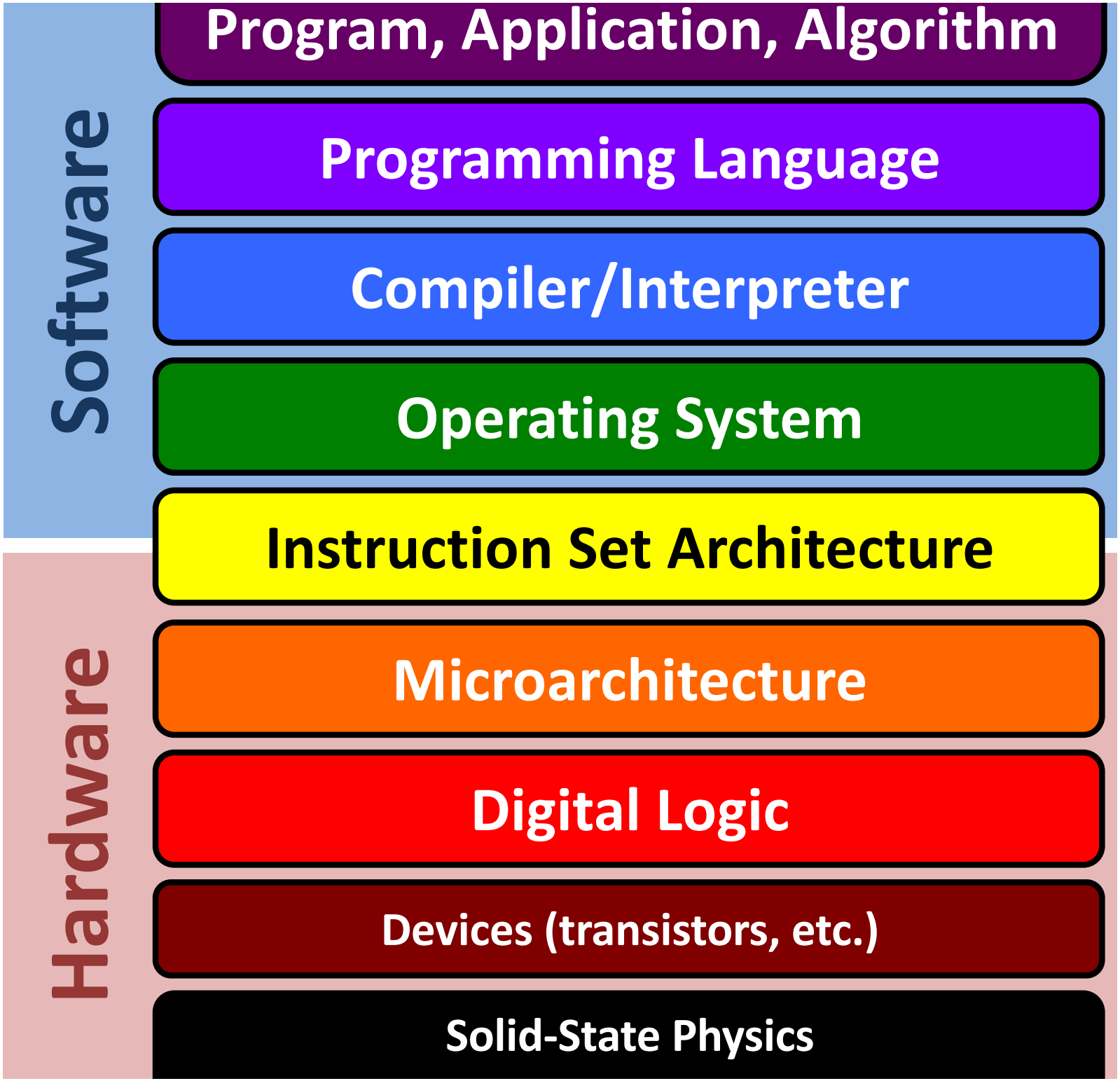
CS 240 in context

1

How Computers Work



2 Big Ideas in CS, Systems, and beyond

Abstraction

Do not start every project with transistors.
Abstraction is beautiful and empowering,
but real abstractions have leaks and wrinkles.

Translation

Between layers of abstraction.
Structured computation.

Representation

No representation without taxation.
Representations have costs.

Performance

Memory: clever, imperfect abstraction.
Tiny code changes, huge impact.

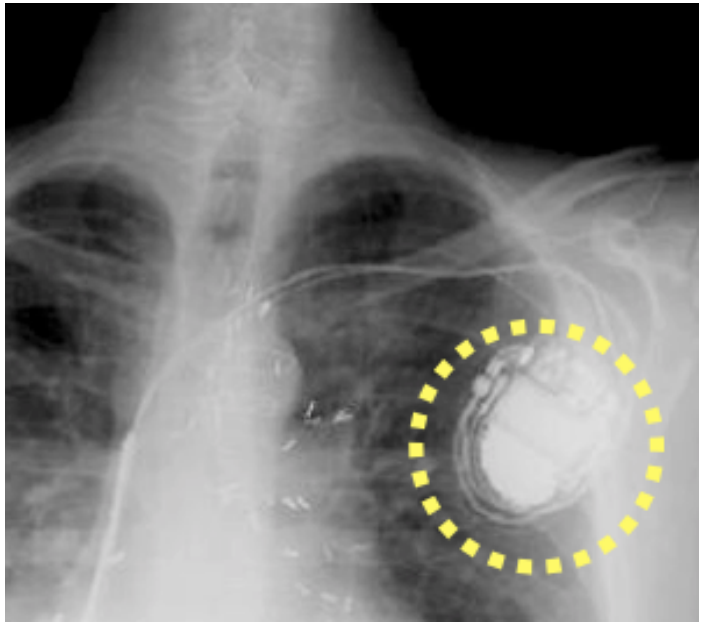
Security + Reliability

Trickiest exploits & errors
involve multiple layers, even hardware!

These things matter more every day.

How to Detect Exploits of the GHOST Buffer Overflow Vulnerability

Wednesday, February 11, 2015 Swati Khandelwal



The *GHOST vulnerability* is a buffer overflow condition that can be easily exploited loc remotely, which makes it extremely dangerous. This vulnerability is named after the *GetHOST* function involved in the exploit.

