ISE SRA

Leibniz Universität Hannover

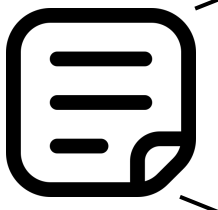
Hardening Rust: Erweiterung des Compilers und der Laufzeitumgebung zur Unterstützung dynamischer Objekt-Layout-Randomisierung

FGBS/2024 Spring

Jan Neugebauer

14.03.24

Angriffsszenario



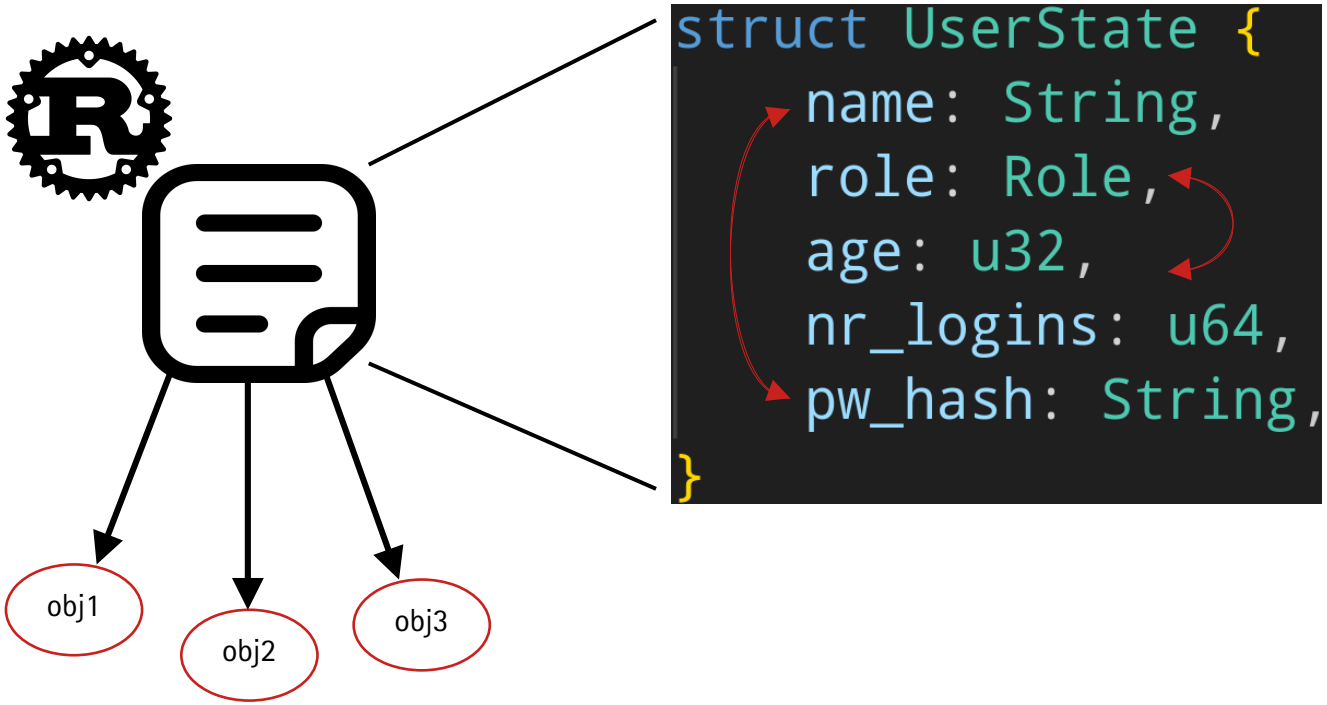
```
enum Role {
    Customer,
    Admin,
};

struct UserState {
    char* name;
    Role role;
};
```

```
UserState {
    name: "bob",
    role: Role::Customer,
};
```

```
UserState {
    name: "bob",
    role: Role::Admin,
};
```

List icons created by Freepik – Flaticon
Computer icons created by Freepik – Flaticon

Layout-Randomisierung zur Laufzeit



Inhalt

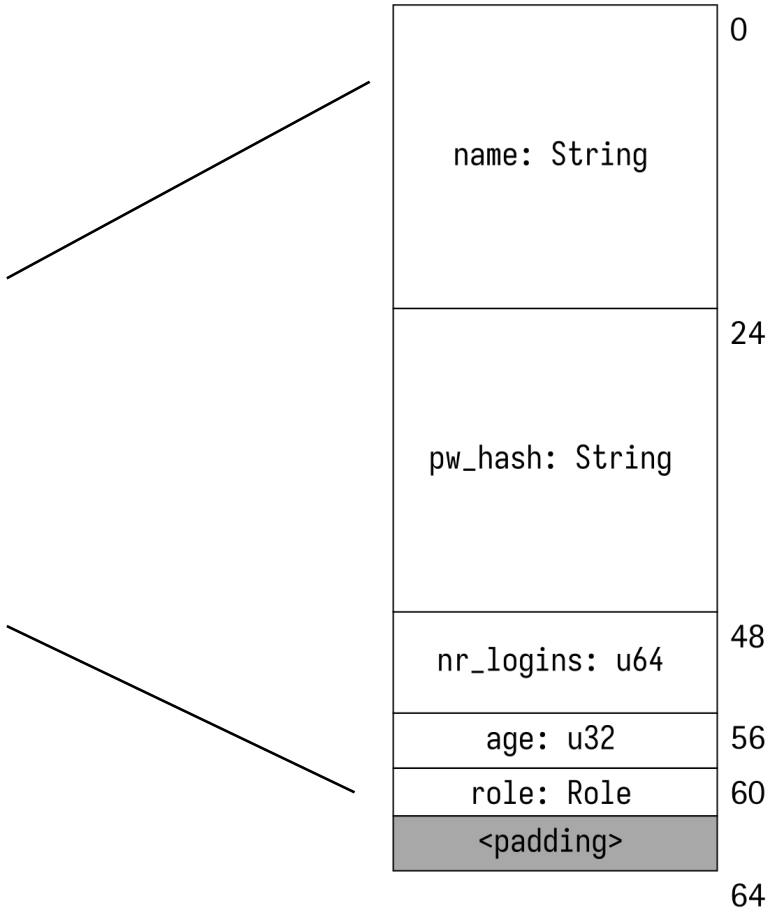
- Struct/Objekt-Layout-Randomisierung
  - Was gibt es bereits?
- Architektur und Implementierung
- Evaluation
  - Kompatibilität
  - Performanz

Inhalt

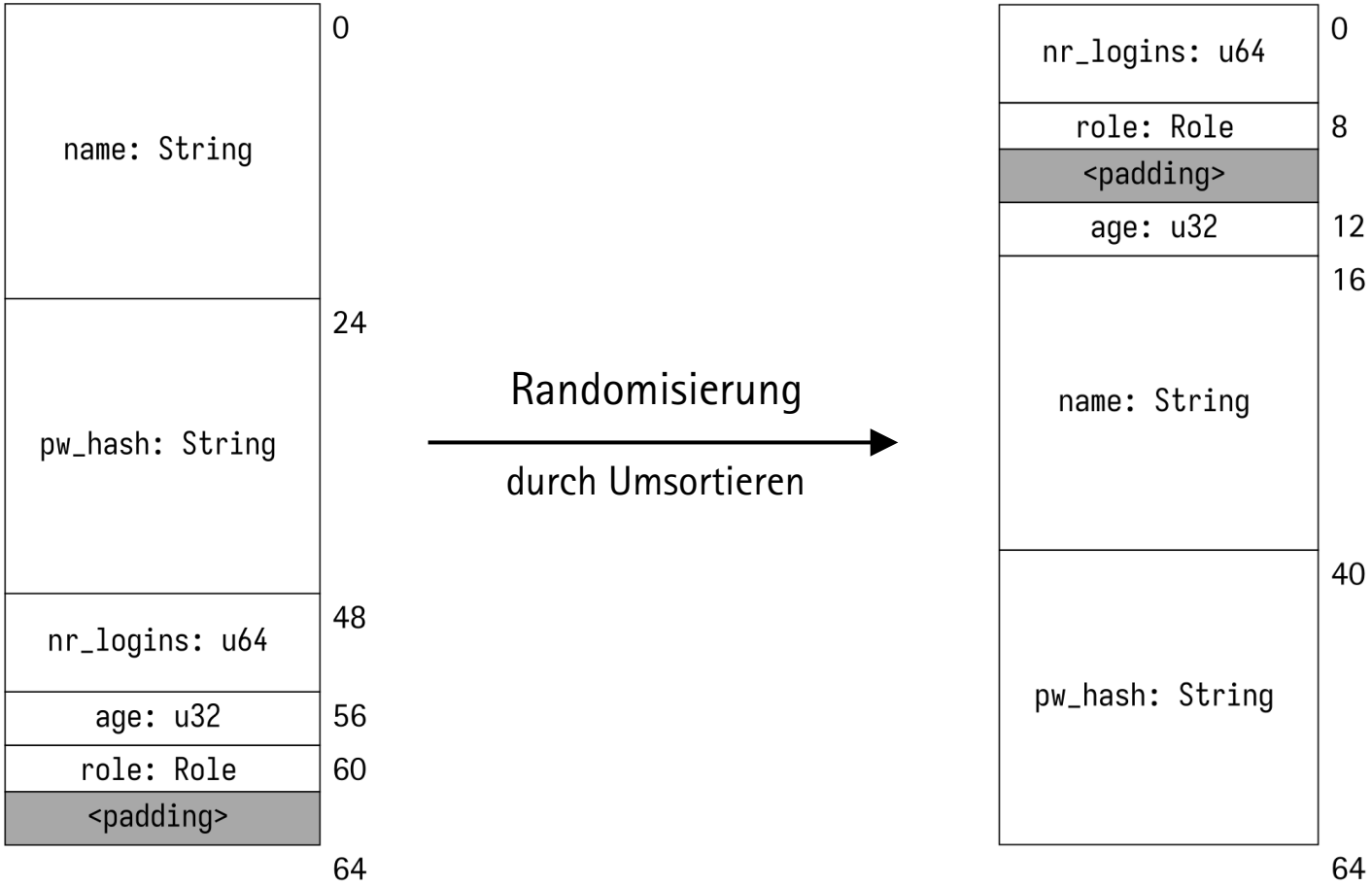
- Struct/Objekt-Layout-Randomisierung
  - Was gibt es bereits?
- Architektur und Implementierung
- Evaluation
  - Kompatibilität
  - Performanz

Struct-Layout

```
struct UserState {
    name: String,
    role: Role,
    age: u32,
    nr_logins: u64,
    pw_hash: String,
}
```



Struct-Layout-Randomisierung



Was gibt es bereits?

Zur Compile-Zeit

Struct

2009:
Polymorphing Software by Randomizing Data Structure Layout.

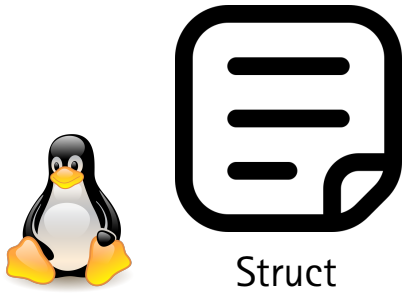
Zhiqiang Lin, Ryan D. Riley, Dongyan Xu

2013:
Improved Kernel Security Through Memory Layout Randomization

Dannie M. Stanley, Dongyan Xu, Eugene H. Spafford

Was gibt es bereits?

Zur Compile-Zeit



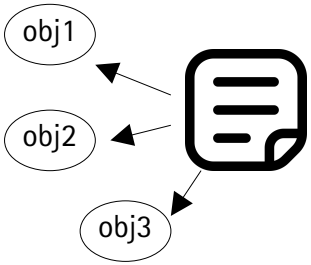
2009:
Polymorphing Software by Randomizing Data Structure Layout.

Zhiqiang Lin, Ryan D. Riley, Dongyan Xu

2013:
Improved Kernel Security Through Memory Layout Randomization

Dannie M. Stanley, Dongyan Xu, Eugene H. Spafford

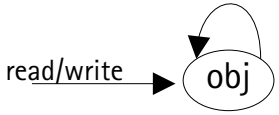
Runtime:
Bei Instanziierung



2019:
POLaR: Per-allocation Object Layout Randomization

Jonghwan Kim, Daehee Jang, Yunjong Jeong, Brent Byunghoon Kang

Runtime:
Nach X Zugriffen rerandomisieren



2015:
A Practical Approach for Adaptive Data Structure Layout Randomization

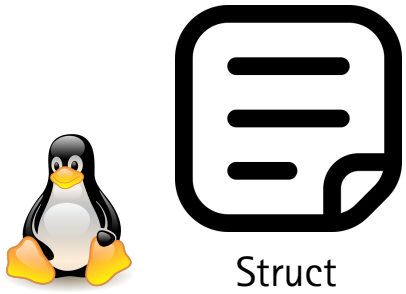
Ping Chen, Jun Xu, Zhiqiang Lin, Dongyan Xu, Bing Mao, Peng Liu

2018:
Feedback control can make data structure layout randomization more cost-effective under zero-day attacks

Ping Chen, Zhisheng Hu, Jun Xu, Minghui Zhu, Peng Liu

Was gibt es bereits?

