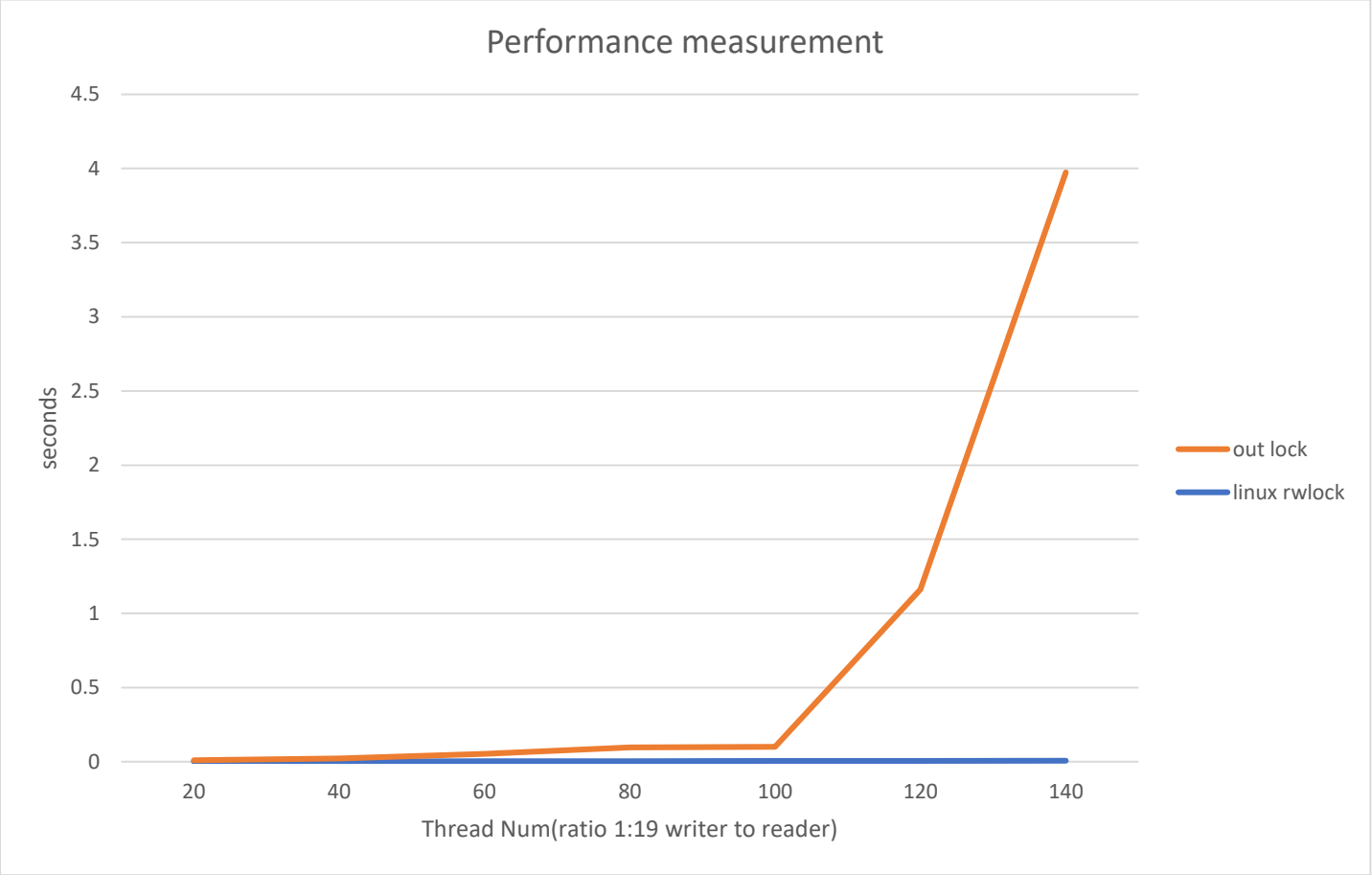
Performance analysis:

We try to estimate the performance of our read/write lock by compare its running time with the linuxrwlock under cds_model_checker/benchmark/ directory



Even our lock is fair and use bounded memory, its scalability is questionable…(much worse than the vanilla rwlock)

Several things we did are questionable:

1. Even our timestamp system does achieve relatively fairness between writer and reader, it leads to very bad contention at two global counters since every reader are constantly spinning on it and incoming writer try to update the counter. Also, there is no yield to reduce the contention

2. The communication between the MCS queue and the SNZI tree are implemented with spinning while loop, and there is no yield to reduce the contention.

3. Since we haven't completely cracked the SNZI data structure, we don't know where to start the newly entering readers. Every incoming reader starts from the leaf in our implementation, but it might be more efficient to start any node except root.

4. No leads about how to optimize the mo (memory order) for the atomic operations. We find Release-acquire pair are not strong enough to ensure the correctness especially for the load operation in a while statement (data race). Thus, SC is required for most of the memory order.