Crafting a Linux kernel scheduler that runs in user-space using Rust

Andrea Righi

Linux Plumbers Conference 2024 | Vienna

Scheduling

What is a scheduler?

- Kernel component that determine:
  - **where** each task needs to run
  - **when** each task needs to run
  - **how long** each task needs to run

Why does scheduling matter?

- Performance
  - Specific workload
  - Specific hardware topology
- Security
  - Isolation
- Energy Efficiency
  - EAS

Scheduling in Linux

- One scheduler to rule them all
  - CFS < v6.6
  - EEVDF >= v6.6
- Really difficult to conduct experiments
- Really difficult to upstream changes
- Development not easily accessible

sched_ext

- Allows to implement schedulers as BPF programs
- Dynamically load/unload them at run-time
- BPF guarantees safety (no kernel panics, memory bugs, etc.)
- sched_ext watchdog prevents deadlock / starvation

sched_ext: pros / cons

- Pros
  - Ease of experimentation
  - Fast edit/compile/test iteration
  - Safety (bugs have almost zero impact)
- Cons
  - Limited programming model
  - BPF verifier complexity
  - Kernel restrictions (no user-space libs, no floating point, etc.)

Scheduling in user space

User-space scheduling

- eBPF + sched_ext
  - Channel scheduling events to user-space
- A scheduler becomes a regular user-space program
- See also
  - ghOSt framework (Google)

User-space scheduling: pros / cons

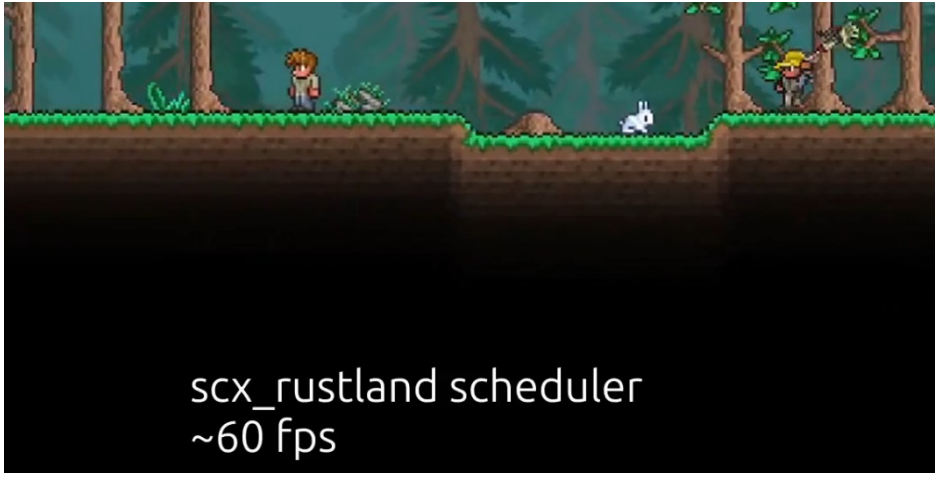
- Pros
  - Unlock all user-space languages
  - Access to a large pool of user-space libraries
  - Access to user-space services
  - Observability
- Cons
  - Overhead
  - Fragmentation / user support

