Node Operator Scoring

To create a scoring system for Node Operators (NOs), it's critical to first identify what outcome the system will optimize for. At the highest level, we want the "best" set of NO's, such that stakers can earn the greatest APY over an infinite period. Doing so requires that validators generate competitive premiums, while also avoiding loss of ETH.

The Lido scorecard identifies some factors that would allow it to reach these goals:

1. NO's run their own nodes (no white-labeling)
2. Good performance
3. NO's should earn enough to build a profitable, dependable staking business
4. No operator has more than 1% of the total stake
5. Distributed geographically and jurisdictionally
6. Distributed variation of on-premise infra and different cloud providers
7. Best practices in security and key management
8. Client Diversity
9. Node NO's are disincentivized from acting maliciously
10. Lido DAO can't suddenly change the validator set
11. There's a way for permissionless entry to the set

Building a NO Scoring System

Before we look at possible datasets and metrics, it's necessary to first decide on a desired scoring approach. Our approach will influence the usefulness of datasets and our tolerance for their faults.

Many of the Lido scorecard goals can be distilled down to reducing risk while simultaneously achieving "good performance".

Lido has already committed to **expanding diversity** to reduce risk across three dimensions; technical, geographical, and jurisdictional.

Viewing a scoring system through the lens of risk will allow us to mitigate losses and hence earn a respectable APY over a long timeframe. Therefore, it's from a risk

standpoint that we should review possible data for inclusion in our scoring to craft a healthy and resilient validator set.

The decisions that NO's make can influence their rewards, more significantly than I expected. However, the financial penalty of mistakes is so great as to overwhelm many of these optimizations over the long run. We should look to include performance data in scoring, it matters and we cannot expect stakers to have an infinite timeframe. But we must simultaneously ensure that our primary focus is on managing risk.

Therefore, I don't believe it's possible to score NO's without using datasets that examine the largest risks and that will require using off-chain data.

To include these datasets, it will be necessary to operate a less-trustless scoring system. Such data is likely to rely on NO's truthfulness. The tradeoffs are obvious - to score the most impactful factors you introduce the risk of NO's lying.

It will be necessary to create incentives for truthfulness where possible, checks to validate their claims, and a system to investigate red flags and punish those in violation.

In the next section, we will review the existing scoring systems and then examine specific datasets for their quality, characteristics, and overall usability. We'll also consider how we can create new data, beyond what is currently available, to fill gaps in the existing datasets.

Existing Scoring Systems

Rated Network

Rated Network operates one of the most popular scoring systems. The Rated Validator Effectiveness Rating (RAVER) is a "measure of how well a validator has been performing its deterministic duties over time".

It aims to attribute to NO's "only what is under their direct control". It's evident post-merge that the performance of duties correlates only loosely to validator rewards. Hence, they believe that APY is not a sufficient measure of operator performance.

The Rated scoring system:

1. proposer_effectiveness == [non_empty_blocks + empty_blocks * 0.25] / total_proposer_slots_attributed

2. attester_effectiveness == participation_rate * correctness_score / aggregate_inclusion_delay

3. validator_effectiveness == [3/8 * proposer_effectiveness] + [5/8 * attester_effectiveness]

Proposal effectiveness

The calculation for proposer effectiveness in Rated post-merge is:

proposer_effectiveness == [non_empty_blocks + empty_blocks * 0.25] / total_proposer_slots_attributed

This formula scores how many times a validator has successfully proposed a block out of the times that they were awarded proposer duties.

Case	CL	EL	score
A.1	proposed	proposed	1
A.2	proposed	empty	0.25
B	missed	missed	0

For cases where the EL is empty, they score 0.25, rather than the expected 0.5. This harsher penalization is because of their opinion that validators exist to facilitate transactions and therefore an empty EL is a severe failure to contribute to the network.

Attestation effectiveness

attester_effectiveness == participation_rate * correctness_score / aggregate_inclusion_delay

There are now additional penalties for cases when the attestation does not get included in a timely manner; namely within sqrt(EPOCH_LENGTH) or 5 slots. But Rated does not use this when calculating participation. When a validator is more than 5 slots late, but less than 32 slots late, they will not be negatively scored in Rated, however, it would be penalized on the blockchain.

Rated prioritizes generalizability and legibility in their design goals, hence not scoring this lateness. For our scoring system, this is unlikely to be the best approach because it would allow NO's to accumulate penalties and missed rewards without being negatively scored.

This can be moderated by monitoring of inclusion delay, which can be a helpful signal for predicting an NO's propensity for lateness. Rated does not use inclusion delay, instead looking at lateness as binary based on correctness.

However, it's not obvious that the inclusion delay contains zero useful information. If a validator is attesting extremely slowly, but never slow enough to be penalized, this matters to us. This slowness suggests that for a given volatility in their network or hardware performance, they will be more likely to be so delayed as to result in a penalty when compared to other NO's.

While the Rated system exists to track existing performance in a backward-looking fashion, our system must make forward predictions to minimize risk and maximize the quality of the set. Therefore, we must attempt to weigh the risk that something will happen, not just track whether it has in the past.

Due to the rare occurrence of events on the Ethereum blockchain, particularly slashing but also penalties, tracking in a binary fashion is unsuitable for our purposes. We must be more predictive and value magnitude not just frequency.

Validator effectiveness

validator_effectiveness == [3/8 * proposer_effectiveness] + [5/8 * attester_effectiveness]

Currently, Rated applies disproportionate weighting to proposer_effectiveness in relation to the expected long-term reward distribution of ⅛ to ⅞.

They have observed that “execution to consensus layer rewards come at a 1:4 ratio” but expect that the ratio will become “more balanced over time” once we see:

1. More active validators on the Beacon Chain
2. More adoption of MEV Relays and out-of-protocol PBS
3. Greater demand for blockspace

Weighting proposer and attester effectiveness based on existing data rather than the long-run expectation seems smart. If we choose to score based on the long-run expectation we risk lower rewards in the interim. With an unknown period until the long-run average is achieved, it’s necessary to follow Rated and adopt a non-standard weighting.

Currently, more recent data is significantly closer to the long-run expectation so I would expect Rated to adjust their proportions in the future.

Instead of dictating a fixed weighting, we can track the proportion of rewards across the network across some period and apply it on a monthly cadence. This should allow the system to be sufficiently dynamic without requiring expensive tracking of a large number of data points.

Slashing

RAVER currently does not include slashing, but Rated appears to be working to add it. For our purposes slashing occurrence is critical for scoring a NO. Slashing is the most serious offense and not tracking it creates a serious hole in the scoring that would result in highly perverse incentives.

Sync committees

RAVER currently does not factor in whether a validator is part of the sync committee or not. They say that because of the “stochastic nature of sync committee selection and the low probability of a single validator to be selected in a sync committee set, we have opted for not including sync committee performance”.

It’s important to recognize that RAVER operates by scoring each validator separately and then averaging the scores for higher-level aggregations like NO's and deposit addresses.

Not factoring for sync committees makes sense because of the low likelihood of an individual validator being included. When looking at shorter timeframes the sync committee inclusion is too deterministic and would unfairly impact small or solo NO's.

Scoring sync committees differently could also add perverse incentives where NO's want to avoid missing sync committees and therefore defer maintenance or necessary downtime.

This could encourage NO's to take offline a subset of a cluster of validators on a single server to perform some maintenance while trying to keep the validators in the sync committee online. This adds the additional risk of slashing if transferring of keys between servers is required.

Penalties and Missed Rewards

Rated appears to attempt to follow the existing Ethereum spec when calculating their penalties and missed rewards for attestations.

We noted that the RAVER attester effectiveness score doesn't appear to measure correctness and participation exactly as the spec. However, the underlying penalties and missed rewards data available via their API seem to follow the specification precisely.

Calculating missed rewards on the execution layer is much more challenging than for attestation rewards and there are multiple seemingly acceptable approaches.

Rated looks at four approaches, and will be offering the first two in their API:

Approach 1: Simple average of block value in an epoch

EL rewards can be averaged for a period by dividing the sum of rewards per block by the number of proposed blocks in the period.

Rated notes that this approach is simple to understand, calculable with only on-chain data, and replicable. But it flattens the impact of MEV. One operator running mev-boost would earn significantly more for the same blocks, on average, compared to producing vanilla blocks. Taking a simple average will falsely reduce the missed rewards of those using mev-boost and increase the estimated missed rewards of those not.

Lastly, block rewards have a high variance in reality. If the missed block was wildly profitable, the truly missed rewards were much larger than the average, and vice versa.

Approach 2: Referencing relay bids for the opportunity cost

You can call various relay APIs for a specific block to understand what rewards were possible, then take an average.

Rated notes that they have “found that in the majority of cases winning_bid ≠ max_bid”, and therefore opt to use the average of all builder bids, rather than solely the max bid.

It’s likely that “bids keep arriving after the winning_bid gets picked; naturally these bids pack more transactions” which explains why the max bid is often higher than the winning bids.

This approach is perhaps the closest approximation of missed rewards because it references the market clearing price at the time that block was proposed. The problem with taking an average of all bids is that you will likely approximate the median, given the expected linear increase in bids. Meanwhile, as I’ll show later, most NO’s select a bid before the slot has started. So, their missed rewards are expected to be lower than what this metric would suggest.

Lastly, using relay data will unfairly punish those NO’s not using mev-boost blocks as frequently and rely on the availability of data not on-chain.

Approach 3: Referencing the next block produced

“The validator who missed the proposal would have had access to the same transactions as the next proposer and could then have at least built the same block with the same transactions and corresponding fees.”

The problem with this approach is that the next block can include transactions that did not exist at the time that the initial validator would have proposed a block. Therefore, this can, and likely will, increase the value of the next block produced because the greater number of transactions potentially allows for more MEV extraction.

Approach 4: Abstracting to the operator level

When aggregating at the operator level we can use information about their ratio of mev-boosted and vanilla blocks to get a more precise approximation of their missed rewards.

This approach could allow us to more accurately predict the missed rewards for a specific operator. However, the approach Rated suggests still relies on the average rewards for vanilla blocks and an average of relay bids, multiplied by the ratio of each type.

In reality, NO's are likely only opting for vanilla blocks when the bids are below their min-bid threshold or when the mev-boost API fails to reply in time for the proposal.

Therefore, using only the average of vanilla blocks doesn't give a particularly precise estimate of missed rewards, because it doesn't know how far below min-bid the rewards were. Or, if the API simply didn't respond.

Lastly, it doesn't attempt to decipher if the specific missed block would have been proposed using a relay or as a vanilla block.

Scoring Missed EL Rewards Accurately

Instead, I would propose that we try to encapsulate more information when estimating missed rewards. While Rated has the hard task of trying to generalize their scoring across all validators, we are fortunate to have the much easier task of only scoring a known group of NO's, which we can communicate with.

Each operator can, and should, make public what min-bid threshold they set. Lido DAO has already soft-enforced a list of "must-include some" and "may include" relays, so we could rely on these for getting the builder's bids. Specifically, NO's do not have to use every relay in that list they must use at least one. However, because it's impossible to know ahead of time which they will use, we will need to include all of them.

We can take all of the bids that existed on these relays for the block and choose the highest available bid at the average timestamp when NO's select a bid.

If this bid is above the min-bid threshold for the operator, we will presume that this is the reward they have missed. If it's below their min-bid threshold, then we must presume that they would have created a vanilla block.

The value of a vanilla block can be calculated by taking the EL rewards earned per vanilla block in a TBD period and dividing it by the number of vanilla blocks.

Theoretically, this should deliver a more accurate representation of the missed rewards.

The downsides to this approach are:

1. It's heavy to calculate because it relies on pulling data from each missed block, and from many different relay APIs, rather than using aggregates for an epoch.

2. The number of vanilla blocks is relatively low and so there is variability in EL rewards for these blocks. However, the variability is low enough to be preferable to taking an average of all blocks, which would include mev-boosted blocks.

3. We rely on NO's truthfully updating their min-bid in our system whenever they change it locally. This adds extra complexity to the creation of the system.

If this approach is undesirable due to complexity, I would recommend opting to use approach two and referencing relay bids. The downside is that you unfairly punish those not using mev-boost and those with higher min-bid thresholds, which risks adding censorship to the network.

Aggregation

Lastly, we'll look at how Rated aggregates across groups of validators and periods. Surprisingly, the different approaches to aggregation can meaningfully impact scores.

For groups of validators i.e. aggregation by operator or deposit address, Rated uses a validator-up approach. They simply take an average of each metric score in the group of validators and attribute that to the operator.

While Rated does not use slashing in their RAVER scoring, it's likely not sufficient to treat rare events like slashing and perhaps even missed blocks as averages. Taking an

average will compress the dataset, making an operator who has a few validators slashed score very similar to those who have had no slashing events.

For rarer events, it's likely better to use a binning approach, such that even having only 0.01% of validators slashed would give an operator a meaningfully poorer score than those without slashes.

When aggregating by period Rated takes an average of daily averages. For example, when looking at 30-day inclusion-delay they would calculate the average inclusion-delay for each 24-hour daily period. Then, they take an average of those 30 data points. This calculation is different from taking the inclusion-delay of each attestation over the 30 days and calculating the average.

The former approach, which Rated uses, is superior. It reduces the amount of calculation required because past days' data can be aggregated and stored to a daily group, rather than requiring us to iterate across the data each time the calculation is run.

More importantly, it prevents changes in the number of validators in a group from impacting the result.

For example, if you have 10 validators for the first 15 days, you have 225 * 15 * 10 ~ 33,750 attestations.

Then, if you have 20 validators for the next 15 days, you have 225 * 15 * 20 ~ 67,500 attestations.

If you average across all attestations then the last 15-days will represent ⅔ of the result, making the first 15-days of data less impactful. Whereas if you aggregate by day first, each day in the 30 days of data is treated equally.

The positive to this Rated approach is that it prevents sudden changes in the number of validators from skewing the data such that it's unreliable. The downside is the inverse, which is that this makes the data less responsive. For example, in the case where an operator spins up many new validators on the same hardware, this could meaningfully decrease performance. Yet, the data will not reflect this as quickly because we're aggregating first by day.

Overall, I think the tradeoffs are worth it because it's unlikely that the number of validators an operator has will change so rapidly as to cause meaningful issues in the data.

Observatory Zone Scoring

(https://observatory.zone/cosmos-hub/validators)

Observatory Zone uses a radically different scoring approach. While Observatory Zone has very low sensor participation to report data, the logic of the whitepaper (avoid downloading, as I'm currently getting anti-virus reports saying the PDF contains a trojan… could be a false report but be cautious) is still interesting.

Observatory Zone does not attempt to score individual validators and rank them. Instead, its goal is to score blockchains based on their decentralization, performance, and governance.

Their system is very simple so we will only discuss it briefly and note any interesting data points that we might try to replicate.

Outside of on-chain performance data, they rely heavily on scoring decentralization. To do so, they rely on the number of validators, and geographic and ISP-level concentration data. When calculating ISP and geographic concentration they calculate at the node level, without attempting to try to estimate the total number of validators that those nodes are responsible for.

Therefore, the data is less useful for understanding concentration. While it's helpful to know the concentration of nodes among geographies, to gauge the risk of non-finalization or even downtime leading to penalties and missed rewards, you want to know the concentration of validators not just nodes.

Using this node-level concentration data they apply a fixed weight to decide on a level of decentralization for a chain. While this binary approach is acceptable, a granular binning approach with an upper limit might be better. Using binning would allow us to negatively score validators located in countries that are under our maximum threshold, but still overly concentrated, rather than being binary.

Using this network-level data would be helpful for our scoring. For example, just because a relatively large percentage of the Lido protocol set of validators is located in country X is not reason enough to be concerned. We must also factor in the global network concentration as well, and only with both data points can we accurately gauge the risk. Over an infinite number of games, Lido protocol must contribute positively to the overall health of the network, as well as minimize its own internal concentration risk.

Observatory Zone no longer scores based on software versions, but this was a feature in their whitepaper. They would monitor the latest versions of software, particularly clients, and then penalize based on the magnitude of the version “delay”.

The problem with version monitoring is that by punishing those who are using “outdated” versions you increase concentration risk. The upside is that in cases where the outdated client has a bug, the scoring system should push NO's to move to the newer version. In the inverse case, where the new version has a bug, the problem is only magnified by additional usage. It makes no sense to punish those with deliberate upgrade strategies that involve waiting until versions are battle tested.

Presumably, NO's are incentivized to move away from faulty outdated versions because they want to avoid loss of ETH. Therefore, it’s not necessary to use version tracking because the same incentive can be created only by scoring based on rewards, penalties, and slashing.

Overall, the approach taken by Observatory Zone is not highly useful to our problem. One takeaway should be that there can be a benefit to tracking network-level data in aggregate, which we also see from Rated Network.

Minimizing Risk

The Lido scorecard identifies the following risk factors; white-labeling, operator concentration, geographic concentration, infrastructure concentration, security and key risk, client concentration, malicious intent, and governance risk.

Other risks exist beyond those mentioned, but these cover the bulk of the present risk for stakers.

Internal Processes

Internal processes would include security, key management, change management, user access management, failover/failback processes, and access policies.

To date, most slashing has been the result of poor processes, particularly for failover and failback scenarios.

Failover can maintain higher uptime - a critical factor in the rewards that a validator earns. But failovers without sufficient failback processes can lead to slashing, as we've seen in the **case of RockLogic**. Validators were taken offline through a failover process and restored elsewhere. Later, they attempted to failback to their original server, where they believed all keys had been cleared. However, the keys were unexpectedly re-imported, leading to slashing offenses. This event exemplifies the importance of correct failover and failback processes.

The ETH lost to slashing was orders of magnitude more than would have been lost to any downtime from waiting for a bug fix to be implemented. Successful failover and subsequent failback is preferable to downtime, but an unsuccessful failover and failback. A preference for safety over liveness will ensure greater performance over time.

Significant diligence needs to be paid to the internal processes of NO's to reduce risk at the Lido set level.

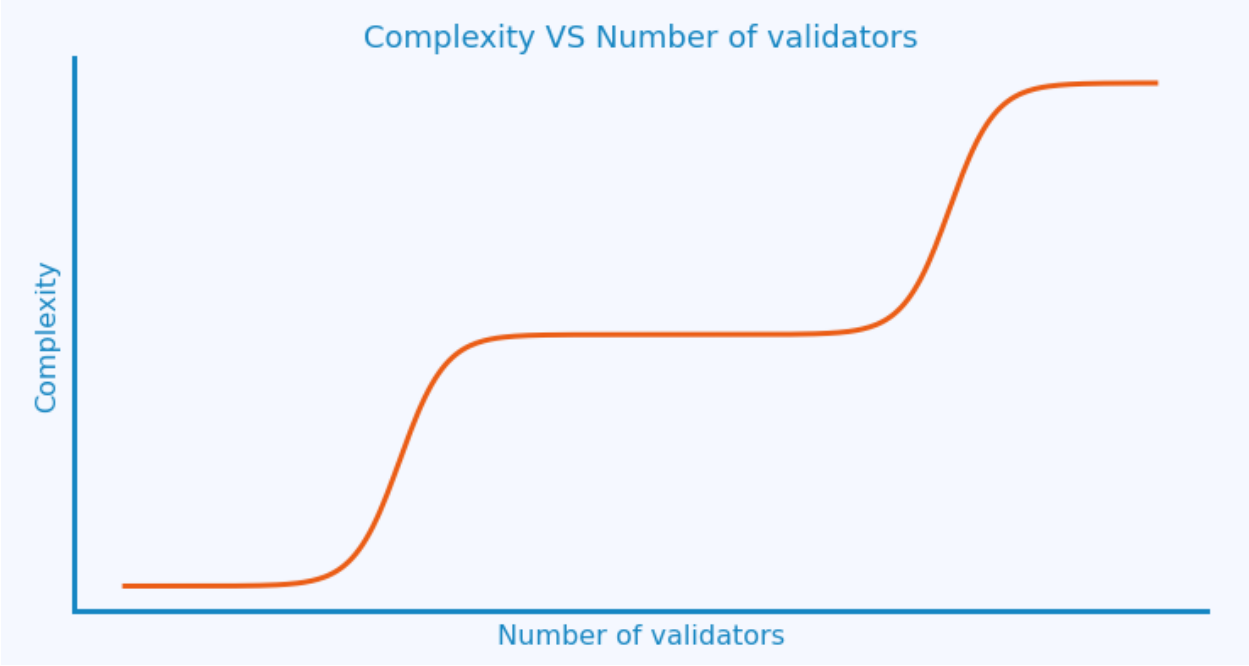
Hardware

The diversity in individuals staking is critical to the security of the network. The success and fairly widespread nature of home staking (sometimes referred to as solo staking) proves that individuals can run validators on local hardware, or cloud servers.

