

Verlässliche Echtzeitsysteme

Übungen zur Vorlesung

Codierung, Fehlerinjektion

Tobias Klaus, Florian Schmaus, Peter Wägemann

Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl Informatik 4 (Verteilte Systeme und Betriebssysteme)
<https://www4.cs.fau.de>

19. April 2016



- 1 C-Quiz Teil IV
- 2 Fehlerinjektion mit FAIL*
- 3 Wiederholung Software-TMR
- 4 Eliminierung von Bruchstellen in TMR
- 5 Aufgabenstellung



- 1 C-Quiz Teil IV
- 2 Fehlerinjektion mit FAIL*
- 3 Wiederholung Software-TMR
- 4 Eliminierung von Bruchstellen in TMR
- 5 Aufgabenstellung



- C99
- x86 bzw. x86-64, d. h.
 - vorzeichenbehaftete Integer als Zweierkomplement implementiert
 - char hat 8 Bit
 - short hat 16 Bit
 - int hat 32 Bit
 - long hat 32 Bit auf x86 und 64 Bit auf x86-64



Frage 10

Angenommen x hat Typ `int` und ist positiv. Ist $x \ll 1 \dots$

1. definiert für alle Werte
2. definiert für manche Werte
3. definiert für keinen Wert

von x ?

Erklärung

- Es darf nicht in das Vorzeichenbit hineinverschoben werden
- ⇒ nicht definiert für große Werte von x



Frage 11

Angenommen x hat Typ `int`. Ist $x \ll 31 \dots$

1. definiert für alle Werte
2. definiert für manche Werte
3. definiert für keinen Wert

von x ?

Erklärung

- Es darf nicht in das Vorzeichenbit hineinverschoben werden
- ⇒ funktioniert hier nur mit $x == 0$



Frage 12

Angenommen `x` hat Typ `int`. Ist `x << 32 ...`

1. definiert für alle Werte
2. definiert für manche Werte
3. definiert für keinen Wert

von `x`?

Erklärung

- Verschiebung um Bitbreite eines Datentyps nicht zulässig



- 1 C-Quiz Teil IV
- 2 Fehlerinjektion mit FAIL***
- 3 Wiederholung Software-TMR
- 4 Eliminierung von Bruchstellen in TMR
- 5 Aufgabenstellung



- 1 C-Quiz Teil IV
- 2 Fehlerinjektion mit FAIL***
- 3 Wiederholung Software-TMR
- 4 Eliminierung von Bruchstellen in TMR
- 5 Aufgabenstellung



 https://www4.cs.fau.de/Lehre/SS16/V_VEZS/Uebung/Folien/Fail.pdf



```
#ifndef _include_fail_h
#define _include_fail_h

//Mark start of injection
volatile void __attribute__((noinline)) fail_start_trace(void);
//Mark end of injection
volatile void __attribute__((noinline)) fail_stop_trace(void);

//everything ok: valid code and right values
volatile void __attribute__((noinline)) fail_marker_positive(void);
//everything ok: valid code but wrong values
volatile void __attribute__((noinline)) fail_marker_negative(void);
//invalid code
volatile void __attribute__((noinline)) fail_marker_detected(void);

#endif /* _include_fail_h */
```

- Markierung Start/Ende für „Golden Run“
- Ziel: Minimierung Auftreten von fail_marker_negative



FAIL* – Beispiel: Verwendung Marker

```
void cyg_user_start() {
    ...
    fail_trace_start() // Start Fehlerinjektion
    filter()
    vote()              // (codierter) Mehrheitsentscheid
    fail_trace_stop()  // Ende Fehlerinjektion

    actuate()          // Signaturen entfernen, Fehler überprüfen
    ...
}

void actuate() {
    ...
    if (checksum == expected_checksum){
        fail_marker_positive() // SDC behandelt :-
    }else{
        fail_marker_negative() // undetektierte SDC :-
    }
}
```