The New York Times

Business

A Heart Device Is Found Vulnerable to Hacker Atta

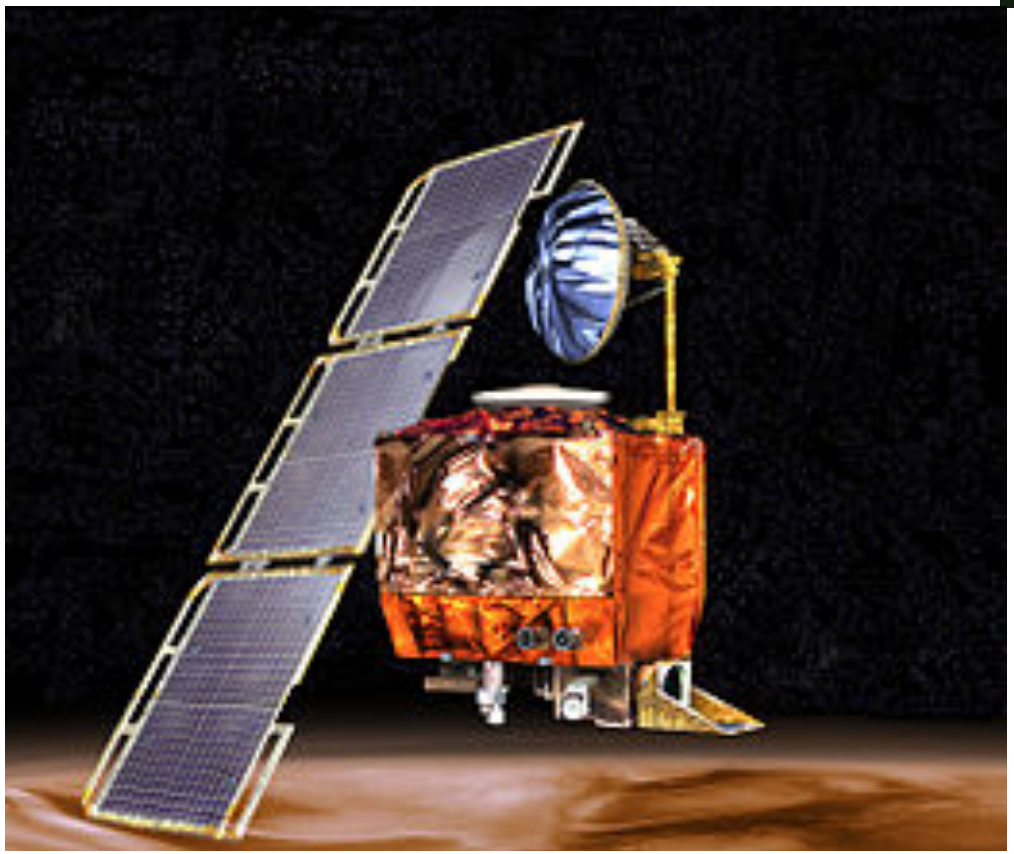
By BARNABY J. FEDER
Published: March 12, 2008

To the long list of objects vulnerable to attack by computer hackers, add the human heart.

The threat seems largely theoretical. But a team of computer security researchers plans to report Wednesday that it had been able to gain wireless access to a combination heart defibrillator and pacemaker.

Ariane 5 Rocket, 1996

Exploded due to cast of 64-bit floating-point number to 16-bit signed number. **Overflow.**



1998
Mars Climate Orbiter

Disintegrated due to mismatched units in Lockheed-Martin / NASA software components.

Toyota "Unintended Acceleration Events"

Oklahoma jury:

"Spaghetti Code" = "reckless disregard"

>10,000 global variables

81,514 violations of MISRA-C coding rules

Expect 3 minor bugs + 1 major bug per 30 violations

Task/process monitoring failed to monitor tasks/processes

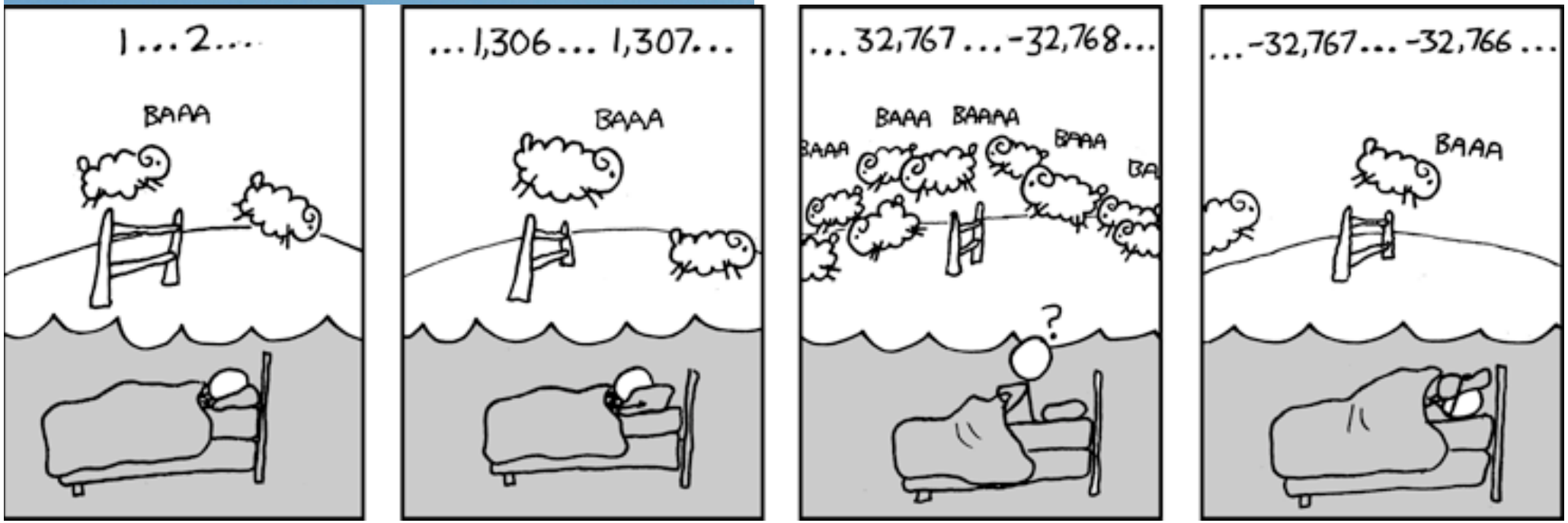
Memory corruption

(Wait, it was written in C?!?!?!)

http://www.safetyresearch.net/blog/articles/toyota-unintended-acceleration-and-big-bowl-%E2%80%9Cspaghetti%E2%80%9D-code

https://xkcd.com/571/

"... a **Model 787 airplane** that has been powered continuously for 248 days can lose all alternating current (AC) electrical power due to the generator control units (GCUs) simultaneously going into failsafe mode ... This condition is caused by a **software counter** internal to the GCUs that will **overflow** after **248 days** of continuous power. We are issuing this AD to prevent loss of all AC electrical power, which could result in **loss of control of the airplane**." --FAA, April 2015



How could we improve computer systems?

