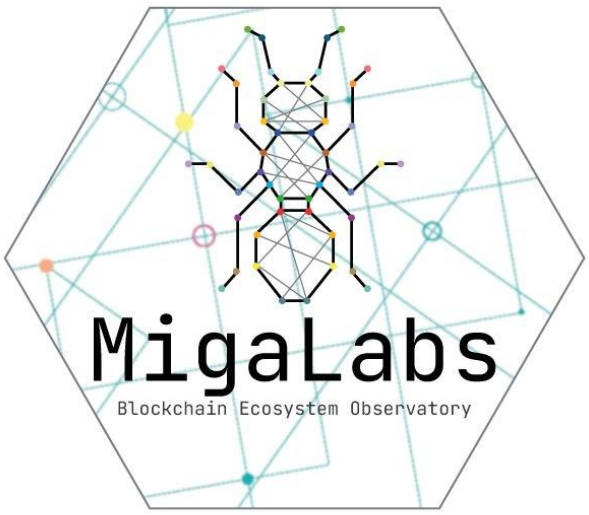
Obol DVT Performance Analysis



Study done by **MigaLabs**

Table of Content

Obol DVT Performance Analysis

Introduction

The Ethereum Beacon Chain was launched in December 2021, which was the beginning of the journey towards *Proof-of-Stake* (**PoS)**. In this new chain, the validators are the main actors, instead of the miners in the previous *Proof-of-Work* (**PoW)** chain.

After **The Merge** in September 2022, validators now govern the Ethereum Mainnet network. Anyone can activate a validator in the chain by depositing 32 ETH and running a node, which will earn rewards for actively performing duties (attestation, sync committees, and block proposals) in the network.

An Ethereum node is composed of:

- An **Execution Client**:
  - It operates in the **Execution Layer.**
  - It executes block payloads (transactions, smart contracts, etc.).
- A **Beacon Client**:
  - It operates in the **Consensus Layer.**
  - It decides which is the canonical chain with the rest of the network.
- A **Validator Client**:
  - It only communicates with the beacon client.
  - It signs the needed duties with the validator's private key.

Several validators can run together in a single node (for example, 500 validators in the same node). However, a validator today is always run by a single operator (on one single machine). So, if that operator or machine goes offline, then the entire validator stops running until the operator is online again.

Distributed Validator Technology

Distributed validator technology, or DVT, is a critical security primitive that allows a single Ethereum validator to be run on a cluster of nodes working together as a distributed validator. DVT removes the single-point-of-failure for validators, creating an active-active redundancy with a failure threshold. That means that if one or several nodes fail to send their partial signatures, the distributed validator keeps performing its duties for as long as enough nodes (over the threshold) submit their partial duties.

Charon is a middleware client that sits between the beacon client and the validator client of each node within a distributed validator cluster and creates consensus on what to sign. Each of these nodes signs with a partial signature that, when aggregated, generates the full validator signature.

When a new duty is planned for a validator, the validator client retrieves the duty to be signed and sends it to the Charon client. The Charon client now waits for enough partial signatures from the rest of the nodes. After receiving enough partial signatures to meet the

threshold, Charon broadcasts the signed duty to the beacon node, which broadcasts it to the network.

The threshold of each cluster (minimum number of partial signatures to perform duties) depends on the number of nodes in the cluster:

- A 4-node cluster has a threshold of 3 partial signatures (1 node failure tolerated).
- A 7-node cluster has a threshold of 5 partial signatures (2 node failure tolerated).
- A 10-node cluster has a threshold of 7 partial signatures (3 node failure tolerated)

A cluster can stay active as long as more than 66% of its nodes send their partial signatures.

For a more in-depth explanation of the DVT, please refer here.

Motivation

Latency Experiment

In every slot, the assigned validators need to perform their planned duties, which are then included in blocks. A new slot occurs every 12 seconds and several things happen inside it:

- A new block proposal is broadcasted to the network.
- Validators vote (attest) on the validity of the new block.
- Committee aggregators receive all votes and create aggregated attestations, which will be broadcasted and used by the block proposer at the next slot.

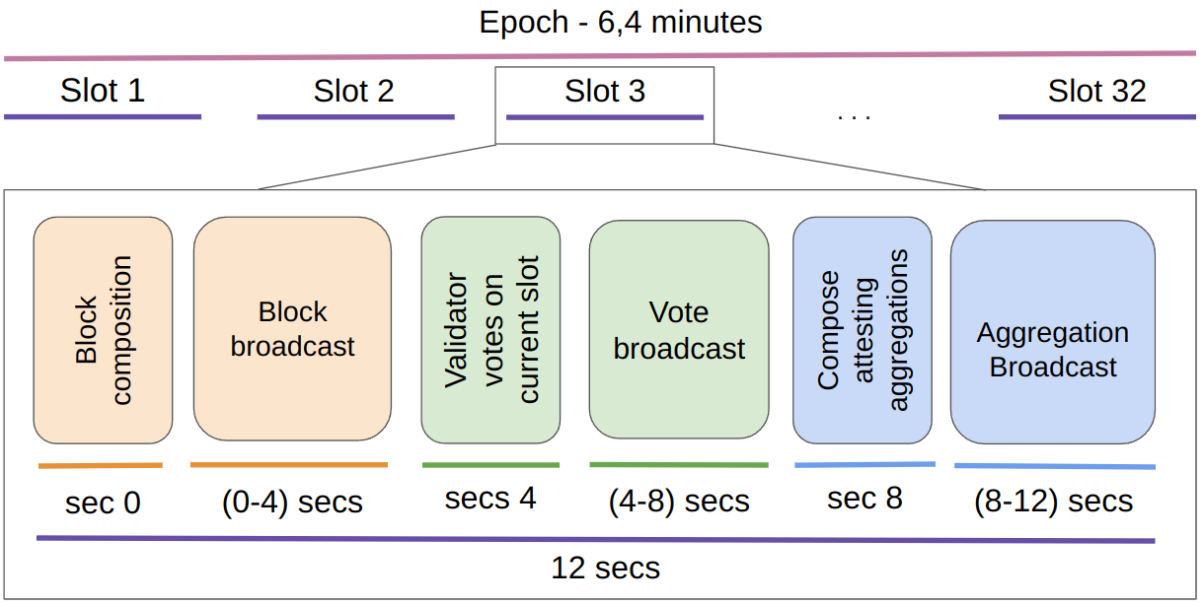


Figure1: Slot time division

In *Figure 1*, we can observe that each of the above events has a limited time window, which is not strict, but strongly recommended. Not respecting these timings may result in missed blocks or attestations not being included in the next block. Focusing on the broadcasting of validator votes (**attestations**), if the duty is not sent within a defined time window, it may not

be seen in time by the rest of the network and, therefore, not included in the next block. This would increase the inclusion delay of the attestation.

DVT adds a new step before validators broadcast their duties to the network, which is the aggregation of partial signatures. It is critical to ensure that this additional step does not affect the validator's performance by delaying the broadcast of signed duties.

While DVT clearly offers a novel, more resilient way of staking, it is unclear whether it can match the performance of classic (non-distributed) validators. In particular, for setups that include clusters of nodes distributed around the world and from different cloud providers, it is necessary to demonstrate that despite their latency, DVT can still provide the same level of performance as classic validators. This is the objective of this study.

Experiment Description

To evaluate Obol DVT, we performed a long multi-phased experiment. The experiment started at epoch **163000** (***Mar-18-2023 00:40:00 UTC***) and finished at epoch **173000** (***May-01-2023 11:20:00 UTC***) in the **Ethereum Prater network**. The Ethereum Prater network is a testnet and, as such, it may have dangling validators which may not be running. This means there are more missed blocks than in the Ethereum Mainnet network, resulting in delayed attestation inclusion and fewer rewards.

The experiment consisted of running three different clusters (one of each type 4, 7, 10) with 1000 validators attached to each of them. Therefore, the experiment involved running 3000 distributed validators.

The experiment's goal was to test if running distributed validators results in a similar performance as running non-distributed validators. To give the experiment more robustness and stress test the client for extreme cases based on locations, providers, and different beacon-client implementations, we tested the Charon setup with different geolocations and two different beacon clients.

Client Versions

During the experiment, a new version of the Charon client was released and all nodes were upgraded. At the same time, the Ethereum clients were also upgraded.

Initial Setup

- Charon v0.14.0
- Nethermind v1.17.1
- Lighthouse v3.5.1
- Teku v23.3.0

At Epoch 168511 we upgraded the versions to:

- Charon v0.15.0
- Nethermind v1.17.3
- Lighthouse v4.0.1
- Teku v23.3.1

Machine Specification

We deployed 21 machines distributed across the world to test the latency in edge cases. The 21 machines were distributed along 3 different clusters: a 4-node cluster, a 7-node cluster, and a 10-node cluster. To make a more robust and less biased experiment, two different service providers were used to deploy the machines: **OVH** and **DigitalOcean**.

OVH

CPU	RAM	Disk	IO Speed
8x Intel(R) Xeon(R) E-2274G CPU @ 4.00GHz	32GB	900GB	READ: bw=422MiB/s WRITE: bw=141MiB/s

Table1: OVH machine specification

Digital Ocean

CPU	RAM	Disk	IO Speed
4x Intel(R) Xeon(R) Platinum 8358 CPU @2.60GHz	32GB	600GB	READ: bw=234MiB/s WRITE: bw=78.3MiB/s

Table2: Digital Ocean machine specification

Non-Distributed Validator Machine

To compare the distributed validators with non-distributed validators, we also ran 5 nodes in a separate machine (one node per **main consensus client**: *Prysm, Lighthouse, Teku, Nimbus and Lodestar*). These 5 nodes were running on the same machine and each of them hosted 600 non-distributed validators, therefore 3000 non-distributed validators in total. The machine was located in Helsinki.

CPU	RAM	Disk	IO Speed
32x AMD Ryzen 9 5950X 16-Core Processor	128GB	10.5TB	READ: bw=512MiB/s WRITE: bw=511MiB/s

Table 3: Non-distributed validator machine

Machine Setup & Deployment (Locations)

4-Node Cluster

Provider	Location	Network	Services
OVH	Frankfurt	Goerli	Lighthouse + Lighthouse + Loki
OVH	Strasbourg	Goerli	Teku + Teku
OVH	Warsaw	Goerli	Teku + Teku + Loki
DigitalOcean	London	Goerli	Lighthouse + Lighthouse + Loki

Table 4: 4-node cluster locations and services

7-Node Cluster

Provider	Location	Network	Services
OVH	Frankfurt	Goerli	Teku + Teku + Loki
OVH	Strasbourg	Goerli	Teku + Teku
OVH	Warsaw	Goerli	Teku + Teku
DigitalOcean	Bangalore	Goerli	Lighthouse + Lighthouse + Loki
DigitalOcean	Frankfurt	Goerli	Lighthouse + Lighthouse + Loki
DigitalOcean	London	Goerli	Lighthouse + Lighthouse
DigitalOcean	Singapore	Goerli	Lighthouse + Lighthouse

Table 5: 7-node cluster

10-Node Cluster

Provider	Location	Network	Services
DigitalOcean	Bangalore	Goerli	Lighthouse + Lighthouse + Loki
DigitalOcean	Toronto	Goerli	Lighthouse + Lighthouse
DigitalOcean	Frankfurt	Goerli	Teku + Teku + Loki
DigitalOcean	London	Goerli	Lighthouse + Lighthouse + Loki
DigitalOcean	Singapore	Goerli	Lighthouse + Lighthouse
DigitalOcean	New York	Goerli	Lighthouse + Lighthouse
OVH	Beauharnois	Goerli	Teku + Teku + Loki
OVH	Frankfurt	Goerli	Teku + Teku
OVH	Strasbourg	Goerli	Teku + Teku
OVH	Warsaw	Goerli	Teku + Teku + Loki

Table 6: 10-node cluster

Methodology

Services Deployment

Each machine described above hosted six services:

- Execution client (always **Nethermind**)
- Beacon Node (either **Lighthouse** or **Teku**)
- Charon client
- Validator Client (either **Lighthouse** or **Teku,** respectively)
- Prometheus service
- Node Exporter service
- Loki (optional): a log aggregation system that connects to a given Grafana.

Please find here a more in-depth description of how these services were deployed.

Validator Creation

To keep up with the Charon cluster deployment guidelines, the validator creation was carried out through the Charon client. Please find the guide here.

The process consists of performing a *Distributed Key Generation* (**DKG**) ceremony among the peers that will form each of the clusters.

Before this ceremony, we configured each of the nodes' ENRs and defined one of them as the host (one in each cluster), which collected all the ENRs into a single file. During the DKG, all peers in the cluster connect to each other and start generating the distributed validator partial keys shares.

After this process, at each machine, we had a Charon folder containing the cluster definition and the validator keys shares. To activate all the validators in an automated way we followed this guide.

Measuring & Data Collection

To analyze the performance of the nodes in the experiment, we collected data through two different services:

- Prometheus
- GotEth

Prometheus

Both the execution and consensus clients expose several metrics that can be scraped with Prometheus. The Charon client does the same. Apart from this, we also wanted to measure the machine resource usage, which can be achieved with the Prometheus Node Exporter module. This module enables anyone to measure and monitor the machine resource consumption regarding the network, memory, CPU, and disk, among many others. With these metrics, we are able to compare which resources are more used.

GotEth

As explained in other publications, measuring a validator's rewards results in a good analysis of its performance. As this experiment consisted of running distributed validators, it is interesting to analyze if the achieved rewards are similar to a non-distributed validator (what we know as a common validator until now).

For this, we used GotEth, an open-source tool that can retrieve the achieved and max reward from any validator in the Ethereum network at each epoch.

Data Analysis

As the clients' versions were upgraded during the experiment, we may divide part of the analysis into two subsections, identified by the Charon version (**v0.14.0** for epochs before **168511** and **v0.15.0** for epochs after **168511**).

Latency Between Peers

As mentioned in the experiment description, it is important that the beacon node broadcasts the signed duties in the defined time window. Therefore, it must be ensured that the additional complexity of the distributed validator does not add any delays.

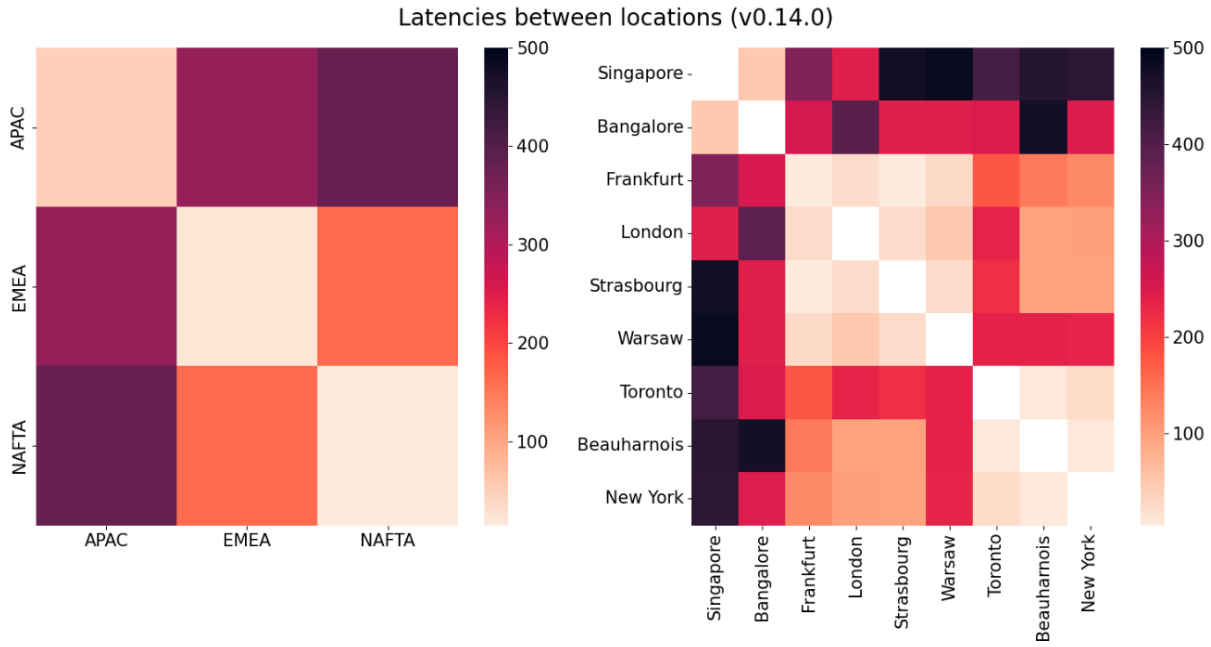


Figure 2: Latency between locations in v0.14.0

In *Figure 2* it is shown the latency between the peers by location and region. We can observe that peers that are closer to each other location-wise share a lower latency, as expected.

After upgrading to version **v0.15.0** we can observe that the latency in the worst-case scenarios (very far distance between peers) has improved.

For example, the latency between Strasbourg and Singapore has decreased from 470ms in **v0.14.0** to 300ms in **v0.15.0**, which represents a 37% improvement.

Similarly, the latency between Beauharnois and Singapore has decreased from 448 in **v0.14.0** to 245ms in **v0.15.0**, which represents a 46% improvement.

In general, we see the same pattern in both versions.

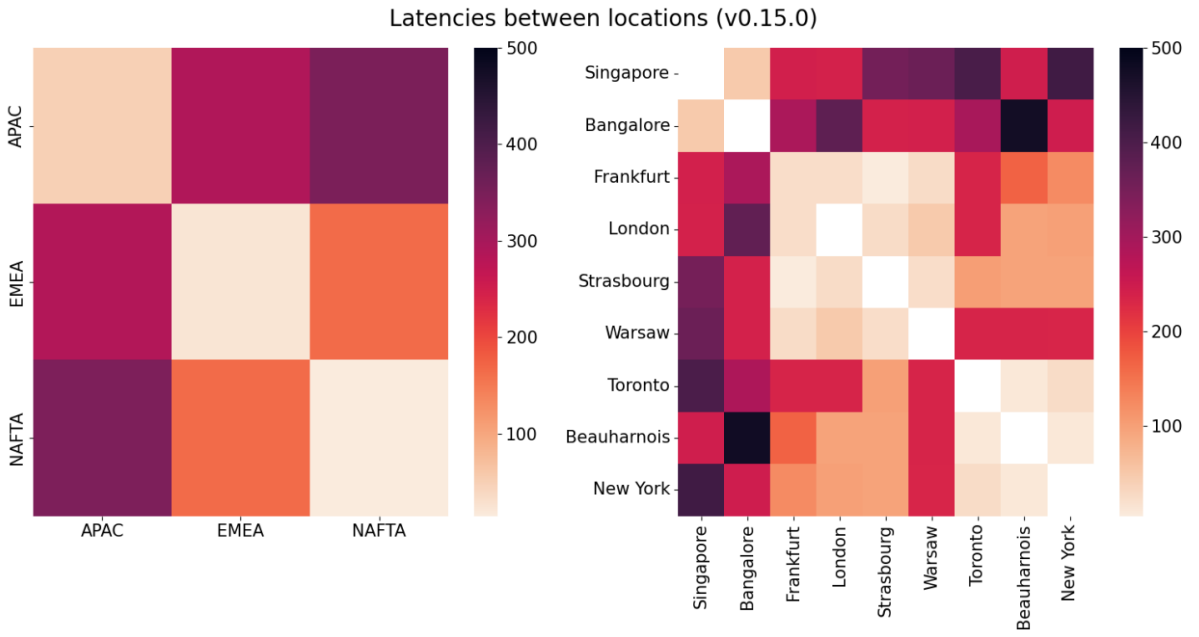


Figure 3: Latency between locations in v0.15.0

Maximum Extractable Reward

In *Figure 4* it is shown the achieved aggregated reward by each group of validators against the maximum aggregated compensation these could have obtained, as well as the average grouping by distributed and non-distributed validators.

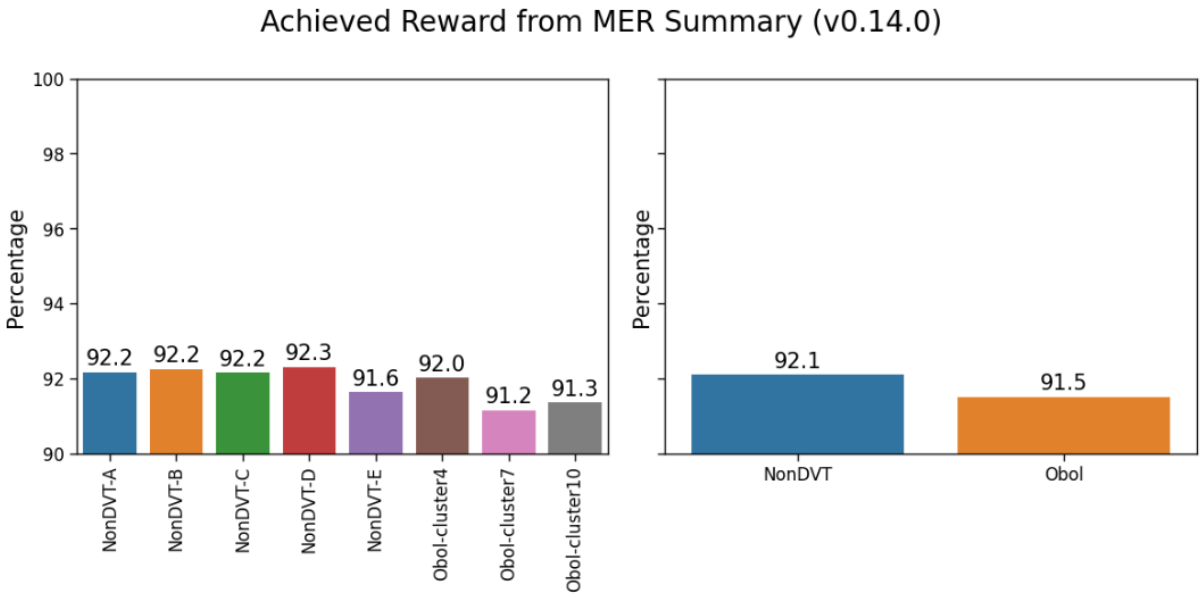


Figure 4: Achieved reward from Maximum Extractable Reward in v0.14.0

We can observe that in both versions the average extracted reward of distributed validators is **between 91% and 92%**, not even 1% below the achieved by the non-distributed validators. As mentioned before, the experiment runs on the Prater network, which involves an expected lower performance than running on the Mainnet network. Looking at the data

we can observe a similar performance for both, distributed and non-distributed validators. Note that with Charon v0.15, the difference between DVT and non-DVT is only **0.4%**.

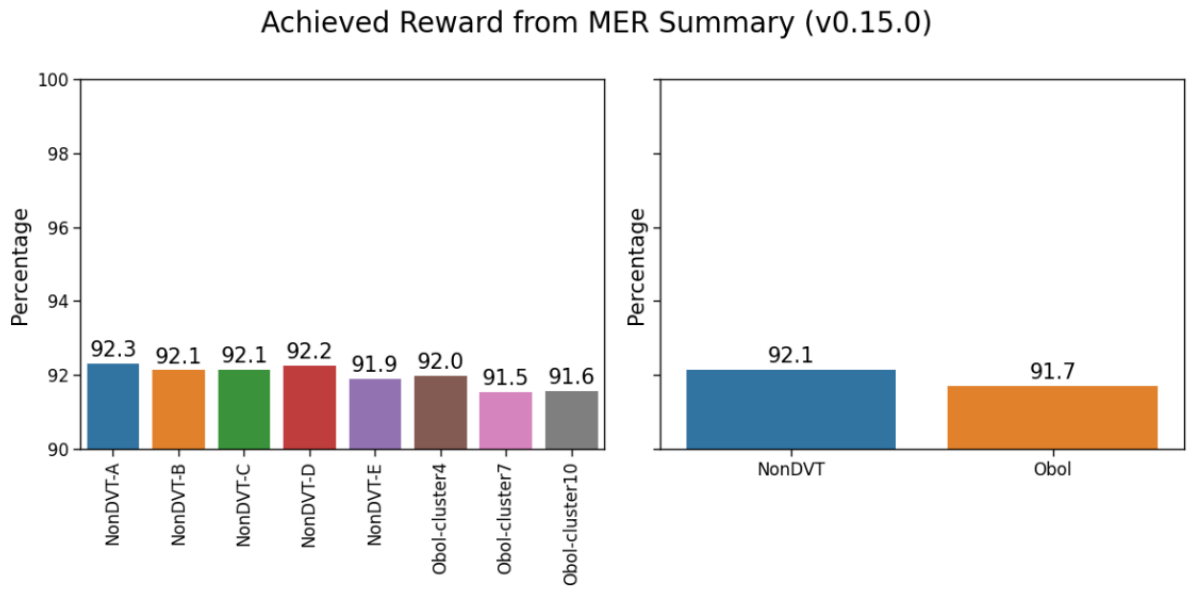


Figure 5: Achieved reward from Maximum Extractable Reward in v0.15.0

Missed Attestations

At every epoch, validators are assigned to attest to the validity of one block. This means that they must submit one attestation per epoch. Taking into account *Figure 1*, the node must broadcast the assigned attestation duties (depending on the validators running on this node) after receiving the new block (after the second 4th), and this duty must have been received by the beacon committee aggregators by the second 8th. After that, the committee aggregators broadcast the aggregated attestation to the network, which would be included in the next block. Delaying this last broadcast could risk the inclusion of the aggregated attestation in the next block.

If the use of **DVT** added any delay during this process (sending signed attestations) it could happen that the node broadcasts the signed attestations too late, compromising the inclusion of the latter in the aggregation. Thus, the inclusion delay would increase, and, depending on how much it is increased, some flags could be missed, compromising the achieved reward. See here to understand better how attestation flags are computed.

Figure 6 shows the percentage of missed attestation flags out of the planned duties (one per validator per epoch), as well as the average per group: distributed vs non-distributed validators.

We can observe that in all cases the most failed flag is the head flag, followed by the source and finally the target, as expected. The difference in all cases between distributed and non-distributed validators is less than **1%**, which shows similar performance in terms of attestation duties.

In *Figure 7,* we can observe the same performance as with v0.14.0, which means no major changes were applied regarding the attestation duties during the upgrade.

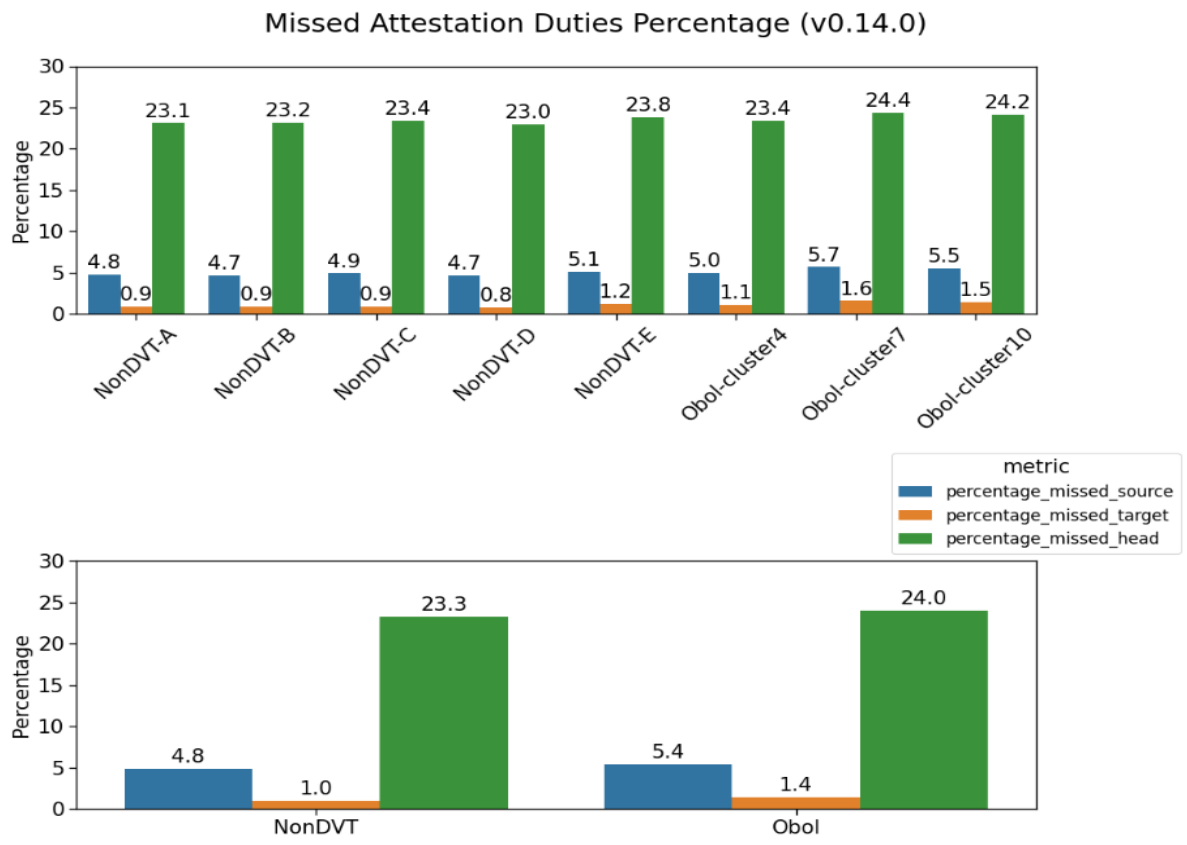


Figure 6: Missed attestation duties in v0.14.0

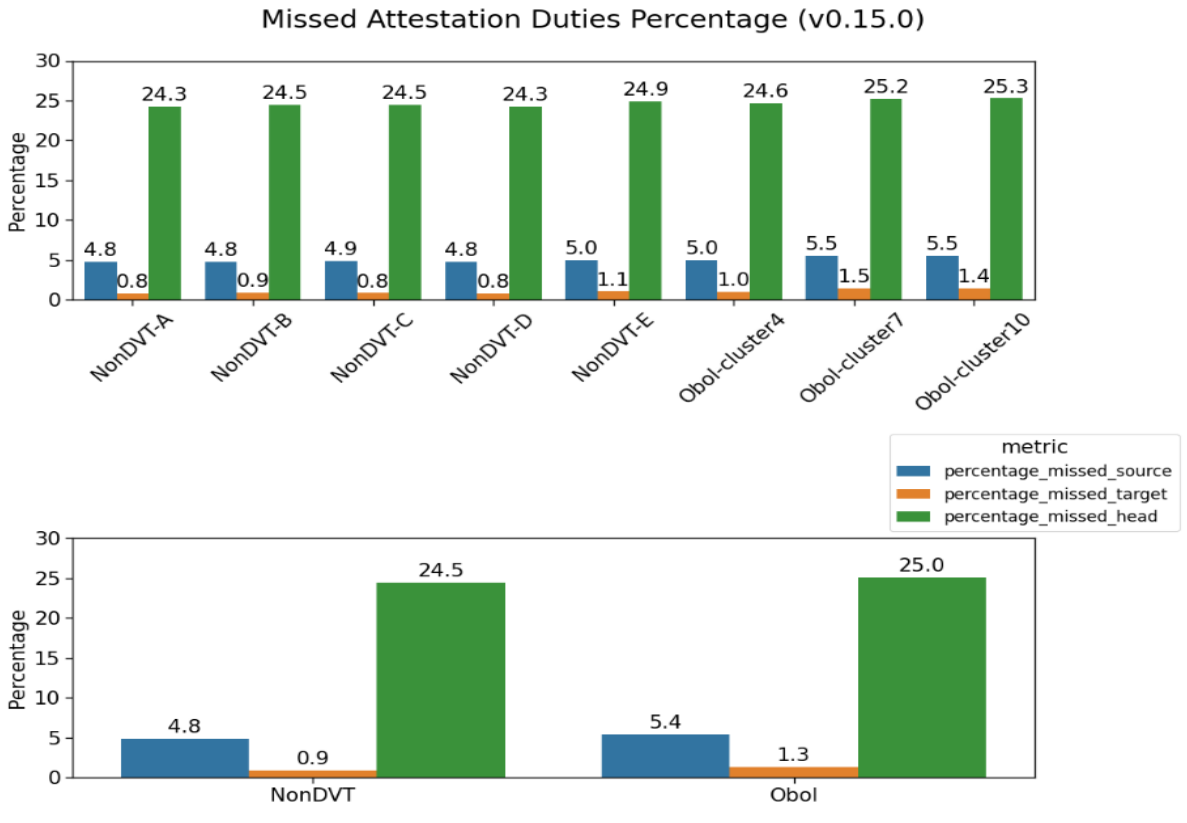


Figure 7: Missed attestation duties in v0.15.0

Block Proposals

At every slot, one random validator is chosen to propose a new block, which will include attestations broadcasted in the previous slots. The block proposal is the first task inside a new slot. This means that if the block proposal is received by the rest of the network after the estimated (4 seconds), there is a risk the rest of the validators vote on a missing block. It is therefore very important to ensure that DVT does not add any delays to submitting the new block.

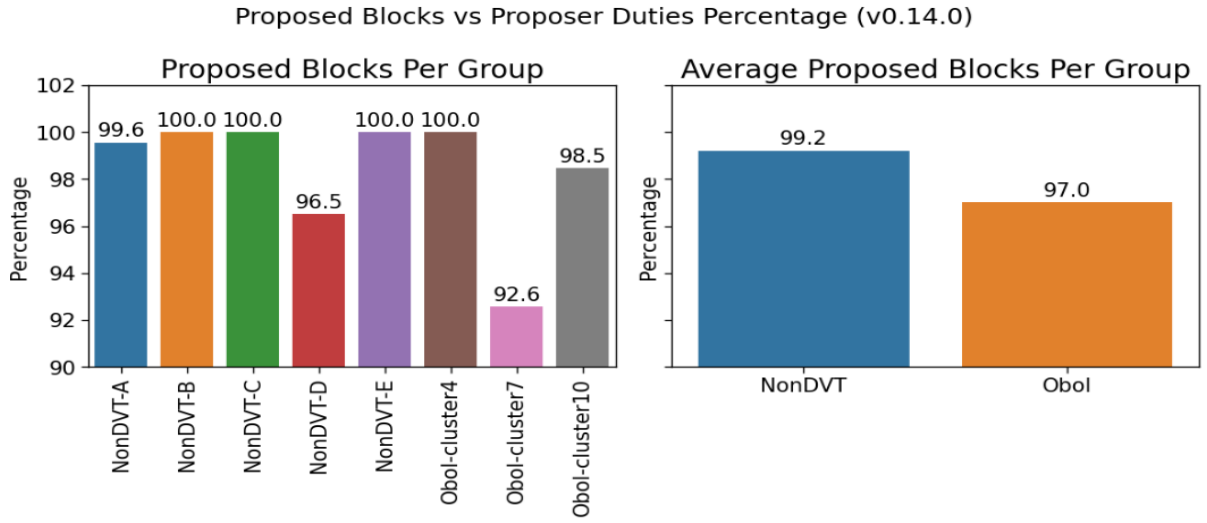


Figure 8: Block Proposal Duties in v0.14.0

In *Figure 8*, we can observe that *cluster7* registers the worst proposed blocks ratio, with 92.6% of proposed blocks. Comparing distributed and non-distributed validators, we can observe that there is a difference of 2% in proposed blocks. The missing blocks are concentrated in the biggest clusters (*cluster7* and *cluster10*), while the validators in *cluster4* did not fail to propose any block.

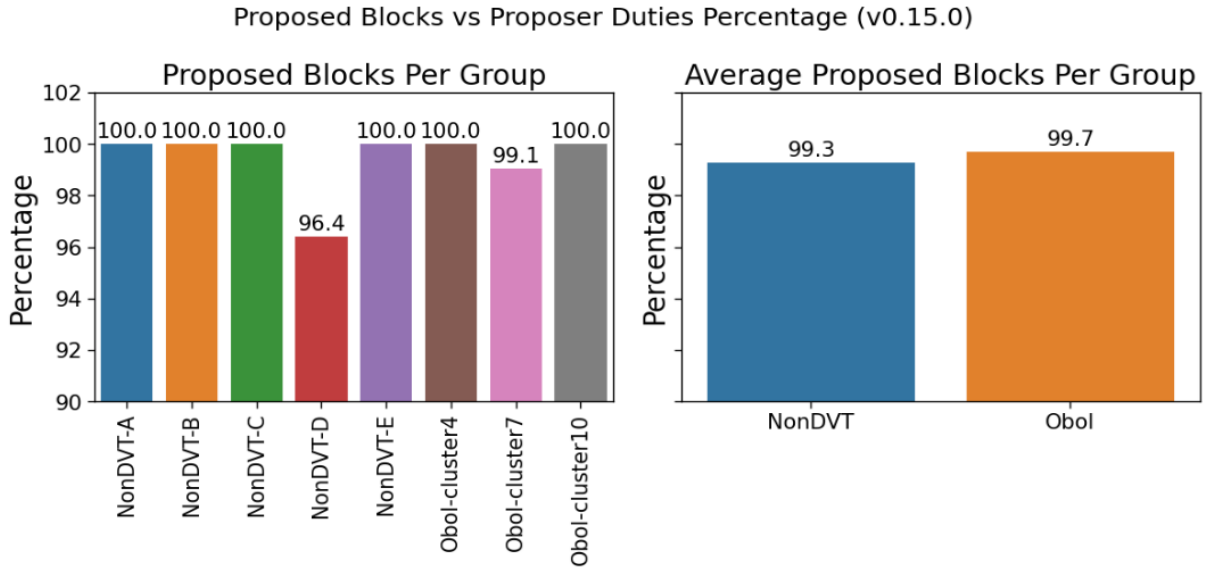


Figure 9: Block Proposal Duties in v0.15.0

In *Figure 9*, we can observe the same data as in *Figure 8* but after upgrading the *Charon* client to **v0.15.0**. Regarding the distributed validators, we now observe a big improvement in the ratio of proposed blocks, getting closer to the 100% and even getting over the non-distributed validators. In general, we see an improvement of **2.7%** in proposed blocks in distributed validators. All missed blocks now concentrate on the validators in *cluster7*, while cluster4 and *cluster10* proposed all its scheduled blocks.

Performance Timeline

As mentioned before, the obtained rewards by a validator are a very good representation of the node performance. This is why we have compared the obtained rewards against the maximum extractable reward at each epoch to detect any downtimes or underperformance in any of the validators.

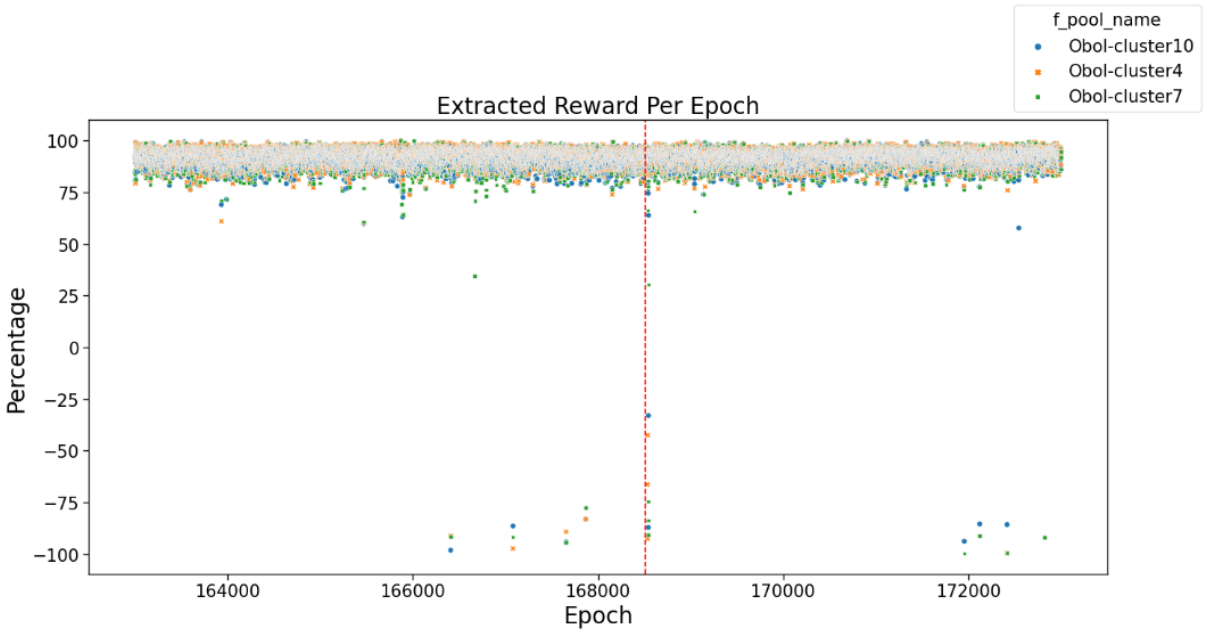


Figure 10: Achieved Rewards Timeline

In *Figure 10*, we can observe that the validator usually obtained an average reward of **75% - 100%** of the maximum extractable reward, concentrating around the 90%, which is expected in the **Prater** network. However, we see some drops even to negative values, which is probably due to missing attestations or sync committee duties.

Some execution clients started having issues with corrupted databases so we had to restart some of the nodes, which could have caused the first drops between epoch **166000** and epoch **168000**.

During epoch **168511** we upgraded all nodes, as explained before. Since all clusters have a threshold, we tried to maintain a minimum number of nodes running at all times, but nonetheless, during a few epochs the cluster underperformed. This is why we see several reward drops near the red dashed line (which marks the epoch when we upgraded the clients). Note that non-distributed validators running on a node would also be perturbed by software updates, and in some cases to a greater extent.

We also see some drops around epoch 172000, in which we observed some beacon nodes receiving blocks later than expected. More specifically, during these slots, the Charon client sent the attestation to the beacon node by the second 5th of the slot, at which the block had not arrived yet. This explains why there are some small drops in rewards on that epoch.

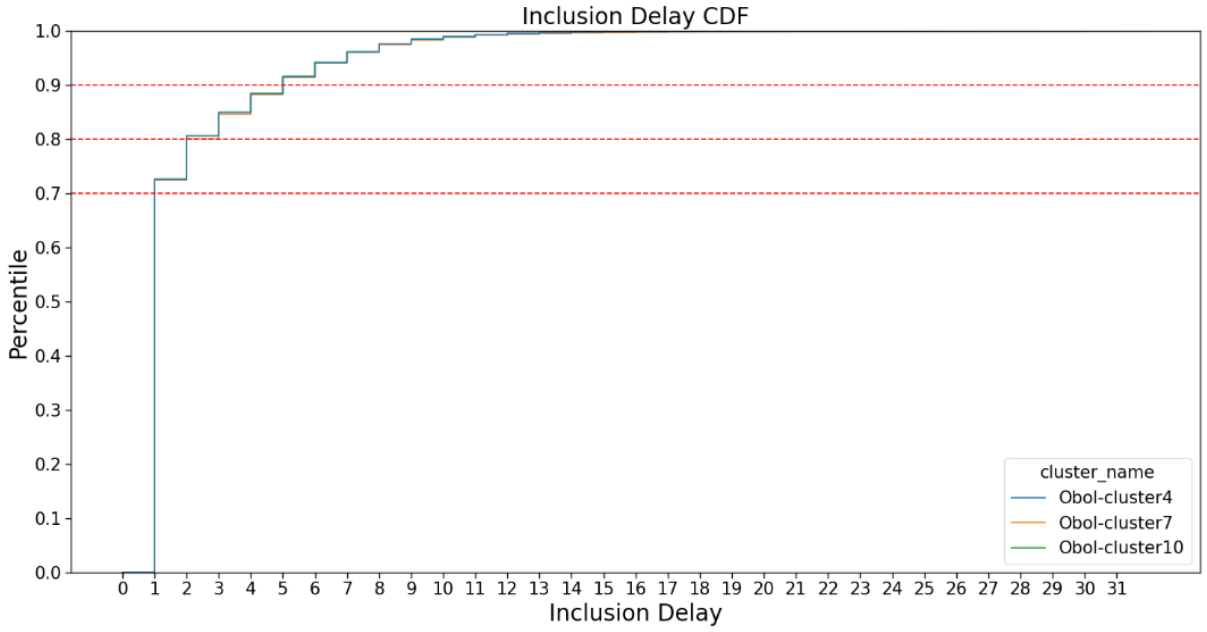


Figure 11: Inclusion delay by cluster

Similar to the achieved reward, the inclusion delay is a good representation of the validator's performance. Taking into account *Figure 1*, the more time a validator takes to send its signed attestation, the higher the probability of not getting the attestation included in the next block. As a result, less attestation rewards would be obtained.

In *Figure 11*, we can observe that, most of the time (**over 70%**) the inclusion delay is 1, which is the ideal scenario. However, we can see that there are some cases of inclusion delay between 2 and 5 as well (**around 20%** of the time), and the rest over 5 (**less than 10%** of the time). We can observe that the three clusters show the same behavior and percentiles. Therefore, even though some clusters have temporarily underperformed, in general, all of them had an overall similar performance.

Conclusions

We have analyzed Obol DVT performance from a resource and rewards perspective. In terms of latency, we have noticed that, as expected, clusters perform better when the nodes are located close to each other in nearby regions, regardless of the region. For example, a cluster running with 4 nodes in Singapore, Tokyo, Sydney and Bangalore should have a similar latency model to another cluster with 4 nodes located in Warsaw, Frankfurt, London, and Strasbourg. However, even large clusters with nodes dispersed all over the world manage to get a good performance, with a negligible difference in comparison to non-distributed validators.

From a rewards and consensus duties perspective, we can observe that the more nodes located in extreme locations, the more missed duties there are on average. This is why cluster-4 shows the best performance in this experiment, in which no remote nodes are included. Cluster-7 presents the lowest performance in the experiment, as it includes 2 nodes in remote areas, and when they underperform, it gives the rest of the nodes no possibility to fail, as the threshold is 5 nodes.

Finally, cluster-10 places in between the other 2 clusters, as it includes 2 remote nodes but it still enables another node to fail before hitting the threshold of 7 partial signatures.

Comparing distributed and non-distributed validators there is a very small difference of not even 1% in achieved rewards. Similarly, we see the same behavior when it comes to attestation flags.

Taking into account the upgrade in the Charon client we must mention the improvement in the block proposal duties ratio, which even surpassed the performance of non-distributed validators.

Additional Notes

Combine Keys Feature

After the experiment had finished it was our goal to combine all keys so they could be reused for other testing purposes. Charon contains a feature to combine splitted keys into a single key. We tried merging the 3k split keys, but we were not able to do so as the feature had a bug that would not allow combining more than 10 keys.

After reporting this issue to the Obol team, the bug was solved and the algorithm was improved so that the process would take 1 minute (before the patch it would take 10 minutes). Finally, the changes were merged into the main branch.

Migration of Nodes In a Cluster

After finishing the experiment, several machines were shut down and some redistribution was needed. Specifically, the 4-node cluster kept running, but one of the nodes was replaced, this required us to migrate the Charon node from the old machine to the new machine. To do this, we did not find any guidelines or walkthroughs to do this process, so we migrated it based on our knowledge of how the Charon client works.

We copied the .charon folder from the machine that was getting shut down and transferred it to the new machine. The new machine was running other validators, so we erased any volumes of data folders from the Validator Client and imported the validators again. After that, the new machine ran a node in the 4-node cluster.

Scaling Performance Test

Additionally, another experiment proposed by the Obol Labs team was performed on DVT. This experiment consisted of scaling the number of validators that are run on a cluster, with

the main goal of analyzing how the Charon client handles such a high load of duties to perform. As the main analysis included the activation of 3000 validators, scaling took the following steps:

- Run a 4-node cluster with 1000 keys for 1k epochs
- Run a 4-node cluster with 2000 keys for 1k epochs.
- Run a 4-node cluster with 3000 keys for 1k epochs.

The same machines were reused for all steps, but cleanup was needed when changing the number of validators. This is because it is not possible to add key shares to an existing cluster. Instead, it is necessary to run the DKG process to generate all the key shares that will be used in the cluster.

As the number of validators increased with every step of the experiment, we analyzed the achieved rewards by the whole cluster, normalized by the number of validators that were performing duties during that period of time. Surprisingly we saw an increase in achieved rewards when we increased the number of validators.

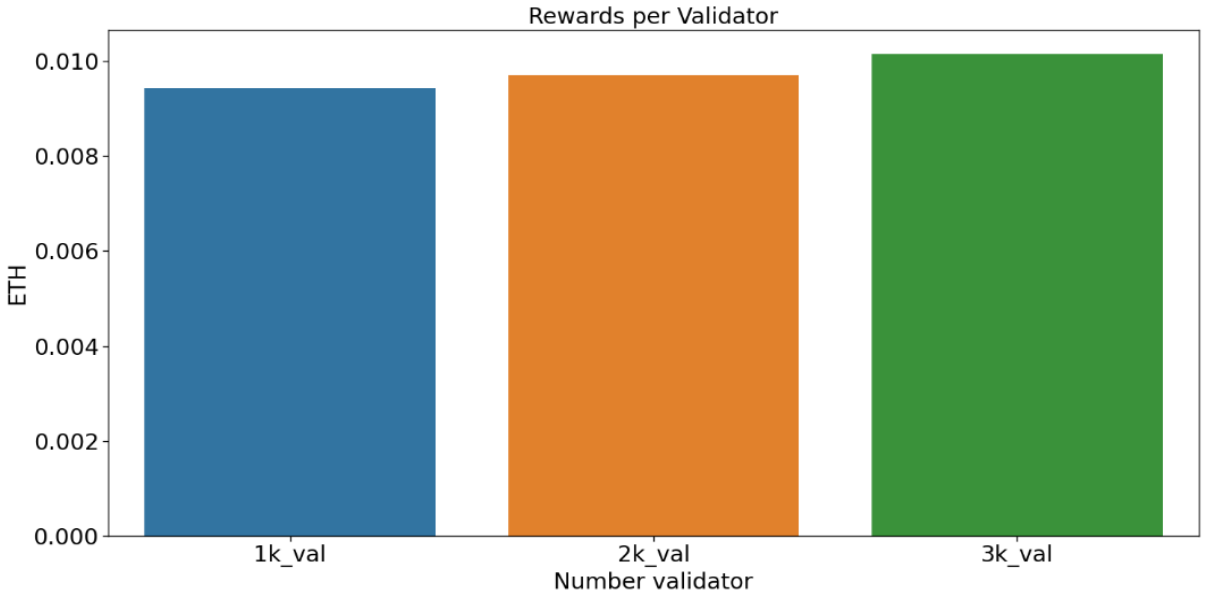


Figure 12: Achieved rewards per validator

We can see that after 1000 epochs, the 4-node cluster obtained:

- 0.009424 ETH per validator when composed of 1000 validators
- 0.009702 ETH per validator when composed of 2000 validators
- 0.010146 ETH per validator when composed of 3000 validators

Even though more rewards are achieved with a higher number of validators, a higher load is also expected, as more duties need to be performed. For every duty, the beacon node creates a duty to be signed by the validator client, who sends its signature to the beacon node. The more validators, the more duties, and therefore, more load is expected.

Figure 13, presents the beacon node score of the peers that form the 4-node cluster at the three steps (each with a different number of validators). We can observe that, the more validators in the cluster, the lower the beacon node score.

The beacon node score is calculated based on the Charon Prometheus metrics (using the number of errors and the latency of the queries).

On average, the cluster obtained a beacon node score of *81.5%* with 1000 validators, *77.3%* with 2000 validators (around 4% worse), and 69.9% with 3000 validators (more than 10% worse). Despite the decreasing beacon node score, the amount of rewards increases as the number of validators increases. This just shows that the beacon node score is still a work in progress and can be improved in the future.

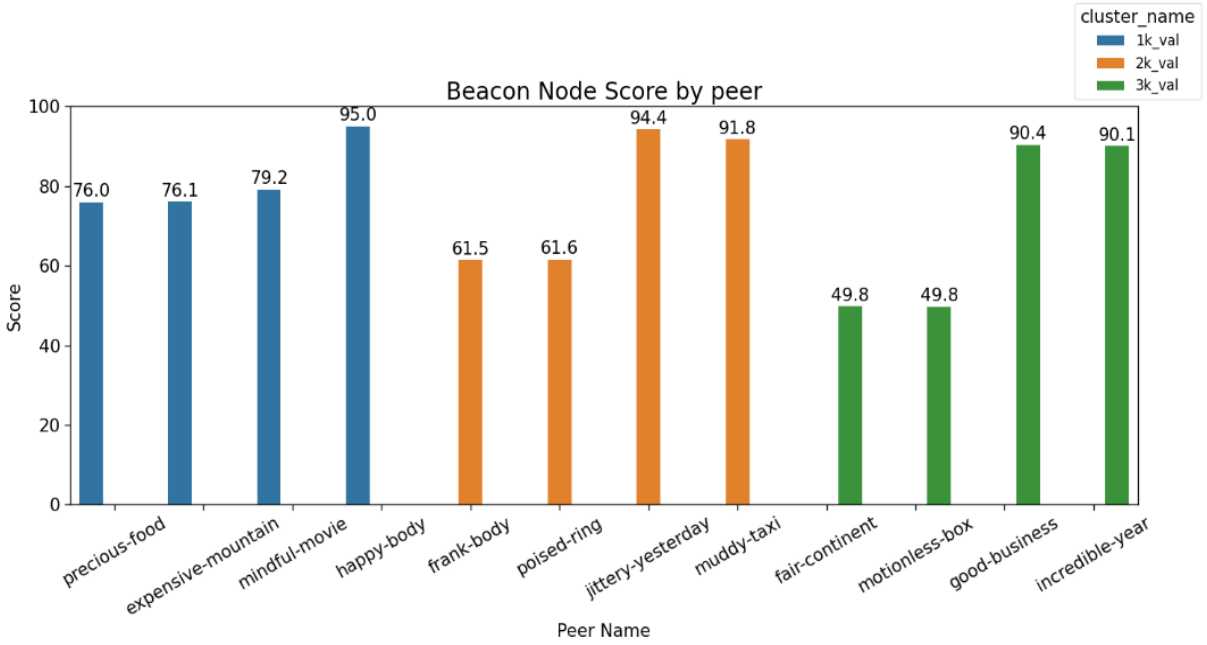


Figure 13: Beacon node score per peer and step