Zur Compile-Zeit



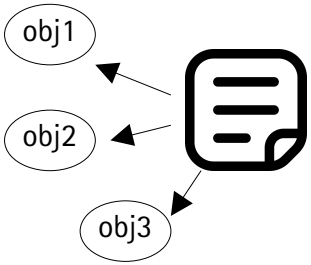
2009:
Polymorphing Software by Randomizing Data Structure Layout.

Zhiqiang Lin, Ryan D. Riley, Dongyan Xu

2013:
Improved Kernel Security Through Memory Layout Randomization

Dannie M. Stanley, Dongyan Xu, Eugene H. Spafford

Runtime:
Bei Instanziierung



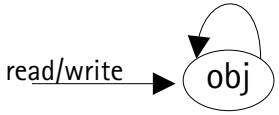
2019:
POLaR: Per-allocation Object Layout Randomization

Jonghwan Kim, Daehee Jang, Yunjong Jeong, Brent Byunghoon Kang

2024:

Diese Arbeit

Runtime:
Nach X Zugriffen rerandomisieren



2015:
A Practical Approach for Adaptive Data Structure Layout Randomization

Ping Chen, Jun Xu, Zhiqiang Lin, Dongyan Xu, Bing Mao, Peng Liu

2018:
Feedback control can make data structure layout randomization more cost-effective under zero-day attacks

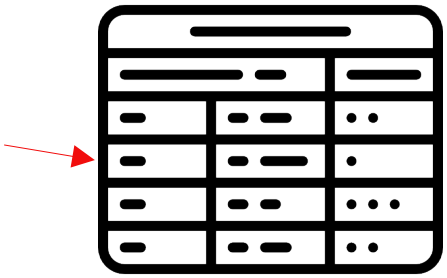
Ping Chen, Zhisheng Hu, Jun Xu, Minghui Zhu, Peng Liu

Bisherige Arbeiten

C/C++

0x4200	8	4	...
0xff30	4	1	...
0x3553	16	2	...
0x1111	8	2	...
0x2222	4	8	...
0x3333	2	4	...
0x4444	1	2	...
0x5555	8	16	...

Komplexe Buchführung aller Instanzen

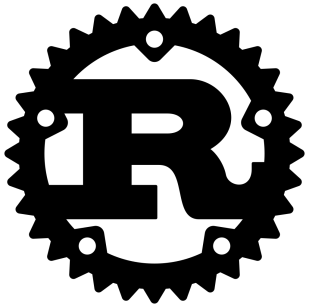


Ungeschützte Metadaten

Scheme icons created by Freepik - Flaticon

Was ist Neu?

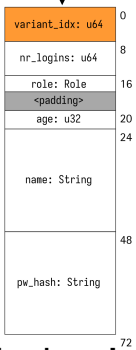
C/C++



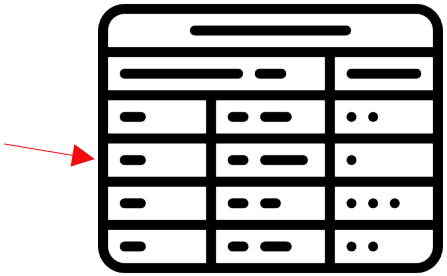
Rust

0x4200	8	4	...
0xff30	4	1	...
0x3553	16	2	...
0x1111	8	2	...
0x2222	4	8	...
0x3333	2	4	...
0x4444	1	2	...
0x5555	8	16	...

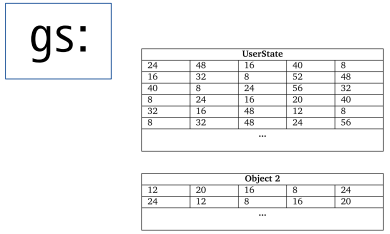
Komplexe Buchführung aller Instanzen



Variantenindex im Layout



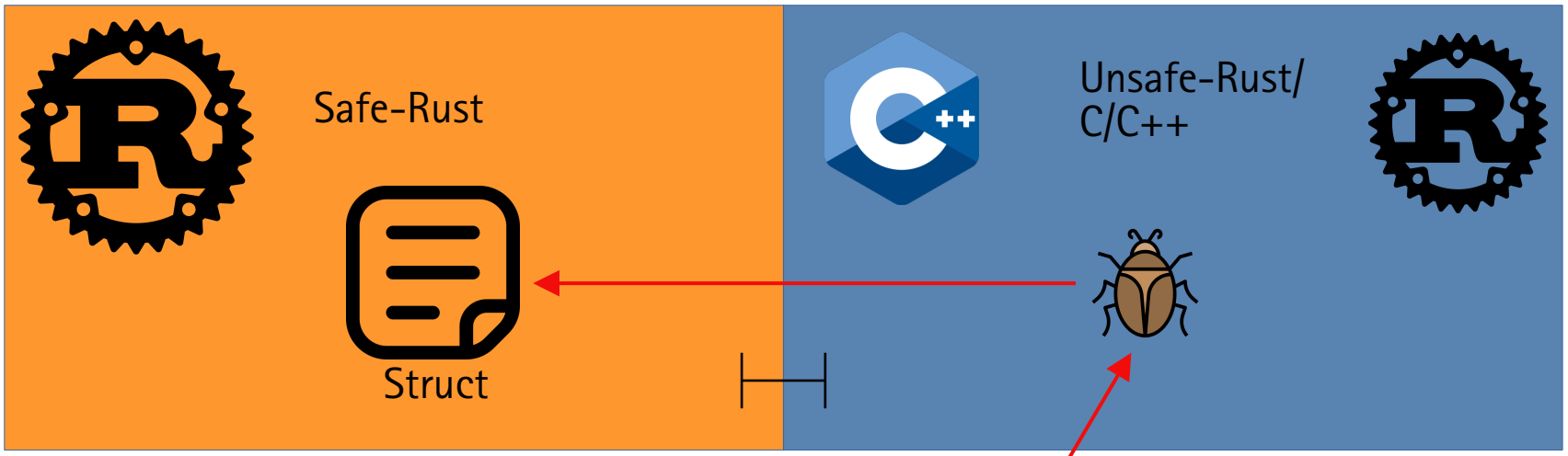
Ungeschützte Metadaten



Schutz über Information-Hiding

Scheme icons created by Freepik - Flaticon

Mixed-Binaries (Rust/C++)



[1]

2021: Michalis Papaevripides, ...
[Exploiting Mixed Binaries]

2022: Samuel Mergendahl, ...
[Cross-Language Attacks]

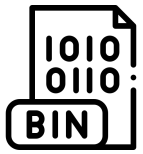
[1] is a trademark of the Mozilla Foundation in the US and other countries.

