How can we catch problems that can break the PREEMPT_RT preemption model?

Daniel Bristot de Oliveira

What is the main preempt rt feature?

- A preemption model in the kernel
- Our preemption model tries to make the kernel as preemptive as possible, by:
  - The preemption is enabled by default
    - Disabled on demand
  - Code that are specific for us
    - Enabled with #ifdeffery
  - We have the same lock assumptions, but different lock “positions”

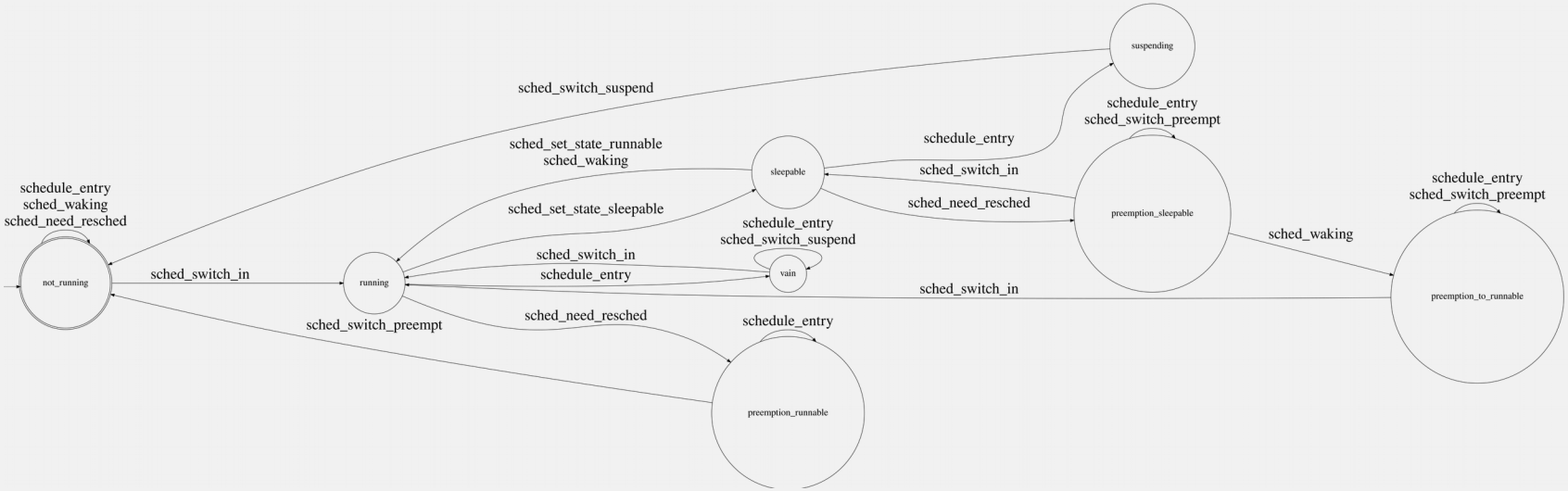
What is the main preempt rt feature?

- How do we catch problems nowadays?
  - Sched while in atomic?
  - Lockdep
  - We have some fragments of a check
- But we do not have a specific model check
  - What should we do?

What do I plan to do

- A formal model checker for the PREEMPT_RT
- It is based in the model I presented
  - Although it is for single core, it works for SMP as well
    - I just need to add migrate_disable/spin_locks to it

