Dual Operating System Architecture for Real-Time Embedded Systems

Daniel Sangorrín, Shinya Honda, Hiroaki Takada

Nagoya University 名古屋大学

Jul 6, 2010

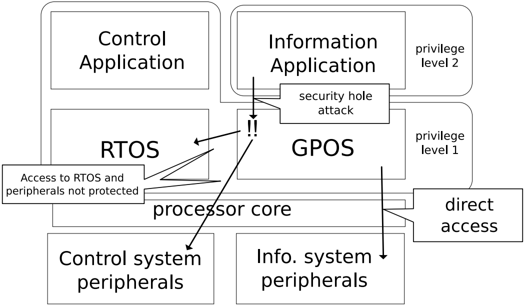
This presentation includes work done under the Monbukagakushou (文部科学省) scholarship funded by the Japanese government.

Outline

Virtualization for Real-Time Embedded Systems

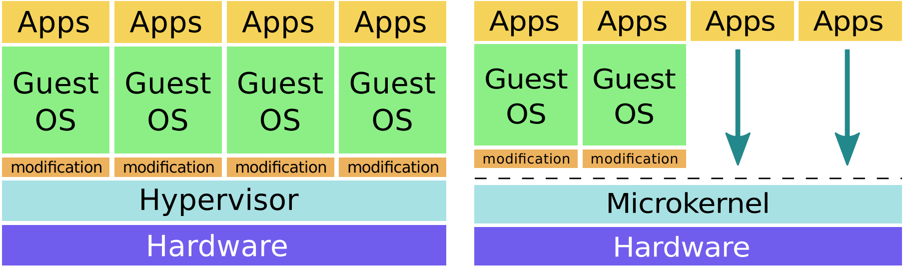
App: **Execute GPOS and RTOS applications on a single platform**

- GPOS kernel patches (e.g., Linux RT patch)
  - Soft Real-Time only, low security and reliability
- Hybrid kernels (e.g., Xenomai, RTAI, RTLinux, Linux on ITRON)
  - Hard Real-Time, native performance but no isolation



Virtualization for Real-Time Embedded Systems

- Hardware extensions (e.g., multicore)
  - Increased price and power consumption
  - Underutilization of RTOS core
- VMM/Hypervisors (e.g., OKL4, XtratuM, Integrity OS)
  - Good isolation with some overhead
  - Paravirtualization is hard to maintain

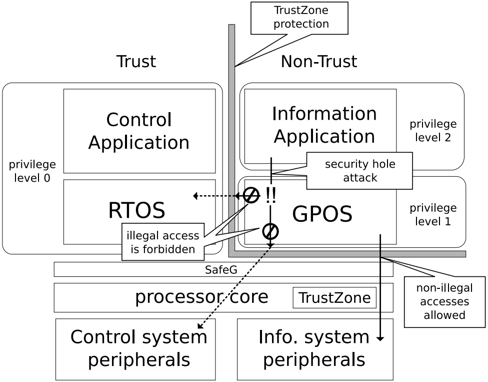


Virtualization challenges

- Modifications to the GPOS are difficult to maintain
- It is not possible to provide complete isolation
  - Bus masters as DMA or GPUs can bypass protections
  - Virtualizing them would severely damage performance
  - Hardware-assisted Virtualization
- Embedded virtualization requires Integrated Scheduling
  - Some GPOS tasks and interrupts require a certain QoS
  - Not all RTOS activities need high priority

ARM TrustZone

- System-wide approach to security (e.g., authentication, DRM)
  - Trust and Non-Trust states (orthogonal to privileges)
  - Monitor mode to switch between them
- ARM 1176 and Cortex-A series

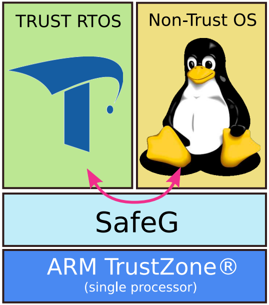


VMM requirements

- Support concurrent execution of a GPOS and an RTOS
- Spatial isolation of the RTOS
- Time isolation of the RTOS
- Integrated scheduling of GPOS soft-real time tasks and interrupts
- Mechanisms to implement health monitoring and device sharing
- No modifications to the GPOS core
- Minimum size. Easy to verify.

