LINUTRONIX
LINUX FOR INDUSTRY

Realtime Linux uncovered

Jan Altenberg

Linutronix GmbH

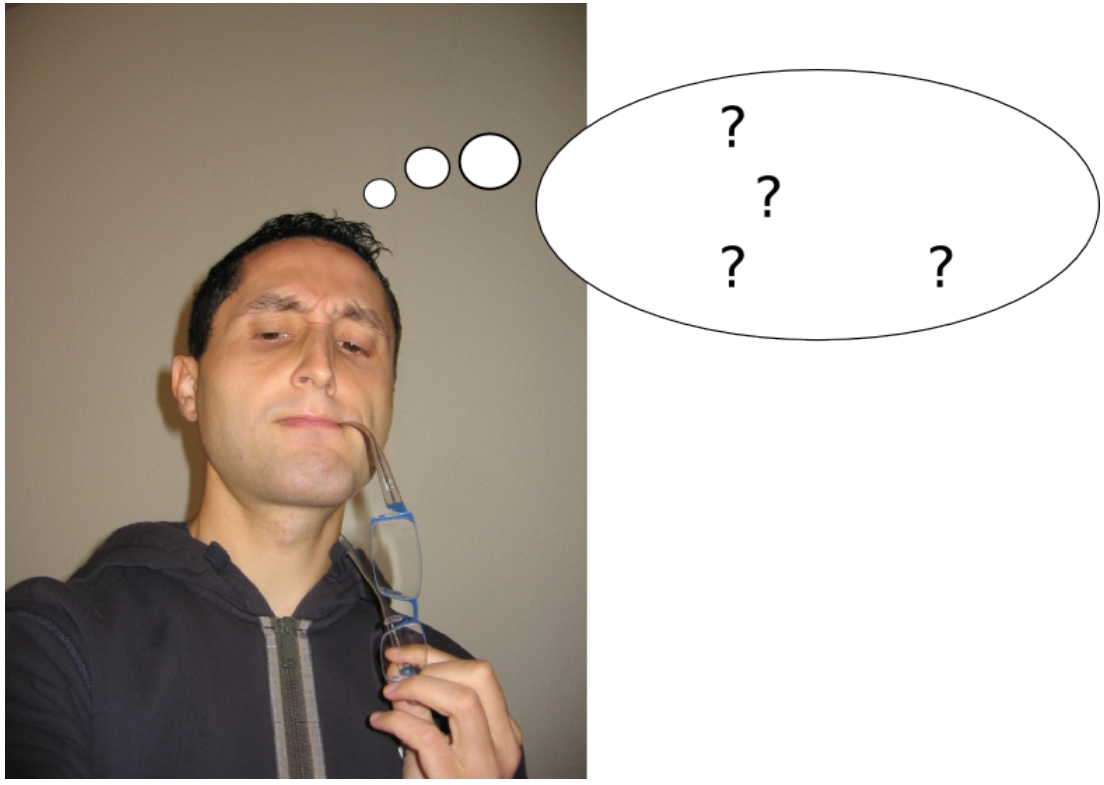
LINUTRONIX
LINUX FOR INDUSTRY

Overview

1. What is Realtime?
2. Linux and Realtime - History and approaches
3. Preempt RT
4. Results: Which latencies can be achieved with the different approaches?
5. Conclusion

What is Realtime?

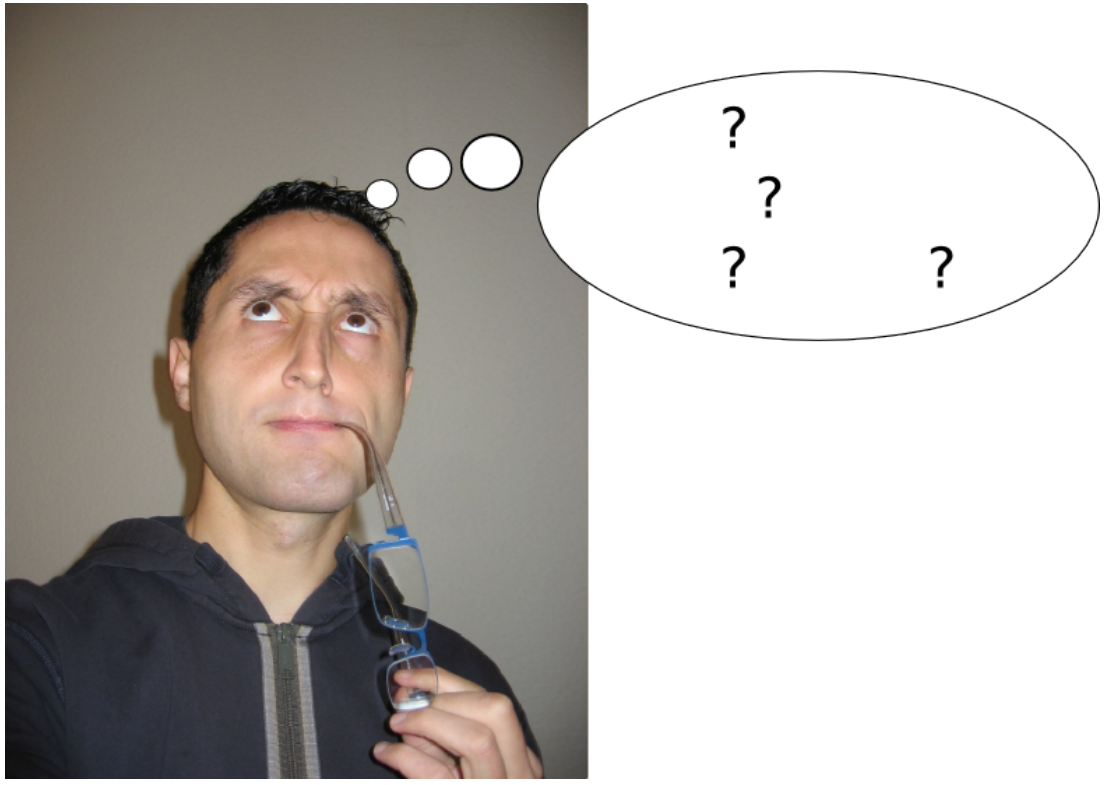
Fast execution time?



LINUX PITER

What is Realtime?

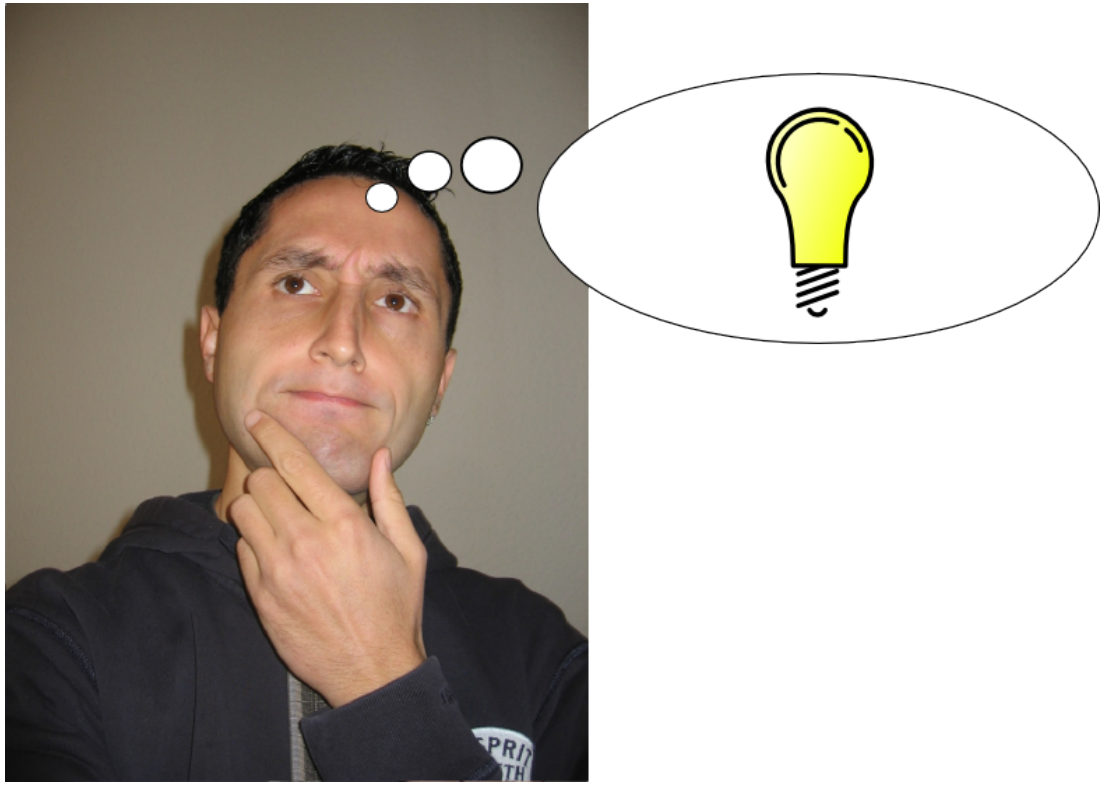
Performance?



LINUX PITER

What is Realtime?

It's all about DETERMINISM!



LINUX PITER

What is Realtime?

