Contribution ID: **451** Type: **not specified**

Adaptive futexes: spin if the owner is still running on another CPU

Thursday 11 December 2025 16:00 (20 minutes)

Currently a futex does not expose any information about the owner of the futex to the kernel. When a task blocks on a futex it updates information on the shared memory of the futex about it waiting and enters the kernel to sleep until the owner wakes it up. For a futex that is held for a short time, this can cause a noticeable performance hit because the time it takes for a blocked task to sleep and then be woken up can be much longer than the futex was held. Contention in these cases has a significant performance impact.

If the memory for the futex held ownership information, instead of sleeping when a task can't get the futex, it could act like the adaptive mutexes in the kernel and spin if the owner is running on another CPU. When the owner releases the lock, the blocked task will detect that the owner is releasing the futex and exit out and even take the futex from within the kernel. This could possibly be a tremendous speedup to futexes.

To make this work, a contract is needed between the C library and the kernel to allow this information to be passed between user and kernel space.

Primary author: ROSTEDT, Steven

Presenter: ROSTEDT, Steven

Session Classification: Toolchains MC

Track Classification: Toolchains MC