technische universität
dortmund

Software ubiquitärer Systeme (SuS)

DA/MA-Themen SS18

https://ess.cs.tu-dortmund.de/DE/Teaching/Theses/

Olaf Spinczyk

olaf.spinczyk@tu-dortmund.de
https://ess.cs.tu-dortmund.de/~os

AG Eingebettete Systemsoftware
Informatik 12, TU Dortmund

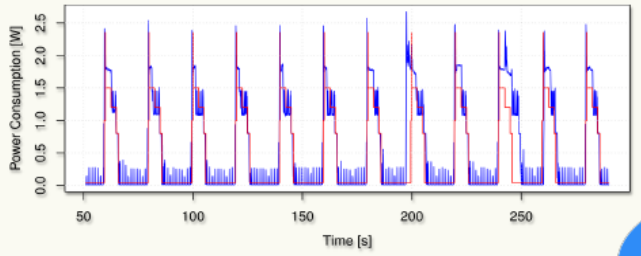
Wir suchen: (Diplom|Master)and(en|innen)

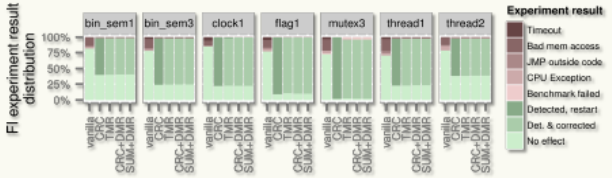
Allgemeines

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

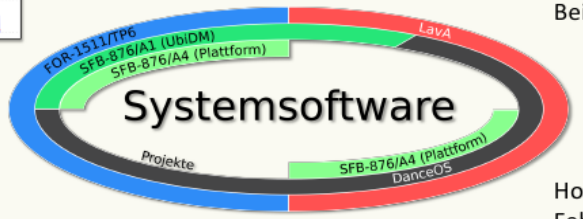
Themen

- **Ressourcenverbrauch**
- Modellierung und Vorhersage

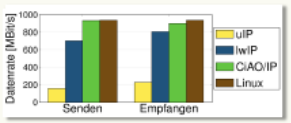


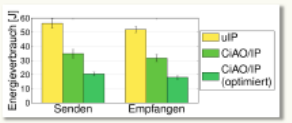


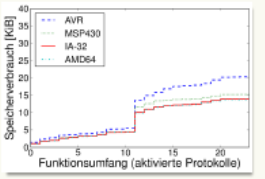
- **Fehlertoleranz** in Echtzeitsystemen
- Softwarebasierte Maßnahmen gegen Versagen durch Hardware-Fehler



- Effizienz durch **Maßschneiderung**
- Hardware/Software-Produktlinien





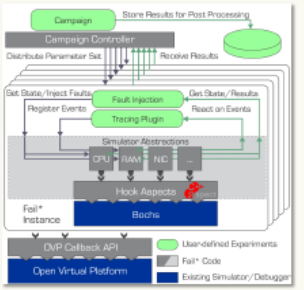


Beispiel ***CiAO/IP***

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

Beispiel ***Fail****

Hocheffiziente parallele Fehlerinjektionsexperimente



aspect
www.aspectc.org

Beispiel ***AspectC++***

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

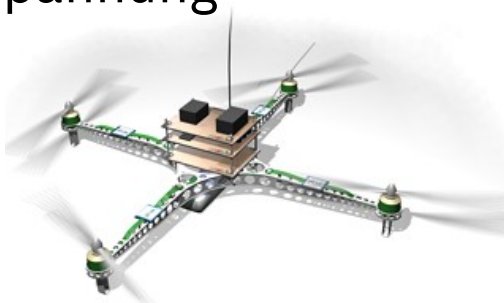
- **Werkzeuge** für Systementwickler

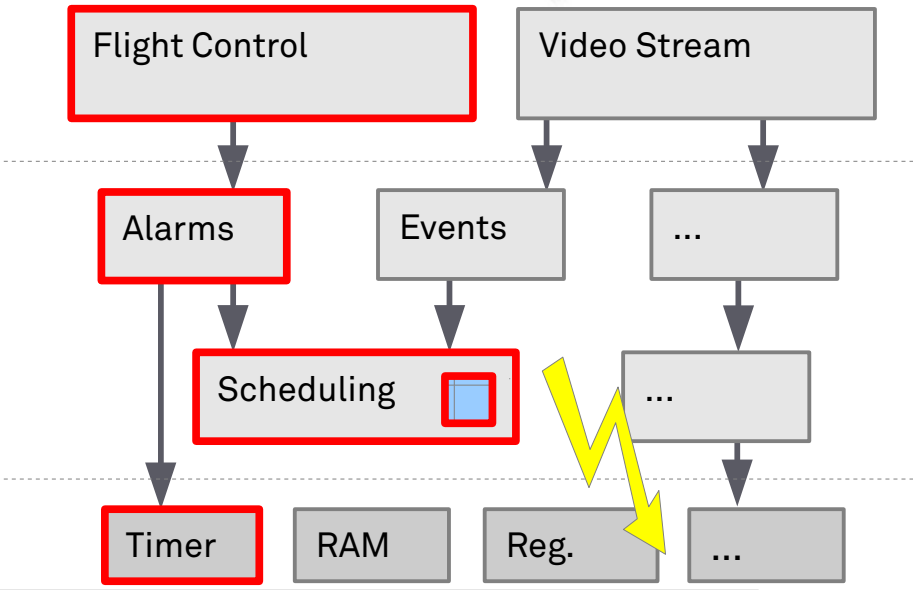
Themenbereiche

- **Fehlertoleranz und Fehlerinjektion** (Horst Schirmeier)
  - Experimentierplattform „FAIL*“
  - 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 Härtungs-verfahren?
    (Stürzt das System jetzt weniger häufig ab? Wird die für den Anwendungsfall benötigte Dienstgüte jetzt erreicht?)

