



# *Betriebssystembau (BSB)*

## **Abschlussarbeitsthemen**

<https://ess.cs.tu-dortmund.de/Teaching/SS2019/BSB/>

---

**Horst Schirmeier**

[horst.schirmeier@tu-dortmund.de](mailto:horst.schirmeier@tu-dortmund.de)  
<https://ess.cs.tu-dortmund.de/~hsc>





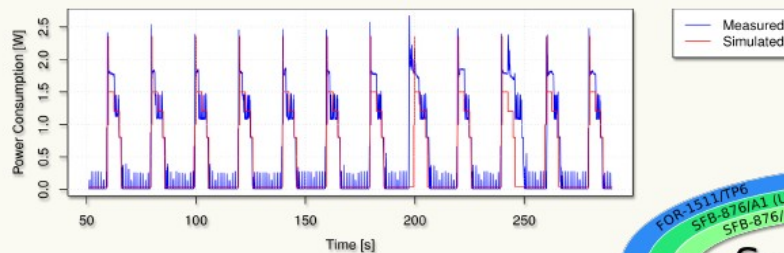
# Wir suchen: (Diplom|Master)and(en|innen)

## Allgemeines

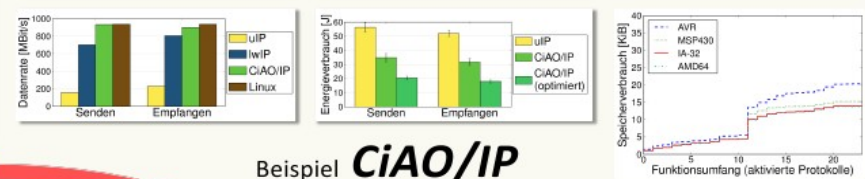
- Empirisches Arbeiten → Bauen, Messen, Bewerten
- Anwendungsorientierung ist unser Querschnittsthema
  - in verschiedenen Bereichen der eingebetteten Systemsoftware

## Themen

- Ressourcenverbrauch
- Modellierung und Vorhersage

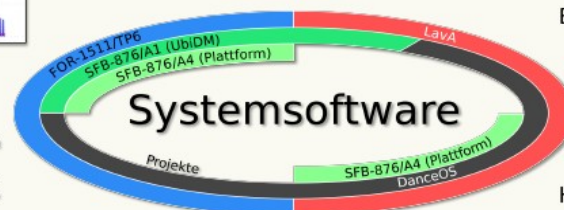
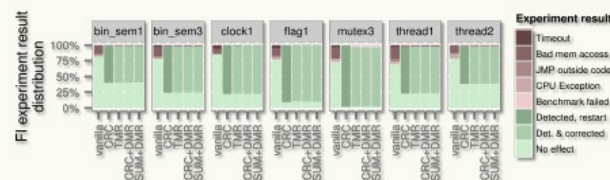


- Effizienz durch Maßschneiderung
- Hardware/Software-Produktlinien



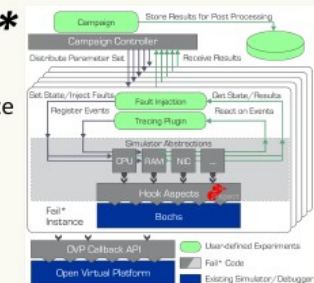
Beispiel **CiAO/IP**

Eine Speicherplatz und Energie sparende Produktlinie eingebetteter TCP/IP-Stacks



Beispiel **Fail\***

Hocheffiziente parallele Fehlerinjektionsexperimente



- Fehlertoleranz in Echtzeitsystemen
- Softwarebasierte Maßnahmen gegen Versagen durch Hardware-Fehler

