Automated Debugging in Datacenters

Bogdan A. Stoica, James R. Larus, Swarup K. Sahoo, Vikram S. Adve

1 Ecole Polytechnique Federale de Lausanne
2 Carnegie Mellon University
3 University of Illinois at Urbana-Champaign

Motivation

• Bugs are difficult to find until they surface
  → Scale, complexity and fault tolerance hinder analysis
• Live (“in-production”) debugging is disruptive
  → Large overheads limit runtime tracing capabilities
• Post-mortem debugging is difficult
  → Logs capture too much, yet little is useful to diagnosis

Premises

• Bugs occur repeatedly at scale
  → Need not catch the first occurrence, it will happen again
• Bugs often reside on cold execution / data paths
  → Cold paths are rarely tested, if at all
• Monitoring can be adapted as new evidence is collected
  → Refine assumptions, re-instrument and deploy again

Insights

• Control flow runtime monitoring
  → Intel Processor Trace for “always on” low overhead tracing
• Data dependencies runtime monitoring
  → Hardware breakpoints for low overhead memory tracing
  → Statistical aggregation to cover multiple addresses
• Piecewise dynamic slice reconstruction over multiple runs
  → Aggregate pieces observed over many production runs
  → Increase accuracy by instrumenting imprecise dependencies

Diagnosis Cycle

1. Deploy
2. Instrument
3. Datacenter
4. Live tracing
5. Refine

Under the Hood

Challenges

• Missed cold runtime data dependencies?
  → Re-instrument cold basic blocks and / or …
  → Fallback on control flow and static dependencies
• Incorrect runtime data dependencies for the failing run?
  → Re-instrument control flow deltas between ✓ and ❌ runs
• Overhead vs. diagnosis time tread-off
  → Low overhead = larger sample rate ⇒ longer diagnosis time

Work in Progress

• Finalize prototype
  → Dynamic instrumentation via DynamoRIO
• Evaluate and validate
  → Accurate backward slice vs. approximate backward slice on real-world reported bugs