Reth Roadmap 2024 & Beyond

Georgios Konstantopoulos
CTO & Research Partner
@Paradigm @gakonst

Agenda

- Recap, motivation, where we are, some stats
- 2024 Roadmap
- Beyond
- Q&A at the end for zoomed out / non-technical discussion!

Introducing Reth

Dec 07, 2022 | Georgios Konstantopoulos

Contents

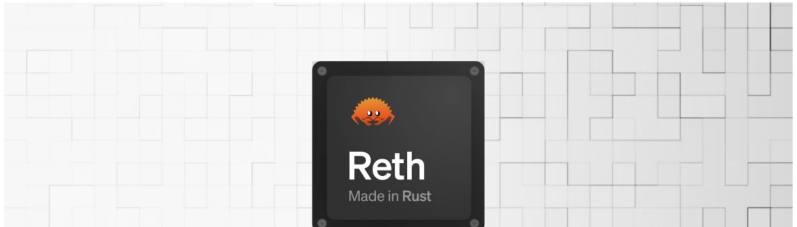
We're excited to announce Reth, a free, open-source Ethereum execution layer client built by Paradigm. In this post, we'll discuss why we are making Reth and what to expect from us in the future.

Releasing Reth!

Jun 20, 2023 | Georgios Konstantopoulos

Contents

Introduction



Announcing Reth Beta!

Mar 11, 2024 | Georgios Konstantopoulos

Contents

We are excited to announce that after 21 alpha releases since July 2023, Reth is entering beta.1 and preparing for our 1.0 "production ready" release once audits are done in the next few months. In this post, we'll discuss the core features, performance, and stability of Reth Beta. In the near future, we will be publishing guides on how to maximally leverage Reth in your infrastructure, and roadmaps for the future of Reth.

Why Reth?

- **Client diversity:** We need independent client implementations for staking.
- **Talent resilience:** We need clients that onboard new core devs, more eyes on Ethereum protocol.
- **Client scalability:** We need clients built for a high gas per second L2 world.
- **Code extensibility:** We need clients that are easy to extend with principle, without “cowboy” forking/rebasing.

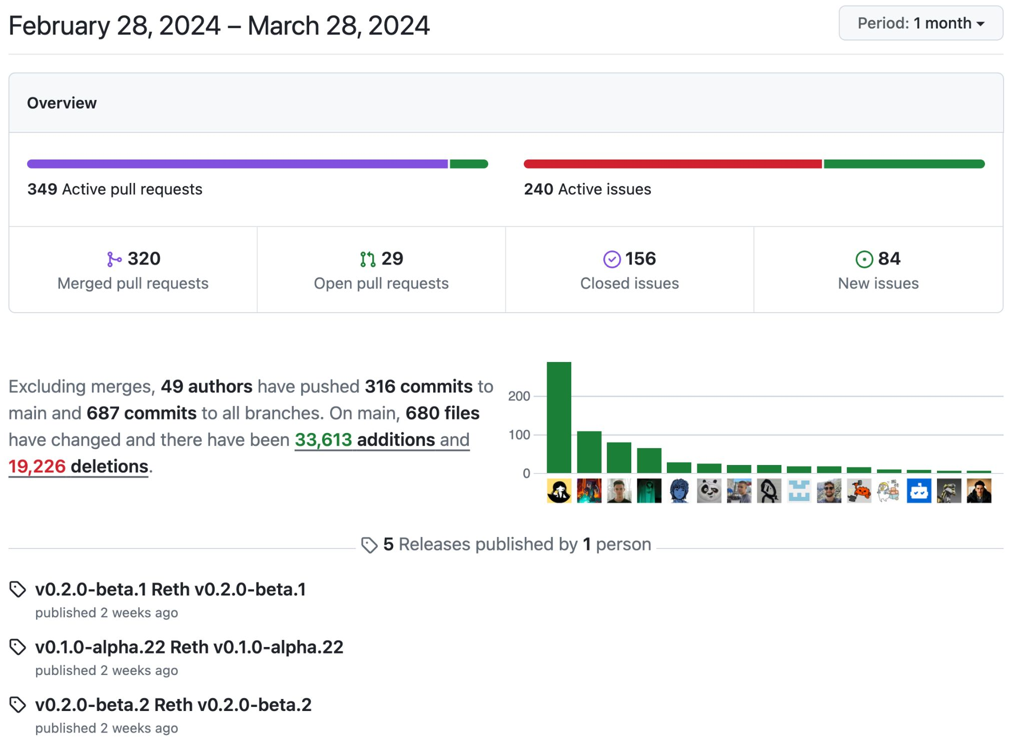
→ *“Reth is a blazing-fast, modular, contributor-friendly Ethereum client in Rust”*!