I looked at 7 months of historic validator performance data from Rated, including only validators that had data for the full period. The data shows that deposit addresses with only 1 validator have marginally better median uptime and correctness than Lido NO's. You could conclude from this that the hardware and infrastructure of an operator is irrelevant to core metrics that underlie performance.

However, according to interviews with large NO's, as the number of validators that an operator runs increases the complexities increase superlinearly, to a point. One cannot presume to run 1000 validators by simply replicating their hardware 1000 times. The additional scale adds problems around security, maintenance, downtime recovery, and management. Beyond this, at scale risk management requires diversification, adding further complexity when multiple clients may be run simultaneously.

According to NO's, network bandwidth can become a serious bottleneck as the number of validators increases. This bottleneck could explain some differences in performance when compared against solo stakers.



The above graph is a simplistic representation of the curve that can be expected between scale and complexity. No values are displayed, as I only intend to show the curve, with the precise number of validators where the step function occurs is dependent on the NO's decisions.

At the scale of NO’s insufficient hardware will cause higher downtime, more missed attestations and block proposals, and therefore lower earnings.

When examining the NO’s data there is a relatively large dispersion in these performance metrics due to NO’s different infrastructure, which we will discuss later.

Hosting vs On-Premises

Estimates show that between **57%** and **58%** of validators are connected to nodes run on hosting, cloud, or otherwise. This concentration presents a systemic risk of mass downtime and the risk of slashing from a malicious actor.

Within the Lido protocol, a **less severe concentration** exists, with 48.3% of validators being connected to nodes run on shared hosting.

Much of the world's hosting capability falls under the legal jurisdiction of the United States, creating a single point of failure. Similarly, there is a meaningful concentration of physical server farms in a handful of cities around the globe.

Lastly, greater security risks can exist on shared servers. With the largest Lido NO's being responsible for hundreds of millions of dollars worth of ETH, there is an enormous incentive for bad actors in hosting companies to take malicious action. While it would be incredibly hard to execute, data breaches occur at major providers. This risk is particularly present when an external signer is not used.

Jurisdictional and geographical concentration

Much of the geographic risk to Ethereum exists because of the concentrated server locations of shared servers, and the legal jurisdictions of hosts. However, it's also important to consider the same concentration among NO's, regardless of whether they use shared hosting or on-premises servers.

Across the Ethereum network, between **33%** and **41%** of validators are estimated to be physically located on servers in the United States, with 14% to 19% in Germany. Given that ~40-50% of validators in each country are in Ashburn and Frankfurt respectively, it's highly likely that these two cities represent the bulk of AWS exposure globally and Hetzner's USA exposure.

Data identifying the number of nodes for each hosting provider varies widely so it's not possible for us to reliably use this granularity of data for scoring. Notably, there is enormous variance in the estimations of AWS and Hetzner concentration.

Currently, **data is self-reported by NO's** detailing their server locations and jurisdictional dispersion. In the available data, primary server locations are significantly more diversified than the global average, with only 16% of primary servers located in the USA, but 15% in Germany.

Jurisdictionally, NO's are somewhat concentrated, with ~58% exposure to Europe, ~28% in the Americas, with the remaining 14% across APAC. Jurisdictional risks exist when they lead to mass downtime across the network, with the possibility of an inactivity leak.

For this risk to be managed NO's must be diversified to the network-wide concentration. However, with no reliable data on the network-wide jurisdictions of validators, we must instead be particularly cautious and ensure a greater diversification of jurisdictions among NO's. Therefore, it's sufficient to use only internal NO jurisdiction data for scoring.

Operator concentration

Currently, there is a soft cap such that a NO should be limited to a 1% allocation (via the Lido protocol) of the total ETH in the Lido protocol. In my opinion, a limit on NO concentration should exist to reduce the financial impact on stakers in the case of an event and to reduce the control a single entity has over the network.

We know that the magnitude of the financial impact under negative scenarios scales with the number of validators impacted; because slashing has a correlation penalty and inactivity leaks only occur during non-finality (when more than a third of the network is offline). Similarly, an entity's impact on the network is not dependent only on the number of validators they operate using the Lido protocol, but the total number they operate.

For a NO concentration limit to "reduce the financial impact" and "control a single entity has" we must look at the total validators a NO controls, not their percentage of Lido's ETH.

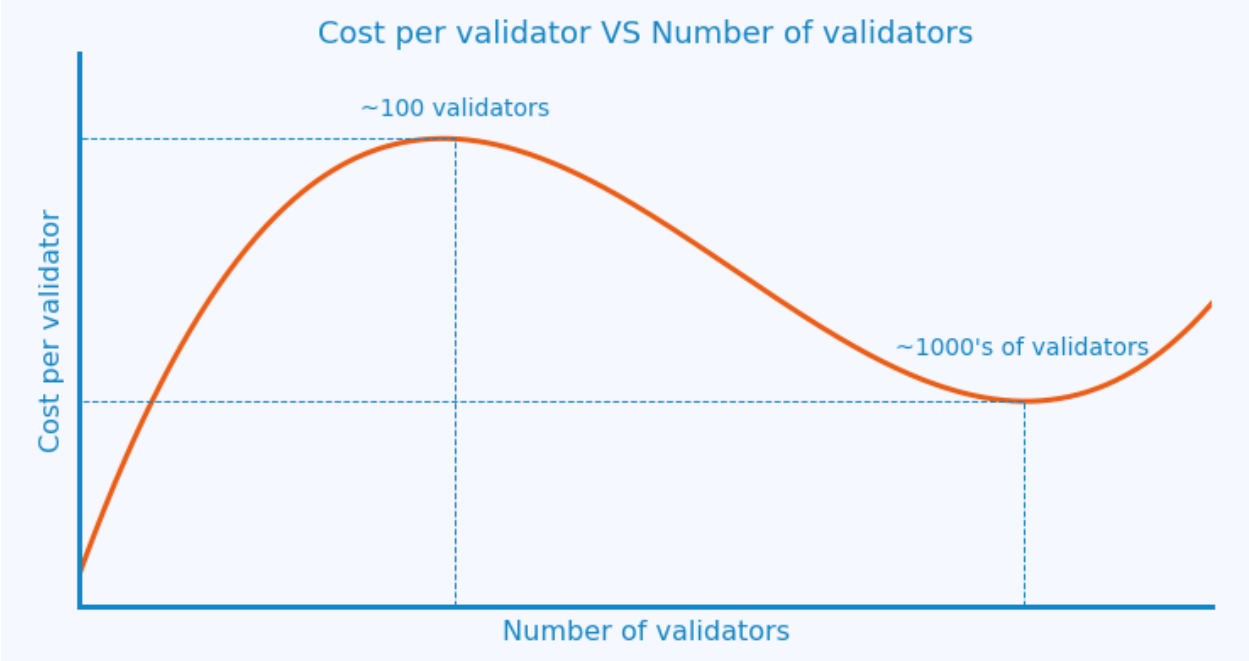
If the current 1% cap is included in a scoring system as a hard limit it would necessarily override the other scoring criteria. The result is to attempt to reduce internal NO concentration, but without considering the impact that this has on other risks. Hence, to utilize a scoring system for ETH allocation it will be necessary to adjust the limit.

To manage the risks of financial impact and network control we must factor in the total number of validators a NO controls. However, this could be through a scoring system that punishes an entity's network concentration, rather than a low discrete limit on the percentage of Lido ETH a NO controls. The level of concentration at which it would impact a score is highly subjective and should be decided by the DAO. But the critical difference is that this level is relative to the total number of validators a NO controls across the network.

Both the concentration of an operator network-wide and their concentration of Lido stake are important. Considering both metrics is critical for accurate risk scoring.

Ensuring a Profitable Business Opportunity

NO's face high sunk costs when starting their business, particularly if they choose to run on-premises. Even those hosting will need to invest considerable resources in internal software for management, security audits, and staffing. This significant upfront investment makes it challenging for NO's to be immediately profitable with a relatively low number of validators.



The above graph is a simplistic interpretation of the NO's unit cost, where running a small number of validators is cheap per unit, then requires high fixed and sunk cost investments, before becoming highly profitable per unit at scale. This is based on interviews with NO's, who maintained that cost decreases at scale, but operations at a very large scale can start to become more costly again.

Therefore, it's expected that new NO's will require a high minimum number of validators if they are to operate similar infrastructure to existing NO's.

These economics dictate that the percentage of monthly revenue remaining to invest in improvements is correlated to the NO's number of validators.

We've already established that running high-quality infrastructure is expensive by global standards. Currently, NO's are well-funded entities that can afford the sunk costs to achieve high-quality infrastructure from the start. As the NO set increases we can expect that the resources of new NO's will reduce, particularly with significant additional size. Over the long run, NO's should act as rational profit maximizers. So, NO's with fewer validators are likely to run lower-quality setups. Similarly, as an operator loses validators there should be a superlinear degradation in their setup quality to maintain profitability.

Hence, a future scoring system should consider the economics of NO's and ensure that the distribution and velocity of changes in the distribution do not negatively impact the infrastructure of the set.

Performance and Rewards

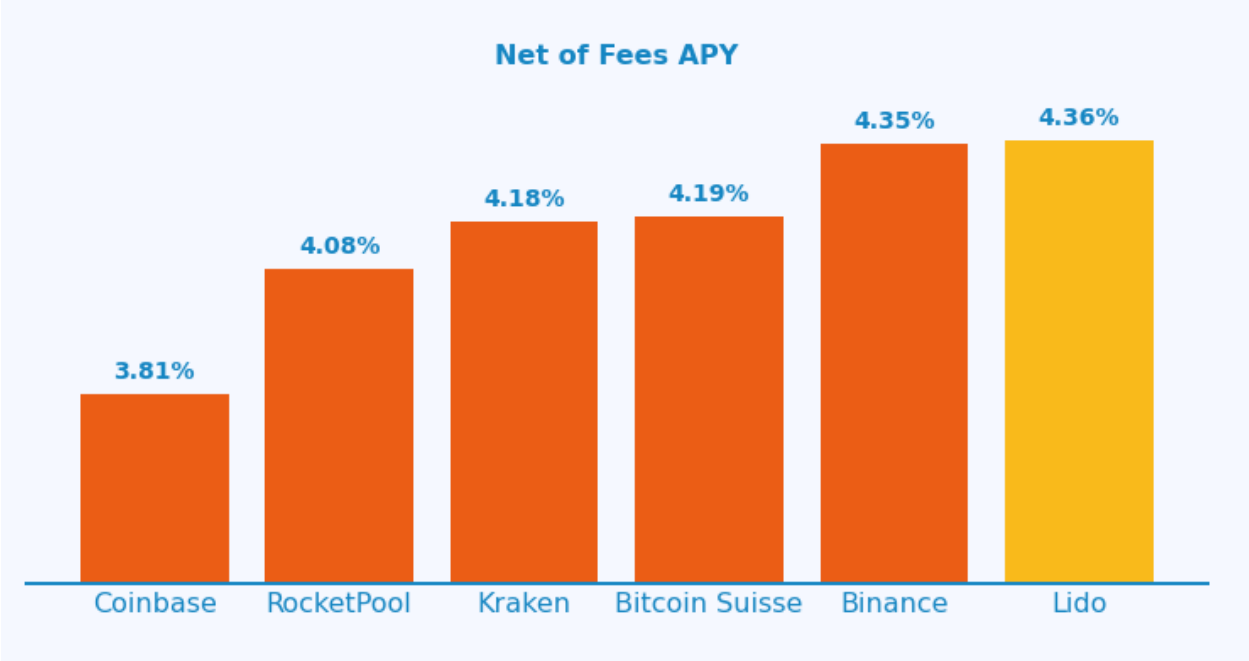
Stakers decisions are influenced by the APR that they will earn but what's not clear is the exact curve that explains the relationship between APR and Lido stake.

Lido offers a unique advantage through the utility of stETH. While centralized competitors typically do not offer a liquid token and DeFi competitors liquid tokens have orders of magnitude lower utility due to less inclusion in DeFi protocols. When we consider how users are using stETH, we can see that the majority do not actively use their stETH. Among the minority that is they are overwhelmingly using it to stake in lending protocols, earning additional rewards for providing liquidity. These actions suggest that for a subset of Lido stakers the total APR that they can earn is a driving factor, given their decision to take on incremental risk to earn higher net rewards.

For a different subset of global ETH stakers they choose CEX's, where they are earning significantly lower rewards than with Lido, regardless of whether they subsequently re-stake stETH. Clearly, for these users, APR is not the driving factor in their decision.

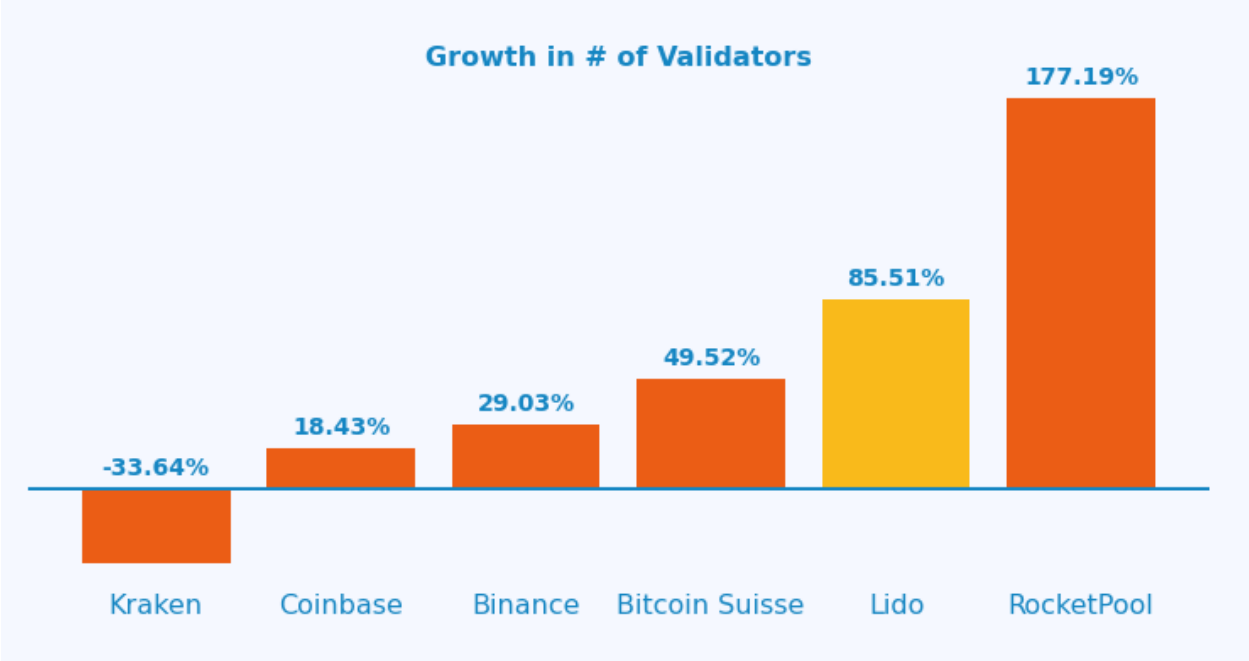
Having spoken to many of these users and their providers, I believe that their driving factor is compliance. Much of the ETH staked with CEX's is from institutions whose compliance teams may have only approved specific counterparties, often with the requirement of no co-mingling of funds with non-KYC'd individuals.

Therefore, my understanding is that much of this ETH would be unable to be staked in Lido currently and so it's not as relevant to our analysis as protocol competitors. Here, we want to look primarily at how APY drives decision-making among those who could use Lido.



According to data from Rated for the latest 90 days, Lido has the highest after-fee APY among the largest six pools.

In contrast, when we look at the growth of each provider we see this over the past 365 days of data:



This graph looks at the relative (current validators / previous validators) growth in the number of validators by pool.

Clearly, despite RocketPool having the lowest APY after fees among the pools, it's had the highest relative percentage growth.

Using only percentage growth in validators doesn't tell the full story. In these 365 days, RocketPool had a pre-existing queue which they were able to begin to stake. They also transitioned to allow 8 ETH bonded pools, significantly increasing their capacity.

When we look at the absolute growth in the number of validators over the period we can see that Lido has **added more validators than the current size of any of the other pools**.

Lido is growing significantly quicker than any other single pool and this may be influenced by the APY that stakers can earn. The correlation between these factors is weakly positive, but I did not include the rewards that would be earned if you re-utilized your stETH in another DeFi protocol. When considering this, the correlation is much stronger, as would be expected.

A plausible explanation is that for those users for whom APR is the driving factor, Lido is the clear choice because of the far superior potential net earnings. This alone could be responsible for much of the growth, though it's hard to isolate any specific factor.

The takeaway from this data is that APY plays an important role in Lido's continued growth, but it's not the only factor. Given Lido's position as the highest earning provider, additional increases in APY are likely to impact growth relatively less than for competitors. However, we can presume there is some relationship between the two whereby at a low enough APY Lido would lose users, and at a high enough APY growth could be even greater.

When thinking about pursuing additional APY we must weigh the risk against the opportunity cost and this will depend on how the extra APY is intended to be earned. This chasing of **attestation performance at the expense of reliability** resulted in a significant slashing event for Staked.us.

If implemented as a metric in a scoring system we must think about how APY could encourage NO's to cut corners in pursuit of higher earnings. Yet, we must balance this against the knowledge that it's a key factor for users. Later, we'll discuss whether underlying performance metrics are a sufficient proxy for APY or whether we should include it in scoring.

'On-Chain' Performance Data

The following NO on-chain performance data is entirely sourced from Rated Network, to ensure consistency when comparing metrics for usage. The data looks at 7 months of results. We also consider the latest 50,400 blocks when looking at Relay data and dig deeper into the payloads from the Flashbots relay.

Attestations

Attestations include three different "votes" - the source, head, and target of the chain. In terms of importance, the source is most important, followed by the target and then the head.

Rewards:

Timeliness	1 slot	<= 5 slots	<= 32 slots	> 32 Slots (missing)
Wrong source	0	0	0	0
Correct source	W_s	W_s	0	0
Correct source and target	$W_s + W_t$	$W_s + W_t$	W_t	0
Correct source, target and head	$W_s + W_t + W_h$	$W_s + W_t$	W_t	0

Penalties:

Timeliness	1 slot	<= 5 slots	<= 32 slots	> 32 Slots (missing)
Wrong source	$-W_s - W_t$	$-W_s - W_t$	$-W_s - W_t$	$-W_s - W_t$
Correct source only	$W_s - W_t$	$W_s - W_t$	$-W_s - W_t$	$-W_s - W_t$
Correct source and target only	$W_s + W_t$	$W_s + W_t$	$-W_s + W_t$	$-W_s - W_t$
Correct source, target and head	$W_s + W_t + W_h$	$W_s + W_t$	$-W_s + W_t$	$-W_s - W_t$

For more intuition, we can put in the numbers, $W_s = 14$, $W_t = 26$, $W_h = 14$, and normalise with $W_\Sigma = 64$:

Timeliness	1 slot	<= 5 slots	<= 32 slots	> 32 Slots (missing)
Wrong source	-0.625	-0.625	-0.625	-0.625
Correct source only	-0.188	-0.188	-0.625	-0.625
Correct source and target only	$+0.625$	$+0.625$	$+0.188$	-0.625
Correct source, target and head	$+0.844$	$+0.625$	$+0.188$	-0.625

Without a correct source vote, you incur the maximum penalty and the greatest missed rewards. While with only a correct source, but an incorrect target, you will still earn a net positive earnings after the penalty. Lastly, with a correct source and target, but an incorrect head, you will receive no penalty and miss out on a small amount of rewards.