- Correctness means execution at the correct time
- Missing the timeslot will lead to an error condition

Realtime

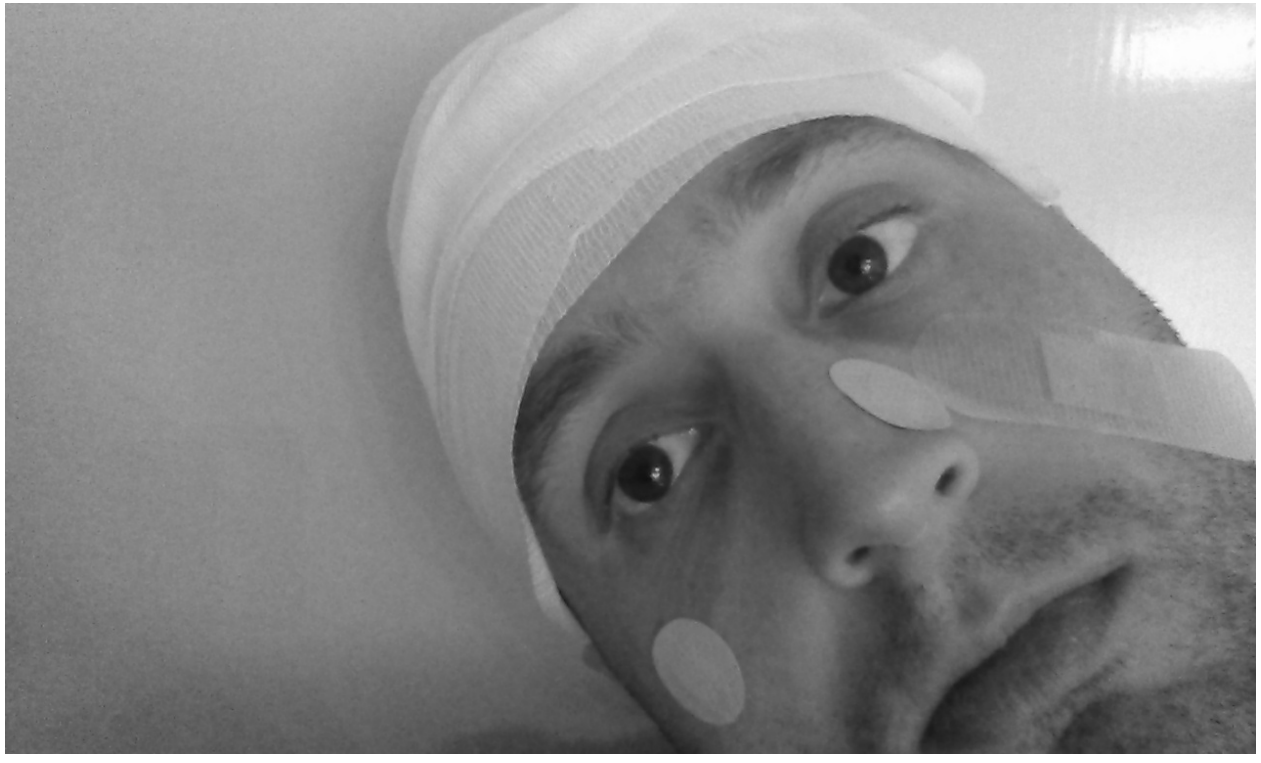
Remember!

Missing the timeslot will lead to an error condition

Realtime

Missing the timeline

will cause a damage to your machine or even a person might get hurt:



LINUX PITER

What about "Softrealtime"?

...PLEASE PLEEAAASSEEE forget about this word!!! :)

Who is using it?

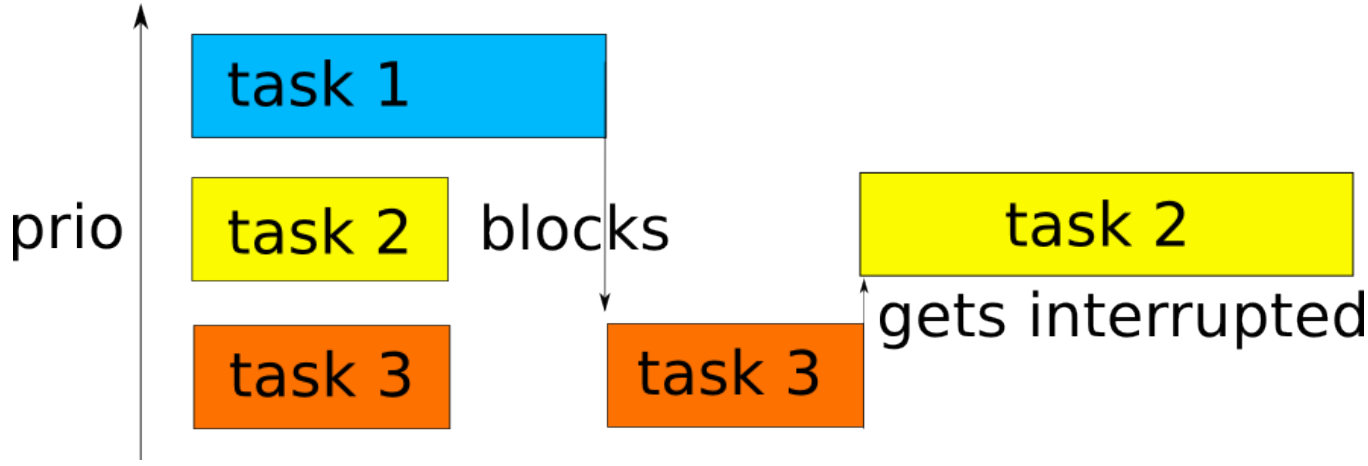
- industry / automation
- multimedia systems
- aerospace
- financial services
- ...

LINUX PITER

Requirements

- Deterministic timing behaviour
- Preemption
- Priority Inheritance / Priority Ceiling