2024 Goals

1. Credible alternative client for L1 Ethereum usage.
2. Fastest L2 EVM client.
3. State of the art framework for building EVM infrastructure.

What have we done so far?

Inclusive Open Source Culture & Shipping


February 28, 2024 – March 28, 2024

Period: 1 month

Overview

349 Active pull requests

240 Active issues

320	29	156	84
Merged pull requests	Open pull requests	Closed issues	New issues

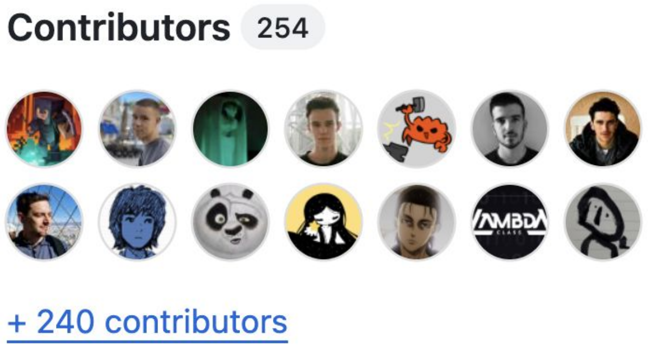
Excluding merges, **49 authors** have pushed **316 commits** to main and **687 commits** to all branches. On main, **680 files** have changed and there have been **33,613 additions** and **19,226 deletions**.

5 Releases published by **1** person

v0.2.0-beta.1 Reth v0.2.0-beta.1
published 2 weeks ago

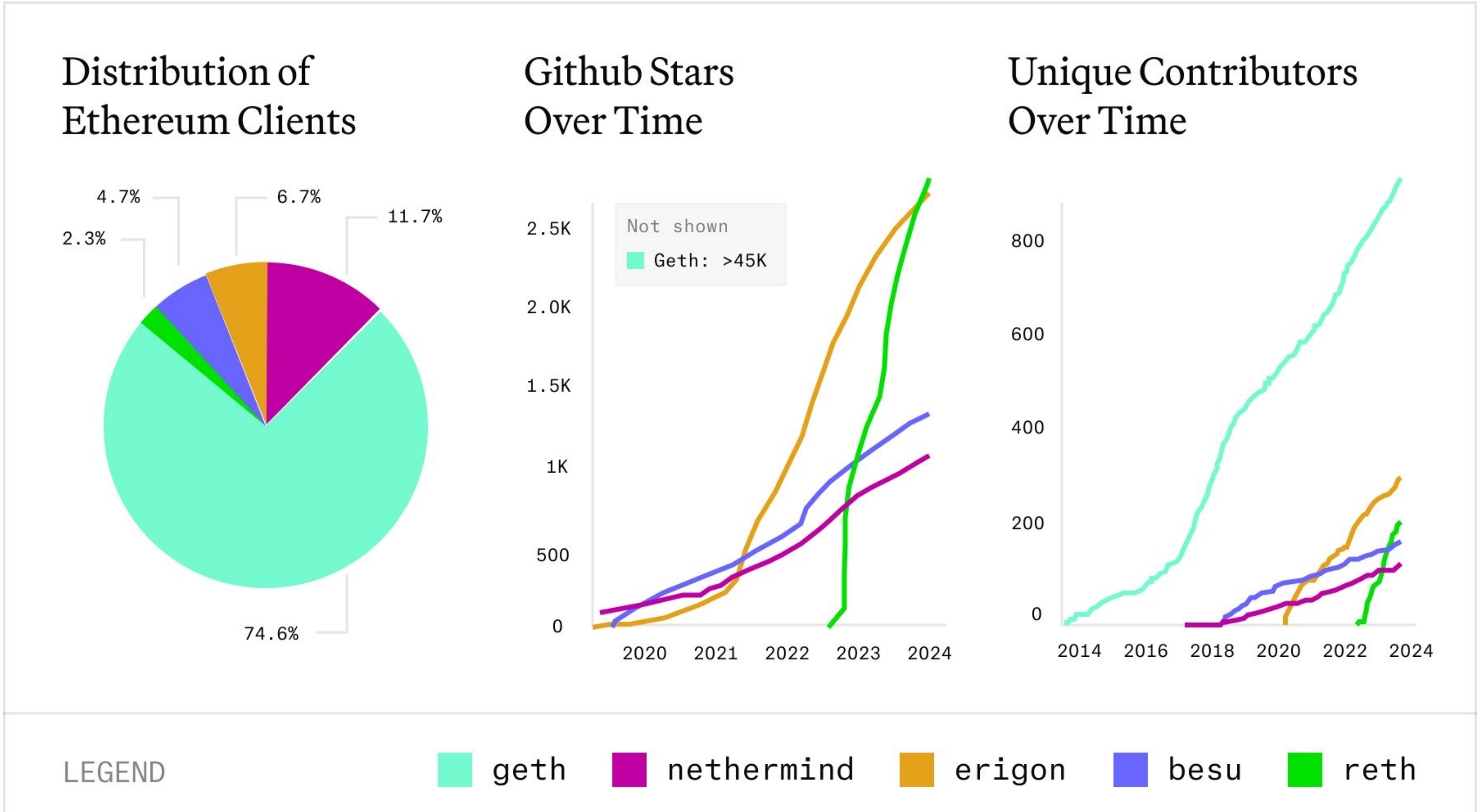
v0.1.0-alpha.22 Reth v0.1.0-alpha.22
published 2 weeks ago