Given that each “vote” relies on the previous “stage” to be correct, we can say that the source vote is the most important.

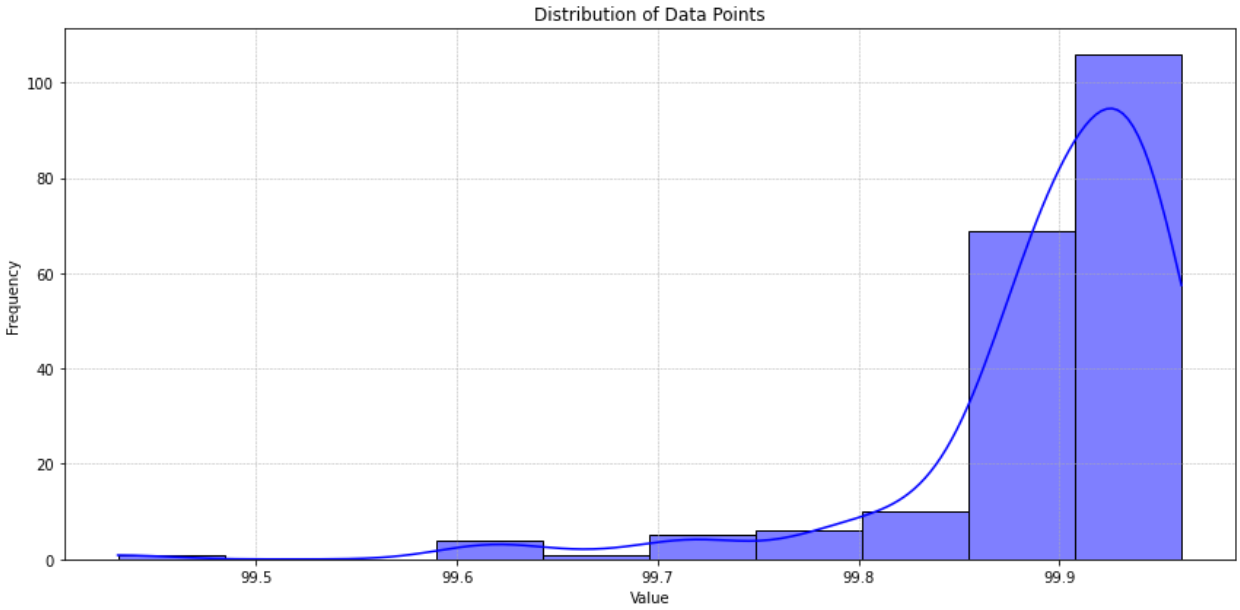
Also, we know that the head vote is significantly less important than the target, with the target being worth 3.7x more in net earnings (missed rewards and penalties) for correctness relative to the head vote.

Therefore, I don't believe that it's sufficient to only measure average correctness across votes when it's evident that correctness in one vote is more important than in another. Instead, we will need to separate them into individual metrics and score each differently.

Again, it could be argued that this data is accounted for in the average rewards, missed rewards, and penalties data. However, these metrics are directly influenced by uptime. Uptime is critical, but using only "higher level" metrics consistently throughout the scoring will result in uptime accidentally being a dominant factor because it would influence all of these higher-level data points directly. The same is true for other 'hidden' factors that are present throughout many high-level metrics.

Average source correctness

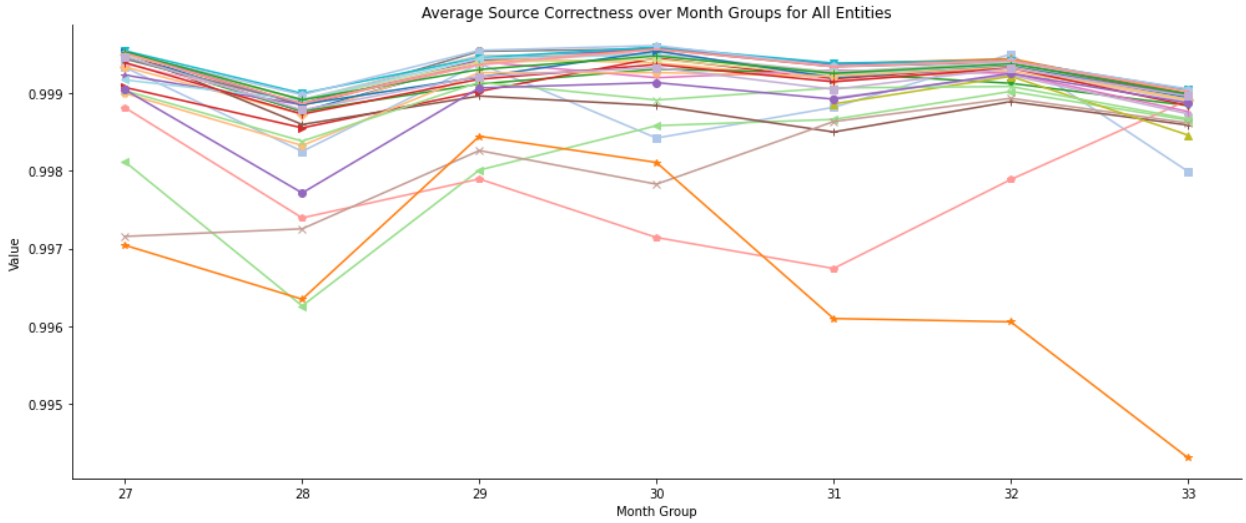
Average source correctness is calculated as the number of correct source votes made divided by the number of votes made by the operator in the period.



Variance & Standard Deviation: The standard deviation value we calculated for the entire dataset is approximately 0.0721%. Such a low standard deviation suggests that average source correctness doesn't have a wide variation.

Skewness: The skewness value is −2.94, which is notably negative. This indicates a pronounced left-skewed distribution, which we can see in the graph (most data is on the

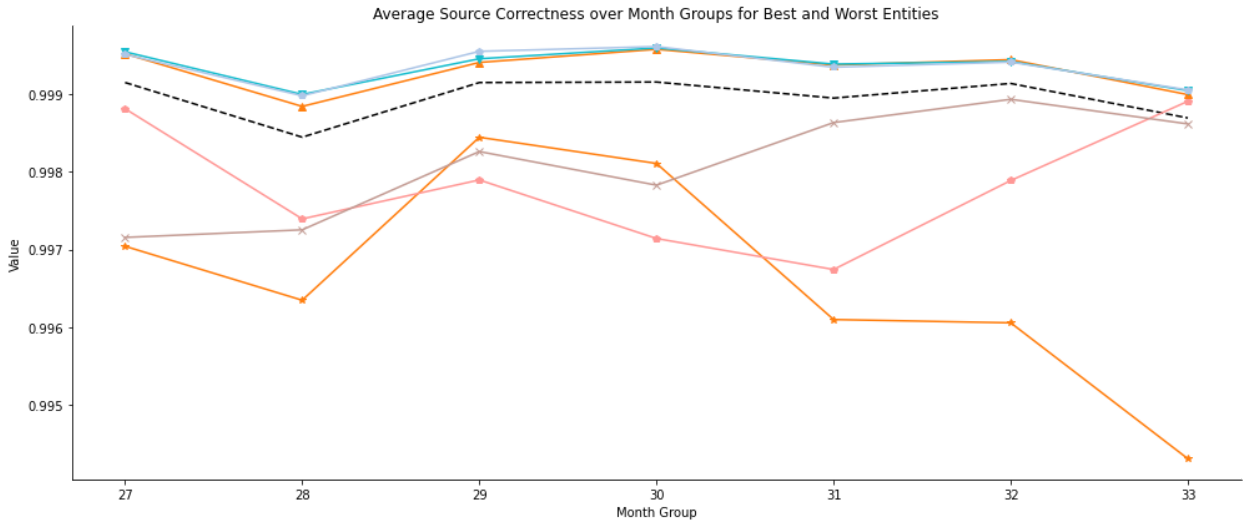
right). This suggests that while the majority of scores are close to the upper limit (close to 100% correct), there are a few entities with significantly lower scores in a given month. These underperforming entities can be seen as serious exceptions rather than the norm, given the pronounced skewness.



When we look at each operator, grouped by backward-looking 30-day ‘months’, we can see a significant disparity between the bulk of NO's around the mean, and those performing worse.

Most importantly we should note that the NO's that are far below the average tend to always be below the average, rather than fluctuating above and below. Therefore, we can have a much higher degree of confidence when choosing to score NO's performing poorly.

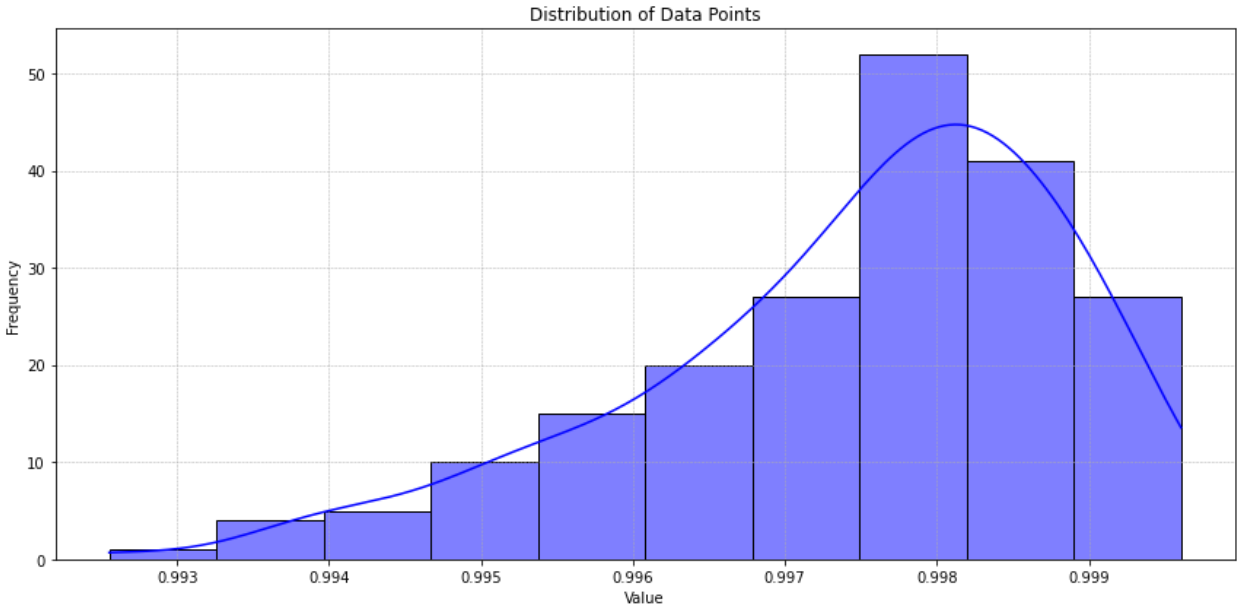
Given the strong concentration around the mean, it’s not suitable to score NO's against a curve because we want to avoid punishing NO's whose recent performance is only a little below the mean. Again, the easiest approach here would be to use binning and to negatively score only the lowest-scoring bins. Alternatively, we could look to mark a floating percentage below the average based on the level of variance in the previous month.



When we plot only the best and worst three NO's against the average it becomes clear that the difference is very notable and fairly consistent over multiple months. Specifically, we see that the best NO's are highly consistent with less variance month-to-month compared to the lower-performing NO's.

Average target correctness

Average target correctness has quite a different distribution to average source correction.



We see a much wider distribution of values, with higher standard deviation. Of course, the STD is still quite small, 0.14% around the mean of 99.74%.

Variance & Standard Deviation: Given the scale of the values in the dataset (which are close to 1), this standard deviation is relatively low, though slightly higher than the previous dataset.

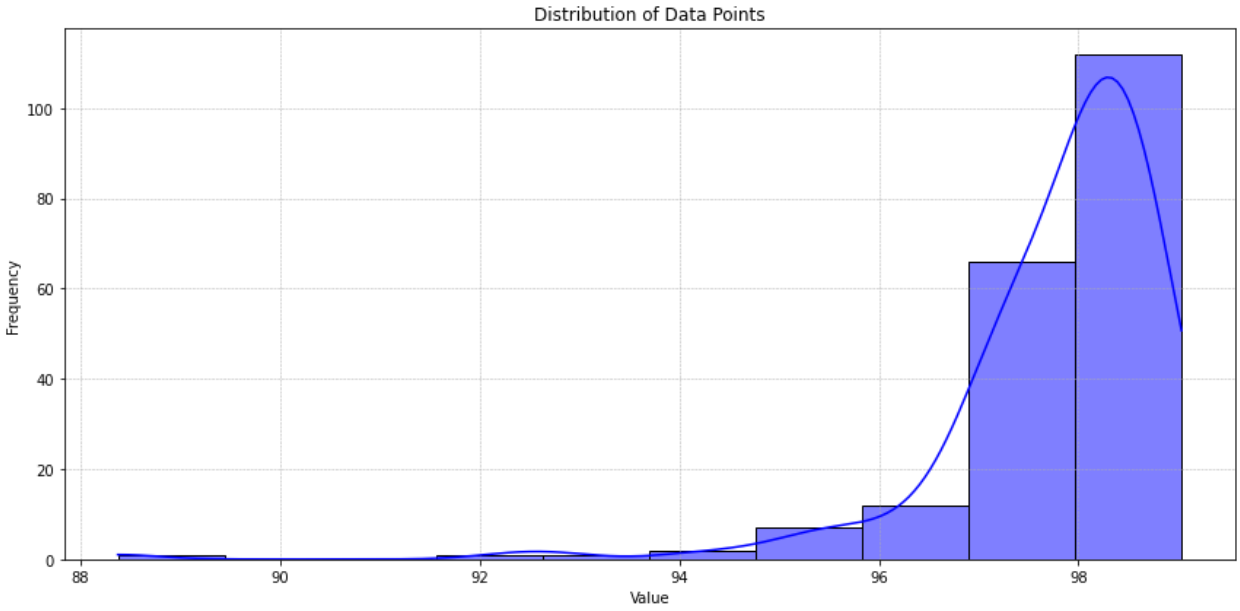
Implication: A low standard deviation suggests that the average target correctness for the NO's are fairly consistent. There isn't a wide variation in scores, meaning that most NO's have average target correctness that are close to each other. However, the slightly higher value compared to the source correctness indicates a bit more variability among NO's.

Skewness: The skewness is −0.8999, which is negative, indicating a left-skewed distribution.

Implication: Similar to the source correctness, this left skewness suggests that while many entities have scores that are close to the upper limit (close to 1), there are a few NO's with significantly lower scores. These underperformers can be seen as exceptions rather than the norm.

In summary, average target correctness exhibits characteristics similar to the source correctness dataset, with a bit more variability in performance. Most NO's have closely packed results around the upper end, but there are a few NO's that are notably underperforming.

Average head correctness



Variance & Standard Deviation: The standard deviation is approximately 1.208%. Given the scale of the values in the dataset (which are close to 1), this standard deviation indicates a moderate level of variability in the scores.

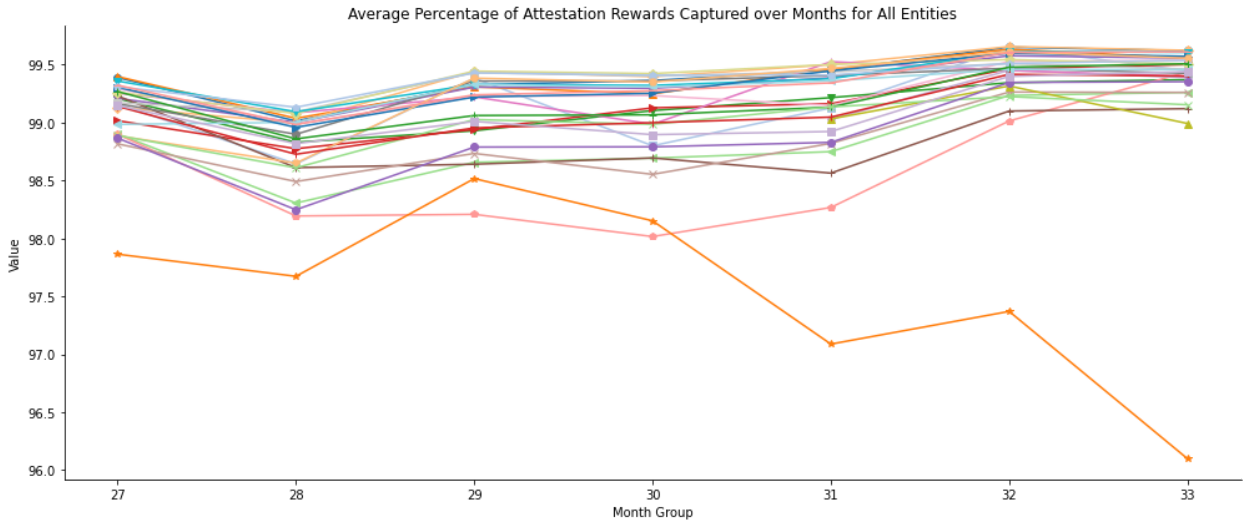
Implication: A moderate standard deviation suggests that there's some variation among NO's. This means that while many entities might have scores close to each other, there are entities that diverge from the average, both on the higher and lower ends.

Skewness: The skewness is −3.5637, which is notably negative, indicating a pronounced left-skewed distribution. This pronounced left skewness suggests that many NO's have scores that are close to the upper limit (close to 1), but there are some significantly lower scores.

In summary, head correctness shows a broader range compared to the target and source correctness datasets. The pronounced left skewness and high kurtosis hint at a few NO's that might be outliers in terms of their performance. These outliers, especially the ones on the lower end, can be of particular interest as they diverge significantly from the majority.

How attestation correctness impacts earnings

We can see that some patterns emerge from looking at the correctness of NO's by each of the votes. However, for this to be a useful metric to score NO's with it needs to tell a story about an operator being more risky to the set, or lowering the earnings.



What we can see from the data is that there is one clear outlier, with another operator frequently below the mean, but returning to a value close to the mean in the most recent month.

The standard deviation is approximately 0.36%. So, this lowest-scoring operator with a mean rewards capture of 97.54% is ~4.41 standard deviations below the NO mean of 99.13%.

While it's certainly possible to rank NO's based on this data, we need to think about what the variance between NO's attestation rewards captured means.

In the very worst case, we're looking at capturing 96% of the rewards. If we just incorrectly presume that the attestation rewards are the only rewards possible, you can think about capturing 96% of the ~4.5% APY on Ethereum validators currently. So, instead of earning 4.5%, you'd earn 4.32%. This calculation does not consider the impact of penalties, which would lower the APY, but not drastically. Some difference exists here, but it's relatively small and less than the difference in after-fee earnings between Lido and Coinbase.

The standard deviation is approximately 0.4574%. In APY terms, being one standard deviation worse than the mean operator would mean earning ~ 4.48% instead of 4.5% APY.

We might consider whether the correctness of votes can tell us something about risk.

Vote correctness and risk

To assess the risk that may be represented by lower correctness, we need to first think about how a validator will typically get votes incorrect or delayed.

The most **common reason for an incorrect source vote** is "downtime in bandwidth or other networking issues".

