One concern that I have with current sharding models is that there are many situations in which there is an enshrined mechanism for performing some specific function related to state storage, but that enshrined mechanism has weaknesses, leading people to sometimes want to use their own layer-2 alternatives.

In all current proposals, there is a notion of “enshrined state storage”, but this storage costs rent. For applications that do not want to pay rent, there is the option of simply storing a 32-byte storage root in the enshrined state, and requiring every transaction that reads or updates that state to include a Merkle branch proving the update.

This leads to the following layer-2 machinery:

- Markets for paying people to continue storing the Merkle tree of application state, for the benefit of clients that are not online all the time to receive the updates. With current proposals, this exists on two levels: (i) contract storage, and (ii) data for reviving accounts that have been dormant for more than one year
- Proposers being able to update witnesses included with transactions (or even simply store application state and add witnesses) so that users don’t need to worry about witnesses themselves
- Incentives or other mechanisms to discourage/prevent users from filling up an application state tree forever, imposing permanent load on its user community

This if not handled well could lead to duplication of effort and fragmentation of tooling.

Questions:

- Is this even a problem? Arguably, sharding (tight coupled) vs plasma (loose coupled) is itself a “duplication of institutions” situation, and we’re not really complaining about it.
- How can this be mitigated at the standards/application layer? Can we try to make “enshrined” and “default layer 2” versions of each protocol maximally similar to each other to reuse code?
- Is there some fancy way to extend the protocol layer to make the layer 2 versions of all of these capabilities more like first-class citizens? That is, can we try to come up with a model where there are simply “objects” (like UTXOs but mutable) and then all of the differences arise in different types of relationships between objects? Would this be better than the status quo? How would the arguments around, for example, DAO soft fork-like censorship resistance play out?