Obol DVT Performance Analysis

Study performed by MigaLabs

Table of Content

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.

Background & Motivation

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.

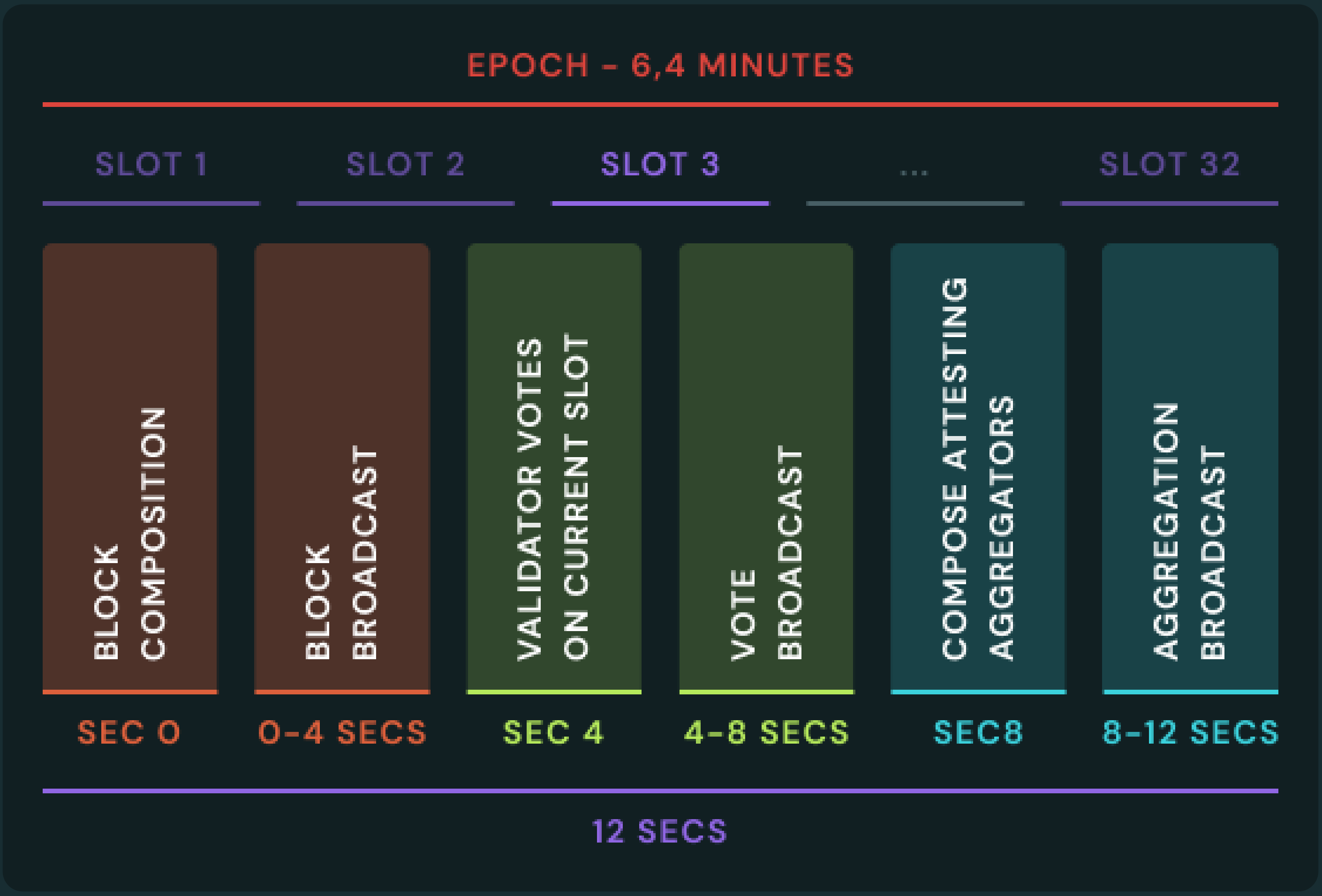

Figure 1: Slot time division

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 Setup

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 more chain reorgs. These conditions are harder than Mainnet, which stresses even further the software.

The experiment consisted of running three different clusters (one of each type **4**, **7**, **10 nodes**) 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, as well as two different cloud providers, all combined in a DVT cluster.

Client Versions

During the experiment, we tested two different versions of the Charon client: **v0.14** and **v0.15**. At the same time that all nodes were upgraded from v0.14 to v0.15, the Ethereum clients were also upgraded, both at the execution layer and at the consensus layer.

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

Distributed Validator Machines

We deployed 21 machines (4+7+10) 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. Different providers offer different hardware, which was also part of the §purpose of this experiment. The specific hardware resources in the two providers were as follows

OVH

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

Table 1: OVH machine specification

Digital Ocean

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

Table 2: 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.

CPU	RAM	Disc	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

CPU	Location	Services	Node Name
OVH	Frankfurt	Lighthouse + Lighthouse + Loki	happy-body
OVH	Strasbourg	Teku + Teku	precious-food
OVH	Warsaw	Teku + Teku + Loki	expensive-mountain
DigitalOcean	London	Lighthouse + Lighthouse + Loki	mindful-movie

Table 4: 4-node cluster locations and services

7-Node Cluster

CPU	Location	Services	Network
OVH	Frankfurt	Teku + Teku	unusable-state
OVH	Strasbourg	Teku + Teku	selfish-rule
OVH	Warsaw	Teku + Teku	determined-party
DigitalOcean	Bangalore	Lighthouse + Lighthouse	enthusiastic-area
DigitalOcean	Frankfurt	Lighthouse + Lighthouse	affectionate-day
DigitalOcean	London	Lighthouse + Lighthouse	jolly-life
DigitalOcean	Singapore	Lighthouse + Lighthouse	cloudy-flowers

Table 5: 7-node cluster

10-Node Cluster

CPU	Location	Services	Network
DigitalOcean	Bangalore	Lighthouse + Lighthouse	dangerous-mobile
DigitalOcean	Toronto	Lighthouse + Lighthouse	clear-fish
DigitalOcean	Frankfurt	Teku + Teku	beautiful-word
DigitalOcean	London	Lighthouse + Lighthouse	amazing-tea
DigitalOcean	Singapore	Lighthouse + Lighthouse	plain-news
DigitalOcean	New York	Lighthouse + Lighthouse	tough-city
OVH	Beauharnois	Teku + Teku	fine-adult
OVH	Frankfurt	Teku + Teku	alert-waterfall
OVH	Strasbourg	Teku + Teku	delightful-dates
OVH	Warsaw	Teku + Teku	powerful-road§

Table 6: 10-node cluster

Methodology Description

Services Deployment

Each machine described above hosted six services:

- Execution client **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 this experiment consists of running distributed validators and measuring their performance, 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). At MigaLabs we have developed a tool (GotEth) capable of measuring and tracking a validator's rewards results obtained during a long period of time, as explained in previous publications. More specifically, GotEth is an open-source tool that can retrieve the achieved and max reward from any validator in the Ethereum network at each epoch.

Performance Analysis

In this section we analyze the data obtained during the experiment described in the sections above. First, we start by analyzing the hardware resource consumption of the nodes running Obol DVT. Then, we analyze the latency between nodes running in the same cluster as this is a critical aspect to get good reward performance. We follow the study by looking into attestation submissions and block proposals.

Then, we compare the total rewards obtained by DVT validators vs non-distributed validators. Finally we perform a scaling test in which we increase the number of validators up to 3000 in one single DVT cluster.

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**).

Hardware Resource

During the whole experiment we measured the resource consumption of all the nodes running DVT validators to get an idea of the workload of DVT technology on the machines.

Among the collected metrics, we focused on:

- CPU consumption
- Memory consumption
- Network sent bytes ratio

By comparing these metrics from the different machines and clusters we aim to identify any weak points or underperforming peers, due to the different clients, machines, and locations in the experiment.