Security

Efficiency

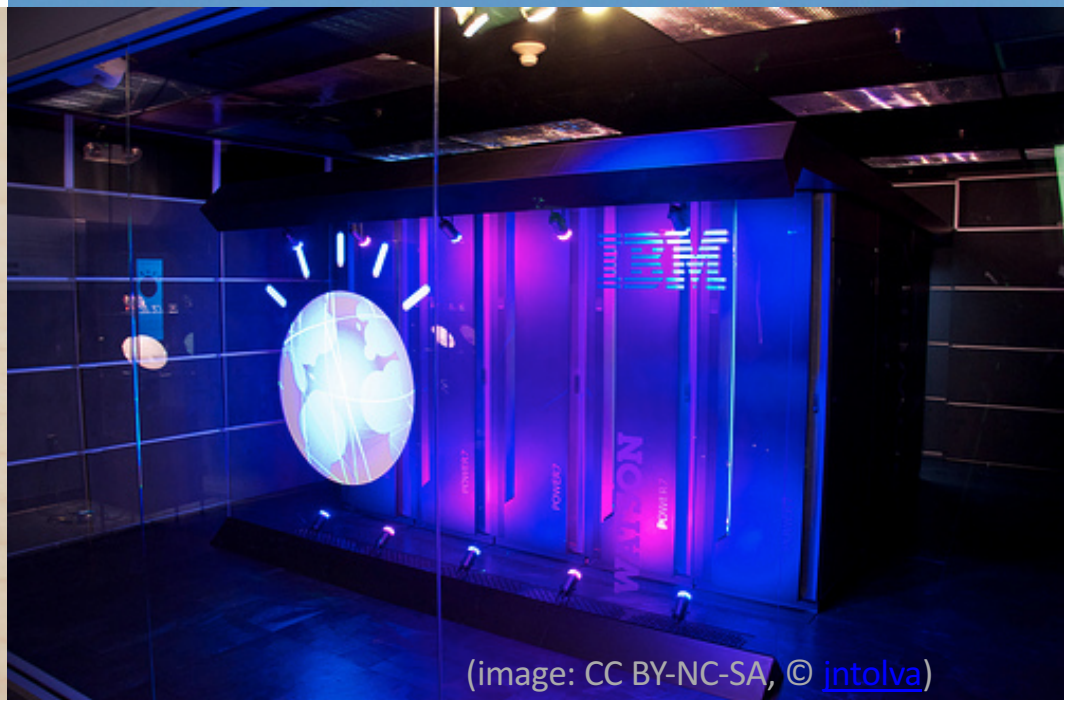
- Speed
- Space
- Programmer
- Cost, availability
  - What a simple phone can do for people: https://opendatakit.org/about/deployments/
- Energy, materials
  - A few of the impacts we usually don't see:
    http://www.nytimes.com/2015/06/07/magazine/making-and-unmaking-the-digital-world.html?_r=0

Reliability

…

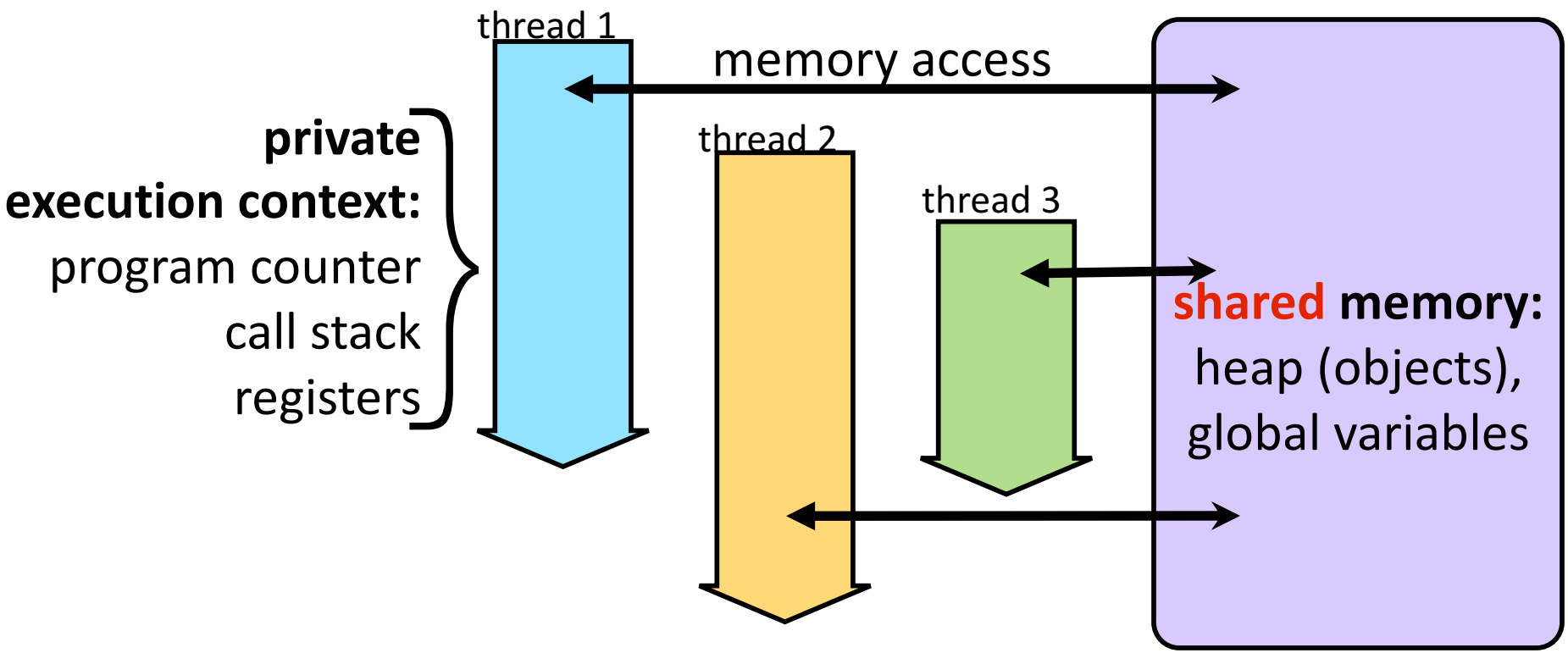
(image: CC BY-SA, © Kentaro IEMOTO@Tokyo)

(image: CC BY-SA, © William Hook)



