Linearizability

Correctness

Need to define what it means to be correct

- Will look at a KV store

In a sequential/serial specification, it is intuitive:

- Get operations should reflect the result of all previous Put

Under concurrent operations, operations could be sent while another one is still processing

TODO IMAGES

We formally specify correctness through a model called linearizability where every operation appears to execute atomically / instantaneously at some point between invocation and response

- Similar to strong consistency
- Linearizable systems tend to have poor performance because we have to wait for everything to replicate first
- Sharding can help with performance while maintaining linearizability
  - I.e. a different raft group can handle each shard

Linearizable

TODO IMAGES

Nonlinearizable

TODO IMAGES

Testing

General approach is to test for correct operation while randomly injecting faults such as machine failures and network partitions

- Tests are randomized, so we need to run them many times

Example Test

```
for client_id = 0..10 {
    spawn thread {
        for i = 0..1000 {
            value = rand()
            kvstore.put(client_id, value)
            assert(kvstore.get(client_id) == value)
        }
    }
}
wait for threads
```

If this test fails, then the KV store is not linearizable, but there are also non-linearizable KV stores that pass this

- Better idea would be to repeatedly call `kvstore.put(rand(), rand())` and `kvstore.get(rand)` on a limited number of keys
- However, this introduces complexity since how would we be able to tell if something is "correct" operation?
  - We have to record the entire history of operations on the system and check if this is linearizable

A linearizability checker takes a sequential specification and concurrent history and decides if it is linearable

- However, this is NP-complete since it can be reduced to the subset sum problem

- It can still work pretty well in practice