- Datei anlegen: `~/ .my.cnf`
- Folgende Daten eintragen:
`[client]`
`user=vezs20`
`password=XXXXXXXXXXXXXXXXXX`
`host=i4jenkins.cs.fau.de`
`database=vezs20`
- Zugangsdaten wurden verteilt



1. make trace

- Erstellung des „Golden Runs“
- Speicherung der Experimente in Datenbank
- Faltung des Fehlerraums

2. make inject

- Durchführung Fehlerinjektion
- Überblick der Ergebnisse in lokal angelegter Datei: `ean_result.csv`
 - 👉 Abschätzung des zeitlichen Aufwands

3. make resultbrowser

- Startet lokalen Webserver
- Ergebnisse per Webbrowser einsehbar



Fail* Results x

localhost:5000/instr_details?table=result_GenericExperimentMessage&instr_address=1048643&variant_id=1

FAIL* Home About

Result by Instruction

Result Table **result_GenericExperimentMessage**
Variant **main**
Benchmark **mem**
Details
Instruction Address **0x100043 (Dec: 1048643)**
Total Results **32**

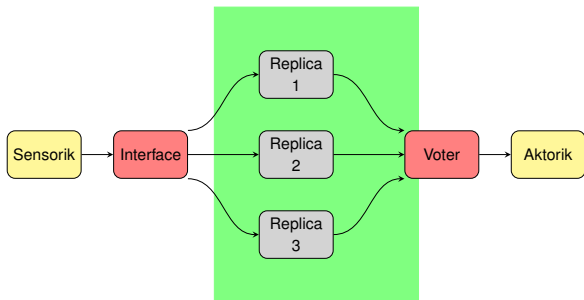
Code

Address	Opcode	Disassembly	Comment
0x10002f	90	nop	
0x100030	c7052010200064	movl \$0x64,0x201020	
0x100037	0000	00	
0x10003a	c3	ret	
0x10003b	6690	xchg %ax,%ax	
0x10003d	6690	xchg %ax,%ax	
0x10003f	90	nop	
0x100040	83ec0c	sub \$0xc,%esp	
0x100043	d9442418	flds 0x10(%esp)	
0x100047	dc442410	faddl 0x10(%esp)	
0x10004b	dd1c24	fstpl (%esp)	
0x10004e	dd0424	fldl (%esp)	
0x100051	83c40c	add \$0xc,%esp	
0x100054	c3	ret	
0x100055	6690	xchg %ax,%ax	
0x100057	6690	xchg %ax,%ax	
.....		...	

- 1 C-Quiz Teil IV
- 2 Fehlerinjektion mit FAIL*
- 3 Wiederholung Software-TMR**
- 4 Eliminierung von Bruchstellen in TMR
- 5 Aufgabenstellung



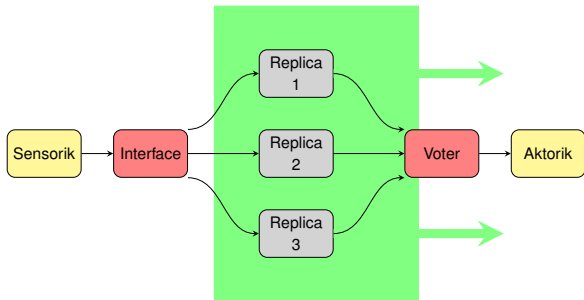
Klassische “Triple Modular Redundancy” (TMR)



- Schnittstelle sammelt Eingangsdaten (Replikdeterminismus)
- Verteilt Daten und aktiviert Replikate
- Mehrheitsentscheider (Voter) wählt Ergebnis
- Ergebnis wird an Aktuator versendet



Klassische “Triple Modular Redundancy” (TMR)



Redundanzbereich

Ausschließlich Replikatausführung.

