LLFI: An Intermediate Code-Level Fault Injection Tool for Hardware Faults

Qining Lu, Mostafa Farahani, Jiesheng Wei, Anna Thomas, and Karthik Pattabiraman

Department of Electrical and Computer Engineering, University of British Columbia
Motivation

• Hardware faults’ frequency is increasing [Borkar’11][Constantinescu’08]
  – Temperature variations
  – Soft errors due to particle strikes
  – Manufacturing defects
Hardware Errors: Traditional “Solutions”

- **Guard-banding**
  
  Guard-banding wastes power as gap between average and worst-case widens due to variations.

- **Duplication**
  
  Hardware duplication (DMR) can result in 2X slowdown and/or energy consumption.
Application-level techniques

Impactful Errors

Application Level
Operating System Level
Architectural Level
Device/Circuit Level
Fault Injection

- To obtain coverage estimates for applications
- Iteratively improve coverage based on the errors missed by fault tolerance mechanisms
- Analyze the errors that are missed by the FTMs

- Insert fault tolerance mechanisms in the application’s source code
- Inject Faults into application protected with fault-tolerance mechanisms
- Is obtained coverage sufficient?
High-Level Vs. Low-Level Injectors

Accuracy

Low-level Injectors

Gap is not large in practice [Wei-DSN’14]

High-level Injectors

Ease of Analysis and Configurability
Outline

• Motivation

• LLFI: High Level Fault Injector

• Experimental Setup and Results

• Conclusion
LLFI: High-Level Fault Injector

- A fault injector based on LLVM
  - Intermediate representation (IR) level injection

- Features
  - Easy to customize the fault injection
  - Easy to analyze fault propagation
  - Easy to use by developers
LLFI: Structure

Works at LLVM compiler’s intermediate (IR) code level [Lattner’05] – widely used in industry
LLFI: Workflow

**Compile-time**
- Source code
  - LLVM Compiler
  - IR code
  - Instrument
    - Injection executable
    - Profiling executable
      - Profile
        - Dynamic counts of injection locations
      - Run injection executable
        - Inject Fault
          - Run injection executable
            - Injection Results
            - Statistics of injected faults
          - Run-time options
            - Compile-time options
LLFI: Easy to Customize

• Fault injection instruction selector
  – Based on instruction type (LLVM IR)
    • Include: add + cmp
    • Exclude: load
  – Based on register target (source/destination)
  – Can choose type of fault to inject (single/ double bit flip, stuck at fault etc.)
LLFI: Easy to Analyze

- Graphical output of trace differences
LLFI: Easy to Use
Outline

• Motivation

• LLFI: High Level Fault Injector

• Experimental Setup and Results

• Conclusion
Research Questions

- **RQ1**: Effect of injecting faults into different types of instructions in the program

- **RQ2**: Effect of injecting faults into different kinds of registers (source or destination)

- **RQ3**: Effect of single bit flip vs. double bit flip fault type, and ‘distance’ between them
Experimental Setup

• **Benchmarks:**
  - *PARBOIL* benchmark suite for CPUs

• **Method:**
  - 1000 fault injections per benchmark per experiment
  - One fault per run (single bit flip)

• **Outcomes:**
  - Crash, Hang, Silent Data Corruption (SDCs) or Benign
Results: RQ1 (Instruction Types)

Failure distribution of injecting different instruction types
Results: RQ2 (Register Types)

Failure distribution of injecting into source register
Results: RQ2 (Register Types)

Failure distribution of injecting into destination register
Results: RQ3 (Single vs. Double Bit Flip in same Register)

Failure distribution of single bit-flip vs. double bit-flip
Results: RQ3 (Window Size)

Effect of window size for double bit flip
Outline

• Motivation

• LLFI: High Level Fault Injector

• Experimental Setup and Results

• Conclusion
Conclusion

• LLFI is a highly configurable fault injection tool that works at the LLVM IR level

• Both Instruction type and register target are highly correlated with failure outcome

• Double bit fault increases the crash rate but not the SDC rates, and that too only within a short window of <5 instructions

https://github.com/DependableSystemsLab/LLFI