Experiment Setup:

- KVM hypervisor, virt-manager, Ubuntu 16.04 VM, 2GB RAM, 2 CPUs
- Avg workload:
    - CPU1: ~15%
    - CPU2: ~15%
    - MEM: ~50%
- We have used following set of parameters for comparison
    - % read=100, threads=8, ops/thread=50000
    - % read=99, threads=8, ops/thread=50000
    - % read=95, threads=8, ops/thread=50000

Result:

- Average time:
    - Average time taken by all locks increases with the increase in writers %
    - For all cases, RCU takes least time as it does not have any overhead of changing an external variable indicative of a lock. However, it does have overhead of copying shared data again and again, but it is outweighed for our cases.
    - For all cases, spinlock performs worst as it does not allow parallel readings
- Average Read time:
    - Spinlock read time is invariant to the % writers because of its fair nature towards writers and readers
    - For each case, RCU takes least time followed by seqlock, RWlock and custom RW. Spinlock takes longest reading time
    - RCU takes least time because of its not dealing with any external variable.
    - This is followed by Seqlock because a reader can can enter CS without any constraint (it may have to reread).
    - RWlock and custom RW lock follows as sometimes readers have to wait for completion of a writer
    - Barring Spinlock, RCU read time changes the least with writers % change as RCU readers don’t have to care about no. of writers
- Average Write Time:
    - Spinlock write time is same irrespective of % writers
    - RWlock and custom RWlock write time is largest because writers in this case can starve.
    - As % writers increases, RWlock write time decreases. This is because of lesser possibility of writers starvation per writer as % readers decreases
    - With the increase of % writers, Seqlock write time does not increase as writers can enter CS whenever they want

Insights:

- Spinlock does not differentiate between writers and readers, therefore it suffers the most. Hence, most inefficient.
- Seqlock is the most unfair lock as it allows every writer to write whenever they want causing the active readers to reread the data again
- RCU lock is quite fair as it allow readers and writers to do their operations independently
- The design of Custom RW lock is inspired from RWlock, therefore their behavior for all cases remains similar

- RCU lock is the most efficient lock as far as time taken to copy shared data again and again does not become dominant
- For very small numbers of writers as compared to readers, RWlock write time is quite high. This is because in such cases, writers may starve waiting for their chance