Bug icons created by surang - Flaticon
Computer icons created by Freepik – Flaticon

Inhalt

- Struct/Objekt-Layout-Randomisierung
  - Was gibt es bereits?
- Architektur und Implementierung
- Evaluation
  - Kompatibilität
  - Performanz

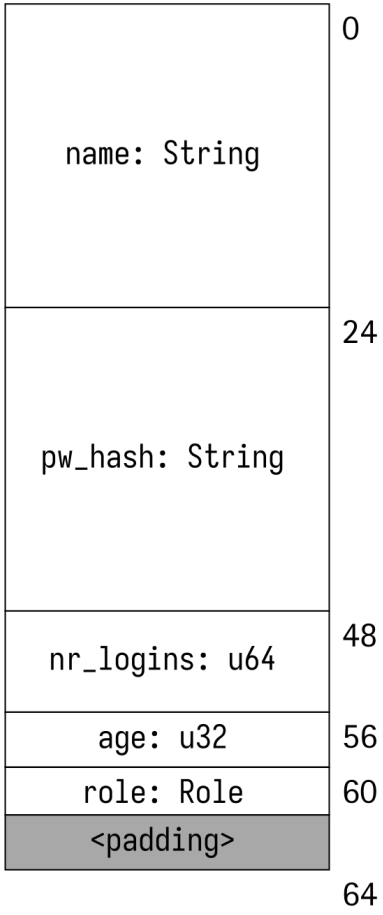
Angriffsmodell



Bekannt!

RAM:
Schreiben, aber kein Lesen

Gezielter Angriff



Binary icons created by Freepik - Flaticon

Welche Typen werden randomisiert?

Repr-Attribut:

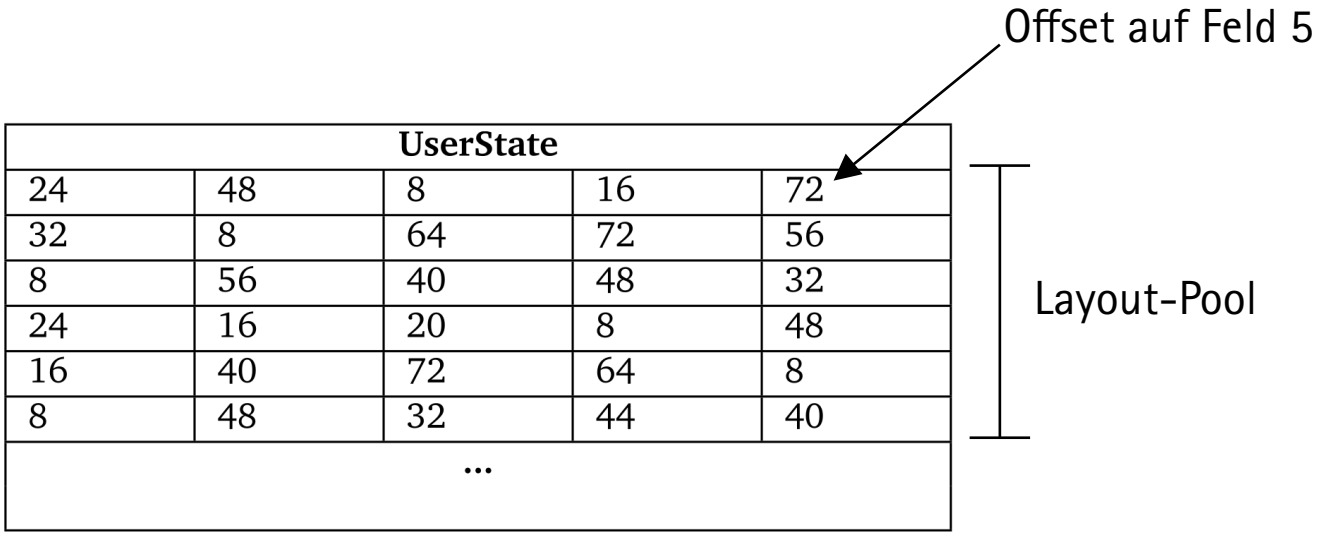
```
#[repr(dolr)]
struct UserState {
    name: String,
    role: Role,
    age: u32,
    nr_logins: u64,
    pw_hash: String,
}
```

Feldschwellwert:

Randomisiere alle Typen,
die mindestens
<n> Felder haben

rustc -Z dolr-default=<n>

Indirektionstabelle

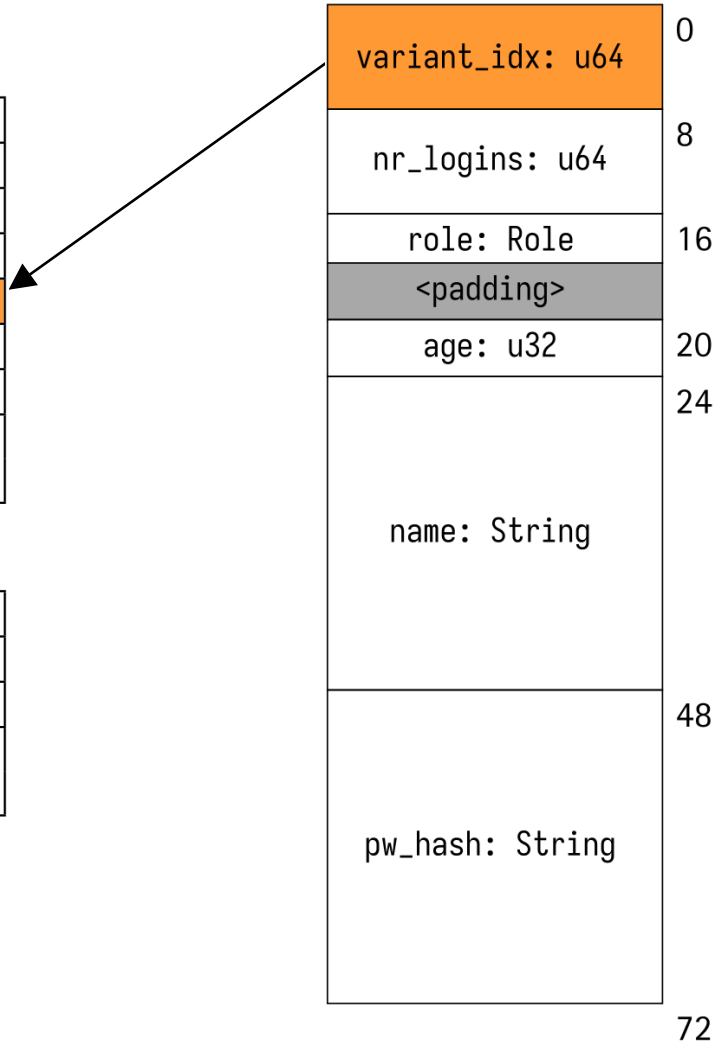


Object 2				
12	20	16	8	24
24	12	8	16	20
...				