Calling scheduler

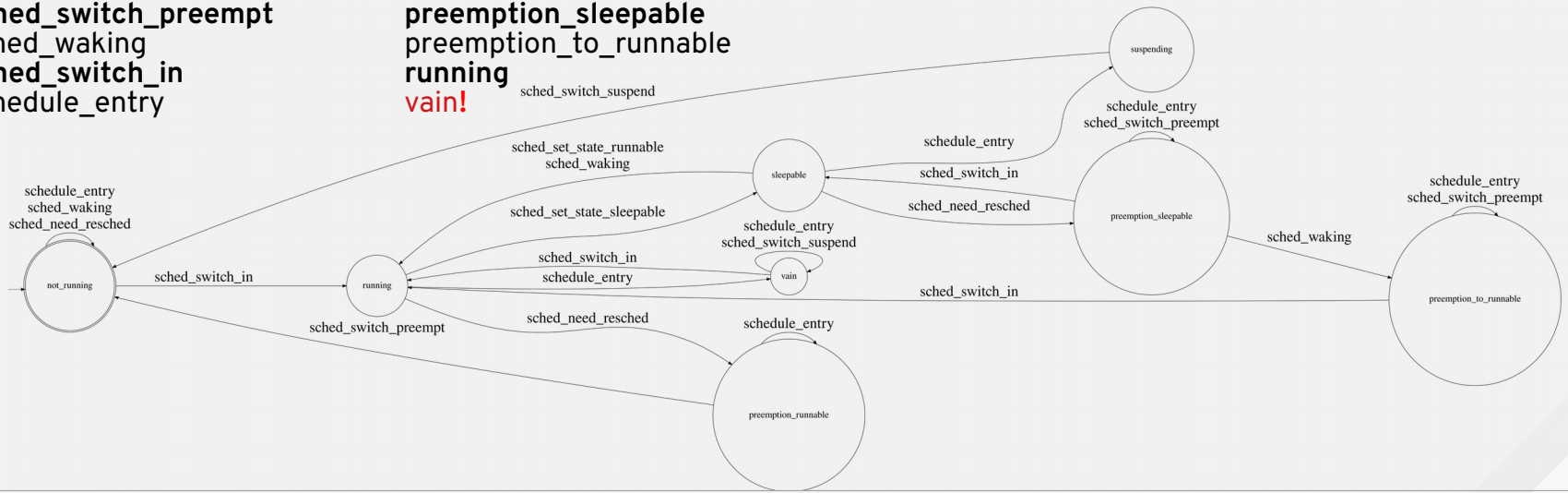


Reference tracing:

```
 1:   ktimersoftd/0   8 [000]   784.425631:  sched:sched_switch: ktimersoftd/0:8 [120] R ==> kworker/0:2:728 [120]
 2:     kworker/0:2 728 [000]   784.425926:  sched:sched_set_state: sleepable
 3:   kworker/0:2   728 [000]   784.425932:  sched:sched_waking: comm=kworker/0:1 pid=724  prio=120 target_cpu=000
 4:     kworker/0:2 728 [000]   784.425936:  sched:set_need_resched: comm=kworker/0:2 pid=728
 5:     kworker/0:2 728 [000]   784.425941:  sched:sched_entry: at preempt_schedule_common
 6:     kworker/0:2 728 [000]   784.425945:  sched:sched_switch: kworker/0:2:728 [120] R ==> kworker/0:1:724 [120]
 7:  irq/14-ata_piix 86 [000]   784.426515:  sched:sched_waking: comm=kworker/0:2 pid=728 prio=120 target_cpu=000
 8:     kworker/0:1 724 [000]   784.426610:  sched:sched_switch: kworker/0:1:724 [120] t ==> kworker/0:2:728 [120]
 9:     kworker/0:2 728 [000]   784.426616:  sched:sched_entry: at schedule
10:     kworker/0:2 728 [000]   784.426619:  sched:sched_switch: kworker/0:2:728 [120] R ==> kworker/0:2:728 [120]
```

Calling scheduler

Event	State
sched_switch_in	**running**
sched_set_state_sleepable	sleepable
sched_need_resched	**preemption_sleepable**
schedule_entry	preemption_sleepable
sched_switch_preempt	**preemption_sleepable**
sched_waking	preemption_to_runnable
sched_switch_in	**running**
schedule_entry	vain**!**



Logical correctness for task model

- Example of patch catch'ed with the model
  - [PATCH RT] sched/core: Avoid__schedule() being called twice, the second in vain
- I am doing the model verification in user-space now:
  - Using perf + (sorry, peterz) tracepoints
  - It works, but requires a lot of memory/data transfer:
    - Single core, 30 seconds = 2.5 GB of data
    - We don't need all the data, only from a safe state to the problem.
  - It performs well, because the automata verification is O(1).
  - But still, the amount of data is massive.

Should I move it to kernel?

- Think of a lockdep for PREEMPT_RT model:
  - If an unexpected event takes place, we explain why
  - Enabled in compilation time
  - Running in kernel would avoid copying data/keeping data after reaching a safe state
- This is helpful for safe critical systems
  - CI
  - We might face more problems with merge with the non-rt
  - It observes more than just latency