Linux 5.11 To Land Optimization That Helps IO_uring Performance - Phoronix

At the start of October we mentioned a kernel optimization that can help IO_uring performance. Now as we approach the end of the month, Linux 5.11 is poised to land the optimization that especially helps out with threaded workloads.

The change to task_work to use TIF_NOTIFY_SIGNAL when available is queued as part of the tip.git core/entry code ahead of the Linux 5.11 merge window opening in December. Currently TIF_NOTIFY_SIGNAL is wired up for x86/x86_64 while Jens is working on adding this support to other CPU architectures as well. We'll see how many architectures get supported in time for Linux 5.11 as once completing that work he'll be able to move on with a set of clean-ups.

Even more reading of the Linux-kernel Documentation/RCU/Design/Requirements/Requirements.rst file encounters RCU’s memory-barrier guarantees. These guarantees are a bit ornate, but roughly speaking guarantee that RCU read-side critical sections lapping over one end of a given grace period are fully ordered with anything past the other end of that same grace period. RCU’s overall approach towards this guarantee is shown in the Linux-kernel Documentation/RCU/Design/Memory-Ordering/Tree-RCU-Memory-Ordering.rst file, so one approach would be to argue that these guarantees are
proven by a combination of this documentation along with periodic code inspection. Although this approach works well for some properties, the periodic code inspections require great attention to detail spanning a large quantity of intricate code. As such, these inspections are all too vulnerable to human error.

Another approach is formal verification, and in fact RCU’s guarantees have been formally verified. Unfortunately, these formal-verification efforts, groundbreaking though they are, must be considered to be one-off tours de force. In contrast, RCU needs regular regression testing.

---


The first guarantee is trivially verified by inspection of the RCU API. The type of rcu_read_lock(), rcu_read_unlock(), synchronize_rcu(), call_rcu(), and rcu_assign_pointer() are all void. These API members therefore have no way to indicate failure. Even primitives like rcu_dereference(), which do have non-void return types, will succeed any time a load of their pointer argument would succeed. That is, if you do rcu_dereference(*foop), where foop is a NULL pointer, then yes, you will get a segmentation fault. But this segmentation fault will be unconditional, as advertised!

---

AMD Navi "Blockchain" Card Support Being Added To Linux 5.10[6]

Last week we were first to report on a PCI ID being added for a Navi 1 "Blockchain" graphics card without display outputs and seemingly focused on cryptocurrency mining. This card wasn’t talked about at yesterday’s Radeon RX 6000 series launch but that support is now on the way to the Linux 5.10 kernel.

The code sent out last week added the new Navi 10 PCI ID and disabled DCN/VCN support for that ID with this card not having video acceleration or display functionality. Aside from that patch, AMD hasn’t officially acknowledged this new part that is RDNA (1) and not to be confused with the forthcoming RDNA2 / RX 6000 series products.

---

Source URL: http://www.tuxmachines.org/node/143815

Links: