

# PREEMPT\_RT

How a simple interface up event might  
break your realtime

**Florian Bezdeka**

Senior Embedded Linux & Real-Time Expert

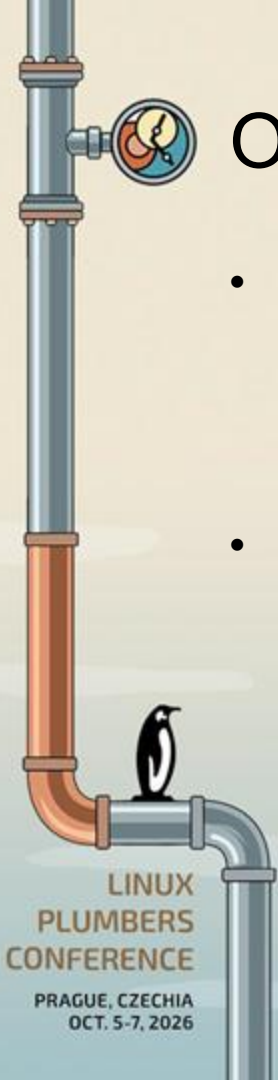
E-mail [florian.bezdeka@siemens.com](mailto:florian.bezdeka@siemens.com)



LINUX  
PLUMBERS  
CONFERENCE

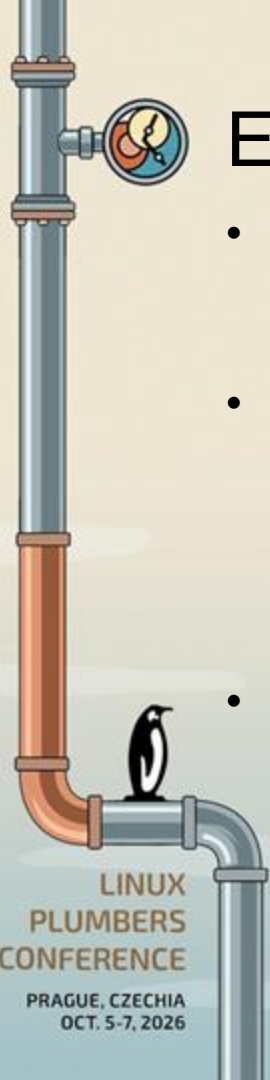
PRAGUE, CZECHIA / OCT. 5-7, 2026





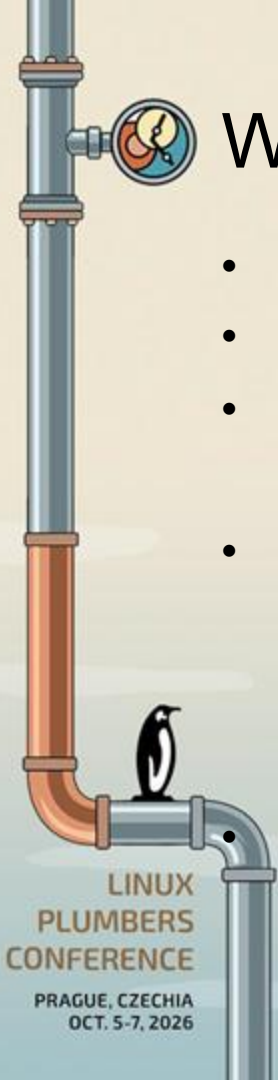
# Our use case: dynamic RT workload deployment

- More details about the use case at hand:  
See "Where PREEMPT\_RT meets the dynamics of container runtimes" talk given yesterday at the RT users forum
- Short summary
  - Container runtimes used for RT workload deployment
  - We don't know the concrete RT workload at boot time
  - Partial system re-configuration on application installation (CPU isolation, IRQ affinities, device passthrough, ...) needs to be configured at run time, while other workloads may continue to run at the same time



# Existing IRQ affinity APIs

- `/proc/irq/smp_default_affinity`  
User can write the affinity mask used for upcoming IRQs
- `/proc/irq/<irq>/smp_affinity[_list]`  
User can write the affinity of one specific IRQ.  
IRQ migration is triggered in case the current CPU is no longer part of the new affinity.
- Shortcomings:
  - Only visible after the first `request_irq()` call
  - "unused" devices are hidden after boot
  - Detection of new IRQs requires "scanning", which is racy
  - Some IRQ chips are marked with `IRQCHIP_MOVE_DEFERRED` one extra shot for IRQ migration required