SafeG: Implementation of the TrustZone monitor

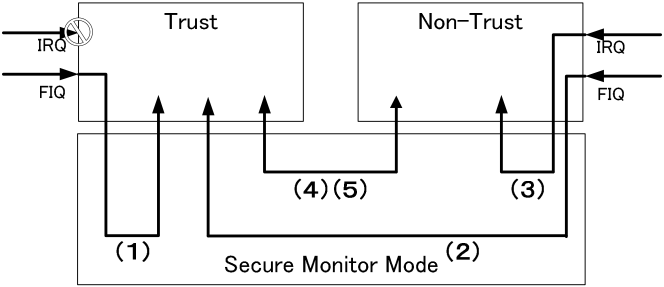
- Runs with interrupts disabled (FIQ and IRQ)
- Isolation: RTOS runs in Trust state, GPOS in Non-Trust state
- RTOS interrupts (FIQ) can not be disabled by the GPOS (IRQ)
- The GPOS is represented as an RTOS task
  - RTOS interface (e.g., μITRON) can be used on the GPOS



SafeG

Execution paths

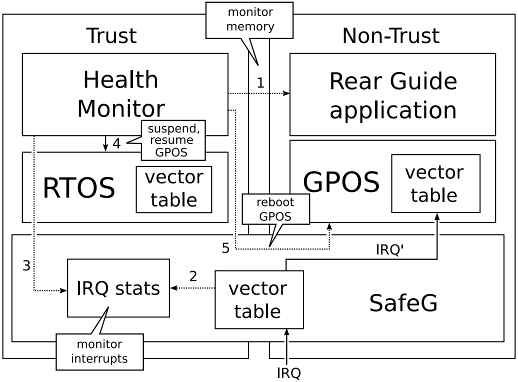
1. An FIQ occurs in Trust state
2. An FIQ occurs in Non-Trust state (SafeG switches to Trust state)
3. An IRQ occurs in Non-Trust state
4. SafeG switches state after an SMC call



SafeG

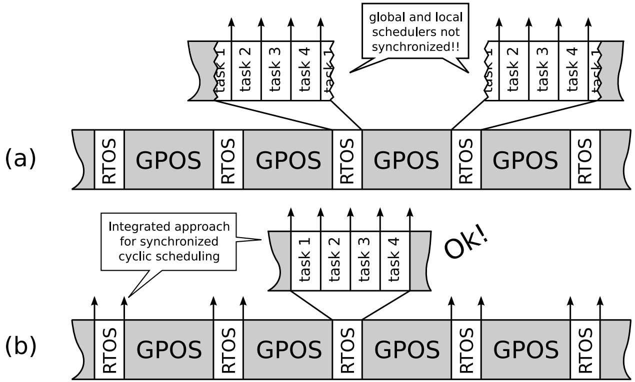
Health monitoring

- Mechanisms to monitor, suspend, resume and restart the GPOS



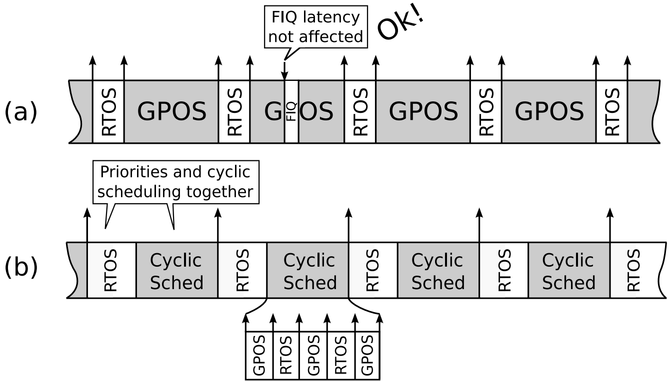
Black box vs. Integrated cyclic scheduling

