Low latency audio

...through core isolation

David Henningsson

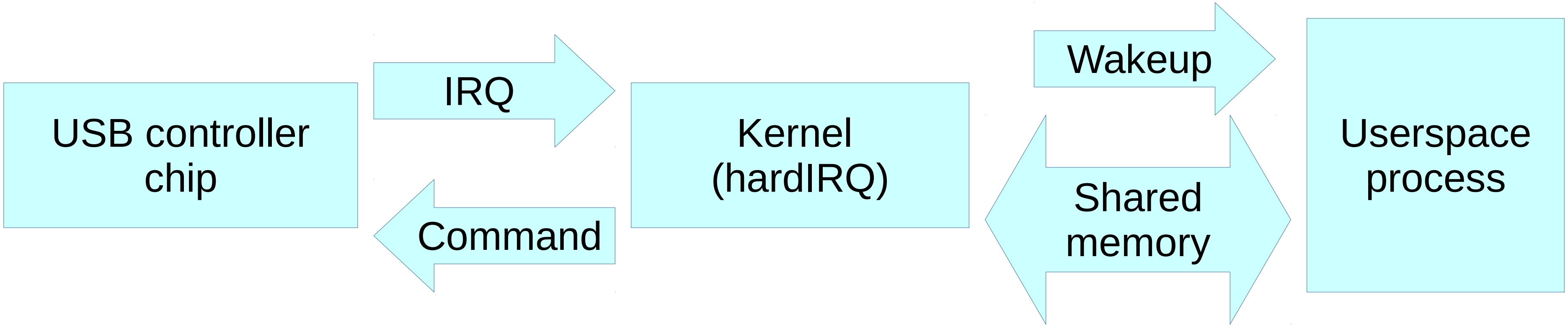
Their advertising said...

- This soundcard can do 2.7 ms round-trip latency on Mac and 3.2 ms on Windows 10
- Can we beat that on Linux?

Sources of latency

ADC hardware latency	unknown
Period size (16 samples @ 96 kHz)	0.16 ms
USB microframe	0.125 ms
Computer processing	Variable
Buffer size (32 samples @ 96 kHz)	0.33 ms
USB microframe	0.125 ms
Period size (16 samples @ 96 kHz)	0.16 ms
DAC hardware latency	unknown

USB audio buffer handling



USB controller chip

IRQ

Command

Kernel (hardIRQ)

Wakeup

Shared memory

Userspace process

The classic way

- Make the kernel schedule your process ASAP
- Use RT priority scheduling to do so
- But the CPU might be busy:
  - Processing an interrupt
  - Running a kernel thread that disables preemption (kernel cannot “schedule while atomic”)
- Other kernel drivers (graphics, wifi etc) don’t always consider other real time critical processes

The “isolated core” way

Dedicate one core for audio

- Run audio process on that core
- Audio IRQ processing on that core
- And **nothing** else.

- Kernel, please leave my core alone!

Basic tool: isolcpus

- Kernel parameter
- Keeps the kernel from scheduling tasks on specific cores
- Use taskset to move task to core
- Example:

  Boot with: `isolcpus=1`

  Run: `taskset -c 1 <cmd>`

Basic tool: irqaffinity

- Kernel parameter

  (added in 4.6, but mostly a convenience)

- Schedules irqs only on specific cores
- Use /proc/irq/<x>/smp_affinity_list to move an irq to a specific core
- Example:

  Boot with: `irqaffinity=0,2,3`

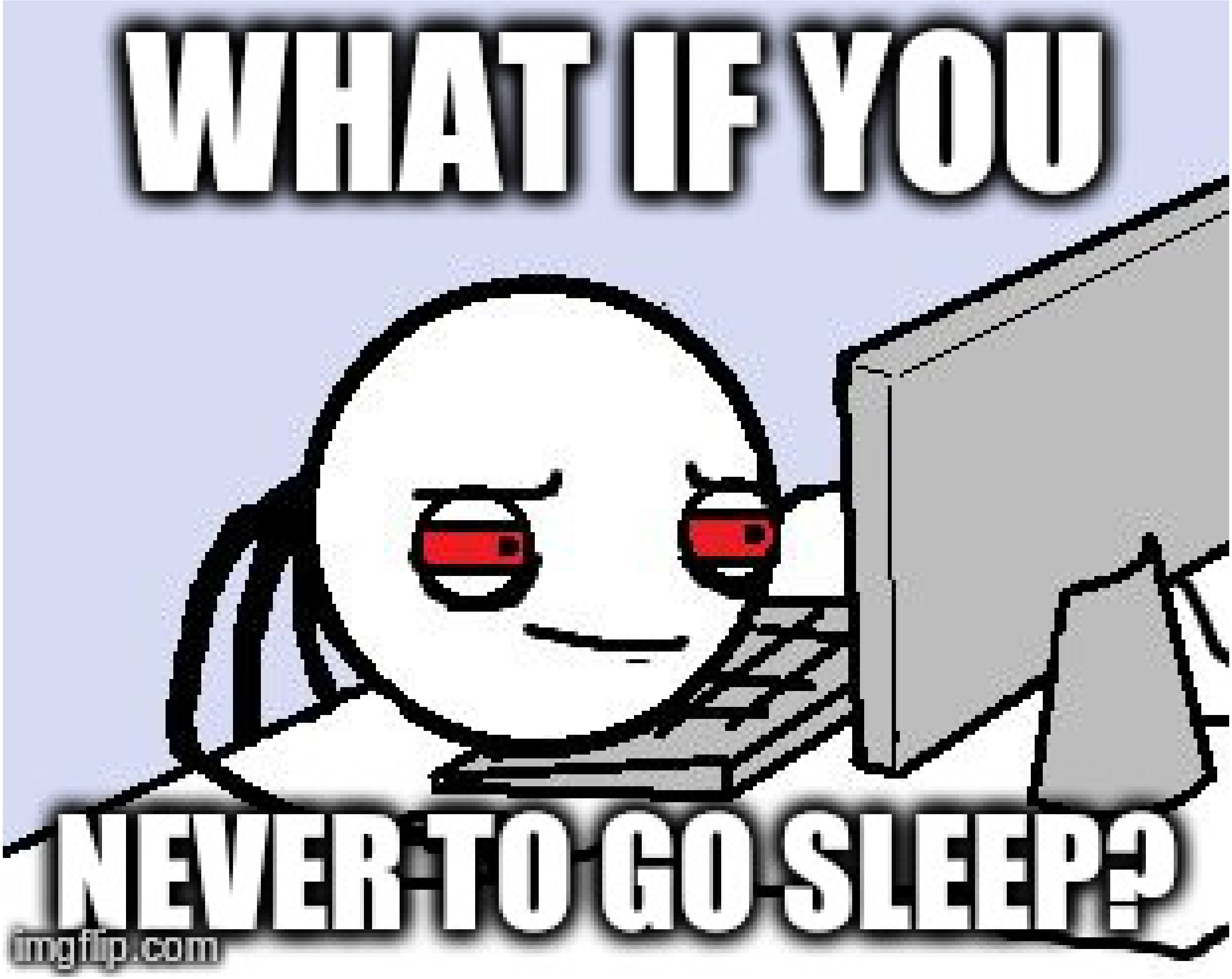
  Run: `echo 1 > /proc/irq/<x>/smp_affinity_list`

Basic tool: nohz_full

- Timer interrupts are run on all cores by default (regardless of `irqaffinity` and `isolcpus`)
- But we can turn them off, if there is only one process to be scheduled on a core.
- Example:
  - Boot with `nohz_full=1` to turn off timer interrupts on core 1

Sleeping

- If your thread goes to sleep, the kernel still needs to wake you up (i e schedule your process)
- But the kernel might be in “preempt disable” mode


WHAT IF YOU
NEVER TO GO SLEEP?
imgflip.com

Basic tool: scaling_governor

- Controls the frequency of cores
- By setting it to `performance` for our particular core, we avoid two problems:
  - The core pauses during frequency changes (hardware limitation)
  - No kernel thread run periodically to check load
- Example:

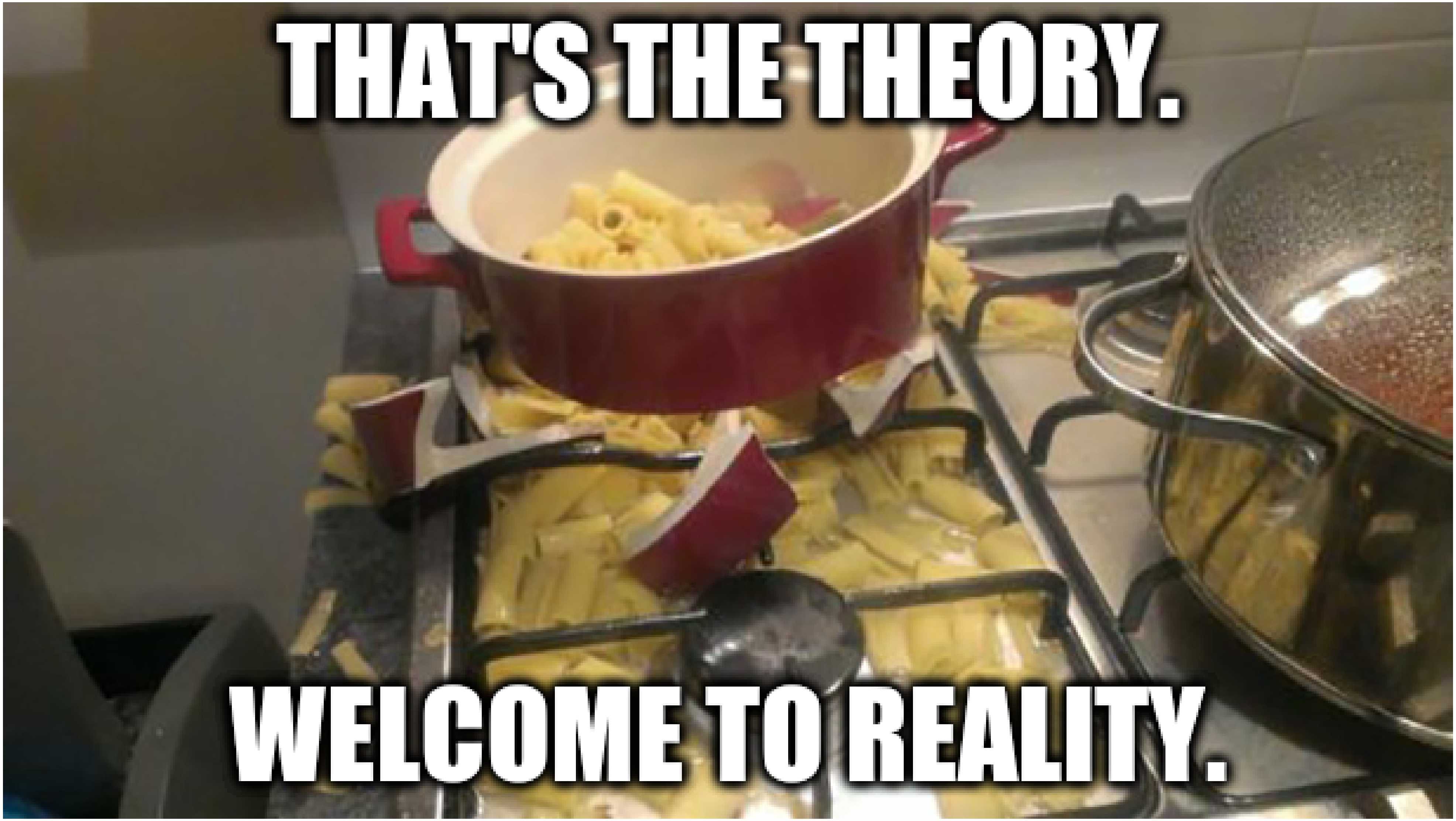
  -
    ```
    echo performance > /sys/devices/system/cpu/cpu1/cpufreq/scaling_governor
    ```

The setup:

- We boot with `isolcpus=2,3 irqaffinity=0,1 nohz_full=2,3`
- Move the right IRQ to our isolated core:
  - `echo “2,3” > /proc/irq/20/smp_affinity_list`
- Start the task on our isolated core: `taskset -c 2,3 ./lowlatencytask`

Isolated core strategy summary

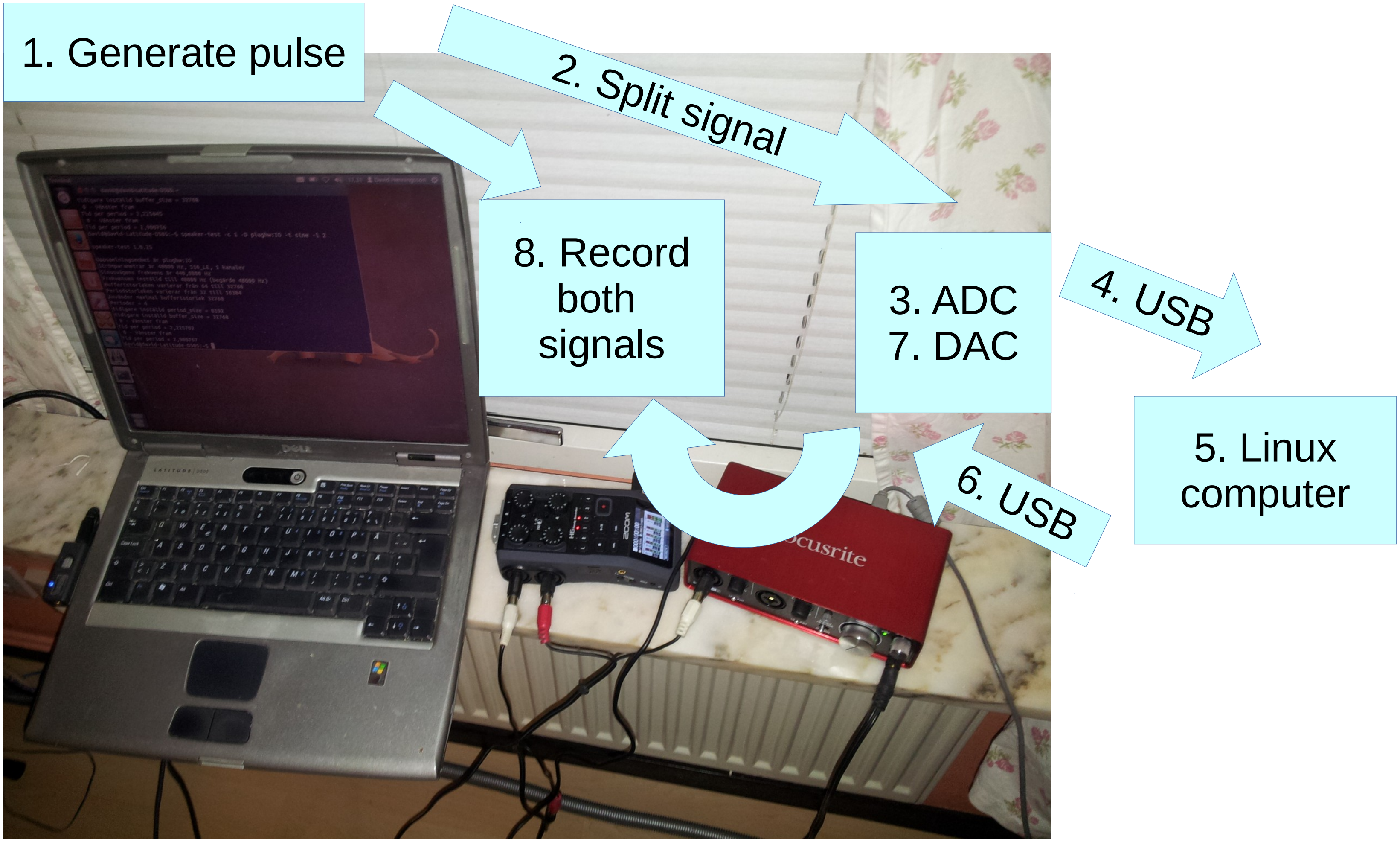
- **RT prio**? Doesn’t matter – no other process should run on my core
- **IRQ RT prio**? Doesn’t matter – no other IRQ should run on my core
- Kernel is **Preempt disabled**? Doesn’t matter – my process doesn’t sleep, so it doesn’t need the kernel to wake it up
- **Threaded IRQ** handlers? Probably makes things worse if the kernel has to schedule my IRQ.


