Ethereum 2.0 Client Metrics 10/2020

Afri Schoedon, @q9f

October 1, 2020

I. Introduction

Ethereum 2.0 will be a new blockchain protocol enabling – amongst others – horizontal scalability through sharding and transitioning the chain to a proof-of-stake consensus algorithm.

i. Motivation

None of the features that Ethereum 2.0 will bring are being implemented in established Ethereum 1.x clients such as Geth or Besu. Therefore, a new generation of core clients to power the beacon chain is under development. None of these clients has ever been used in production before.

With the launch of the beacon chain supposedly happening in 2020, the third compilation of key metrics of four selected Ethereum 2.0 clients will be conducted, namely Lighthouse, Prysm, Teku, and Nimbus.

This work shall allow insights into the performance and stability of the given beacon-chain node implementations.

ii. Previous Benchmarks

In June 2020, a similar, preliminary benchmark has been conducted[1] gathering first insights into client metrics and getting feedback from the Ethereum 2.0 core-developer community.

In July 2020, a similar, more careful benchmark has been conducted[2].

Before diving into the results, please note the following.

1. The four profiled clients are still the same, however, three months of active development have passed since the last benchmark.
2. The numbers in this report are *not* comparable with numbers in the previous reports. This is mainly due to the use of different hardware for each individual benchmark.
3. Last but not least, this benchmark was conducted on the Medalla testnet[3]. In contrast to the Witti and Altona testnet used in the previous reports, the Medalla testnet has a different composition of validators and is entirely run by the community.

iii. Commented Data

This article seeks to document the gathered metrics of different clients adhering to scientific methodology. It does not, however, intend to replace a peer-reviewed publication. It simply represents a version of the data commented by the author.

The raw data is available on Github[4] for further analysis.

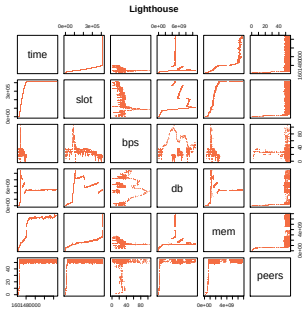
II. Clients

Four clients are used for comparing key-performance metrics.

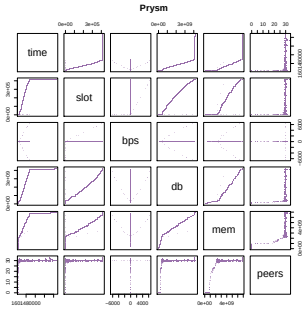
Lighthouse is an Ethereum 2.0 client developed by Sigma Prime[5]. It is implemented in the Rust programming language. Data referring to the Lighthouse client is depicted in orange throughout this document (figure 1a).

Prysm is a beacon-chain implementation written in Go[6]. It is being maintained by the

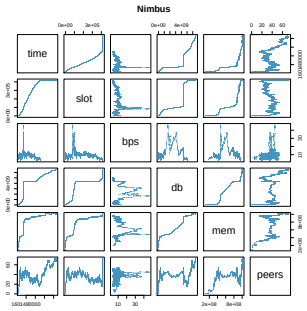
[1] github.com/q9f/eth2-bench-2020-06
[2] github.com/q9f/eth2-bench-2020-07
[3] github.com/goerli/medalla
[4] github.com/q9f/eth2-bench-2020-10
[5] github.com/sigp/lighthouse
[6] github.com/prysmaticlabs/prysm



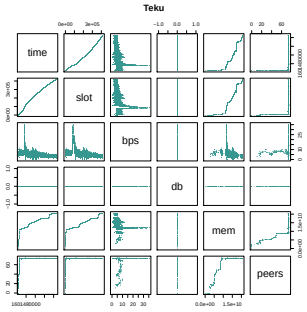
(a) *Lighthouse is depicted in orange.*



(b) *Prysm is depicted in purple.*



(c) *Nimbus is depicted in blue.*



(d) *Teku is depicted in turquoise.*

Figure 1: *All data collected is displayed in these matrices: time running, slot height, blocks per second, database size, memory usage, and peer count.*

Prysmatic Labs team. Data referring to the Prysm client is depicted in purple throughout this document (figure 1b).

Nimbus is a beacon-node implementation written in Nim built by the Status team[7]. Data referring to the Nim-Beacon-Chain client is depicted in blue throughout this document (figure 1c).

Teku is an enterprise-grade Ethereum 2.0 client built by the PegaSys Engineering team[8]. It is implemented in Java and data referring to the Teku client is depicted in turquoise throughout this document (figure 1d).

Other clients implementing the Ethereum 2.0 protocol exist, namely ChainSafe Systems' Lodestar[9], Nethermind's Cortex[10], and the Ethereum Foundation's Trinity[11]. Due to the different progress of implementing the protocol specification and core components, these clients were not considered for comparison.