Incorrect target votes, with a correct source vote, are typically the result of delay. Either in the form of network delay, whereby the block was proposed late and so broadly validators are incorrectly voting for the target, or because of our peering issues. With some blocks arriving later than others, the difference in an NO's bandwidth can make the difference between a correct and an incorrect target vote.

Lastly, the head vote is far more frequently wrong than the target or source. Typically, this is because of the same delayed block issues, particularly when attesting in the first slot of an epoch.

Sometimes it's extremely hard to be correct. When a block is extremely late, but not so late as to be missed by the next proposer, you'll see a lower level of correctness across the network. However, it's evident by persistent differences in correctness among NO's that much of the variance is in fact in the control of the operator.

That variance seems to largely be a result of differences in setup that cause some NO's to more reliably turn relatively delayed blocks into correct votes, while others vote incorrectly.

For example, an operator with a higher bandwidth and a greater monthly data transfer limit might choose to connect to more peers per validator. As a result, they might get these relatively delayed blocks marginally faster than another operator, allowing them to make a correct vote rather than one that's incorrect.

Or, an operator using slower hardware might find that even though they receive at the same time, they are unable to turn around a vote quickly enough.

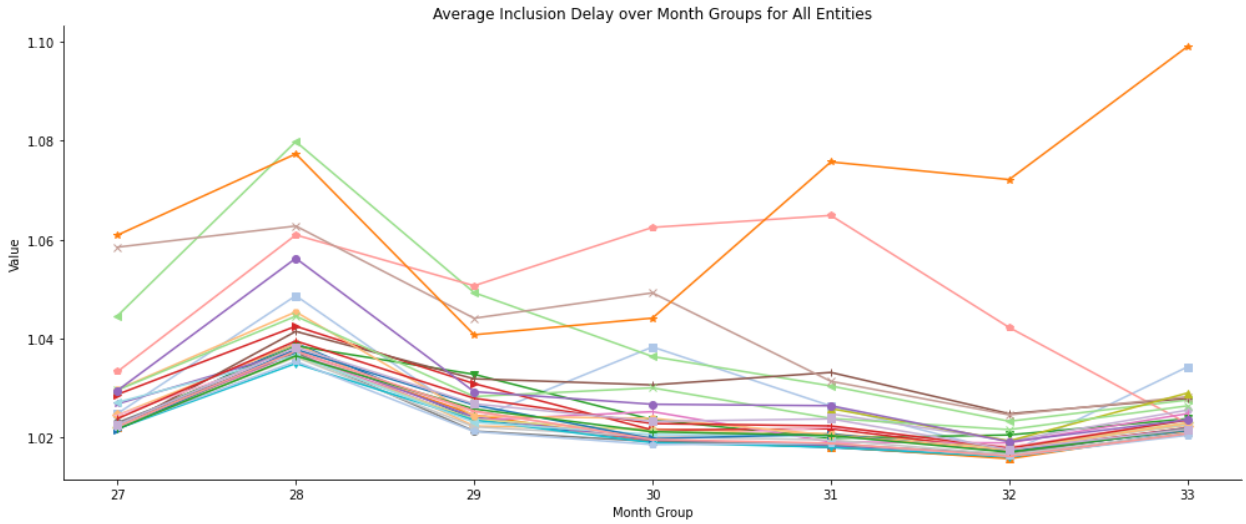
However, it's not only bandwidth and hardware that can have an impact. Perhaps you could be using Vouch with multiple beacon nodes, with different consensus clients, and you're specifically waiting for multiple clients to give the same answer before you attest. This seems very unlikely, but it exemplifies how an operator could be taking the 'right' actions to minimize risk and result in lower vote correctness.

A more realistic example might be an operator hosting multiple nodes in different geographies, but the one on the same server as Vouch is offline. In this case, Vouch would be forced to wait for a response from the nodes abroad before it can attest, which would add a meaningful delay when a block is already delivered fairly late.

So, we should be careful with the weighting of correctness as some proxy for the quality of an NO's setup. It gives us some information, particularly when NO's are regularly performing below the mean in all votes. Overall, I believe this is a strong metric for scoring, but as you'll see with most of the data available, we still must be careful when weighing these in a scoring algorithm.

Average inclusion delay

Average inclusion delay is the mean of the distance between the slot where a validator's attestation is expected by the network and the slot where the attestation is included on-chain.



Variance & Standard Deviation: A low standard deviation relative to its mean suggests that most inclusion delays are close to the average, but, as indicated by the skewness and kurtosis, there are some extreme values.

Skewness & Kurtosis: High positive skewness and kurtosis suggest that there are extreme values that might be outliers.

Range: The delay varies from 0 to 2.61 slots. There are outliers in terms of incorrect data (zero delay isn't possible so this would need to be removed from the data, while 2.61 slots is extremely delayed).

Relevance: Delays in inclusion can be a sign of poor operation and because delay can result in penalties and missed rewards, is a factor for evaluating the risk of those events. Hence, it's not sufficient to only look at historical rewards and penalties alone, as we must factor in future risk.

Transformation: Given that inclusion delay is easy to understand, we should try to retain that characteristic. We can bin or categorize the values. For instance, we can create categories like 'low delay', 'medium delay', 'high delay', and 'very high delay' based on quantiles or predefined thresholds. This approach retains the impact of outliers as well, and since our goal should primarily be to punish poor outliers, maintaining outliers is critical.

Handling Strategy: Values in the 'very high delay' category can be heavily penalized in the scoring system.

As you would expect, inclusion delay is fairly similar to the correctness data, but with a slightly greater dispersion of outlying NO's above (worse than) the mean.

Notably, Rated chooses to ignore inclusion delay for scoring, instead having it feed in based on whether the delay results in missed rewards and penalties. As mentioned earlier, the magnitude of the delay holds important information regardless of whether the delay was so extensive as to result in missed rewards and penalties.

With heavy skew and kurtosis, what we see is that the bulk of all validations happen within a small range of delays. It doesn't make sense to penalize NO's with greater than usual average inclusion delay when that delay is still fairly close to the mean.

Alone this data is not overly helpful, but when paired with the correctness rates for different votes the inclusion delay can help tell us why a vote is incorrect.

When we see excessive average inclusion delay we can be more confident that incorrect votes are as a result of the NO's setup leading to late attestations out-of-slot. This is because an incorrect vote in the correct slot would suggest that the NO's setup is operating well enough to receive data, parse it, and return an attestation on time, though incorrectly. That could be a result of client failure for example.

A delayed and incorrect attestation is more likely to be a result of a delay in receiving data and outputting an attestation, which is acceptable periodically, but less than ideal as a pattern.

This data is well-suited to be used for identifying significant outliers via a binning approach.

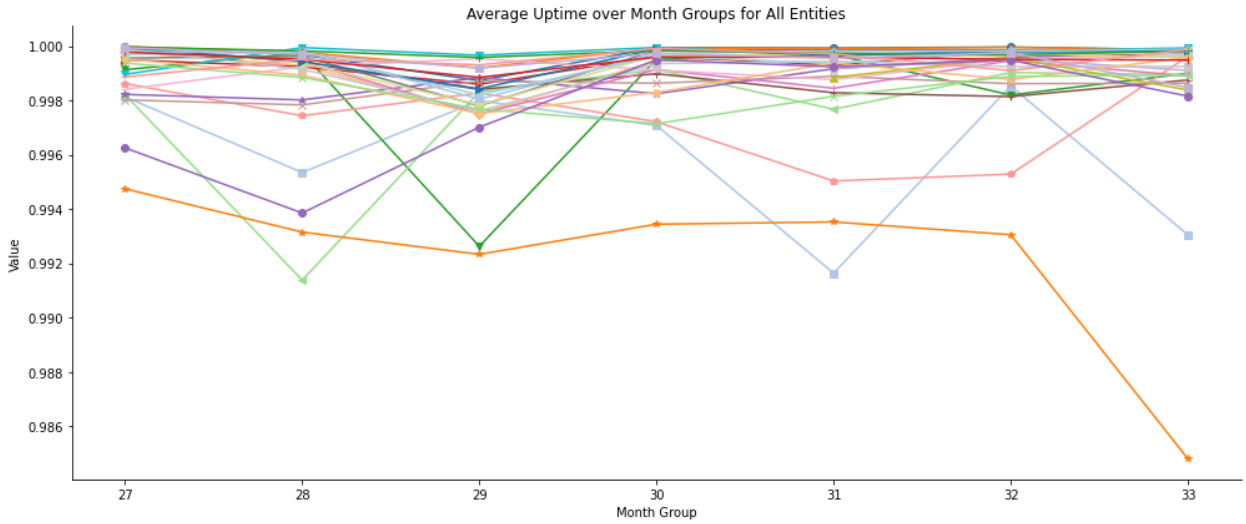
Average uptime

Uptime has some of the same patterns that we've seen so far, but noticeably the NO's performing below the mean are not necessarily the same as those with below-average correctness. However, they do more closely match those with higher inclusion delay, another signal of potential setup issues, particularly around connection quality.

Most notably, the same operator has had the poorest performance in all metrics so far.

Uptime is specifically measured as the number of epochs a validator's attestation was included on-chain divided by the number of epochs that the validator is active, aggregated across the NO's set.

So, it should have little correlation to correctness, which only looks at attestations that were made. Given that this same operator is last in all categories, as well as uptime, this hints at systemic issues.



The standard deviation is approximately 0.2%. So, this operator with a mean uptime of 99.21% is ~4.42 standard deviations below the NO mean of 99.87%.

The logic for using uptime in a scoring system is simple - you cannot maintain sufficient earnings if your uptime results in significantly more missed attestations and block proposals than other NO's. The metrics that we've considered so far only look at events that have happened. If we do this, an operator could be offline, then make a single correct attestation and score 100%. This is not good, so we do need to include some proxy for uptime, or at least a factor that looks at a broader set of factors.

However, scoring on uptime could incentivize NO's not to update regularly, or to rush their updates, such as to minimize downtime. Later, we'll look at if there's another proxy that we can use.

Block Proposals

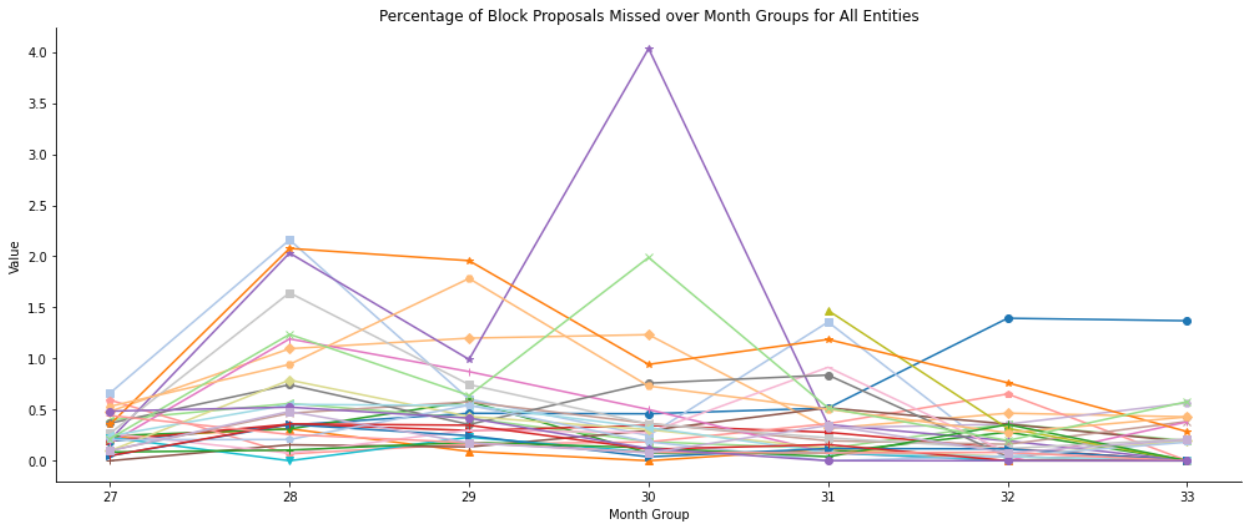
We expect that ~1/8th of our rewards should come from block proposals and the associated rewards. For this period of data, it's a little bit lower, with attestations representing more than 7/8ths of the earnings, likely because of relatively low demand for blockspace.

Yet when we look at relatively shorter periods you'll notice that there can be meaningful differences in APY among NO's with similar attestation performance. The reason is because of the huge variance in EL rewards that can come with proposing a block. This effect is still noticeable even across NO's with thousands of validators.

The majority of this variance is due to luck. NO's can take steps to maximize their EL rewards by delaying proposals to include more transactions and using relays to re-order transactions more profitably. However, the bulk of the variance in rewards between blocks is out of the NO's control.

Therefore, it's evident that the actual rewards earned as a block proposer are not applicable for scoring NO's.

However, because these blocks contribute 1/8th of our earnings over infinite epochs and can contribute an order of magnitude more over years, they are critical.

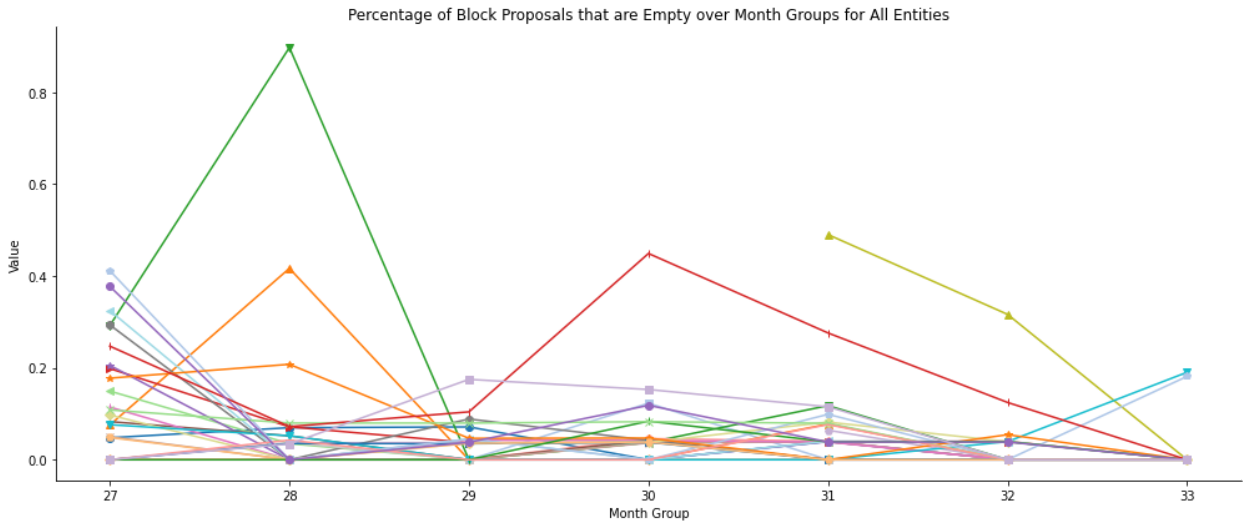


When looking at the percentage of block proposals missed we can see some dispersion among NO's.

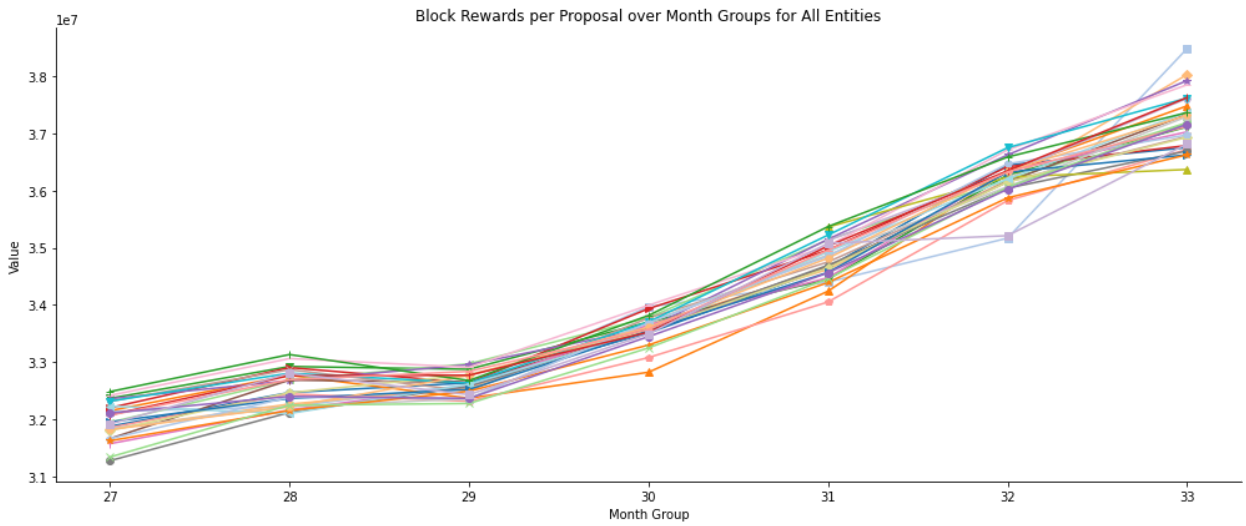
The standard deviation is approximately 0.5015. This indicates a moderate to high level of variability, suggesting that there's a notable variation in the missed proposals among NO's.

The skewness value is 3.1409, which is positive and indicates a pronounced right-skewed distribution. This pronounced right skewness suggests that many NO's are close to the lower limit (zero), but there are some NO's with significantly higher missed proposals. These NO's with higher missed proposals can be seen as exceptions rather than the norm.

Notably, there is some consistency in where NO's rank via a quintile binning approach, which could make this a suitable metric to score NO's.



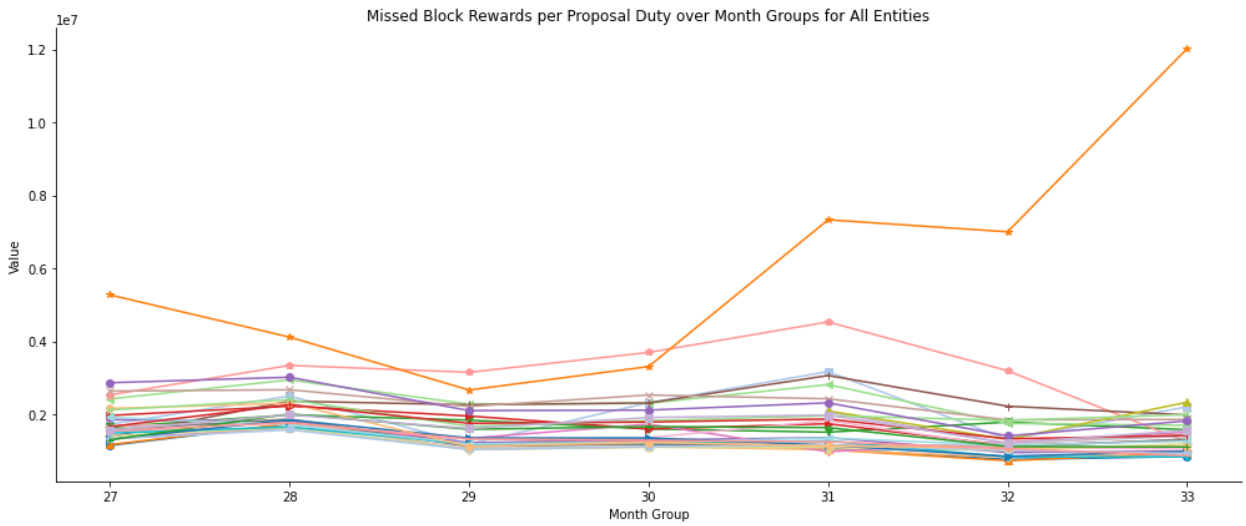
The percentage of blocks that are empty is less ideal for scoring because smaller NO's can frequently have zero empty blocks in a period only because they propose few blocks. Therefore, the rarity of these events makes it an insufficient metric, when used alone, for usage in scoring and this will likely be the case as Lido onboards a greater number of NO's, leading to higher variance in the number of block proposals per operator in a given period.