CPU Consumption

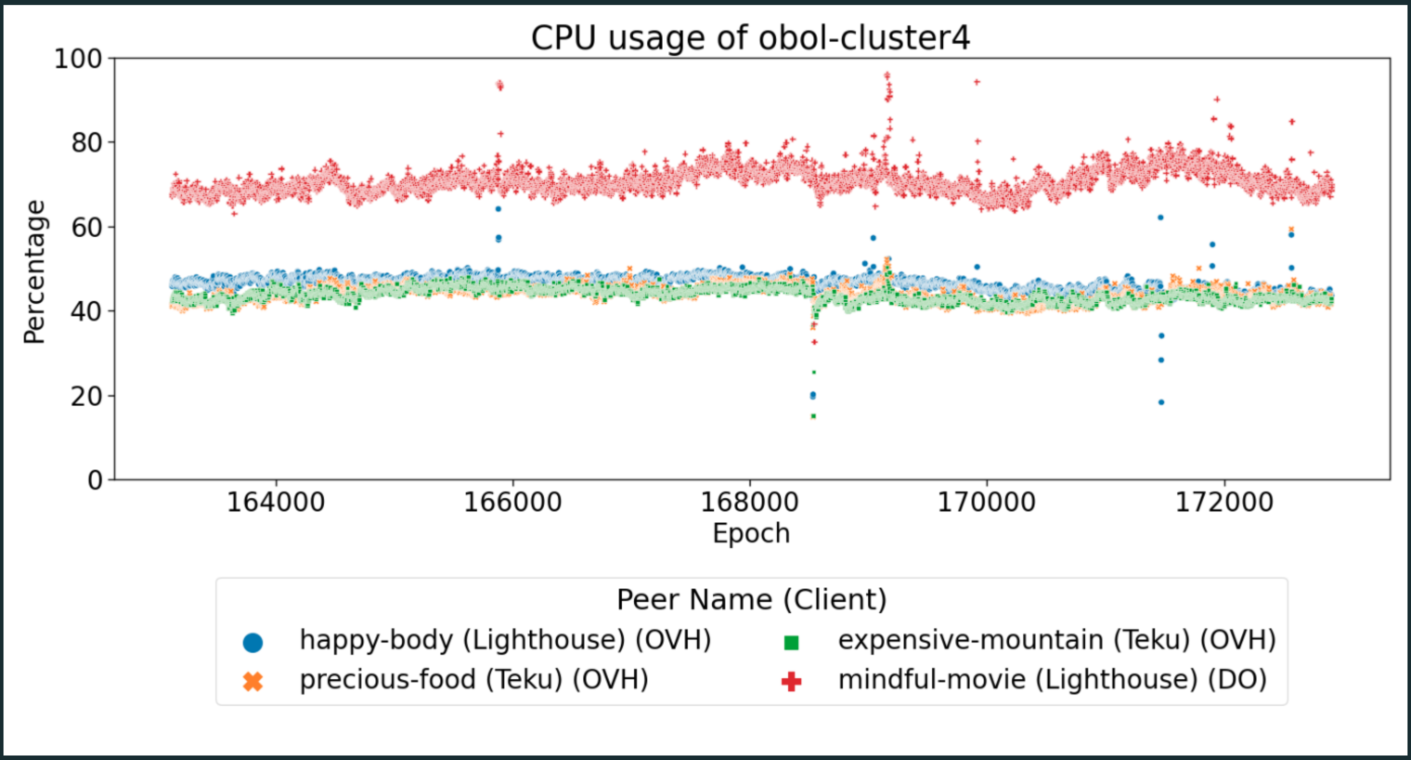

Figure 2: CPU consumption in Obol-cluster4

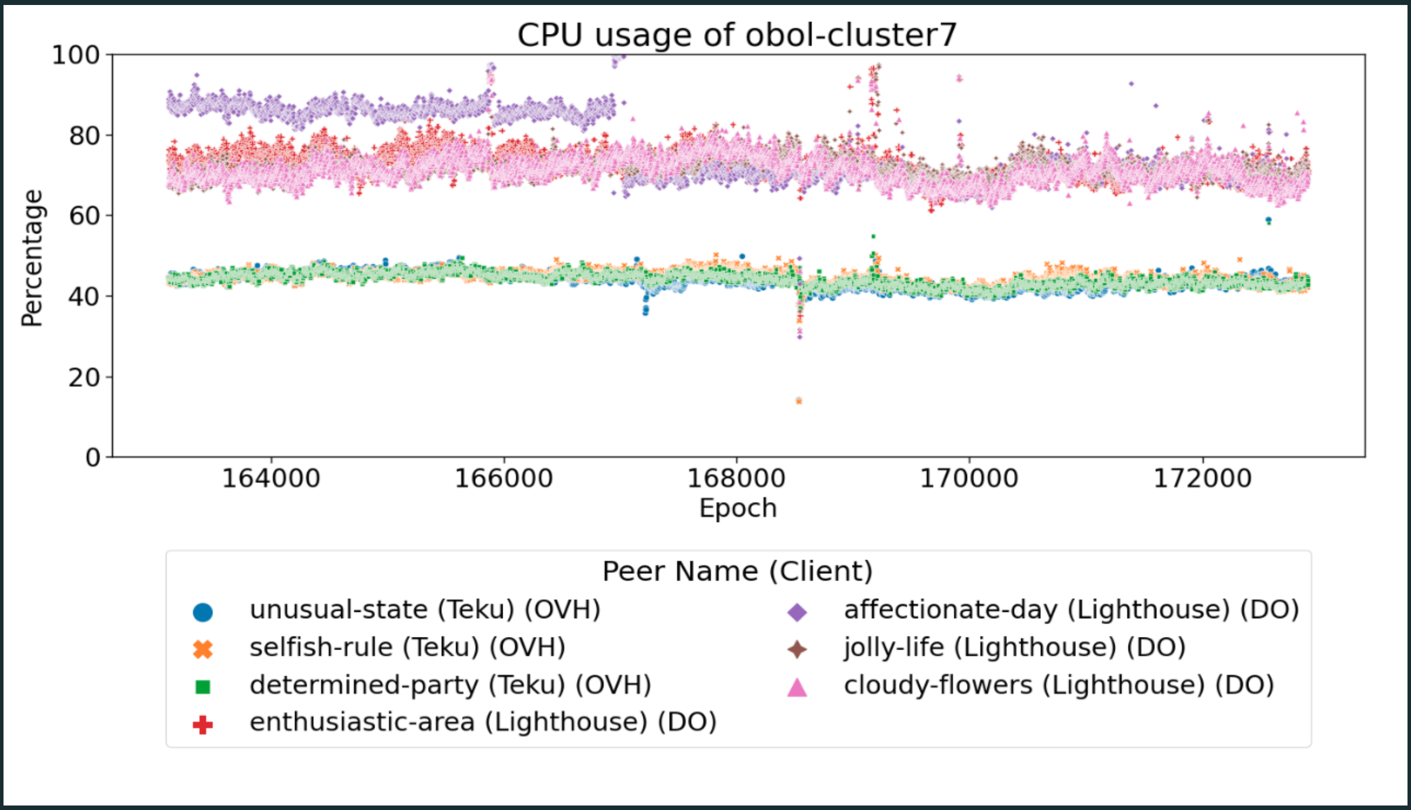

Figure 3: CPU consumption in Obol-cluster7

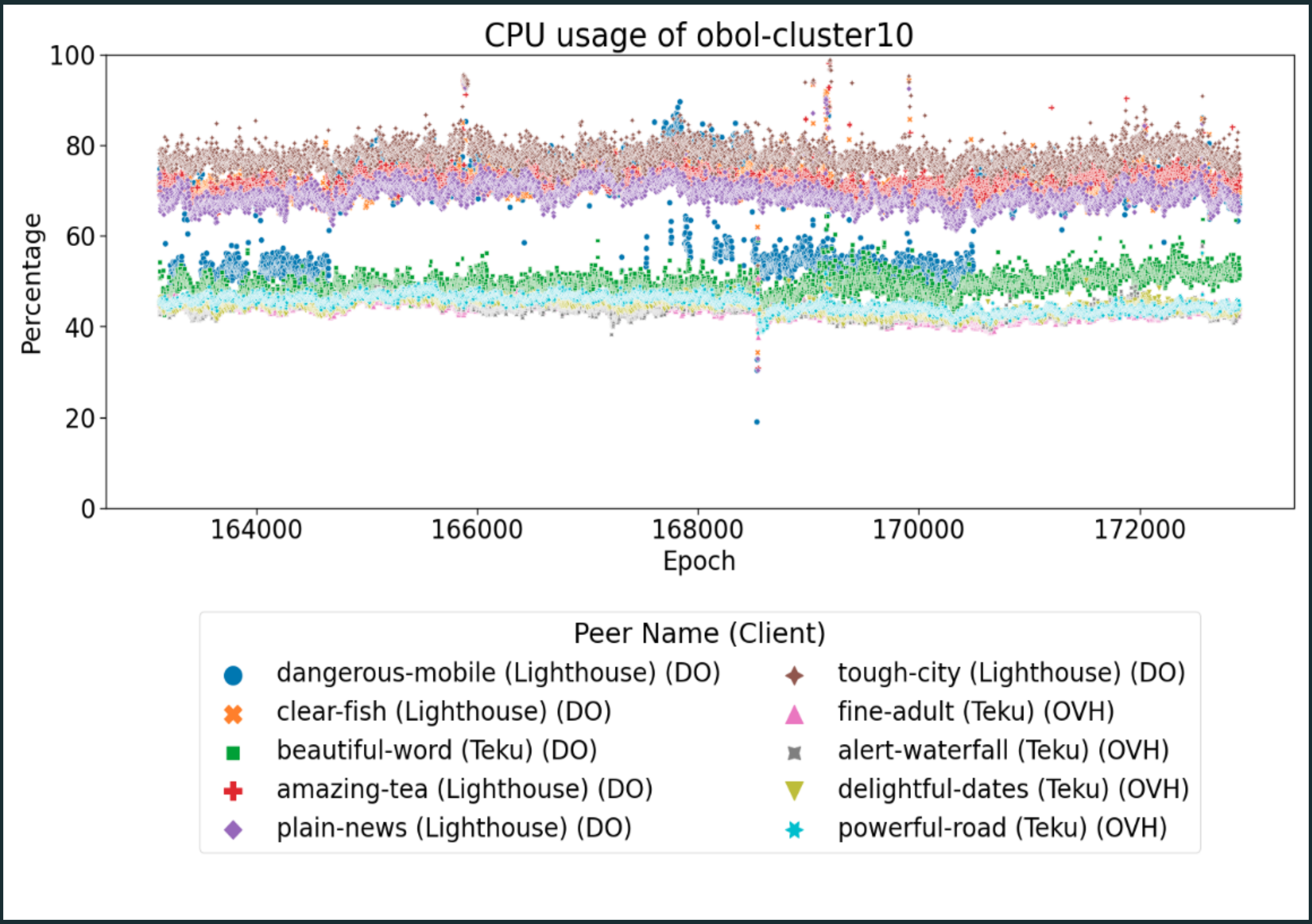

Figure 4: CPU consumption in Obol-cluster10

If we observe the CPU consumption of the three different clusters, we can clearly spot two different groups. While some peers maintain a 40% - 60% CPU consumption, others keep themselves at 60 - 80% CPU consumption. The latter are the peers that were hosted in the CPU consumption. The latter are the peers that were hosted in the DigitalOcean machines, which had less CPU capacity and, thus, had to use more CPU resources. Other than that, there are no major outliers or differences between different locations or CL clients.