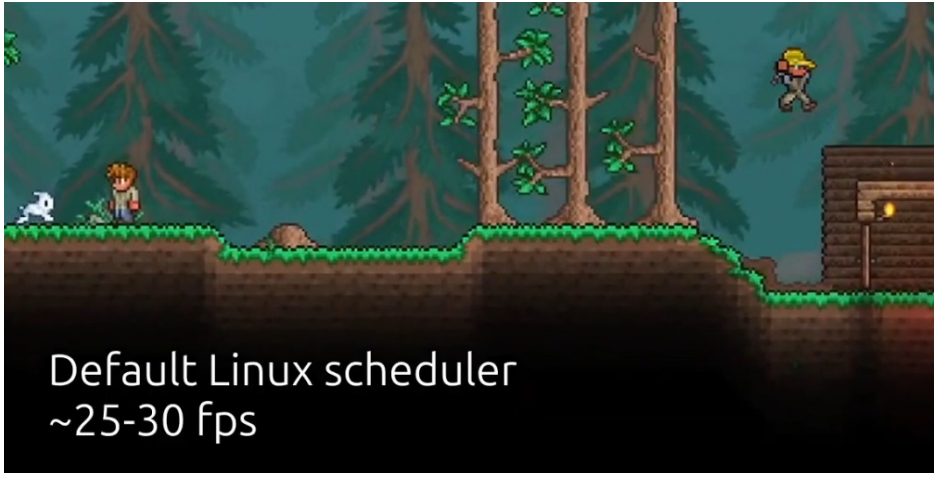
User-space scheduler in Rust

scx_rustland

- A fully functional Linux scheduler based on sched_ext
- Written in Rust (nothing to do with Rust-for-Linux)
- The scheduler runs as regular user-space process
  - 100% of the scheduling decisions done in user space
- vruntime scheduler that prioritizes interactive tasks

Playing Terraria while building the kernel





Goal of scx_rustland

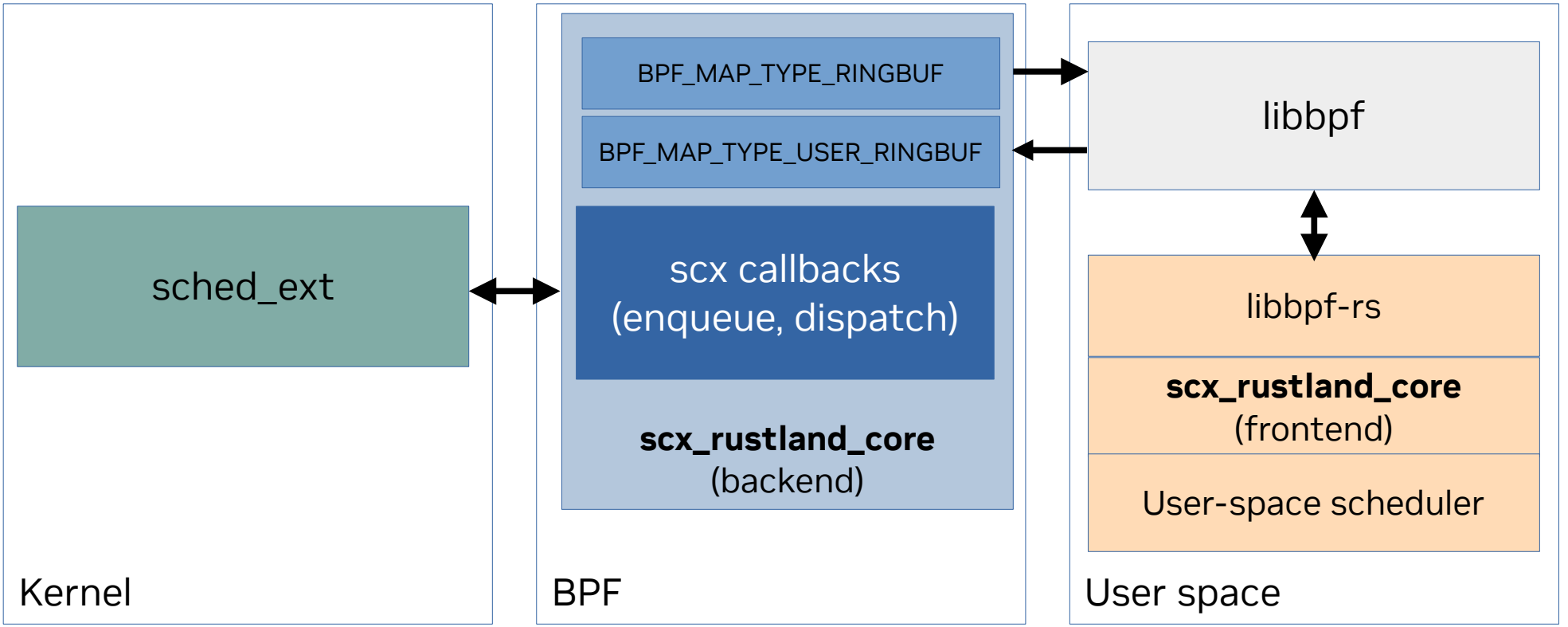
- scx_rustland is a **proof of concept** to show that user-space scheduling is a viable option
- Despite the overhead we can achieve performance close to in-kernel scheduling
- Make **scheduling development** more **accessible**

