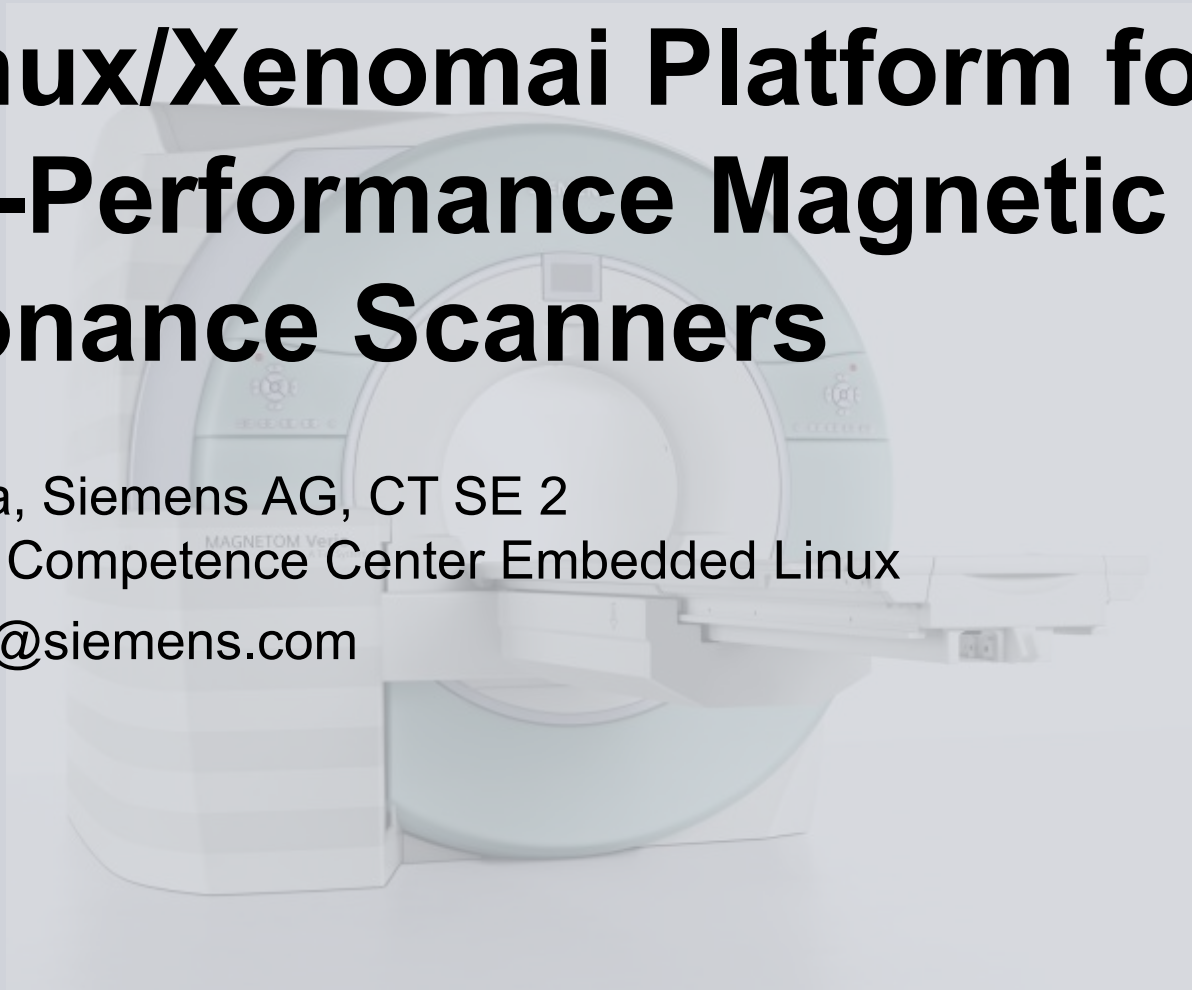


A Linux/Xenomai Platform for High-Performance Magnetic Resonance Scanners

A large, light blue and white magnetic resonance scanner (MRI) machine is shown in the background, slightly faded. The machine has a large circular opening in the center and various components around it. The text "MAGNETOM Ver" is visible on the side of the machine.