- ~ Erweiterung der Ausgangsseite mit Informationsredundanz
- ~ Mehrheitsentscheid über codierte Prüfsumme



- 1 C-Quiz Teil IV
- 2 Fehlerinjektion mit FAIL*
- 3 Wiederholung Software-TMR
- 4 Eliminierung von Bruchstellen in TMR**
- 5 Aufgabenstellung



nach Forin 1989: "Vital coded microprocessor principles and application for various transit systems" [1]

- Arithmetisch codierter Wert V_C
- Ausgangswert

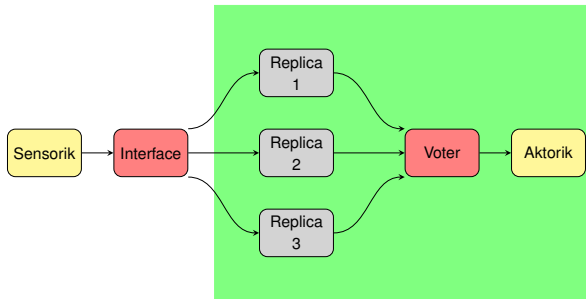
$$V_C = V * A + B_V + D$$

- Schlüssel
- Variablenspezifische Signatur
- Zeitstempel



- Schlüssel A sollte so groß wie möglich sein:
 - ↪ Möglichst geringe Restfehlerwahrscheinlichkeit ($P = 1/A$)
- Wertebereich des dynamischen Zeitstempels
 - $D = \{x \mid x \in \mathbb{N}_0 \wedge x \leq D_{max}\}$
 - Zeitstempel darf überlaufen: $D_{max} + 1 = 0$
- Für jede Signatur B_* muss dann gelten
 - $B_* + D_{max} < A$
 - Die minimale Distanz zwischen jeweils zwei Signaturen im System muss kleiner D_{max} sein: $\forall i, j : |B_i - B_j| < D_{max}$





- Replikate liefern arithmetisch codierte Ergebnisse
- Mehrheitsentscheid auf codierten Prüfsummen
- Übertragung codierter Ergebnisse



Für diese Übungsaufgabe:

- Kein Zeitstempel
- Nur Absicherung der Ausgangsseite!



EAN Vergleichsoperator

- Voting basiert auf codierter Vergleichsoperation:

$$\rightsquigarrow X_C = Y_C \Rightarrow X * A + B_X = Y * A + B_Y$$

- Im fehlerfreien Fall gilt:

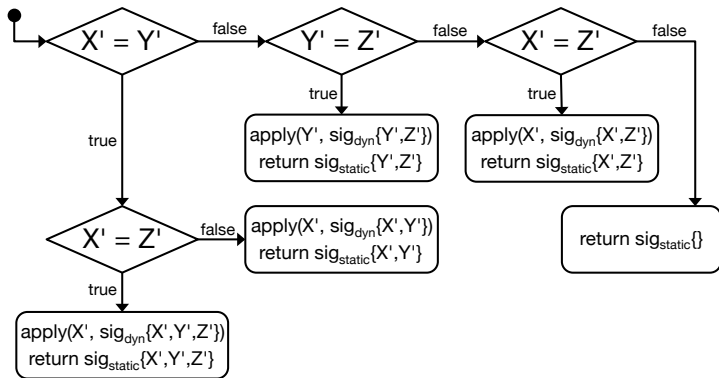
$$X = Y, A = A \text{ aber } B_X \neq B_Y !$$

- Rohwerte sind identisch
- Schlüssel ist per Definition identisch
- Signaturen sind unterschiedlich (aber konstant!)

