Relyzer: Application Resiliency Analyzer for Transients Faults

Siva Kumar Sastry Hari†, Sarita Adve†, Helia Naeimi‡, and Pradeep Ramachandran†

†University of Illinois at Urbana-Champaign, ‡Intel Corporation

Resilient System Task # 5.5.3

GSRC Annual Symposium

Nov 16, 2011

PUB ID 2684

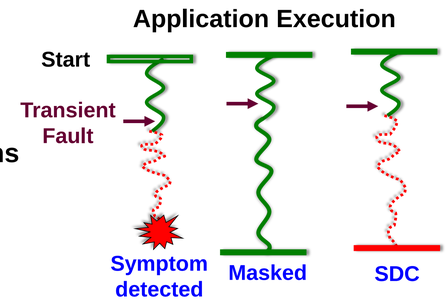
Motivation

Software anomaly detectors are low-cost and effective

Evaluated through statistical fault injection experiments

Limitations: Cannot provide guarantees

Cannot identify SDC-prone application sections



Goal: Analyzing all transient faults affecting an application

Advantages: Provide guarantees on detection mechanisms

List all SDC causing fault locations

Challenges: Do we need fault injections for all the faults?

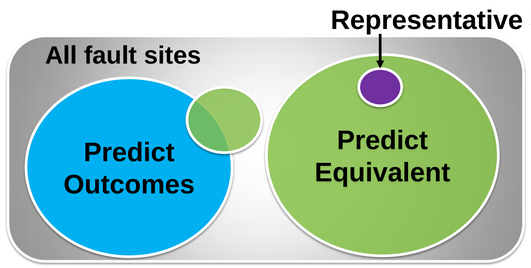
How to analyze all faults with fewer fault injections?

Relyzer Overview

Fault Model: Transient bit flips in register operands of every executing instructions

Example: opcode rs1, rs2, rd

Fault sites: Single-bit flips in all bit locations of rs1, rs2, and rd



Remaining Fault sites = - - +

Fault Pruning Techniques

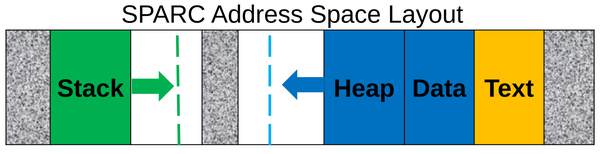
Determining Outcomes: Address Bounding

Prune out of bounds accesses

Memory addresses (&)

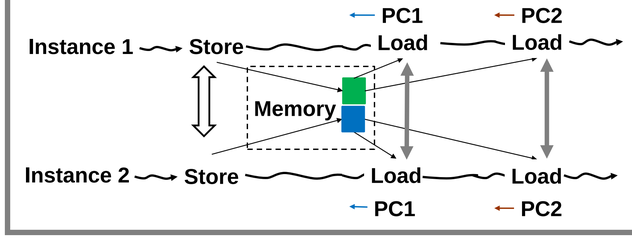
Branch targets ( )

SPARC Address Space Layout



Predicting Equivalence: Store Analysis

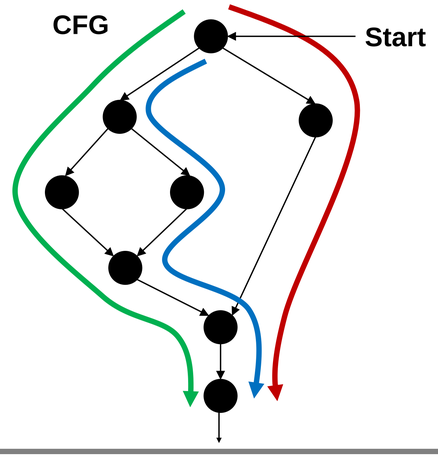
Idea: Faults in store instructions may behave similarly if the value usage pattern is same



Predicting Equivalence: Control Analysis

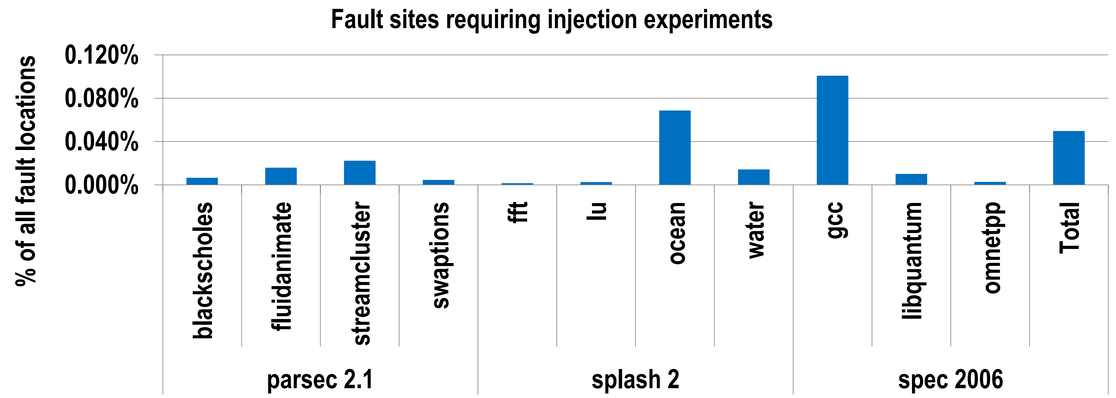
Idea: Faults flowing through similar control paths may behave similarly

Example: Faults at *start* are categorized based on how they flow through the app



Pruning Results

Total fault locations = 1.02 Billion



Total Remaining Faults (for 11 apps) = 505.9 Million

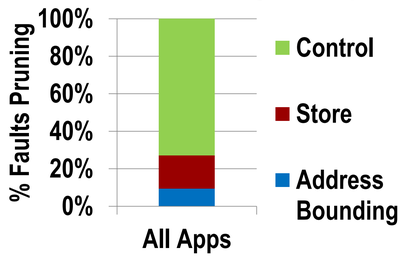
99% execution is represented by 42.5 Million fault sites

Injected faults in all the remaining fault sites for 6 applications

First study to analyze impact of all fault sites

Results are similar to high confidence random fault injections

Contribution of Pruning Techniques

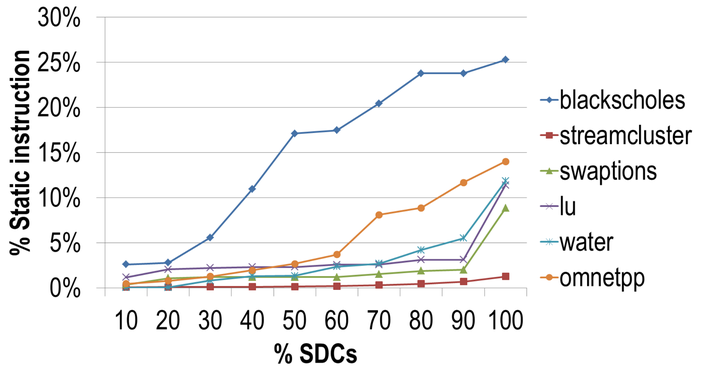


Finding SDC-causing Application Sites

Relyzer lists all the SDC causing sites

Over 90% of SDCs come from <18% of app instructions

Aids low-cost development of fault detectors



Conclusions and Future Work

Relyzer analyzes all application-level fault sites by studying fewer faults

Less than 0.005% fault sites require fault injections

Lists SDC causing application instructions

Understand application properties leading to SDCs

Thoroughly validate the pruning techniques

Develop low-cost application-level hardware fault detectors