Contribution ID: **54** Type: **not specified**

Reduce synchronize_rcu() latency

Thursday, 19 September 2024 10:00 (45 minutes)

Read-copy update (RCU) ensures that any update carried out prior to the beginning of an RCU grace period will be observed by the entirety of any RCU reader that extends beyond the end of that grace period. Waiting for grace periods is the purpose of the synchronize_rcu() function, which waits for all pre-existing readers in a throughput-optimized manner with minimal impact on real-time scheduling and interrupt latencies, but which might well wait for many tens of milliseconds.

This synchronize_rcu() function is a key component of per-CPU reader-writer semaphores, where it enables writers to wait until all readers have switched to the writer-aware slow path. In the scheduler's CPU-deactivate code, synchronize_rcu() waits for all readers to become aware of the inactive state of the outgoing CPU. A few other examples uses include module unload, filesystem unmount, and BPF program updates. Therefore, improving synchronize_rcu() latency should improve the latency of a great many Linux-kernel components.

This talk will present an analysis of synchronize_rcu() latency that identified issues during RCU callback floods, that is, high call_rcu() invocation rates. These issues motivate a new approach that decouples processing of synchronize_rcu() wakeups from the processing of RCU callbacks. This approach provides from 3-22% improvements in launch latency for an Android camera application when running on devices that do not boot with synchronize_rcu() mapped to synchronize_rcu_expedited(), a choice that might help avoid jitter in real-time applications.

However, there are currently a few downsides of this low-wait-latency synchronize_rcu() implementation: (1) The global wait list will result in excessive contention on systems with many CPUs; (2) Wakeups depend on kworker execution which might degrade wait latency; (3) Wakeups are carried out in LIFO order, and (4) Potential issues that might arise from high synchronize_rcu() invocation rates, for which RCU's existing callback handling has been heavily optimized. Due to these downsides, the current implementation is enabled only on systems such as embedded devices having low CPU counts. Future work will address these downsides in the hope that low-wait-latency synchronize_rcu() can be deployed by default on all systems.

Primary authors: UPADHYAY, Neeraj; MCKENNEY, Paul (Meta); REZKI, Uladzislau

Presenters: UPADHYAY, Neeraj; MCKENNEY, Paul (Meta); REZKI, Uladzislau

Session Classification: Kernel Summit

Track Classification: Kernel Summit Track