Enforcing PI locks by default

Qais Yousef <qyousef@layalina.io>
John Stultz <jstultz@google.com>

東京 2025
Linux Plumbers Conference
TOKYO, JAPAN / DEC. 11-13, 2025

What is the problem?

- Inversion problems are common in practice
- Modern application are complex and layered, no single entity knows all the possible locks that can be held
- **SCHED_NORMAL** is the major concern; not a SCHED_FIFO/SCHED_RR only problem
- We need to enforce **PTHREAD_PRIO_INHERIT** by default
- We need help answering 4 questions:
  - ***How can we give admins the option to choose PI as the default behavior for all locks?***
  - ***How can we help lock implementations outside of pthread/libc to opt in?***
  - ***What do we need to implement PI for non blocking primitives?***
  - ***Importance of unfair locks and performance issues***

P is for Performance

- Priority is one aspect of performance that needs to be inherited
- A generalized solution is required. Proxy Execution is making great progress and should be available ‘soon’
    - Inheriting CFS bandwidth, cgroup.shares, uclamp and other scheduler attribute the impacts performance is important and required
    - Without the generalized inheritance **and** userspace opt-in, inversion problems will remain a major performance bottleneck in practice
    - Unrealistic to expect all lock users to opt-in, it must be enforced at toolchain/system level somehow

How can we make PI the default behavior?

- Apple has flipped the switch since many years for pthread_mutex
- What does it take for Linux to follow suit?
  - We can't switch syscall to futex_pi by default, userspace must opt-in

Possible solutions:

ENFORCE_PI_LOCK env variable to help switch the default behavior at runtime. **If defined** all locks will be PI. libc and private lock implementations from all apps/toolchains can use it to switch the implementation at **exec/load** time. 1	Can a new **Futex lock** primitive help? Similar to futex_pi, so lock owner can be determined, but without the strict rt lock handoff rules. Will it make changing the default easier at kernel level? 2

Not all lock users are pthread based

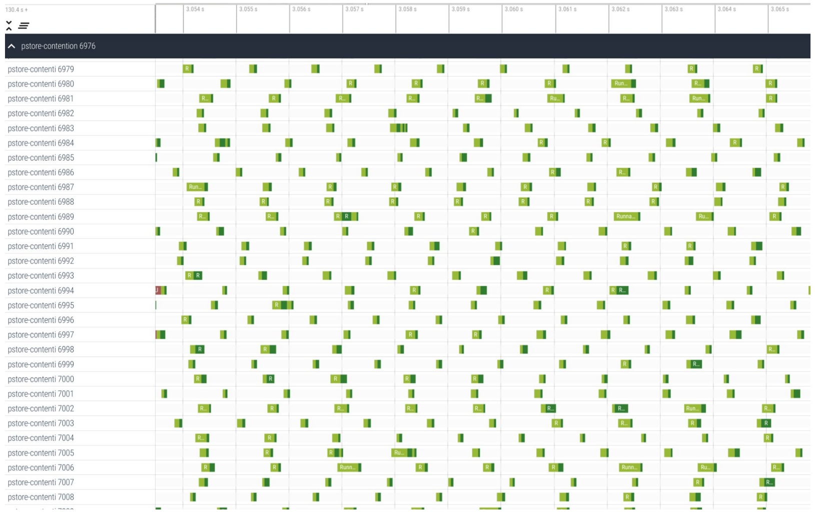
- Language runtimes such as Java or golang often implement their own locks without using pthread
- How can we ensure these locks become PI aware?
  - Is this a technical or social problem? Or both?
  - Technical: Is there anything preventing current users to move to futex_pi by default?
    - futex_pi API is more constrained comparing to regular futex
      - Various requeue methods, ability to wakeup multiple waiters, etc
    - Some languages have built in synchronization semantics that makes it harder to track the owner, which is required for PI
  - Social: Do we need to advertise the need more loudly and broadly?

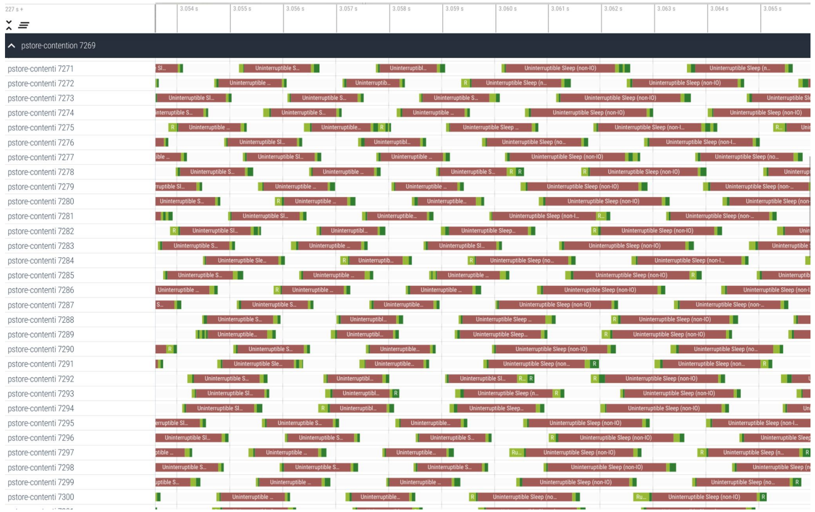
Inheritance is important outside of locks

- Condition variables and binders are example of non blocking dependency where inheritance enforcement is required
- Do we need a **new annotation mechanism** (call it **Performance Lock**) to denote important performance critical section?
    - Potentially allow for timeslice extensions / preemption avoidance
    - Possibly usable to help with futex owner issue without additional Futex lock?
- What other higher level primitive beside condition variables and binder we need to work on?

PI lock performance and fairness

- Experimental data shows that futex_pi suffers in performance by default
  - Mostly due to fairness issue and strict lock handoff rules
- Desire for unfair lock primitive that enables opportunistic grabbing of the lock which is not supported by futex_pi today





Questions