

Verlässliche Echtzeitsysteme

Testen Statistischer Analyse

Peter Wägemann

Lehrstuhl für Verteilte Systeme und Betriebssysteme

Friedrich-Alexander-Universität Erlangen-Nürnberg

<https://sys.cs.fau.de>

KW04 2023



Platform Software for Safety-Critical Multicore Systems

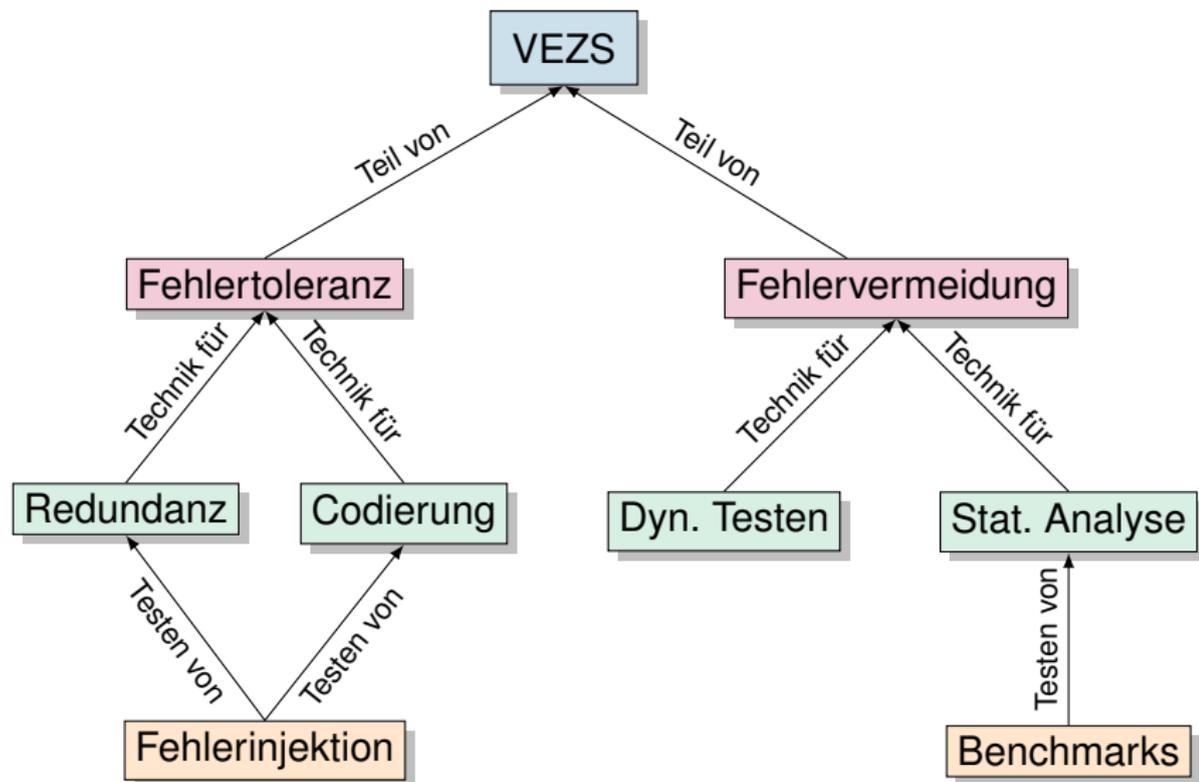
Echtzeitsysteme 09.01.2023

Dr. Isabella Stilkerich

SCHAEFFLER

- https://sys.cs.fau.de/extern/lehre/ws22/vezs/Skript/20230109_VEZS-Embedded.pdf





- Testen von Härtungsmechanismen durch Strahl von Neutronen
- Los Alamos Neutron Science Center (LANSCE)¹
- Santini et al.: *Effectiveness of Software-Based Hardening for Radiation-Induced Soft Errors in Real-Time Operating Systems* [2]
- https://www4.cs.fau.de/Publications/2017/santini_17_arcs.pdf

¹<https://lansce.lanl.gov/>

```
@inproceedings{kashyap:2019:sca,  
  title={Automated Customized bug-benchmark generation},  
  author={Kashyap, Vineeth and Ruchti, Jason and Kot, Lucja  
    and Turetsky, Emma and Swords, Rebecca and Pan, Shih An  
    and Henry, Julien and Melski, David and Schulte, Eric},  
  booktitle={Proceedings of the 19th International Working  
    Conference on Source Code Analysis and Manipulation  
    (SCAM '19)},  
  pages={103--114},  
  year={2019},  
}
```

- Pre-Print verfügbar: <https://arxiv.org/pdf/1901.02819.pdf>



- Frage nach Präzision von WCET-Analyse?
- *Wie nahe ist abstrakte Semantik an konkreter Semantik?*
- WOET: Unterabschätzung
- obere Schranke: Überabschätzung
- https://www4.cs.fau.de/Publications/2017/waegemann_17_rtas.pdf

- [1] Kashyap, V. ; Ruchti, J. ; Kot, L. ; Turetsky, E. ; Swords, R. ; Pan, S. A. ; Henry, J. ; Melski, D. ; Schulte, E. :
Automated customized bug-benchmark generation.
In: *Proceedings of the 19th International Working Conference on Source Code Analysis and Manipulation (SCAM '19)*, 2019, S. 103–114
- [2] Santini, T. ; Borchert, C. ; Dietrich, C. ; Schirmeier, H. ; Hoffmann, M. ; Spinczyk, O. ; Lohmann, D. ; Wagner, F. R. ; Rech, P. :
Effectiveness of software-based hardening for radiation-induced soft errors in real-time operating systems.
In: *Proceedings of the International Conference on Architecture of Computing Systems (ARCS '17)* Springer, 2017, S. 3–15
- [3] Wägemann, P. ; Distler, T. ; Eichler, C. ; Schröder-Preikschat, W. :
Benchmark Generation for Timing Analysis.
In: *Proceedings of the 23rd Real-Time and Embedded Technology and Applications Symposium (RTAS '17)*, 319–330

```
const size_t N = 1024;
int32_t values[N];

TASK( $\tau_{\text{sort}}$ ){
    ...
    for(i = 0; i < N-1; i++){
        for(j = 0; j < N-1-i; j++){
            if(compare(values[j], values[j+1]))
                swap(&values[j], &values[j+1]);
        }
    }
}
```

```
const size_t N = 1024;
int32_t values[N];

TASK( $\tau_{\text{sort}}$ ){
    ...
    for(i = 0; i < N-1; i++){
        for(j = 0; j < N-1-i; j++){
            if(compare(values[j], values[j+1]))
                swap(&values[j], &values[j+1]);
        }
    }
}
```

- Verschachtelte Schleifen

```
const size_t N = 1024;
int32_t values[N];

TASK( $\tau_{\text{sort}}$ ){
    ...
    for(i = 0; i < N-1; i++){
        for(j = 0; j < N-1-i; j++){
            if(compare(values[j], values[j+1]))
                swap(&values[j], &values[j+1]);
        }
    }
}
```