Variantenindex

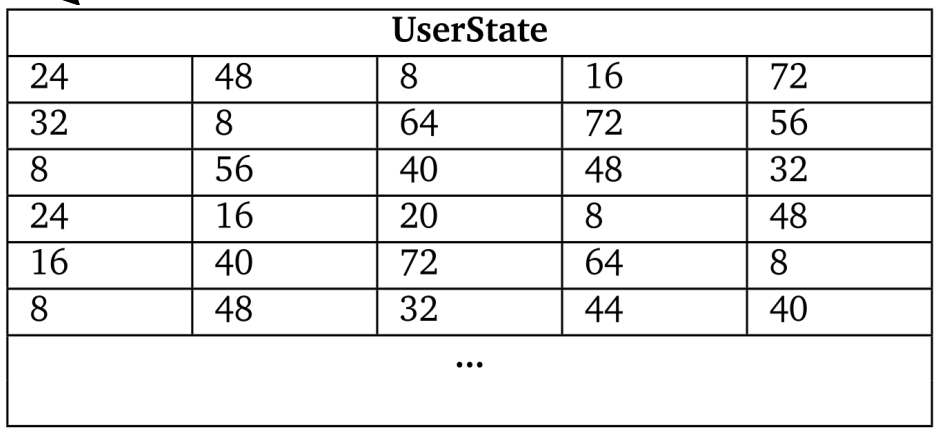
UserState				
24	48	8	16	72
32	8	64	72	56
8	56	40	48	32
24	16	20	8	48
16	40	72	64	8
8	48	32	44	40
...				

Object 2				
12	20	16	8	24
24	12	8	16	20
...				



Schutz der Indirektionstabelle

gs:



UserState				
24	48	8	16	72
32	8	64	72	56
8	56	40	48	32
24	16	20	8	48
16	40	72	64	8
8	48	32	44	40
...				

Object 2				
12	20	16	8	24
24	12	8	16	20
...				

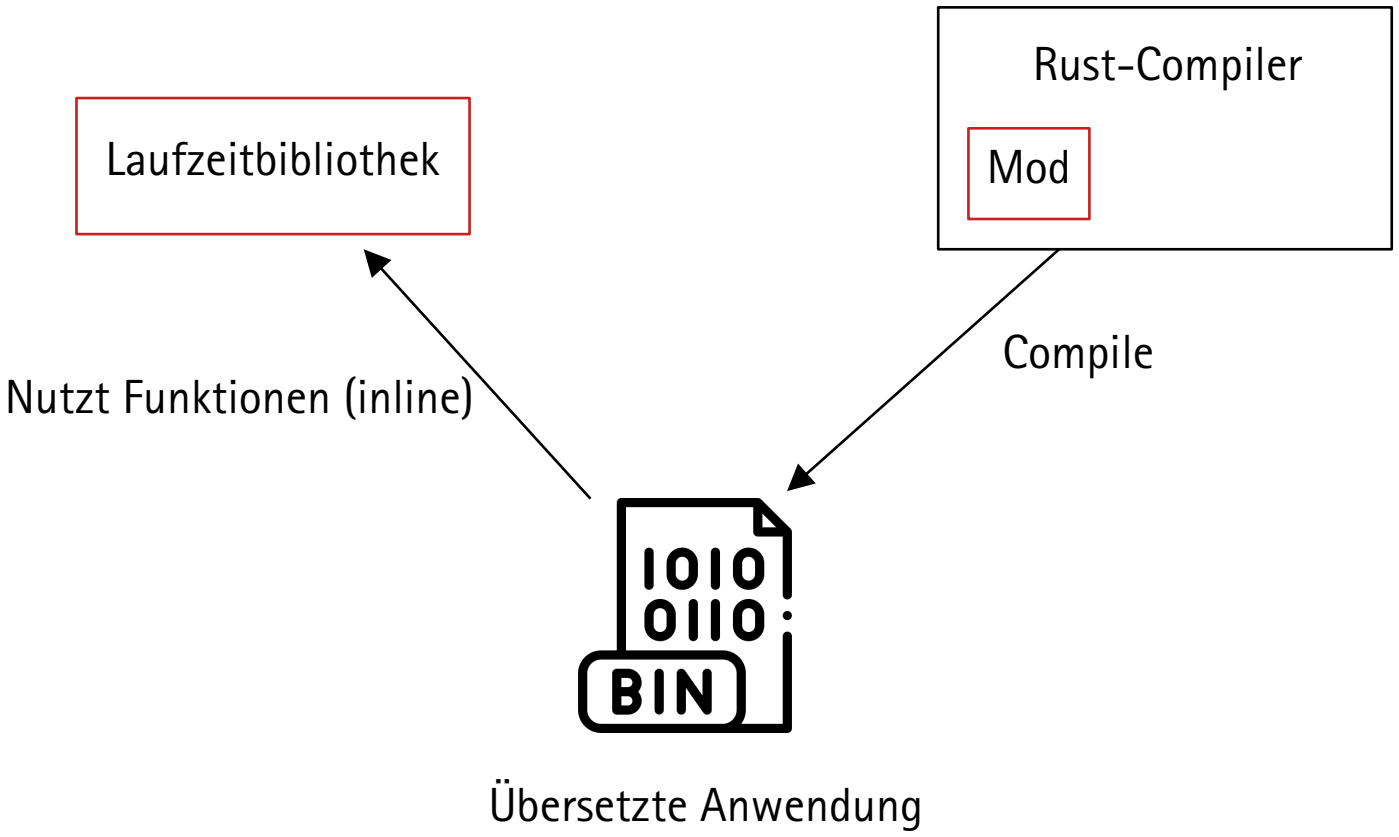
Jeder Zugriff indirekt über `gs:`

```
mov rax, gs:[rcx]
```

