REPT: Reverse Debugging of Failures in Deployed Software

Weidong Cui¹, Xinyang Ge¹, Baris Kasikci², Ben Niu¹, Upamanyu Sharma², Ruoyu Wang³, and Insu Yun⁴
Microsoft Research¹
University of Michigan²
Arizona State University³
Georgia Institute of Technology⁴

OSDI 2018, Carlsbad, CA
What happened before the crash?
REPT: Reverse Execution with Processor Trace
REPT: Reverse Execution with Processor Trace

- Online hardware tracing (e.g., Intel Processor Trace)
  - Log the control flow with timestamps
  - Low runtime overhead (1 – 5%)
  - *No data!*

- Offline binary analysis
  - Recovers data flow from the control flow
REPT Data Recovery

• Single-threaded execution reconstruction

• Multi-threaded execution reconstruction
Core Dump

Instruction Sequence

Execution History

How to recover overwritten states
lea rbx, [g]

mov rax, 1

add rax, [rbx]

mov [rbx], rax

xor rbx, rbx
lea rbx, [g]
mov rax, 1
add rax, [rbx]
mov [rbx], rax
xor rbx, rbx

rax=3, rbx=0, [g]=3
lea rbx, [g]
mov rax, 1
add rax, [rbx]
mov [rbx], rax
xor rbx, rbx

rax=?, rbx=?, [g]=3
rax=?, rbx=g, [g]=3
rax=1, rbx=g, [g]=3
rax=3, rbx=g, [g]=3
rax=3, rbx=?, [g]=3
rax=3, rbx=0, [g]=3
lea rbx, [g]

mov rax, 1

add rax, [rbx]

mov [rbx], rax

xor rbx, rbx

rax=3, rbx=g, [g]=3

rax=1, rbx=g, [g]=3

rax=?, rbx=?, [g]=?

rax=3, rbx=g, [g]=3

rax=3, rbx=g, [g]=?
lea rbx, [g]
mov rax, 1
add rax, [rbx]
mov [rbx], rax
⇒ xor rbx, rbx

rax=?, rbx=?, [g]=2
rax=?, rbx=g, [g]=2
rax=1, rbx=g, [g]=2
rax=3, rbx=g, [g]=3
rax=3, rbx=0, [g]=3
rax=3, rbx=g, [g]=3
lea rbx, [g]
mov rax, 1
add rax, [rbx]
mov [rbx], rax
xor rbx, rbx

rax=1, rbx=g, [g]=2
rax=3, rbx=g, [g]=2
rax=3, rbx=g, [g]=3
rax=3, rbx=0, [g]=3
lea rbx, [g]

mov rax, 1

add rax, [rbx]

mov [rbx], rax

xor rbx, rbx

rax=3, rbx=g, [g]=2

rax=3, rbx=g, [g]=3

rax=3, rbx=0, [g]=3
Key Techniques

• Forward Execution
  • Recovers states before irreversible instructions

• Error Correction
  • Handles errors introduced by “missing” memory writes
REPT Data Recovery

• Single-threaded execution reconstruction

• Multi-threaded execution reconstruction
How to determine the thread interleavings?
Key Techniques

• Hardware Timestamps
  • Constructs a partial order

• Concurrent memory write detection
  • Constrains their usage to avoid propagating a wrong value
With REPT, ...
Hey, client, turn on tracing next time.

I want history information!
Demo
16 bugs
14 bugs
1-5% overhead
92% accuracy
Conclusion

• Debugging production failures is important but hard

• REPT is a practical reverse debugging solution for production failures
  • Online hardware tracing to log the control flow with timestamps
  • Offline binary analysis to recover the data flow with high accuracy

• REPT has been deployed on Microsoft Windows