- Verschachtelte Schleifen
- Eingabeabhängige Pfade

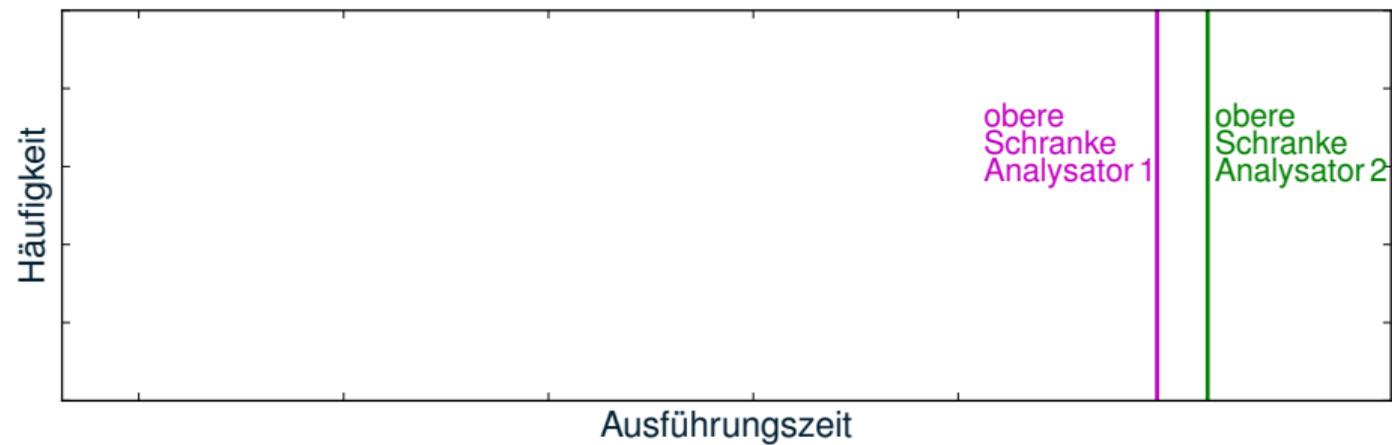
```
const size_t N = 1024;
int32_t values[N];

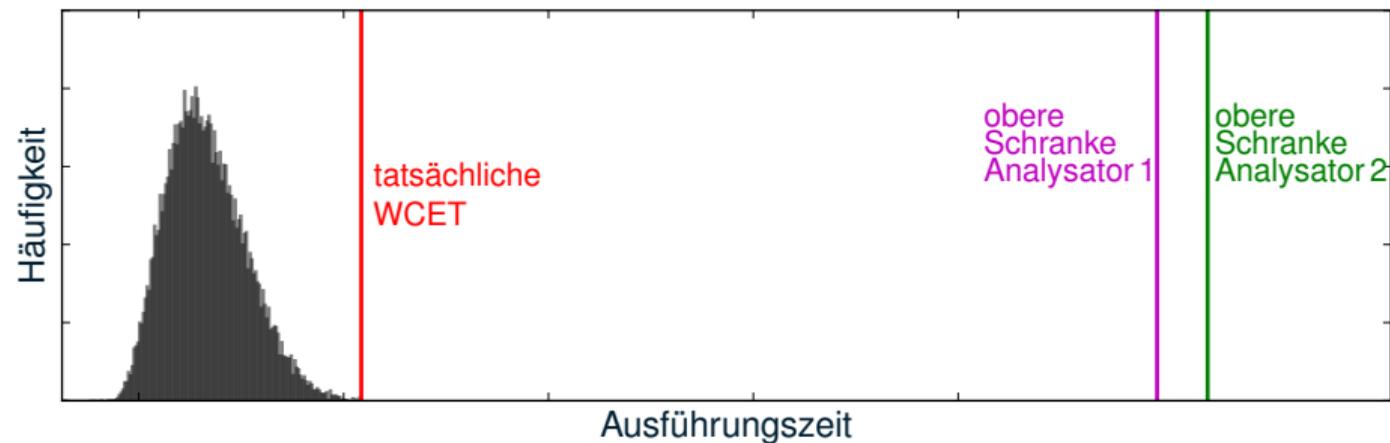
TASK( $\tau_{\text{sort}}$ ){
    ...
    for(i = 0; i < N-1; i++){
        for(j = 0; j < N-1-i; j++){
            if(compare(values[j], values[j+1]))
                swap(&values[j], &values[j+1]);
        }
    }
}
```

- Verschachtelte Schleifen
- Eingabeabhängige Pfade
- Hardware-Modellierung









- Bestimmung Genauigkeit der Analyse: **tatsächlicher Worst-Case** essenziell

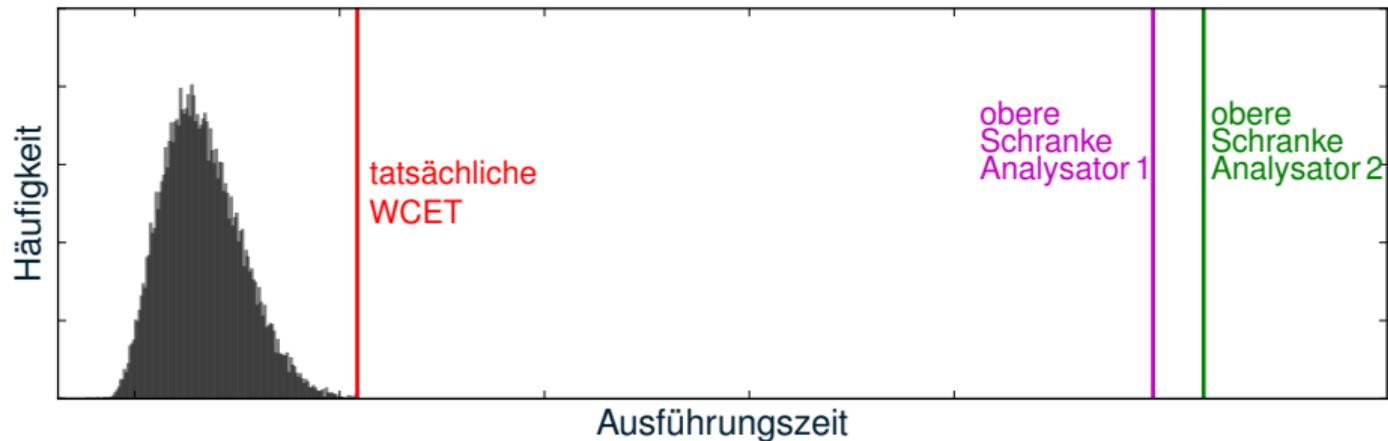


- Bestimmung Genauigkeit der Analyse: **tatsächlicher Worst-Case** essenziell
- **Existierende Benchmarks**: automatische Ermittlung des Worst-Cases nicht möglich

