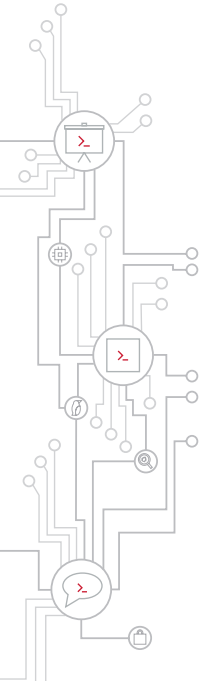


Realtime Linux Beyond PREEMPT_RT

Exploring Xenomai's Dual-Kernel Approach

Richard Weinberger

2025-08-25

Hello

Richard Weinberger

- Co-founder of sigma star gmbh
- Linux kernel developer and maintainer
- Strong focus on Linux kernel, low-level components, virtualization, security, code audits

sigma star gmbh

- Software Development & Security Consulting
- Main areas: Embedded Systems, Linux Kernel & Security
- Contributions to the Linux Kernel and other OSS projects

Goals and Non-Goals of this Talk

Goals:

- Add another tool to your toolbox
- Provide the knowledge to get started with Xenomai
- Show an alternative way to achieve realtime with Linux

Non-Goals:

- Promoting PREEMPT_RT as the ultimate solution
- Promoting Xenomai as the ultimate solution
- Showing off with results from cyclictest or similar tools

Realtime OS

- Deterministic
- Meets deadlines (within a reasonable time)
- It's *not* about speed in terms of throughput
- Usually you *don't* want a general-purpose OS as a realtime OS

Teaching Linux Realtime: Before PREEMPT_RT was merged

- Linux is a general-purpose OS
- Focus is mostly on throughput
- But it implements POSIX Realtime extensions
  - Just the API
  - Does not enforce it
  - “soft realtime”

Teaching Linux Realtime

- Option 1: Just don't
  - Run your realtime workload somewhere else
  - Many SoCs have a remote processor
  - Keep things simple
- Option 2: Turn Linux itself into a realtime OS: PREEMPT_RT
- Option 3: Tame Linux using a dual/co-kernel: Xenomai

PREEMPT_RT in a Nutshell

- Exploits Linux's preemption feature to the maximum
- If running code is at *any* point in time preemptable, you can have realtime
- Interrupts run as regular kernel threads
- IRQ-disabled and preempt-disabled sections are minimal
- Spinlocks become RT mutexes
- RT mutexes have owners and allow priority inheritance
- Optimizations all over the kernel (RCU, fast scheduler, ...)
- 25 years of work, mainline since v6.12

Xenomai

- “Xeno”: From Ancient Greek ξένος (xénos, “alien”)
- “mai”: To make it sound pleasant
- Built on top of an interrupt pipeline
- Dual-kernel approach: Cobalt kernel inside Linux

Interrupt Pipeline

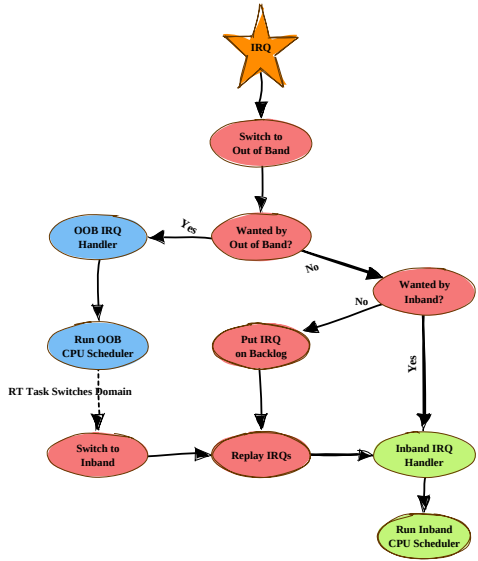
- Interrupts are *the* event source of computers
- Critical regions disable interrupts for a certain amount of time
- Linux has many critical regions
- No interrupts -> no timer tick -> no task scheduling
- The interrupt pipeline makes sure that the out-of-band core *always* gets interrupts
- Inband core can only virtually disable interrupts

Interrupt Pipeline (cont'd)