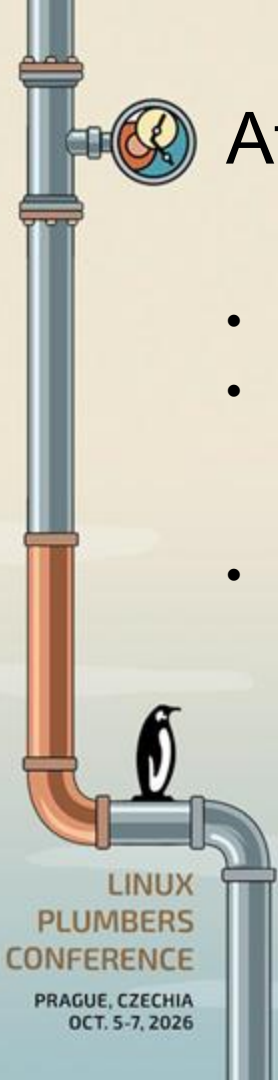


# What happened and where is the problem?

- Network interface was unused (=administratively down) on boot time
- An application decided to use that interface
- Responsible driver (stmmac) set IRQ affinity hints during device bring up
- Issues:
  - We can not set affinities for that device upfront (no `request_irq()` yet)
  - Driver overwrites user set affinities during each device bring up

## Device use cases:

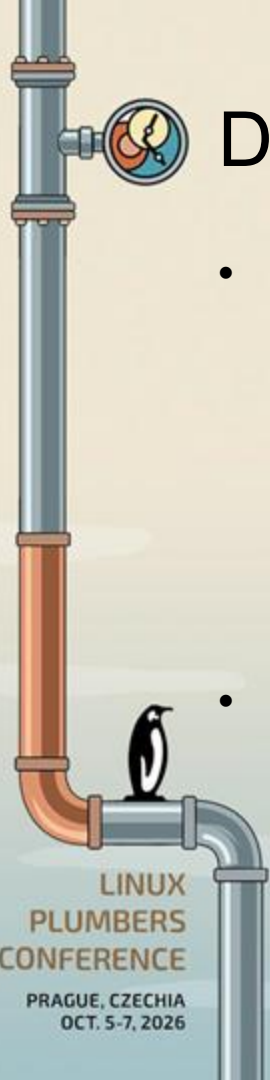
- Stay where you are, you are used by an RT application
- Stay away from RT cores, spreading (with constraints) still allowed



# Affected (net) drivers and why they use affinity hints

- Colocation of RX/TX interrupts per hardware queue
- IRQ spreading: Use all available CPUs for better throughput / link utilization
- Driver implementations are all based on `cpu_online_mask`, ignoring any CPU isolation configurations

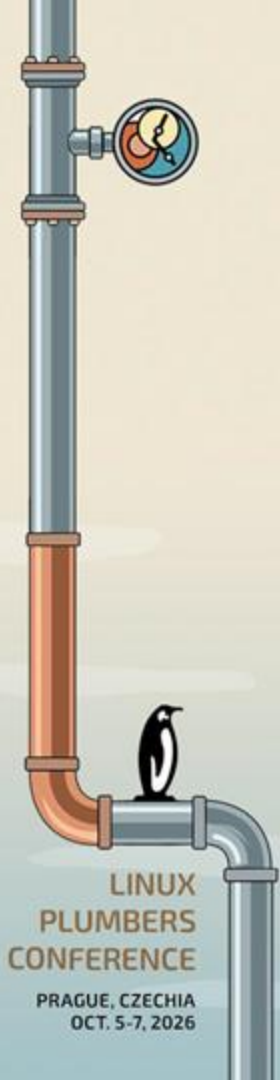
```
$ rg -l -e "irq_set_affinity(_and)?_hint" drivers/net | wc -l  
31
```

A vertical blue pipe with a pressure gauge and a penguin perched on a horizontal section.

# Discussion

- Should drivers really deal with affinity hints / cpumasks?
  - Let drivers express their intent / request special needs instead?
    - "co-locate this IRQ with..."
    - "spread out for performance"
  - Core sets affinities based on system rules
- Extend CPU isolation via cgroups?
  - New flag for the cpuset controller like "irq\_isolated"
  - Migrate away all device IRQs targeting the newly created cpuset?
  - Update the smp\_default\_affinity on "irq\_isolated" cpuset creation?

# Thank you!



LINUX  
PLUMBERS  
CONFERENCE

PRAGUE, CZECHIA  
OCT. 5-7, 2026