Beispiel **AspectC++**

Unsere Implementierungssprache für CiAO: Eine aspektorientierte C++-Erweiterung

- Werkzeuge für Systementwickler



# Themenbereiche

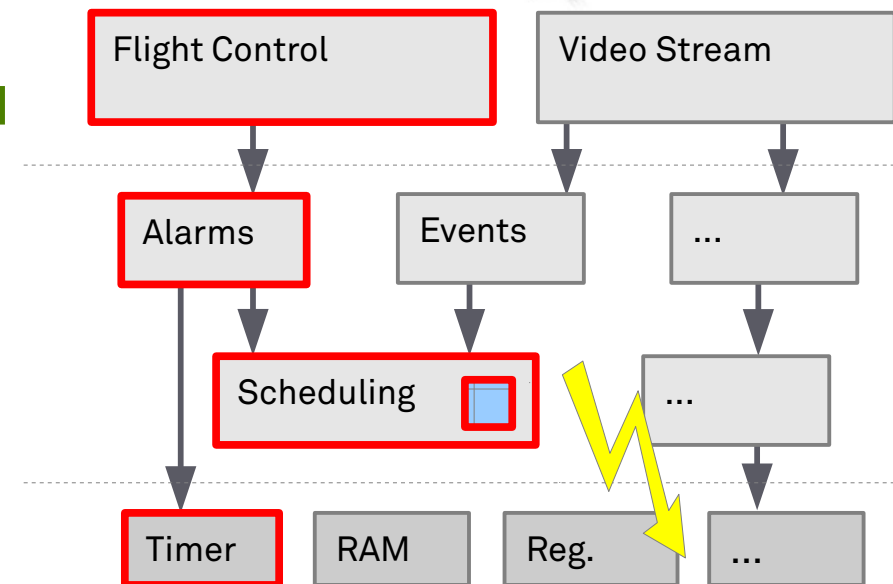
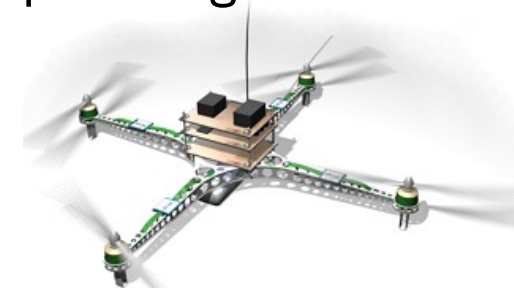
- **Fehlertoleranz und Fehlerinjektion (Horst Schirmeier)**
  - Experimentierplattform „FAIL\*“
- **Synchronisation und BS-Schnittstellen (Horst Schirmeier)**
  - Pfadausdrücke
  - POSIX-Betriebssystemschnittstellen
- **Analyse komplexer Softwaresysteme (Alexander Lochmann)**
  - LockDoc: Lock-Analyse in Betriebssystemen
  - Pfadüberdeckungstests in Linux durch Benchmark-Synthese



# Fehlertoleranz: Projekt



- **Annahme:** zukünftige Hardware teilweise **unzuverlässig**
  - Strukturgrößen werden immer kleiner
  - Energiesparen durch weiteres Senken der Betriebsspannung
- **Idee:**
  - **selektives Härten** von Anwendungs- und Betriebssystemsoftware
- **Fragestellung:**
  - Welche Teile des Systems **passend für die Anwendung härten**?
  - **Wie gut wirkt** ein Härtenungsverfahren?  
(Stürzt das System jetzt weniger häufig ab? Wird die für den Anwendungsfall benötigte Dienstgüte jetzt erreicht?)