Memory Consumption

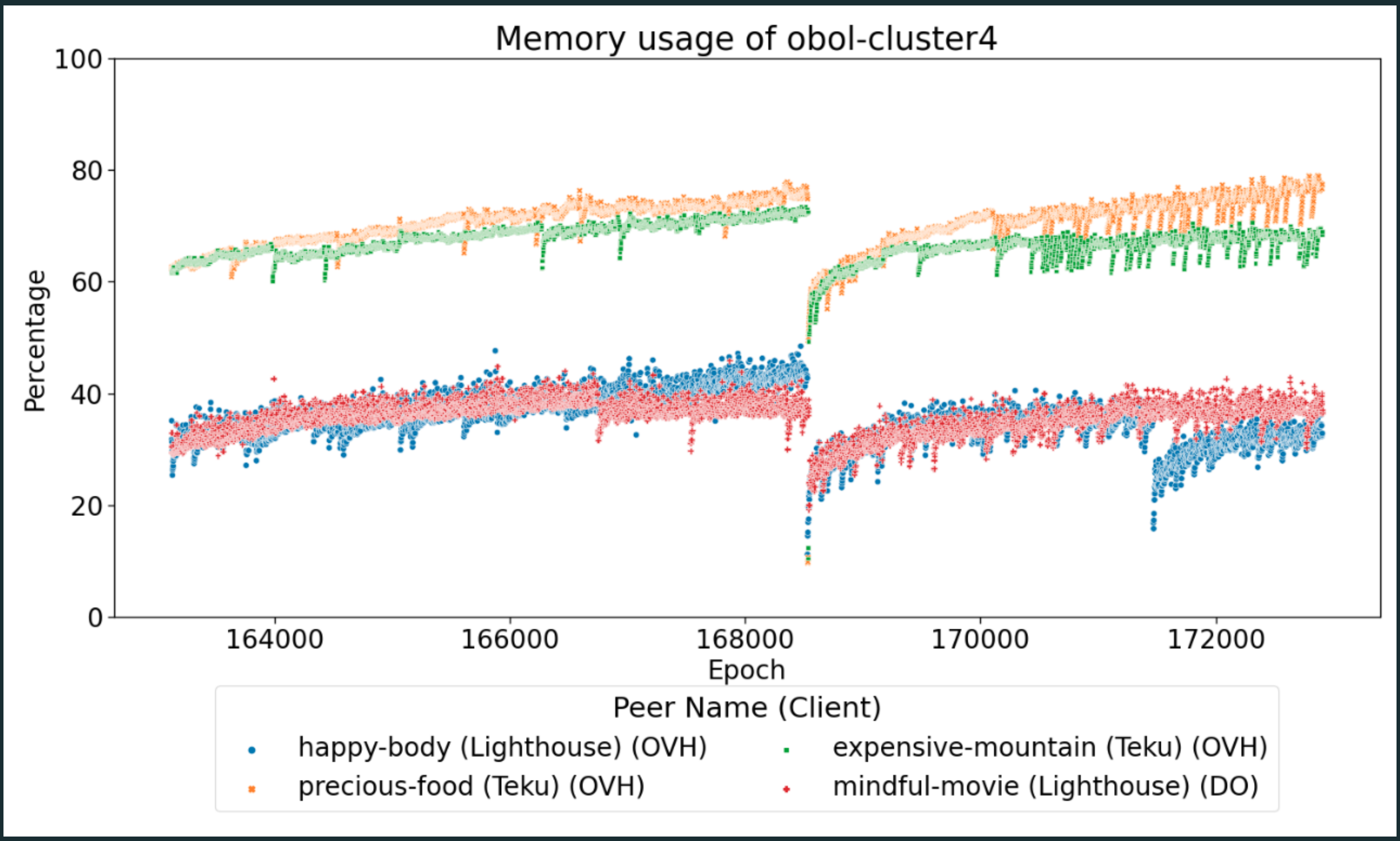

Figure 5: Memory consumption in Obol-cluster4

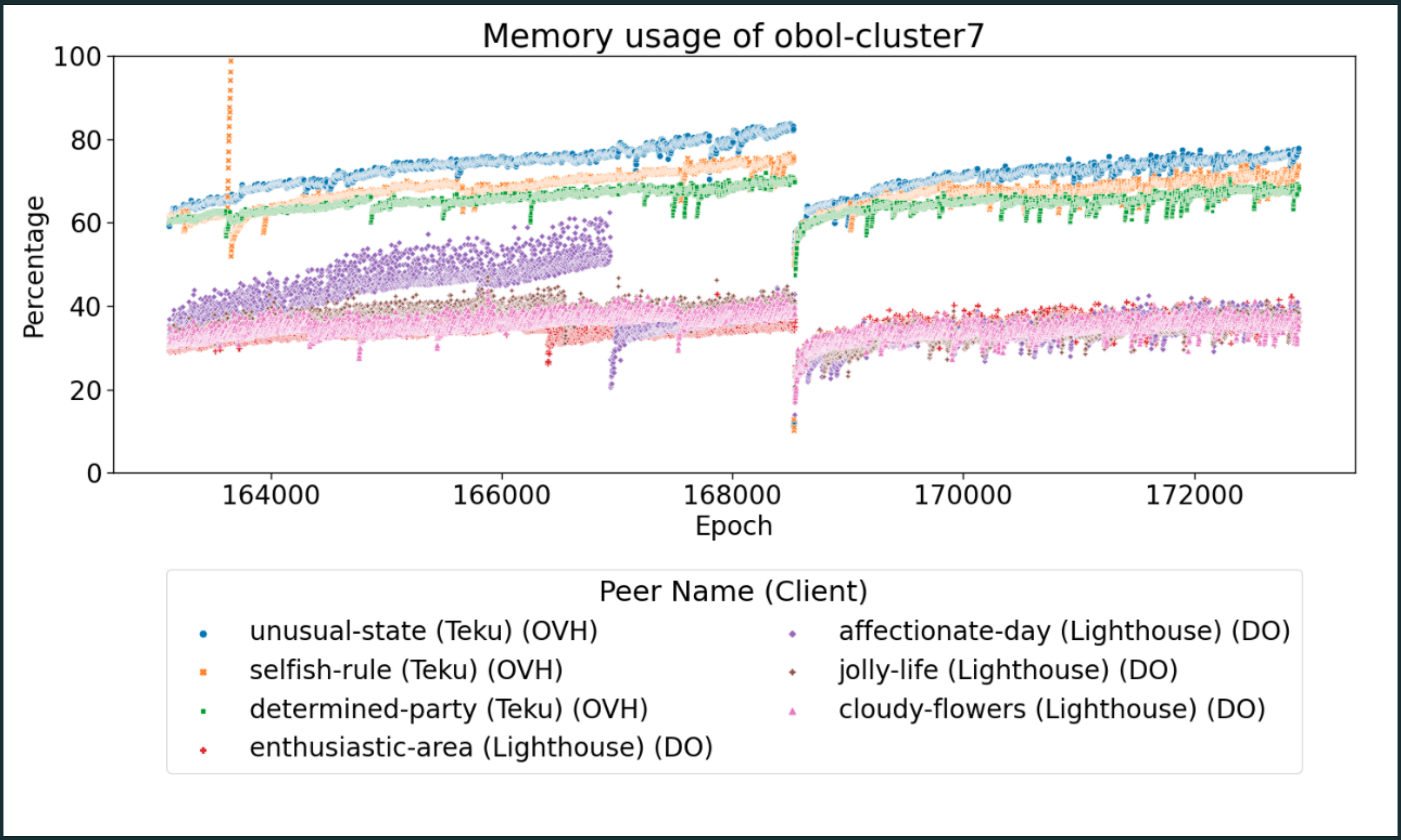

Figure 6: Memory consumption in Obol-cluster7

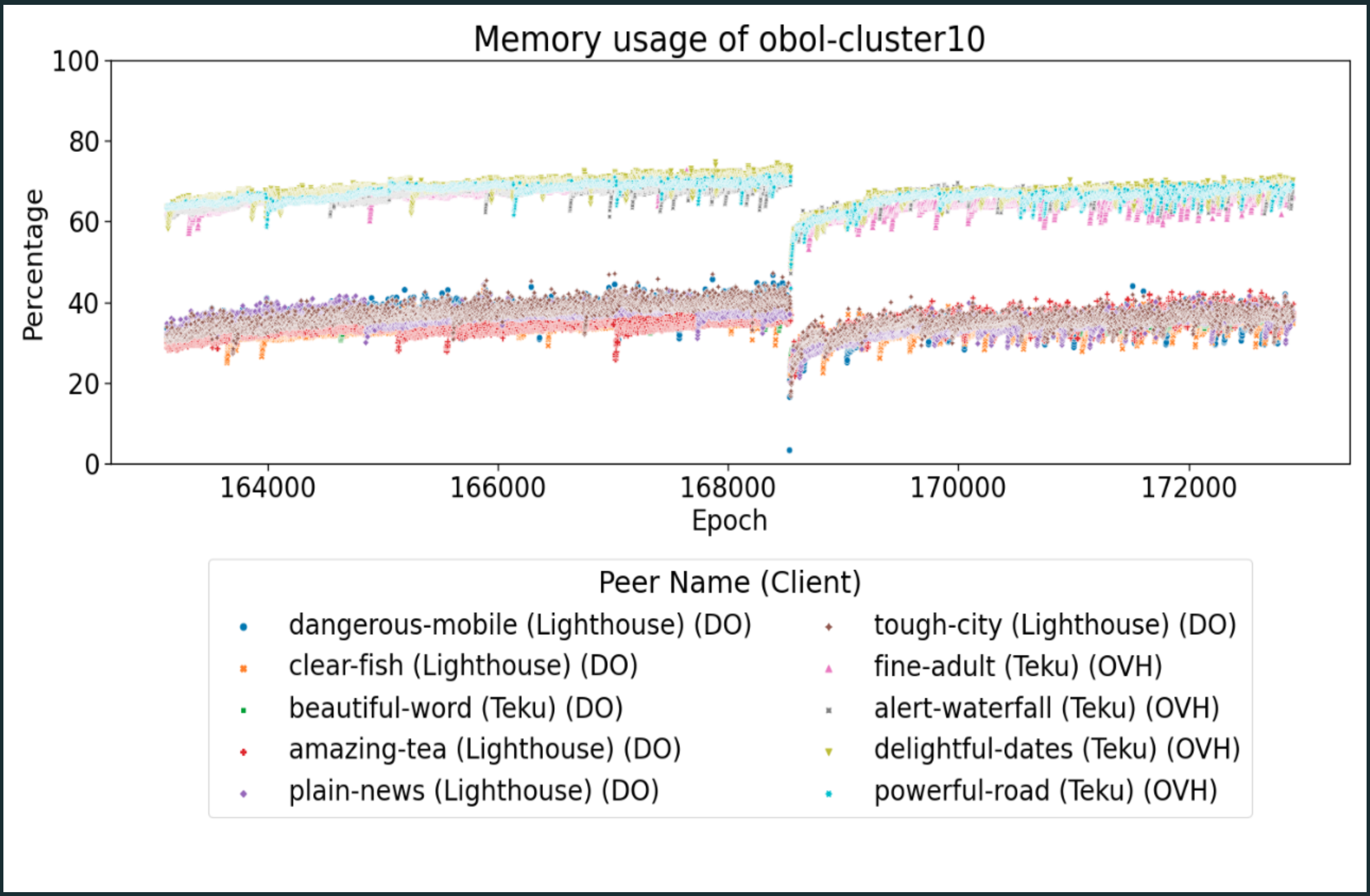

Figure 7: Memory consumption in Obol-cluster10

Similar to the CPU consumption, we can observe that some peers at each of the different clusters consumed more memory than others. However, this time the difference is due to the deployed beacon client. While peers running with the Lighthouse client consumed most of the time between 20% - 40% of the memory, peers running with Teku consumed most of the time between 60% - 80%.

Network Bandwidth

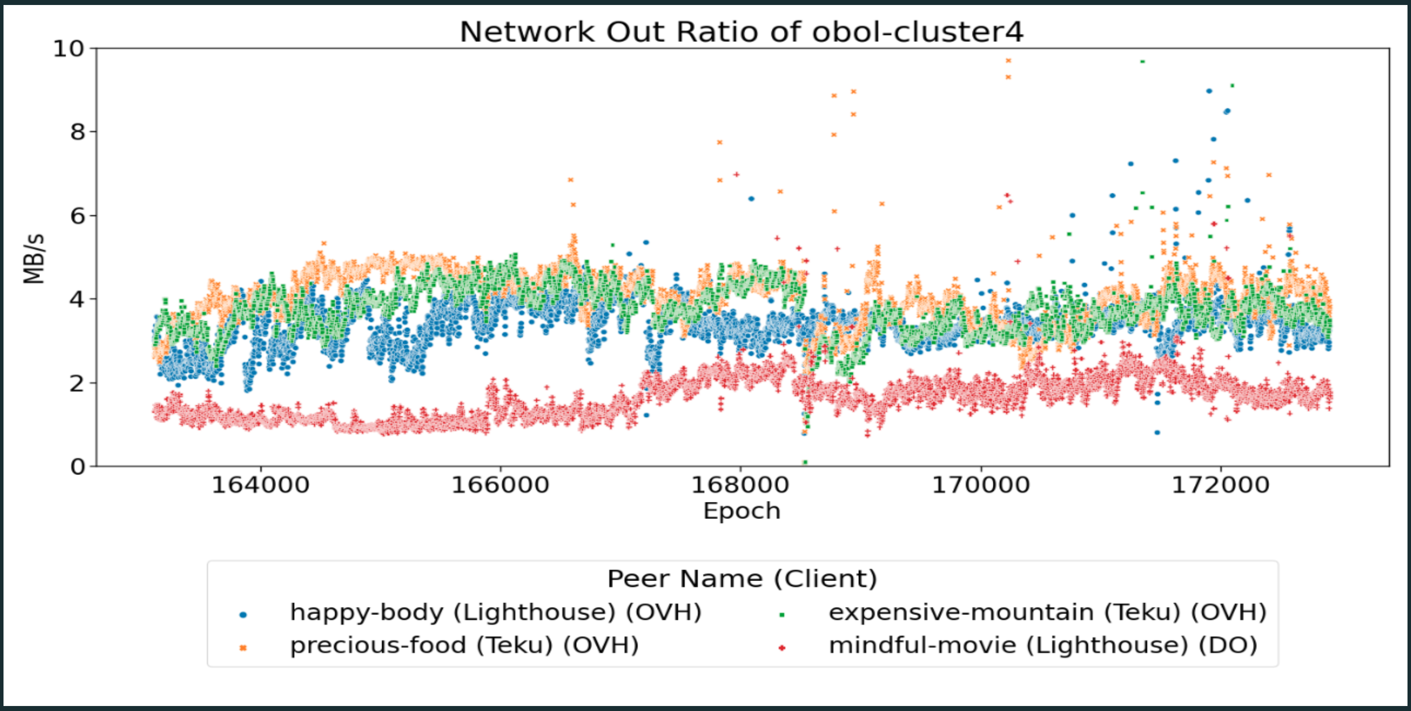

Figure 8: Network Sent Bytes Ratio in Obol-cluster4

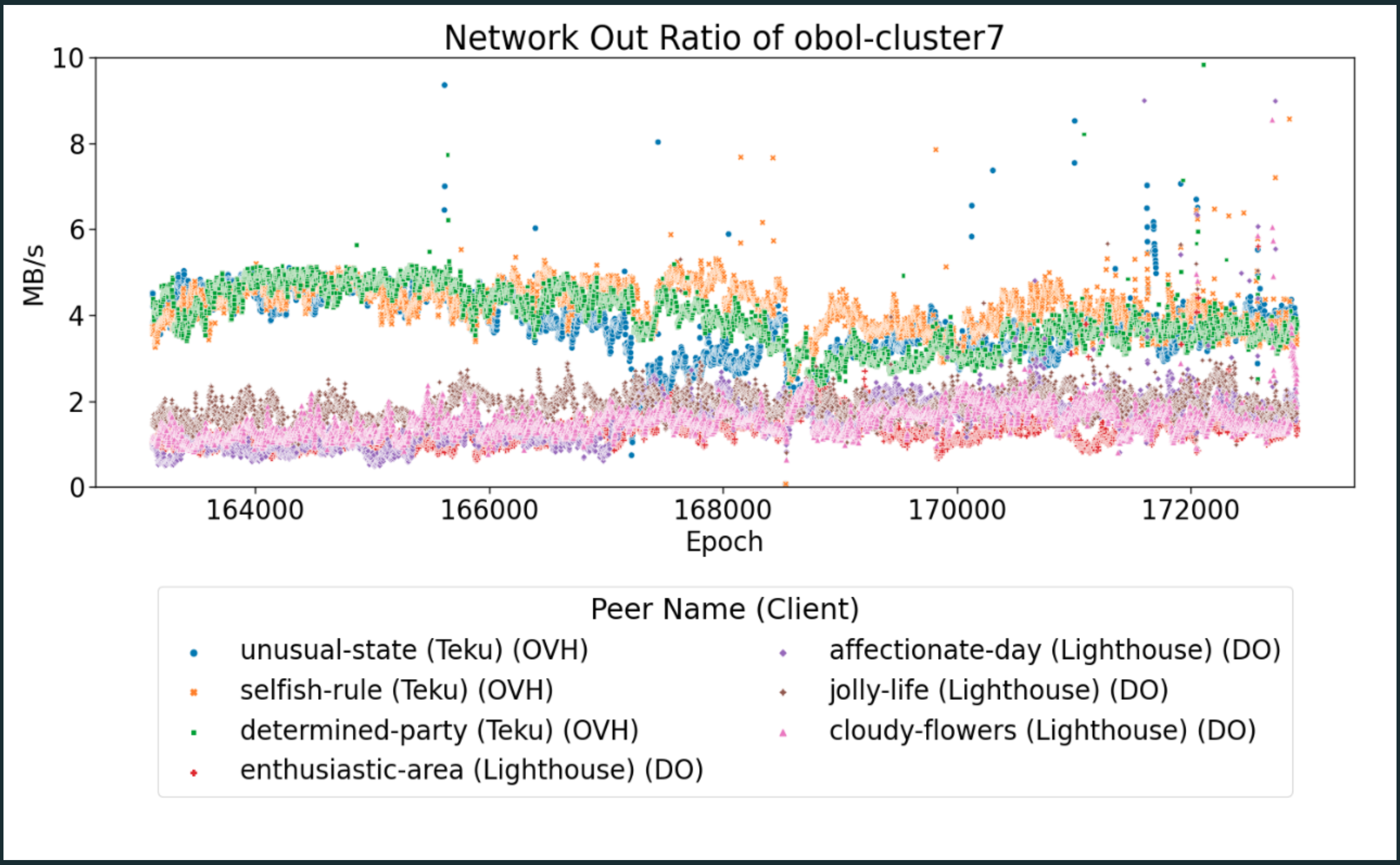

Figure 9: Network Sent Bytes Ratio in Obol-cluster7

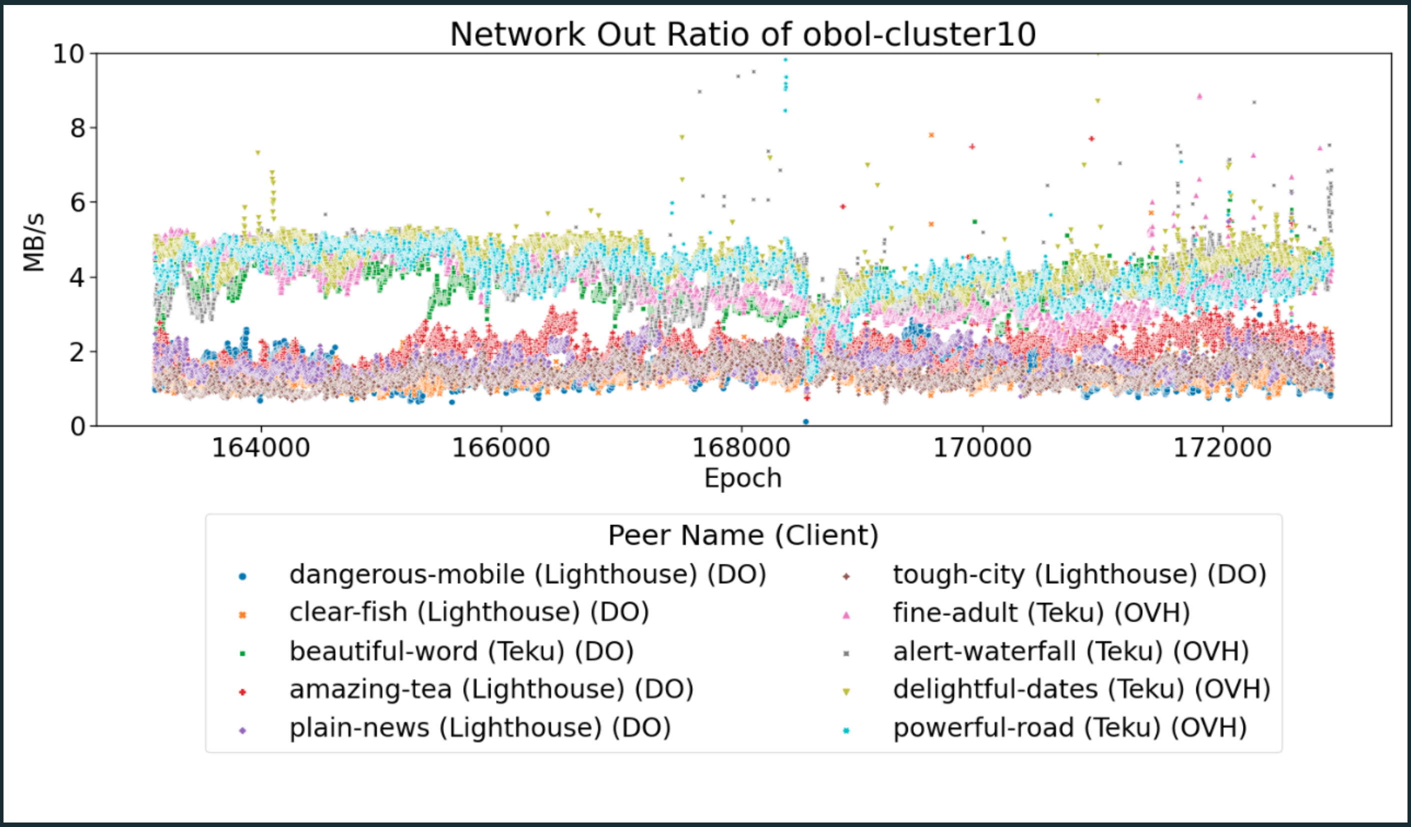

Figure 10: Network Sent Bytes Ratio in Obol-cluster10

Regarding network usage, we have extracted the ratio of megabytes sent per second throughout the whole experiment. In the three figures above, we can always observe two groups of peers. While some of them spend around 4MB/s of bandwidth most of the time, some others maintain themselves at around 2MB/s. Similarly to the CPU consumption, we can spot that these differences are due to the cloud provider: peers hosted on an OVH machine were capable of reaching a higher network throughput than the ones hosted on a DigitalOcean machine.

Overall, it is important to note that the nodes that were used for this experiment are not powerful machines, quite the opposite, they are considerably limited in terms of resources, and Obol DVT runs smoothly in this constrained environment.

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.