THAT'S THE THEORY.
WELCOME TO REALITY.

Test subjects

- AMD Ryzen 1700 desktop
- A Dell laptop from 2016, i3-6100U (Skylake)
- An HP laptop from 2010, core i3 M350 (Arrandale)
- Raspberry Pi 2

Testing equipment



1. Generate pulse

2. Split signal

3. ADC
7. DAC

4. USB

5. Linux computer

6. USB

8. Record both signals

Raspberry Pi 2 (or 3)

- Quad-core, but…
- All interrupts arrive at core 0. This is a hardware limitation.
- Note: many other ARM boards do not suffer from this.

Skylake (i3-6100U) laptop

- This one has one USB interrupt only
- If you have a laptop, everything else is on that USB bus too:
  - Bluetooth
  - Touchscreen
  - Webcam
  - Sensor chip (accelerometer / gyro / etc)
- Didn’t boot with irqaffinity=0,1

Mic splitter

- Passive microphone splitter – added up to 0.5 ms of latency (for tapping mic).

Core / thread lesson learned

- If you’re on a system with hyperthreading, dedicate an entire physical core to audio
- Look for `core id` in `/proc/cpuinfo`
- This made a lot of difference on the Ryzen system

Hardware latency varies

For this card (Focusrite 2i2 2nd gen), hardware latency is lower at 96 kHz than 48 kHz

So roundtrip latency for 16 samples @ 48 kHz is larger than 32 samples @ 96 kHz, despite both buffers being 0.33 ms

Test results

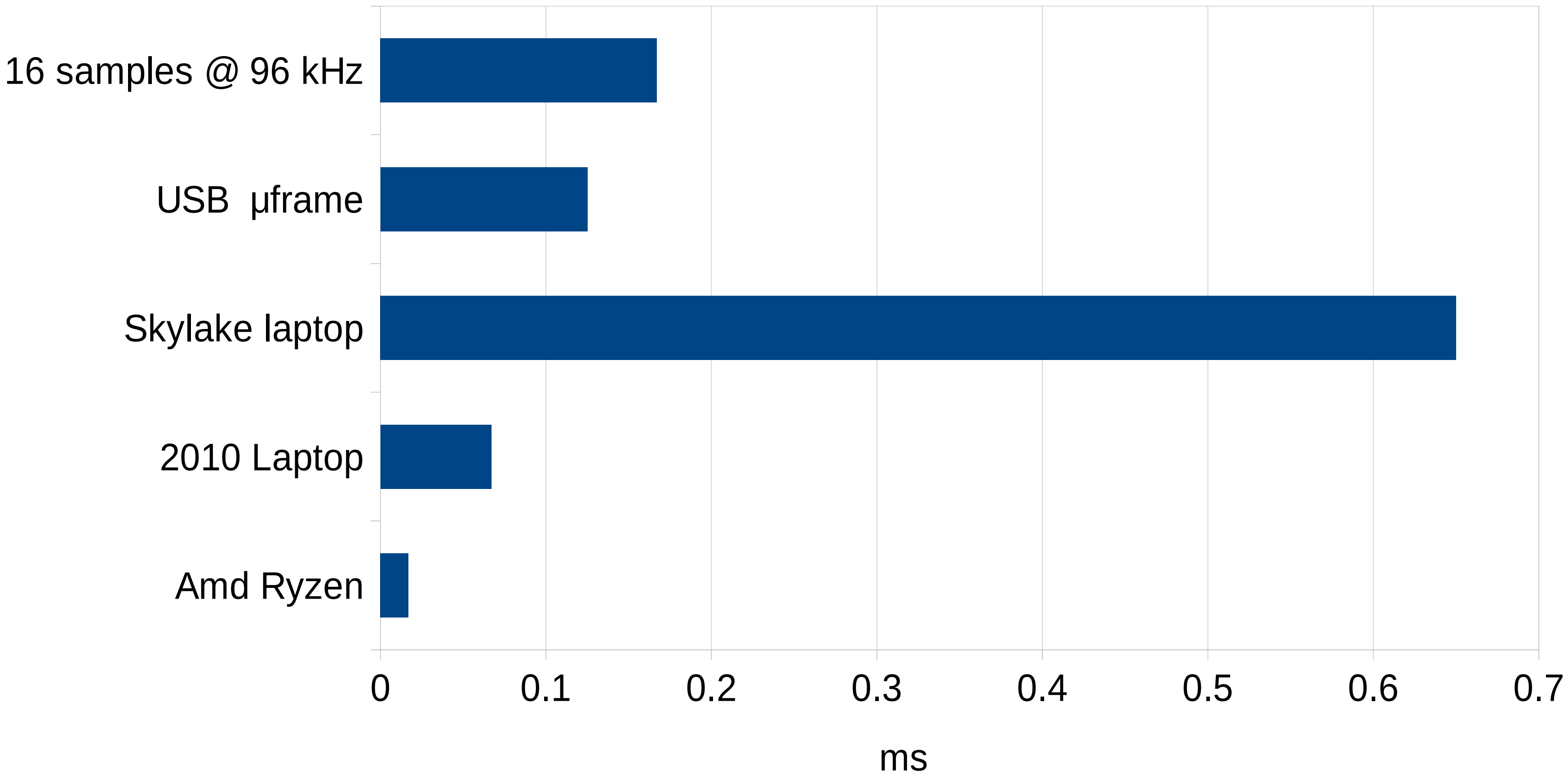
Two systems tested successfully:

- AMD Ryzen 1700 desktop
- An HP laptop from 2010, core i3 M350 (Arrandale)

First test: timedloop

- Rdtsc is a computer instruction which reads the cpu’s internal clock tick count.