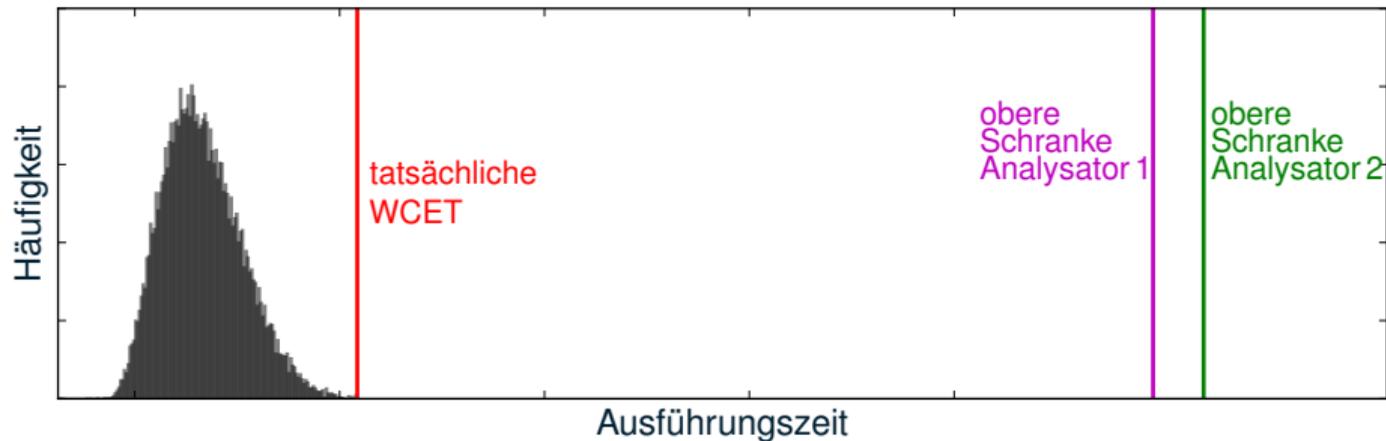


- Bestimmung Genauigkeit der Analyse: **tatsächlicher Worst-Case** essenziell
- **Existierende Benchmarks**: automatische Ermittlung des Worst-Cases nicht möglich

👉 **Evaluation & Validierung**: Wissen über Worst-Case notwendig

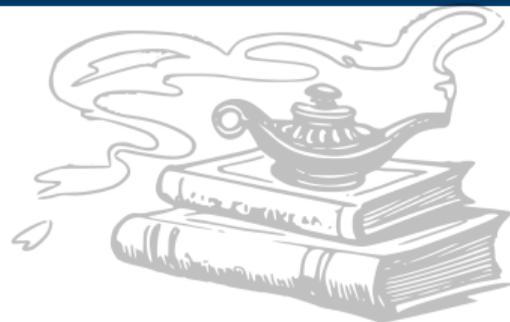
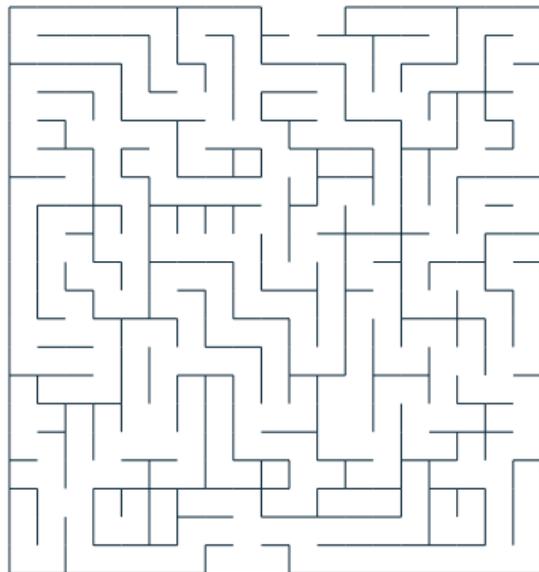


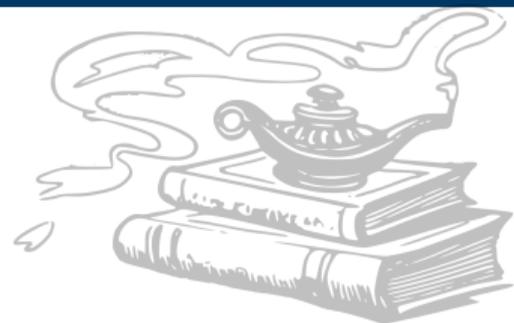
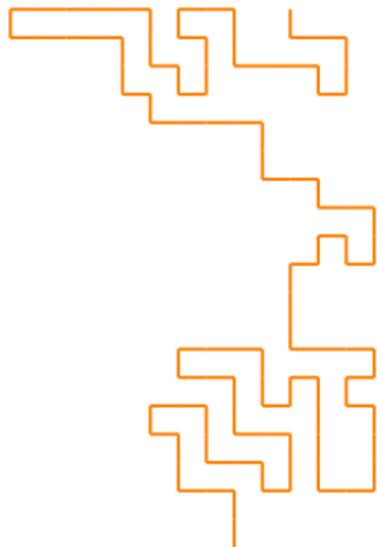
Wie lässt sich der tatsächliche Worst-Case ermitteln?

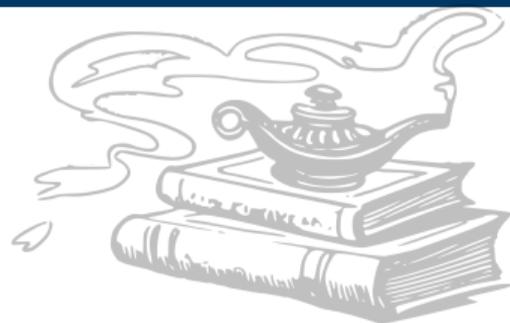
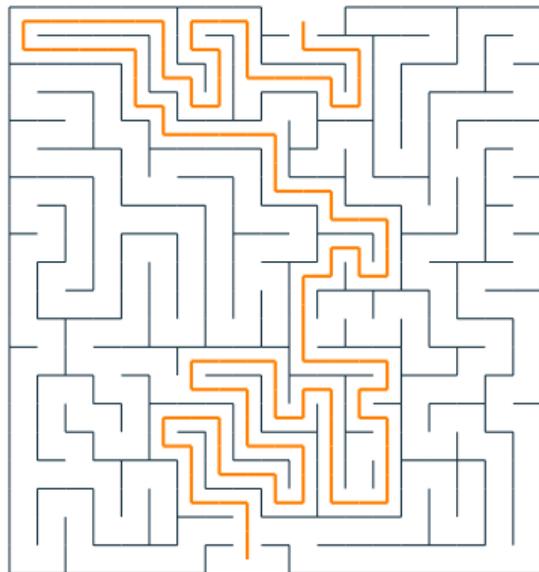


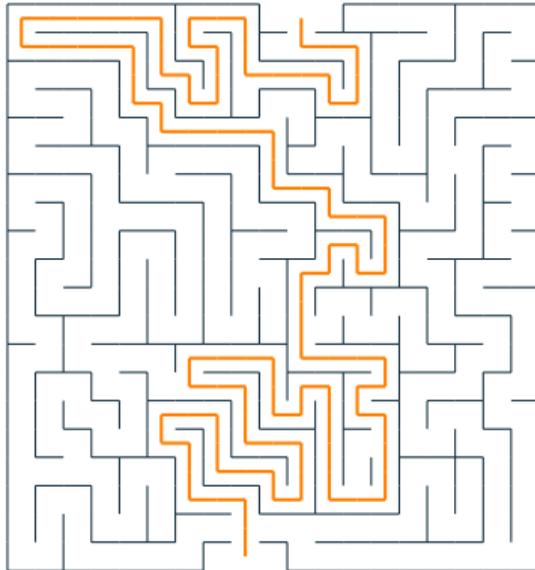
Wie lässt sich der tatsächliche Worst-Case ermitteln?

- ☞ *GenE*: generiert Benchmarks, deren Programmfakten bekannt sind

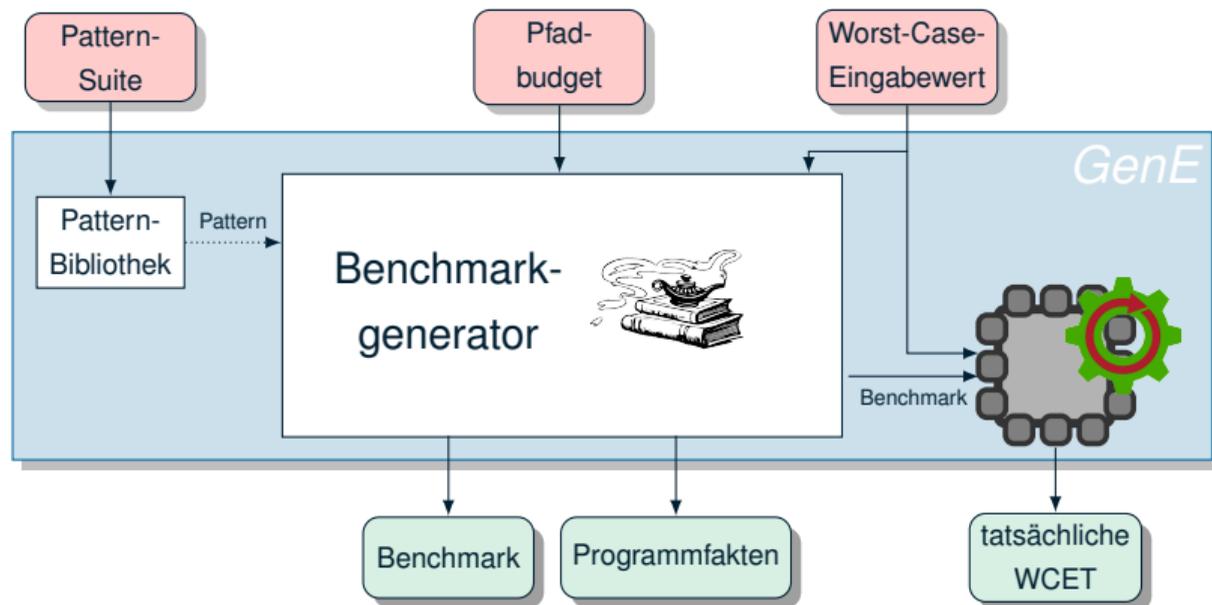


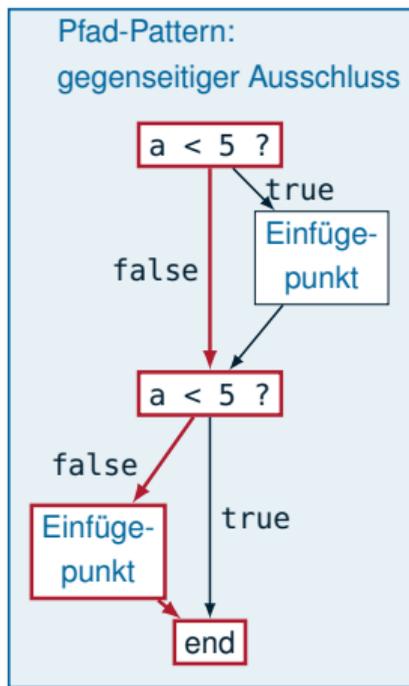






👉 Generativer Ansatz: **Worst-Case-Pfad** bekannt





Pattern

- **Enthalten Programmfakten**
- **Kombinierbar**: ineinander verwoben
- Pfad-Pattern, Schleifen, arithmetische Operationen, Variablen, Funktionsaufrufe, ...
- **Realistisch & herausfordernd**
 - Industrie-Anwendungen
 - Literatur
- **Pattern-Suites** verhindern monolithische Benchmarks

```
store volatile i32 %arith8619, i32* @cons0
%10 = load i32* @gene_glob_input_variable
ret i32 %10

br434.then.wcp: ; preds = %br433.then.wcp
call void @iofunc71()
%arith7873 = or i32 %4, 1239184893
%arith7874 = sdiv i32 %arith7873, #779274285
store volatile i32 %arith7874, i32* @cons0
store volatile i32 %arith7041, i32* @cons0
%arith7875 = sdiv i32 %arith7873, 477723546
%arith7876 = add i32 %arith7875, 212494280
call void @iofunc82()
%11 = load i32* @gene_glob_input_variable
%12 = and i32 %11, 4096
%13 = icmp ne i32 %12, 0
br i1 %13, label %br489.then.wcp, label %br489.else

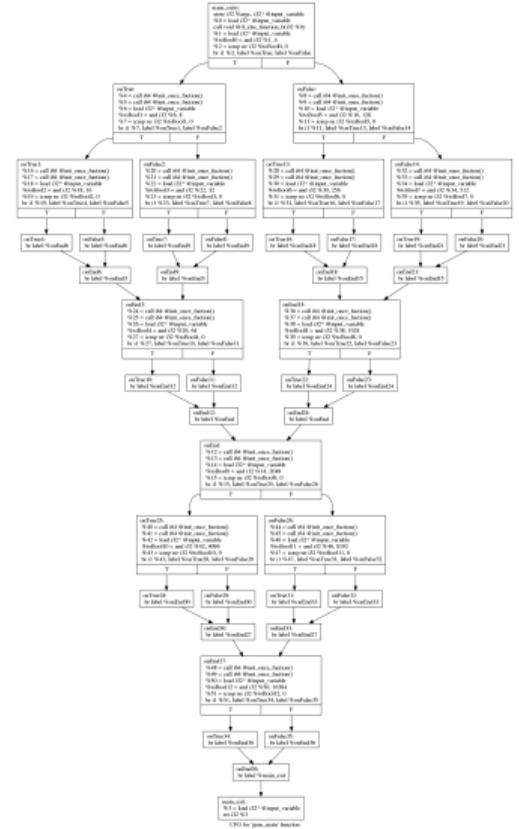
br434.else: ; preds = %br433.then.wcp
%arith8206 = mul i32 %arith8205, 2039652656
%14 = load i32* @gene_glob_input_variable
%15 = and i32 %14, 64
%16 = icmp ne i32 %15, 0
br i1 %16, label %br491.then, label %br491.else

br434.end.wcp: ; preds = %br491.end, %br490.end.wcp
%arith8222 = sdiv i32 %arith7038, 2108165700
call void @iofunc87()
%arith8416 = sdiv i32 %arith8222, 1909311980
%17 = load i32* @gene_glob_input_variable
%18 = and i32 %17, 16
%19 = icmp ne i32 %18, 0
br i1 %19, label %br500.then.wcp, label %br500.else

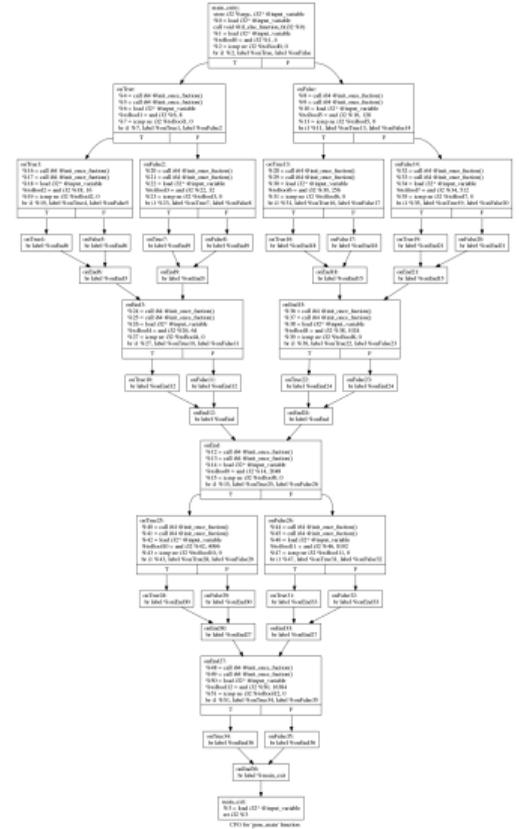
br489.then.wcp: ; preds = %br434.then.wcp
%arith8151 = or i32 %arith7876, 110525940
store volatile i32 %arith8151, i32* @cons0
%arith8153 = xor i32 %arith8152, 938560787
store volatile i32 %arith8160, i32* @cons0
%arith8163 = sdiv i32 %arith8162, 940335150
br label %br489.end.wcp
```

- Vermeidung von **monolithischen Benchmarks**
- Beachtung der Analysephase: **Pfad- & Hardware-Analyse**
- Benchmarks mit **spezifischen Eigenschaften**
 1. Arithmetische Operationen, Zuweisungen, Verzweigungen:

👉 GenE --suite value_analysis

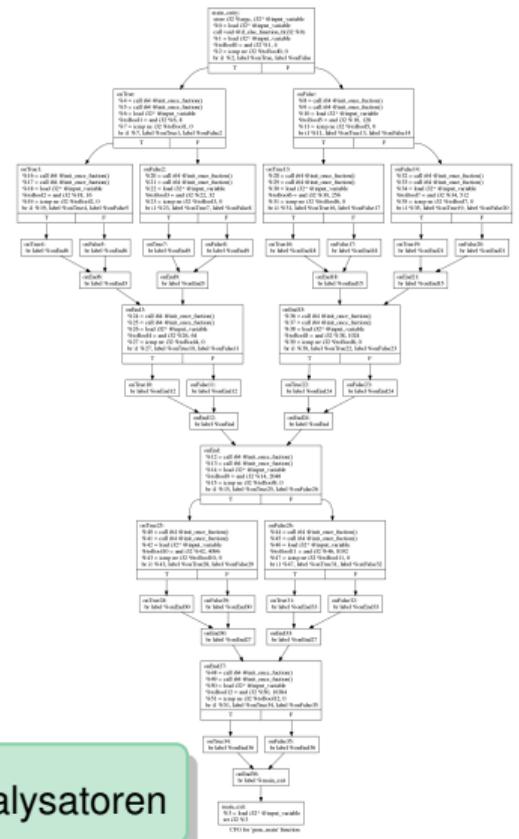


- Vermeidung von **monolithischen Benchmarks**
- Beachtung der Analysephase: **Pfad- & Hardware-Analyse**
- Benchmarks mit **spezifischen Eigenschaften**
 1. Arithmetische Operationen, Zuweisungen, Verzweigungen:
👉 GenE `--suite value_analysis`
 2. Kein Pessimismus durch Schleifen & Pfade:
👉 GenE `--suite single_path`
- Benchmarks **maßgeschneidert für Analyseszenario**



- Vermeidung von **monolithischen Benchmarks**
- Beachtung der Analysephase: **Pfad- & Hardware-Analyse**
- Benchmarks mit **spezifischen Eigenschaften**
 1. Arithmetische Operationen, Zuweisungen, Verzweigungen:
👉 GenE `--suite value_analysis`
 2. Kein Pessimismus durch Schleifen & Pfade:
👉 GenE `--suite single_path`
 - ...
- Benchmarks **maßgeschneidert für Analyseszenario**

✓ Bestimmung **Stärken & Schwächen** von Analysatoren



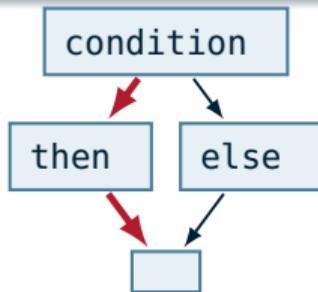
- Anzahl von **Instruktionen entlang Worst-Case-Pfad** (z.B. 10 000)

- Anzahl von **Instruktionen entlang Worst-Case-Pfad** (z.B. 10 000)

Problem: variierende Instruktionslaufzeiten

1. Caches
2. Pipelining
- ...

☞ *Notwendigkeit:* Modellierung **Hardwareverhalten**



- Anzahl von **Instruktionen entlang Worst-Case-Pfad** (z.B. 10 000)

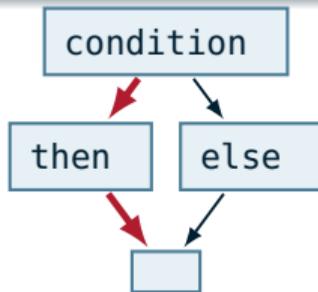
Problem: variierende Instruktionslaufzeiten

1. Caches
2. Pipelining
- ...

👉 *Notwendigkeit:* Modellierung **Hardwareverhalten**

■ Ziele

1. Keine Mikroarchitekturmodellierung während Benchmark-Generierung
2. Worst-Case-Eingabewert führt zu Worst-Case-Pfad

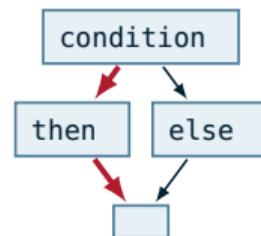


- Anforderung an GenE:

Worst-Case(nicht-Worst-Case-Pfade) \leq Best-Case(Worst-Case-Pfad)