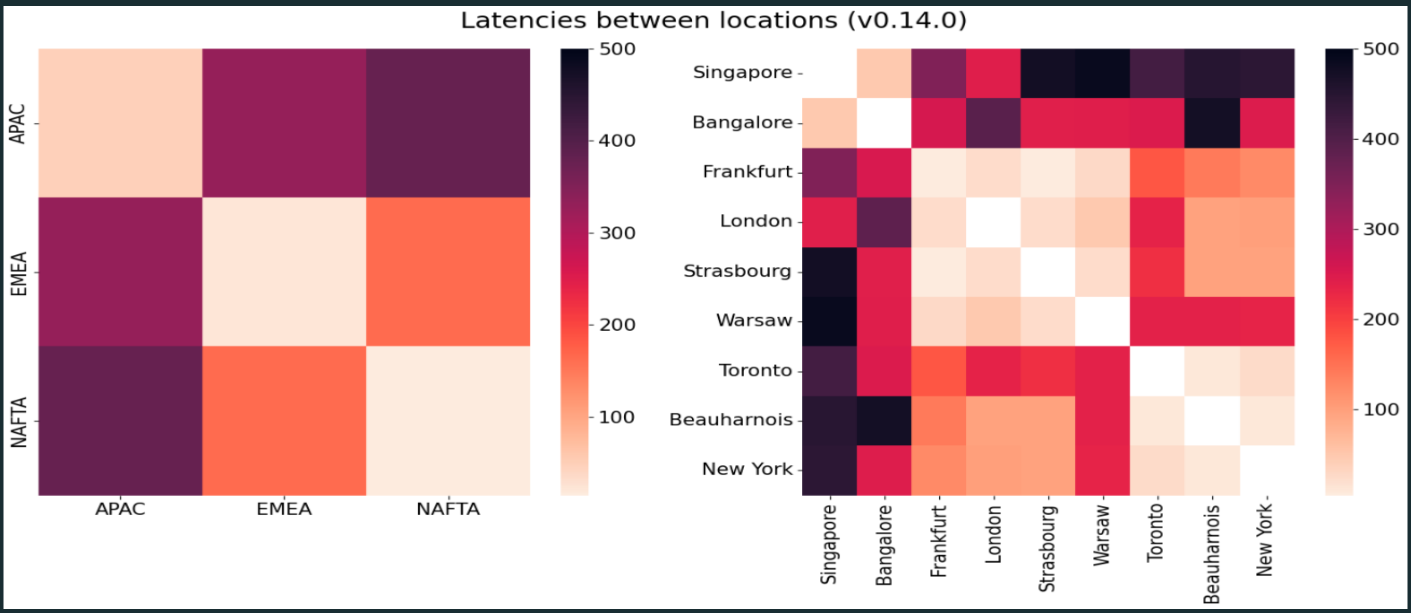

Figure 11: Latency between locations in v0.14.0

In *Figure 11* 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 substantially.

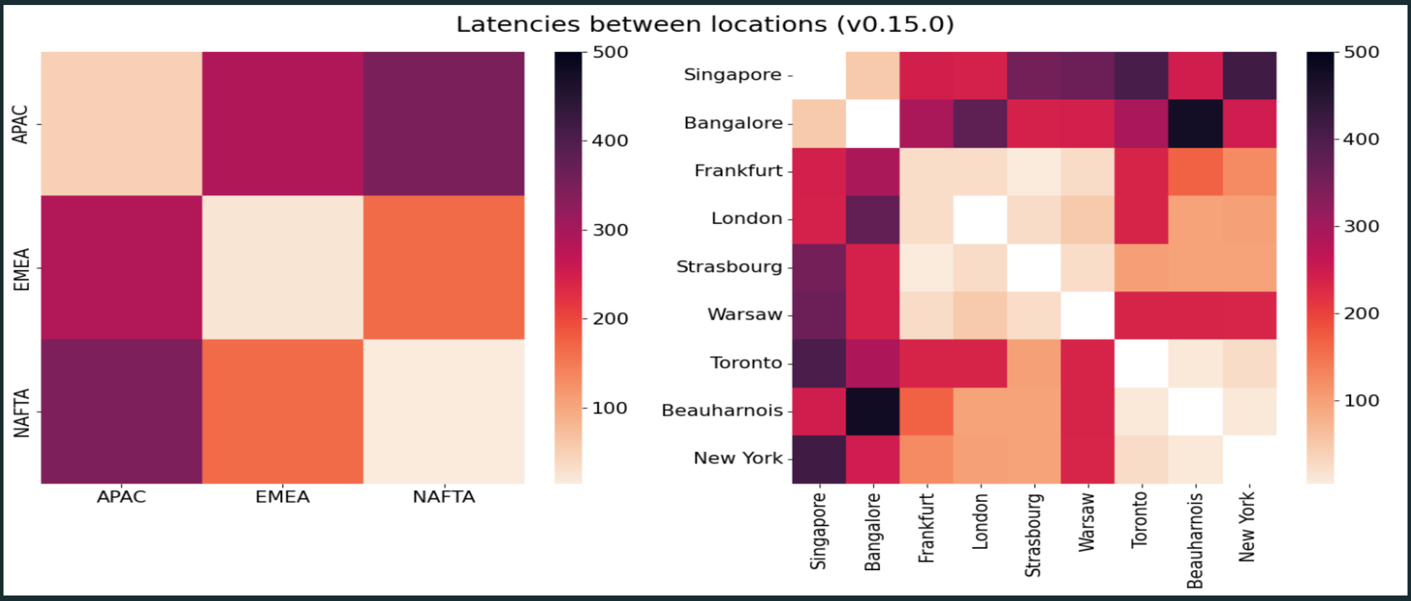

Figure 12: Latency between locations in v0.15.0

In general, we see a similar pattern in both versions but with significant improvements for **v0.15.0** 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.

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 8th), 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.

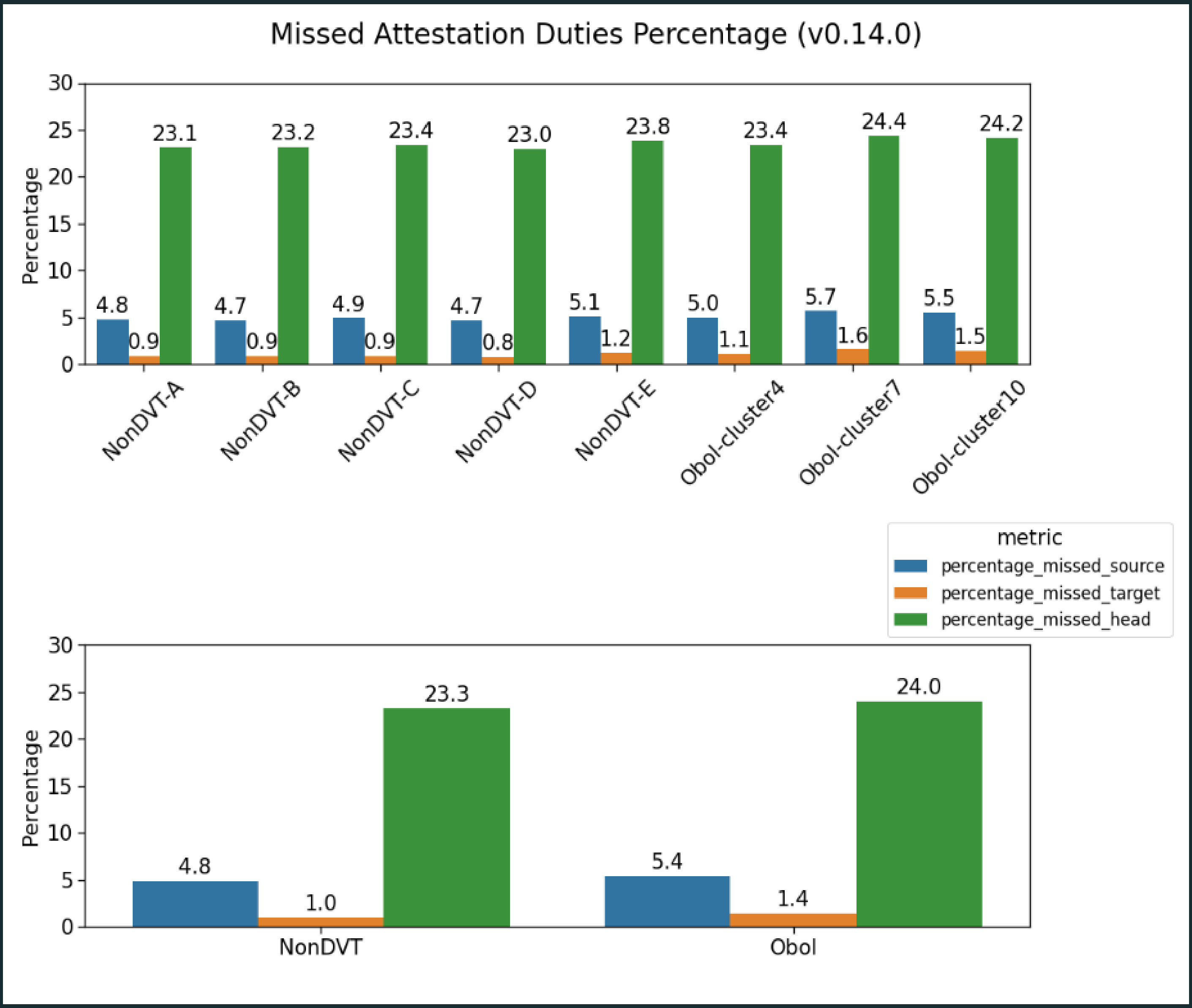

Figure 13: Missed attestation duties in v0.14.0

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 expected reward. See here to understand better how attestation flags are computed.

Figure 13 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. The difference in all cases between distributed and non-distributed validators is less than 1%, which demonstrates that Obol DVT has a similar performance in terms of attestation duties.

In Figure 14, 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.

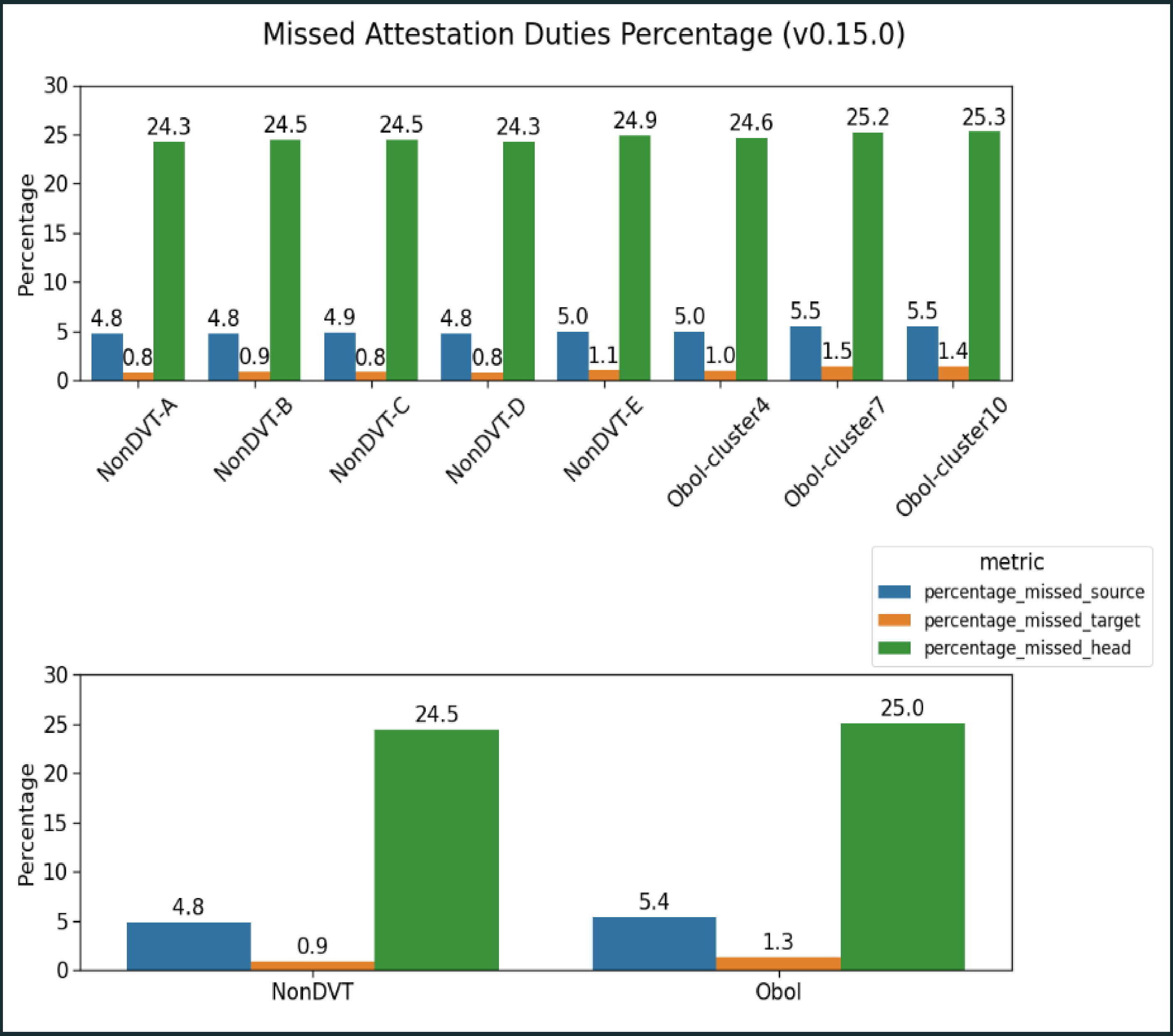

Figure 14: Missed attestation duties in v0.15.0

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 when they are chosen as the block proposer.

