Sloth: Efficient Hardware-based Task Scheduling and Dispatching for the Automotive Domain

Rainer Müller, Wanja Hofer, Fabian Scheler,
Daniel Lohmann, Wolfgang Schröder-Preikschat

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

9. November 2012

Overview: Sloth

Main Idea

When designing an embedded kernel, embrace hardware peculiarities instead of blindly abstracting from them, in order to optimize non-functional kernel properties such as latency, memory footprint, and priority management.

Motivation: Rate-Monotonic Priority Inversion

- Problem:
  high-priority threads disturbed by low-priority ISRs
- Sloth approach:
  threads in the same priority space as ISRs; i.e., threads as interrupts

Sloth: Threads as Interrupts

- **Idea: implement threads as interrupt handlers**
- Thread activation by interrupt requests
- Let interrupt subsystem do the scheduling and dispatching work
- Applicable to priority-based real-time systems
- Advantage: small, fast kernel with unified control-flow abstraction

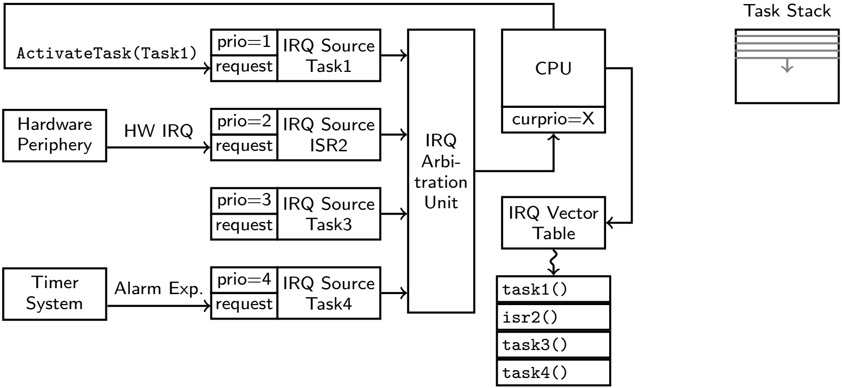
OSEK and AUTOSAR

- Standards developed by automotive industry
- AUTOSAR is the successor of OSEK
- Statically configured, event-triggered OS

- **Tasks** scheduled and dispatched by OS scheduler
- **Category-2 ISRs** can call system services, need kernel sync
- **Category-1 ISRs** cannot call system services, do not need sync

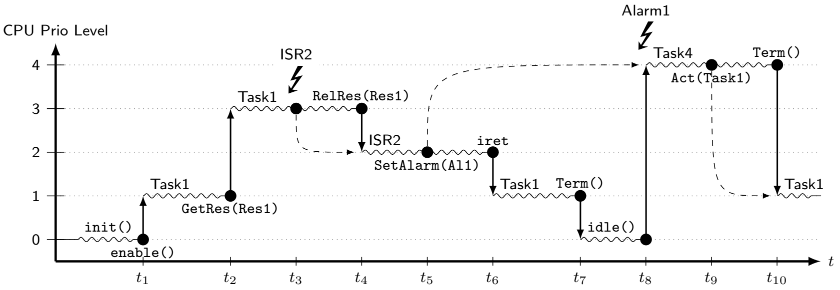
- **Resources** for application sync, with stack-based priority-ceiling protocol
- **Alarms** configured to activate task or execute callback upon expiry

Sloth Design



- Platform must support interrupt priorities
  and interrupts requests from software for synchronous task activation

Example Control Flow



Design Implications

- Concise kernel design and implementation (minimal system: < 200 LoC, < 700 bytes)
- Single control-flow abstraction for tasks, ISRs category 1/2, callbacks
  - handling oblivious to how it was triggered (by hardware or software)
- Unified priority space for tasks and ISRs, no rate-monotonic priority inversion
- Straight-forward synchronization by altering CPU priority
  - Resources with ceiling priority (also for ISRs!)
  - Non-preemptive sections with `RES_SCHEDULER` (highest task priority)