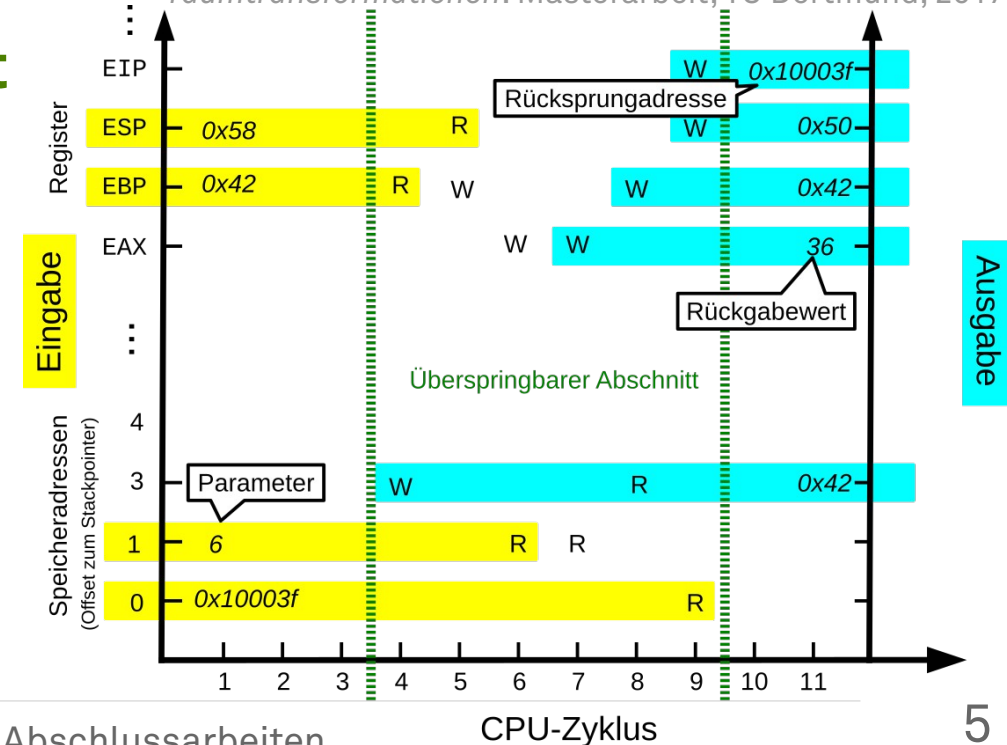




# Analyse/Bewertung: FI-Experimente (1)

- **Problem:** „Echte“ **unzuverlässige Hardware** ist schwer zu bekommen und verhält sich, nunja, ... unzuverlässig.
- **Ansatz: FAIL\***, Simulator-basierte **Fehlerinjektionsexperimente**
  - massiv parallele Durchführung vieler, teils **sehr lang laufender** Einzelexperimente
- **Problem: Experimentlaufzeit**
- **Aufgabe:**
  - Weiterentwicklung eines Verfahrens zum Überspringen von Programmabschnitten
  - Detaillierte Evaluation
  - **Ziel:** Verkürzung der Gesamtlaufzeit bei geringem Genauigkeitsverlust

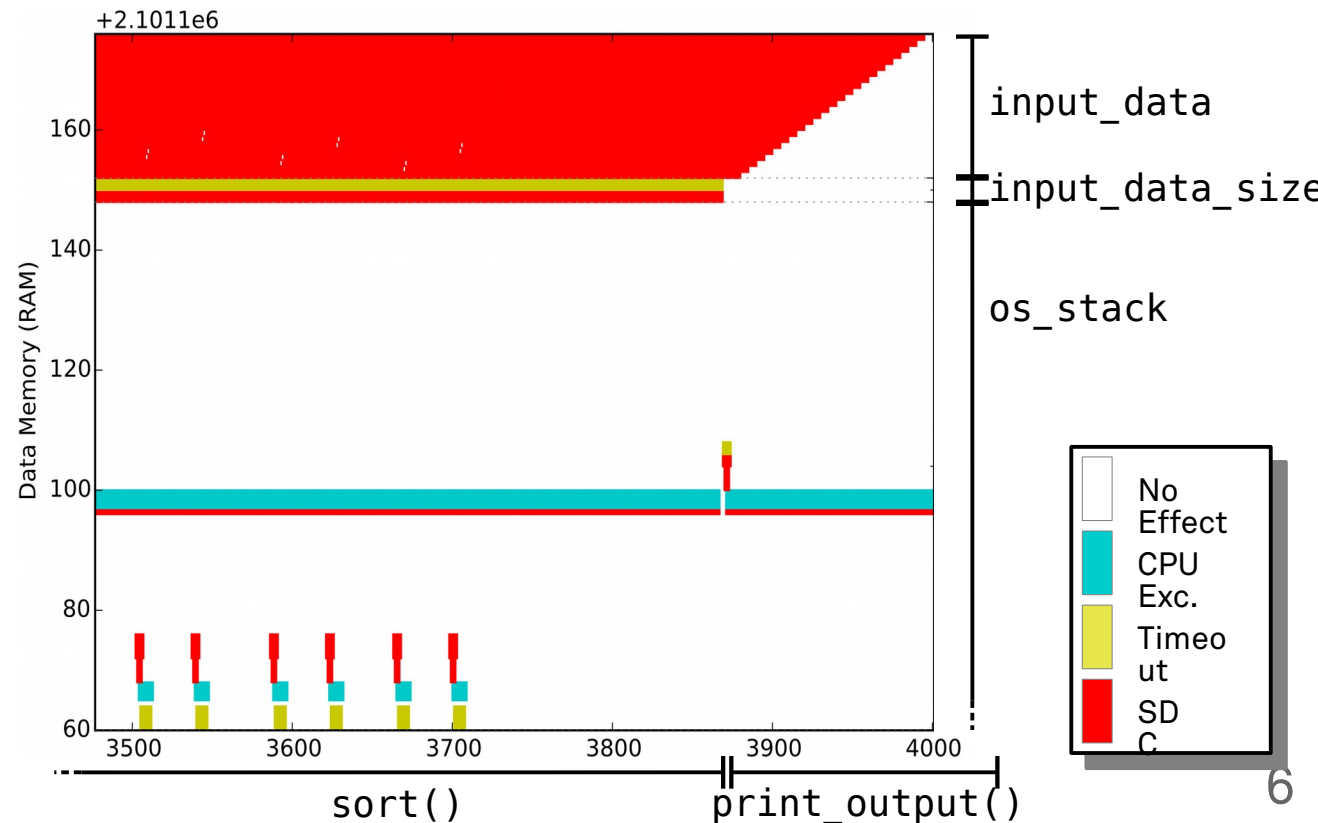
Merlin Stampa, Beschleunigung von Fehlerinjektionsexperimenten durch Zusammenfassung von Zustandsraumtransformationen. Masterarbeit, TU Dortmund, 2017





# Analyse/Bewertung: FI-Experimente (2)

- ... Problem: **Experimentlaufzeit**
- Aufgabe:
  - Maschinelles Lernen zur Verringerung der Experimentanzahl
  - Training eines Prädiktors mit bekannten Ergebnissen
  - **Ziel:** Verkürzung der Gesamtlaufzeit bei geringem Genauigkeitsverlust





# Themenbereiche

- **Fehlertoleranz und Fehlerinjektion** (Horst Schirmeier)
  - Experimentierplattform „FAIL\*“
- **Synchronisation und BS-Schnittstellen** (Horst Schirmeier)
  - Pfadausdrücke
  - POSIX-Betriebssystemschnittstellen
- **Analyse komplexer Softwaresysteme** (Alexander Lochmann)
  - LockDoc: Lock-Analyse in Betriebssystemen
  - Pfadüberdeckungstests in Linux durch Benchmark-Synthese





# Pfadausdrücke

- **path** 10:(1:(*insert*); 1:(*remove*)) **end**
- **Idee:**
  - Transfer **in moderne Programmiersprache**
    - C++? (+ AspectC++?)
  - **Erweiterungen** z.B. um Priorisierungen, verschiedene „+“
  - Intelligente **Zustandsautomatensynchronisation** (lock-/wait-free?)
- **Evaluation**
  - anhand von synthetischen Beispielen
  - anhand von Real-World-Programm(en): Refactoring  
→ hinterher wartbarer?





