FOSDEM

Rust-ifying the Linux kernel scheduler

[...in user space]

Andrea Righi <arighi@nvidia.com>

NVIDIA

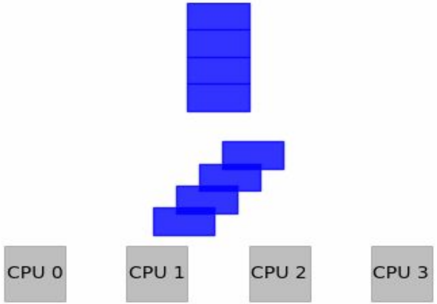
Agenda

- Scheduling
- sched_ext & BPF
- Linux scheduler in Rust
- Conclusion

Scheduling

What is a scheduler

- Kernel component that determines
    - **Where** each task needs to run
    - **When** each task needs to run
    - **How long** each task needs to run



Challenges

- Fairness
  - All tasks should receive a fair share of CPU
- Optimization
  - Make optimal use of system resources
- Low overhead
  - Should run as fast as possible
- Generalization
  - Should work on all architectures and for every workloads

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

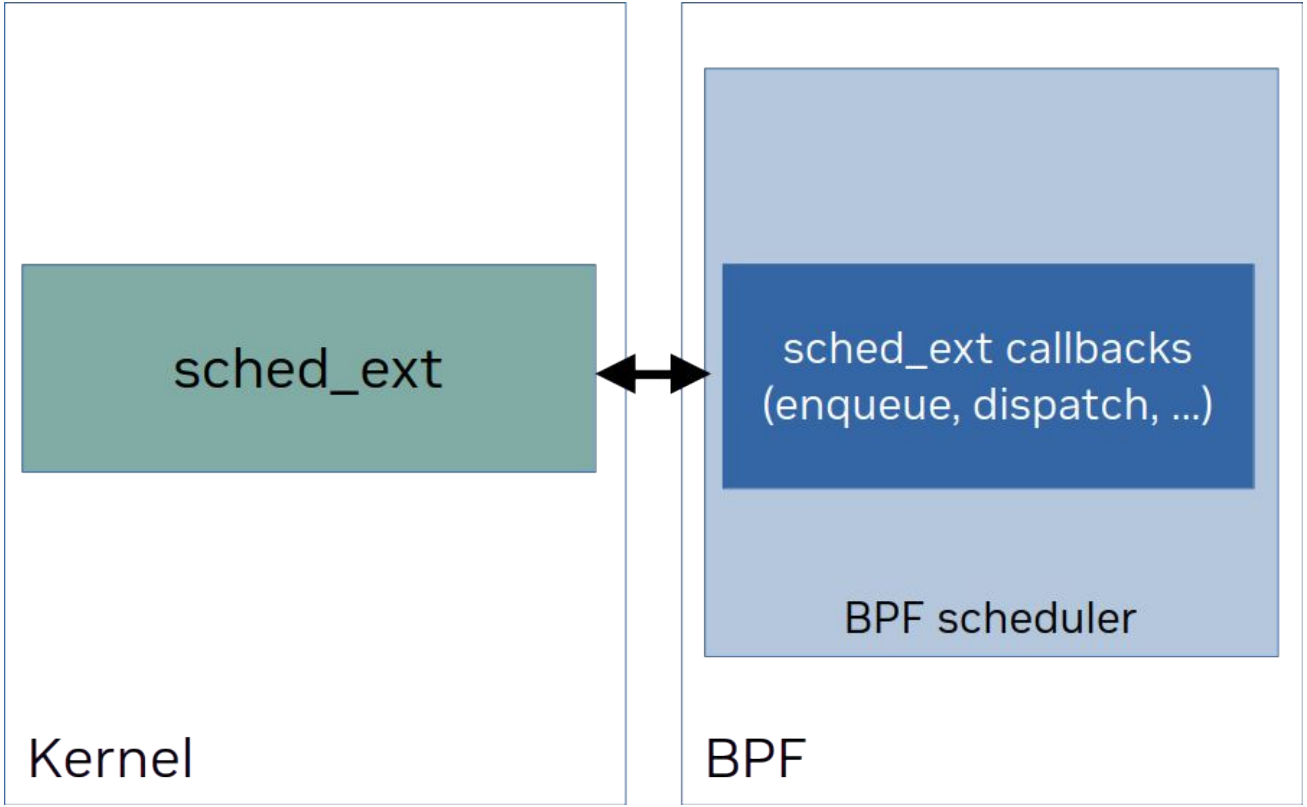
sched_ext & BPF

sched_ext: the extensible scheduling class

- Technology in the Linux kernel that allows to implement scheduling policies as BPF programs (GPLv2)
- Available since Linux v6.12
- Key features:
  - Bespoke scheduling policies
  - Rapid experimentation
  - Safety (can’t crash the kernel)

sched_ext

How does sched_ext work?



sched_ext limitations

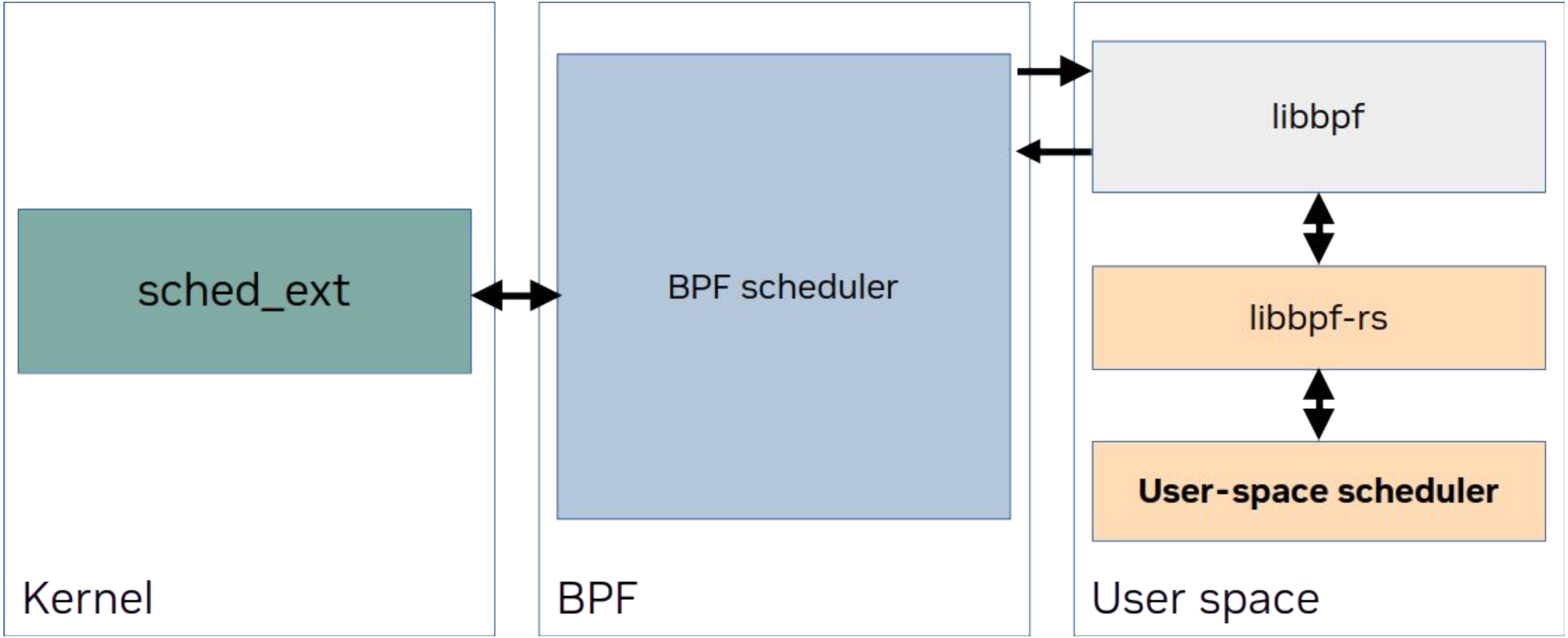
- Limited programming model (BPF)
- BPF verifier complexity
- Kernel restrictions (no user-space libraries, no floating point, etc.)

Linux scheduler in Rust

Idea

- Use sched_ext + BPF to channel scheduling events to user space and make all the scheduling decisions there
  - A scheduler becomes a regular user-space program
  - Offload complexity to user space
  - Access to user-space libraries and languages

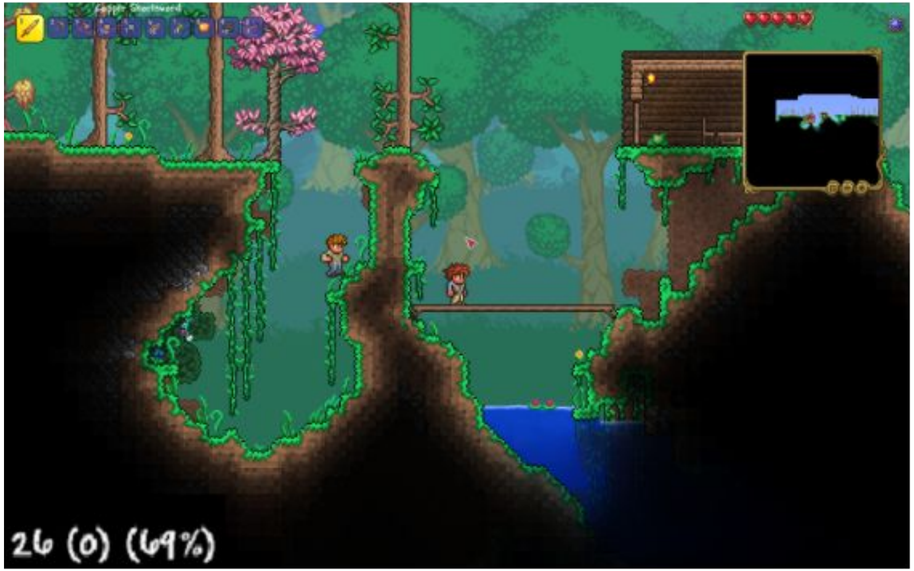
User-space Rust scheduler design



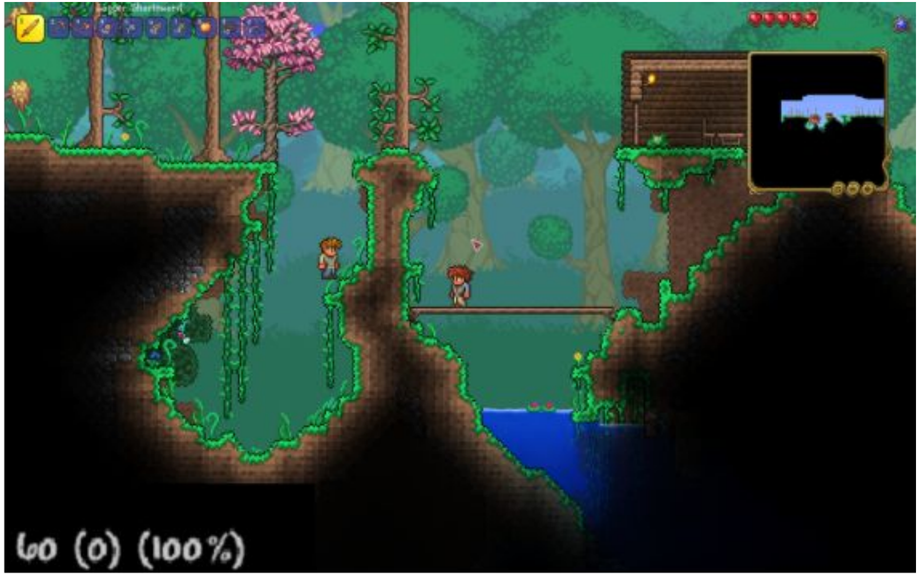
scx_rustland

- EDF-based scheduler
  - Deadline is evaluated as a function of the task’s vruntime and the rate of voluntary context switches
- Tasks receive a variable time slice inversely proportional to the total amount of tasks that are waiting to be scheduled

Playing Terraria while building the kernel



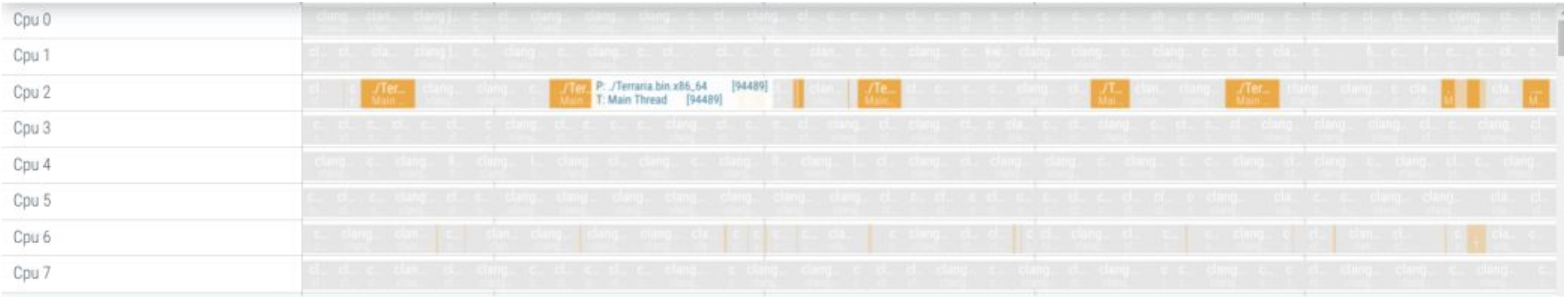
EEVDF



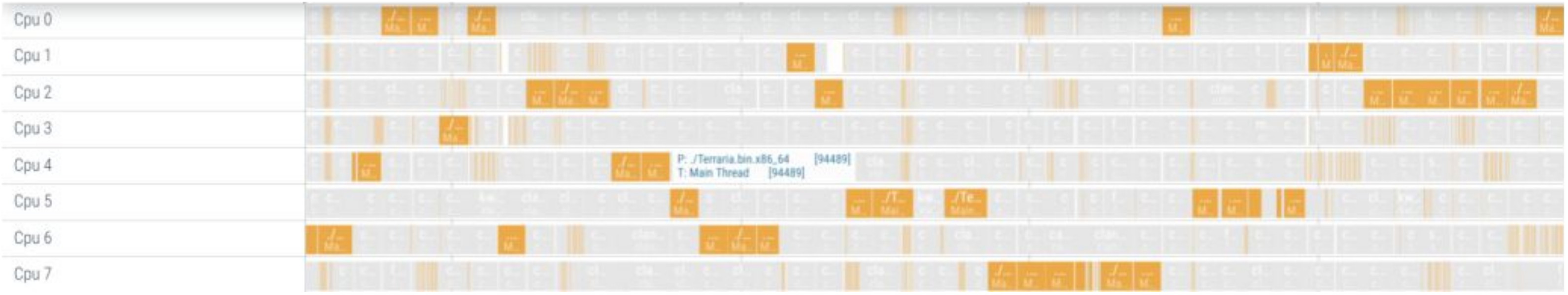
scx_rustland

EEVDF vs scx_rustland (https://perfetto.dev)

