Co-evolving tracing and fault injection with Box of Pain

Daniel Bittman     Ethan Miller     Peter Alvaro

Center for Research in Storage Systems
Disorderly Labs
UC Santa Cruz
"Time of fault" bugs
Box of pain

- Tracing
- Fault Injection
Coevolution

Tracing

Fault Injection
Philosophy

Focus: rare but catastrophic "time of fault" bugs

Coarse tracing: communication across failure boundaries

Simulation: *Effects* of faults rather than their causes

Pragmatism: many runs are possible; few are likely
Box of Pain - architecture

Tracer
Tracker
Fault Injector
Interpose. But where?
Key

- **X↓**: Syscall X entry
- **X↑**: Syscall X return

**W**: write
**R**: read
**C**: connect
**A**: accept
connect to 127.0.0.1:8080

result: TCP conn:
addr: 127.0.0.1:9876
peer: 127.0.0.1:5678

bind socket 3 to 127.0.0.1:8080

wait for connection on socket 3

result: TCP conn:
addr: 127.0.0.1:5678
peer: 127.0.0.1:9876

Key

- X↓: Syscall X entry
- X↑: Syscall X return
- W: write
- R: read
- C: connect
- A: accept
- B: bind
connect to 127.0.0.1:8080

result: TCP conn: addr: 127.0.0.1:9876 peer: 127.0.0.1:5678

bind socket 3 to 127.0.0.1:8080

wait for connection on socket 3

result: TCP conn: addr: 127.0.0.1:5678 peer: 127.0.0.1:9876

Key
- \( \downarrow \) Syscall X entry
- \( \uparrow \) Syscall X return
- W: write
- R: read
- C: connect
- A: accept
- B: bind

A happens-before B
connect to 127.0.0.1:8080

result: TCP conn: addr: 127.0.0.1:9876 peer: 127.0.0.1:5678

bind socket 3 to 127.0.0.1:8080

wait for connection on socket 3

result: TCP conn: addr: 127.0.0.1:5678 peer: 127.0.0.1:9876

Key
-X↓ Syscall X entry
-X↑ Syscall X return
W: write
R: read
C: connect
A: accept
B: bind

A → B: A happens-before B
write to socket n [127.0.0.1:9876]

read from socket n' [127.0.0.1:9876]

Key

- $X\downarrow$: Syscall $X$ entry
- $X\uparrow$: Syscall $X$ return
- $W$: write
- $R$: read
- $A \rightarrow B$: A happens-before B
write to socket n [127.0.0.1:9876]

read from socket n' [127.0.0.1:9876]

Key
- $X\downarrow$: Syscall $X$ entry
- $X\uparrow$: Syscall $X$ return
- W: write
- R: read

A $\rightarrow$ B: A happens-before B
Key
- **X↓**: Syscall X entry
- **X↑**: Syscall X return

W: write
R: read
C: connect
A: accept
Key

X↓  Syscall X entry
X↑  Syscall X return
W: write
R: read
C: connect
A: accept
Key

- X↓: Syscall X entry
- X↑: Syscall X return
- W: write
- R: read
- C: connect
- A: accept
Box of Pain

Tracer

Tracker

Fault Injector

Trace Run

Seen before

Compare to previous runs

Keep tracing, Maybe FI

Different from all previous runs

Finish run & emit new trace
Box of Pain

Tracer

Tracker

Fault Injector
Fault Injection

Philosophy:

Don't overthink it
Fault Injection
Fault Injection

Primitives:

(possibly infinite) delay

Explicit errors
Simulating real-world faults

Process crash:

prevent delivery of outgoing messages

Network partition:

prevent delivery of messages between pairs of processes
Nondeterminism and runs

Philosophy:

Arbitrarily many possible traces

Some are much more common than others!
Preliminary Results: the tails are long
Next Step

Integrate with Lineage-driven Fault Injection

Key challenge: abstract representation of traces
Remember

Focus: rare but catastrophic "time of fault" bugs

Coarse tracing: communication across failure boundaries

Simulation: *Effects* of faults rather than their causes

Pragmatism: many runs are possible; few are likely
Backup slides
Real motivation
Real motivation

Observation → Fault Injection
Real motivation

Observation  LDFI  Fault injection
Real motivation
Real motivation

Observation

LDFI

Fault injection
Real motivation

Observation → LDFI → Fault injection
Real motivation