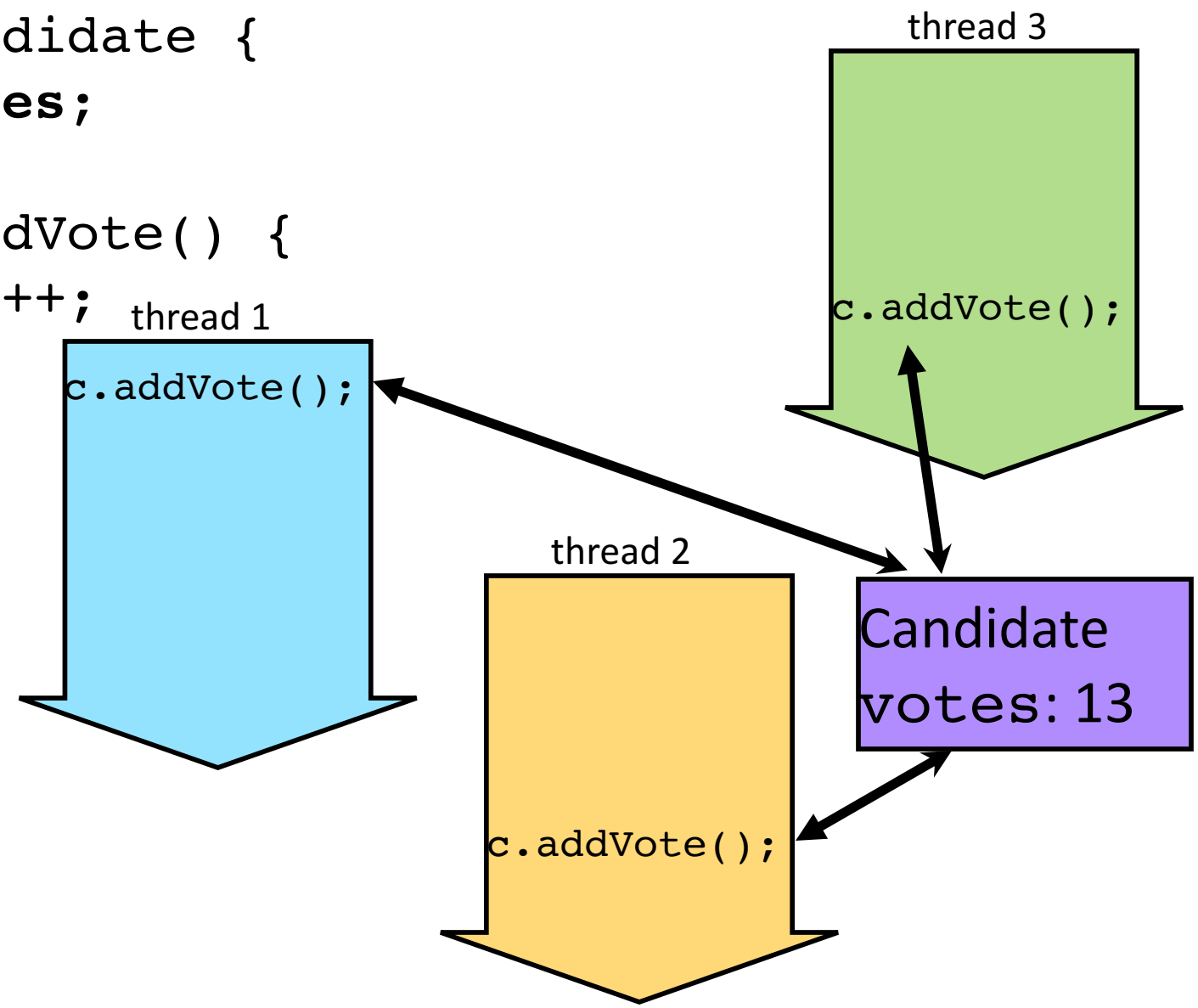
(image: CC BY-NC-SA, © jntolva)

shared-memory multithreading



multithreaded voting service

```
class Candidate {
  int votes;
  ...
  void addVote() {
    votes++;
  }
}
```



concurrent accesses

thread 1	thread 2	votes
		0
`votes++;`		?
	`votes++;`	?

concurrent accesses

thread 1	thread 2	votes
		0
`int v1 = votes;`		0
`votes = v1 + 1;`		1
	`int v2 = votes;`	1
	`votes = v2 + 1;`	2

concurrent accesses

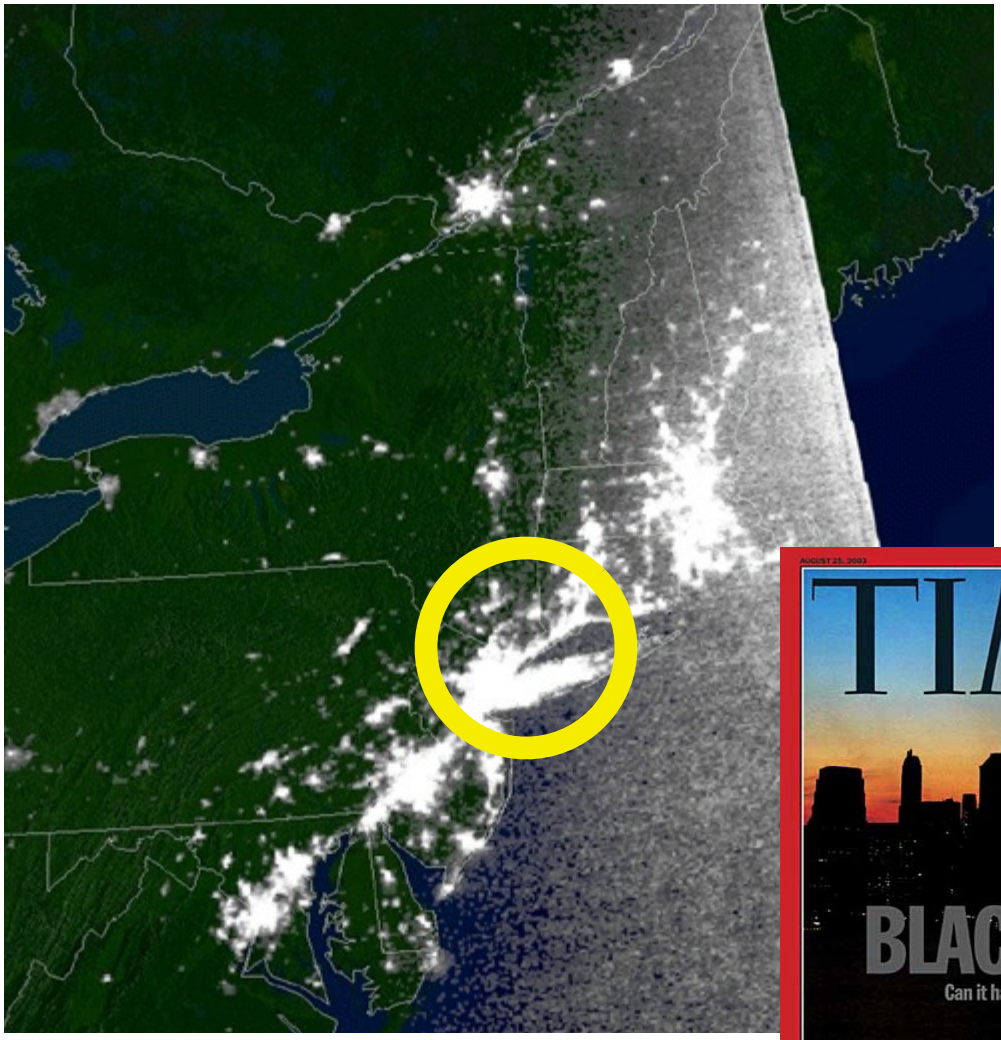
Problem 1: each thread's increment should happen "as one."