- The interrupt pipeline is the foundation to build something like Xenomai
  - It's generic!
- Xenomai sits on top of the interrupt pipeline
- Old implementation: ipipe patch
  - head stage: Xenomai (actually Cobalt or Nucleus)
  - root stage: Linux
- Current implementation: dovetail patch
  - out of band (OOB): Xenomai
  - inband: Linux

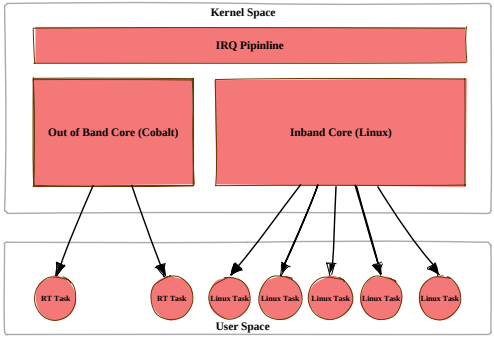
Interrupt Pipeline (cont'd)

- Inband can no longer disable interrupts
- e.g. `local_irq_disable()` in inband (Linux) only *virtually* disables interrupts for Linux
  - out of band (Xenomai) will still receive them
  - Interrupts can be logged and replayed so inband won't lose events
- This allows inband to have working critical regions
- But out of band still receives all interrupts and can provide realtime



Two Worlds

- Two worlds: Realtime and non-realtime
- Tasks can transition between both worlds
- Goal: Stay in the realtime world
- From Xenomai's POV: primary vs. secondary domain
- From Dovetail's POV: out of band vs. inband



Two Worlds: But Still Very Close

- Not a hypervisor: Xenomai and Linux share a lot of code
- It's common to have realtime and non-realtime threads in the very same process
- But: Realtime tasks must only use Xenomai services
- Otherwise, the task will be migrated to the secondary domain and the realtime guarantee is lost!

Two Worlds: The Key to Success

- As long as a thread stays in the primary domain, all is good
- Otherwise, no realtime guarantee anymore!
- Realtime threads must not use Linux services
  - Means: No Linux system calls!

Two Worlds: Xenomai services

- Own set of system calls
- Own set of device drivers
- Own CPU scheduler
- IPC to interact with other tasks (Linux and realtime!)

Domain Switch Detection

- A realtime thread is signaled upon a switch
- The signal contains the reason for the switch, e.g.
  - Use of a Linux system call
  - Page fault
  - Runaway (thread busy-loops)
  - Priority inversion
- Killer feature

In Contrast: PREEMPT_RT

- Remember: PREEMPT_RT makes sure that the kernel can always preempt
- So, when the realtime task is *runnable*, the scheduler can schedule it
- What if the realtime task is not runnable? No more realtime for you!
  - e.g. The task blocks while doing I/O or a driver misbehaves
- Runtime Verification helps, rather new feature
- With Xenomai, as long you stay in primary domain, all good!

Using Xenomai: Kernel

```
1 $ git clone -b v6.16-dovetail1-rebase https://gitlab.com/Xenomai/linux-
      dovetail.git
2 $ git clone -b v3.3.1 https://gitlab.com/Xenomai/xenomai3/xenomai.git
3 $ cd linux-dovetail
4 $ ../xenomai/scripts/prepare-kernel.sh
5 $ make defconfig
6 $ make -j `nproc`
```

Using Xenomai: Boot Messages

- `[ 0.081817] IRQ pipeline enabled`
  - Interrupt Pipeline is ready
- `[ 0.556421] IRQ pipeline: high-priority Xenomai stage added.`
  - Xenomai registered itself

Using Xenomai: Userspace

```
1 $ cd ../xenomai
2 $ scripts/bootstrap
3 $ ./configure
4 $ make -j `nproc`
5 $ make install
```

- /usr/xenomai/bin/latency
- /usr/xenomai/demo/cyclictest
- Add /usr/xenomai/bin/ to your PATH
- Add /usr/xenomai/lib/ to ld.so search dir
- ...or set a sane --prefix while running configure :-)

