Contribution ID: **345** Type: **not specified**

Runtime verification monitors for real-time

PREEMPT_RT has finally been merged. The mainline Linux kernel is now capable of providing real-time guarantees. However, the kernel is only one piece of the puzzle: to achieve real-time behavior, the userspace counterpart must also be designed correctly.

Unfortunately, userspace applications often introduce undesirable latency due to incorrect design. The root cause is that it's unclear what users should and shouldn't do to guarantee a real-time response. For example, many users are unaware that they should call mlockall(), and they suffer unwanted latency from page faults. Another common issue is using mutexes without priority inheritance, which can lead to priority inversion.

Runtime verification (RV) monitors are a recent effort to help users detect these incorrect real-time design patterns. At present, the monitors warn users if one of the following occurs:

1. A real-time task generates a page fault
2. A lower-priority task wakes a higher-priority task
3. A real-time task sleeps using an RT-unsafe API (anything except futex_wait and clock_nanosleep)

In this talk, we will discuss further enhancements to RV monitors:

1. Should RV monitors warn when a real-time task blocks on read()/write() for: timerfd, ext4 file, socket, eventfd, other blocking I/O?
2. Should RV monitors warn when a real-time task blocks on epoll(), poll(), or ioctl()?
3. Should RV monitors warn about real-time tasks calling non-PI (non–priority-inheritance) futexes?
4. What other user-space patterns should or should not be flagged?
5. Can we establish a definitive list of RT-safe interfaces, subsystems, and drivers?

Primary author: CAO, Nam (Linutronix)

Presenter: CAO, Nam (Linutronix)

Session Classification: Scheduler and Real-Time MC

Track Classification: Scheduler and Real-Time MC