Slothful Linux: An Efficient Hybrid Real-Time System by Hardware-Based Task Dispatching

Rainer Müller

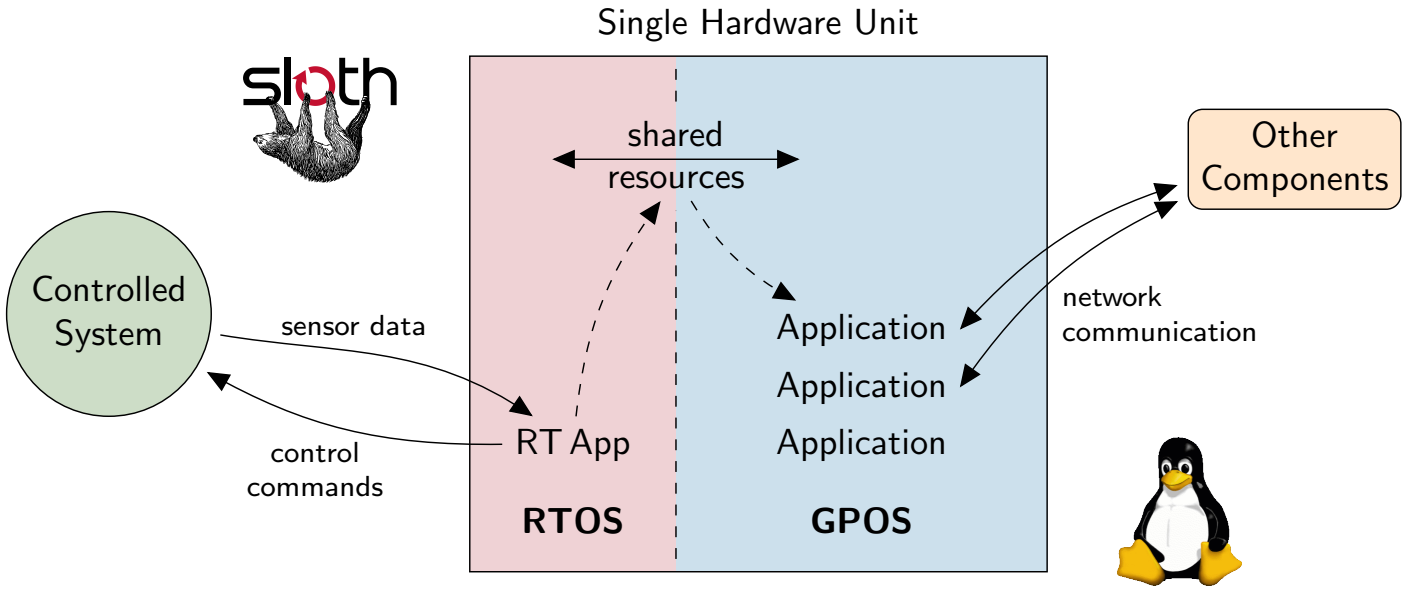
21. November 2013

Spezialisierung von Betriebssystemen

- Vielzweckbetriebssysteme (General Purpose OS, GPOS)
  - Benutzerinteraktion, Datenkommunikation, Entertainment, ...
  - weitreichende Hardwareunterstützung
- Echtzeitbetriebssysteme (Real Time OS, RTOS)
  - Steuerung von Fertigungsmaschinen, Industrieanlagen, ...
  - spezialisiert für bestimmte Aufgaben
  - *weiche*, *feste* oder *harte* Terminvorgaben

Konsolidierung

- Trennung in Echtzeitverarbeitung und restliche Aufgaben
  - RTOS: Maschinensteuerung
  - GPOS: Kommunikation, Statistiken, Visualisierung, ...
- Gemeinsamer Zugriff auf Daten
  - Konsolidierung auf derselben Hardwareplattform



⇒ **Slothful Linux:** Integration des **Linux**-Kernels mit dem bestehendem **Sloth**-Echtzeitbetriebssystem