Skins

- Xenomai offers multiple userspace APIs
- Alchemy
- POSIX
- VxWorks
- pSOS

Using Xenomai: Hello World

```c
#include <alchemy/task.h>
static RT_TASK hello_task;

static void hello(void *arg)
{
    rt_printf("Hello␣from␣Xenomai!\n");
}

int main(int argc, char *argv[])
{
    int err = rt_task_create(&hello_task, "hello_task", 0, 50, T_JOINABLE);
    if (err) return err;

    err = rt_task_start(&hello_task, &hello, NULL);
    if (err) return err;

    rt_task_join(&hello_task);
    return 0;
}
```

Using Xenomai: Hello World (cont'd)

```
1  $ gcc -o hw $(xeno-config --alchemy --cflags --ldflags) -Wall hw.c
2  $ ./hw
3  $ Hello from Xenomai!
```

Using Xenomai: Hello World (POSIX Skin)

```c
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>

static void *hello(void *arg)
{
    printf("Hello from Xenomai!\n");
    return NULL;
}

int main(int argc, char *argv[])
{
    pthread_t thread;
    struct sched_param param;
    pthread_attr_t attr;

    pthread_attr_init(&attr);
    pthread_attr_setinheritsched(&attr,
        PTHREAD_EXPLICIT_SCHED);
    pthread_attr_setschedpolicy(&attr,
        SCHED_FIFO);
    param.sched_priority = 50;
    pthread_attr_setschedparam(&attr, &param);

    int err = pthread_create(&thread, &attr,
        hello, NULL);
    if (err) return err;

    pthread_join(thread, NULL);

    return 0;
}
```

Using Xenomai: Hello World (cont'd)

```
1  $ gcc -o posix-hw $(xeno-config --posix --cflags --ldflags) -Wall posix-
       hw.c
2  $ ./posix-hw
3  $ Hello from Xenomai!
```

Device Drivers

- Xenomai has its own realtime-capable device drivers
- RTDM (Real Time Device Model)
- Drivers are usually a stripped-down copy of Linux drivers
- Drivers for many Ethernet, CAN, UART, SPI devices exist
- UDD (Userspace Device Drivers)
- Huge maintenance burden

Latency Numbers

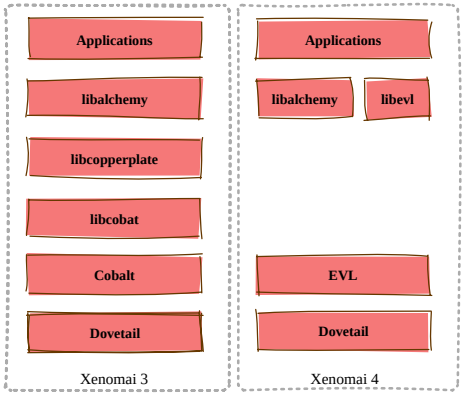
- The slide you've all been waiting for:
  - Xenomai shows much better results at `cyclictest` !!!11elf
  - No.
- cyclictest measures wakeup latency
- An excellent metric to profile a realtime OS and find latency issues
- cyclictest shows similar results on both PREEMPT_RT and Xenomai
- Your realtime application most likely has a different workload than cyclictest
- Benchmark *your* workload, including:
  - Interaction with devices
  - Interaction with other tasks/processes

Common Enemy: Hardware-Inflicted Latencies

- The co-kernel approach won't help you there
- SMI
- Shared caches
- Sleep states
- All of them hurt you just like with PREEMPT_RT

Future Outlook

- All slides so far apply to Xenomai v3
- EVL: Minimal re-implementation of Cobalt
  - Not compatible with Xenomai 3
  - New ideas, fresh code
- libevl: C API to EVL core
- revl: Rust API to EVL core
- No more duplicated device drivers!
  - Linux drivers get an out-of-band mode
- Xenomai 4: libevl plus a compat layer to provide the Xenomai “feeling”



Summary

- PREEMPT_RT is fine *and* mainline
- Depending on the design of your RT application, Xenomai can provide better results
- The co-kernel approach allows you to circumvent Linux completely
- The price is high but sometimes worth it
- Think outside the box: there are more ways to achieve realtime
- Xenomai is fun to work with and quite hackable

Xenomai Community

- www.xenomai.org
- https://gitlab.com/Xenomai/xenomai3/xenomai/-/issues
- xenomai@lists.linux.dev
- Bi-weekly comminity call, announcted on mailinglist

FIN

Thank you!

Questions, Comments?

Richard Weinberger
richard@sigma-star.at