When looking at the actual block rewards earned per proposal we can see a very tight grouping. Instinctively, I would assume that any metric involving block rewards would be a poor choice due to the randomness of rewards.

However, what we see is that there is a fairly low standard deviation and high consistency of ranking position in NO's. The difference between NO's is small and so if used in a scoring system this metric needs to be scored delicately, either by only punishing significant outliers based on magnitude or consistency of rank, or by punishing only very lightly.

Overall, although this isn't a perfect metric because there is some randomness when looked at over a longer period due to the consistency of rankings this is a good choice for inclusion in scoring. If used in conjunction with the percentage of missed blocks, and potentially empty blocks even, this could give a much stronger indication of an NO's proposal effectiveness. Whereas when used as a standalone metric it's weaker and therefore needs to be scored delicately, as we discussed previously.



Missed block rewards here use approach 1 from Rated, the average of recent blocks.

Looking at missed block rewards there is a huge variance. This metric has the potential to be skewed by being unlucky and simply missing a specific block that was worth tens of ETH. You could argue that this is outside of the NO's control because they cannot dictate the magnitude of rewards and were simply offline at the time.

The counterargument is that despite this, it’s important to track because of the potential magnitude of rewards, missing a block can be either a non-issue or terrible. So, that makes it even more important to ensure that you miss as few blocks as possible and make efforts to keep validators online.

When compared to scoring on the percentage of missed blocks, the reward-based metric adds only information about the financial impact. The core problem is the missed proposal. Given that this financial impact is out of the NO's control this counterargument is fairly weak and hence it doesn’t seem appropriate to score based on this metric.

The takeaway from all of the proposal effectiveness data is that no single metric is sufficient alone. But when combined even weaker (but still usable) metrics like empty blocks can become useful if, for example, we see that a single operator has an abnormal propensity for empty blocks alongside low rewards per block and higher-than-average missed proposals. We need to combine metrics to get a higher level of confidence in scoring proposal effectiveness because in isolation each metric has

flaws. Combined, I believe that they have the potential to be a very strong part of a scoring system.

Slashing

Existing validator scoring systems do not typically take into account the slashes incurred. Slashing is the harshest penalty that a validator can receive. It can happen because of:

1. Making two differing attestations for the same target checkpoint.
2. Making an attestation whose source and target votes "surround" those in another attestation from the same validator.
3. Proposing more than one distinct block at the same height.
4. Attesting to different head blocks, with the same source and target checkpoints.

All of these slashable behaviors relate to "equivocation", which is when a validator contradicts something it previously advertised to the network. So far, all slashes incurred by validators have resulted in the minimum penalty because the incidents were isolated and therefore had no correlation penalty applied. As a result, the validators were slashed for the minimum amount, which is currently set to 1 ETH.

While slashing is the harshest penalty, for it to be financially painful to Lido it would need to occur simultaneously to a large percentage of validators to meaningfully impact APY.

For example, RockLogic recently incurred slashing penalties on 11 validators. The loss totalled 13.46 ETH. Currently, RockLogic is running 5,789 validators for Lido, earning 4.21% APY based on the 30-day backwards-looking APY. Extrapolated out to a year, they would generate 7,799 ETH in rewards. Therefore, the total loss to this slashing event is only 0.17% of the rewards they would earn. Said another way, net they would earn 4.20% instead of 4.21% for the year.

This makes little difference financially to their performance. We see a significantly greater difference in rewards between NO’s based on their attestation correctness rate and missed block proposals than we see from this slashing event.

This isn't to say that slashing isn't important and potentially far more harmful. This event was particularly isolated, with only 11 validators being slashed from a 500 validator cluster. An event ~50x the magnitude (the entire cluster) would be far more detrimental, taking APY down to 3.85%.

In this case, RockLogic was able to limit the impact by acting somewhat quickly. So, we need to decide how to evaluate slashing.

On one hand, small events like this have very limited financial impact and therefore you could argue that they should only have a relatively small impact on scoring. Perhaps this could be done by simply factoring the lost ETH into the total attestation rewards calculations and using the net number to try to score.

On the other hand, it was fortunate that the extent of this event was contained. Having slashing of any kind is extremely rare, and so this gives us some information as to how well the operator is performing their duties more broadly. Given that other NO's have not been slashed, this could signal that any NO's getting slashed are not performing their services sufficiently.

Deciding which is the right approach is highly subjective and so I'd suggest that the Lido DAO community needs to decide how to handle these events in scoring.

My personal belief is that because of the vast number of validators each operator is running, any slashing event has the potential to impact many hundreds, if not thousands, of validators depending on how they cluster. Therefore, we must look at these events in terms of what could have happened, not only what did. Given the rarity of slashing and the relative ease of preventing it, my opinion is that NO's who are slashed must be scored far more harshly than the financial impact alone.

By not doing so, you create a scoring system that incentives NO's to maximize only for financial performance, knowing that isolated slashing incidents will only marginally impact APY. This approach encourages cutting corners and can lead to an accumulation of risk.

Therefore, although it's not reflected directly in APY, I believe that because of the rarity of slashing in the data, this metric has the potential to be used harshly in scoring NO's.

Correlation of factors

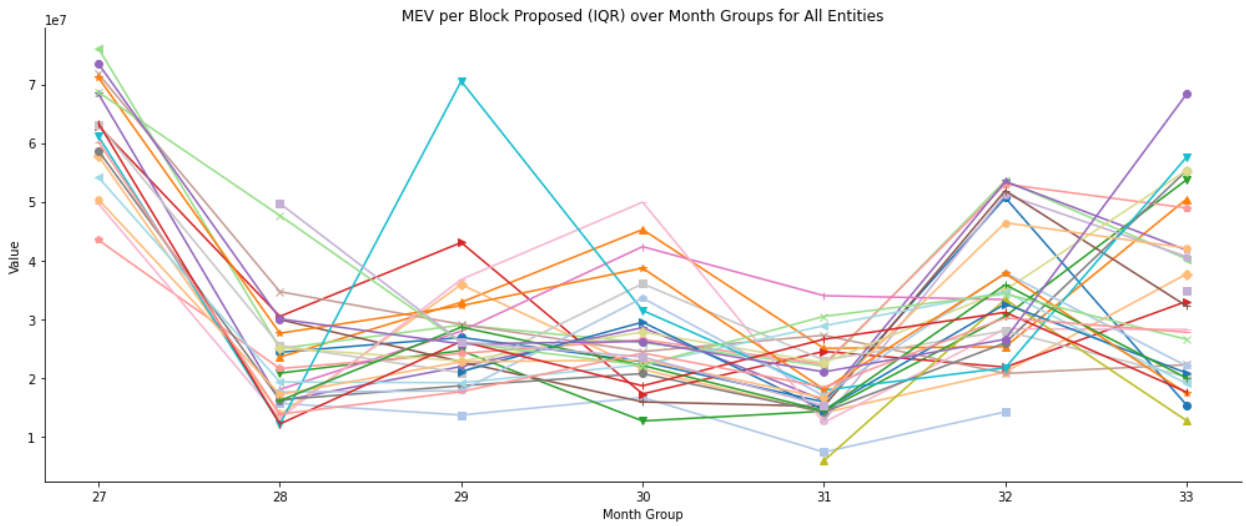
If we look at the on-chain data that we've reviewed so far, we can see that certain NO's consistently fall below the mean on the large majority of metrics. I don't believe that this is an anomaly. There can be some overlap in metrics, for example, uptime will impact rewards. However, I've made an effort to use metrics that are isolated from other factors as much as possible by for example looking at the correctness of attestations, rewards per block proposed, and inclusion delay on attestations made. Therefore, there's a very limited correlation between factors.

Yet, what we see is that **for the lower-ranked NO's** (bottom decile) their presence in the bottom decile for one metric is highly predictive of their presence for another. There is an underlying factor that correlates - the NO themselves. The infrastructure, processes, and decisions are constant across all metrics, and this appears to be the common denominator that's influencing this correlation for the poorest-ranking NO's.

When creating a scoring system it would be worthwhile to consider looking at rank position, rather than only scoring based on each metric individually. If an operator is last in a high percentage of the tracked metrics that gives us additional information beyond only looking at their aggregate score by say scoring each metric equally and taking an average.

Miner extractable value (MEV)

Over the last 30 days, MEV has generated surplus earnings above vanilla blocks of 5.6x the value of the vanilla block. The difference between vanilla and relayed blocks is huge and this can help to explain much of the variability in total CL + EL earnings between validators with similar attestation effectiveness.



We have applied the Interquartile Range (IQR) to the data to remove the values in the upper quartile, for the sake of making the graph more visually useful. Here, we look at the fees that were paid to NO's by builders, over the priority fees (tips) for the block.

We can see strong dispersion in results and most importantly, there is little consistency in the ranking of NO's across months. This inconsistency is due to the huge variability in blocks. One block can have far better permutations for MEV extraction than another and those permutations are out of the control of the operator. What is in the NO's control is whether they choose to use mev-boost, which relays they use (to an extent, depending on their server location vs relay server locations), and which bids from builders they choose to accept.

Therefore, while the MEV per block is not a helpful metric because of the large randomness, we might be able to calculate how much MEV was missed. The logic here is that we don’t care how much MEV an operator earns in a period, or per block. What we care about is how much they leave on the table, because our goal should be to maximize earnings within the bounds of acceptable practices, which should be decided by Lido DAO.

From Rated:

“There are two main on-chain patterns in which we have observed value transfers between a block builder and a validator; (a) one that involves the end_tx as parametrized in the table above, and (b) one in which the value transfer happens exclusively via the fee_recipient.”

When the value is transferred directly to the validator by setting their address as the fee_recipient, tracking the MEV is more challenging and hence it's not included in the Rated data. They do not track this because it's hard to identify specifically which block build was chosen by the validator. Therefore, you cannot easily separate MEV from the priority fees.

In the above data, we simply look at cases where the fee_recipient is set to the builder address and there is end_tx that is more than the priority_fees.

However, for our case, we do not need to know exactly what was the MEV for a block and what was priority fees. What we care about is what could have reasonably been earned by the validator for a given block, and then aggregate that data across NO's over time. This calculation will give us some approximation of their MEV and tip effectiveness, or the percentage of potential MEV and tips that they earned.

To calculate this we will need to total the priority fees and MEV that an operator earns for the period. This should be highly accurate, because although we can only decipher the magnitude of MEV separately to priority fees in 90-95% of blocks, in almost all cases we can see the total.

Then, we need to know what builders had bid for each of those same blocks and calculate the difference between those two values, which is the "missed earnings". We can express this as a percentage of the potential maximum to show the NO's effectiveness at capturing these earnings.

An interesting **paper was released in May** by members of the Ethereum Foundation, looking at the 'time value' of proposers waiting to propose a block. They concluded that "timing games are indeed worth playing for block proposers" because they enable the capture of additional MEV by delaying block proposals. Yet they believe that most "delayed block proposals are primarily due to latency in the block signing process, rather than a conscious strategy to maximize profits". Talking to NO's and Lido team members, much of this "missed MEV' is because attempts to collect it have resulted in increased missed blocks so it may be more challenging to capture than the paper concludes.

The authors believe that there is a lack of maximal MEV capture by proposers currently, either because of a "lack of common knowledge" or due to "existing social norms". Notably, they conclude that these are not "sustainable safeguards for maintaining

economic fairness" and that the blockchain will need to adapt, rather than assuming participants will maintain this behavior.

My data to replicate this paper for Lido NO's shows that there is a significant standard deviation in bids submitted by builders. Therefore, the block proposer can potentially earn far more by picking one bid rather than another. Fairly frequently (at least, far more than I expected), this can be an order of magnitude more rewards.

However, these larger bids typically come later in the slot. The risk that an operator faces by waiting to try and maximize rewards is to miss the block entirely and fail to propose at all. We know that the delay in receiving the payload from a relay can vary enormously, with some returning in less than 100ms and others regularly requiring upwards of 300ms. Errors do happen, meaning that no payload is delivered. Given the time constraints and up to 3000ms delays from relays until a timeout is sent, I would imagine that if an error is returned most will choose to not re-request from the relay and instead propose with the information that they have from others or locally built blocks, potentially missing superior bids.

Understanding the Relay Data

This payload tells us firstly, whether the operator has called the API and received a payload at all, which frequently is not the case because they only call this for the chosen bid. Secondly, from the builder bids data we can learn about the bids that the relay has delivered.

We can see when a specific validator is only receiving payloads from a set group of relays, but not others, ever.

We might want to try and understand why a validator is not accepting certain bids. Rejections could be because a validator is not using a given relay or any, they are accepting before a better bid appears, the bids are below their min-bid, or because an error occurred.

The most naive approach to discovering a validator min-bid is to identify the lowest bid that did not result in a payload and the highest bid that did result in a payload. We know their min-bid must be between these values, and so we can take the average of these values as the min-bid.

Rather than going block by block, we can look at specific NO's, as that's our goal. For each validator, an operator runs we know the address so we can get relay payload data for each validator and aggregate by the operator. Due to the number of validators that NO's run, it's necessary to be smart about when to call a relay.

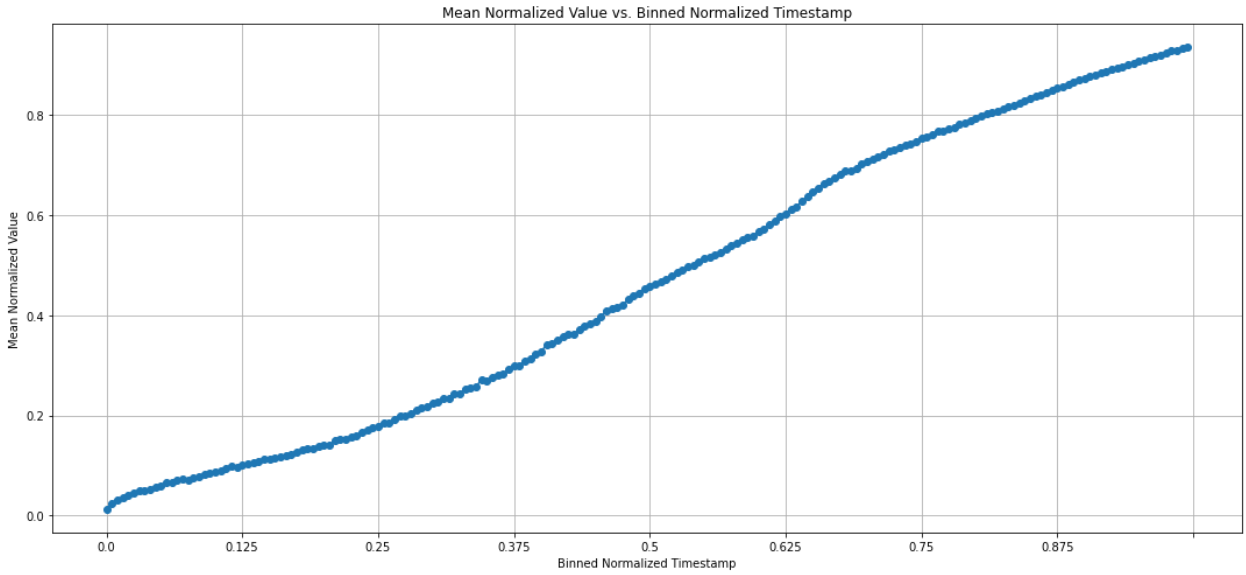
We must first identify which validator proposes a block in the slot and then check this validator index against our list of known Lido indexes. Then, if the proposer is an address in our set we can call the relays to identify which, if any, delivered the payload. Then, get from the relays the list of bids submitted and decide which was the best possible bid the validator could have chosen. From this we can calculate the difference, or the 'missed' earnings.

Generally, we see that Lido protocol NO's appear to be using a single min-bid across all validators. Where we can't be confident that this is true, this may be either due to incorrect data or more likely because the min-bid was implemented at different times for different clusters of validators. Therefore, even without requesting NO's to publicize their min-bid, we can fairly reliably decipher what it is.

It's important to know what an operator has set their min-bid at because currently Lido DAO is not specifically tracking the MEV for the blocks that NO's create. We want to avoid NO's arbitrarily setting their min-bid too high, which would reduce rewards, and we want to monitor when builder bids are not being chosen, to ascertain the missed earnings. To do this reliably, we must know whether a min-bid threshold resulted in no payload, or if it was because of a request error / the operator didn't request at all.

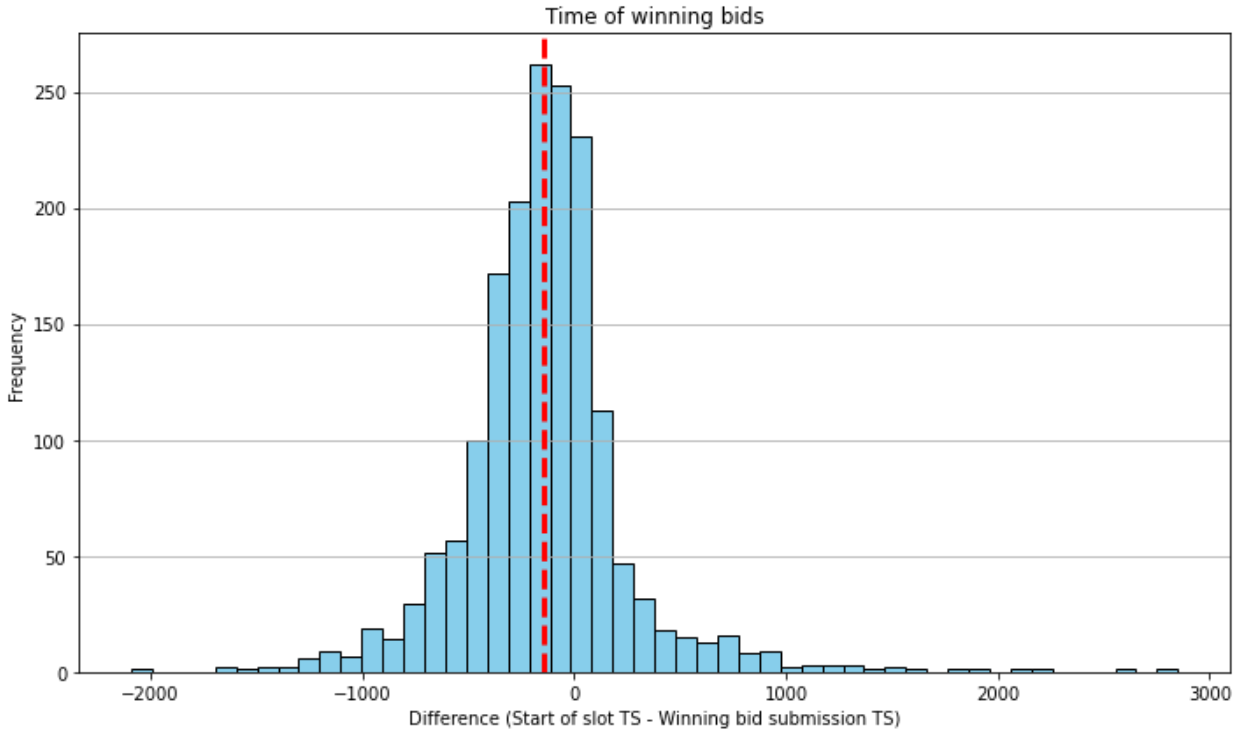
It's also possible that Lido DAO should request or require that NO's publicize their min-bid and update it frequently. If they did so, we could use this data instead, only using the above approach to verify and ensure that NO's are being truthful.

Bids vs Time



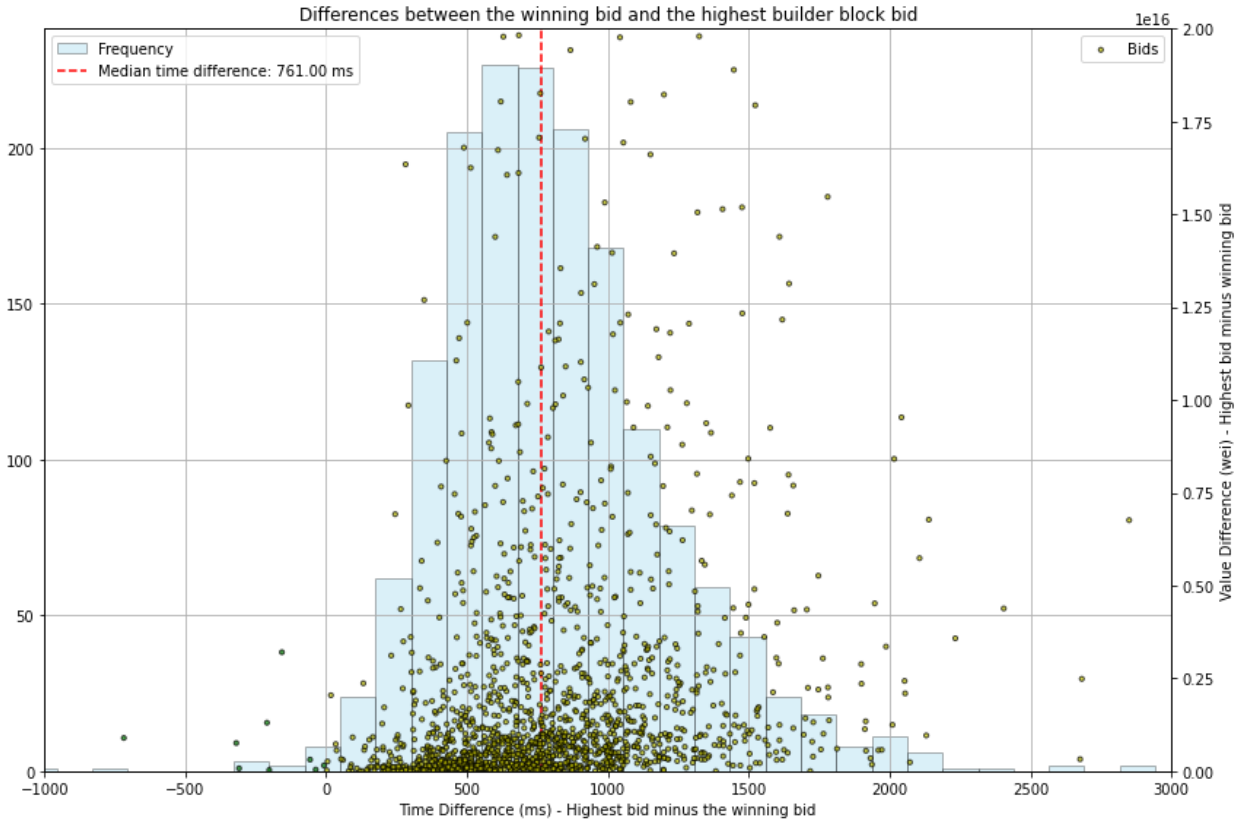
Above, we look at the value of bids by builders against when the bid is received. We've normalized the timestamp against the timestamp of the first received bid for the block and normalized the value against the higher bid received in the block.

What we can see is a fairly linear increase in value over time, with a 0.80 correlation between time and value. Hence, we know that on average, the longer that a validator waits before requesting bids from the relay the higher the value of the block. Of course, we must ensure that we're on time to avoid a missed proposal. But as we'll show, validators are typically requesting bids very early.



What we can see is that the average winning bid is received by the relay 145ms BEFORE the slot begins. NO's are not waiting into the slot before calling the relays to get the best bids, select a bid and request the payload. Presumably, this is due to fear of missing a proposal.

Payload vs Time



What we see is that the highest bids consistently come in after the selected bid, as would be expected. The median time difference is 761ms later, and this would allow the validator to capture an additional 0.0009 ETH per proposal, which is extremely close to the 0.001 that was found in the Öz et al paper. Notably, the median time difference is a little bit lower than they found, at 992ms. As they found, we see that almost all of the winning bids were selected before the highest bid being received by the relay.

We know that the average payload is received 145ms BEFORE the start of the slot, if the validator waited 761ms more they would be at 616ms into the slot. This time would still give them multiple seconds to propose the block and therefore it seems reasonable for validators to wait additional time to capture higher rewards. However, conversations with those more informed dispute this and so further research might be needed to understand feasibility.

A total of 9.8 ETH was uncaptured by Lido protocol validators in the 7 days as a result of accepting the earlier bid. This 9.8 ETH represents 5.62% of the rewards that were captured and so it's a fairly significant amount. Express another way, capturing these extra rewards would take APY from say 4% to 4.22%, and this difference is more than

the difference between Lido and other competitors' current APY. It's a meaningful amount that could benefit stakers.

Of course, not all of this extra could indeed be captured, so we would need to apply a cutoff and see what the surplus is before this cutoff. Deciding on this cutoff timestamp is outside of the scope of this research and will need input from NO's who have more context as to how much time they need to process and propose blocks. If waiting longer to capture these extra rewards resulted in more than a 5.62% increase in missed proposals, it would not be worth it.

MEV vs Vanilla and Min-bid

When comparing MEV blocks to vanilla blocks, in the last 7-days of only Flashbots blocks proposed by Lido protocol validators, we see an MEV payment of 0.03 ETH. This is well below the 14-day average for all blocks of 0.06 ETH from **MEV-Boost Pics**. This difference is likely simply due to the variance in block rewards skewing the mean so much. By only looking at half the time, for ~¼ of the proposed blocks and only for one relay, my data is less extensive and hence less useful for comparing against vanilla blocks.

For the **last 7-days Rated** execution rewards for mev-boost blocks had an average of 0.0799 in extra rewards. Of course, we must be careful when comparing in this way because it's plausible that many vanilla blocks are proposed because the mev-boost relay returned a lower value for that specific block. Or, not sufficiently above their min-bid threshold.

So, it's not obvious that we can simply use a backward-looking rewards view to compare vanilla and mev-boost blocks when attempting to calculate the potential missed rewards. This Rated metric only tells us that proposed vanilla blocks had lower rewards, but we don't know why they chose a vanilla block.

Instead, we need to find all of the vanilla blocks that were proposed and look at the bids on relays for those blocks.

Unfortunately, when a proposer is not registered with a relay the builders will typically not submit bids and therefore we have no way of knowing the potential MEV value for these blocks. All we will see is blocks where the proposer IS registered with the relay but

chose a vanilla block instead. This narrows our scope to only cases where the validator didn't call the relay, there was an error or the bids were not sufficiently high to be accepted over the vanilla block.

For our dataset of the past 7 days of all proposed blocks by Lido NO's we only have 503 (3.42% of the blocks) vanilla blocks. To be specific, these are blocks that do not have a relay tag in the Beaconcha.in data. Potentially these could be relay blocks, but they have not found a corresponding bid in the relay data and so are presumed to be vanilla. In the overall dataset for all validators globally there are 3431 vanilla blocks (6.81% of all proposed blocks in that period). In our limited data, Lido NO's are proposing vanilla blocks substantially less than the global mean.

We filter by blocks where the reward is greater than the vanilla block, and where that greater bid was received less than 1000ms into the slot.

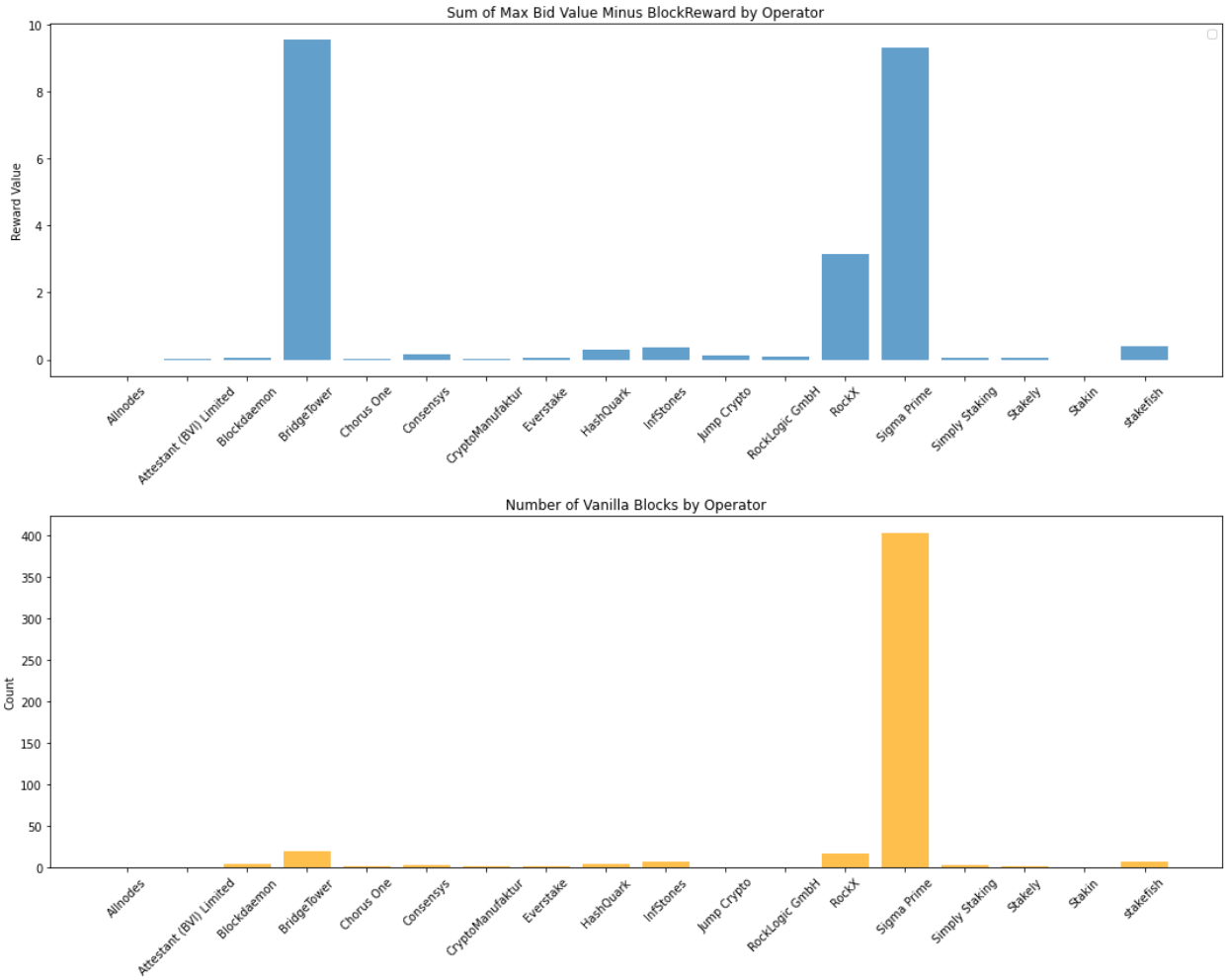
What we see is a mean MEV surplus of 0.049 ETH and a median of 0.02 ETH. For context, the mean timestamp at which the bid was received by the relay was 409ms after the start of the slot, and a 911ms median.

With MEV surplus we see a heavy right-skew, meaning that the mean is being increased significantly by these higher rewards outliers. With the timestamps, we see the opposite, a left-skew where most of the received bids are higher, but a few were received before the slot, skewing the mean.

Using the median surplus and the mean timestamps will likely give us a reasonable estimation. That would estimate a surplus of 0.02 ETH could have been captured ~409ms after the beginning of the slot. This value aligns fairly well with our previous data because it fits the linear increase in possible rewards from the start of the slot up until 761ms where we saw the median highest bid for our larger dataset.

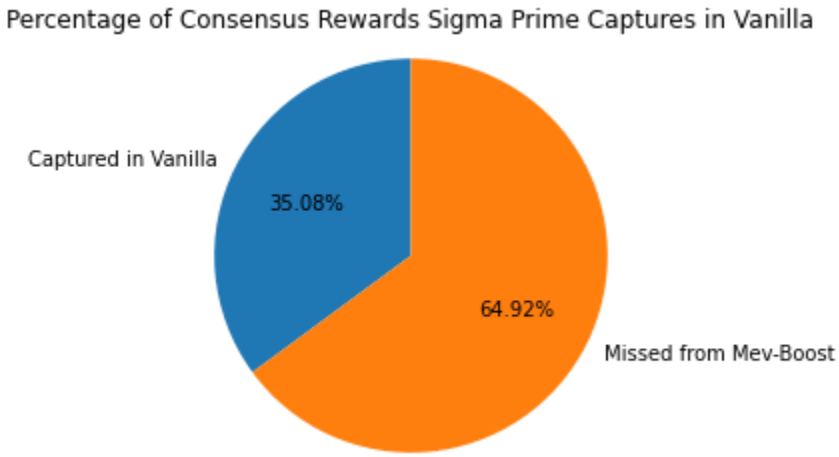
MEV by Operator

When we look by operator we see a big disparity.



BridgeTower is unfortunate in that one of the vanilla blocks had a bid of 8.4 ETH, which skews their total missed earnings. Their median missed rewards by choosing vanilla over mev-boost was 0.022 ETH which is only marginally different from the median of the whole set which is 0.020 ETH.

Sigma Prime is the other outlier, with a median missed rewards per block of 0.020 ETH (hence the median for the set, given it's the vast majority of the blocks). While they have a lower total missed rewards, this is primarily down to the period examined. Over a longer period, we should see that Sigma Prime represents ~80% of the missed rewards because of choosing vanilla blocks over mev-boost, as this is their percentage of the total vanilla blocks for the set.



In these 7 days, Sigma Prime could potentially have captured 9.3 ETH in additional rewards. Again, we need to stress that the amount they could earn will be highly dependent on when in the slot they need to receive a payload to avoid missing a proposal. This length of time will be highly dependent on their hardware, bandwidth, and latency with the relays. Lastly, again, we're only looking at Flashbots relay data here so the potential missed rewards could, and almost certainly are, larger because different relays will have different bids and the highest should be chosen.

Yet, when we look at the overall consensus block rewards earned by NO's over the months, we see that Sigma Prime ranks 1st among NO's in 3 of the 7 months reviewed, 4th twice, 9th once, and then most recently 10th.

Although they are proposing vanilla blocks much more frequently than is typical (~60% of the time over the last 30 days, compared to 2.6% for Lido overall), it's not causing them to earn less than the average. However, that doesn't negate the fact that they could **potentially** be earning more - those missed rewards still existed. The unknown here is whether those missed rewards can be captured in a way that doesn't increase missed blocks due to MEV-boost. Much more detailed and further analysis is needed here.

Therefore, while **I don't think it's appropriate to score based on MEV currently**, I do think it's important to track and consider whether more guidelines can be implemented for NO's. For example, to set a global min-bid more strictly, and lower, such as to capture a greater percentage of rewards. Currently, it appears that only Sigma Prime is

implementing a meaningful min-bid that materially influences their percentage of vanilla blocks.

Lastly, a conversation should be had with NO's to work together to gauge what an appropriate time to request a payload from relays is. By optimizing this it's possible for Lido NO's to significantly increase rewards, as I've shown here based on replicating previous research.

Further research should be able to ascertain precisely what min-bid should be set and the time at which validators should attempt to call relays to maximize rewards without increasing the amount of missed proposals. These small optimizations have the potential to further extend Lido's rewards gap against competitors, and most importantly to drive better returns for stakers.

Currently, it's not obvious that MEV data applies to a rating system. However, seeing that NO's can miss such a large amount of rewards and still rank well among Lido NO's for total consensus rewards suggests that consensus rewards and APY may be a better metric than is obvious at first. While there's a huge variance due to randomness, it's not so large as to skew the data to become useless. This is evident given that an operator can rank first in 3 of the 7 months, despite missing a huge amount of rewards available through mev-boost.

Consensus and Execution Client Diversity

Other great sources:

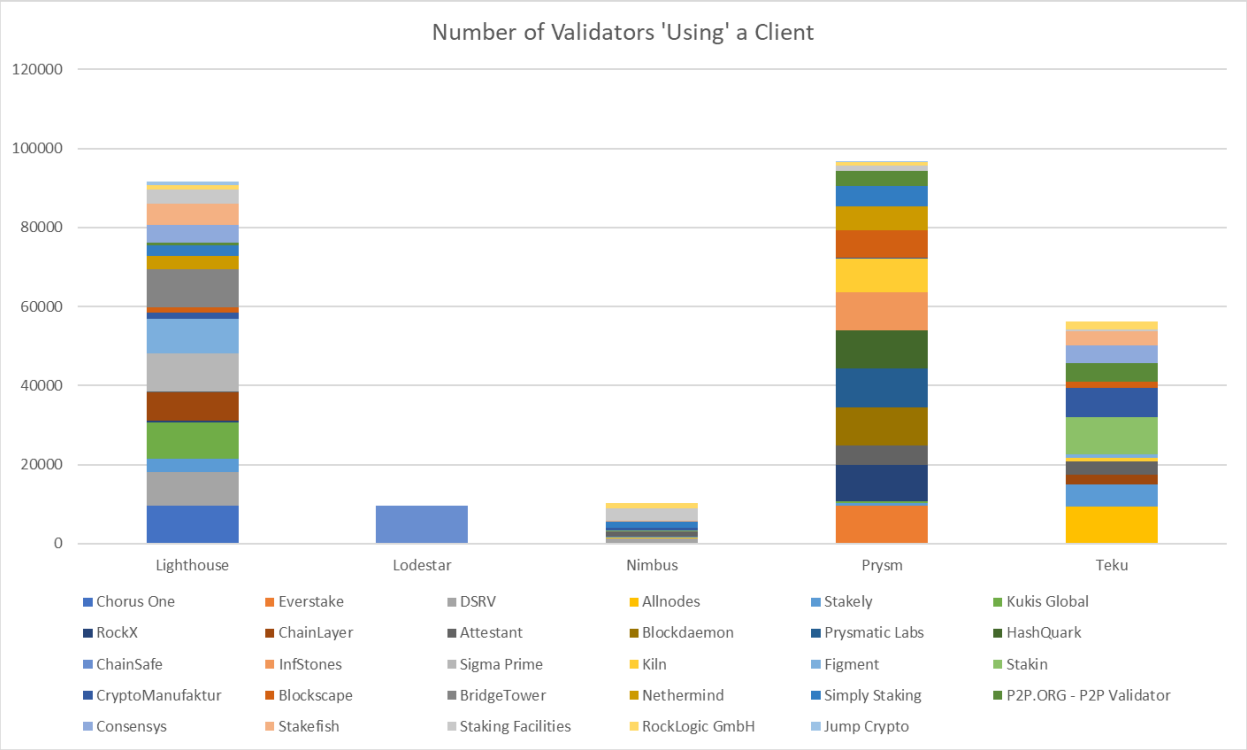
https://mirror.xyz/jmcook.eth/S7ONEka_0RgtKTZ3-dakPmAHQNPvuj15nh0YGKPFriA
https://dankradfeist.de/ethereum/2022/03/24/run-the-majority-client-at-your-own-peril.html
https://our.status.im/the-importance-of-client-diversity/
https://notes.ethereum.org/@afhGjrKfTKmksTOtqhB9RQ/BJGj7uh08

Sourcing Client Data

Currently, NOs self-report node and validator data, attesting to their usage of clients quarterly, which is aggregated at a Lido protocol level. This survey data includes the usage of systems like Vouch, which allows a node to utilize multiple clients and that makes reviewing the data a little more challenging.

First, we'll look at publicly available data sources including Rated, EtherNodes, and ClientDiversity.org data.

The below chart uses Rated data. Rated attempts to classify each validator based on its attesting and proposal behavior to estimate which consensus client it's likely using. They use the Sigma Prime "blockprint" fingerprinting code as a base, and they have built on top of this open-source library to try and improve their accuracy.



Importantly, what this data shows are the consensus clients that a validator is using. Therefore, it doesn't account for Vouch. With Vouch a node could be calling multiple consensus clients to get a response and then it will only attest and propose with one. For example, it might use Prysm, Lighthouse, and Nimbus to get a response but depending on the NO's configuration it will either vote with whichever appears to be the best response for that block, or it may always vote with a specific client unless there is an anomaly.

Given that it's built using Sigma Primes' open-source fingerprinting code, the data here is fairly similar to what Sigma Prime shares. The last option is Miga Labs, which uses a crawler to count beacon nodes and their self-reported identity. However, this means that

validators sharing a node are counted only once and nodes with fewer validators have a greater influence on the estimate. Therefore, what we would expect is that large NO's will be underrepresented, while solo NO's are overrepresented.

Today, the three datasets are highly similar, with only small differences in counts. However, that's not always the case.

Although there is no public historical data, I can attest that the variance between the three trackers fluctuates significantly, with periods where there has been up to double the count for given clients in Miga Labs vs Sigma Prime data.

More recently, this variance has been much lower. This is likely a function of the increased number of validators in the network. As the number of validators increases we would expect that the Sigma Prime and Rated fingerprinting approaches would become more accurate. As the average number of validators per node decreases, we should expect to see an increase in Miga Labs data, all else being equal.

What we've seen is the opposite, a massive increase in the number of validators relative to the number of nodes. Of course, node data is particularly poor so we shouldn't read too heavily into this, but it does imply that the Miga Labs crawler approach is likely suboptimal, presuming that we continue to see the number of validators grow faster than the number of nodes.

Under this paradigm, we'd expect that classification becomes more accurate with a growing dataset. Where classification struggles are with less popular clients because of the smaller set of data.

Both the Rated and Sigma Prime classifiers appear to rely on training a KNN model based on available graffiti. Using this graffiti they trust that it's accurate and can then assign these validators to a client.

Then, based on the attestations and proposals from these 'known' clients they can start to identify which other validators exhibit similar behavior and classify them by client as such.

The accuracy of this fingerprinting is debated, with most of the issues arising around validators that are using Vouch or similar multiclient setups connected to multiple nodes with different clients. The complexity of scoring risk, when Vouch is used, is

because Vouch adds an extra level of validation. It's not true that a block through Prysm has the same risk as a block through Prysm, using Vouch. If Prysm were to send a response that Vouch could not validate, it would reject it. So, when we look at fingerprinting data and identify that a Vouch setup used Prysm for X% of their blocks, that's not precisely the same as saying that a non-Vouch setup used Prysm for X% of their blocks.

However, there is no clear solution to this problem. If we assume that Vouch reduces the risk of the underlying client, we'll only be overcounting risk, which is a much better outcome than the inverse.

The fingerprinting will classify a given node as client A when it attests with the behavior of client A a very high percentage of the time, regardless of Vouch usage. It might not ONLY use client A, but if it's using it most of the time it's not entirely accurate to say that it's incorrectly identifying the client, only that it's failing to account for the usage of other clients, and the assumed risk reduction of Vouch, this is type II error.

A fingerprinting system can be developed to better account for scenarios where multiple clients, similar to Vouch, are used for a single validator and Rated is actively working on this problem. A split score or confidence interval approach might be sufficient here. Rather than trying to identify a specific validator as client A, for example, it's better to say that block X from validator Y is most like client A.

This block-based approach will better account for Vouch and also give us a more accurate representation of an NO's total usage when the risks are block-specific rather than underlying configuration-dependent. Currently, I'm unsure whether Rated fingerprinting data is representing the data by validator, or by aggregated blocks per operator.

Sigma Prime and hence presumably Rated, struggles with classifying Nimbus particularly but is highly accurate for Prysm. Nimbus is often confused with Teku and Lighthouse, which could lead to artificial inflation in their usage. Broadly, Lighthouse and Prysm can be identified with very high precision. More information can be found here: https://twitter.com/sproulM_/status/1440512518242197516

Given that our primary concern is around the most used clients, this gives us a high level of confidence in the data for usage in a scoring system.

It’s often said that identifying consensus clients is more challenging than execution clients, but I believe this is false. Most of the attempts at identifying execution clients rely on individual node data which as I’ve explained is unhelpful with the increasing number of validators per node.

With a presumed high usage of Geth, it makes classifying more challenging, compounded by the reduced amount of data that we can consume to identify execution clients. For classification you would need to rely mainly on block proposals and with the introduction of relays this becomes increasingly challenging. Therefore, current attempts at understanding usage from nodes and classification are likely to be poor.

Another data source comes from execution-diversity.info, which surveyed large NOs about which clients they run. The survey is anonymized and isn’t publicly used in any systems so we can start with a base assumption that these NO's are honest and have no incentive to misrepresent their setups. The surveyors estimate that 75% of the network validators are covered in their survey data and they presume that this data is likely similar to other large NO's on the network that aren’t covered.

However, it’s unlikely to be representative of home stakers, who appear to represent a majority of nodes but a very small minority of validators. Therefore, there is a meaningful gap in the data, but it’s possible that a combination of this survey data representing the large NOs and the node counting data from Ethernodes, representing the majority of nodes (more representative of smaller NO's) can be used to get a composite metric for usage across all validators.

How ethernodes data is obtained

Ethernodes uses a node scraper that connects to a node that has a free peer, collects some data, and then disconnects. This means that if a node doesn’t have any free peers, it will be invisible to the scraper. There are several things that can influence whether a node has free peers and those factors might be more or less relevant to certain clients, which would influence how ‘visible’ they are to the scraper. For example, if a client’s default peer count is set very high, it has more available slots and may therefore be more likely to have a free peer for the scraper to connect to. In this case, that client would be overrepresented in the data.

Ethernodes data doesn't answer the question "What is the client diversity of the overall network" but rather "What clients have the most free peers", with client diversity being only one variable in this equation.

Comparing execution client data

Neither is perfect, and execution-diversity.info data will require manual check-ins to receive updates to the data but, without knowing how exactly ethernodes data correlates to overall diversity it cannot be used in isolation.

Lido VaNoM NO data shows that 78% of Lido validators are using Geth, which more closely aligns with the Execution-diversity.info survey data showing 84% usage than Ethernodes 55% usage. This is to be expected, as Lido NO's are highly represented in the survey data.

For our purposes, the accuracy of global usage data is not highly important. All data sources show that Geth usage is above 50%. We know that the bulk of the risk exists when a client is being used for more than 33% of validators. As long as we can be confident that the data is not incorrectly showing a client as more than 33% when it's less, then the data is usable for our purposes.

As execution client diversity increases it would be necessary to have more precise data, but currently, this is not an issue that we face. If there is a concern around the precision of the data then a scoring system can choose to only use this metric lightly in scoring.

Client Risks

Let's consider what risks are posed by consensus and execution clients, whether they hold a supermajority or not. These are the cases that we can consider:

1. Double signing
2. Mass offline event
3. Invalid blocks

The split nature of consensus and execution in post-merge Ethereum introduces different points of failure, each with its implications. Let's dive into how an error in either

the consensus client or the execution client can affect the chain and the validator's rewards and penalties.

1. Error in the Consensus Client:

a. Block Proposer:

Wrong Block Proposals:

Cause: Bugs in the consensus client cause it to propose blocks at incorrect times, with incorrect parent hashes, or with other invalid header data.

Consequences:

< 33% usage: These blocks are primarily rejected. Result: Missed block proposal rewards.
33% - 50% usage: Chain experiences increased instability. Result: Delayed finality, causing inactivity leak penalties.
50% - 66% usage: High risk of chain forking. Result: Inactivity leak penalties on both forks.
> 66% usage: Faulty block proposals become the standard. Result: Compromised Ethereum chain integrity.

Fork Choice Rule Misinterpretation:

Cause: The consensus client misinterprets the fork choice rule, proposing blocks on a non-canonical chain.

Consequences:

< 33% usage: The majority will ignore these blocks, resulting in missed rewards for the proposer.
33% - 50% usage: Finality is delayed, causing an inactivity leak for all validators until finalization resumes.
50% - 66% usage: A chain fork occurs, with inactivity leak penalties on both sides. Significant financial losses might ensue until a consensus is reached on the correct chain.

> 66% usage: The non-canonical chain becomes the dominant chain, compromising the integrity of Ethereum.

b. Attester:

Double Voting:

Cause: Bugs leading to double votes for multiple blocks in a single epoch.

Consequences:

< 33% usage: Due to such widespread usage, the correlation penalty ensures that the slashing per validator is extremely significant.
> 33% usage: 100% of the ETH on the validator's impact is slashed.

Surround Voting:

Cause: The client allows attestations that "surround" another.

Consequences:

< 33% usage: Due to such widespread usage, the correlation penalty ensures that the slashing per validator is extremely significant.
> 33% usage: 100% of the ETH on the validator's impact is slashed.

Wrong Attestation:

Cause: The consensus client produces incorrect attestation data but without any slashable actions.

Consequences:

< 33% usage: Attestations are largely ignored, leading to missed rewards and potential penalties for these attestors but with no impact on finalization.
33% - 50% usage: Increased chain instability and delayed finality. Inactivity leak penalties begin to affect all validators, particularly those attesting incorrectly.

50% - 66% usage: The network sees two competing versions of the chain, leading to forks and inactivity leaks on both sides.
> 66% usage: The incorrect attestations become dominant, affecting the canonical chain's progression and integrity.

Mass Offline:

Cause: Bugs preventing the consensus client from staying synchronized with the network or crashing frequently, leading to widespread validator inactivity.
Consequences:

< 33% usage: The majority of the network remains active. Validators using the faulty client experience missed rewards.
33% - 50% usage: A significant portion of the network goes offline. Finality is delayed, causing inactivity leak penalties to affect all validators, not just those using the faulty client.
50% - 66% usage: The network struggles to reach consensus due to the mass offline event. Chain finality is at high risk, leading to widespread inactivity leak penalties with high impact.
> 66% usage: The Ethereum network grinds to a halt due to the mass offline event. Emergency interventions might be necessary to restore network operations. Meanwhile, there is a catastrophic inactivity leak causing enormous financial impact.

2. Error in the Execution Client:

a. Block Proposer:

Invalid Block Proposals:

Cause: Execution client produces blocks with invalid state transitions or transactions.

Consequences:

< 33% usage: The majority rejects these blocks. Impacted validators experience missed block proposal rewards.
33% - 50% usage: Chain instability due to increased invalid block proposals leading to inactivity leaks as the corrupted execution clients validate this incorrect block.
50% - 66% usage: Risk of chain forking with inactivity leak penalties on both forks.

> 66% usage: Invalid blocks dominate the chain causing severe loss of trust and the potential request for a fork.

State Miscalculations:

Cause: Incorrect state computations by the execution client.

Consequences:

< 33% usage: The majority maintains the correct state, but the faulty proposers miss rewards.
33% - 50% usage: Frequent chain reorgs as the network contends over the correct state. Inactivity leaks start affecting validators.
50% - 66% usage: Risk of a chain fork based on differing state views. Both sides suffer from inactivity leaks.
> 66% usage: The network may largely operate in an incorrect state, disrupting Ethereum's proper functioning.

b. Attester:

Attesting to Invalid Blocks:

Cause: Attesting to blocks with incorrect state or invalid transitions.

Consequences:

< 33% usage: The majority ignores these attestations. As a result, the attesters missed attestation rewards.
33% - 50% usage: Chain instability due to conflicting attestations, causing delayed finality and inactivity leaks.
50% - 66% usage: The chain risks forking, causing larger inactivity leaks.
> 66% usage: Majority attests to invalid blocks. Result in compromised Ethereum chain integrity.

Mass Offline:

Cause: Bugs causing the execution client to crash or become non-responsive, preventing validators from proposing valid blocks or making accurate attestations.

Consequences:

< 33% usage: The majority of the network remains active. Validators using the faulty client experience missed rewards.
33% - 50% usage: A significant portion of the network goes offline. Finality is delayed, causing inactivity leak penalties to affect all validators, not just those using the faulty client.
50% - 66% usage: The network struggles to reach consensus due to the mass offline event. Chain finality is at high risk, leading to widespread inactivity leak penalties with high impact.
> 66% usage: The Ethereum network grinds to a halt due to the mass offline event. Emergency interventions might be necessary to restore network operations. Meanwhile, there is a catastrophic inactivity leak causing enormous financial impact.

We can try to gauge the likelihood of these events and their severity. We rank from 1 to 5, with 1 being the lowest likelihood, network, and operator impact, and 5 the opposite:

Client	Cause	Likelihood	Network Impact	Operator Impact	Product
Consensus	Wrong Block Proposals	1	5	2	10
Consensus	Fork Choice Rule Misinterpretation	1	5	2	10
Consensus	Double Voting	1	1	5	5
Consensus	Surround Voting	1	1	5	5
Consensus	Wrong Attestation	3	5	2	30
Consensus	Mass Offline	4	4	2	16

Execution	Invalid Block Proposals	2	5	2	20
Execution	State Miscalculations	2	5	2	20
Execution	Attesting to Invalid Blocks	2	5	2	20
Execution	Mass Offline	4	4	3	24

These values are highly subjective, but this is my best impression of where the risks are with consensus and execution clients. **Dankrad Feist** inspired this analysis structure. He combined impact into a single metric, but I think it's better to consider separately the impact on the network and then the financial impact on the operator.

In the case of a mass offline event, it has a large impact on the network because it could prevent finality and allows for malicious NO's to more easily conduct attacks. Yet, for a consensus client bug to cause the offline events the financial impact would be less than for an execution client bug because the breadth of downtime would cause a smaller inactivity leak.

Like Fiest, my conclusion is that the largest risk is posed by an invalid block, despite less immediate financial impact to the NO's. Invalid blocks break the trust of the chain and that fundamentally undermines its purpose. Then, mass downtime is the second largest risk, with the risk of slashing events coming last due to their likelihood.

We also see that despite there being more possible issues for consensus clients, the scale of the impact is reduced because of greater diversity. Hence, while there are extreme risks for both clients, the lack of diversity in execution clients makes it more concerning for me, despite having a longer and stronger track record.

When we look at the causes of the events, presuming they occur, it's primarily as a result of clients attesting to invalid blocks. If a single validator proposes an incorrect block but every other validator rejects it, there is no issue. The issue only occurs when there is usage of a client, either execution or consensus, from both the proposer and a large percentage of those validators attesting.

In his blog post about client diversity, **Jim McDonald therefore suggests** that it's acceptable, perhaps even ideal, to use popular clients for proposals but not for attestations. On the consensus layer, this is perhaps true when the usage is less than ~50%. For the execution layer, there is a significant supermajority currently and therefore it's highly risky to use it for either attestations or proposals.

There is an enormous difference in risk between clients having less or more than 33% usage. However, when we think about using this for scoring NO's we need to think about both the risk for the network, but also for the Lido set of validators itself.

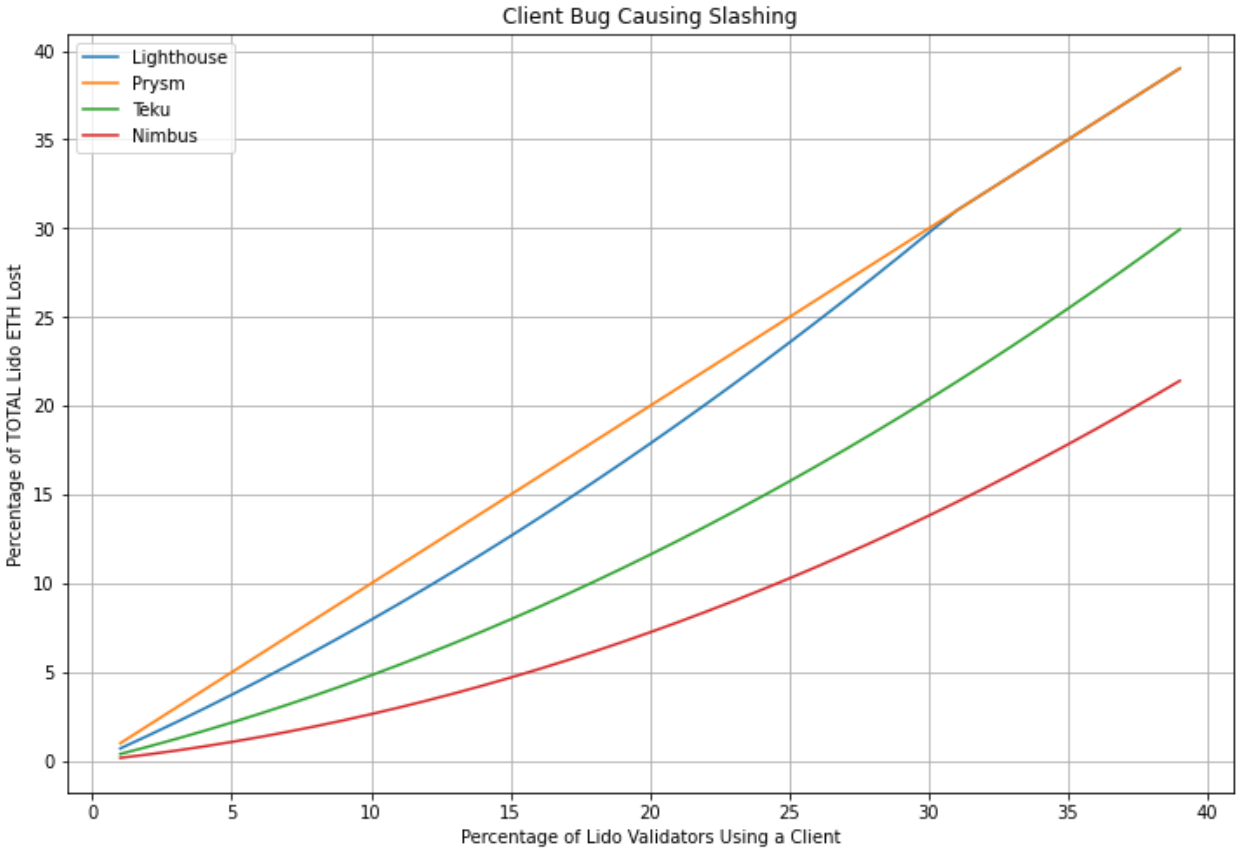
Let's consider the expected change in risk if Lido NO's chose to diversify their client usage much more, to the extreme of using each client equally. We can assume there are six popular consensus clients and four execution clients.

For slashing events the Lido set now faces a much smaller risk of slashing from client bugs, assuming that a bug exists only in a single client.

If we consider the Lighthouse risk, a bug causing slashing for all Lighthouse validators globally would have 32 ETH in slashing per validator because of the high correlation penalty. In the case where Lido had lower exposure, the same event caused 27.5 ETH in slashing per validator. Taking Lidos exposure from 33% of ETH lost to 14.3%, because not only is the correlation penalty reduced due to lower global usage but also our net exposed number of validators.

We know that the correlation penalty reaches its max when ⅓ of the network is slashed and so again, this imposes the importance of clients having lower than ⅓ usage globally.

Yet, if we look at Prysm, taking Lido exposure from 23.1% to 16.6% does not reduce the correlation penalty, it only reduces the total loss because of the reduced number of validators exposed.