- Laufzeit von Instruktion:

$$t_{instr} = t_{icache} + t_{dcache} + t_{pipeline} + t_{calc}$$

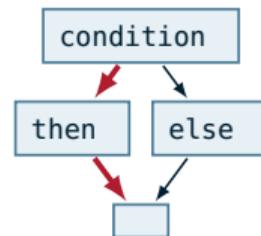


- Anforderung an GenE:

Worst-Case(nicht-Worst-Case-Pfade) ≤ Best-Case(Worst-Case-Pfad)

- Laufzeit von Instruktion:

$$t_{instr} = t_{icache} + t_{dcache} + t_{pipeline} + t_{calc}$$



Lösung: maximale Laufzeitunterschiede

- Finde **Maxima der Worst-Case-Laufzeiten** aller Instruktionen
- Beispiel (ARM Cortex-M4)

`add r4, r5`

$$t_{add,min} = 1$$

`ldr r0, [r0, 42]`

$$t_{ldr,max} = 7$$

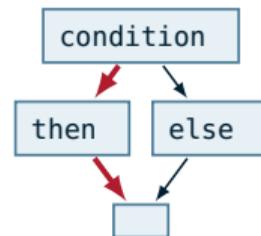
- 7 x `add` kompensieren 1 x `ldr` im Worst-Case
- ☞ Übergewichtungsfaktor: $\mathcal{F} = 7$

- Anforderung an GenE:

Worst-Case(nicht-Worst-Case-Pfade) \leq Best-Case(Worst-Case-Pfad)

- Laufzeit von Instruktion:

$$t_{instr} = t_{icache} + t_{dcache} + t_{pipeline} + t_{calc}$$



Lösung: maximale Laufzeitunterschiede

- Finde **Maxima der Worst-Case-Laufzeiten** aller Instruktionen
- Beispiel (ARM Cortex-M4)

`add r4, r5`

$$t_{add,min} = 1$$

`ldr r0, [r0, 42]`

$$t_{ldr,max} = 7$$

- 7 x `add` kompensieren 1 x `ldr` im Worst-Case
- ☞ Übergewichtungsfaktor: $\mathcal{F} = 7$

☞ Wähle \mathcal{F} **ausreichend groß**: $\mathcal{F} = 25$

Worst-Case-Eingabewert:

·	·	0	1
---	---	---	---

Übergewichtungsfaktor: $\mathcal{F} = 25$

Worst-Case-Eingabewert:

·	·	0	1
---	---	---	---

Budget: 1010

Übergewichtungsfaktor: $\mathcal{F} = 25$

Worst-Case-Eingabewert:

..	0	1
----	---	---

Übergewichtungsfaktor: $\mathcal{F} = 25$

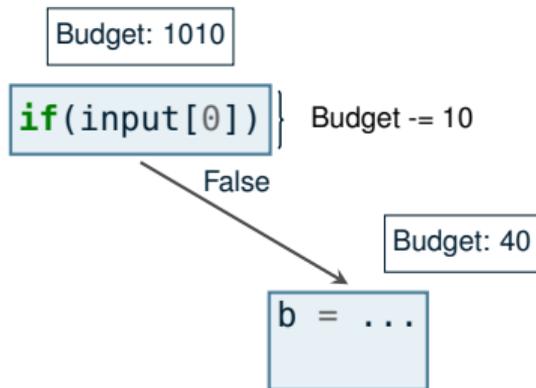
Budget: 1010

`if(input[0])` } Budget -= 10

Worst-Case-Eingabewert:

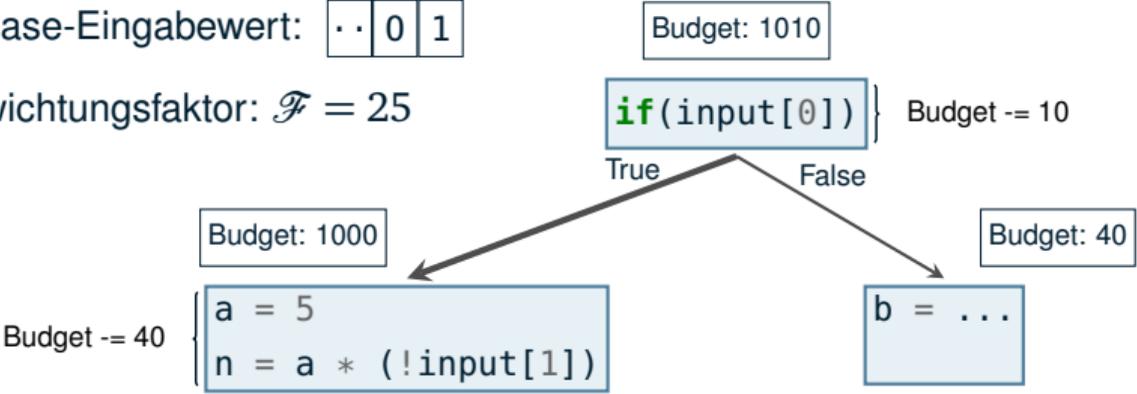
..	0	1
----	---	---

Übergewichtungsfaktor: $\mathcal{F} = 25$



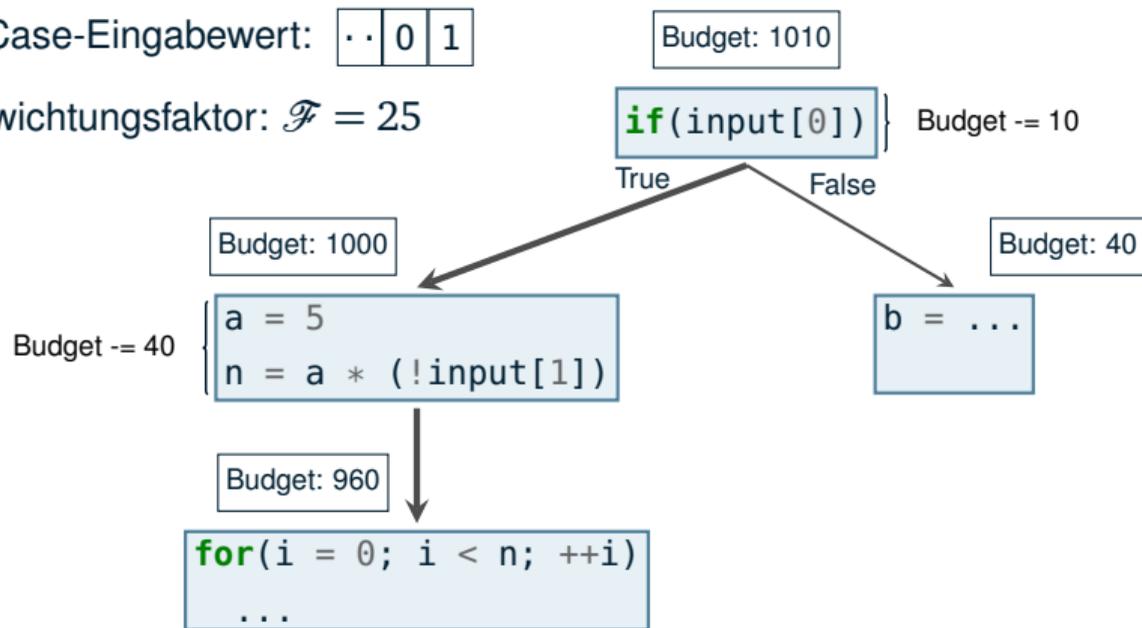
Worst-Case-Eingabewert: `..01`

Übergewichtungsfaktor: $\mathcal{F} = 25$



Worst-Case-Eingabewert: `..01`

Übergewichtungsfaktor: $\mathcal{F} = 25$



Evaluation: Stärken & Schwächen

	constant loop	input-dependent loop	down-sampling loop	triangular loop
 aiT	✓	✓	✓	✗
 Platin	✓	✓	✗	✓