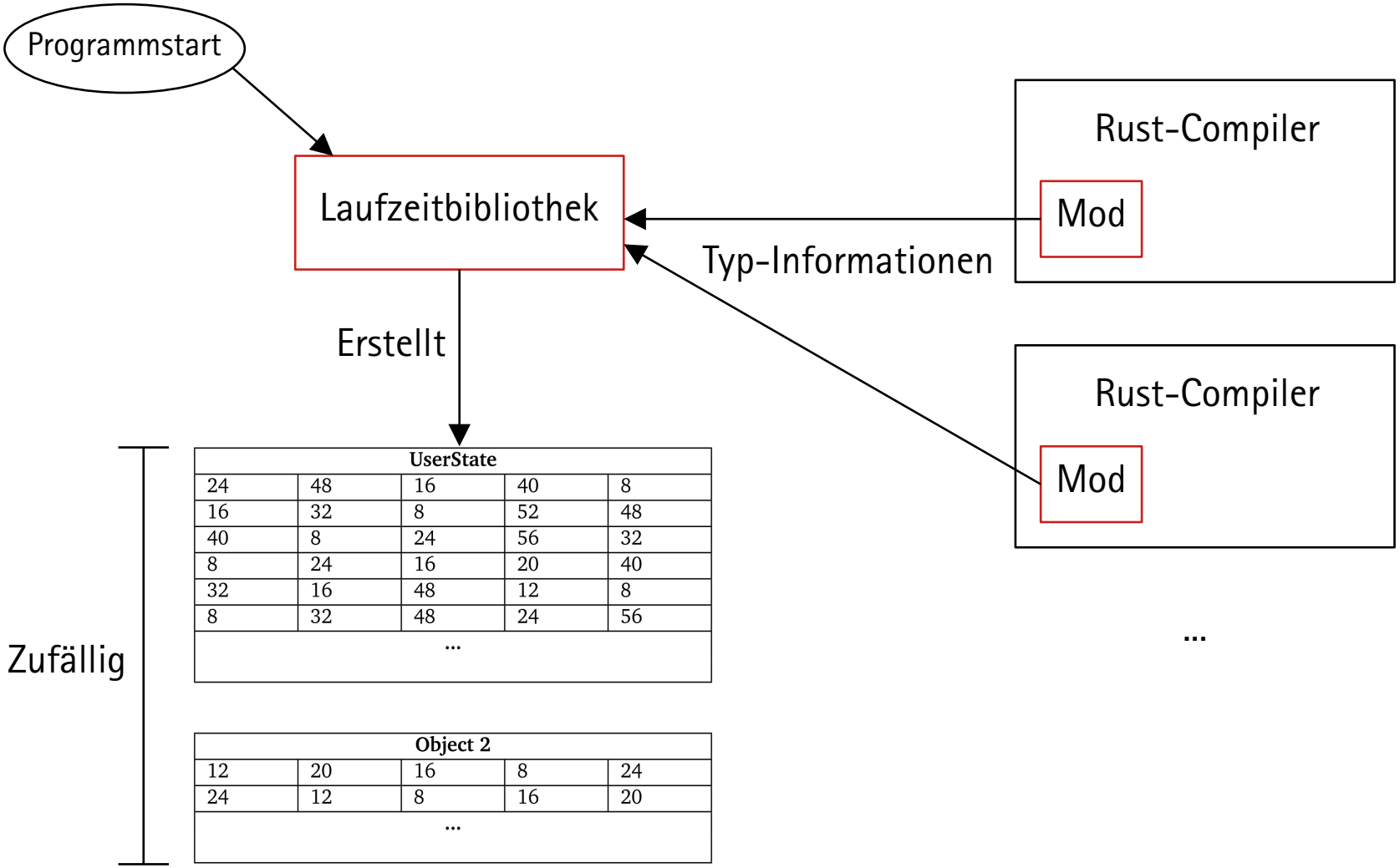
Basisadresse muss nicht gespeichert werden!

Schreibschutz:
Pages sind Read-Only

Architektur



Erstellen der Indirektionstabelle



Inhalt

- Struct/Objekt-Layout-Randomisierung
  - Was gibt es bereits?
- Architektur und Implementierung
- Evaluation
  - Kompatibilität
  - Performanz

Untersuchte Anwendungen

Coreutils

Rocket Webframework

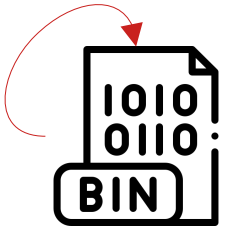
Firefox Webbrowser

Compiler Benchmarks

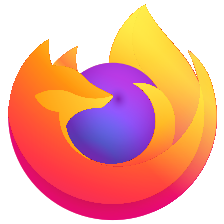
Performance icons created by Didin jpr - Flaticon
Coreutils Logo von https://github.com/uutils/coreutils/blob/main/docs/src/logo.svg

Kompatibilität (Problematische Fälle)