Bestimmung der Gleichheit durch Differenzbildung:

$$\rightsquigarrow X_C - Y_C = B_X - B_Y = \text{const.}$$





■ Bestimmung von dynamischer und statischer Signatur:

~> $sig_{dyn}(X', Y') : X' = Y' \Rightarrow X' - Y'$

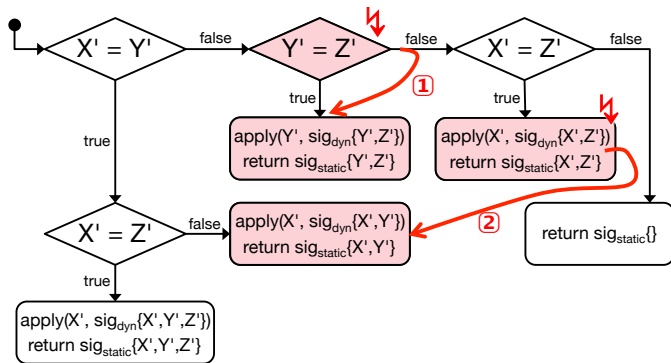
~> $sig_{static}(X', Y') : X' = Y' \Rightarrow B_X - B_Y$

1. Vergleichsoperation wird durchgeführt (z. B. $X' = Y' \wedge X' = Z'$)
 - Berechnung von sig_{dyn}
 - Vergleich mit sig_{static}
2. Verzweigungsentscheidung wird nachberechnet:
 - Wiederholte (redundante) Berechnung von sig_{dyn}
 - Addiere sig_{dyn} (*apply*) zum gewählten Ergebnis
3. Konstante Signatur des durchlaufenen Zweiges identifiziert Gewinner (Rückgabewert: sig_{static})
 - Akteur wählt entsprechendes Replikatergebnisse
 - Führt inverse Operation zu *apply* durch

Im Voter wurde die *dynamisch berechnete Signatur der Verzweigungsentscheidung* hinzu addiert. Im Akteur wird mit der entsprechenden *konstanten Signatur zurückgerechnet*.



Codierter Mehrheitsentscheid - Fehlerfall



1. Falsche Verzweigungsentscheidung: ($Y' \neq Z'$)

- Y' wird als korrekt angenommen, sig_{dyn} wird berechnet
- allerdings ist sig_{dyn} tatsächlich $\neq sig_{static}$
- Fehler wird bei der inversen Operation zu `apply` erkannt

2. Falscher (plötzlicher) Sprung

- X' wird als korrekt erkannt, sig_{dyn} wird erneut berechnet
- Ein fehlerhafter Sprung zu einem anderen Block führt zu einem inkonsistenten

- 1 C-Quiz Teil IV
- 2 Fehlerinjektion mit FAIL*
- 3 Wiederholung Software-TMR
- 4 Eliminierung von Bruchstellen in TMR
- 5 Aufgabenstellung**



Aufgabe

- Absichern des Voters per EAN mittels CoRed-Ansatz
 - Absichern des Akzeptanzmaskierers am Eingang
-
- Jedes Replikat hat genau einen Ausgabewert (integer \mapsto enc_t): eine codierte Prüfsumme des Ergebnisses
 - ↪ Festlegung für jeden der drei Ausgabewerte (X', Y', Z') jeweils *unterschiedliche* aber *konstante* Signatur (SIG_X, SIG_Y, SIG_Z)
 - Nutzung des nächstgrößeren Datentyps X für den ursprünglichen Wert X'
 - ↪ Wahl einer Zahl A mit möglichst großem Hamming-Abstand unter *Vermeidung* möglicher *Überläufe bei der Codierung*



Für jede Operation zwischen zwei codierten Werten

ist eine eigene Operation mit konstanten Signaturwerten notwendig!

- Bestenliste wird automatisch auf Webseite veröffentlicht
- Wie werden Resultate generiert?