- Modulare Struktur von Pattern-Suites \rightsquigarrow verhindert monolithische Benchmarks
- Benchmarks mit **einem herausfordernden Pattern**
 - Variable, arithmetische Operationen, Zuweisungen
 - Herausforderndes Schleifen-Pattern

✓ **GenE deckt Stärken & Schwächen auf**

- WCET-Analysator: aiT von AbsInt
- Hardware: ARM Cortex-M4
- *GenE*: 10 000 generierte Benchmarks
- Optimum: 0% Überabschätzung



- WCET-Analysator: aiT von AbsInt
- Hardware: ARM Cortex-M4
- *GenE*: 10 000 generierte Benchmarks
- Optimum: 0% Überabschätzung



Platin

Instruktionscache

- **Deaktiviert: 96%**
- **Aktiviert: –**



- WCET-Analysator: aiT von AbsInt
- Hardware: ARM Cortex-M4
- *GenE*: 10 000 generierte Benchmarks
- Optimum: 0% Überabschätzung



Platin

Instruktionscache

- **Deaktiviert: 96%**
- **Aktiviert: –**

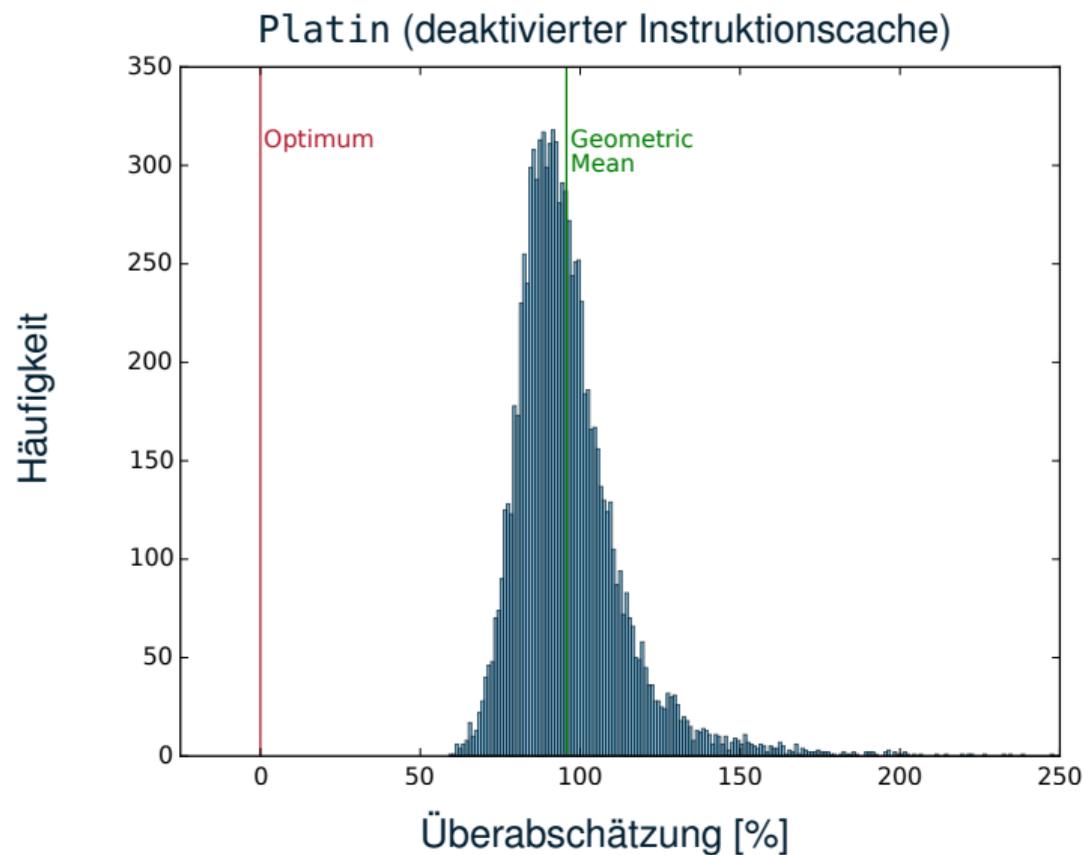


