Contribution ID: **223** Type: **not specified**

Overeager pulling from wake_wide() in interrupt heavy workloads

Monday, 20 September 2021 07:40 (30 minutes)

On the CFS wakeup path, wake_wide() doesn't always behave itself very well in interrupt-heavy workloads. We have systems configured with static IRQ bindings because IRQs are served faster on certain CPUs due to hardware topology. We then noticed on these systems that wakeups kept pulling tasks to the socket serving network IRQs while leaving the other socket nearly idle on a read-only workload from YCSB, an open source database benchmark. On a lightly loaded system with two 32-core sockets, wake_wide() led the scheduler to wake affine most of the time. Wake affine is a two-pass process involving both wake_wide() and wake_affine(), but wake_wide() is the more dominant factor than wake_affine() in our workloads. Periodic and idle load balancing must work to undo wake affine's overeager pulling, but ultimately network interrupts are so frequent in YCSB that wake affine wins out.

So far, we've gotten mixed results when trying to address the performance hit these issues cause. Disabling wake_affine() causes the benchmark to improve by 10-15% on a lightly loaded system (fewer DB connections) but slow down by up to 17% on more heavily loaded systems (more DB connections). wake_wide() works well when the waker and wakee are related, but we need a better heuristic for wakeups in interrupt heavy workloads, where the interrupt may or may not be related to the wakee.

A better heuristic ideally should be able to determine which CPU's cache is warmer for the wakee and doesn't cause excessive pulling.

I agree to abide by the anti-harassment policy

I agree

Primary author: CHEN, Libo (Oracle)

Presenter: CHEN, Libo (Oracle)

Session Classification: Scheduler MC

Track Classification: Scheduler MC