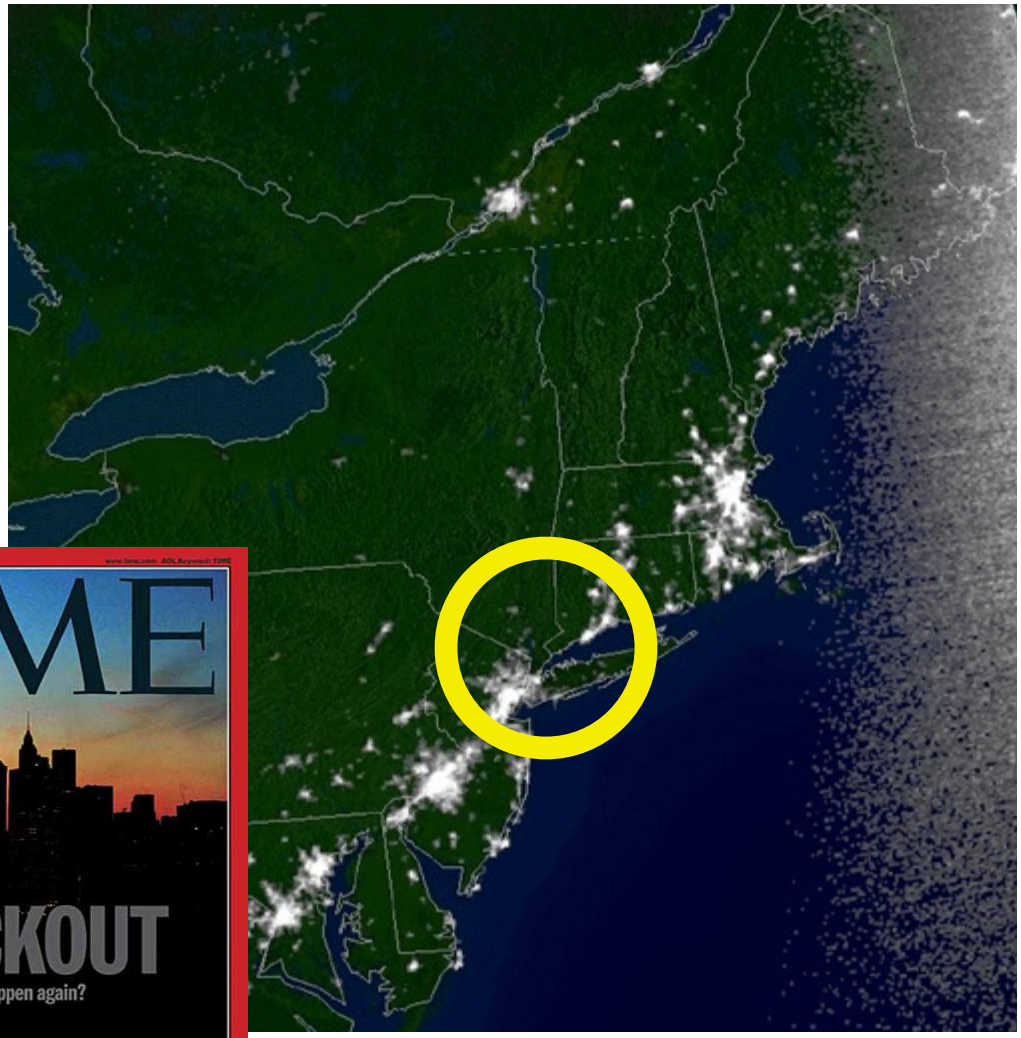
thread 1	thread 2	votes	
		0	
	`int v2 = votes;`	0	
`int v1 = votes;`		0	
`votes = v1 + 1;`		1	
	`votes = v2 + 1;`	1	✗

Problem 2: two threads accessed `votes` "at the same time."
(data race)

Northeast Blackout, 2003

caused in part by a software concurrency error





(images: public domain)

normal

blackout

despite “in excess of 3 million online operational hours” -Mike Unum, GE Energy

data race

Two memory accesses:

1. to the same memory location
2. by different threads
3. at least one access is a write
4. the accesses are not *ordered by synchronization*

thread 1

```
int v1 = votes;
votes = v1 + 1;
```

thread 2

```
int v2 = votes;
votes = v2 + 1;
```

synchronization with locks

Synchronization orders events in separate threads to control access to shared data.

```
class Candidate {
  int votes;
  Lock l = new Lock();

  void addVote() {
    l.lock();
    votes++;
    l.unlock();
  }
}
```

Zero or one threads can *hold* a **mutual exclusion lock** at a time.

only one thread at a time

synchronization with locks

thread 1

```
l.lock();
int v1 = votes;
votes = v1 + 1;
l.unlock();
```

thread 2

```
l.lock();
int v2 = votes;
votes = v2 + 1;
l.unlock();
```

Is there a data race?

votes	l
0	-
0	T1
0	T1
1	T1
1	-
1	T2
1	T2
2	T2
2	-

data race

two memory accesses:

- to the same memory location
- by different threads
- at least one write
- not *ordered by synchronization*.