v0.2.0-beta.2 Reth v0.2.0-beta.2
published 2 weeks ago


Contributors 254

+ 240 contributors

Who cares?

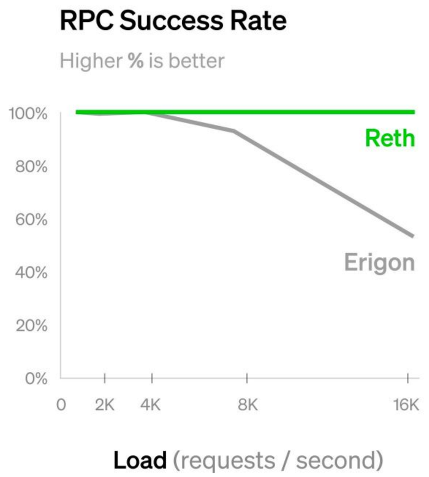

Distribution of Ethereum Clients
4.7%
6.7%
11.7%
2.3%
74.6%
Github Stars Over Time
Not shown
Geth: >45K
2.5K
2.0K
1.5K
1K
500
0
2020
2021
2022
2023
2024
Unique Contributors Over Time
800
600
400
200
0
2014
2016
2018
2020
2022
2024
LEGEND
geth
nethermind
erigon
besu
reth

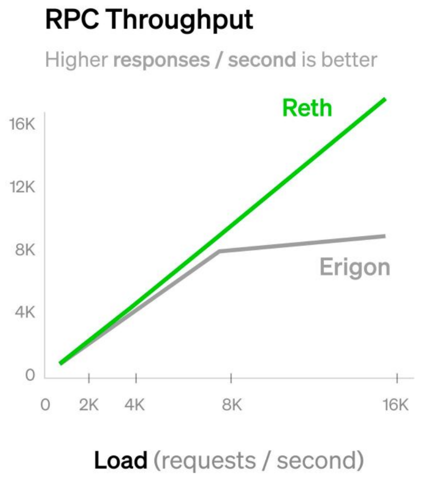
Is this fast?

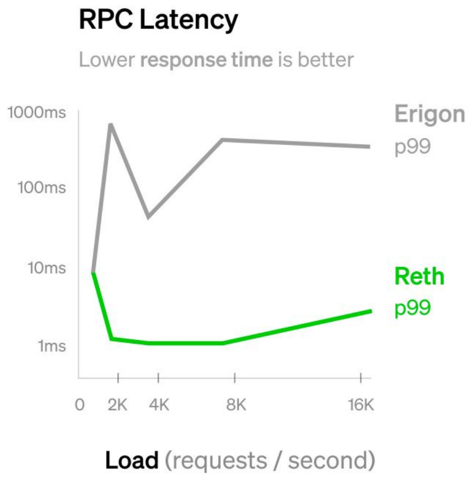
Bunch of numbers and charts ahead

Releasing Reth (2023)

Sync Benchmarks	Reth	Erigon	Geth	Nethermind
Time to Sync	~50 hours	~5 days	1+ month	1+ month
Database Size	1.92 TB	2.2 TB	14.5+ TB	14.5+ TB