III. Metadata

The data is gathered on the Medalla testnet. Medalla is the fourth multi-client testnet launched with the four in section II introduced clients as genesis validators.

At the time of collecting the metrics, the Medalla testnet is based on `v0.12.3` of the Ethereum 2.0 beacon-chain specification. It contains approximately 410,000 slots and is run by 65,136 validators.

i. Host Systems

Four identical host systems have been installed for the sole purpose of the performance inspection. The host systems are dedicated bare-metal servers with an Ubuntu 20.04 LTS operating system kernel version `5.4.0-48-generic`.

The host machines are powered by an Intel Xeon Silver 4114 CPU with 20 cores. The avail-

[7] github.com/status-im/nim-beacon-chain
[8] github.com/PegaSysEng/teku
[9] github.com/ChainSafe/lodestar
[10] github.com/NethermindEth/cortex
[11] github.com/ethereum/trinity

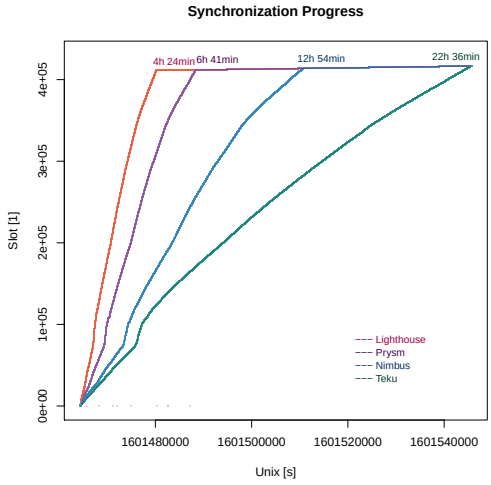


Figure 2: *Synchronization progress over time.*

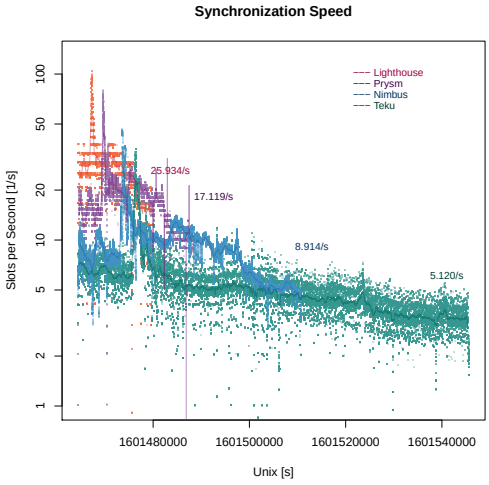


Figure 3: *Synchronization speed over time.*

able memory is 128 GB and the NVMe disks allow for 1024 GB capacity.

ii. Client Versions

All clients were compiled on September 30th, 2020, either from the latest available release tag or latest commit on the main branch targeting the version `v0.12.3` of the Ethereum 2.0 specification.

- **Lighthouse**: version `lighthouse/0.2.13`, compiled from `v0.2.13` tag at commit `c0e76d2c` from September 29th, 2020, with Rust version `1.46.0`-stable through Cargo.
- **Prysm**: compiled from `v1.0.0-alpha.27` tag at commit `eb3e4944` from September 29th, 2020, with Go version `1.13.8` through Bazel.
- **Nimbus**: version `beacon_node v0.5.0`, compiled from `master` branch at commit `78ceeed8` from September 29th, 2020, with Nim version `1.2.6` through Make.
- **Teku**: version `teku/v0.12.8`, compiled from `0.12.8` tag at commit `60bc0316` from September 27th, 2020, with Java version `14.0.1` through Gradle.

All clients contain a built-in Medalla configuration and were provided with a sufficient number of bootstrap nodes to ensure good connectivity and eliminate potential networking bottlenecks (compare section IV point iv).

IV. Performance

This document only inspects the performance metrics of beacon-chain node implementations. Other features such as running validator clients, bootstrap nodes, or other relevant tooling are disregarded for simplicity.

i. Synchronization Metrics

Figure 2 displays the progress of synchronizing the four aforementioned clients. Notably, the Lighthouse client manages to fully synchronize all blocks and verify all signatures in a little less than 4.5 hours with the Prysm client being right behind finishing the same task in just 6 hours and 41 minutes. Nimbus completes the same task in 12 hours and 54 minutes, whereas Teku requires 22 hours and 36 minutes to fully sync and verify the Medalla beacon chain.