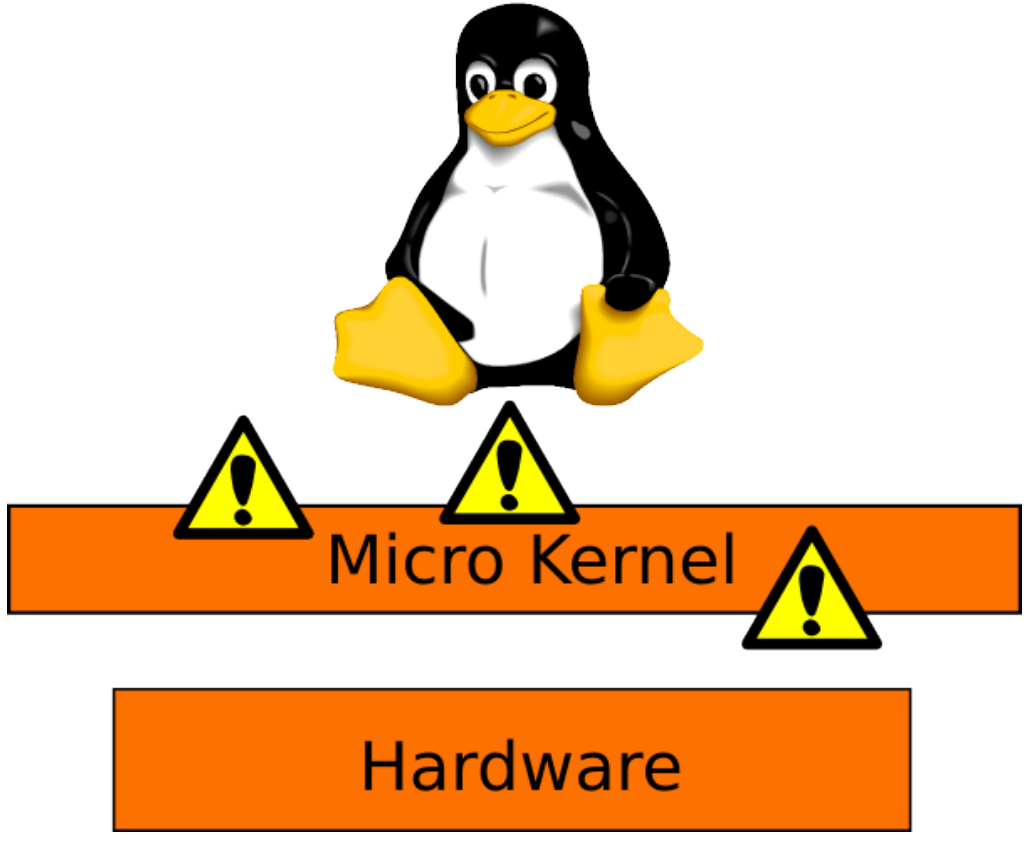
Priority Inversion



Approaches

- Dual-Kernel
- In-Kernel / Single Kernel

Dual-Kernel



LINUX PITER

Single-Kernel

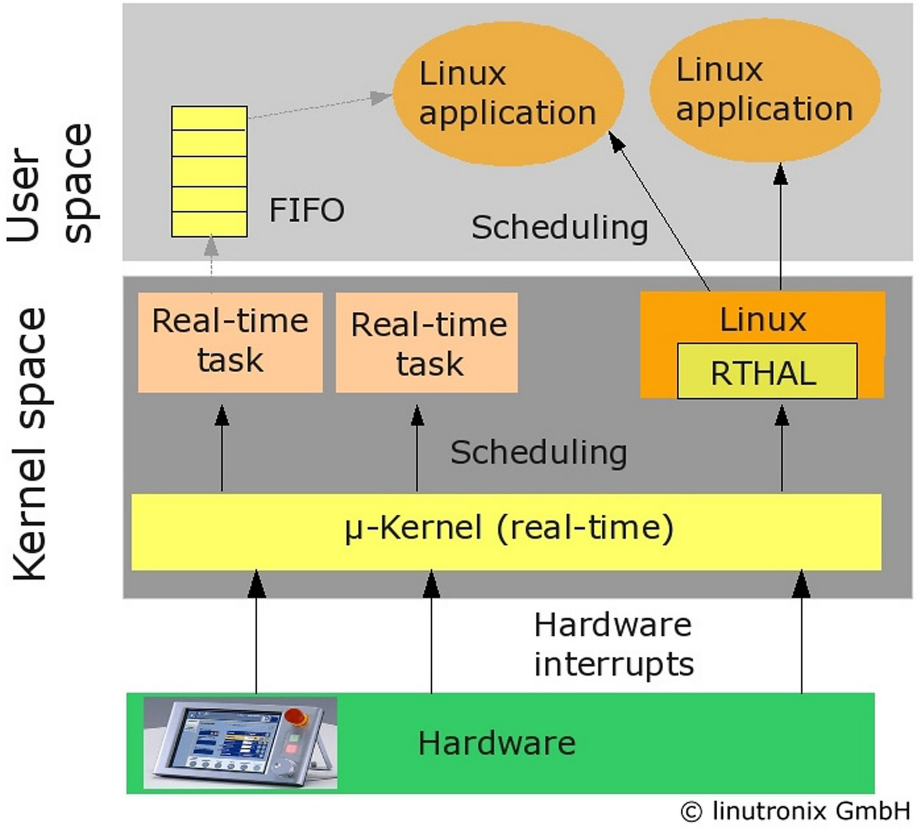
Hardware

LINUX PITER

RTAI

- Prof. Paolo Mantegazza, University of Milano
- Dual-Kernel approach
- Realtime in kernelspace
- Realtime in userspace very limited
- Design goal: Lowest latencies
- Supported platforms: x86, x86_64, and a couple of ARM platforms

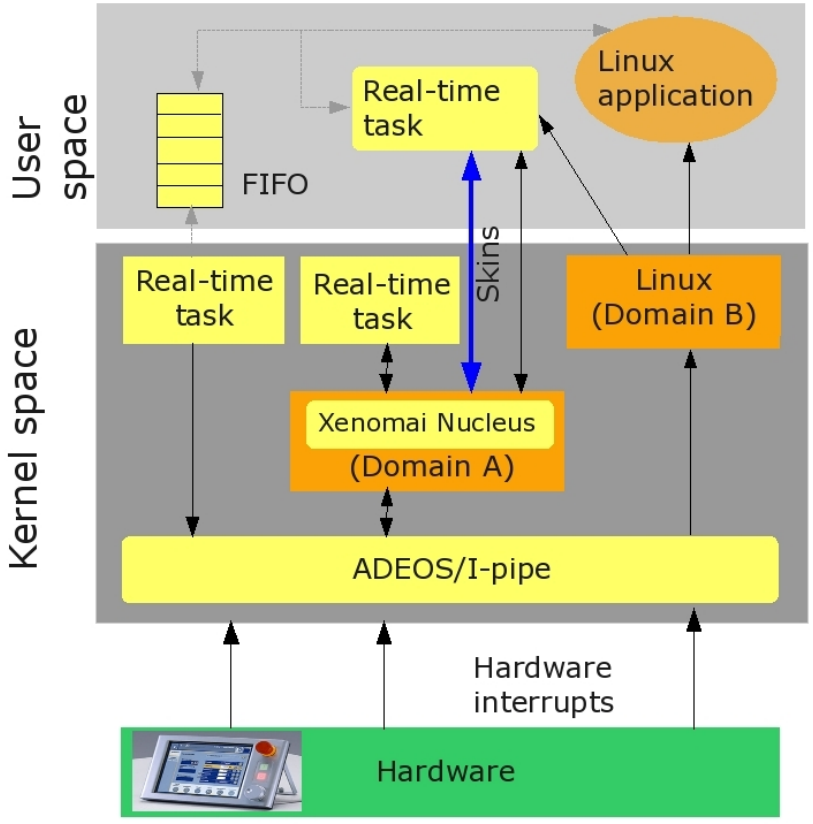
RTAI


User space
Kernel space
Linux application
Linux application
FIFO
Scheduling
Real-time task
Real-time task
Linux
RTHAL
Scheduling
µ-Kernel (real-time)
Hardware interrupts
Hardware
© linutronix GmbH

Xenomai

- Founded 2001
- Realtime in userspace
- Skins can emulate the API of different RTOSes
- Dual-Kernel approach
- Supported platforms: x86, x86_64, PowerPC, ARM, ia64

Xenomai


User space
Kernel space
FIFO
Real-time task
Linux application
Skins
Real-time task
Real-time task
Linux (Domain B)
Xenomai Nucleus
(Domain A)
ADEOS/I-pipe
Hardware interrupts
Hardware

Known issues of dual-kernel approaches

- Special API
- Special tools and libraries
- Microkernel needs to be ported for new HW and new Linux versions
- Bad scaling on big platforms (which is a problem for server people...Remember the financial service example)

Preempt RT

- In-Kernel approach
- Founded by: Thomas Gleixner, Ingo Molnar
- POSIX realtime
- A lot of the features already made it into "Mainline"
- Huge community
- Highly accepted in the community

How Preempt RT brings Realtime to Linux?

Remember once again...

Preemption is the most important requirement for a Realtime System

How Preempt RT brings Realtime to Linux?

- Locking Primitives: It introduces the "sleeping spinlocks"
- Interrupt Handlers run in a kernel thread
- Introduces "CONFIG_PREEMPT_RT_FULL"
- To make the story short: The main aim of the Preempt RT patch is to minimize the amount of kernel code that is non-preemptible

Sleeping spinlocks

- In Preempt RT spinlocks are mapped onto sleeping spinlocks, and raw spinlocks retain their behavior
- In a non Preempt RT preemption model spinlocks are mapped onto raw spinlocks

Threaded interrupt handlers

- Preempt RT forces threaded interrupt handlers
- To force an interrupt handler to be run in IRQ context it has to be marked with IRQF_NO_THREAD
- In mainline this behaviour can also be forced with the "threadirqs" commandline

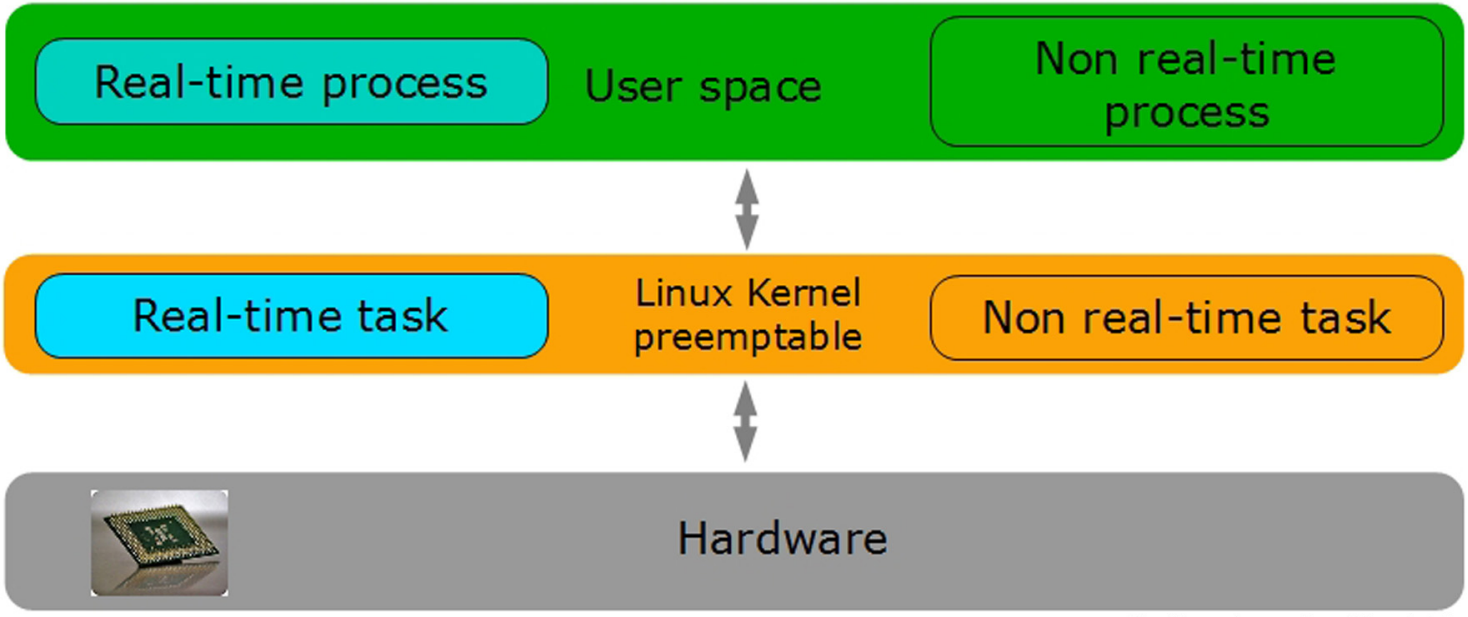
Threaded interrupt handlers

```
$ top
[...]
578 root    -51  0     0     0     0 S  0,0  0,0  0:00.00 irq/62-mei_me
[...]
```

So, how does userspace deal with it?

