Fault Injection using LLFI

EECE 513: Error Resilient Computing Systems

Agenda

- Brief review from last class
- Fault injection
- Introduction to LLFI + Demo

Review from last class

Dependable systems design

- Hardware dependability
- Duplication and TMR
- Software approaches
- Parallel systems
- Distributed systems
- Case studies of real world systems

Dependability evaluation techniques

- Combinatorial methods
- Fault-injection
- State-based methods
- Statistical methods

Review from last class

Dependable systems design

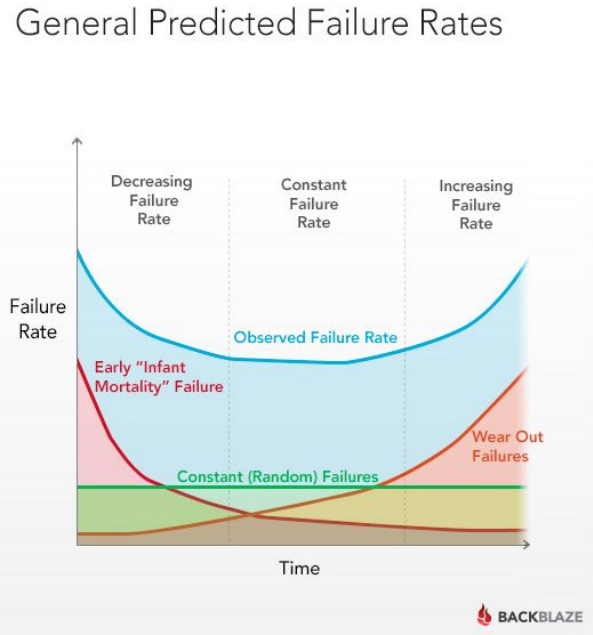
- Hardware dependability
- Duplication and TMR
- Software approaches
- Parallel systems
- Distributed systems
- Case studies of real world systems

Dependability evaluation techniques

- Combinatorial methods
- Fault-injection
- State-based methods
- Statistical methods

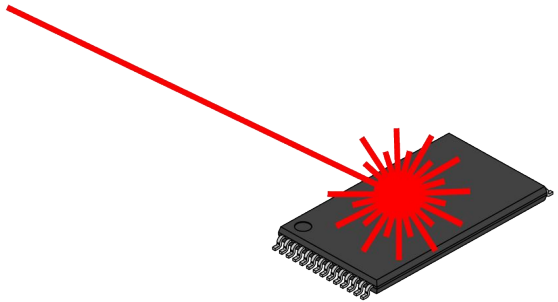
Evaluating Dependability

- We need methods to evaluate the dependability of a system.