Jan Kiszka, Siemens AG, CT SE 2
Corporate Competence Center Embedded Linux
jan.kiszka@siemens.com

Corporate Competence Center Embedded Linux

	pre-product phase	development phase	test phase	maintenance phase
Training	Basic know-how			
Consulting	Directed to target product			
Prototyping, development	Direct responsibility for components			
Evaluation, benchmarking	Support critical design decisions			
OSS - Support, consulting	Architecture, OSS-scanning, release notes, community feedback			

Linux services covering the entire product life cycle



Contact
Oliver Fendt
Tel.: + 49 (89) 636 46033
Oliver.Fendt@siemens.com

Linux, is the ideal platform, which facilitates the cost effective development of complex smart embedded devices. We support you during the entire product life cycle in all activities of embedded Linux development. Our embedded Linux team members are internationally recognized and have extensive product development experience.

Benefits

Maximum benefit from Linux in the specific project context:

- robustness
- cost reduction
- time to market

Our Offer

- Own publicly available & supported embedded Linux Distribution
- Embedded Linux training
- Consulting:
 - Support on legal issues
 - Platform strategy consulting
 - Development environment set up and maintenance
- Development:
 - Kernel & driver development
 - Application development
- Evaluation:
 - Open Source community interaction & quality assessment
 - Profiling and performance tests

Agenda

- **Introduction MRI**
- **MR scanners**
 - Achitecture
 - Requirements
- **RT Platform selection**
- **Xenomai experiences**
- **Community contributions**
- **Conclusion**

Magnetic Resonance Imaging (in a Nutshell)

Strong static magnetic field

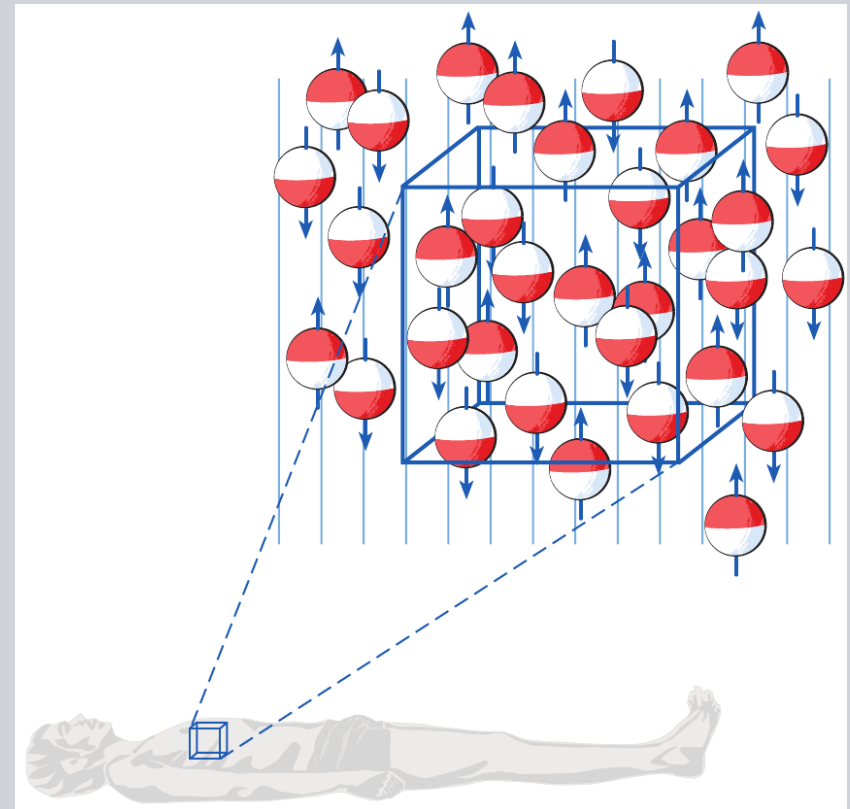
- Causes alignment of nuclei in the body