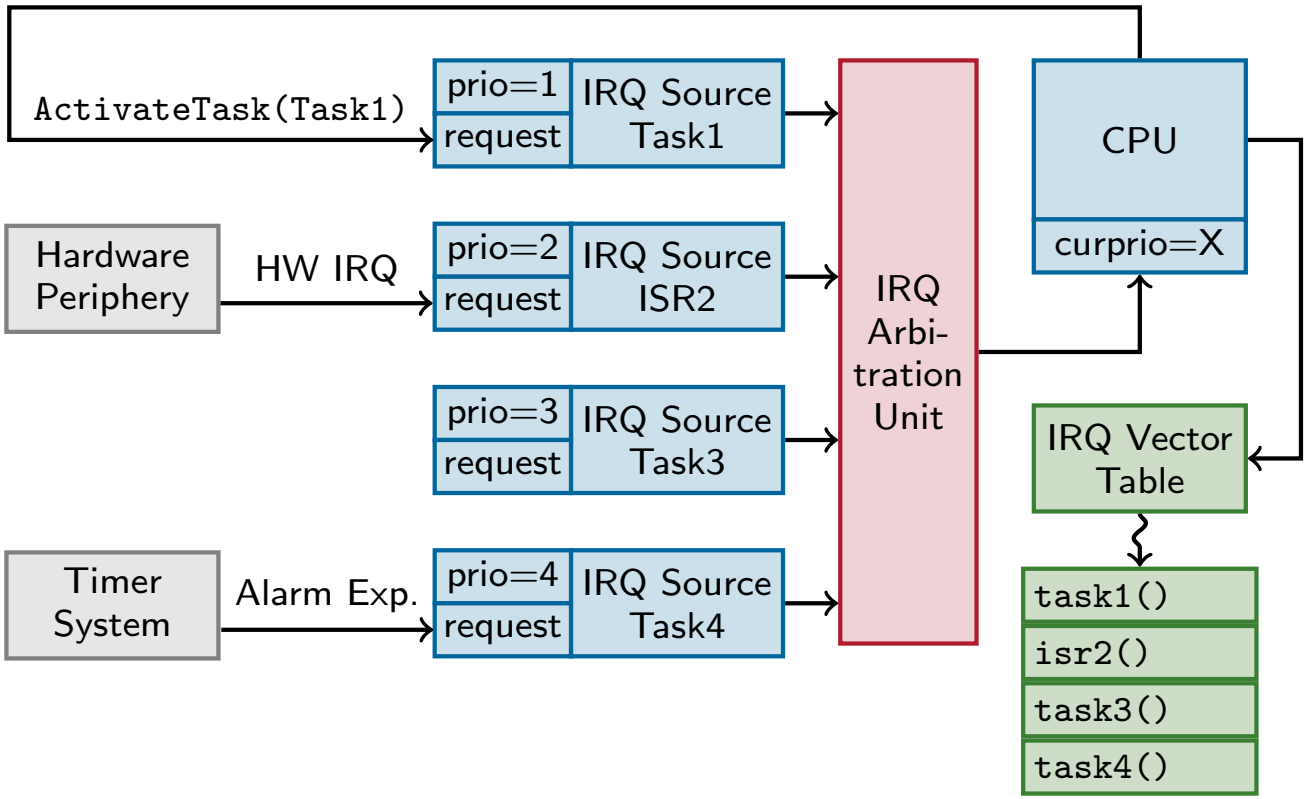
Überblick

- Sloth-Konzept
- Slothful Linux
- Evaluation
- Ausblick

Sloth-Konzept

- Idee: *Threads as Interrupts*
- Task-Aktivierung durch IRQ
- Hardware übernimmt das Einplanen und Einlasten
- Anwendbar auf statisch konfigurierte Echtzeitsysteme mit festen Prioritäten (z. B. OSEK/AUTOSAR OS)

Sloth-Design



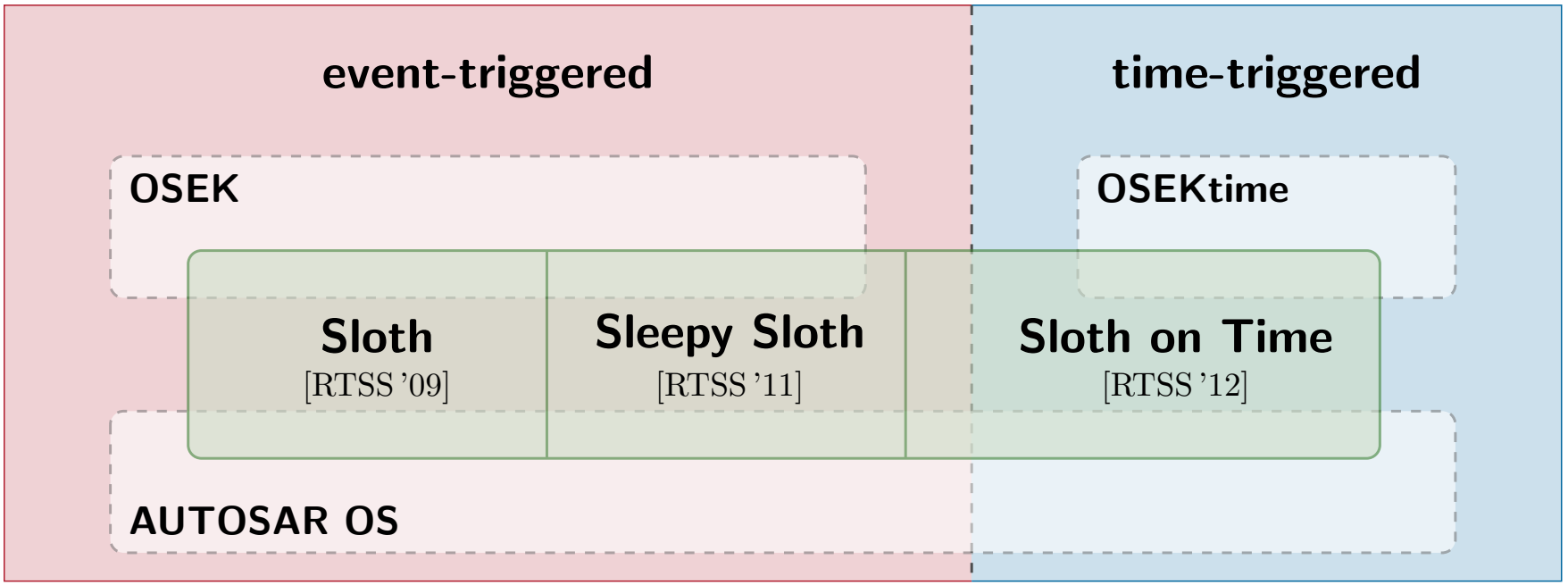
- Hardware muss genügend Prioritäten zur Verfügung stellen
- Unterbrechungsanforderungen aus Software heraus für synchrone Aktivierungen

