We can calculate the actual resource complexity of the state execution protocol that I describe here. Let us suppose that there are O(c2

) state executors, of which O(c) are assigned to each shard. The executors' deposits could be stored in the shards; alternatively, we could dual-use the validators' deposits that are stored in the main shard, and enforce the deposits through shard-to-main-shard asynchronous communication.

Let us now try to figure out the concrete complexity of a standard client (that is, a full client on one shard and a light client on all other shards). Clearly, if a client were to verify all

of the deposits, it would have O(c2

) complexity, which is unacceptable (unless we only run the state execution protocol something like once a day, which is unacceptable for other reasons). However, what a client can

do is randomly select a sample of deposits from each shard, and just check those deposits. The required sample size to have statistical safety doesn't grow with c, so in big-O notation it's O(1), but concretely it's likely ~100. Hence, normal-case standard client complexity would be O(c).

But what if there is an attack? Let's suppose that we make an honest minority assumption that 10% of the clients in each shard are honest (this can be made very defensible if we do something like reshuffling all state execution nodes between shards every month, though even without that it should work); this implies a 10% threshold for disagreement at which point a client demands an execution proof.

If T is the total quantity of deposits, this implies that an attacker would need to burn $\frac{T}{c} * 0.1$

coins to have a client demand to manually check a single period. Let D be the ratio between the reward for a claim during one period and the full deposit, so $\frac{T}{D}$

is the total reward per period. A zero-net-profit attacker with all of the deposits would be able to attack on average of $\frac{T}{D} \div \frac{T * 0.1}{c}$

$= \frac{c * 10}{D}$

times per period. Each attack requires the client to download a single collation, which takes O(c) effort to verify, so the client's effort from the attack would be $\frac{c^2 * 10}{D}$

. In practice, one of the c

terms comes from the number of shards, and D

is vastly higher than the number of shards, so this could be viewed as O(c) with a very low constant factor.

In practice, this means that an attack could happen, but it would cost a lot and it would only slow clients down by a few minutes until they execute all of the collations that the attacker tries to trick the client about.