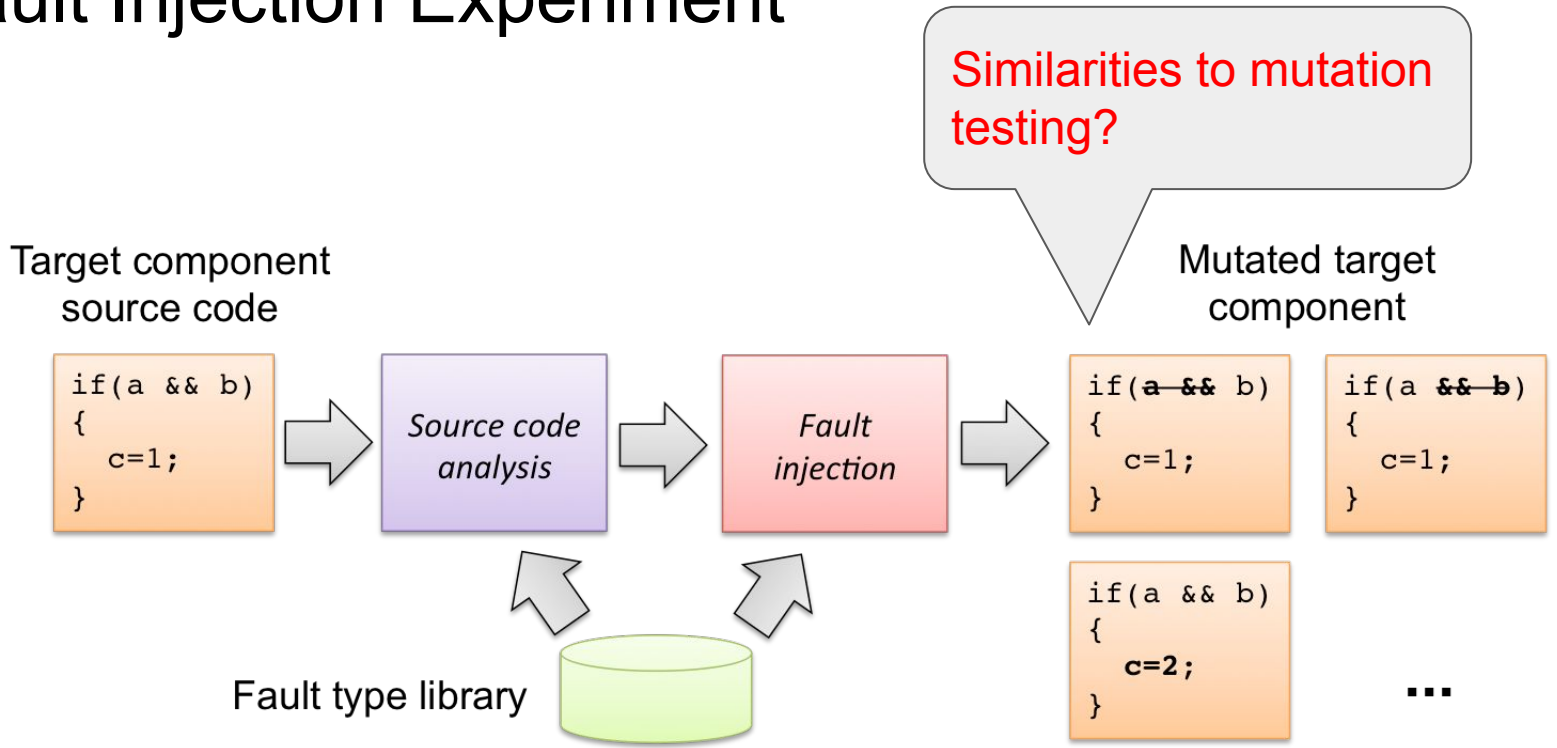


What is Fault Injection?

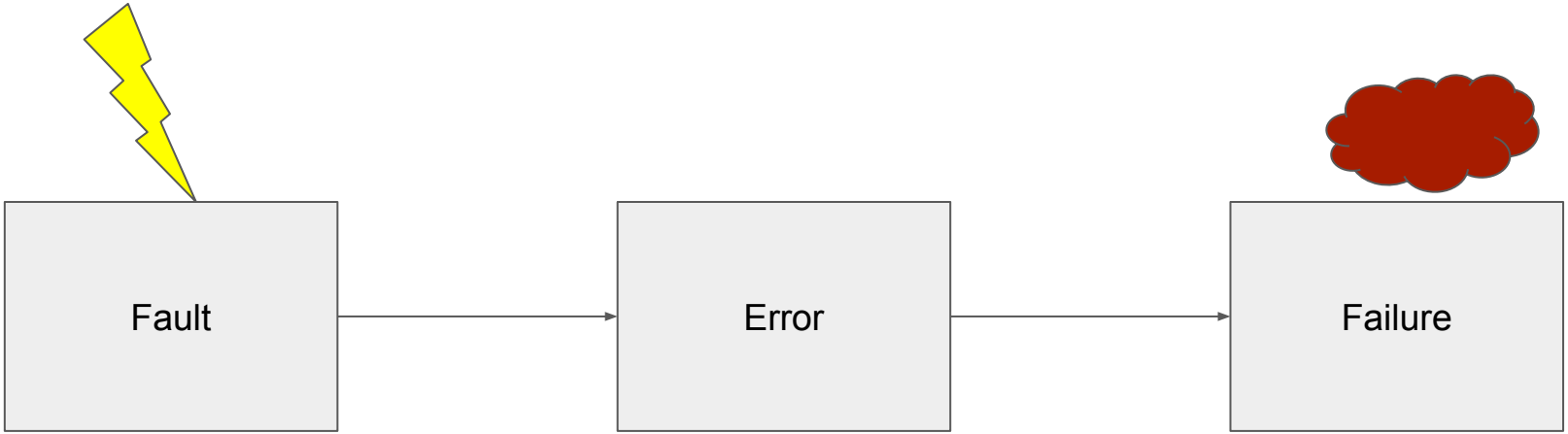
- Inject faults into a program and observe the program's behaviour under the fault
- Evaluates the error resilience of a program
- Guide design decisions around system robustness