Also, figure 3 displays the same data but computing the synchronization speed in slots per second by taking the time required to fully catch up with the beacon-chain head. The plotted data points display a moving average over

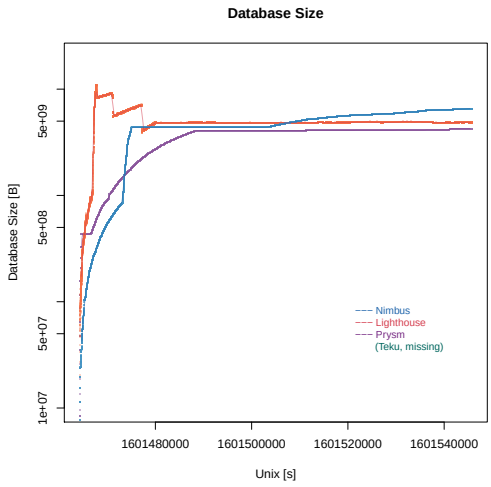


Figure 4: *Database size over time.*

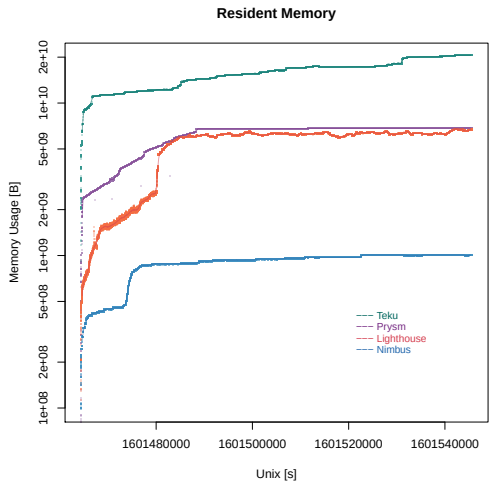


Figure 5: *Resident memory usage over time.*

60 seconds, the plotted line shows a moving average over 10 minutes. Lighthouse and Prysm lead the chart at an average of approximately 26 and 17 slots per second respectively on the dedicated hardware.

The data at glance.

- **Lighthouse** synchronizes 411,181 slots in 15,855 seconds at an overall average speed of 25.934 slots per second.
- **Prysm** catches up with 411,862 slots in 24,059 seconds at 17.119 slots per second.
- **Nimbus** synchronizes 413,727 slots in 46,412 seconds at an average speed of 8.9142 slots per second.
- **Teku** catches up with 416,639 slots in 81,375 seconds at 5.1200 slots per second.

ii. Database Metrics

Figure 4 displays the database size in Bytes plotted over time of running the nodes. The patterns are left uncommented for the anyone to analyze.

Teku database size is missing due to a typo on the data path (author's fault). Sorry.

iii. Memory Metrics

Figure 5 displays resident set size reported by the four clients. Again, the patterns are left uncommented.

Teku reports a little less than 20 GiB. The actual Java heap memory used by Teku on Medalla can be assumed much lower. The off-heap memory that Java allocates is outside of the team's easy control. The JVM is being greedy about available memory, however, it is still possible to run Teku nodes on machines with very small available memory, e.g., 2GB.

iv. Networking Metrics

Figure 6 displays the peer count of every client during operation. There is not much to be commented on. This metric simply serves as a sanity check to rule out networking issues that could impact any of the other metrics.

V. Conclusion

The plots allow for an overview of key performance and stability metrics of the four tested clients.

Notably, both Lighthouse and Prysm appear to be highly optimized in their perfor-

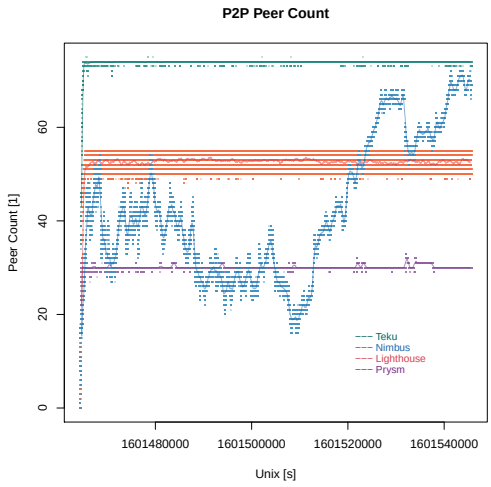


Figure 6: *Client's peer count over time.*

mance and mature in the implementation of the beacon-chain specification.

The relatively new Nimbus client already shows good performance but the metrics allow the conclusion that there is still room for optimization.

Note

The author is not affiliated with any of the teams implementing an Ethereum 2.0 client. The author is independently funded through the Ethereum Foundation's Ecosystem Support Program[12] and the Goerli Testnet Initiative[13].

The author is not speaking on behalf of any organization.

A warm note of thanks goes out to everyone who reviewed the initial June and July reports and provided valuable feedback allowing for a more accurate data gathering in this subsequent report.

And finally, a big thanks to the client teams patiently answering questions and sharing insights about the protocol implementations.

:)

[12] esp.ethereum.foundation
[13] goerli.net