Vorteile des Sloth-Designs

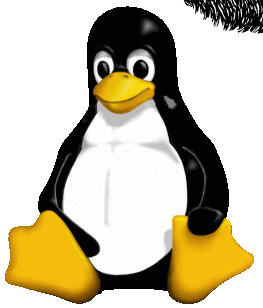
- Schlanker, effizienter Kern
- Einheitliche Abstraktion für Tasks und ISRs
  - unabhängig davon, ob Aktivierung durch Hardware oder Software ausgelöst wurde
- Einheitlicher Prioritätsraum für Tasks und ISRs
  - keine Prioritätsumkehr (*rate-monotonic priority inversion*)
- Synchronisation durch Anheben der CPU-Priorität
  - *Priority Ceiling Protocol*

Sloth-Familie

- Sloth-Konzept bereits in mehreren Bereichen erfolgreich



Slothful Linux



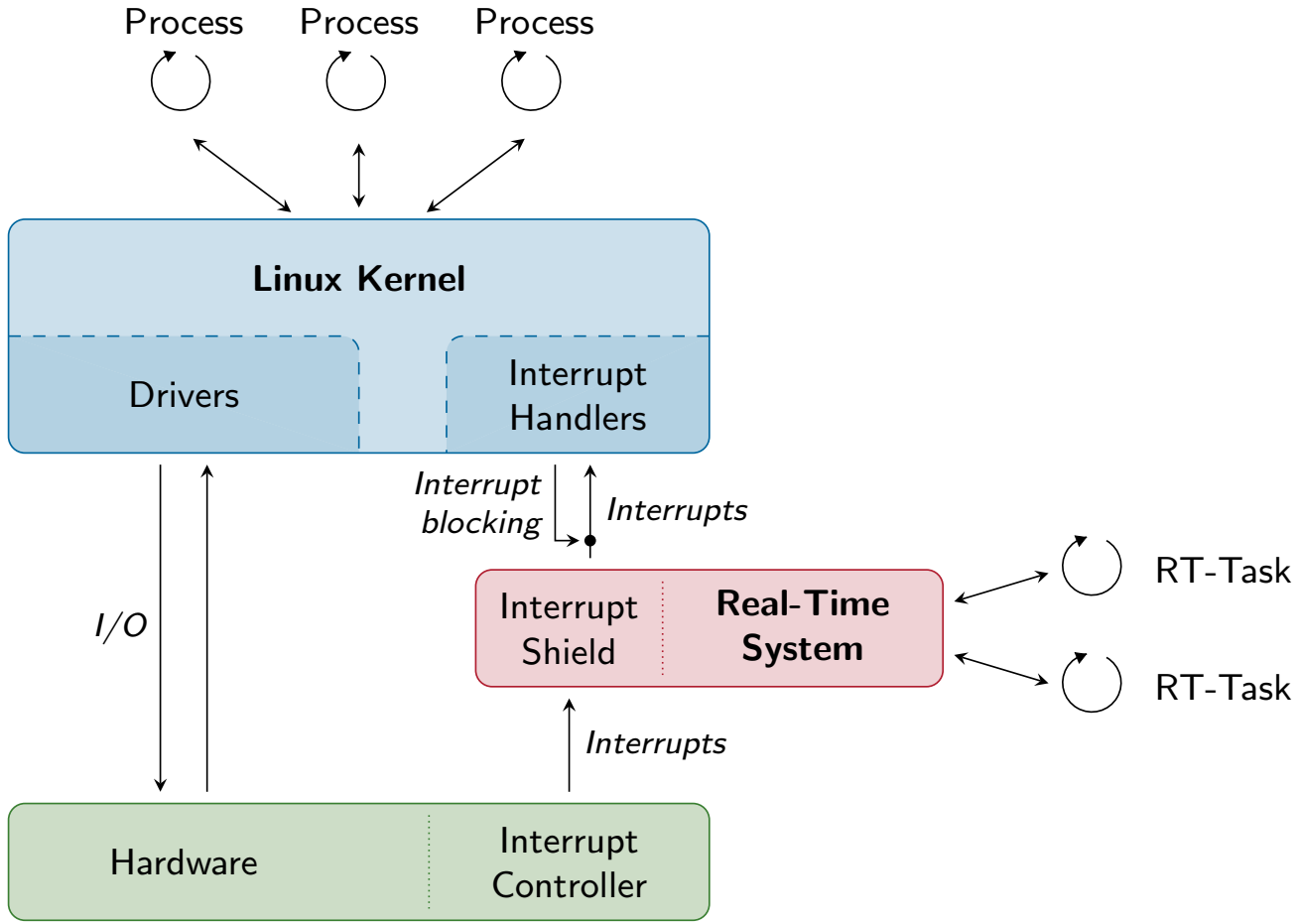
- Übertragung der Vorteile des Sloth-Konzepts in ein kombiniertes System zusammen mit Linux
- Untersuchung der bestehenden auf Linux basierenden Ansätze