- Basically, userland won't even recognize ;-)
- Critical tasks use SCHED_FIFO or SCHED_RR
- Just follow the POSIX rules for realtime programming

Preempt RT



LINUX PITER

Preempt RT and Mainline

"Controlling a laser with Linux is crazy, but everyone in this room is crazy in his own way. So if you want to use Linux to control an industrial welding laser, I have no problem with your using Preempt RT" - Linus Torvalds auf dem Kernel Summit 2006

Preempt RT and Mainline

- Patchset provided for certain kernels
- Patchset at: http://kernel.org/pub/linux/kernel/projects/rt/
- No funding for a couple of years
- In October 2015 LF announced the RTL Collaborative Project at ELCE in Dublin

Who is doing Preempt RT

- RTL Collaborative Project
- Mainline development, new kernels, ... done by Thomas Gleixner and his team at Linutronix
- Testing:
  https://ci-rt.linutronix.de
- Steven Rostedt maintains most of the stable trees
- Julia Cartwright maintains the v4.1 tree

Features which are already mainline

Just a few well known examples...Mentioning all features would take a presentation on its own ;-)

- High Resolution Timers
- Threaded Interrupt Handlers
- Tracing Infrastructure
- ...

Recently accomplished task

- CPU hotplug rework
- CPU hotplug locking rework

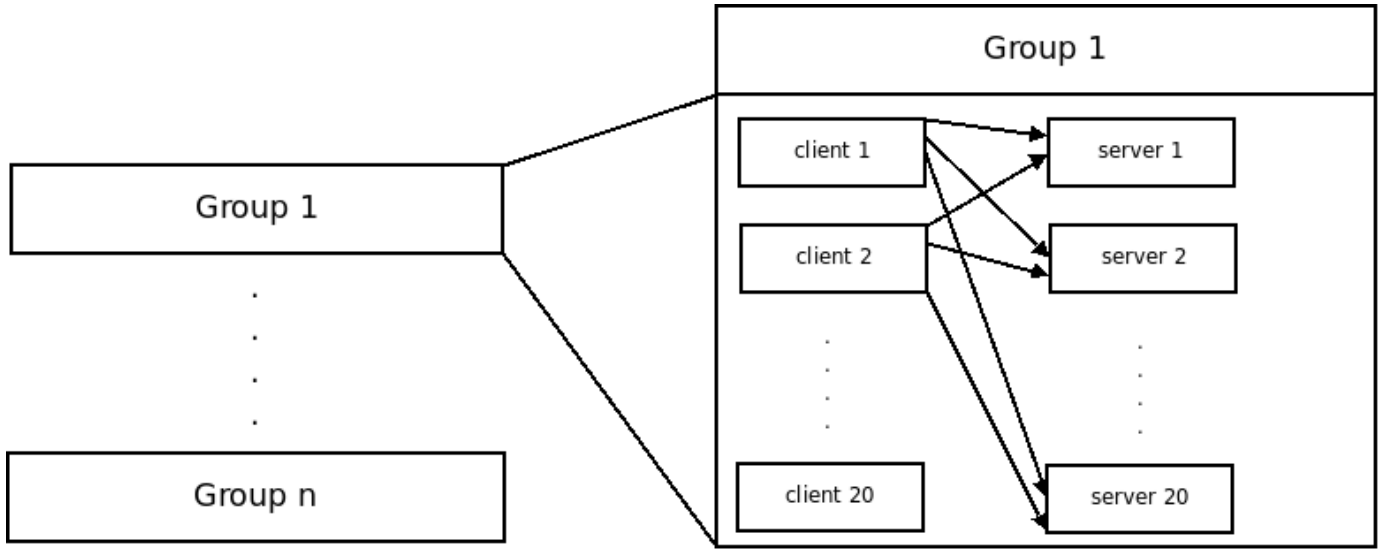
References

- More details about the mainlining status:
  "The Status of the Preempt-RT Patch" at ELCE 2017 by Sebastian Siewior
- Technical docs / Participation:
  https://wiki.linuxfoundation.org/realtime/start

Latency Measurements on a Cortex A9 platform

- ARM Cortex A9 SOC (Altera Cyclone V)
- System load: 100% CPU load with hackbench
- IRQ tests at 10 kHz with the OSADL Latency Box
- Test duration 12h

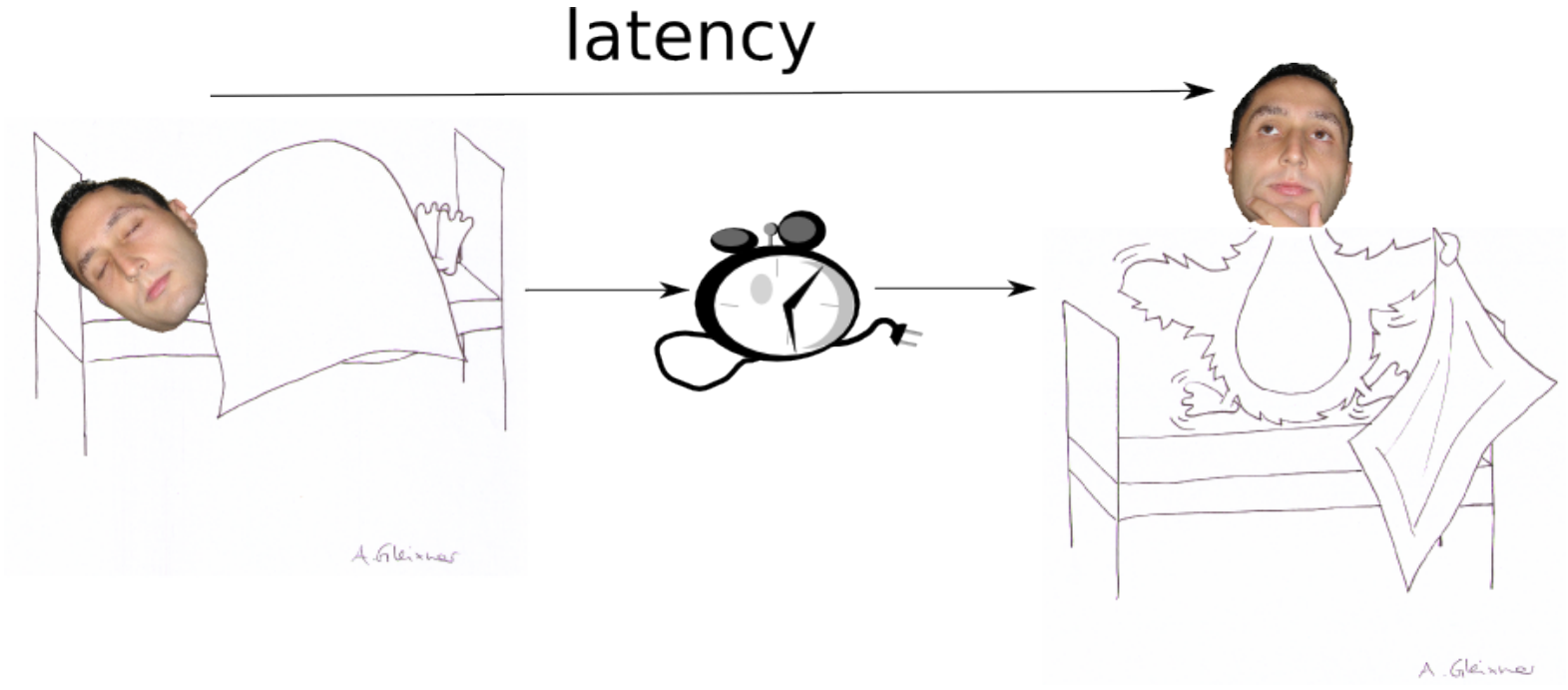
Load scenario: hackbench



- Starts n groups of 20 clients and 20 servers
- Each client sends 100 messages to each server via a socket connection

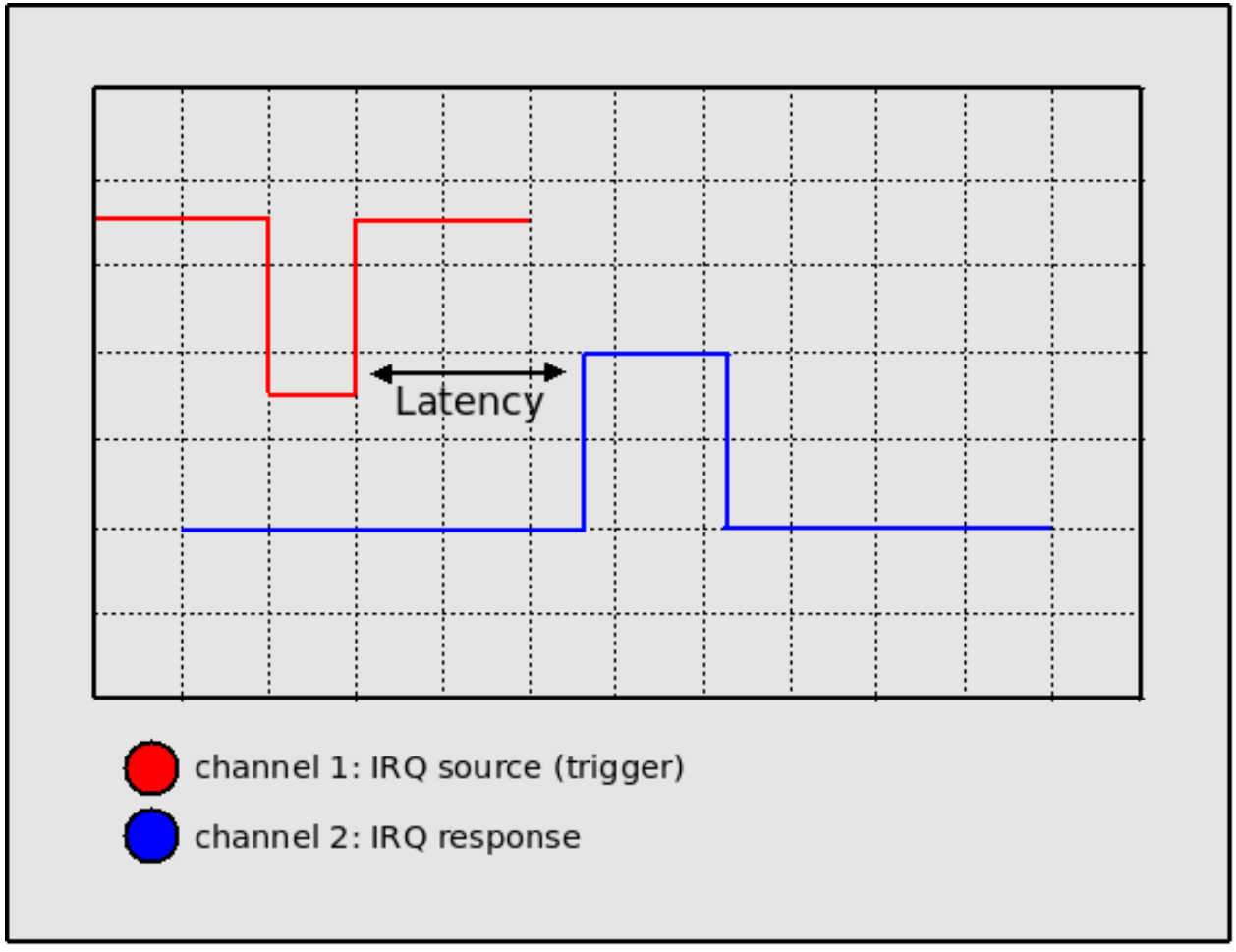
What has been measured?

Latency and Jitter



LINUX PITER

Latency measurement

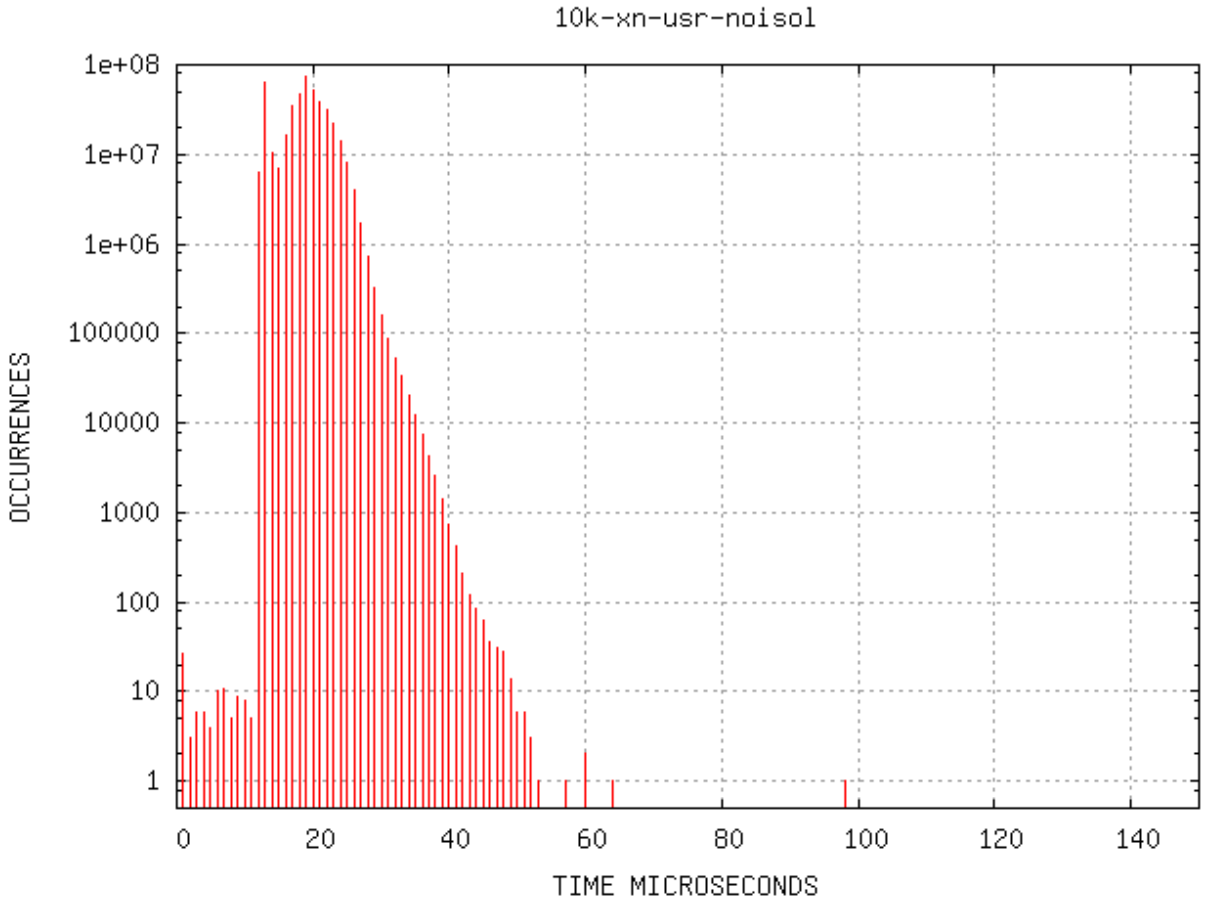


LINUX PITER

Userspace Latency: The most important usecase

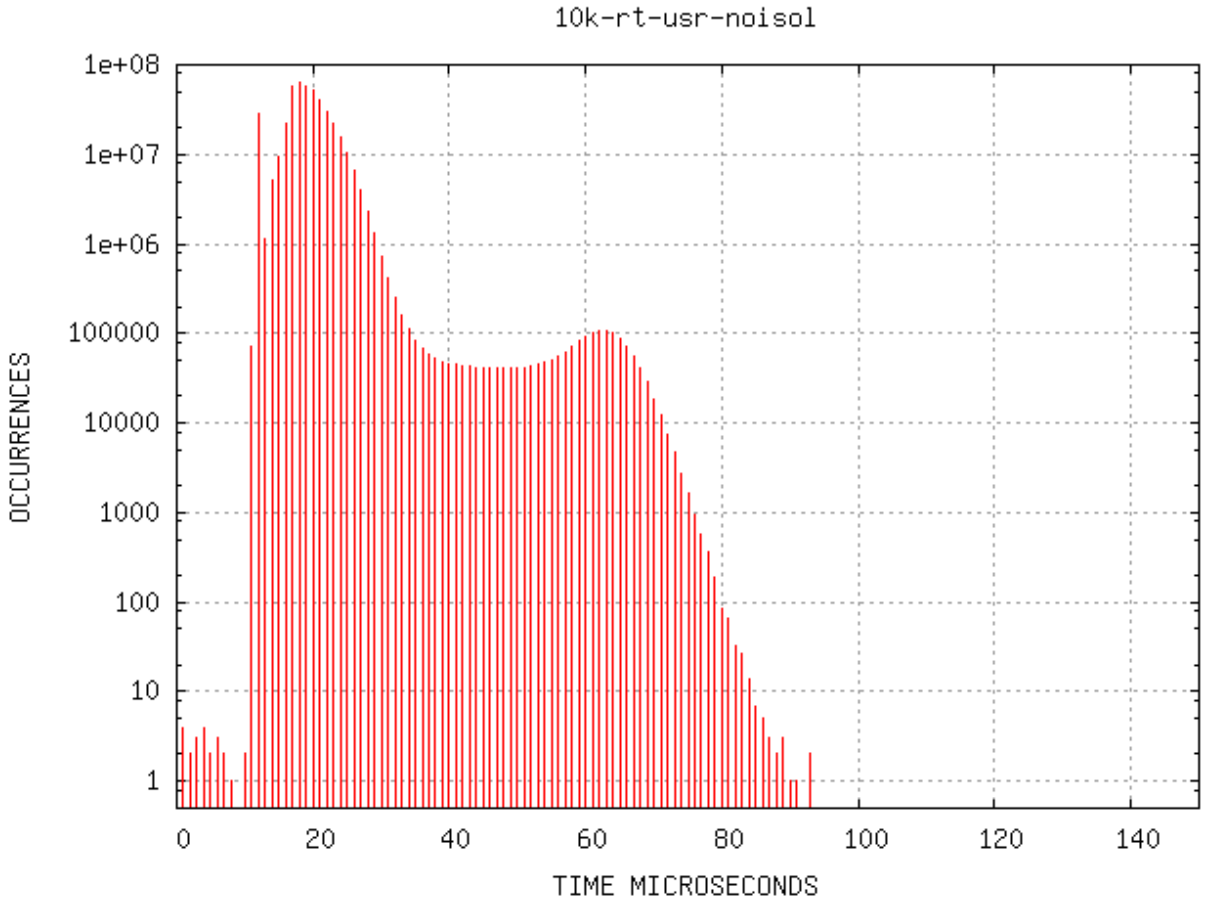
The most important usecase is the latency of a userspace task. Usually a userspace task needs to be synced with an external event.