- Synchronization of internal and global scheduler



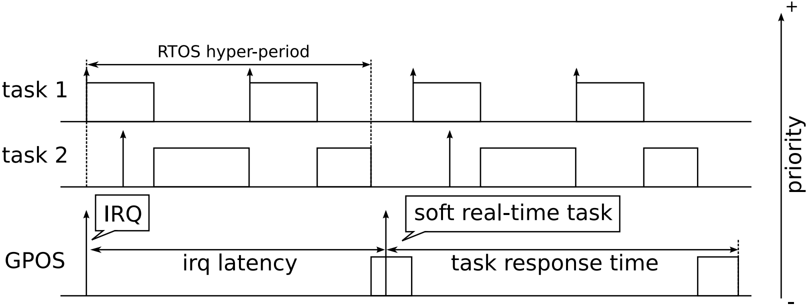
Latency in integrated cyclic scheduling

- FIQ interrupts and High priority tasks



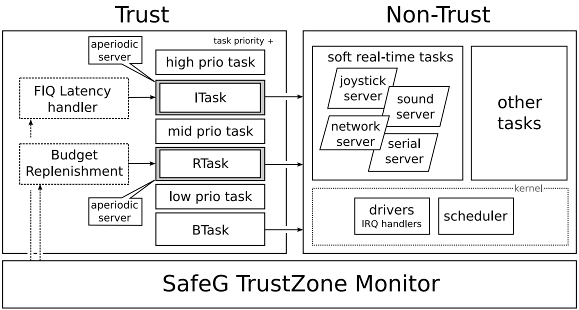
Idle approach

- GPOS interrupts and tasks scheduled as RTOS idle task
- Long latencies (e.g., IRQ handlers)

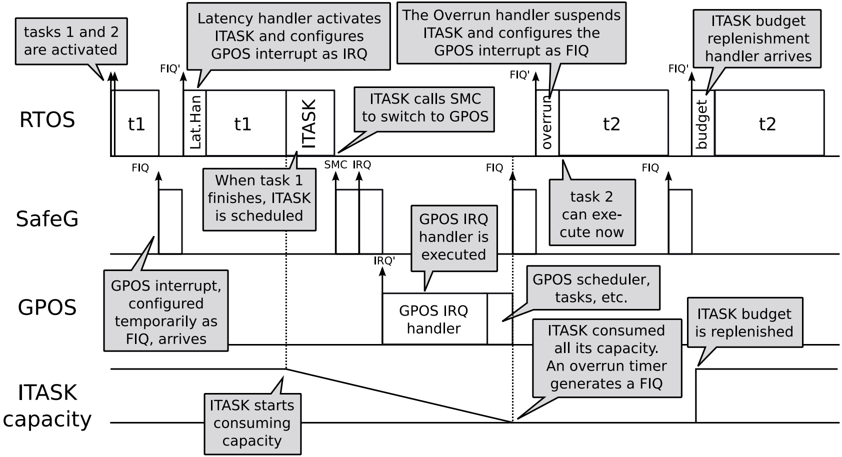


ITask-RTask-BTask approach

- ITask: GPOS interrupts latency
- RTask: Gives a QoS to GPOS (budget-period)
- BTask: like Idle approach



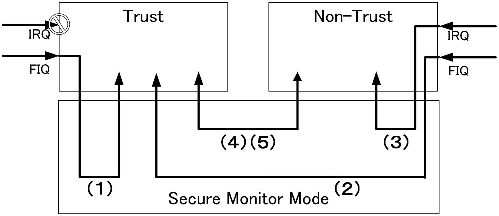
ITask-RTask-BTask Timeline



Implementation

- Platform:
  - ARM PB1176JZF-S (210Mhz, 128MB, 32KB Cache)
- RTOS: TOPPERS/ASP
  - Added overrun handlers (for deferrable servers)
  - Implemented TrustZone device drivers
- GPOS: GNU/Linux
  - High Vector table (0xFFFF0000)
  - Memory and devices allocation

SafeG overhead



Path	WCET
(1) While RTOS runs FIQ occurs	0.7μs
(2) While GPOS runs FIQ occurs	1.6μs
(3) While GPOS runs IRQ occurs	1.2μs
(4) Switch from RTOS to GPOS	1.5μs
(5) Switch from GPOS to RTOS	1.7μs
From ASP IRQ vector until IRQs enabled	5.1μs

SafeG code verifiability

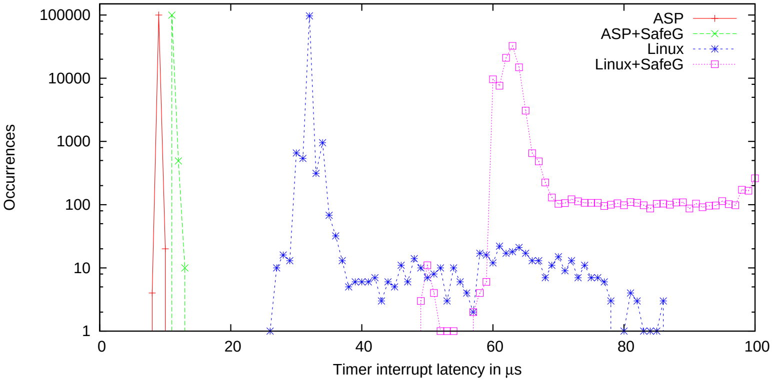
- Code and data size (in bytes)

	text	data	bss	total
SafeG	1520	0	448	1968
ASP	34796	0	83140	117936
Linux	1092652	148336	89308	1330296

- Safeg size is 1/60 of the size of ASP
- 304 bytes in .bss are just for the context
- 4 forks in total: only 8 types of tests needed

RTOS isolation

- Latency of the ASP and Linux system timer interrupt
  - ASP timer interrupt latency increased 2us (bounded)



ITask experiment

- Measure the Serial driver interrupt latency on Linux
- RTOS tasks:

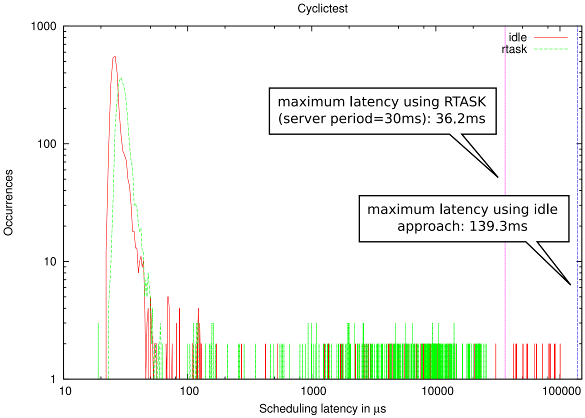
task	priority	period	duration	utilization
1	high	50ms	10ms	20%
2	low	300ms	100ms	33%

- ITask period: 30ms, budget: 2ms
- Serial driver latency (in μs):

approach	min	avg	max
alone	15.7	15.81	19.47
idle	14.6	22681	113833
itask	15.45	2292	30275

RTask experiment

- Execute the cyclictest program in the GPOS
  - Periodic thread that measures the wake up latency



Conclusions

- SafeG
  - A reliable dual hypervisor for embedded real-time systems
- VM Integrated Scheduling
  - Cyclic scheduler
  - ITask-RTask-BTask approach
- ARM TrustZone security extensions
  - Useful for virtualization
  - Proposal: Cache separation
  - Proposal: Instruction for context switch

Future work

- Refine Integrated Scheduling with voluntary return
  - Fine-grained control of tasks and interrupts
  - May require GPOS core modifications
- Android on the Non-Trust side
- Inter-VM communications
- Multi-core porting (Cortex-A9)

Questions

Thank you for your attention
ご清聴ありがとうございました