1. Veränderungen am Kernel, um Echtzeitverhalten zu implementieren (RT-PREEMPT-Patch)
2. Virtualisierung der Unterbrechungsbehandlung

Virtualisierung der Unterbrechungsbehandlung

- I-Pipe-Patch
  - Basis von RTAI oder Xenomai
  - Virtualisierung der Unterbrechungsbehandlung in Abstraktionsschicht
  - Ansatz: *Optimistic Interrupt Protection*

Virtualisierung der Unterbrechungsbehandlung

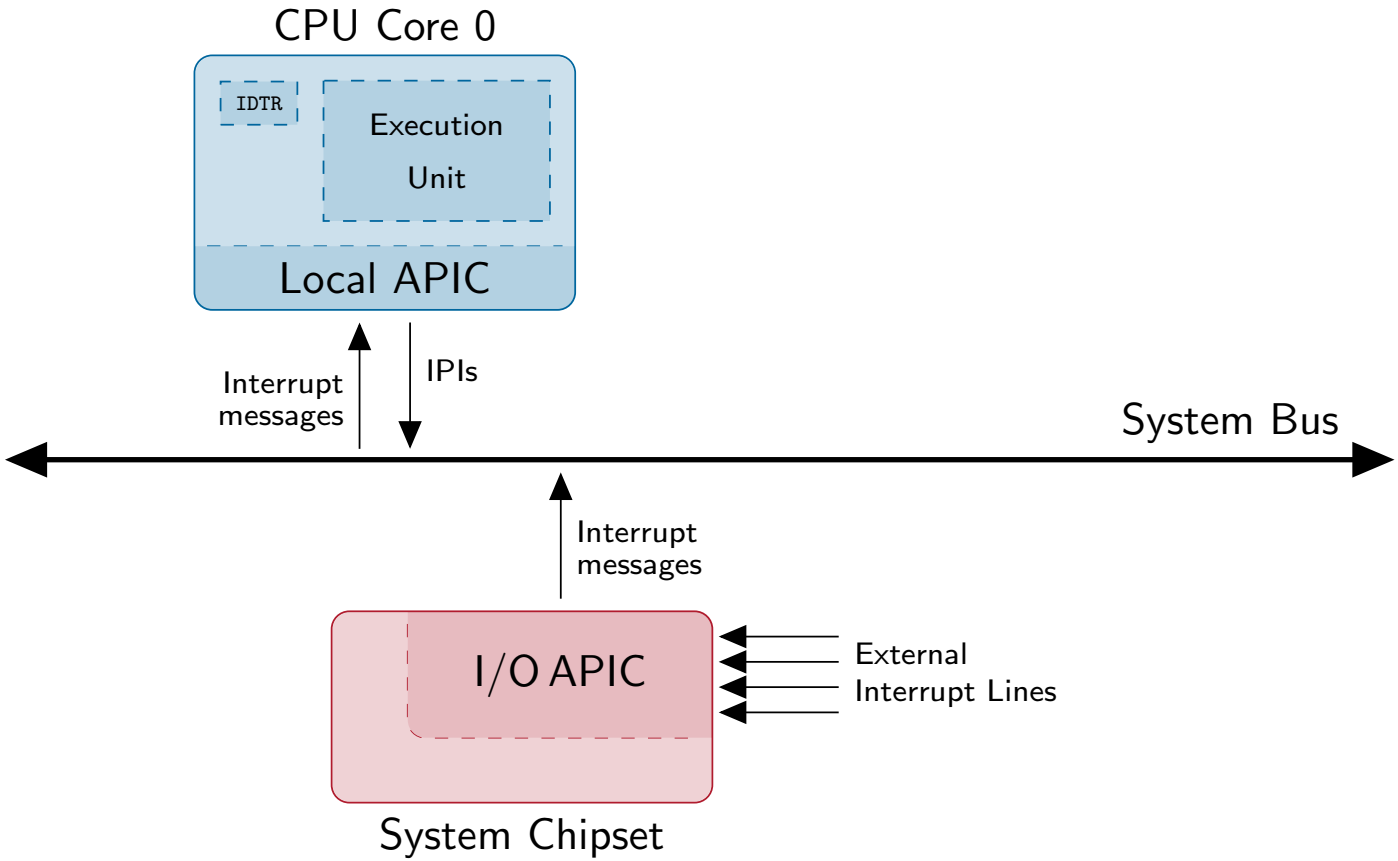


⇒ RTOS kann jederzeit den Linux-Kernel unterbrechen,
Linux-Kernel hat keinen Einfluss auf Ablauf im RTOS

Entwurf und Implementierung

- Zielplattform: Intel x86 APIC
- Sloth (bare-metal)
  - Ziel: Prüfung der Realisierbarkeit
  - Ausnutzung der Hardware ohne Einschränkungen
  - Funktionsumfang: Tasks, Resources
- Slothful Linux
  - Virtualisierung der Unterbrechungsbehandlung des Linux-Kernels
  - Anpassung des Linux-Kernels auf Basis vorhandener Patches
  - Kommunikationskanal zwischen Echtzeitanwendung und Prozessen unter Linux
  - Funktionsumfang: Tasks, Resources, Pipes

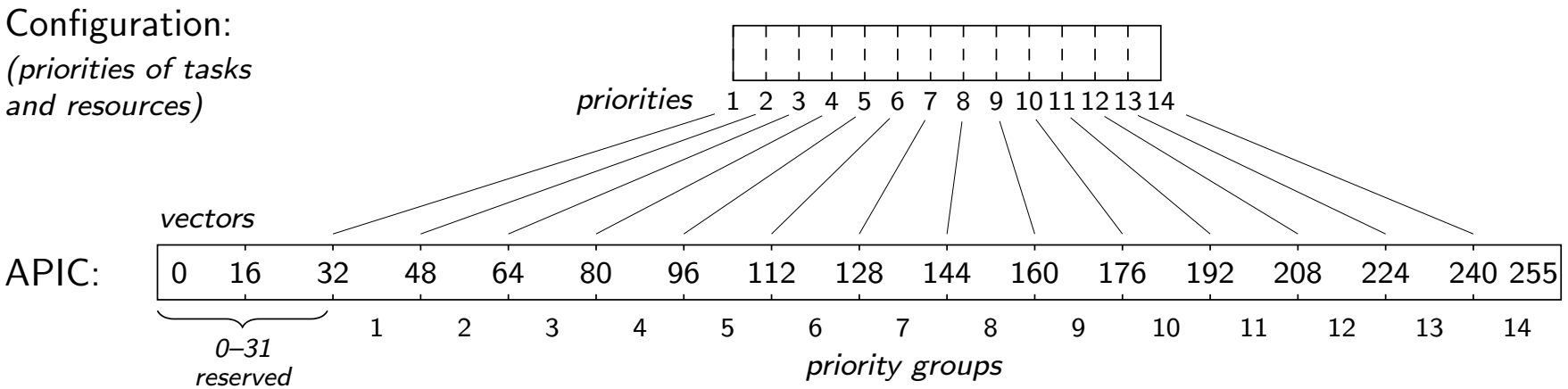
Sloth auf Intel x86: Local APIC



- Asynchrone Aktivierung von Tasks durch externe Interrupts
- Synchrone Aktivierung von Tasks durch *self-IPIs*

Sloth auf Intel x86: Prioritäten

- Verdrängung von Unterbrechungsbehandlungen basiert auf Prioritätsgruppen
  - Abbildung der Prioritäten auf Vektoreinträge



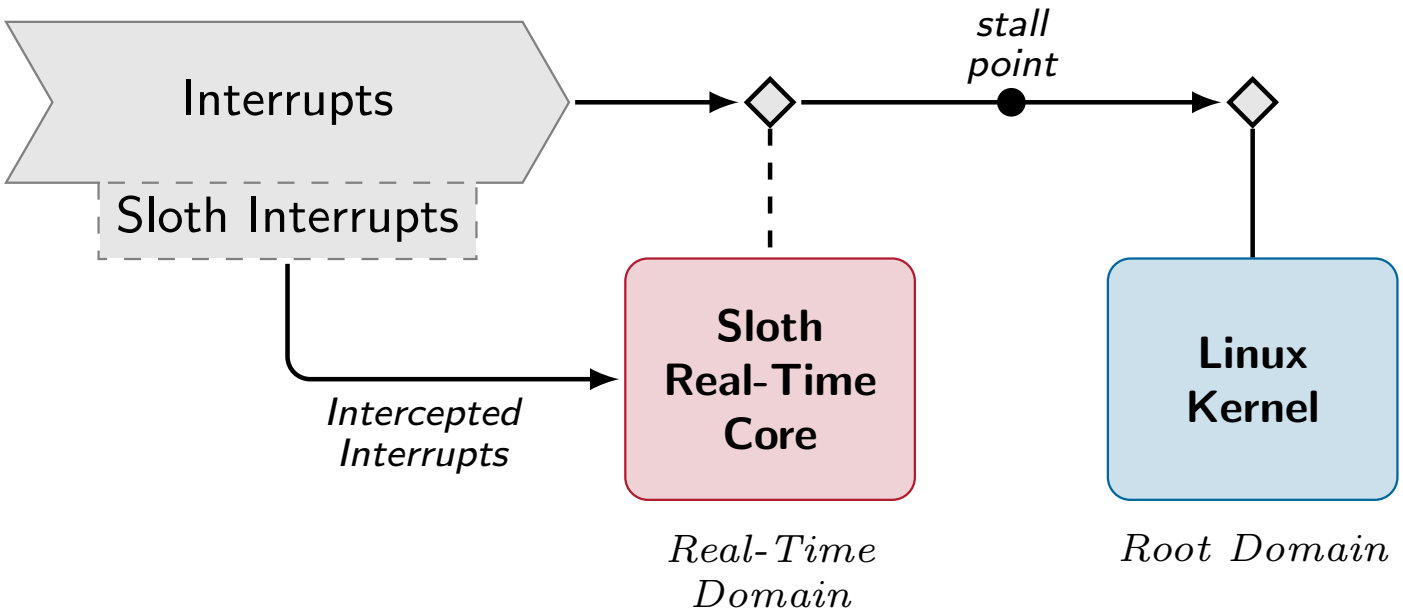
$\Rightarrow$ Sloth auf Intel x86 umsetzbar