Pulsed radio field

- Disturbs alignment
- Nuclei emit electromagnetic signal when field is removed
- Signal decay depends on body tissue

Gradient field

- Linear magnetic field variation
- Generates echo signal too
- Combination with RF field enables tissue localization



MRI Scanner – Main components

Measurement controller

- Medium computing demands
- Time-critical

MR data acquisition system

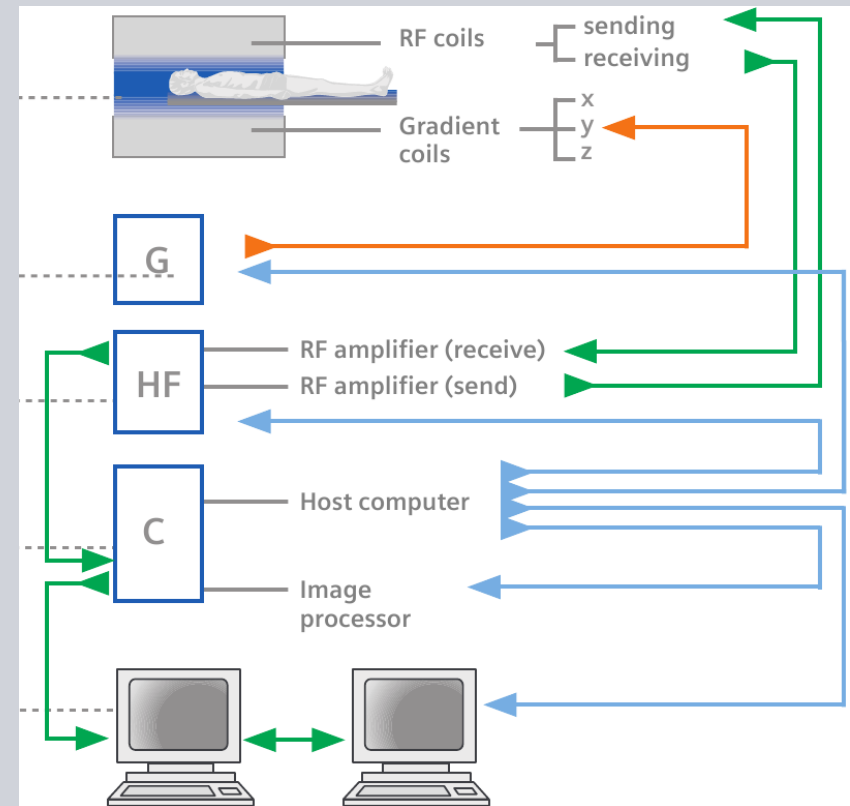
- High data rates

Image processor

- High computing demands
- Challenge: Reconstruct high-resolution images in real-time

Front-end

- High-level controller
- Visualization / user interface
- Hospital IT integration
- ...



Siemens MRI Scanner Architecture (Excerpt)

Measurement and Reconstruction System (MARS)

- High-end multicore 64-bit x86 system
- Large RAID disk array
- DAQ interfaces for MR data (PCI adapters)
- I/O interfaces
 - via TCP/IP over Ethernet
 - via UART
- Linux server distribution as platform
- Applications
 - Measurement control (RT jobs)
 - Image reconstruction (batch jobs)

Host System

- MS-Windows Workstation
- *syngo* software platform

Status Quo in 2007

MARS based on SLERT 10 with Concurrent extensions

- Soft-RT requirements, widely met
- Troubles with existing software stack
 - Kernel suffered from Linux RT limitations like
 - mmap semaphore vs. futexes
 - RT-incapable drivers
 - ...
 - RT-aware application design tricky
 - Designed for classic isolated RTOS, not Linux
 - Unsafe Linux services in RT loop hard to detect