synchronization with locks

thread 1

```
l.lock();
int v1 = votes;
votes = v1 + 1;
l.unlock();
```

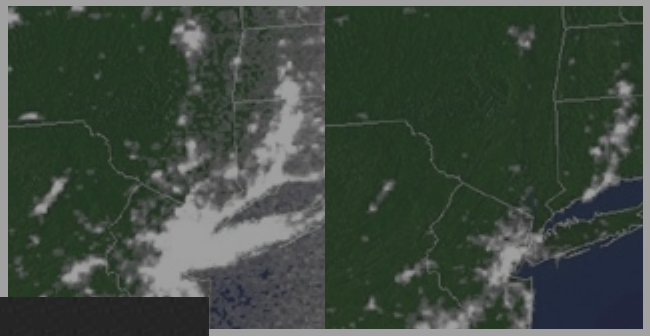
thread 2

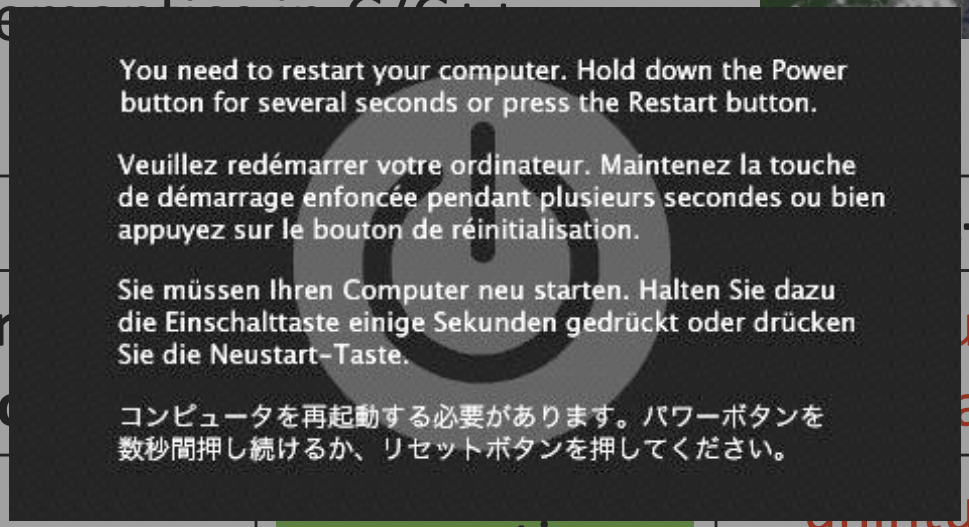
```
l.lock();
int v2 = votes;
votes = v2 + 1;
l.unlock();
```

votes	l
0	-
0	T2
0	T2
1	T2
1	-
1	T1
1	T1
2	T1
2	-

Data races are errors!

- unpredictable outcome
- unintuitive semantics in Java
- undefined se





		++
array in of bo		uitive afe
data race	exception	unintuitive unsafe

Data races should be exceptions.

[Elmas et al., PLDI 2007; Marino et al., PLDI 2010;
Lucia et al., ISCA 2010; Adve and Boehm, CACM Aug. 2010; ...]

data race exceptions

thread 1

```
int v1 = votes;
votes = v1 + 1;
```

thread 2

```
int v2 = votes;
```

data race

exception

implementing exceptions

null pointer deference

```
if (c == null) {
  throw new NullPointerException();
}
c.addVote();
```

array index out of bounds

```
if (i < 0 || array.length <= i) {
  throw new
  ArrayIndexOutOfBoundsException(i);
}
array[i] = 13;
```

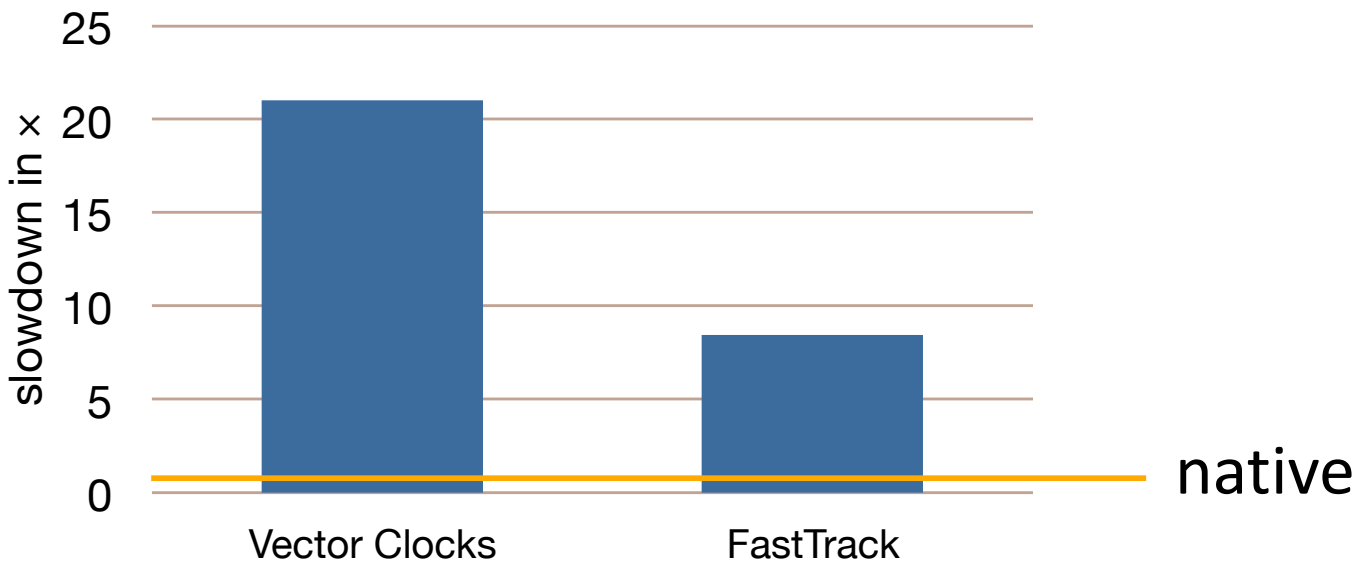
data race

```
????????
votes = 13;
```

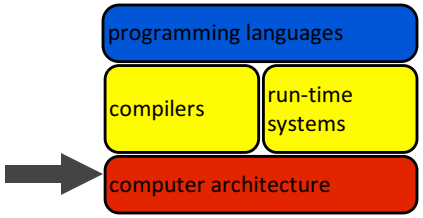
Software data race exceptions are slow.