Xenomai: latency userspace task



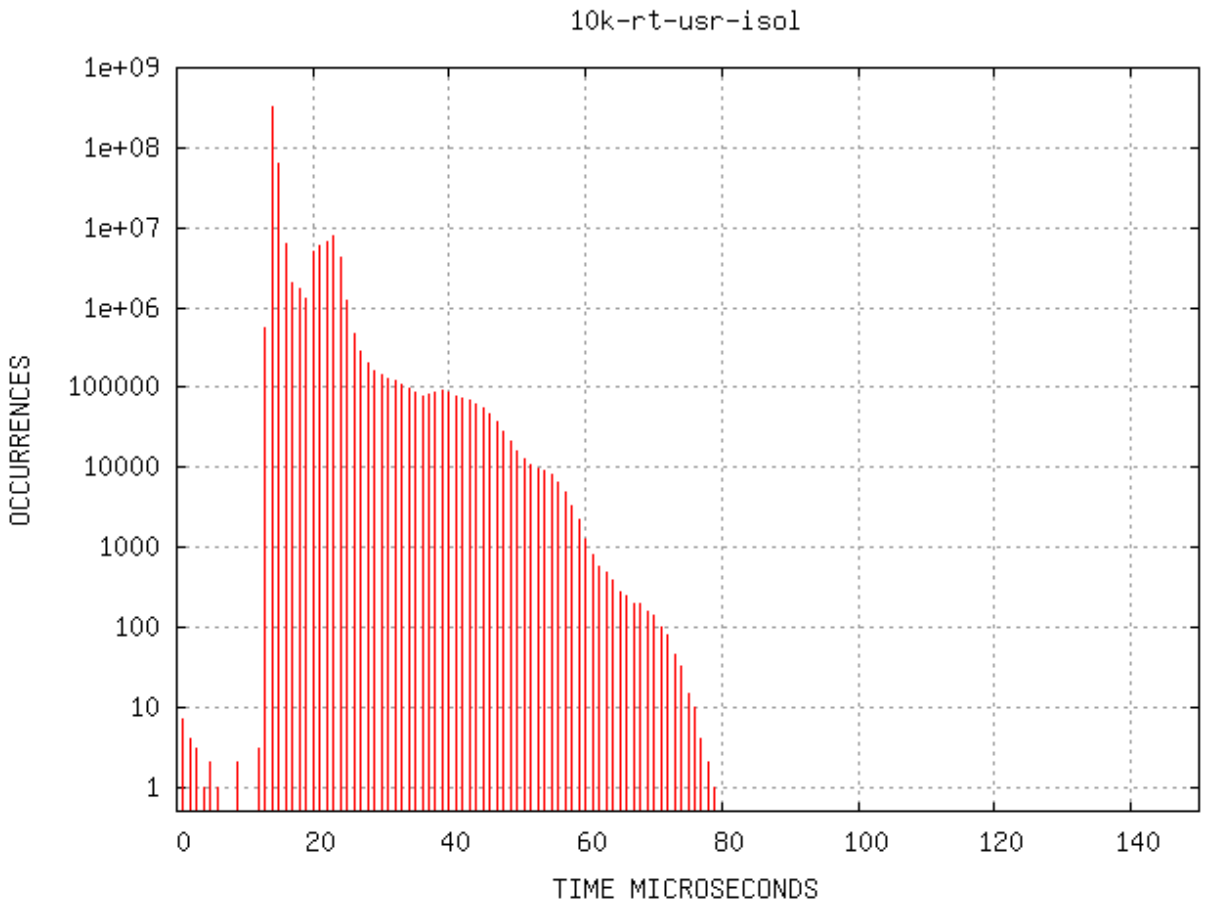
LINUX PITER

Preempt RT: latency userspace task



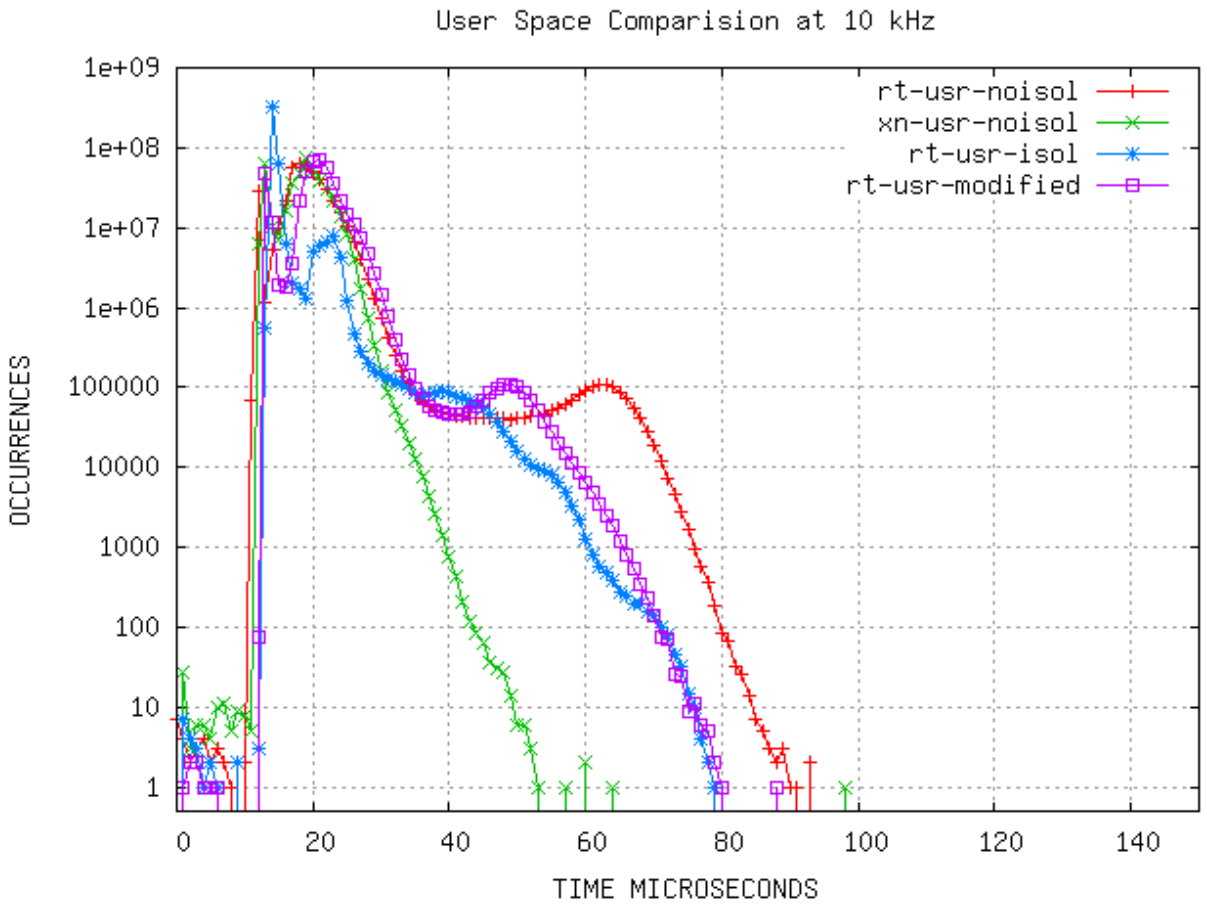
LINUX PITER

Preempt RT: latency userspace task (isolated CPU)



LINUX PITER

Latency userspace task - comparison



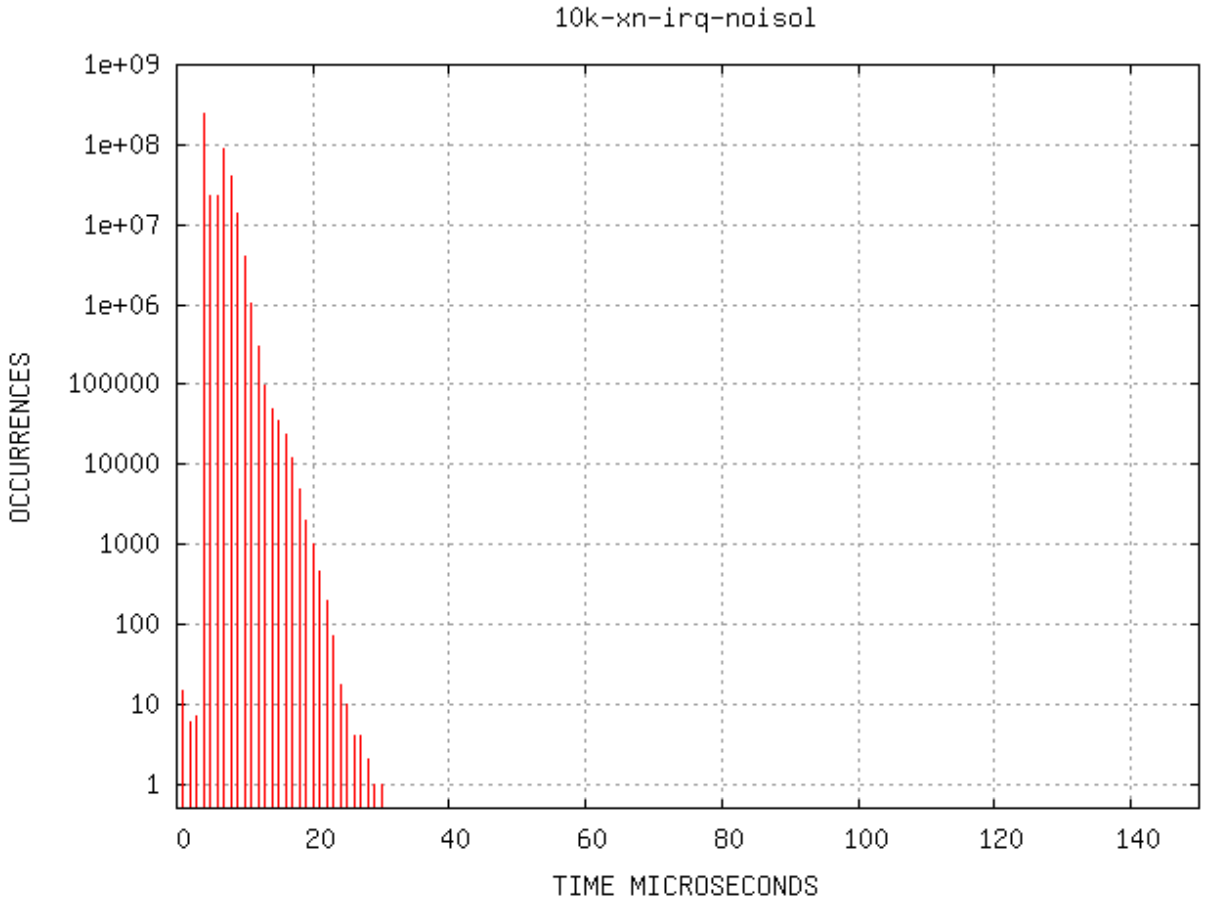
LINUX PITER

Latency within the Kernel

...or how to compare apples with pears!! ;-)