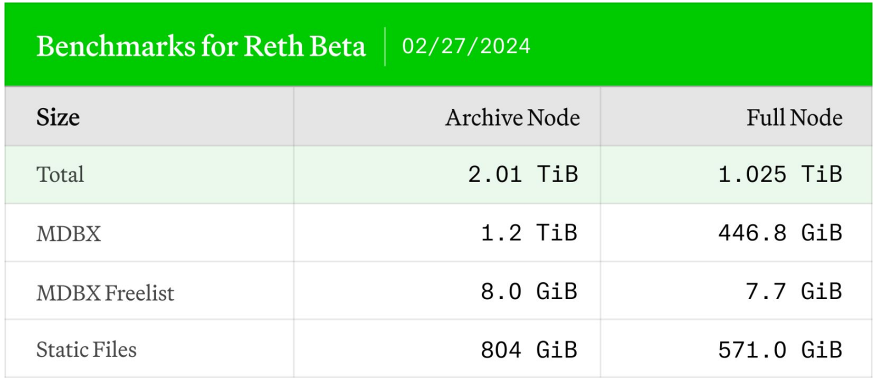






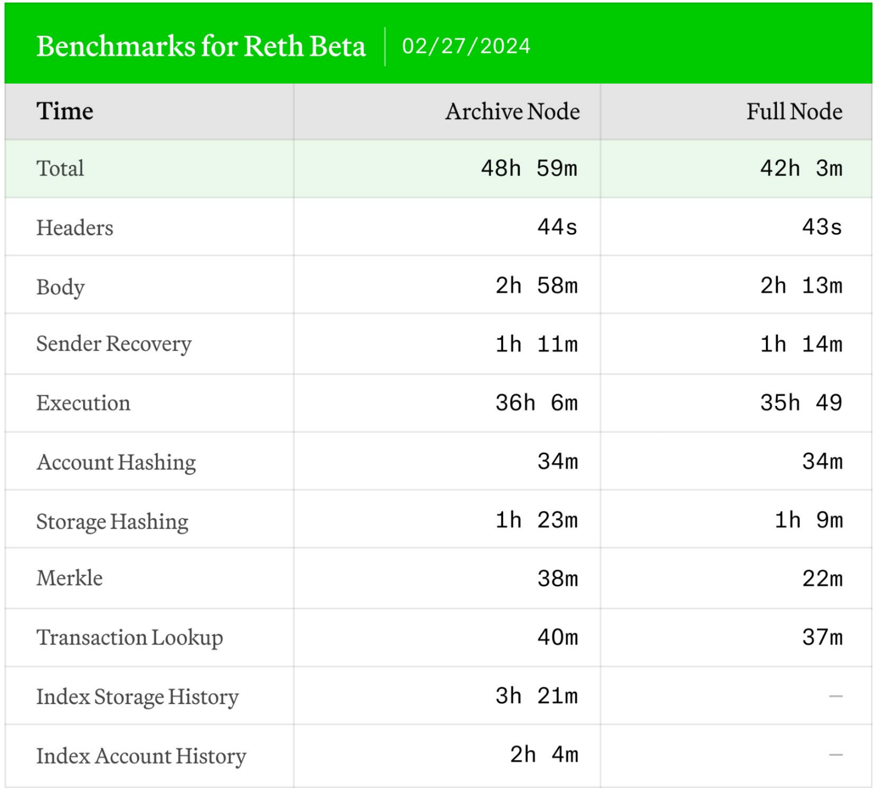
RPC benchmarked via `eth_call` with historical mainnet loads using flood

Historical Sync Breakdown



Benchmarks for Reth Beta | 02/27/2024

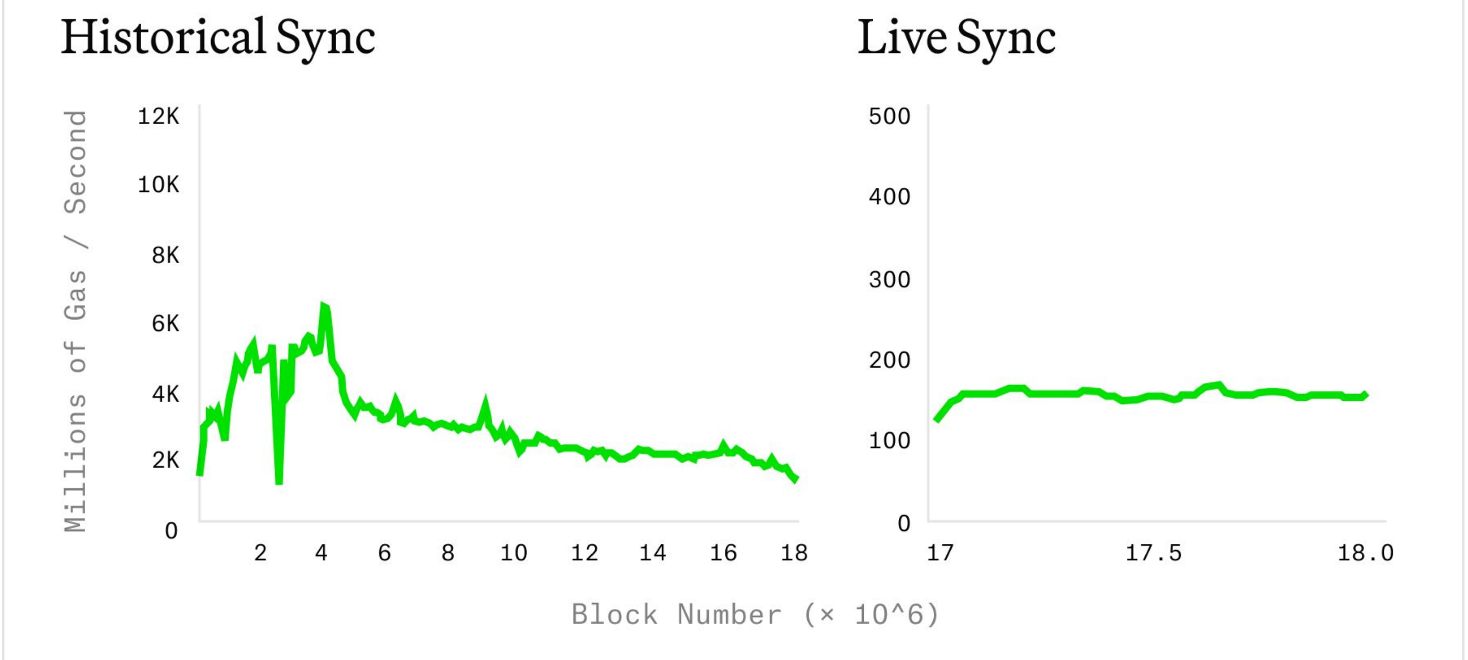
Size	Archive Node	Full Node
Total	2.01 TiB	1.025 TiB
MDBX	1.2 TiB	446.8 GiB
MDBX Freelist	8.0 GiB	7.7 GiB
Static Files	804 GiB	571.0 GiB



