Lecture 4: Reproducing Bugs?

Announcements

- Checkpoint 1
  - Mechanics

- Final Project Proposals? Groups?

Today

Checkpoints in Distributed Programs
→ Why?

1. Relates to two framing questions
   → Obvious: How to use traces.
   → Less Obvious: Trace size: what to record.

2. An old problem with no really good solution
   → Why? Concurrency within processes & between processes.

3. Should, in theory, be more tractable today.
   → Why?

→ Why Useful?

→ Friday attempts to show one reason
But let us briefly compare to Pivot Tracing

- Pivot Tracing Goal
  - Replay Goal

- How/Why does Pivot Tracing help?
  - Reduces trace size

- When is it bad?

- Can Replay help?

Idealized Record + Replay
WHEN CAN THIS IDEALIZED VERSION WORK?

Assumptions:
- In any state, in any time
- Deterministic
Problems with the idealized version

How much to replay?
Approaches to solving these problems?

- Record More:
  - Detect concurrent access & record operation order
  - During replay, don't allow concurrency & control order

- Pros/Cons

- Disallow concurrency
Replay more

- Try different orderings for concurrent/racing accesses
  - Find one that produces the same state.

Problems

- Checking resulting state is the same?
  - How do we know replay order = original order?

- How do we even know all possible places where ordering matters?
  - Replay time/complexity?
Bottom Line: All Approaches Need Some Domain Assumptions

- Way To Identify Concurrency

- Way To Identify Events/Execution Points Where Ordering Might Matter

- Way To Check Equivalence

... 

Approaches Thus Far

Record More (Large Trace, Fast Replay)

Replay More (Smaller Trace, Maybe Infeasible Replay)
Can we use domain knowledge to reduce replay complexity?

- One idea: Commutativity

More generally: Equivalence.

Challenge: State.

Why?
How to solve (maybe) state?

Depth First Search.

Limitations?

Fitting this all back into our setting.

- What we do have

- What we are missing
- Why there is some hope?
  - System calls & scheduler as sources of events

- Data races?

- Do we actually get savings on traces?
Limiting what we replay

Reminder: Why we have a problem?

What do we want ideally?

- Minimize number of events we have to replay.

Tools

- Checkpoints?

- Minimization?
When do replay and minimization make sense?

- Correctness?

- Performance?

- Resource Exhaustion