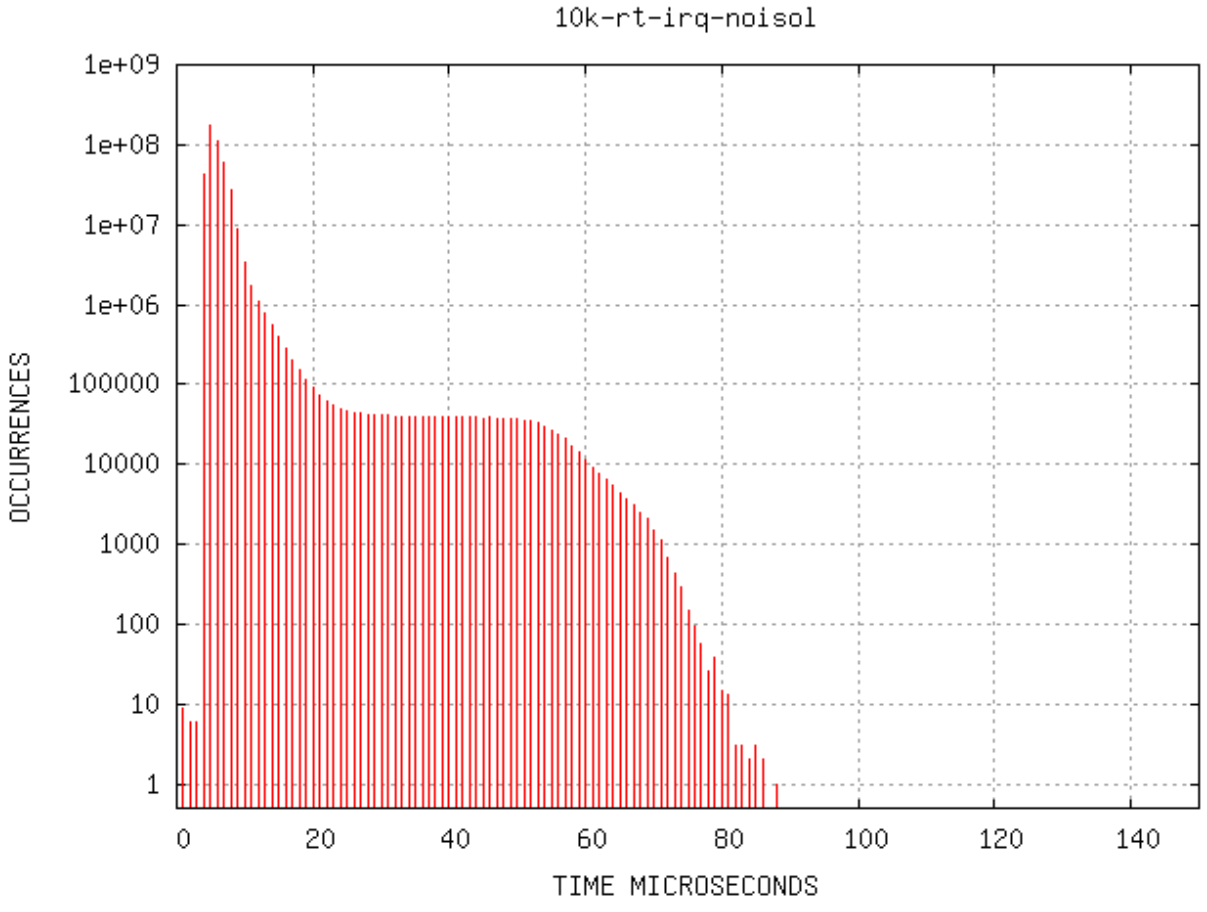
LINUX PITER

Latency: Kernel - Xenomai



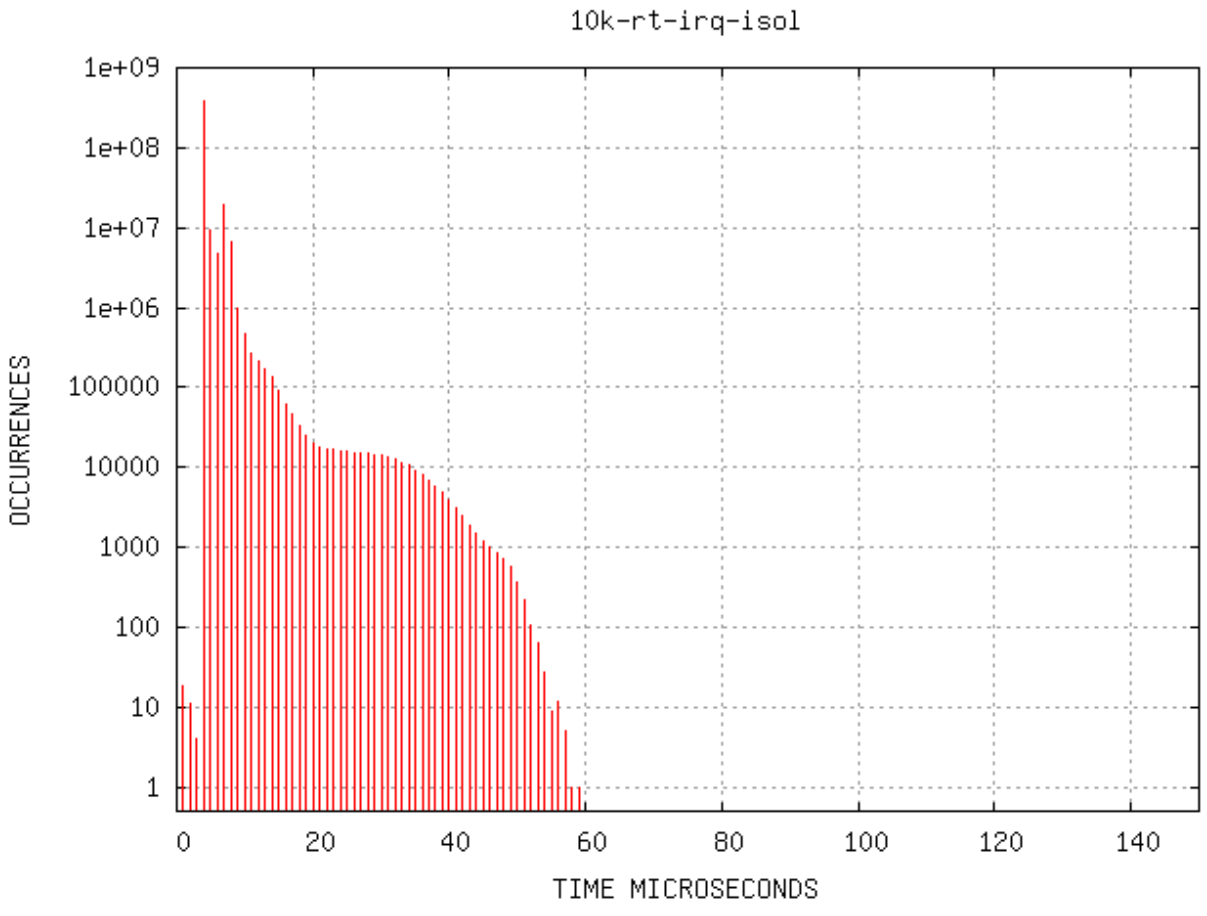
LINUX PITER

Latency: Kernel - Preempt RT



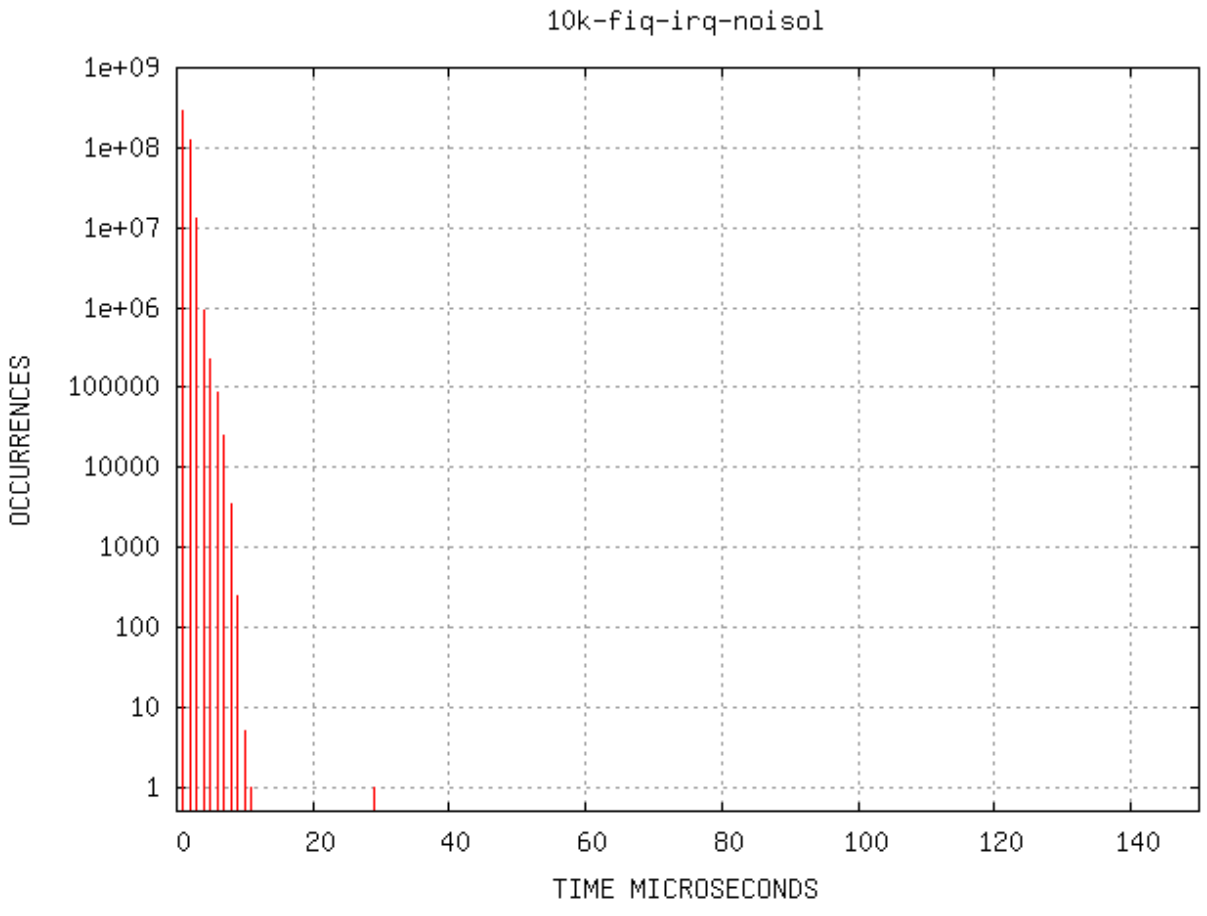
LINUX PITER

Latency: Kernel - Preempt RT (isolated CPU)



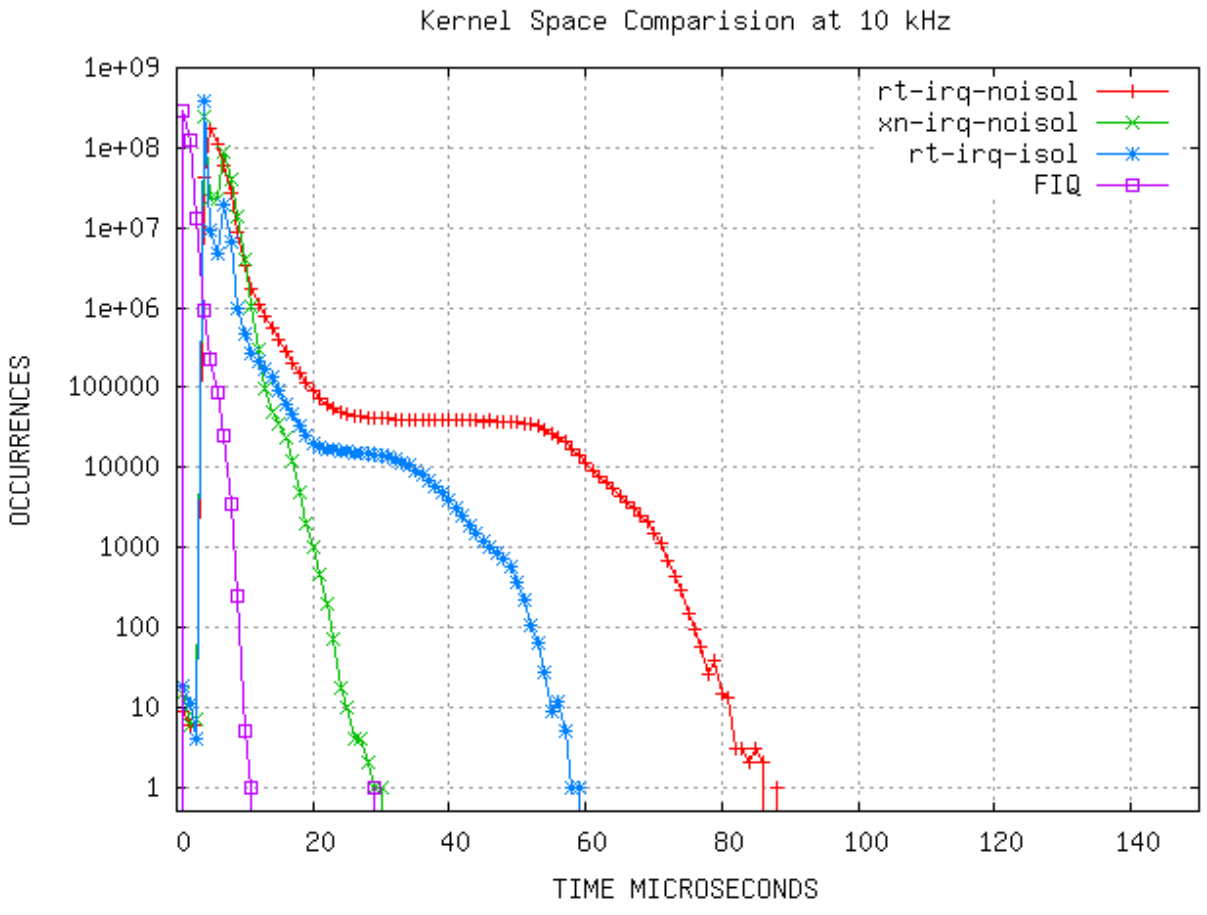
LINUX PITER

Latency: Kernel - Preempt RT with FIQ (fast interrupt)



LINUX PITER