Benchmarks for Reth Beta | 02/27/2024

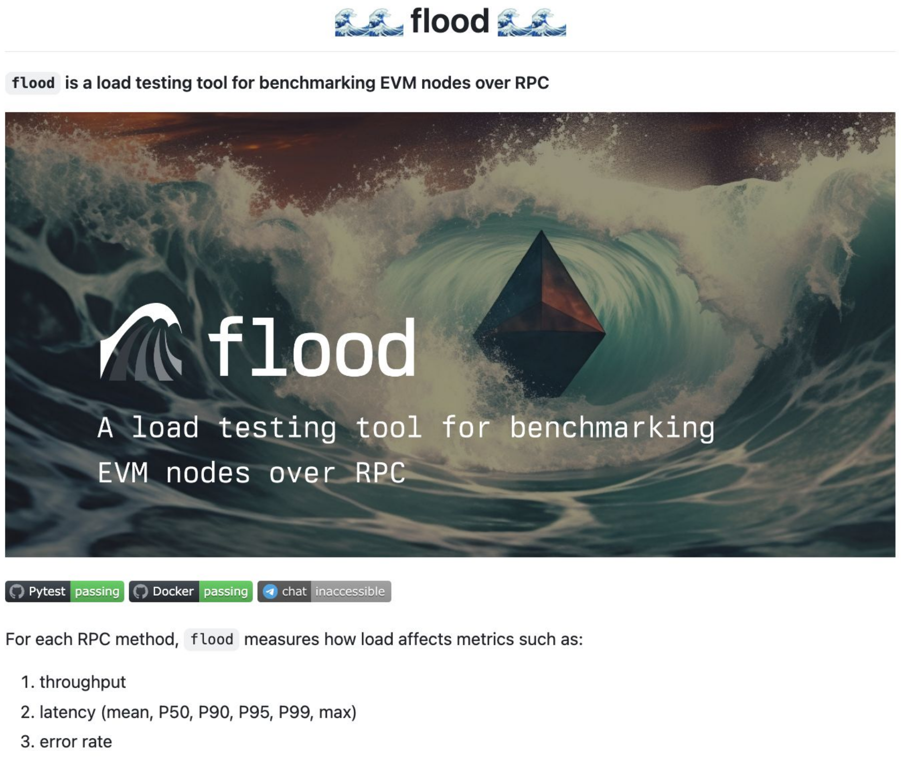
Time	Archive Node	Full Node
Total	48h 59m	42h 3m
Headers	44s	43s
Body	2h 58m	2h 13m
Sender Recovery	1h 11m	1h 14m
Execution	36h 6m	35h 49
Account Hashing	34m	34m
Storage Hashing	1h 23m	1h 9m
Merkle	38m	22m
Transaction Lookup	40m	37m
Index Storage History	3h 21m	—
Index Account History	2h 4m	—