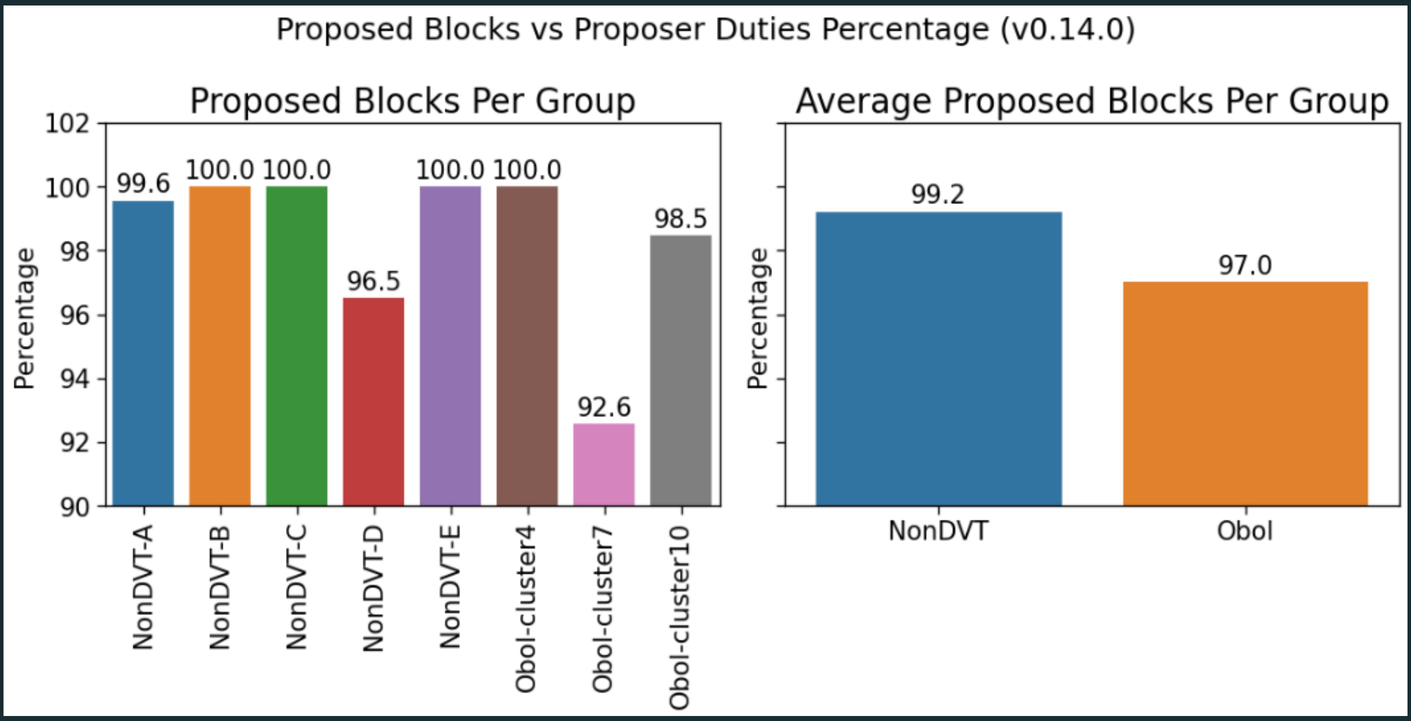

Figure 15: Block Proposal Duties in v0.14.0

In *Figure 15*, we can observe that all validators have a proposal rate over 90%. From the Obol DVT nodes, 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 miss any block proposal.

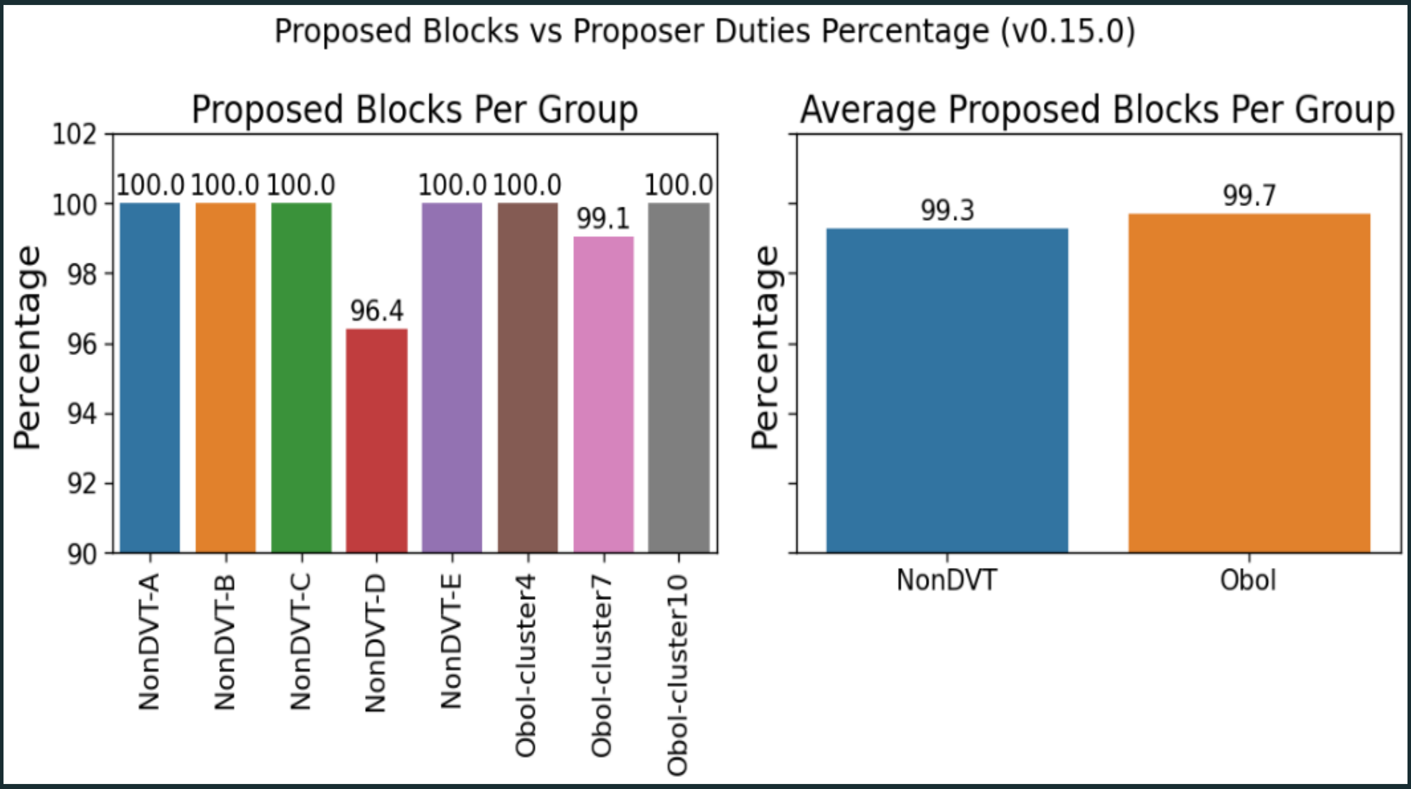

Figure 16: Block Proposal Duties in v0.15.0

In *Figure 16*, we can observe the same data as in *Figure 15* 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 **better performance** than non-distributed validators. In general, we see an improvement of **2.7%** in proposed blocks in distributed validators compared to **v0.14.0**. All missed blocks now concentrate on the validators in cluster7, while cluster4 and cluster10 proposed all its scheduled blocks.

Maximum Extractable Reward

After analyzing the attestation and block duties of validators, we want to check what was the actual reward that validators got with respect to the maximum they could have got. In *Figures 17 and 18* (v0.14.0 and v0.15.0 respectively) 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.

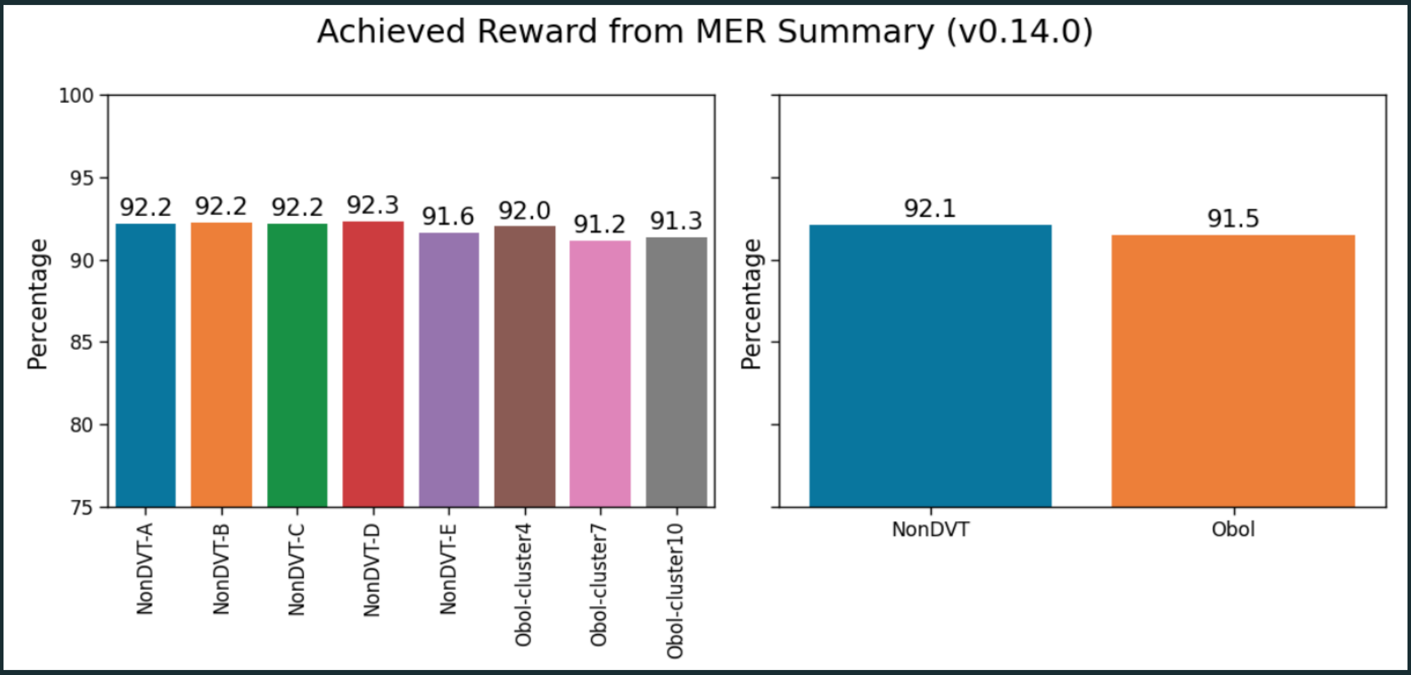

Figure 17: Achieved reward from Maximum Extractable Reward in v0.14.0

Looking at the data we can observe a similar performance for both, distributed and non-distributed validators. We can observe that in both versions the average extracted reward of distributed validators is **between 91% and 92%**, not even 1% below the performance 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 due to dangling validators, among other reasons. Note that with Charon v0.15, the difference between DVT and non-DVT is only **0.4%.**