Latency: Kernel - Comparison



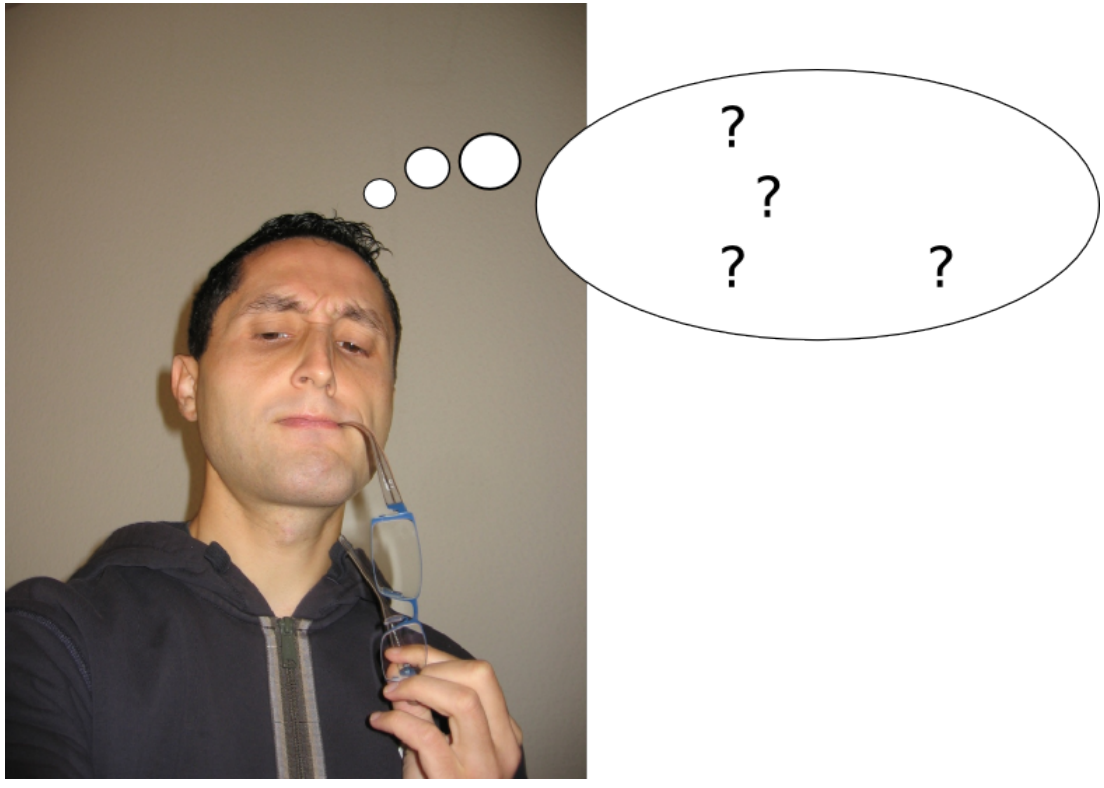
LINUX PITER

Conclusion

- Microkernels are hard to handle
- For the most common use-cases the Microkernels do NOT have better latencies
- Simple usage of Preempt RT
- Preempt RT became the de-facto standard for Realtime Linux
- Integration of Preempt RT in Mainline Linux
- Real Time Linux collaborative project

LINUX PITER

Questions?



LINUX PITER