RCU Seq Visualization and “done” API Behavior Analysis

(Please feel free to add a link to this to the RCU tranche of google docs..)

Summary of Findings

- **rcu_seq_done()**: Returns correct results as long as the distance between the "running sequence" and the "target" is within ULONG_MAX/2. Outside of this range, false results can occur.
- **rcu_seq_done_exact()**: Behaves similarly to rcu_seq_done(), but the range for false-negative results is shrunk to approximately 10 counts from the target
- **False Positives**: Cannot occur as long as the same sequence is being sampled at 2 different times. Example, in rcu_barrier, the same rcu_barrier sequence is sampled and passed to the DONE APIs. If a long time passes and the sequence progresses a lot, between the 2 samples, a positive result is correct and a negative result is false negative.
- **False Negatives**: Can occur if the initial value of the sequence snapshot taken has now progressed to more than ULONG_MAX/2 for rcu_seq-done() distance from the target (with or without wrap around). In the case of rcu_seq_done_exact(), this false-negative band is shrunk down to ~2.5 GPs. Due to this, rcu_seq_done_exact(), a FULL wraparound is likely to have happened.
    - This is why rcu_seq_done_exact() makes sense for polling. It is possible that between get() and poll() a large time passes. A full wrap around has to have occurred in order to induce false-negatives.
    - Even if false-negative happens, unlike rcu_seq_done(), rcu_seq_done_exact() only spends 2-3 GPs in this zone.

Key Questions and Implications

- **Consequences of Failure (API giving wrong results)**
    - False positives are generally unacceptable, as they can break functionality in code. It implies an operation (like GP or barrier both of which use their own sequences) completed when it did not.

    - False negatives may be acceptable if the use case can retry later and afford to wait and the wait is not too long for the system to respond correctly.
- **Purpose of →gpwrap**
    - Further investigation is needed to understand the specific purpose of →gpwrap within the code.unintended consequences, making them generally undesirable.

- **Analyze Specific Use Cases**: Identify the scenarios where the API is employed and assess the consequences of false positives and false negatives for each case.
    - rcu_barrier: The same rcu_barrier sequence is checked at 2 different points of time. A large delay between sampling rcu_barrier sequence at 2 different times is unlikely here as rcu_barrier() is expected to be a relatively quick operation. The worst case is we'd do an rcu_barrier() again if there was a false-negative. For this reason, **rcu_seq_done() suffices for rcu_barrier() but rcu_seq_done_exact() cannot seem to hurt and may be slightly better.** **More notes.**

    - RCU polling: This uses rcu_seq_done_exact() as it should, because long delays are possible between sampling of the rnp→gp_seq. If such long delays ensue and rcu_seq_done() were to be used, it is possible that the **poll() API would get stuck returning false-negatives for a long period of time.**

        - RCU NOCB: Used in path that queues CBs, but queue is too long or too full of bypass CBs (also rcu_cb_wait()) has a similar pattern. Here rcu_advance_cbs_nowake() is best-effort anyway. So a false-negative doesn't really hurt correctness. It just means we wont advance CBs when we could have but we would advance them in the next opportunity.

```
        if (j != rdp->nocb_gp_adv_time &&
            rcu_segcblist_nextgp(&rdp->cblist, &cur_gp_seq) &&
            rcu_seq_done(&rdp->mynode->gp_seq, cur_gp_seq)) {
            rcu_advance_cbs_nowake(rdp->mynode, rdp);
            rdp->nocb_gp_adv_time = j;
        }
```

Later the nocb GP thread does this in nocb_gp_wait(), here it seems a false negative might be a bad thing. This can happen say if a CB was queued long time ago and GP thread was delayed, however that does seem unlikely:

```
        // Advance callbacks if helpful and low contention.
        needwake_gp = false;
        if (!rcu_segcblist_restempty(&rdp->cblist,
                                     RCU_NEXT_READY_TAIL) ||
            (rcu_segcblist_nextgp(&rdp->cblist, &cur_gp_seq) &&
             rcu_seq_done(&rnp->gp_seq, cur_gp_seq))) {
            raw_spin_lock_rcu_node(rnp); /* irqs disabled. */
            needwake_gp = rcu_advance_cbs(rnp, rdp);
```

Then later during the actual wait it does this:

```
else {
        rnp = my_rdp->mynode;
        trace_rcu_this_gp(rnp, my_rdp, wait_gp_seq, TPS("StartWait"));
        swait_event_interruptible_exclusive(
            rnp->nocb_gp_wq[rcu_seq_ctr(wait_gp_seq) & 0x1],
            rcu_seq_done(&rnp->gp_seq, wait_gp_seq) ||
            !READ_ONCE(my_rdp->nocb_gp_sleep));
        trace_rcu_this_gp(rnp, my_rdp, wait_gp_seq, TPS("EndWait"));
    }
```

Here also rcu_seq_done() returning false-negative might be bad as the wakeup may fail.

- SRCU in srcu_funnel_gp_start():
  Here a false-negative just means we would start a GP unwantedly when we didn't need to. So perhaps it is a bit faster if we would not unwantedly do so by using done_exact() but no issue either way.

```
  /* If grace period not already in progress, start it. */
  if (!WARN_ON_ONCE(rcu_seq_done(&sup->srcu_gp_seq, s)) &&
      rcu_seq_state(sup->srcu_gp_seq) == SRCU_STATE_IDLE) {
      srcu_gp_start(ssp);
```

- SRCU polling:

poll_state_synchronize_srcu() uses rcu_seq_done()

```
    if (cookie != SRCU_GET_STATE_COMPLETED &&
        !rcu_seq_done(&ssp->srcu_sup->srcu_gp_seq, cookie))
        return false;
```

where as poll_state_synchronize_rcu() uses rcu_seq_done_exact()

```
bool poll_state_synchronize_rcu(unsigned long oldstate)
{
    if (oldstate == RCU_GET_STATE_COMPLETED ||
        rcu_seq_done_exact(&rcu_state.gp_seq_polled, oldstate)) {
```

I believe rcu_seq_done_exact() makes more sense for polling API, as there is a higher chance that there is a significant delay between the get_state..() and poll_state..() calls.

I think I am pretty convinced now looking at all the call sites that rcu_seq_done_exact() should be used everywhere. I am vetting more callsites, but that's what I'm leaning towards. I think rcu_seq_done_exact() makes the code more robust to false-negatives (duration during which a false-negative is in effect)...

Next Steps

1. **Investigate the Purpose of →gpwrap**: Understanding the role of →gpwrap may provide insights into the design and potential implications of the API's behavior.
    - →gpwrap is set when CPU noting its QS notes that gpwrap may happen
    - If a wrap happens, it is possible that rnp→gp_seq will be less than the current value of rdp→gp_seq.
    - Any CBs assigned rdp→gp_seq in the segcblist should not look like they can be forwarded in the CB list to the done list.
    - advance() should be called carefully.

Resources:

1. This helps visualize the responses of the rcu_seq_done* APIs and shows the narrowing of the false-negative band (example, see the last column of both tables)
   RCU Sequence Visualization - 5bit.pdf