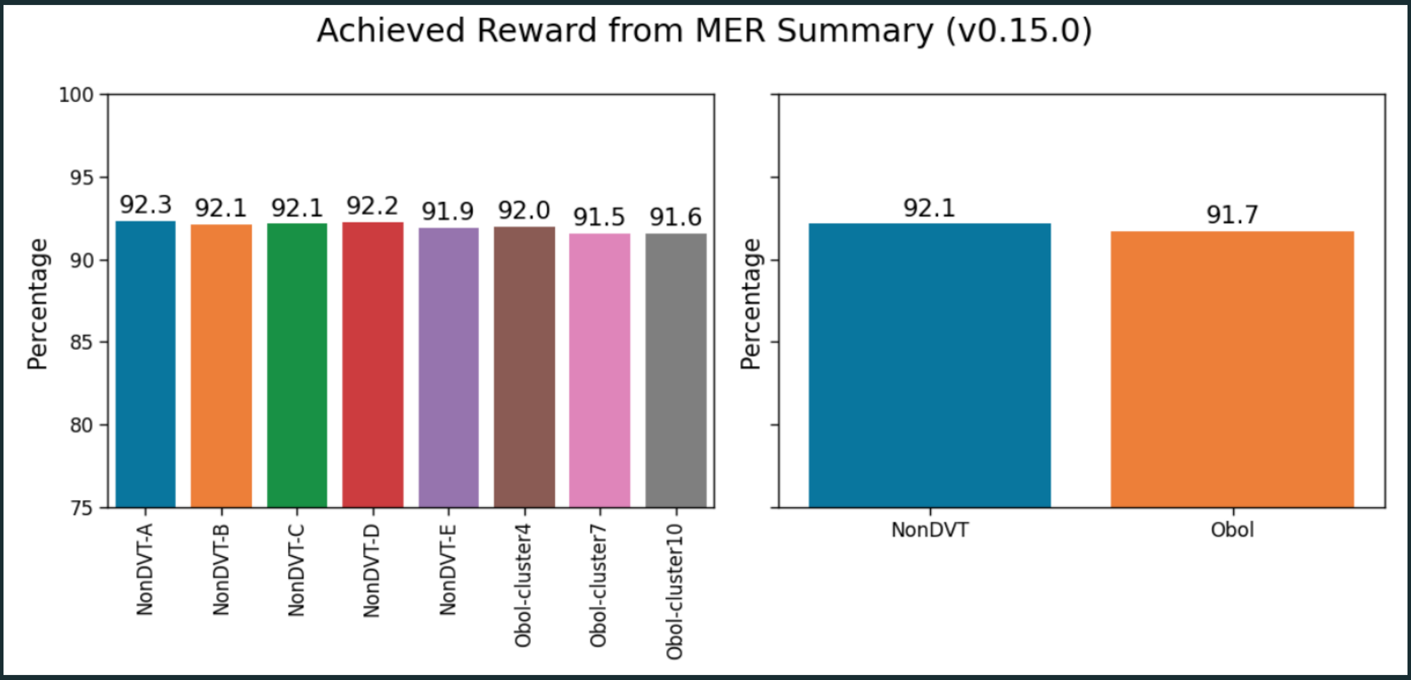

Figure 18: Achieved reward from Maximum Extractable Reward in v0.15.0

Performance Timeline

As mentioned before, the obtained rewards by a validator clearly shows 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.

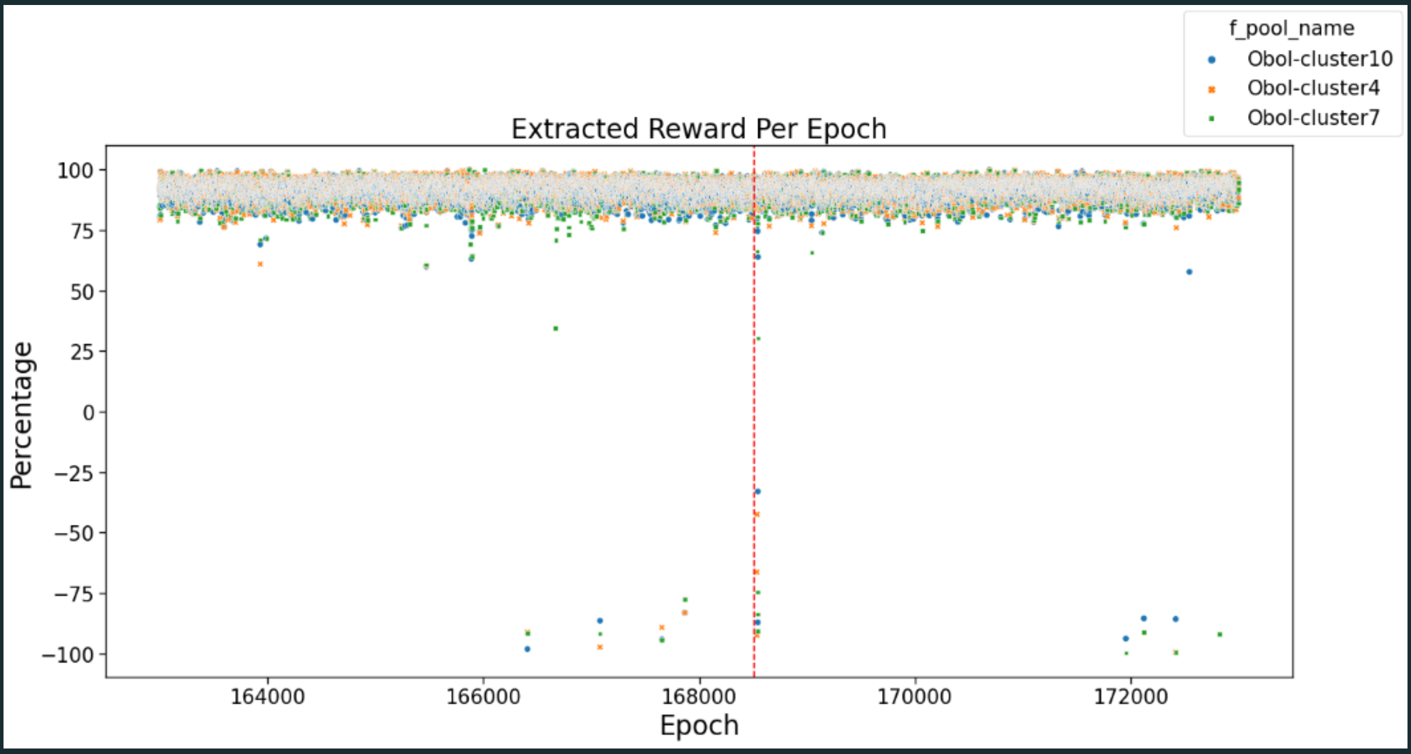

Figure 19: Achieved Rewards Timeline

In *Figure 19*, we can observe that the validator usually obtains 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 due to missing attestations or sync committee duties.

Some execution clients had issues with corrupted databases so we had to restart some of the nodes, which 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.

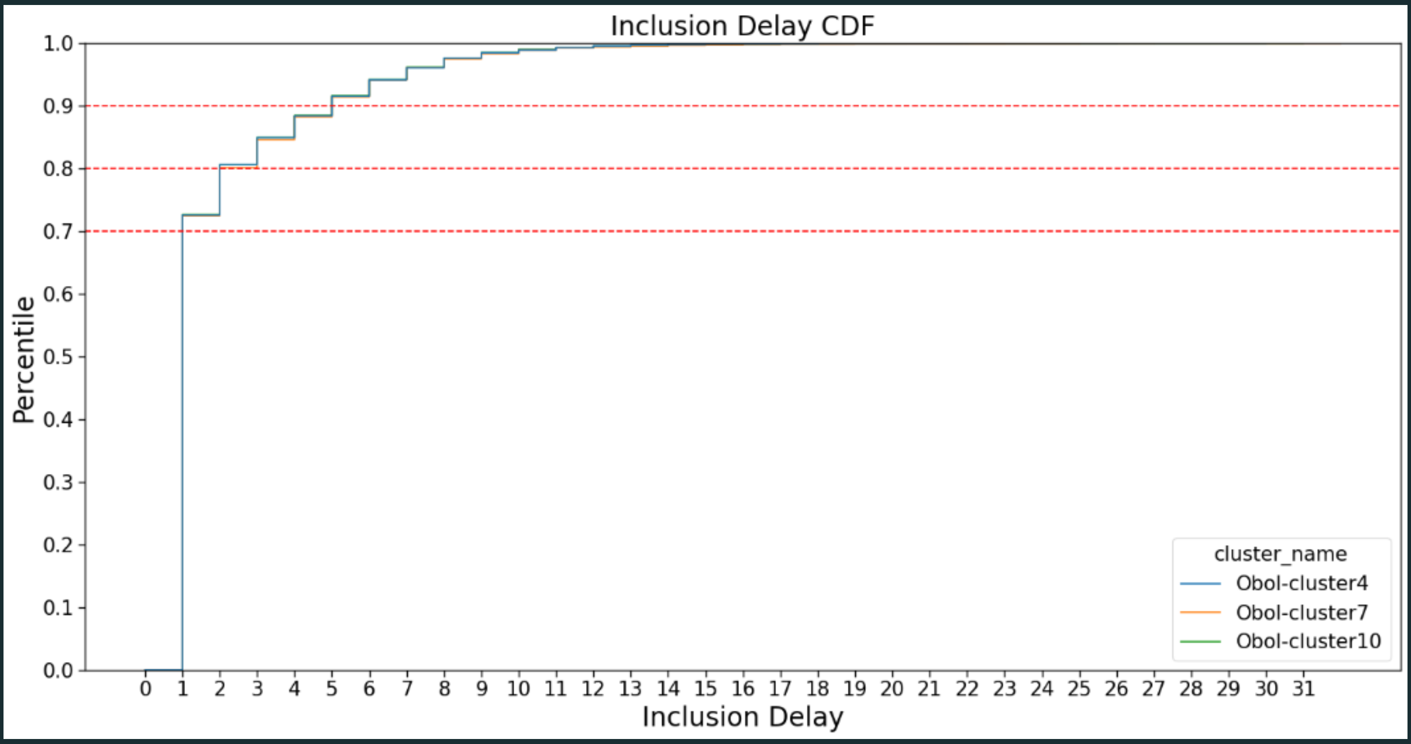

Figure 20: Inclusion delay by cluster

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 20*, we can observe that, most of the time (**over 70%**) the inclusion delay is 1 slot, which is the ideal scenario. However, we can see that there are some cases of inclusion delays between 2 and 5 as well (**around 20%** of the time), and the rest over 5 (l**ess 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.