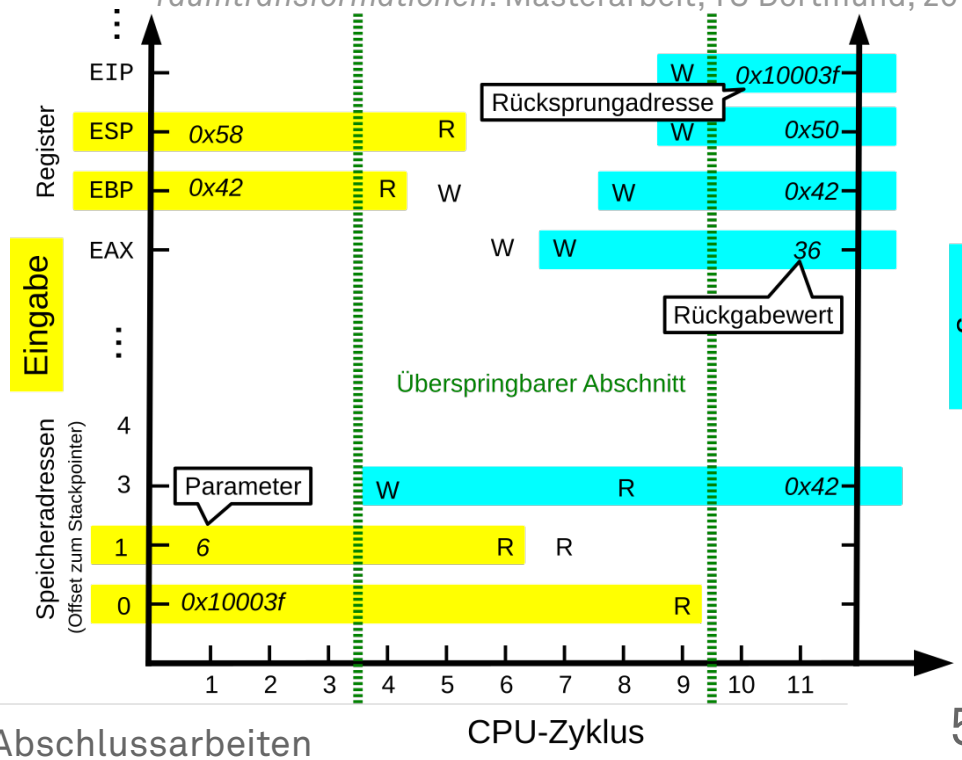




Analyse/Bewertung: FI-Experimente

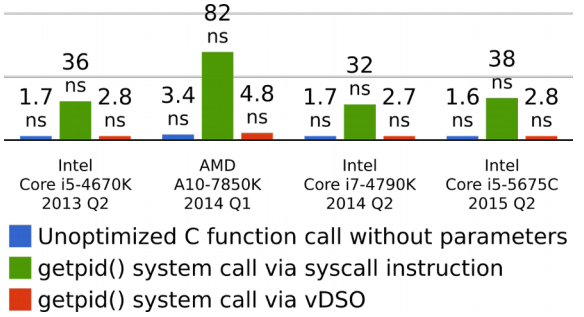
- **Problem:** „Echte“ **unzuverlässige Hardware** ist schwer zu bekommen und verhält sich, nunja, … unzuverlässig.
- **Ansatz: FAIL***, Simulator-basierte **Fehlerinjektionsexperimente**
  - massiv parallele Durchführung vieler, teils **sehr lang laufender** Einzelexperimente
- **Problem: Experimentlaufzeit**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

Merlin Stampa, *Beschleunigung von Fehlerinjektions-experimenten durch Zusammenfassung von Zustands-raumtransformationen*. Masterarbeit, TU Dortmund, 2017



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)**:



http://arkanis.de/weblog/2017-01-05-measurements-of-system-call-performance-and-overhead

```
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 (typsichere Sprache, ggf. Verwendung von eBPF), Konzept zur Angriffssicherheit

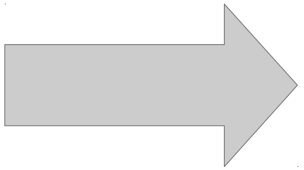
Themenbereiche

- **Fehlertoleranz und Fehlerinjektion** (Horst Schirmeier)
  - Experimentierplattform „FAIL*“
  - 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:** Linux-**Benchmark** führt **Codepfade mehrfach** oder gar nicht aus
- **Problem:**
  - Verfälschung der Ergebnisse
  - Bugs bleiben unentdeckt
- **Ziel:** Jeden **Codepfad** genau **einmal** ausführen (Pfadüberdeckung)
- **Aufgabe:** Finden und Ausführen von eindeutigen Codepfaden im BS

So, das war's

Arbeitsgruppe Eingebettete Systemsoftware
Lehrstuhl 12

Bei Interesse bitte einfach melden!