Fencing Tokens

Unreliable Nodes

Even when we control every single node in a system, it is very possible that one of the nodes believes something to be true, while the majority of the other nodes disagree with it. This can be very problematic if there is only supposed to be one of something.

One leader

One process holding a distributed lock

One account with username “poonslayer69”

Process Pauses

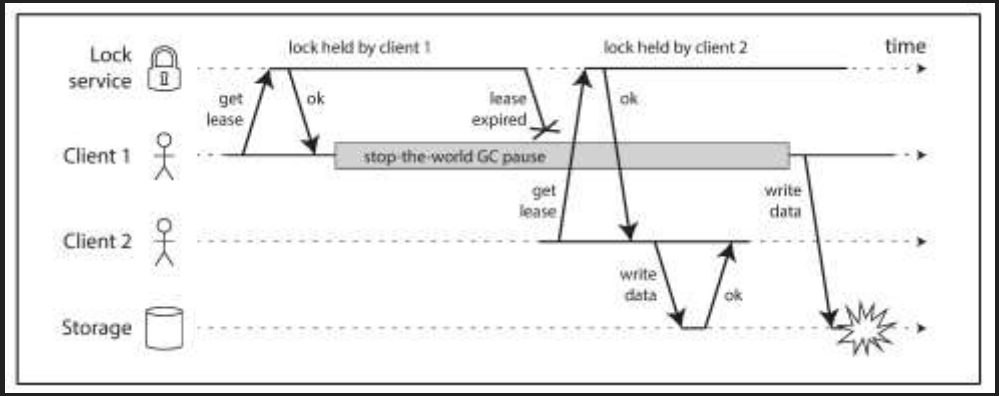
Often times, a process will be running and be preempted by another “stop the world” process for many seconds at a time

Examples:

- Garbage collection algorithms
- Virtual machines being moved from one host to another
- A user closes their laptop lid

Process Pauses

Often times, a process will be running and be preempted by another “stop the world” process for many seconds at a time



Process Pause Significance

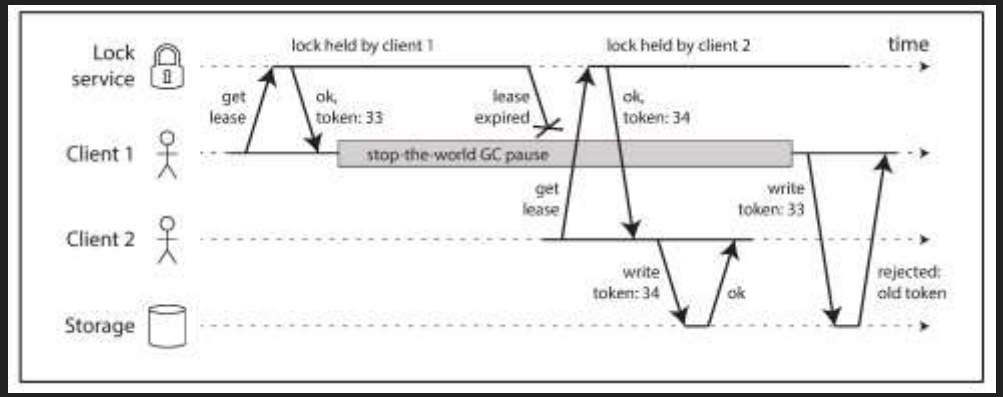
Since we don’t have any shared memory in distributed systems, a bunch of locking/election distributed implementations may use something called a “lease” which expires after a certain amount of time. A paused process could have its lease expire, and then potentially process a few more requests while it thinks it is holding the lease, until it checks the time again to realize it has expired.

Another case is where the other nodes in a cluster assume that a process paused node is dead, and when it comes back, it still thinks it is a leader/holding a lock and now we have a split brain situation!

Problem: How can we make sure that once the majority of nodes make a decision about a node, the other node can eventually cooperate with it? Fencing tokens

Fencing Tokens Overview

Idea: Everytime the majority of nodes make a decision, assign it a monotonically increasing ID. Pass this ID with every write, and if the storage service ever comes across a write with a lower fencing token value than the current one, reject it.



Fencing Tokens Conclusion

Due to process pauses, or possibly even just unreliable client nodes, we can only trust any decisions made by a majority of nodes in our cluster - any single node can be faulty (possibly without it even realizing!)

As a result, using something like fencing tokens is easy to implement and guards against things like multiple lock holders and split brain in single leader replication, and is an important part of any distributed system that requires consensus.