Upcoming scanner platform

- Hard RT requirements
- Code inside RT loop will grow
- Increased bandwidth and computing requirements
=> large NUMA multicore systems

Real-Time Requirements

Measurement control loop

- Sample scanner states
- Sample patient signals (ECG)
- Generate measurement commands
- Generate table positioning commands
- Repeat with millisecond-range period time

MR data acquisition

- Relaxed real-time requirements
- Decoupled via large buffers
(DAQ buffer, RAM and hard disk)
- Achievable on standard Linux kernels

Selecting Real-Time Linux (2007)

Former RT technology of SLERT (Concurrent extensions)

- Only available for few kernel versions
- Single-file kernel patch => problematic maintenance
- Dead-end on mid-term
- Novell switched to Preempt-RT later

Preempt-RT

- Easy to migrate to
- Promising early test results
- But problems on NUMA boxes
- Community versions not yet ready for prime-time

Xenomai

- Clean cut between RT and non-RT domain
- Well-maintained stable series
- Known to work well on smaller boxes (1..4 way),
larger systems to be explored

Preempt-RT or Xenomai – The Decision

Arguments for/against both

- Standard tracing/debugging tools available / easily portable
- Kernel bugs expected due to challenging setup (large SMP, NUMA, brand-new hardware, ...)
- Community-driven projects

Key arguments for Xenomai

- Strong RT/NRT separation: expected to affect application positively
- Many RT design violations cause verbose alarms
- Less impact on NRT load, specifically on I/O load

Porting was challenging

- RT and NRT application components required more untangling than expected
- Additional need for fast user space mutexes
- Fast mutexes forced switch to Xenomai development branch

Strong domain separation helps developers

- Warn-on-switch support quickly adopted
- Back-traces help to find obvious violations
- Driver developers obtain kernel warnings/bugs early
- Increase of RT-awareness

Tracing support essential

- LTTng recommended for system-level analysis
- Continuously running, freeze on application error

gdb use problematic

- Unusual Linux scheduling order due to Xenomai
- Revealed bugs in nucleus and Linux signal handling
- Still triggers bugs in gdb and/or Linux kernel (ongoing analysis)
- Developers: “gdb doesn't work!”
So they use different “tools”
- It's hard to debug a debugger in the field...

Valgrind support for Xenomai

- High on the wish list
- ...but not quickly realizable

Project-driven Community Contributions

LTTng support

- Xenomai nucleus and RTDM instrumentation
- I-pipe over LTTng port and maintenance (x86)
- Merge planned once LTTng is upstream

xnsynch_fast – fast user space mutexes

- Generalization and enhancement of existing POSIX support
- Native skin support
- Lengthy but very helpful public discussions
- Feature now matured and broadly used

Xenomai scalability enhancement

- Restrict nucleus to CPU sub-set
- Works around nklock scalability limit
- Nicely fits “small RT / large NRT” scenarios
- Smarter solution required in the future

Project-driven Community Contributions (2)

I-pipe fixes & enhancements

- SMP issues (e.g. switch_mm locking fix)
- Linux IRQ migration
- Domain state checks (e.g. to detect RTDM driver bugs)

RTnet enhancements

- Real time TCP support
- Intel IGB (82575) driver
- To be merged upstream soon

Conclusion

- Xenomai ready for high-performance platforms
- Strong real-time / Linux separation helps application and driver developers
- Carefully estimate porting effort of existing hybrid (mixed RT/NRT) applications
- Work **within** community pays off
 - Less need for private patch queues
 - Broader test coverage helps maturing
- Tool support good, but still improvable
 - gdb needs more work
 - We are all waiting for LTTng in mainline
 - Valgrind would be nice

Thank you for your attention!