Fast: 2-4K mg/s historical sync & 100-200mg/s at tip



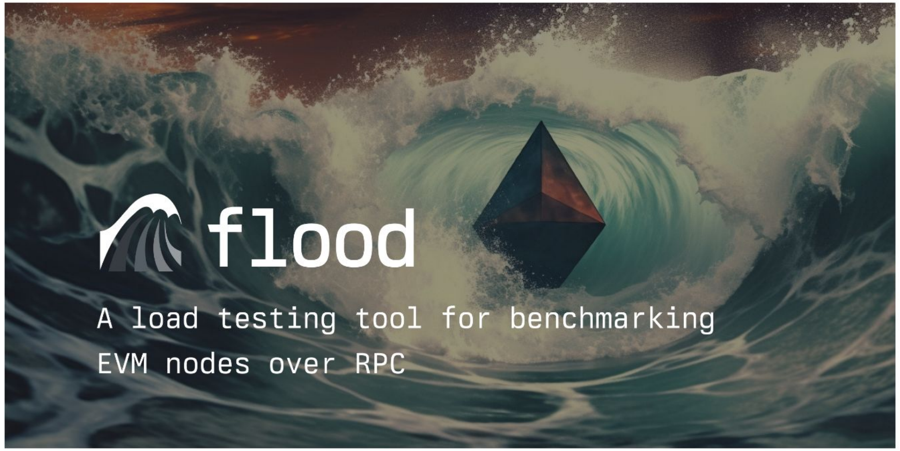
16-core CPU @ 3.1GHz, 512GB RAM	Source: MegaETH Labs	🟩 Moving Average

How do you benchmark?



🌊🌊 flood 🌊🌊

`flood` is a load testing tool for benchmarking EVM nodes over RPC

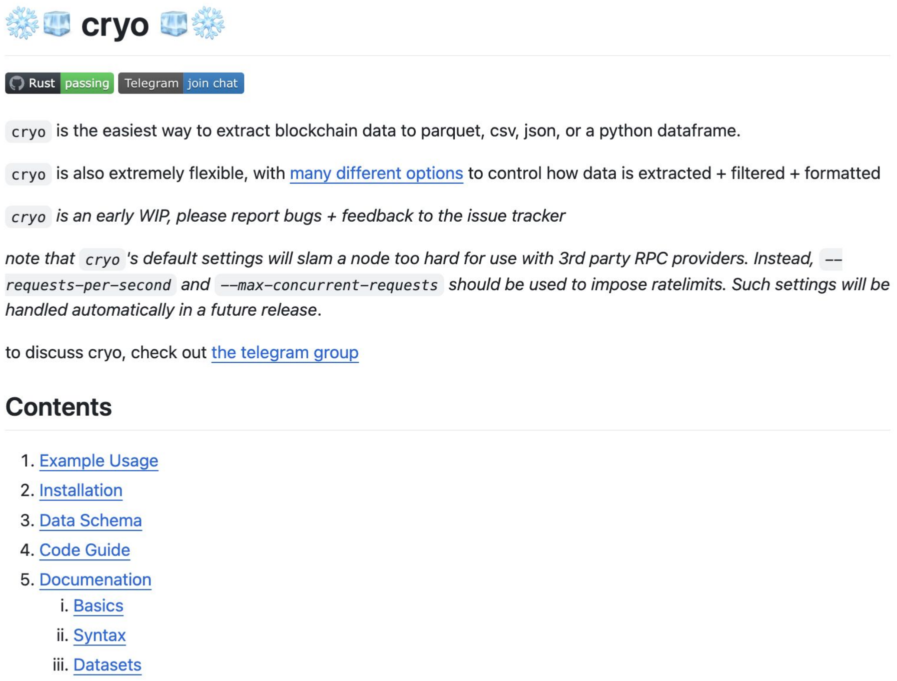


Pytest passing | Docker passing | chat inaccessible

For each RPC method, `flood` measures how load affects metrics such as:

1. throughput
2. latency (mean, P50, P90, P95, P99, max)
3. error rate

https://github.com/paradigmxyz/flood



❄️🧊 cryo 🧊❄️

Rust passing | Telegram join chat

`cryo` is the easiest way to extract blockchain data to parquet, csv, json, or a python dataframe.

`cryo` is also extremely flexible, with many different options to control how data is extracted + filtered + formatted

`cryo` is an early WIP, please report bugs + feedback to the issue tracker

note that `cryo`'s default settings will slam a node too hard for use with 3rd party RPC providers. Instead, `--requests-per-second` and `--max-concurrent-requests` should be used to impose ratelimits. Such settings will be handled automatically in a future release.

to discuss cryo, check out the telegram group

Contents

https://github.com/paradigmxyz/cryo

0.1.0 – alpha.1 → 0.2.0 beta.4

- State of the art performance on all axes except block notifications (#6286)
- Stress tested on EF’s EVM Fuzzing / Chaos Testing, for staking & RPC.
- L1: Cancun ready, tested on Devnets, Sepolia etc.
- L2: OP Stack support for usage in L2s/L3s, aka OP Reth
- Snapshot sync: https://snapshots.merkle.io
- Extensive Docs: https://paradigmxyz.github.io/reth/
- Full JSON-RPC incl. Geth-style & Parity-style traces
- Protocol Guild / RPGF Recipients for impact, to be redistributed.