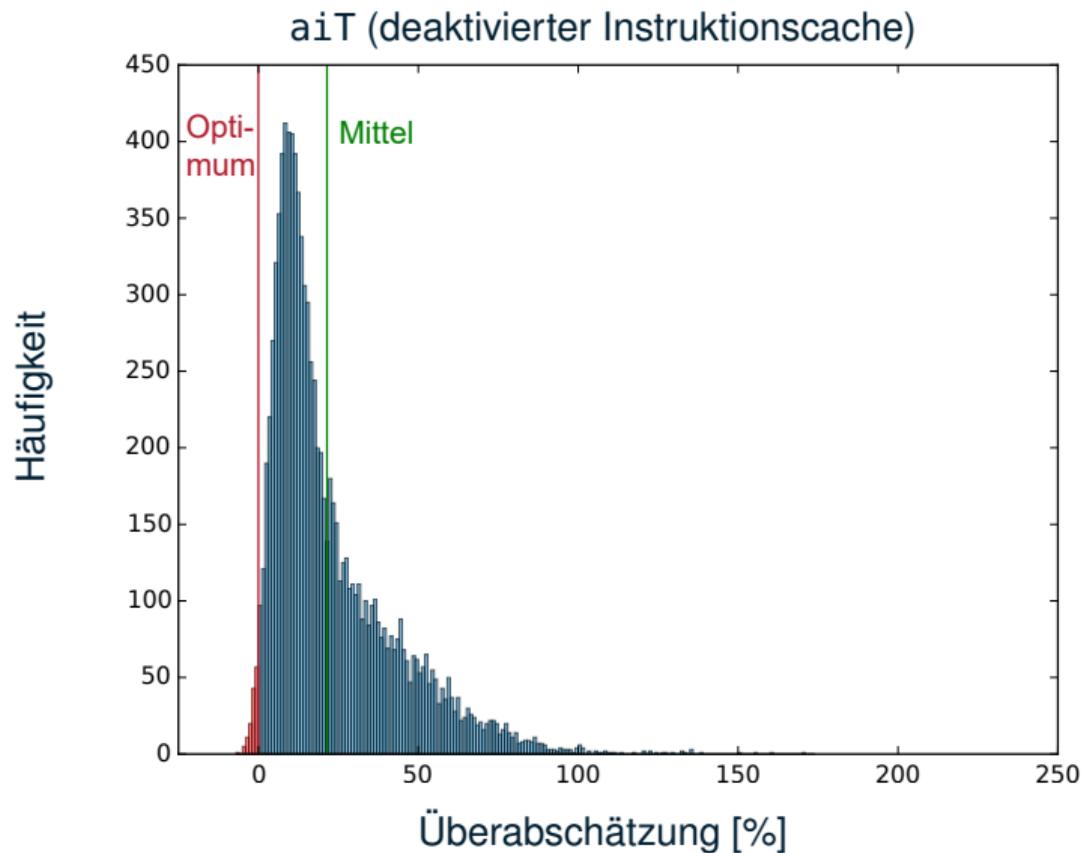
Überabschätzung aiT

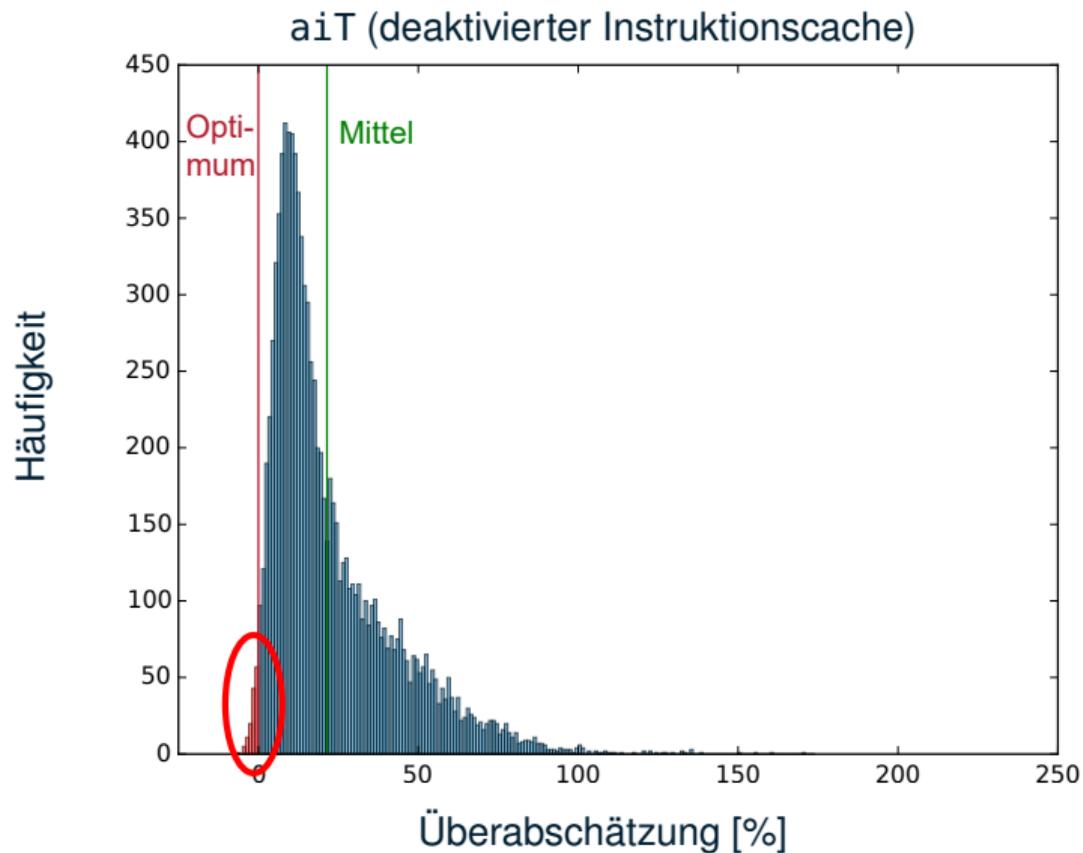
Instruktionscache

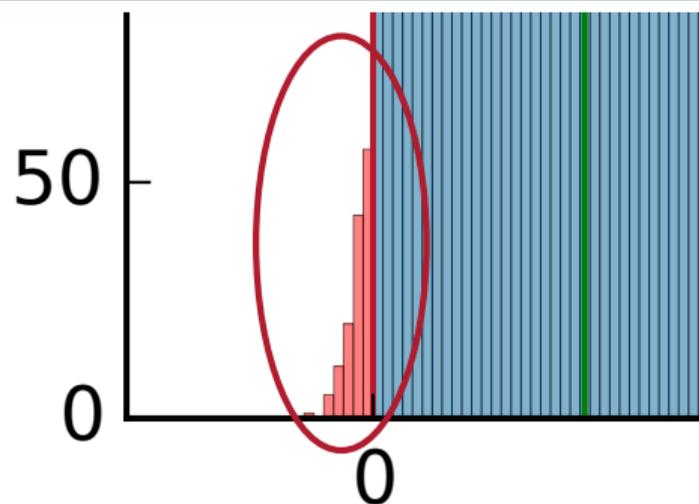
- **Deaktiviert: 23%**
- **Aktiviert: 36%**





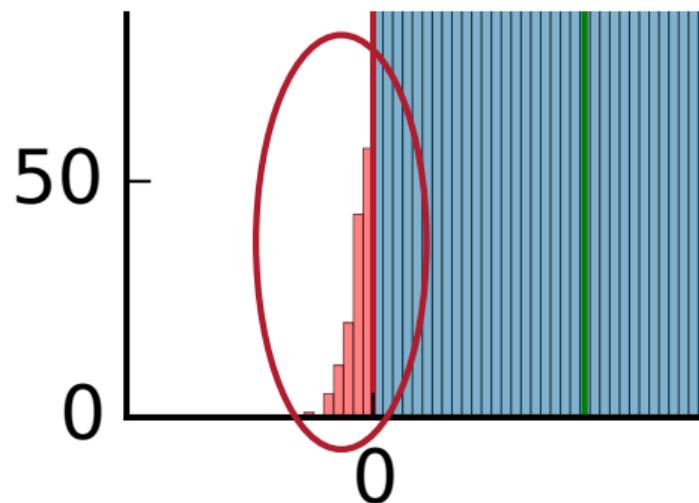






- Beobachtung: **Unterabschätzungen** der tatsächlichen WCET
- *AbsInt*: **Fehler in WCET-Analysator** entdeckt durch Benchmarks von *GenE*
 1. Pipeline-Modell: spekulative Sprungvorhersage
 2. Speicher-Modell: fehlerhafte Zugriffsdauer

Evaluation: Bugs in WCET-Analysator aiT



- Beobachtung: **Unterabschätzungen** der tatsächlichen WCET
- *AbsInt*: **Fehler in WCET-Analysator** entdeckt durch Benchmarks von *GenE*
 1. Pipeline-Modell: spekulative Sprungvorhersage
 2. Speicher-Modell: fehlerhafte Zugriffsdauer

✓ *GenE*: Bestimmung Genauigkeit & **Unterabschätzungen aufgedeckt**