Contribution ID: **259** Type: **not specified**

QPW: How to improve latency and CPU Isolation without cost

Thursday, 19 September 2024 16:06 (22 minutes)

Some kernel code implement a parallel programming strategy
that grabs local_locks() for most of the work, and then use schedule_work_on(cpu) when some rare remote operations are needed. This is quite efficient for throughput, since it keeps cacheline mostly local and avoid locks in non-RT kernels, paying the price when you need to touch a remote CPU.

On the other hand, that's quite bad for RT kernels, as touching other CPU's data will require that CPU to interrupt any RT task it's running in favor of executing the requested task, while the requestor CPU waits for it's completion.

To solve that, I propose a new QPW interface that harness the local_lock() -> spin_lock() implementation in PREEMPT_RT to avoid above mentioned interruption without requiring extra cycles in the hot-paths, and actually causing a major reduction in the time spent by the requesting task itself.

This presentation will show the idea behind the interface, and bring numbers on latency and throughput improvements for some of the potential users of this interface.

Primary author: SOARES PASSOS, Leonardo Bras (Red Hat)

Presenter: SOARES PASSOS, Leonardo Bras (Red Hat)

Session Classification: Real-time MC

Track Classification: Real-time MC