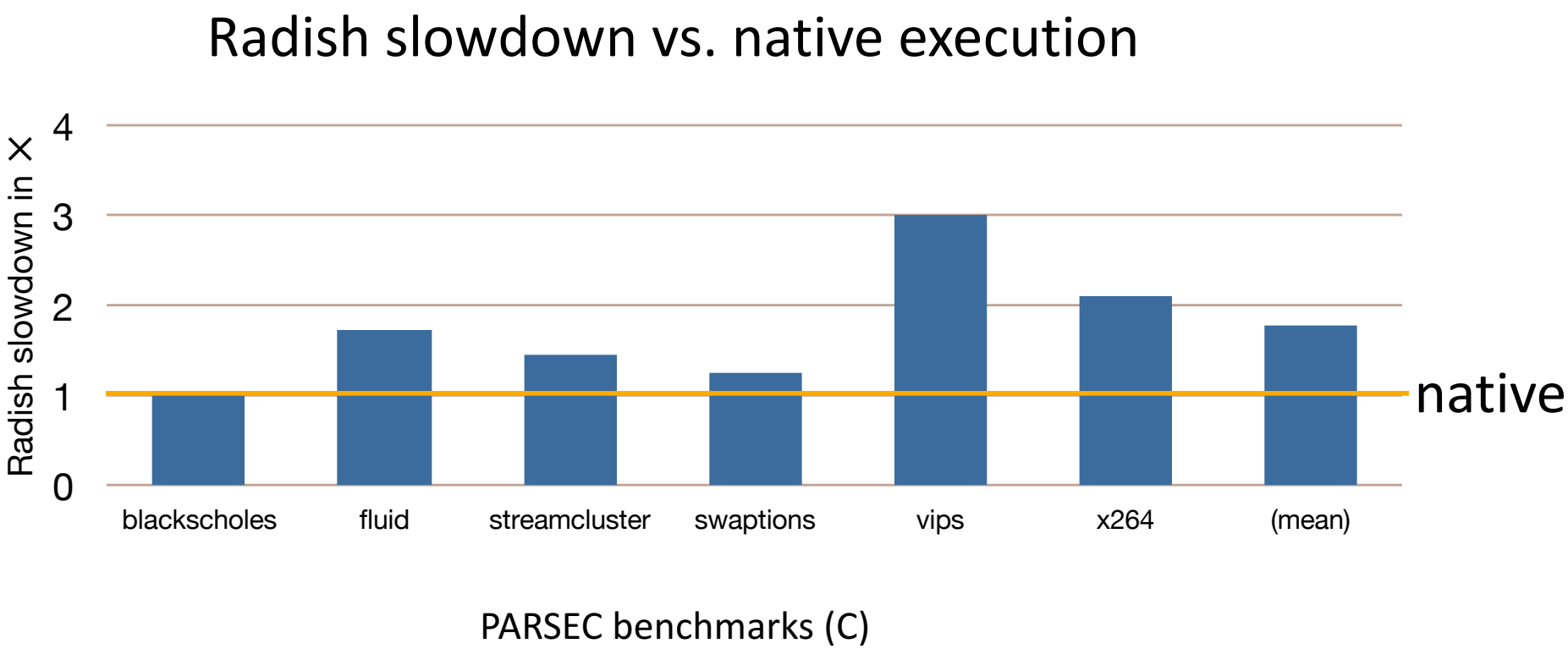
Recipe:

- Access history of every memory location
  - Check/update on every memory access.
- Sync history of every lock
  - Update on every lock operation.



