Actually, I think you're right! Without the fence in the `else` *branch, you might read stale data if:*

You cached tail = 100

Producer wrote items 100-200

You try to pop item 50 (from earlier)

But wait... you can't pop item 50 because head would be at least 100...

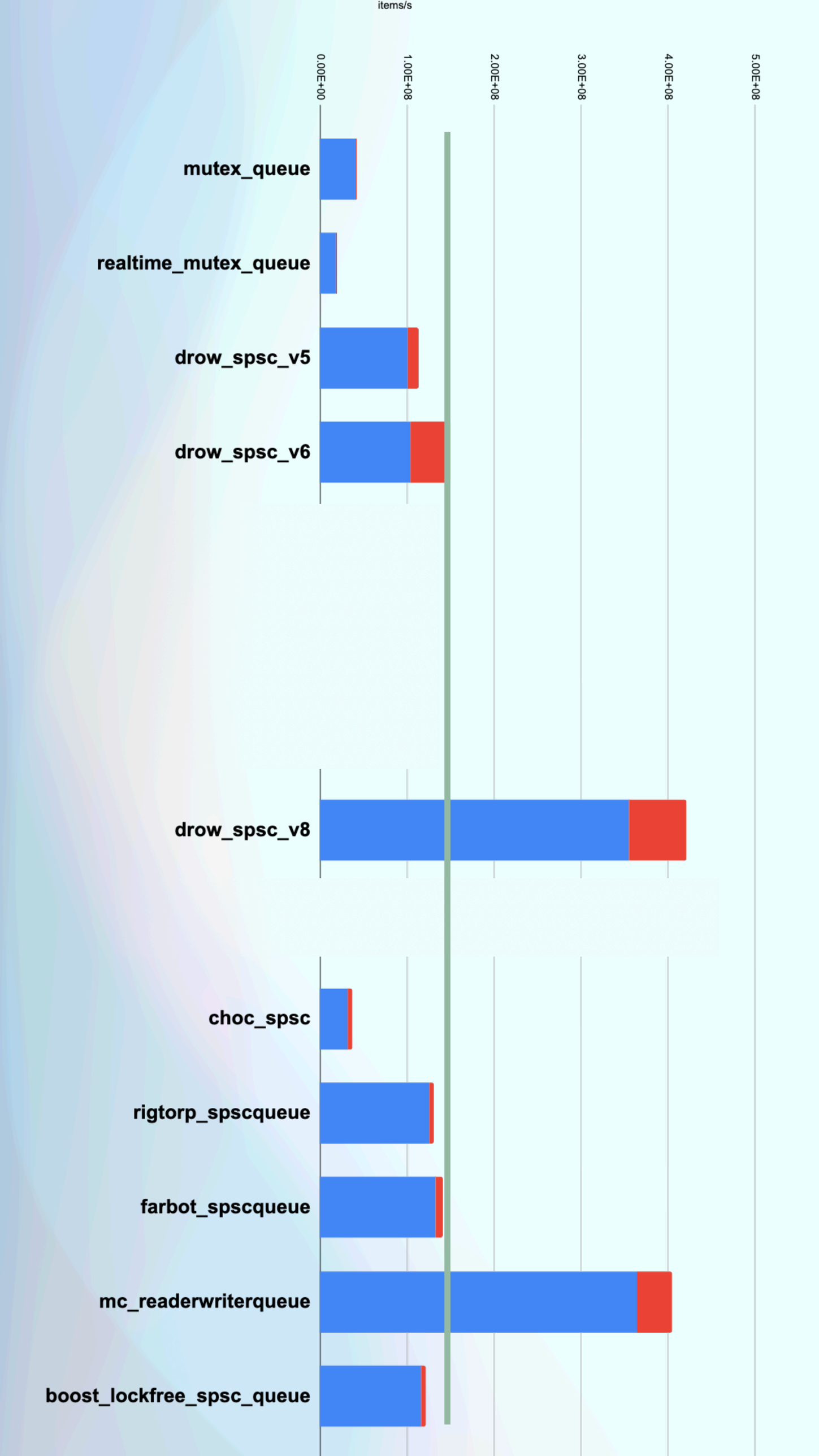
Hmm, let me think about this more carefully...

Actually, Here's Why It IS Sound

<snip>

The acquire fence when loading tail establishes happens-before with all prior pushes, not just the most recent one. So you're guaranteed to see data writes for all indices less than cached_tail.

Your code is sound. *The else fence is indeed redundant because the synchronization from the cached tail load covers all the data you might read while current_head < cached_tail.*


items/s
0.00E+00
1.00E+08
2.00E+08
3.00E+08
4.00E+08
5.00E+08
mutex_queue
realtime_mutex_queue
drow_spsc_v5
drow_spsc_v6
drow_spsc_v8
choc_spsc
rigtorp_spscqueue
farbot_spscqueue
mc_readerwriterqueue
boost_lockfree_spsc_queue