User-space scheduling framework

scx_rustland_core

- Abstraction layer over sched_ext
- Interface between BPF/sched_ext and user space
- Kernel scheduler is a user-space process
- Can be used in any Rust project
- GPLv2 license

scx_rustland_core architecture



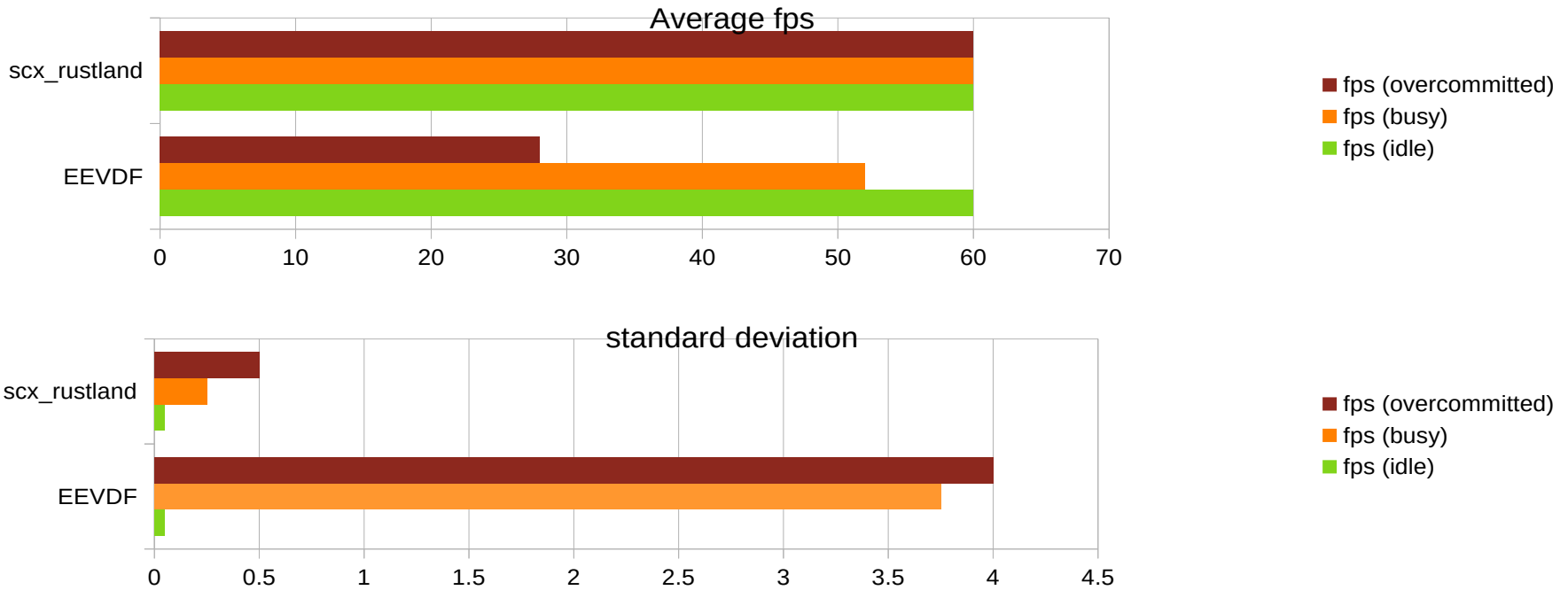
NVIDIA

scx_rustland_core API

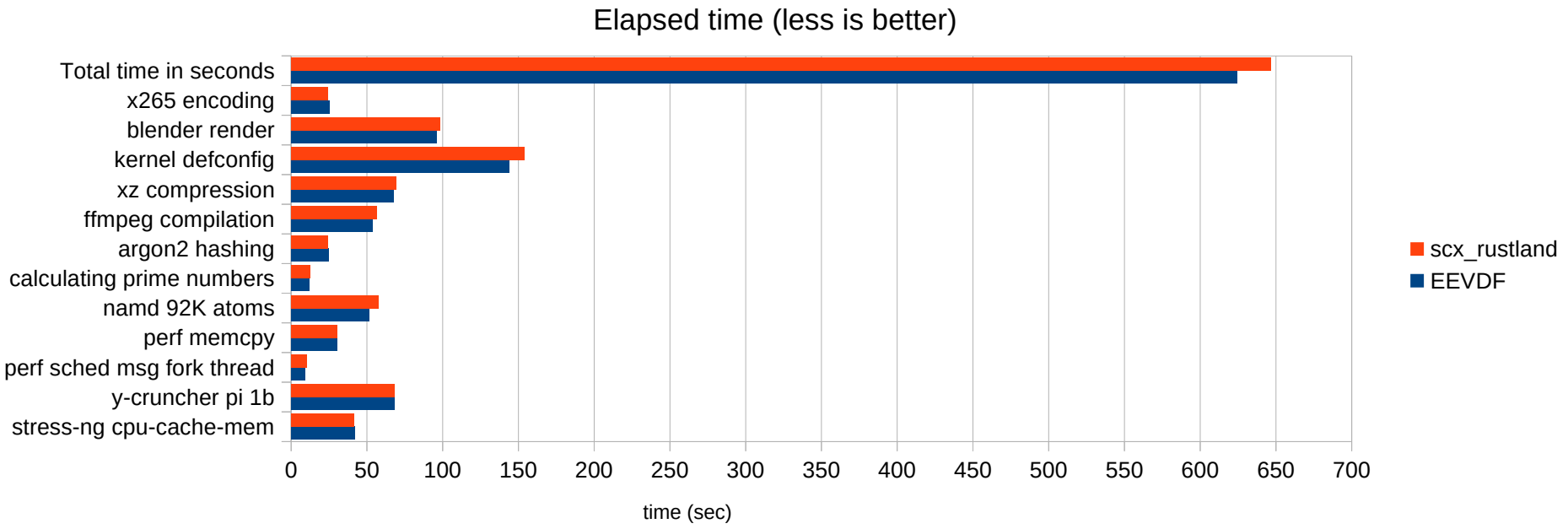
- struct BpfScheduler
  - **Task management**
    - dequeue_task(&mut self) -> Result<Option<QueuedTask>, i32>
      - consume a task that wants to run
    - select_cpu(&mut self, pid: i32, cpu: i32, flags: u64) -> i32
      - find an idle CPU for the task
    - dispatch_task(&mut self, task: &DispatchedTask) -> Result<(), Error>
      - dispatch a task
  - **Completion notification**
    - notify_complete(&mut self, nr_pending: u64)
      - notify BPF component that some tasks have been dispatched

Results

Gaming while building the kernel



User-space overhead



Conclusion

Benefits of user-space scheduling

- Lower the barrier of scheduling development
- Faster edit/compile/test iterations
- Useful to quickly prototype and test crazy ideas
- Better debugging and observability

Future plans

- Standardize the user-space framework APIs
- More topology awareness APIs
  - Cache, GPU, ...
- Introduce concept of scheduling domains
  - Allow to define domains (via cpumask)
- More modularization
- Auto-generate schedulers
  - Profile-based schedulers? AI?

References

- scx_rust_scheduler template (github)
  - https://github.com/arighi/scx_rust_scheduler
- scx_rustland_core (github)
  - https://github.com/sched-ext/scx/blob/main/rust/scx_rustland_core/README.md
- Writing a scheduler for Linux in Rust that runs in user-space (arighi blog)
  - https://arighi.blogspot.com/2024/02/writing-scheduler-for-linux-in-rust.html
- The extensible scheduler class (LWN.net)
  - https://lwn.net/Articles/922405/

Questions?

Andrea Righi
Linux Plumbers Conference 2024 | Vienna