We are excited to announce that after21 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.

TL;DR

- Supported Networks & HardForks

: * Ethereum: Cancun

- OP Stack: Ecotone
- No other networks planned, please reach out with specs and ideas.
- Ethereum: Cancun
- OP Stack: Ecotone
- No other networks planned, please reach out with specs and ideas.
- Prod-readiness:

Audits are underway, out before end of Q2

- Performance (Feb 27th):
- TPS: Live sync benchmarks to >1K TPS, historical sync >10K TPS.
- RPC: State of the art in every call except eth_subscribe newHeads

which we are investigating.

- Sync Time: Our fastest archive sync is now at 49 hours, and we have another ~10-20% to go in the coming weeks. Non-archive sync with execution from genesis is now at 42 hours. Both state of the art.
- DB Size: Our archive node database sits at just over 2TB (state of the art), and our full node at just over 1TB (geth/nethermind are better by ~100GB). Similarly, we have another 10-20% improvements to get here in the next releases.
- TPS: Live sync benchmarks to >1K TPS, historical sync >10K TPS.

- RPC: State of the art in every call except eth_subscribe newHeads

which we are investigating.

- Sync Time: Our fastest archive sync is now at 49 hours, and we have another ~10-20% to go in the coming weeks. Non-archive sync with execution from genesis is now at 42 hours. Both state of the art.
- DB Size: Our archive node database sits at just over 2TB (state of the art), and our full node at just over 1TB (geth/nethermind are better by ~100GB). Similarly, we have another 10-20% improvements to get here in the next releases.
- Ethereum: Cancun
- OP Stack: Ecotone
- No other networks planned, please reach out with specs and ideas.
- TPS: Live sync benchmarks to >1K TPS, historical sync >10K TPS.
- RPC: State of the art in every call except eth_subscribe newHeads

which we are investigating.

- Sync Time: Our fastest archive sync is now at 49 hours, and we have another ~10-20% to go in the coming weeks. Non-archive sync with execution from genesis is now at 42 hours. Both state of the art.
- DB Size: Our archive node database sits at just over 2TB (state of the art), and our full node at just over 1TB (geth/nethermind are better by ~100GB). Similarly, we have another 10-20% improvements to get here in the next releases.

Introduction

Last year, we released Reth alpha.1, the first version of our modular, contributor-friendly, and blazing fast Ethreum execution node, written in Rust.

Our goals behind the Reth project remain the same:

1. Provide a great execution client for Ethereum mainnet's resilience and long-term innovation.
2. Create an ecosystem of polished libraries for developers to build a new wave of EVM-centric infrastructure, including Rollups, RPCs, P2P services, Indexers, and more.

Today, we're excited to announce Reth v0.2.0, the beginning of our Beta release cycle.

Reth Beta is Cancun-ready, syncs from genesis, keeps up with the tip, performs extremely well under heavy RPC load for both point and range queries, and has improved validator performance. Find the release here.

This release includes breaking changes that are not compatible with the nodes synced onalpha versions. Alpha versions will be released with only critical fixes. All users are recommended to upgrade to the Beta release and resync.

Read our docs on how to do that.

We are also releasing alpha.22 as a patch release to the alpha series. All alpha node operators should update; this is not a breaking change.

Adoption Update

Despite being alpha software, Reth is found inabout 2-3% of all Ethereum nodes. We are excited for this number to grow with the beta release. Our Github repository is nearing 3000 stars, putting us behind go-ethereum's massive 45K stars (omitted from the chart below—a testament to go-ethereum's massive success). Finally, the most important metric for us is the 244 unique contributors to Reth, as that is part of our mission of growing the number of developers intimately familiar with Ethereum Protocol implementations.

Features & Stability Update

Reth Beta comes with a lot of exciting features and stability improvements.

- Cancun Ready

- Reth is ready for Ethereum mainnet's Cancun hardfork on March 13th.
- Reth has participated in multiple rounds of devnets and shadow forks that have surfaced bugs related to cancun EIP implementations. Big shoutout to the Ethereum Foundation's DevOps team—truly an amazing team to be working with.
- Fun fact: Reth was used to send the first blob on holesky: https://holesky.blobscan.com/block/894735.
- Reth is ready for Ethereum mainnet's Cancun hardfork on March 13th.
- Reth has participated in multiple rounds of devnets and shadow forks that have surfaced bugs related to cancun EIP implementations. Big shoutout to the Ethereum Foundation's DevOps team—truly an amazing team to be working with.
- Fun fact: Reth was used to send the first blob on holesky: https://holesky.blobscan.com/block/894735.
- OP Ecotone Ready
- OP Reth is ready for the upcoming Ecotone hardfork on Base Mainnet, Zora Network & other OP Stack networks.
- Reth started supporting the OP Stack November 2023 via a collaboration with the OP Labs and BASE teams. We think this is critical for supporting the Optimism Collective and the Rollup-centric roadmap.
- We do not recommend updating to beta.1 yet for Ecotone as we're still improving edge cases around the interaction between static files and op-node. OP Stack users should use alpha.21, until beta.2 is out.
- OP Mainnet support is underway and expected before Reth 1.0.
- OP Reth is ready for the upcoming Ecotone hardfork on Base Mainnet, Zora Network & other OP Stack networks.
- Reth started supporting the OP Stack November 2023 via a collaboration with the OP Labs and BASE teams. We think this is critical for supporting the Optimism Collective and the Rollup-centric roadmap.
- We do not recommend updating to beta.1 yet for Ecotone as we're still improving edge cases around the interaction between static files and op-node. OP Stack users should use alpha.21, until beta.2 is out.
- OP Mainnet support is underway and expected before Reth 1.0.
- Full JSON-RPC Support
- Reth has full JSON RPC support, including the debug_

namespace with Javascript tracers and the trace_

namespace. See our docs.