Implementations

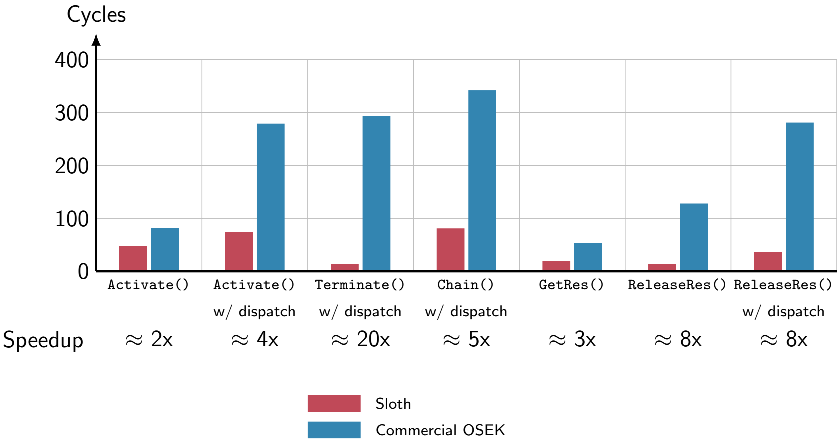
- Small kernel makes porting efforts easy
- Implementations for several platforms
  - Infineon TriCore (widely used in automotive systems)
  - ARM Cortex-M3
  - Freescale MPC56xx (embedded Power Architecture)
  - Intel Atom
- Configuring an application includes
  - mapping tasks to interrupt sources
  - assigning priorities to each interrupt source

Performance Evaluation: Methodology

- Evaluation of task-related system calls:
  - Task activation
  - Task termination
  - Task acquiring/releasing resource
- Performance of other system calls and application similar to traditional systems
- Comparison with performance of commercial OSEK implementation

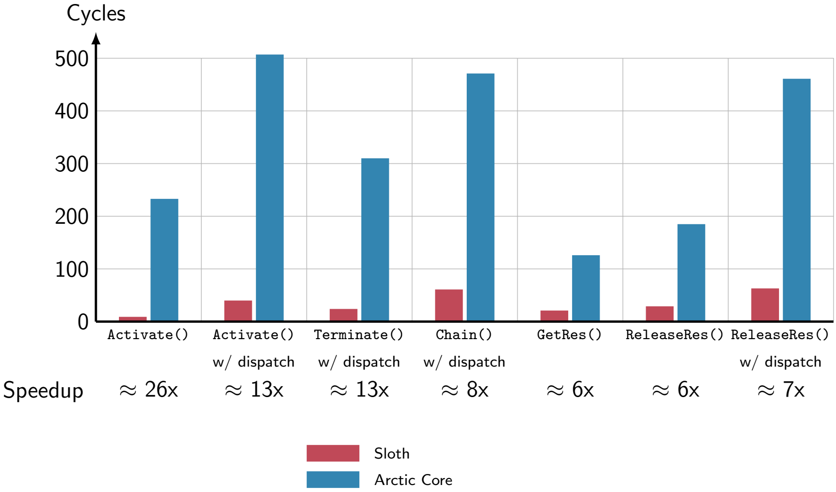
Performance Evaluation: Results

- Infineon TriCore



Performance Evaluation: Results

- ARM Cortex-M3



Further Work: The Sloth Family

- Sloth [RTSS '09]
  - support for basic tasks (OSEK BCC1)
- Sleepy Sloth [RTSS '11]
  - support for extended tasks (OSEK ECC1)
- Sloth on Time [RTSS '12]
  - time-triggered execution using hardware timer arrays (OSEKtime, AUTOSAR schedule tables)
- Slothful Linux
  - hybrid system running Linux and Sloth in parallel on the same hardware
- Multi-core Sloth
  - running on multiple cores (AUTOSAR multi-core OS specification)

Summary

- Sloth implements tasks by using IRQs and interrupt handlers on commodity hardware platforms
  - Makes tasks a low-overhead abstraction
  - Avoids rate-monotonic priority inversion
  - Keeps software footprint low

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

Bibliography

Wanja Hofer, Daniel Lohmann, Fabian Scheler, and Wolfgang Schröder-Preikschat.
Sloth: Threads as interrupts.
In Theodore P. Baker, editor, *Proceedings of the 30th IEEE Real-Time Systems Symposium (RTSS 2009)*, pages 204–213, Los Alamitos, CA, USA, 2009.

Wanja Hofer, Daniel Lohmann, and Wolfgang Schröder-Preikschat.
Sleepy Sloth: Threads as interrupts as threads.
In Luis Almeida and Scott Brandt, editors, *Proceedings of the 32nd IEEE Real-Time Systems Symposium (RTSS 2011)*, pages 67–77, Los Alamitos, CA, USA, 2011.

Wanja Hofer, Daniel Danner, Ranier Müller, Fabian Scheler, Wolfgang Schröder-Preikschat, and Daniel Lohmann.
Sloth on Time: Efficient hardware-based scheduling for time-triggered RTOS.
In Chenyang Lu, editor, *Proceedings of the 33rd IEEE Real-Time Systems Symposium (RTSS 2012)*, Los Alamitos, CA, USA, 2012.