Repository Zugriff

The screenshot shows a web browser window displaying the GitLab interface for a repository named 'Flansch / awesome-vezs'. The browser's address bar shows the URL 'https://gitlab.cs.fau.de/flansch/awesome-vezs'. On the left side, there is a dark sidebar with navigation options: Project, Activity, Files, Commits, Graphs, Milestones, Issues (0), Merge Requests (0), Members, Labels, Wiki, Forks, and Settings. The main content area shows the repository name 'Flansch / awesome-vezs' with a search bar and a profile picture 'A'. Below the repository name, there are buttons for 'Clone or download' and 'Fork' (0). The SSH URL 'SSH git@gitlab.cs.fau.de:Flansch/awesome-vezs.g' is highlighted with a red circle. Below the URL, there are statistics: '13 commits', '1 branch', '0 tags', and '0.64 MB', along with buttons for 'Add Changelog' and 'Add License'. A section titled 'Add Contribution guide' is also visible. At the bottom, a commit message '1a8e9a34 _gitignore · 1 hour ago by Flansch' is shown, followed by a message stating 'This project does not have a README yet' and a recommendation to 'add a README' file.

Repository URL mit Gruppe per Mail an Übungsleiter

Repository Einstellungen

Verlässliche Echtzeit x Settings · Flansch / a x wosch.h x FAIL* - Shootout x

← → ↻ https://gitlab.cs.fau.de/flansch/awesome-vezs/edit

GitLab

Go to project

Project Settings

Groups

Deploy Keys

Webhooks

Services

Protected Branches

Flansch

Flansch / awesome-vezs · Settings

This project Search

Project settings

Project name awesome-vezs

Project description (optional)

Default Branch master

Visibility Level

- Private
Project access must be granted explicitly to each user.
- Internal
The project can be cloned by any logged in user.
- Public
The project can be cloned without any authentication.

Tags

Separate tags with commas.

Features:

Privates Repository konfigurieren

Repository Einstellungen – Deploy-Key hinterlegen

Verlässliche Echtzeit | New Deploy Key | wosch.h | FAIL* - Shootout

https://gitlab.cs.fau.de/flansch/awesome-vezs/deploy_keys/new

GitLab

Go to project

- Project Settings
- Groups
- Deploy Keys**
- Webhooks
- Services
- Protected Branches

Flansch

Flansch / awesome-vezs · Settings

This project Search

New Deploy Key

Title

Key Paste a machine public key here. Read more about how to generate it [here](#)

```
UBYHxVcWZuWcRIUqdPIKEOfj0JjcN1/qS5fkDc3TuoqeV/C1/lbZldwCuwC1VgLw1k7Tvahwc4
yk4fJjKbeGN32fg5cGDmxZCUMFsUnyxISZ/U4dJXO0Tqg/ResNj48WpCEy6sw88E5+FmKbT
35hkvNAUikq+oy4xv8RBwWMSQUmV+pqsvE4SjUinRRLh5LBJxhh9fkUB6QriU7X/UIOq7+hN
DrrPmDx7qdkrHIU76r4oK6m4V3E18s5JWD2AoZlwtCAYCITu1m45JgMoQqGk3I75rTxylO9L
8tgGLTw9 Deploykey for i4vezs
```



Repository Einstellungen – Überprüfung Deploy-Key

The screenshot shows the GitLab web interface for the 'Flansch / awesome-vezs' repository settings. The left sidebar contains navigation options: 'Go to project', 'Project Settings', 'Groups', 'Deploy Keys' (selected), 'Webhooks', 'Services', and 'Protected Branches'. The main content area is titled 'Deploy keys allow read-only access to the repository' and includes a '+ New Deploy Key' button. Below this, it states 'Deploy keys can be used for CI, staging or production servers. You can create a deploy key or add an existing one'. A section titled 'Enabled deploy keys for this project' lists one key: 'i4ezs-deploy' with a 'Remove' button and a long alphanumeric key ID. Below the key ID is a link to 'Flansch / awesome-vezs' and the text 'Created 8 minutes ago'. To the right, a section titled 'Deploy keys from projects you have access to' contains a placeholder box stating 'Deploy keys from projects you have access to will be displayed here'.





Forin.

Vital coded microprocessor principles and application for various transit systems.

IFA-GCCT, pages 79–84, 1989.