EEVDF



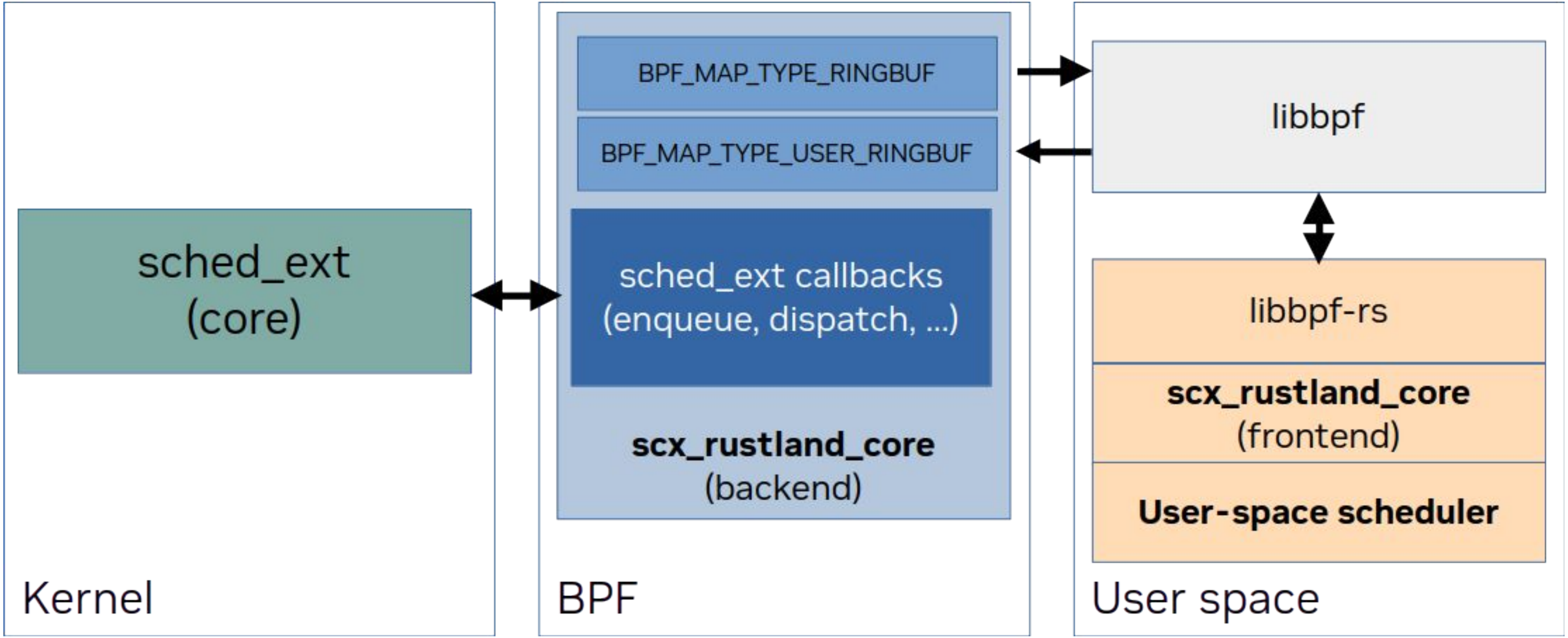
scx_rustland



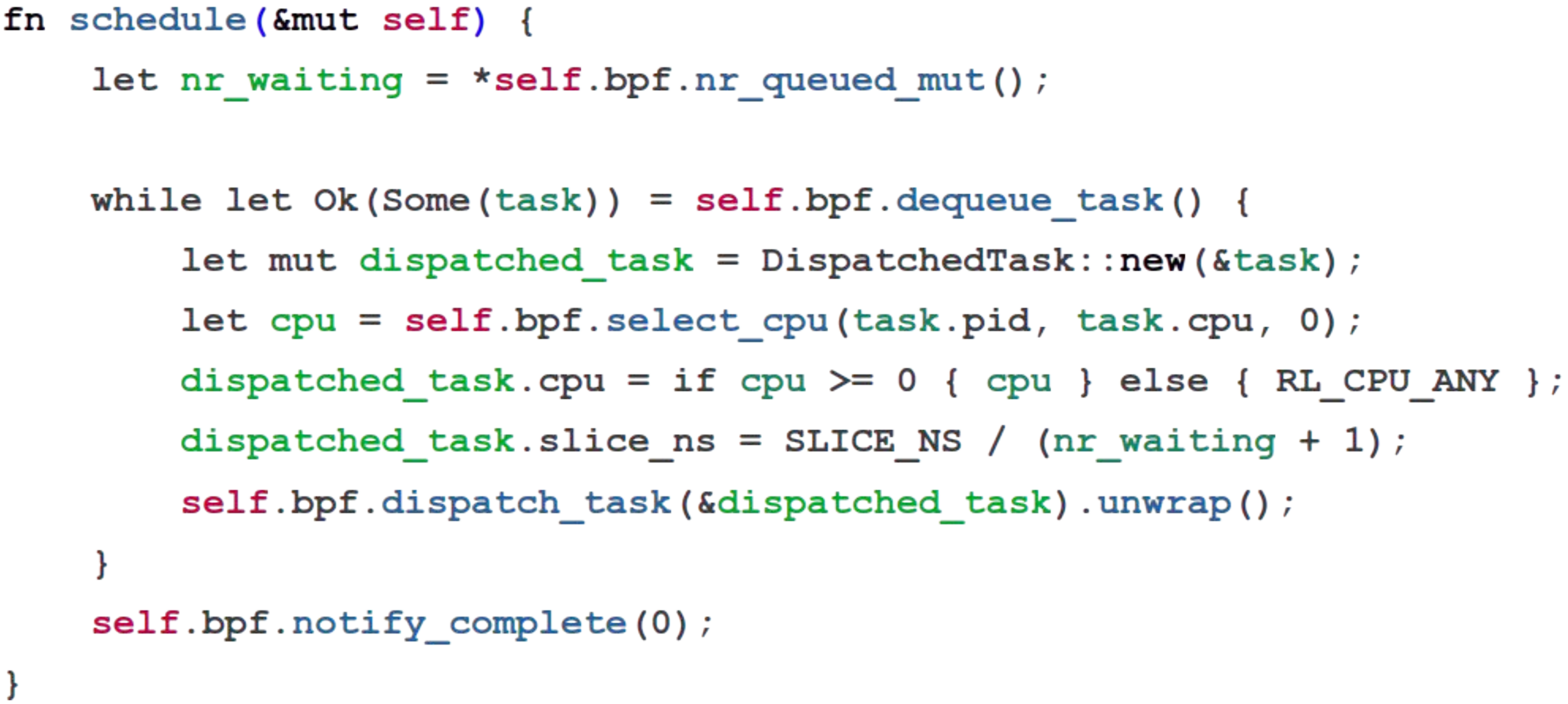
Generalize user-space Rust scheduling

- Rustland core framework
  - Abstract scx_rustland backend
  - Expose generic scheduling APIs
  - Available as a Rust crate (**scx_rustland_core**)
  - Allow to implement host-wide Linux scheduling policies easily, as regular Rust programs

scx_rustland_core design



FIFO scheduler in scx_rustland_core



```
fn schedule(&mut self) {
    let nr_waiting = *self.bpf.nr_queued_mut();

    while let Ok(Some(task)) = self.bpf.dequeue_task() {
        let mut dispatched_task = DispatchedTask::new(&task);
        let cpu = self.bpf.select_cpu(task.pid, task.cpu, 0);
        dispatched_task.cpu = if cpu >= 0 { cpu } else { RL_CPU_ANY };
        dispatched_task.slice_ns = SLICE_NS / (nr_waiting + 1);
        self.bpf.dispatch_task(&dispatched_task).unwrap();
    }
    self.bpf.notify_complete(0);
}
```

Conclusion

Key takeaways

- scx_rustland is not a better scheduler in general
- Rust itself doesn’t make scheduling faster
- Ease of experimentation is the key
  - Fast edit/compile/test turnaround
  - Integration with user-space components (Rust)

References

- Main scx repo
  - https://github.com/sched_ext/scx
- Rust scheduler template
  - https://github.com/arighi/scx_rust_scheduler

Questions?

Andrea Righi <arighi@nvidia.com>