Entwurf und Implementierung

- Zielplattform: Intel x86 APIC
- Sloth (bare-metal)
  - Ziel: Prüfung der Realisierbarkeit
  - Ausnutzung der Hardware ohne Einschränkungen
  - Funktionsumfang: Tasks, Resources
- Slothful Linux
  - Virtualisierung der Unterbrechungsbehandlung des Linux-Kernels
  - Anpassung des Linux-Kernels auf Basis vorhandener Patches
  - Kommunikationskanal zwischen Echtzeitanwendung und Prozessen unter Linux
  - Funktionsumfang: Tasks, Resources, Pipes

Slothful Linux: Veränderungen am Linux-Kernel

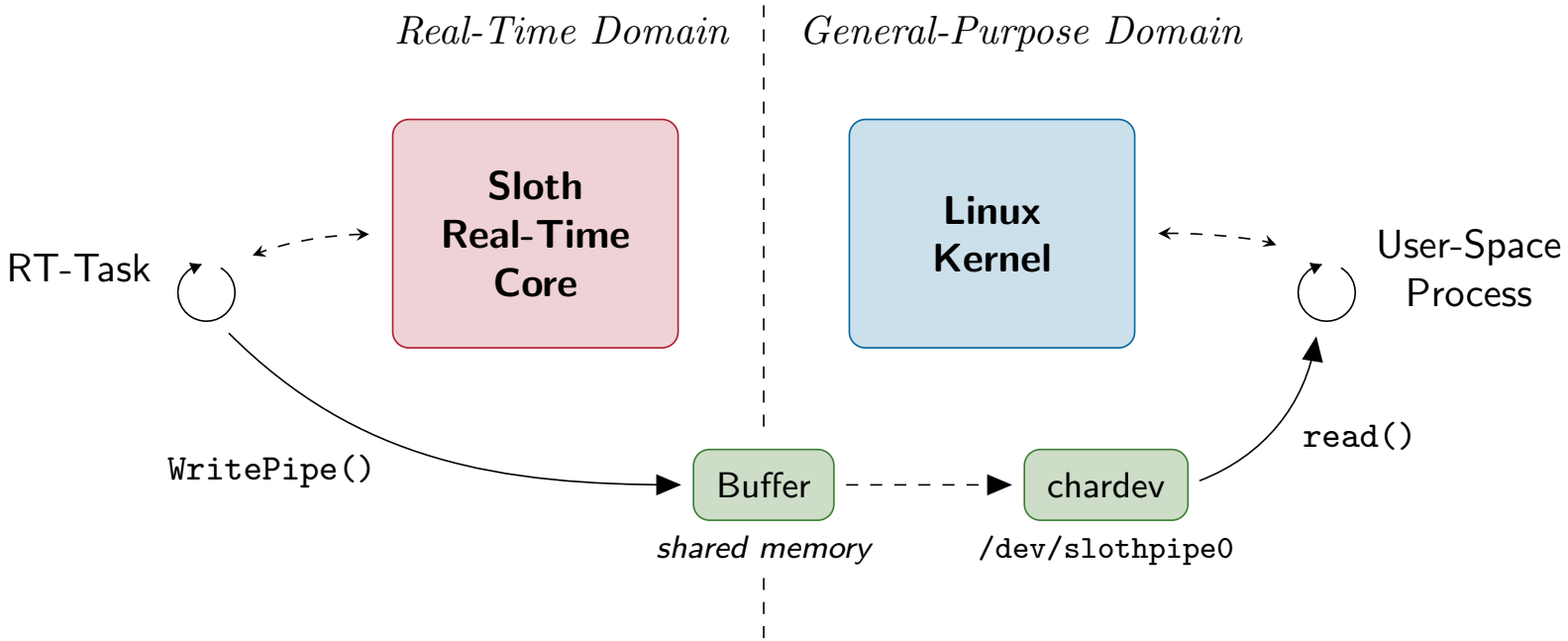
- Anpassung des I-Pipe-Patches
  - Funktionsumfang sehr komplex
  - bietet *Optimistic Interrupt Protection* für den Linux-Kernel



