Fehlerinjektion

Department Informatik 4
Systemsoftware
Friedrich-Alexander-Universität Erlangen-Nürnberg

WS 2026/2027



Fehlerinjektion

Möchte man ausprobieren, wie

- Fehler in Komponenten,
- Fehler in Leitungen

sich auf das Systemverhalten auswirken, kann man Fehler in virtuelle Maschinen injizieren und deren Verhalten beobachten.

Problem: Fehlerinjektion soll Zeitverhalten gar nicht/wenig beeinflussen.

Idee: Nur Zugriffe auf fehlerhafte Komponenten/Leitungen abfangen und Zugriff mit Fehlern behaftet emulieren.



Fehlerinjektion – Registerfehler

Fehler in der CPU selbst können durch andere Just-In-Time-Compilierung modelliert werden. Z.B. Bit 1 in %eax defekt (immer 0):

Original-Code	JIT-Code ohne Fehler	JIT-Code mit Fehlern

`movl $2, %eax`
`addl $3, %eax`
	`movl $2, offeax(%ebp)`
	`addl $3, offeax(%ebp)`
		`movl $2, offeax(%ebp)`
		`andl $~2, offeax(%ebp)`
		`addl $3, offeax(%ebp)`
		`andl $~2, offeax(%ebp)`

Code nur unwesentlich langsamer als im fehlerlosen Fall.



Fehlerinjektion – Speicherfehler

Sollen Fehler im Speicher simuliert werden, kann man der CPU direkten Zugriff auf alle Pages im Speicher bis auf die eine defekte geben.

Zugriffe auf die eine defekte Seite im Speicher werden über Callback-Funktionen an das Speicher-Modul weitergeleitet. In den `load`- bzw. `store`-Funktionen kann dann das fehlerhafte Verhalten modelliert werden.

- Zugriff auf fehlerfreie Seiten nicht verlangsamt.
- Zugriff auf fehlerhafte Seite i.A. selten.

=> (fast) keine Performance-Einbuße.



Fehlerinjektion – Gerätefehler

Zugriffe auf alle Geräte laufen über Callback-Funktionen. In den `inb`- bzw. `outb`-Funktionen kann dann das fehlerhafte Verhalten modelliert werden.

„Defekte Platte einfacher zu simulieren als fehlerfreie...”

=> keine Performance-Einbuße.



Fehlerinjektion

Fehlerinjektion i.A. mit Aufwand verbunden (zusätzliche Fehler-Simulationsmethoden notwendig).

Jedoch keine (großen) Performance-Einbußen!