# POSIX-Betriebssystemschnittstellen

- **Binsenweisheit:** Kontextwechsel bei Systemaufrufen sind teuer
  - Das hat sich im Kontext **Meltdown** und KPTI/KAISER noch verschärft!
- **Idee: Bündelung von Systemaufrufen**
  - Wird schon seit Langem für Spezialfälle gemacht, z.B. **sendfile(2)**:

```
while ((cnt = read(fd_in, buf, BUFSIZ)) > 0) {
    write(fd_out, buf, cnt);
}
```

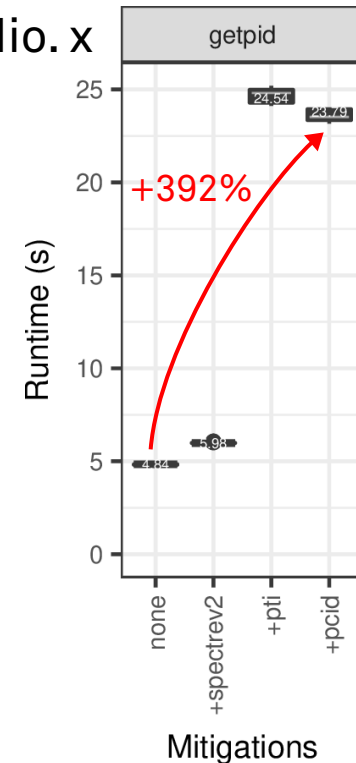


```
sendfile(fd_out, fd_in, NULL, size);
```

- **Aufgabe:**

- Entwurf und prototypische Implementierung von verallgemeinerter Systemaufruf-Bündelung unter Linux
- **Herausforderungen:** Entwurf der Schnittstelle (typesichere Sprache, ggf. Verwendung von eBPF), Konzept zur Angriffssicherheit

100 Mio. x





# Themenbereiche

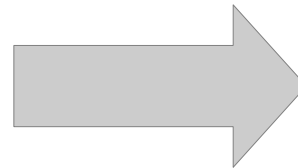
- **Fehlertoleranz und Fehlerinjektion** (Horst Schirmeier)
  - Experimentierplattform „FAIL\*“
- **Synchronisation und BS-Schnittstellen** (Horst Schirmeier)
  - Pfadausdrücke
  - POSIX-Betriebssystemschnittstellen
- **Analyse komplexer Softwaresysteme** (Alexander Lochmann)
  - LockDoc: Lock-Analyse in Betriebssystemen
  - Pfadüberdeckungstests in Linux durch Benchmark-Synthese



# LockDoc: Lock-Analyse in BS (1)

- **Problem:** **Locking** von komplexen Softwaresystemen, wie BS, ist **nicht trivial**
- **Ansatz:**
  - Aufzeichnung der **Speicherzugriffe** und **Lock-Operationen**
  - Ableiten von **Lock-Hypothesen**

Hypothese	Anzahl
A	10
B	5
A → B	20



„Lock A und Lock B müssen gehalten, um Datenstruktur Inode zuzugreifen.“

- **Problem:** **Verifikation** des Ansatzes
- **Aufgabe:** Portierung und Überprüfung dokumentierter Locking-Regeln in OpenBSD

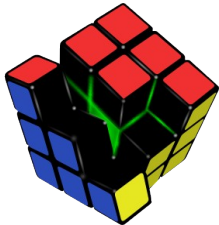


# LockDoc: Lock-Analyse in BS (2)

- **Situation:**
  - **Regeln** sind **unpräzise** bzgl. der Position des Locks
  - `EMBOTHER(inode.i_lock)` für `dentry.d_flags` → **Lock** ist in **irgendeiner** anderen **Datenstruktur eingebettet**
- **Problem:** **Nicht** besonders **aussagekräftig** für den **Entwickler**
- **Ziel:** **Regeln** für den Entwickler **präzisieren** → `EMBSAME(dentry.d_inode->inode.i_lock)`
- **Aufgabe:** **Objektbeziehungen aufzeichnen** und post-mortem **auswerten**.



# So, das war's



Arbeitsgruppe Eingebettete Systemsoftware  
Lehrstuhl 12

**Bei Interesse bitte einfach melden!**