- Echtzeitsysteme als Module für den Linux-Kernel

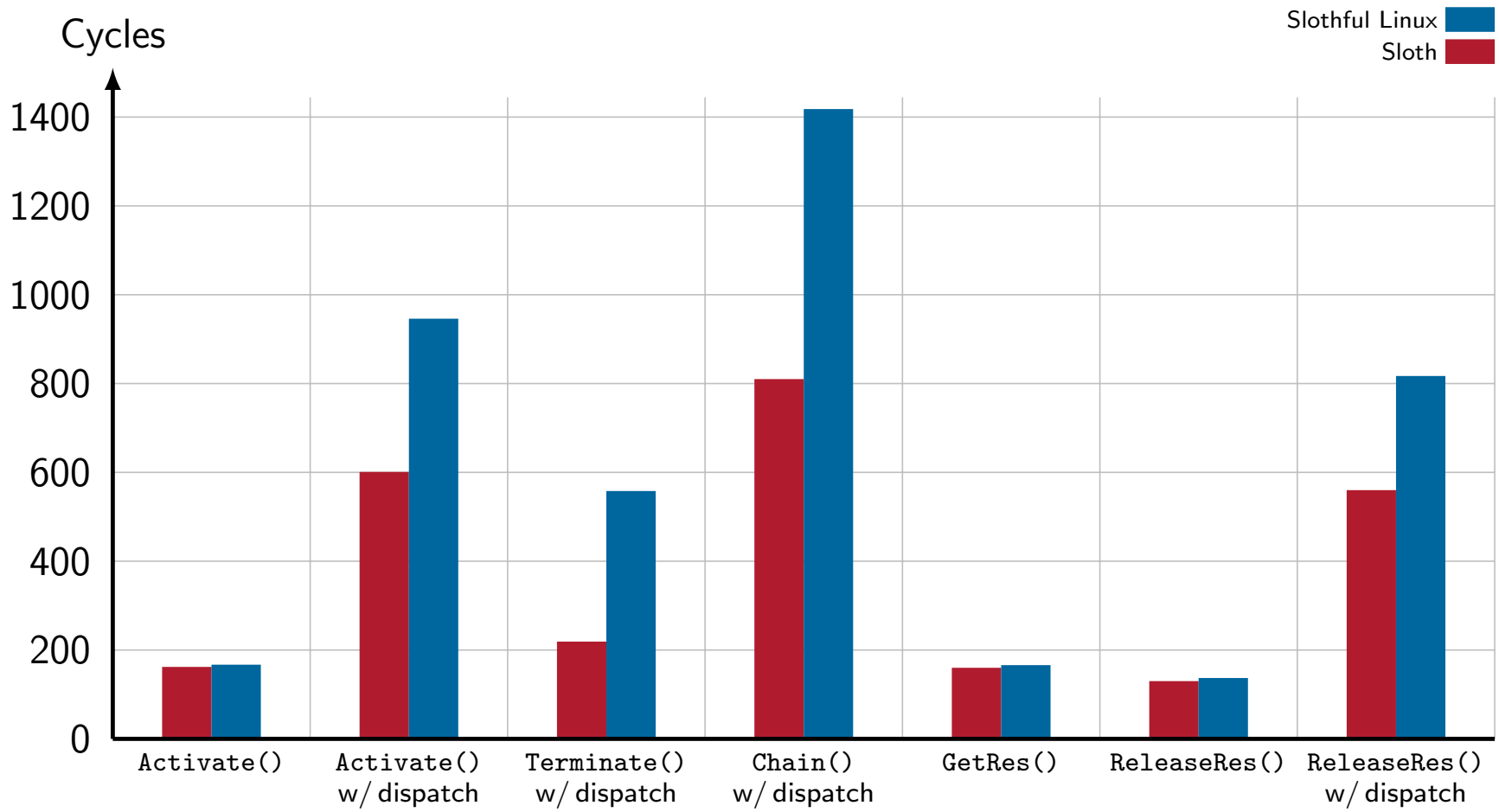
Slothful Linux: Pipes

- Pipes ermöglichen Datentransfer von Echtzeitanwendung zu Prozessen des Linux-Kernels



