Finding concurrency bugs

Subtle bugs. Testing is difficult → coverage → reproducing.

TSVD: effective ideas/techniques.

Static analysis

Non-local
Inter-procedural
Spec for function → hard for concurrency to infer.

Key problem: interleavings

Runtime testing

1) Define a bug.
   Wrong result.
   Crash
   Invariant

2) Explore schedules.
   Core non-determinism.
Strawman 1: run many times

Linux kernel
“lock torture”
“RCU torture”.

Hard to explore diff. schedules.
Hard to reproduce.

Strawman 2: exhaustive

Good: coverage?
Bad: not practical, too many, too similar.
Good: repeatable.

Research on exhaustive checking.
“Preemption points” $\rightarrow$ lock acquire
Few preemptions to needed.
Optimizations to avoid equiv. schedules.
Approach: concurrency invariants

Invariant: lock sets

Eraser
Inv: any shared mem. loc. accessed while holding some lock.

Invariant: data races
Inv: no concurrent r/w to same mem. loc.

Invariant: happens-before checking
Track dependencies between threads.
Inv: no un-ordered r/w to same addr.

% go test -race.
Clang TSan.

+ Fewer runs.
+ No need for exact race
- False positive

HB hard to get right.
Q: What is a bug (TSV)?
False pos? Missed bugs?

TSV invariant
C# data structures
Dictionary
List
Queue
ArrayList...

Read set
Write set.
Invariant: nothing concurs w/ write op.

Why low pp?
Many benign data races.
~ No benign TSV races.

Data race inv.
One mem. loc. u64

ATOMIC IN HW
u328

Read R set
Write SW set.
Inv: nothing concurs w/ write op.

FP?
Pair (R,W) that have no crash/issue.

Scheduling
register "tag" R/W.
sleep(300 msec)
List.append()
DataCollider.

List.locked()
When to inject sleep?

Near miss:

Δt < 100 msec [insert TSV!]

Log

Infer happens-before

sleep (100 msec)

insert unlock

Δt ≈ 100 msec

lock lookup

lookup