Should I use this? We think so!

Is this production ready?

Raul Jordan
gm - I am running validators with reth + prysm and successfully attesting with great performance
2
1
1
Georgios Konstantopoulos owne
Raul...testnet right?
Georgios Konstantopoulos owne
Raul...testnet right?

Is this production ready?

1.0 – “Prod-ready” in May

- Reth audit with Sigma Prime (Lighthouse)
- Revm fuzzing engagement with Guido Vranken (#1 ETH Bug Bounty Leaderboard)

OK so what is the roadmap???

3 tracks + their mission:

1. Core Development: Ethereum resilience.
2. Performance: Commoditize the gigagas per second and beyond.
3. Kernel: Commoditize customizing chains and building rollups.

Core Development – Ethereum L1 resilience

- Cancun shipped!
- Ship Electra ASAP
    - EOF
    - Verkle Tries
    - Account Abstraction
- Contribute to Core Dev process with precise writing & benchmarking

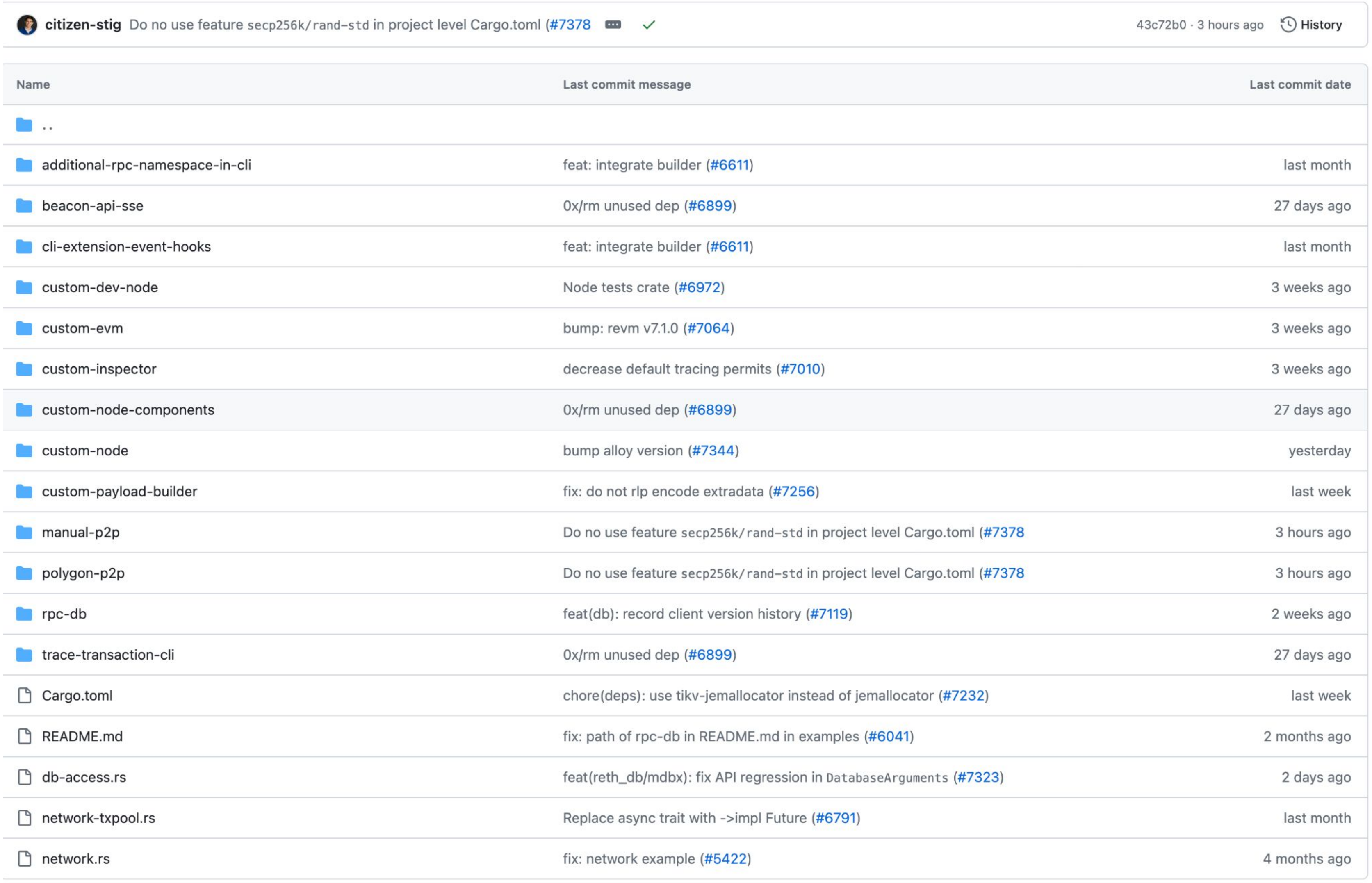
Performance – Maximize Gas Per Second

- Parallel EVM: Parallelized execution for any of historical, live sync, builder, or sequencer use cases. Each one is different!
- JIT/AOT EVM: Run native code instead of interpreter overhead.
- Optimized State Commitment: Parallelized state root calc + new algos.
- Optimal Database: Replace MDBX with Firewood/MonadDB-style DB or new perfect hash table index-based design.
- Rigorous Benchmarking: Regression testing in CI, aggressive systems optimizations.

Reth Kernel – State of the art EVM infra in <1 day

Reth Kernel is the SDK for building bleeding-edge EVM infrastructure:

- Node Builder API: Pluggable components, stop forking nodes, `use reth::cli::*`
- Library Usage: Build P2P Crawlers, Indexers, simulations APIs, MEV bots etc.
- Import Rust node infra and run as 1:
  - Rethhouse = Reth + Lighthouse = CL + EL in 1 binary
  - Reth + Helios = Light client CL + EL in 1 binary
  - OP Reth + Reth + Helios + Magi = L1 CL/EL + L2 CL/EL in 1 binary
- *Project Idea: Testnet Rollup w/ custom EIPs?*
- Tons of examples in the repository already!



citizen-stig Do no use feature `secp256k/rand-std` in project level Cargo.toml (#7378 ··· ✓ 43c72b0 · 3 hours ago History

Name	Last commit message	Last commit date