- timedloop just reads this value in a loop, and checks for the maximum difference (compared to the previous read).

Timedloop test results



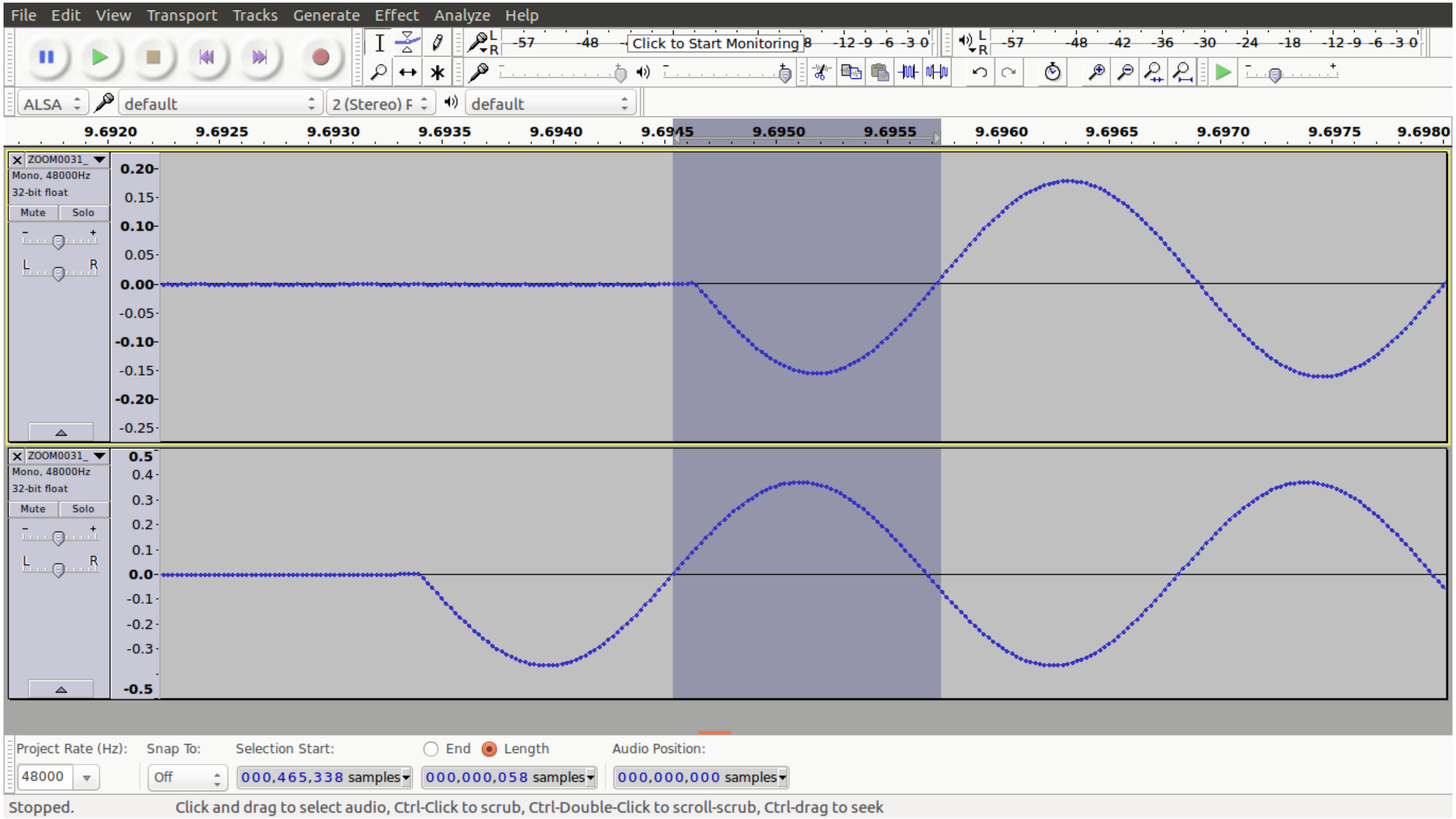
Main test: neversleep-alsa

- Starts capture and playback
- Busy-loops checking for incoming data in capture buffer
- Any data in capture buffer is copied to playback buffer ASAP

Test result...

- AMD Ryzen 1700 system:
  - 32 (2 x 16) samples @ 96 kHz (0.33 ms buffer size)
- 2010 (i3-350M) laptop:
  - 72 (3 x 24) samples @ 96 kHz (0.75 ms buffer size)

1,2 ms roundtrip latency!



Project Rate (Hz): 48000

Snap To: Off

Selection Start: 000,465,338 samples

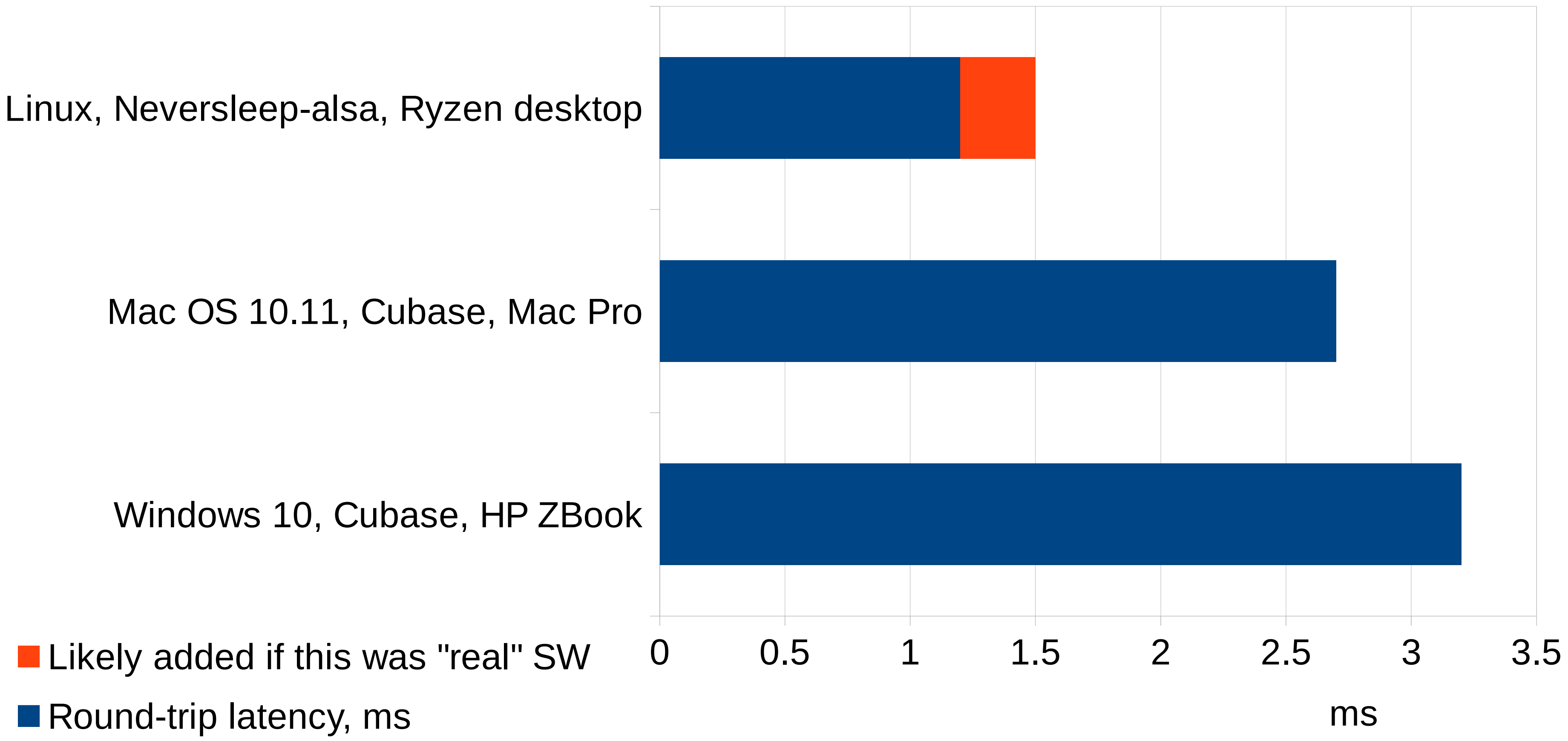
○ End ● Length 000,000,058 samples

Audio Position: 000,000,000 samples

Stopped.

Click and drag to select audio, Ctrl-Click to scrub, Ctrl-Double-Click to scroll-scrub, Ctrl-drag to seek

32 samples @ 96 kHz



Conclusions

- ALSA drivers are really good
- For USB, a “prioritize my audio” feature could be useful
- Core isolation (as a kernel feature) could be improved (`isolcpus` is not a silver bullet)
- Still a matter of knowing your hardware

Interesting future work

- Implement core isolation in JACK to see how well it works, both with sleeping and with busy-waiting
- Investigate (e g with ftrace) if wakeup time can be trimmed further, i e what still runs on the core and why

AMD Ryzen system information

- Ryzen 1700, 8 cores, 16 threads
- MSI Tomahawk B350, Bios 1.3
- Soundcard connected to the “Ryzen” USB controller, other USB devices connected to the “B350” USB controller
- Overclocked to 3.5 GHz, 16 GB DDR4 @ 2667 MHz
- Kernel: 4.10.1-041001-generic, isolcpus=6,7 nohz_full=6,7
- Cmd: taskset -c 6,7 timedloop (Result: **17 us** max latency)
- Cmd: taskset -c 6,7 neversleep-alsa hw:USB **32** 16 96000 (Result: **1,2 ms** roundtrip latency)

Laptop from 2010, system info

- Core i3 M350, 2,3 GHz (2 cores, 4 threads)
- HP Probook 4520s
- Kernel: 4.8.0-46-generic, isolcpus=2,3 irqaffinity=0,1 nohz_full=2,3
- “performance” set as scaling_governor for cpus 2 and 3
- Cmd: taskset -c 2,3 timedloop (Result: max **67 us** latency)
- Put soundcard in outermost USB jack on right side (otherwise it ends up on the same IRQ as other devices)
- /proc/irq/20/smp_affinity_list set to 2
- Cmd: taskset -c 2,3 neversleep-alsa hw:USB **72 24** 96000 (Result: **2,3 ms** roundtrip latency)

Source code at...

https://github.com/diwic/core-isolation

Questions?