Scaling Performance Test

Additionally, another experiment 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 in 3 different clusters running for a total of 10,000 epochs, for the scaling experiment we took the following steps:

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

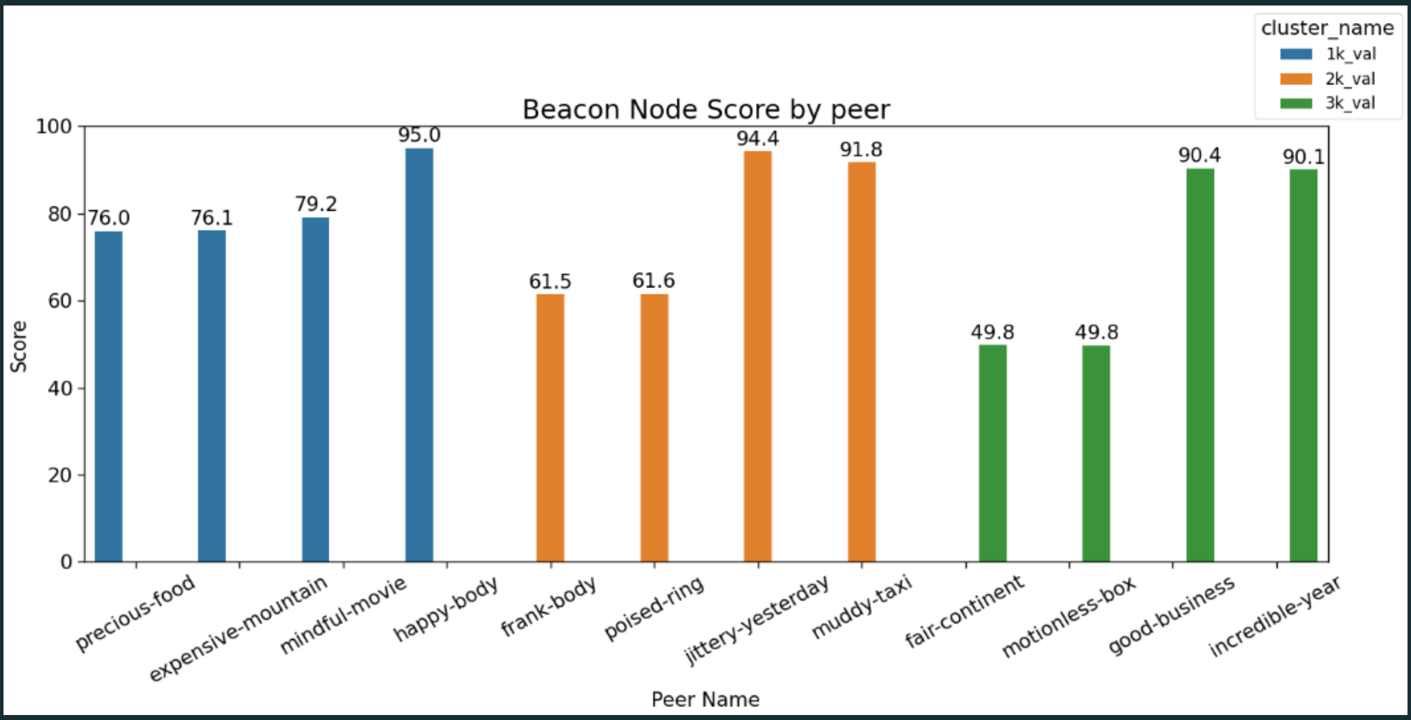

Figure 21: Beacon node score per peer and step

A larger number of validators per cluster means that 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 21, presents the beacon node score of the peers that form the 4-node cluster during the three phases of the experiment (each with a different number of validators). The beacon node score is calculated based on the Charon Prometheus metrics (using the number of errors and the latency of the queries). We can observe that, the more validators in the cluster, the lower the beacon node score.

On average, the DVT 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 11% worse compared to 1000 validators).

While the beacon node score gives us some indication about the performance of the DVT cluster, there is no clearer benchmarking than measuring the amount of rewards obtained during the three phases.

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.

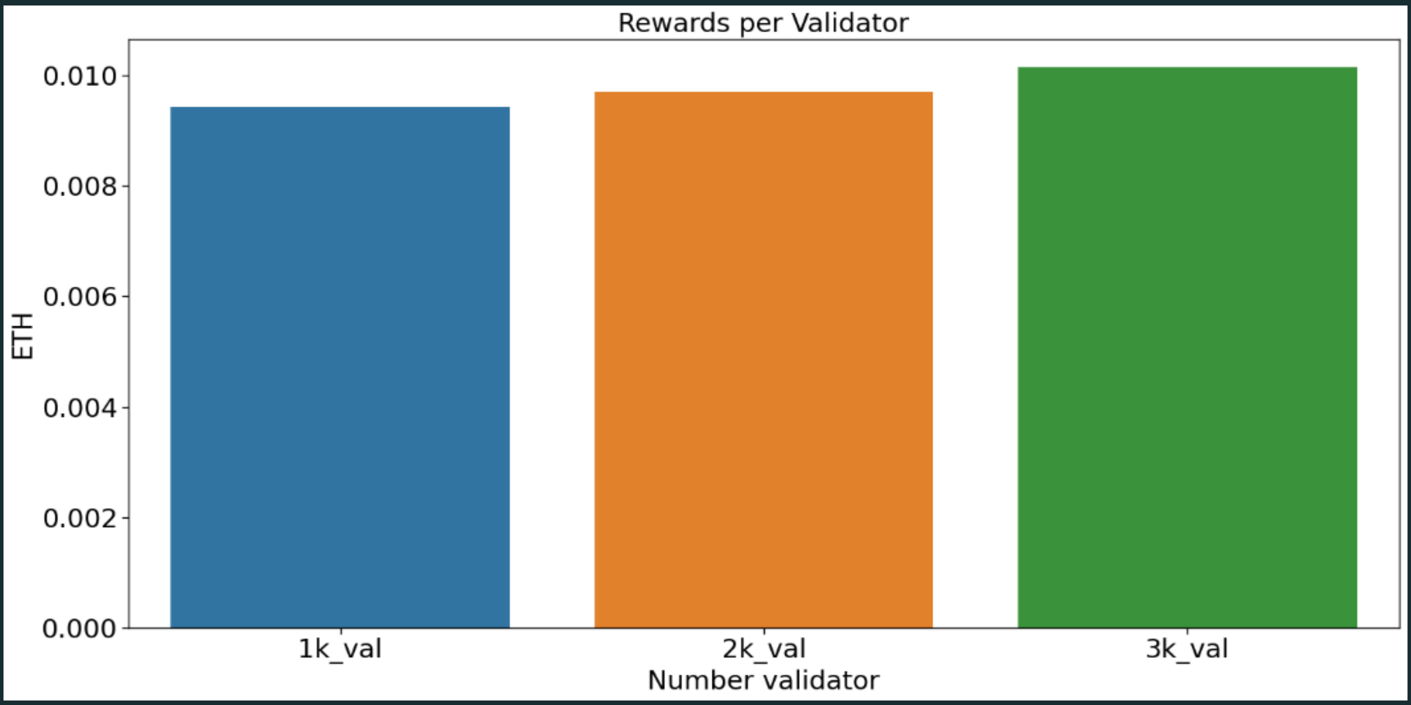

Figure 22: Achieved rewards per validator

Figure 22 shows the amount of ETH obtained by validators running during the previously mentioned three phases. To our surprise, we saw an increase in achieved per-validator normalized rewards when we increased the number of validators. We can see that after 1000 epochs, the 4-node DVT 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

Despite the decreasing beacon node score, the amount of rewards increases as the number of validators increases. This just shows that the internal beacon node score reported by Obol is still a work in progress and can be improved in the future.

Software Usability

In addition to the performance, we also analyzed the ease-of-use of Obol DVT in different uncommon scenarios.

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.

As with every software, less common use cases sometimes break something. Overall, the issues found were fixed and we managed to perform all the experiments we wanted to do.

Conclusions

We have analyzed Obol DVT performance from a resource and rewards perspective.

- From the hardware perspective, it is clear that resource consumption depends on the machine that hosts the node and the deployed beacon consensus layer client. More importantly, it is clear that **Obol DVT does not need to run on powerful hardware, it works smoothly running on limited hardware resources**

- In terms of latency, we have noticed that, as expected, clusters perform better when the nodes in a cluster are located close to each other in nearby regions, regardless of the region. However, even large clusters with nodes dispersed all over the world manage to get a good performance. In particular, there was **a great improvement (over 40%) in latency obtained by v0.15.0.**

- From the **attestation duties** perspective, we found that **Obol DVT performs very well with a difference lower than 1% compared to non-distributed validators.**

- Regarding **block proposals**, our study shows that Obol DVT misses very few proposals, and after the upgrade to v0.15.0**, the performance of Obol DVT is even higher than non-distributed validators**, with a total score of 99.7% for Obol DVT, compared to 99.3% for non-distributed validators.

- From the **achieved rewards** perspective, **the difference between Obol DVT and non-distributed validators is again almost negligible (less than 1%)**, which was expected after seeing the attestation duties and block proposal performance.

- Inclusion delay was also analyzed and our study shows that over **70% of attestations are included in the next slot**, and over 90% of them in the first 5 slots.
- During the scaling experiment we noticed that **the internal beacon node score reported by Obol technology is not a great performance indicator** and it still needs to be improved.
- **The combining keys feature was fixed during this study** and it now can merge thousands of keys in less than a minute, improving the usability of Obol DVT.
- **Node migration is feasible with Obol DVT, but it is important to improve the documentation,** in particular regarding less common usages. The objective is to facilitate the work of staking operators as much as possible.
- The most surprising finding of this study was probably to discover that **Obol DVT performs better as you increase the number of validators** to several thousands. Larger scaling experiments should be performed in the future to stress as much as possible the hardware resources and measure the limit of this scaling.