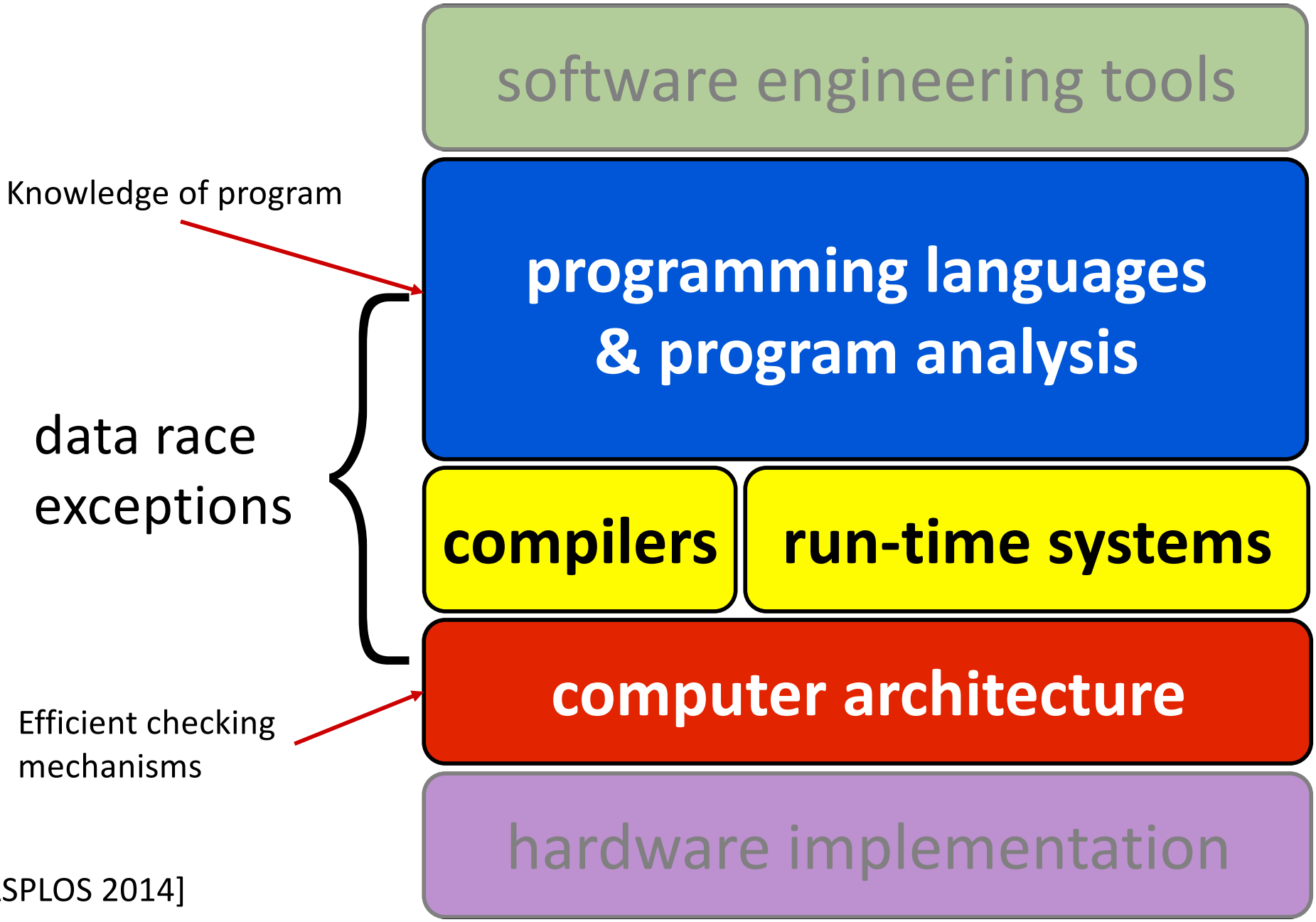
Radish: faster + accurate





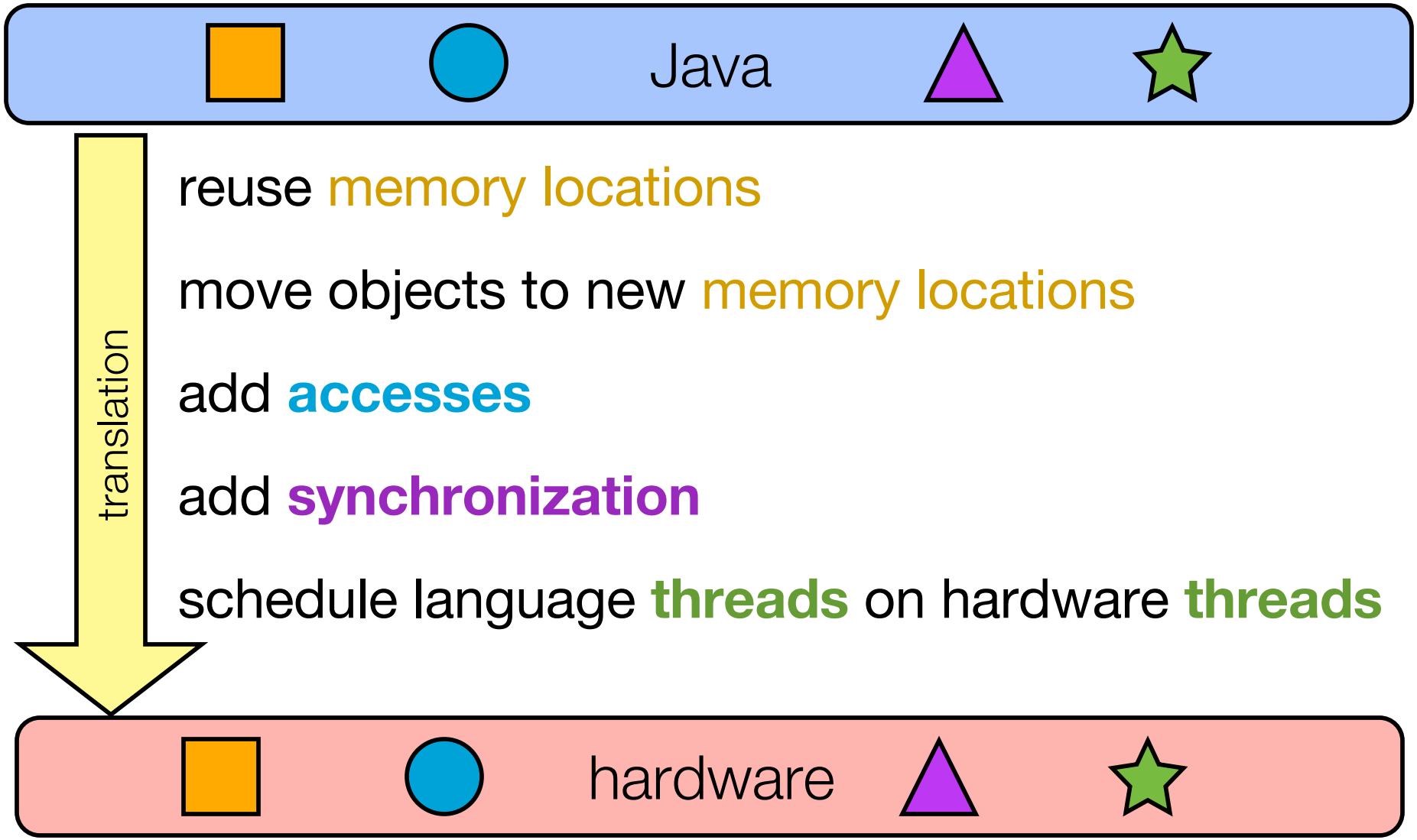
[ISCA 2012]

software reliability toolbox



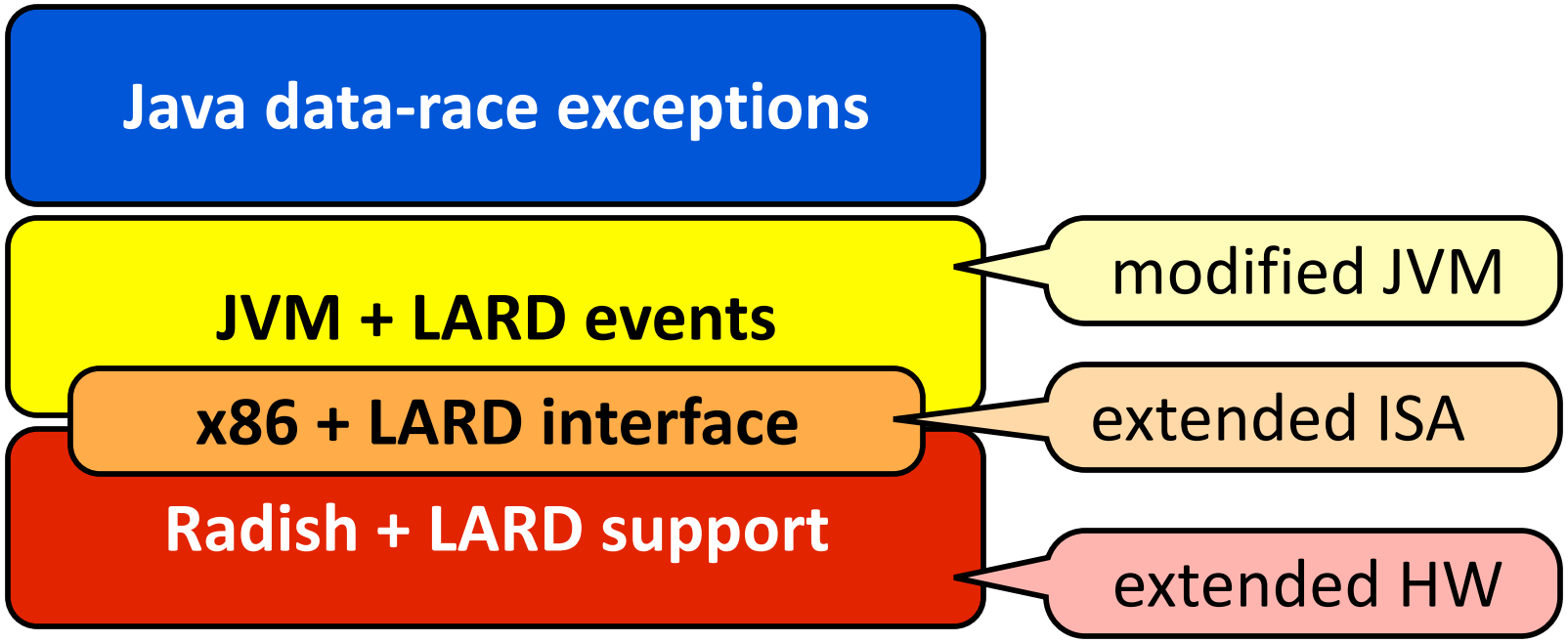
[ASPLOS 2014]

Translation affects data races.



[ASPLOS 2014]

Fast (HW) + Accurate (SW)



[ISCA 2012, ASPLOS 2014]

3 Skills for Thinking and Programming

Few of you will build new HW, OS, compiler, but…

1. Effective programmers understand their tools and systems.
2. The skills and ideas you learn here apply everywhere.

Reason about computational models, translation.

Debug for correctness and performance (with tools to help).

Assess costs and limits of representations.

"Figure it out" via documentation, experiments, critical thinking.

4 Foundations