..		
additional-rpc-namespace-in-cli	feat: integrate builder (#6611)	last month
beacon-api-sse	0x/rm unused dep (#6899)	27 days ago
cli-extension-event-hooks	feat: integrate builder (#6611)	last month
custom-dev-node	Node tests crate (#6972)	3 weeks ago
custom-evm	bump: revm v7.1.0 (#7064)	3 weeks ago
custom-inspector	decrease default tracing permits (#7010)	3 weeks ago
custom-node-components	0x/rm unused dep (#6899)	27 days ago
custom-node	bump alloy version (#7344)	yesterday
custom-payload-builder	fix: do not rlp encode extradata (#7256)	last week
manual-p2p	Do no use feature `secp256k/rand-std` in project level Cargo.toml (#7378	3 hours ago
polygon-p2p	Do no use feature `secp256k/rand-std` in project level Cargo.toml (#7378	3 hours ago
rpc-db	feat(db): record client version history (#7119)	2 weeks ago
trace-transaction-cli	0x/rm unused dep (#6899)	27 days ago
Cargo.toml	chore(deps): use tikv-jemallocator instead of jemallocator (#7232)	last week
README.md	fix: path of rpc-db in README.md in examples (#6041)	2 months ago
db-access.rs	feat(reth_db/mdbx): fix API regression in `DatabaseArguments` (#7323)	2 days ago
network-txpool.rs	Replace async trait with ->impl Future (#6791)	last month
network.rs	fix: network example (#5422)	4 months ago

Beyond??

Reth 2.0: “Rethink nodes for rollups in the cloud”

- BigQuery / Amazon Aurora moment for L2s – “disaggregated compute and storage”
- Multi-rollup node: Rollup as post-execution hook.
  - Each rollup you run requires adding 2 services (op-node equivalent + op-reth equivalent)
  - Today: `reth node –chain=mainnet`
  - Tomorrow: `reth node` **`–chains=mainnet,op-mainnet,base-mainnet,zora, …`**
- Multi-tenant node architecture
  - Based on Actor model
  - Built for cloud usage
  - Separate compute from storage
- Multi-rolllup + multi-tenant → low devops cost while scaling up → “true” elastic scalability

Paradigm

Reth
Made in Rust

Q&A?

What do you want to build on Reth?

Are you running a node?

Reach out: georgios@paradigm.xyz / @gakonst

Book: https://paradigmxyz.github.io/reth/
Rust Docs: https://paradigmxyz.github.io/reth/docs/
Repo: https://github.com/paradigmxyz/reth
Telegram: https://t.me/paradigm_reth