Fault Injection Experiment



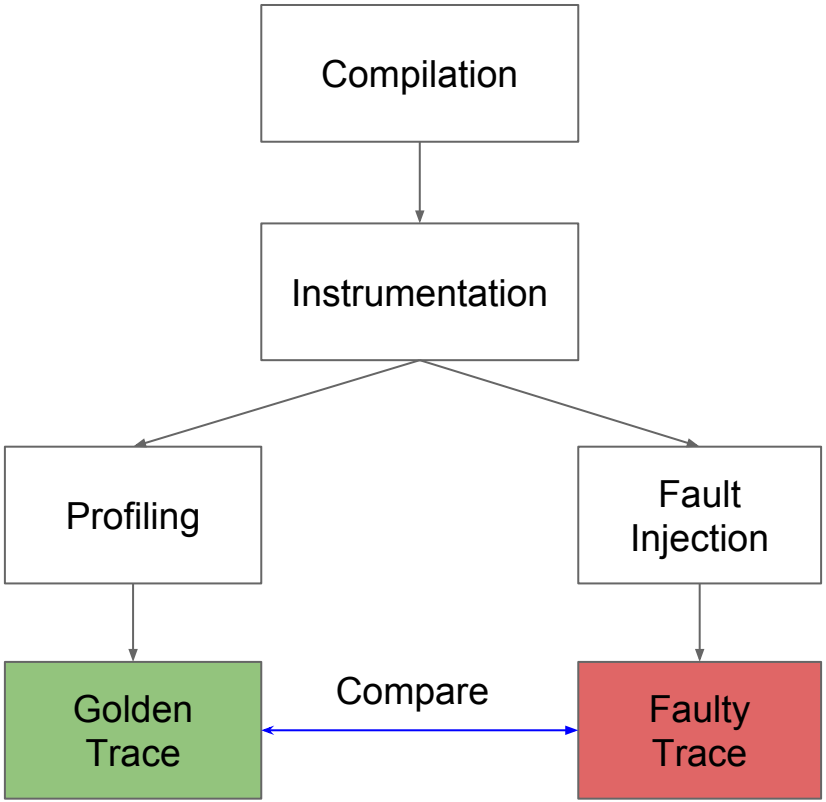
Fault Model



LLFI

- Software-Implemented Fault Injection (SWiFI) tool
- Implemented using the LLVM compiler
- Randomly injects faults into programs and enables us to study their effects
- Developed at the UBC Dependable Systems Lab
- Available on Github:
  - https://github.com/DependableSystemsLab/LLFI

LLFI Workflow



Fault Types

- Hardware Faults
  - Bit flip
  - Stuck at 0/1

- Software Faults
  - File I/O Buffer Overflow
  - Buffer Overflow Malloc
  - Function Call Corruption
  - Invalid Pointer
  - Race Condition

Challenge: Creating faults representative of real world scenarios

Failure Modes

- Benign
  - Program executes and returns the correct outputs
- Crash
  - Program prematurely terminates
- Hang
  - Program never terminates (within a timeout period)
- Silent Data Corruption (SDC):
  - Program executes but returns erroneous outputs

Example Scenario: Bit flip

Objective: Evaluate the resilience of controller firmware on an airplane to bit flips

- A bit flip is a common type of soft error in hardware: 0 → 1, 1 → 0
- Soft errors arise randomly and naturally from alpha particles / cosmic radiation
- Cosmic radiation is more prominent at higher altitudes
- Upon encountering a bit flip, will the controller firmware crash, return corrupted output(s), or return the correct output(s)?

Getting Started with LLFI

1. Install LLFI with all its dependencies
2. Execute the command line version of LLFI by navigating to its folder
3. Run the factorial example

More info on LLFI

- Check out the README and Wiki pages on the LLFI GitHub page
- Post additional questions on Piazza