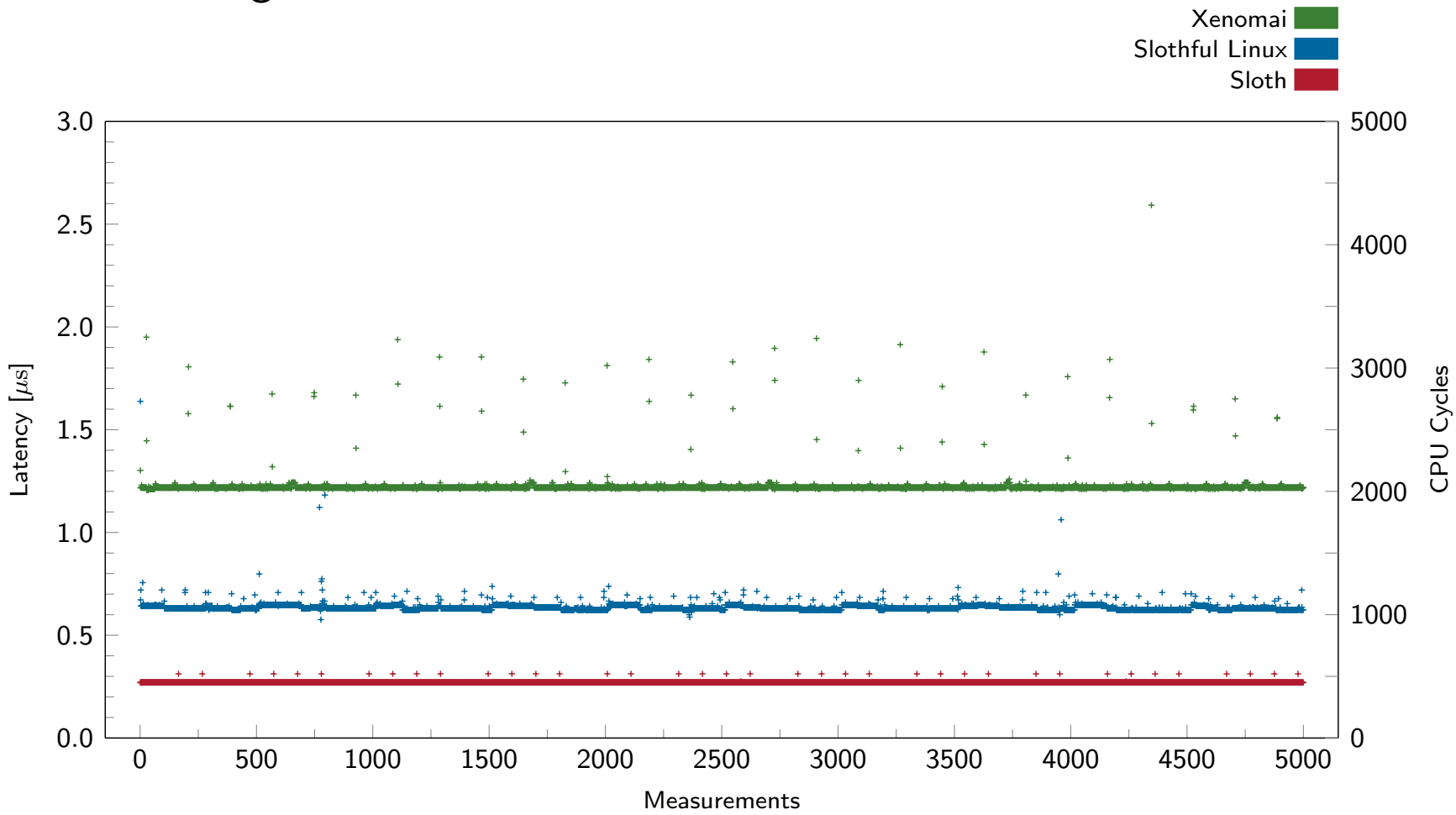
Evaluation: System Services

- Laufzeitmessung mit *Timestamp Counter* auf Intel Atom mit 1.6 GHz



Evaluation: Latenz

- Messung der Interrupt-Latenz mit Local-APIC-Timer
  - vom Zeitpunkt der Aktivierung bis zur ersten Instruktion der Behandlungsfunktion



Zusammenfassung

- Slothful Linux ermöglicht parallele Ausführung von Sloth und Linux auf derselben Hardware
  - Sloth läuft ohne Beeinträchtigungen durch Linux
  - konstanter zusätzlicher Aufwand für Wechsel zwischen den Domänen
  - niedrige Latenz für Verarbeitung von Unterbrechungen bzw. Aktivierungen von Tasks
- Ausblick:
  - Unterstützung von Multi-Core-Systemen

`http://www4.cs.fau.de/Research/Sloth`

Literaturverzeichnis

Wanja Hofer, Daniel Lohmann, Fabian Scheler, and Wolfgang Schröder-Preikschat.
Sloth: Threads as interrupts.
In *Proceedings of the 30th IEEE International Symposium on Real-Time Systems (RTSS '09)*, pages 204–213. IEEE Computer Society Press, December 2009.

Wanja Hofer, Daniel Lohmann, and Wolfgang Schröder-Preikschat.
Sleepy Sloth: Threads as interrupts as threads.
In *Proceedings of the 32nd IEEE International Symposium on Real-Time Systems (RTSS '11)*, pages 67–77. IEEE Computer Society Press, December 2011.

Wanja Hofer, Daniel Danner, Rainer Müller, Fabian Scheler, Wolfgang Schröder-Preikschat, and Daniel Lohmann.
Sloth on Time: Efficient hardware-based scheduling for time-triggered RTOS.
In *Proceedings of the 33rd IEEE International Symposium on Real-Time Systems (RTSS '12)*, pages 237–247. IEEE Computer Society Press, December 2012.