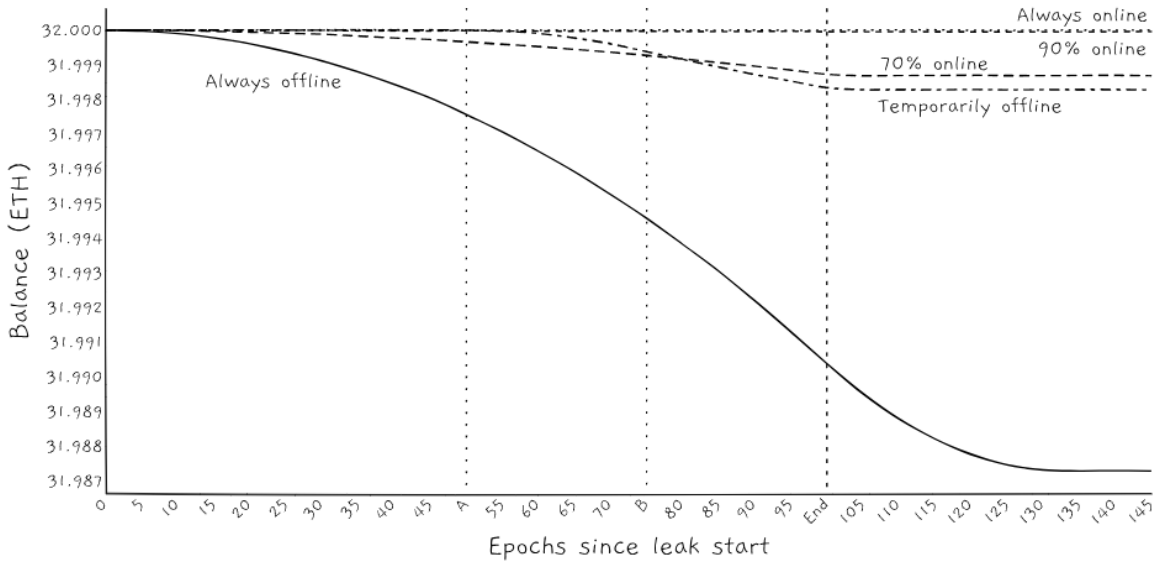
We can see that the curves for each client are very different and that's because of the global usage, regardless of Lido usage. For example, if if a single Lido validator runs Prysm it will still be slashed for 100% of its ETH in the case of a client bug causing slashing because Prysm has greater than ⅓ usage across the network.

Therefore, we can argue that the data suggests that the appropriate usage of clients for the Lido set will be different for each client. However, we can't only think about a slashing event, we need to consider other risks, particularly as we've identified that slashing is one of the lower-importance risks, though, the most financially impactful to each validator involved.

The three other scenarios to consider are the inactivity leak, missed rewards/penalties and the finalization of incorrect blocks.

Let's tackle missed rewards and penalties first, as it's the simplest. These will scale linearly with the usage of a client inside of the set and the global usage here is irrelevant to our financial impact.

Finalization of incorrect blocks can only occur when the network has greater than 50% concentration, and this would still give participants time to try to rectify the situation during the inactivity leak. With more than 66% concentration, as with geth, finalization will be nearly unavoidable, and this depends on global usage, not just Lido. However, given that Lido is ~⅓ of the network, we must consider that our usage does have a large impact on the global concentration, particularly with geth. For example, zero geth usage in Lido would instantly ensure that geth is no longer a supermajority.



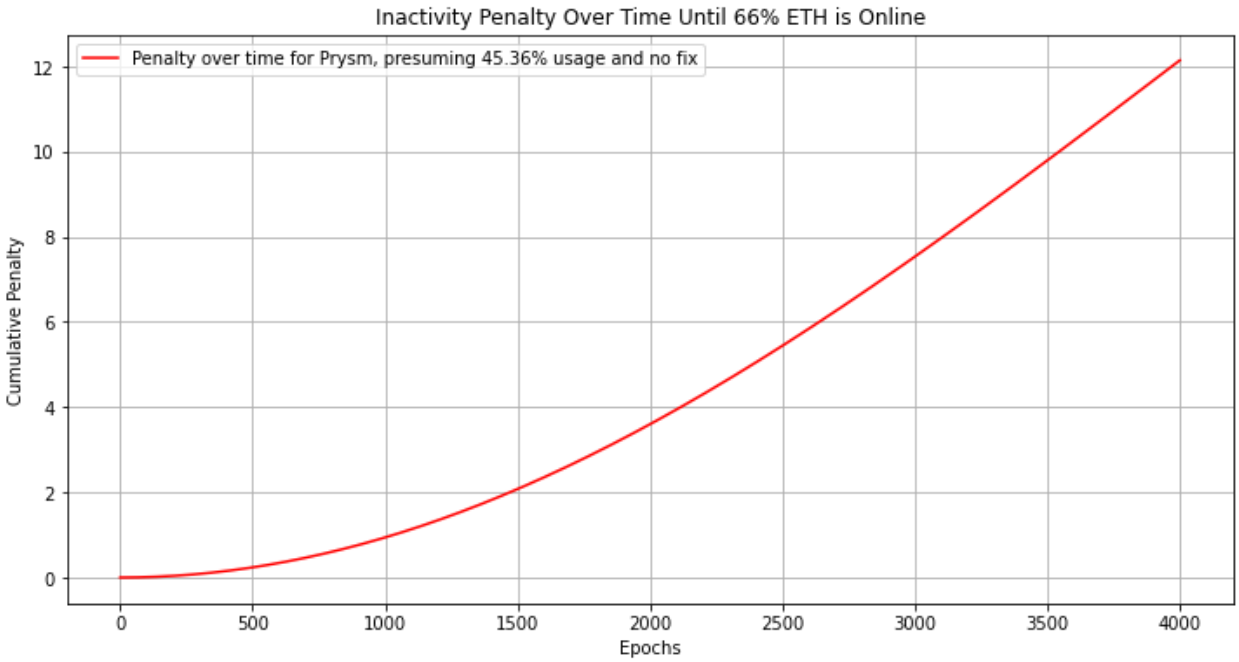
For the inactivity leak, the financial impact on Lido depends on:

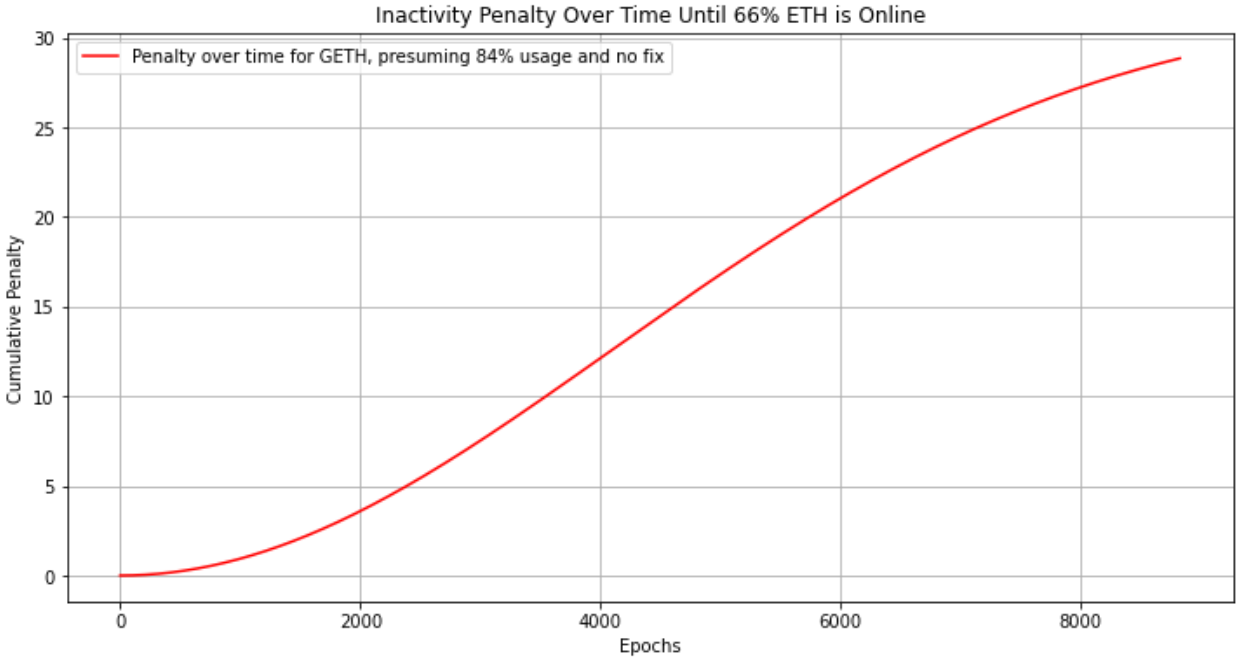
1. The % of Lido validators using the impacted client
2. The time it takes NO's to get back online, either using an alternative client or with a fixed version of the impacted client
3. The time that the inactivity leak lasts, which is variable based on the global client usage if we exclude 2.

If we presume that the 2nd component is fixed regardless of which client is used, then the only variables are the global usage and the Lido usage of the client. This simplifies the situation, of course, but not to the extent of being an ineffective review.

The governing factor will be the level of Lido usage because although those unimpacted by the client bug can still receive penalties as a result of normal downtime, it’s not

quadratic like those impacted. Therefore, again, the dominant factor in the total Lido penalties accrued is the variable number of Lido validators running the impacted client.





This supermajority on the execution client side is suboptimal. However, we need to consider what risk this poses to the network. With a consensus bug or attack, we would expect to see different votes and this could lead to an inability to finalize, or incorrect blocks being accepted. With an execution client bug or attack, we could see issues in transaction handling and the construction of payloads. Depending on the issue, we might see that the execution client never produces a payload, preventing the validator from submitting a block with transactions, but it likely can still submit some block. In another case, we might see that the constructed payload and transaction handling result in a payload that the consensus client cannot handle, and this would result in a missed block or even an incorrect block.

Of course, an inactivity leak because of a client bug can only occur when that client is more than 33% of the network so it's also important that we track this global usage.

Again, it's clear that risk starts to become more significant when the global usage of a client is higher than 33%. But ideally, we would see usage below this, because slashing risk will reduce linearly as usage decreases.

Internally, the correct level of Lido usage will depend on the broader global usage. It might be possible for the DAO to identify some correct level for another metric that can be applied broadly, and that will then dictate the correct usage. For example, by aiming to limit the maximum slashing risk because of a client bug to 20% of the total Lido ETH, this would allow us to create a scoring system that scores NO's lower when the total Lido usage of clients is above ~20% for Prysm, ~22% for Lighthouse and ~28% for Teku. Where precisely this "cutoff" starts is somewhat subjective, but the above data can help determine it because it will allow the thresholds to adjust over time with the usage data. Given that we would want to avoid a scenario whereby NO's are incentivized to regularly change clients, which introduces a new host of issues, any scoring system will need to use a light touch when client usage is not extreme.

The final case to consider is Vouch. Given the large number of possible configurations, we must use a rule of thumb that can be broadly applied to Vouch. My prior is that Vouch is a net positive for NO's, and that its use should be encouraged. However, there does again reach a level of usage that starts to introduce risk. Although Vouch cannot directly sign and therefore does not have the same risk profile as clients, it does have the potential to cause downtime in the case of a bug that prevents Vouch from outputting a result. Hence, we must also consider the risk of this downtime and how it would financially impact us. Broadly, this risk is only significant in the case of very high

usage in the Lido set, or when global usage is more than 33% of the network. Tracking Vouch usage via fingerprinting is extremely challenging, which currently prevents us from obtaining any reliable data on network-wide use. Currently, I have no good suggestion on how to monitor and score Vouch usage directly.

For scoring the underlying clients being used as part of a Vouch configuration, we have two possible approaches. The first would be to use fingerprinting data by NO's, such as the Rated or Sigma Prime data.

The advantage to this is that it looks at the actual attestations and blocks proposed by a validator and classifies that validator as a single client. This approach is helpful when we want to factor in the actual usage of the clients that are connected to Vouch. For example, although you may configure Vouch with say 3 clients, if client A is making the vast majority of attestations, it might be more appropriate to use the fingerprinting data which counts this cluster of validators as more than 33% for client A, despite it being configured with 3 clients.

The downside is that the fingerprinting is fairly brute force, and so if a cluster of validators with this configuration happens to attest on average with client A, it might only classify them as client A. Here we get both type I and type II errors, but for our scoring system the most impactful is likely to be type II, the non-counting of usage of a client that truly is being used i.e. we're not correctly identifying our true risk exposure. While overcounting risk exposure could negatively impact a specific operator, for the Lido set this is a smaller issue than undercounting our true risks.

The second approach would be to survey NO's, as is done quarterly currently, on their client usage and to request additional information about their precise Vouch configurations. Knowing that a specific cluster of validators is using Teku, Nimbus, and Lighthouse in a configuration that doesn't wait for full agreement would allow us to assign ⅓ usage to each.

The downside to this approach is that we assign usage and calculate risk based on configuration, not on the actual attestations and block proposals made. Again, Teku might be dominating the actual usage here and that would not be represented via this survey approach. The upside is that we reduce our type 2 error per client by at least assigning ⅓ of the risk to each.

When Vouch is working correctly, the downtime of a specific client should have a minor impact, as it will use responses from those nodes that are still online. Therefore, the risks are primarily around invalid blocks and slashing.

Here, the risks occur based on the content of the attestations and block proposals, and so which client is being used in reality is more important than the actual configuration. Hence, my suggestion is that we rely more heavily on fingerprinting data than on survey data.

Although it has issues with accuracy, it will still be more accurate than the survey, because a client can be configured into Vouch and yet never be responsible for an attestation or proposal. If folks believe there is a need, it would be possible to combine the fingerprinting and survey data to create a composite score, which is likely to reduce type 2 errors, by increasing type 1 errors, which may be an acceptable tradeoff.

Identifying Lido NO's Client Usage

For network data, we must rely on public sources that we've discussed already. But for Lido protocol validators we have direct contact with the NO's and hence we have an opportunity to gather more accurate data on our specific set of validators.

There are two reasonable approaches; either conduct a periodic survey (similar to VaNoM) or require that Lido NO's use graffiti for signing. In the future it may be possible for clients to write data into the blocks detailing which client is used - though this isn't currently the case, Lido DAO may have some power to push for clients to include this feature.

Graffiti has the nice characteristic of being saved to history forever, making it easy to cross-reference in the case of an operator being accused of lying about their true client usage. Likewise, by being on-chain it would allow a scoring system run on-chain to more easily access this information, though the feasibility of doing this at scale could be problematic, I have little context here.

For both graffiti and the survey we rely on the NO's being truthful. However, any claims can be checked. I believe that the fingerprinting data available currently is highly accurate at identifying what client a validator is using. It struggles at counting the

potential clients used by nodes connected to Vouch, but this shouldn't impact the success of the classifier at identifying the primarily used clients by the validator.

Based on this graffiti it will be possible to check it against the fingerprinting data. Given that the graffiti will be representative of the actual client used for the attestation or proposal, Vouch is irrelevant here. If there is a significant discrepancy to create suspicion this could initiate a more manual investigation to understand the difference and validate the truthfulness of the graffiti.

Lido already enforces certain standards like requiring that NO's utilize at least one relay from the "must-use" relay set. Therefore, it doesn't seem unfair or onerous to require that NO's use graffiti to disclose their client usage.

The major benefits of using graffiti over a periodic survey are that it should be more real-time and it will contribute to the accuracy of the fingerprinting. These systems utilize graffiti and so by increasing the usage of graffiti, the classifier should become more accurate.

Lastly, a more accurate but perhaps more onerous suggestion was inspired by Jim McDonald. It might be possible to have NO's run a batch script that would share certain information. Taken to the extreme, it could be possible to use such a script for NO's to share information about their hardware, setup, and client usage. Although I imagine that this would be unpopular, it's the most accurate approach to obtaining off-chain data.

Web3Signer, Dirk, Key Management Software

We've established that much of the Lido risk exists with the NO's procedures and policies. Key management is a particular concern and ideally would be an appropriate way to monitor and potentially score NO's.

The issue is with data. There is currently zero public data around key management nor for the Lido NO's. Even if data were to exist, it's hard to imagine how it could exist in a numerical format that could be consumed into an automated scoring system.

Hypothetically, we would want to understand the number of unique withdrawal keys that exist for the NO's set, how they are stored, and in what size clusters are they stored. Using a single withdrawal key for multiple validators makes that single key more valuable, which is more likely to result in attempts to steal the key. However, storing

multiple keys in the same place leads to the same problem, so multiple keys will need to be physically or logically separated to reduce the impact of any loss.

Validator keys provide no access to funds themselves. Indirect attacks such as blackmail are possible, as are spoiling attacks where the attacker's goal is for you to lose funds rather than for the attacker to gain them.

As of November 2022 Lido NO's were using:

Node NO's	Dirk	Web3Signer	No External Signer
27	26%	19%	59%

The usage of an external signer can greatly increase key security. Given this is the only data available, the DAO may consider whether it’s reasonable to score NO's lower if they do not use an external signer.

Broadly, I think the benefits of using an external signer far outweigh the cons and so a scoring system that incentivizes NO's to use an external signer is something I would support. A second consideration would be how precisely the external signer is being used, which is equally important, but likely this stretches into the realm of being too hard to score.

Conclusion

Our focus is on developing a NO scoring system that effectively balances stakeholder rewards with risk mitigation. While NO’s can influence the rewards their validators earn, they have far greater control and responsibility for minimizing penalties and slashing. Therefore, optimizing for these risk mitigation factors should be a better proxy for long-term performance.

By viewing a scoring system through the lens of risk, it will allow us to craft a healthy and resilient validator set, that should necessitate higher performance over infinite epochs.

Our investigation centers on identifying potential data sources and metrics crucial for an effective NO scoring system. We explore both on-chain and off-chain factors,

recognizing their distinct impacts on overall NO performance and risk management. On-chain, we analyzed millions of data points to identify whether the datasets have the characteristics necessary to be used in scoring.

Key Findings and Recommendations

Limitations of On-Chain Data-Only Systems: While it is feasible to create a NO scoring system solely based on on-chain performance data, such a system would be substantially deficient. It would fail to comprehensively account for the myriad of risk factors, which are pivotal in ensuring a robust and reliable scoring system. We found that overwhelmingly risk mitigation data was necessary for the creation of any NO scoring system. Without it, any other system would optimize for another outcome, without any insight and transparency into the unknown accumulation of these risk factors. Therefore, an on-chain data-only approach would be considerably limited in its effectiveness.

Priority of Off-Chain Risk Data: Our research underscores the paramount importance of off-chain risk data in any effective scoring system. While this necessitates a departure from a fully trustless scoring system, it's a necessary compromise to achieve a realistic assessment of NO performance.

Risk-Based Scoring Framework: The scoring system should prioritize minimizing penalties and slashing, key factors in long-term performance and stability. Key risk factors include internal processes, hardware, client and server locations, jurisdiction, and operator concentration.

Incorporating On-Chain and Off-Chain Data: The system should utilize a blend of on-chain data and critical off-chain risk data. This approach acknowledges the necessity of human involvement and increased transparency from NOs for a comprehensive risk assessment. Gathering this data is likely to conflict with a transition to permissionless anonymous NO's. We find that it will be critical to create an incentive structure for NO's to truthfully disclose information and a remediation system to investigate discrepancies. Without this information, the DAO has little transparency into the accumulation of risk in these factors and hence cannot properly maintain the health of the set.

MEV Data Exploration: Our study has identified MEV (Maximal Extractable Value) data as an intriguing area for future exploration. This includes potential optimizations for capturing MEV and tracking/preventing MEV theft by operators. However, currently, MEV data is not a viable metric for the scoring system due to implementation challenges and its relative unreliability as a dataset.

Data Source Reliability and Selection: Rated.network is identified as a suitable source for on-chain data, given its accuracy and robust API.

Community Engagement: Engaging with the Lido DAO community is crucial, especially around areas like client diversity, MEV strategies, and key management, to ensure the scoring system aligns with community values and risk tolerance. We find that the DAO may benefit from creating stricter mandates for NO's regarding systems, internal processes, and information disclosure. The economic value to NO's from participation in Lido is immense and hence the DAO has significant power to enforce standards that will allow for the creation of a stronger scoring system and a healthier validator set.