1. Transmute-Casts
2. Uninitialisierte Objekte
3. offset_of!(Typ, Feld)
4. Code vor der main()
5. IPC



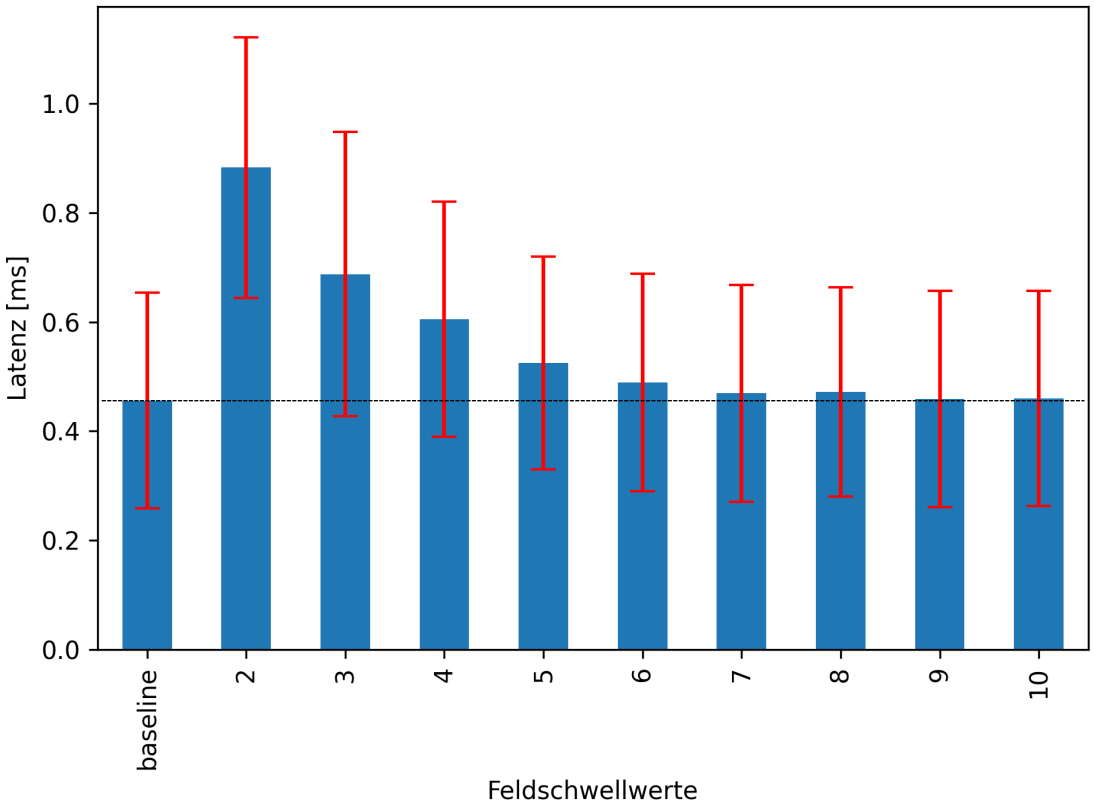
Kompatibilität



	Coreutils	Rocket	Rust-Compiler-Runtime-Benchmarks	Firefox
Lauffähig?	Leichte Modifikation.	✓	Leichte Modifikation.	Leichte Modifikation. Nur lauffähig ab Feldschwellwert 3.

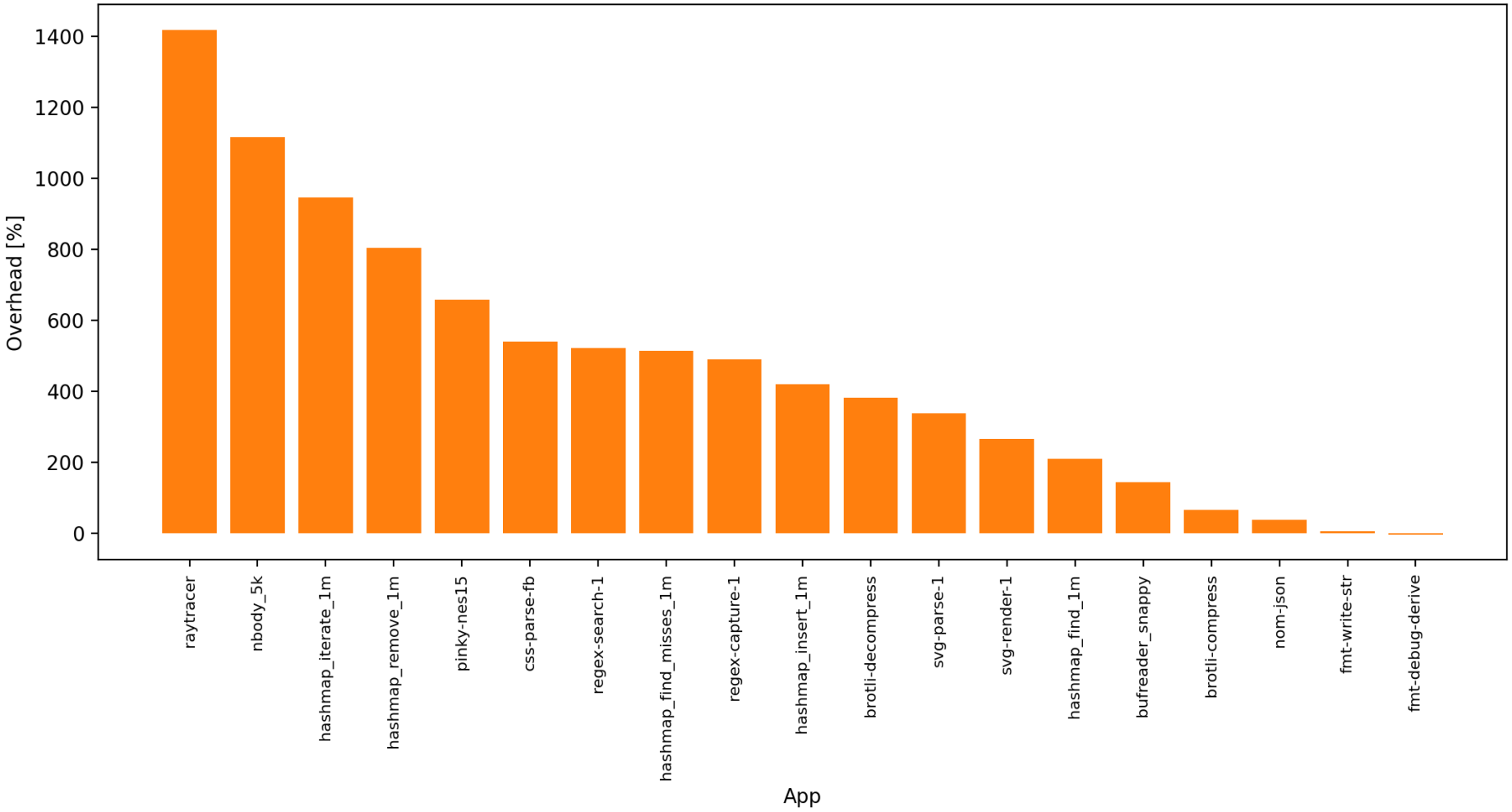
Performanz – Rocket (Hello-App)

Latenz [ms]:

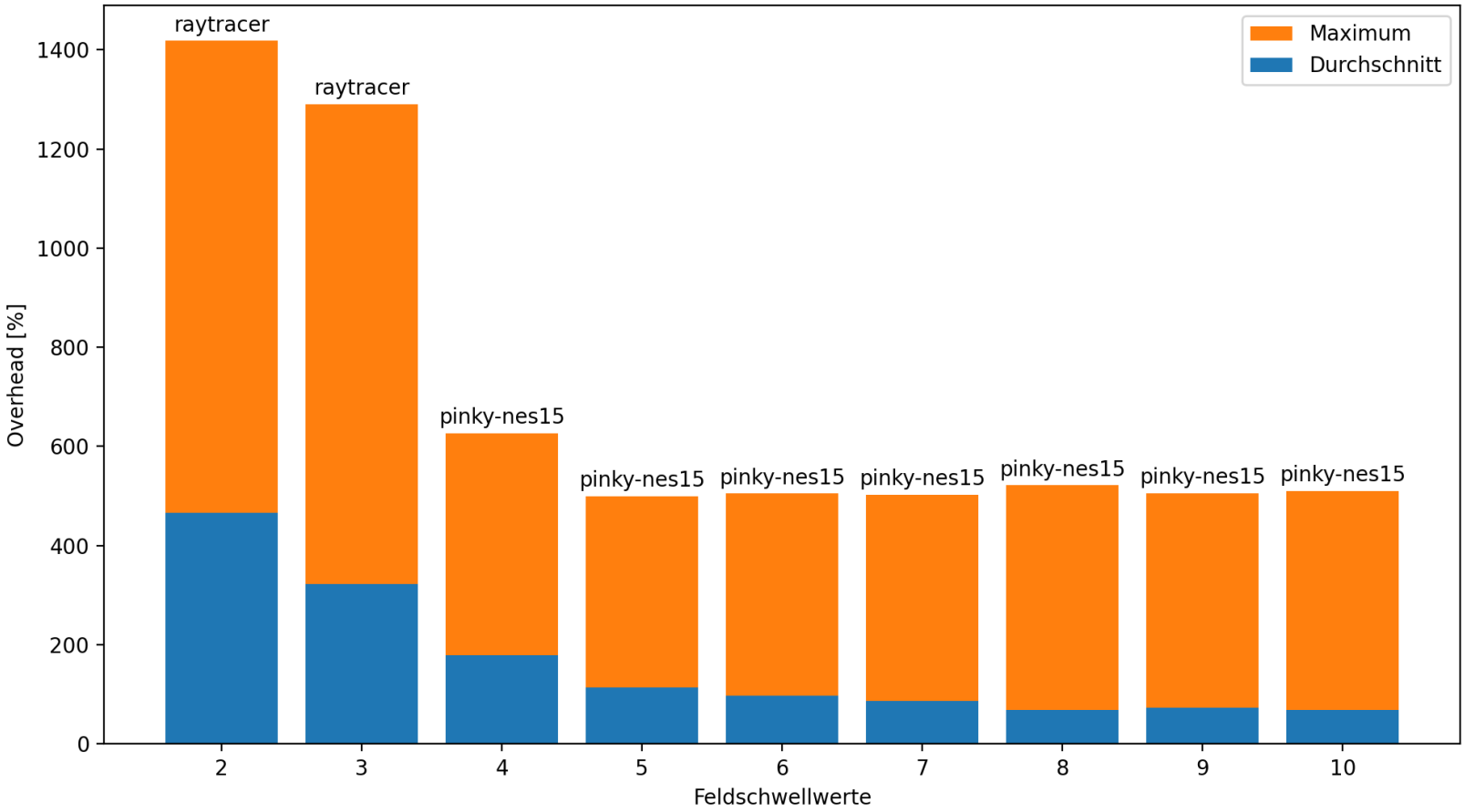


Performanz – Compiler Runtime Benchmarks

Feldschwellwert: 2



Performanz – Compiler Runtime Benchmarks



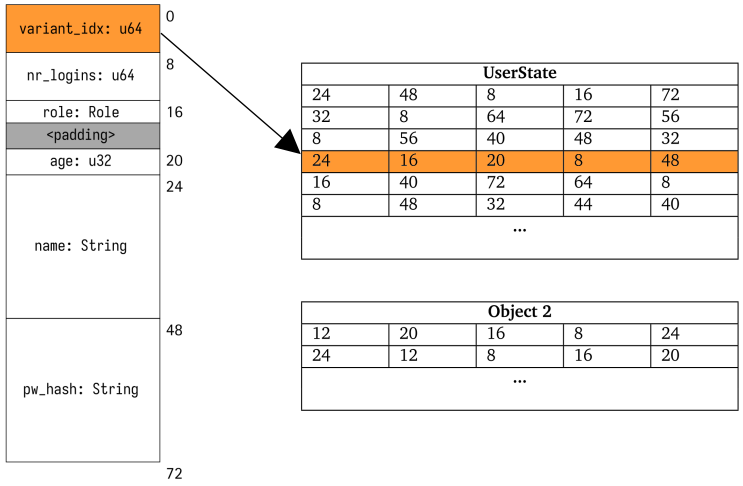
Performanz - Firefox

Feldschwellwert: 3

Benchmark	Overhead
Jetstream 2	+/- 0
Kraken	+/- 0
Speedometer	4.34%
MotionMark	34.74%

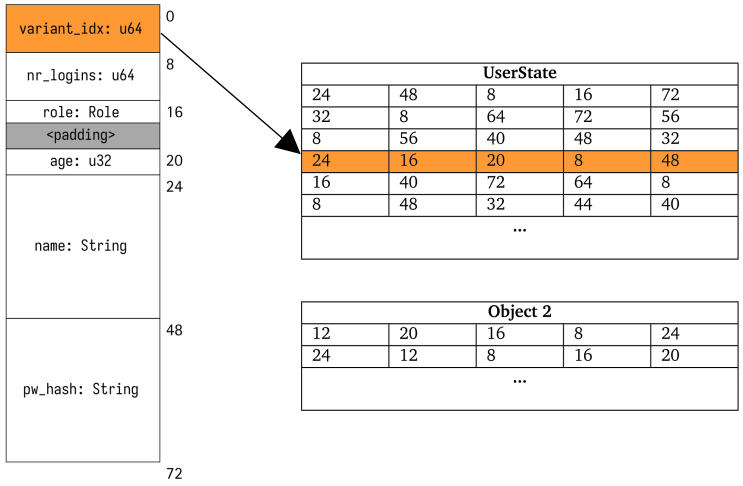
Zusammenfassung

- Randomisieren von Struct-Instanzen in Rust
  - Felder im Layout dynamisch umsortieren
  - Jede Instanz hat ein eigenes Layout
- Läuft mit dem Firefox Web-Browser
  - Responsiveness Reduktion: 4.34%
  - Grafikleistung Reduktion: 34.74%
- Future Work:
  - Zielgerichtete Auswahl an Structs kann Performanz verbessern



Zusammenfassung

- Randomisieren von Struct-Instanzen in Rust
  - Felder im Layout dynamisch umsortieren
  - Jede Instanz hat ein eigenes Layout
- Läuft mit dem Firefox Web-Browser
  - Responsiveness Reduktion: 4.34%
  - Grafikleistung Reduktion: 34.74%
- Future Work:
  - Zielgerichtete Auswahl an Structs kann Performanz verbessern



Vielen Dank für Ihre Aufmerksamkeit!