- We have been making incremental bug fixes here. Our top priority has been accuracy of the results, in particular around gas consumption in tracing.
- Reth has full JSON RPC support, including the debug_

namespace with Javascript tracers and the trace_

namespace. See our docs.

- We have been making incremental bug fixes here. Our top priority has been accuracy of the results, in particular around gas consumption in tracing.
- Database Breaking Changes
- We are entering V2 of our Database (reth db version

), bringing exciting improvements, at the cost of breaking the format and requiring a resync. We do not expect more of these in the near future and we appreciate users' patience.

- We released Static Files and ETL for better separation of mutable/immutable data and faster initial sync.
- Initial sync time reduced by 4-5 hours, getting close to the sub-48h mark.
- Access speed to historical data increased by up to 2x-10x.
- Disk usage for historical data reduced by up to 10%.

- Ability to offload the historical static files to a separate read-only drive and serve it to multiple nodes.
- Initial sync time reduced by 4-5 hours,getting close to the sub-48h mark.
- Access speed to historical data increased by up to 2x-10x.
- Disk usage for historical data reduced by up to 10%.
- Ability to offload the historical static files to a separate read-only drive and serve it to multiple nodes.
- We made database schema improvements:
- Switch to Roaring Bitmaps encoding for historical access indices resulted in almost 50% disk space reduction (10% of total database size) and improved querying speed.
- Nicer database interface (better table names, removal of redundant wrappers).
- Switch to Roaring Bitmaps encoding for historical access indices resulted in almost 50% disk space reduction (10% of total database size) and improved querying speed.
- Nicer database interface (better table names, removal of redundant wrappers).
- We are entering V2 of our Database (reth db version

), bringing exciting improvements, at the cost of breaking the format and requiring a resync. We do not expect more of these in the near future and we appreciate users' patience.

- We released Static Files and ETL for better separation of mutable/immutable data and faster initial sync.
- Initial sync time reduced by 4-5 hours,getting close to the sub-48h mark.
- Access speed to historical data increased by up to 2x-10x.
- Disk usage for historical data reduced by up to 10%.
- Ability to offload the historical static files to a separate read-only drive and serve it to multiple nodes.
- Initial sync time reduced by 4-5 hours,getting close to the sub-48h mark.
- Access speed to historical data increased by up to 2x-10x.
- Disk usage for historical data reduced by up to 10%.
- Ability to offload the historical static files to a separate read-only drive and serve it to multiple nodes.
- We made database schema improvements:
- Switch to Roaring Bitmaps encoding for historical access indices resulted in almost 50% disk space reduction (10% of total database size) and improved querying speed.
- Nicer database interface (better table names, removal of redundant wrappers).
- Switch to Roaring Bitmaps encoding for historical access indices resulted in almost 50% disk space reduction (10% of total database size) and improved querying speed.
- Nicer database interface (better table names, removal of redundant wrappers).
- Resource Usage Improvements
- We reworked the networking interface, which has speeded up Reth's responsiveness to I/O, made it more DoS resistant, and improved the performance of the transaction pool when deleting blobs or evicting transactions.
- We fixed a lot of resource leakages, such as amemory leak in our Transaction Pool.
- We addressed all our pain points around MDBX's freelist such as leaky and timing out long-running DB read-transactions that caused uncontrolled freelist growth. The freelist's size also no longer affects the performance of write transactions, prioritizing the validator performance.
- We reworked the networking interface, which has speeded up Reth's responsiveness to I/O, made it more DoS resistant, and improved the performance of the transaction pool when deleting blobs or evicting transactions.
- We fixed a lot of resource leakages, such as amemory leak in our Transaction Pool.

- We addressed all our pain points around MDBX's freelist such as leaky and timing out long-running DB read-transactions that caused uncontrolled freelist growth. The freelist's size also no longer affects the performance of write transactions, prioritizing the validator performance.
- Reth is ready for Ethereum mainnet's Cancun hardfork on March 13th.
- Reth has participated in multiple rounds of devnets and shadow forks that have surfaced bugs related to cancun EIP implementations. Big shoutout to the Ethereum Foundation's DevOps team—truly an amazing team to be working with.
- Fun fact: Reth was used to send the first blob on holesky: https://holesky.blobscan.com/block/894735.
- OP Reth is ready for the upcoming Ecotone hardfork on Base Mainnet, Zora Network & other OP Stack networks.
- Reth started supporting the OP Stack November 2023 via a collaboration with the OP Labs and BASE teams. We think this is critical for supporting the Optimism Collective and the Rollup-centric roadmap.
- We do not recommend updating to beta.1 yet for Ecotone as we're still improving edge cases around the interaction between static files and op-node. OP Stack users should use alpha.21, until beta.2 is out.
- OP Mainnet support is underway and expected before Reth 1.0.
- Reth has full JSON RPC support, including the debug_

namespace with Javascript tracers and the trace_

namespace. See our docs.

- We have been making incremental bug fixes here. Our top priority has been accuracy of the results, in particular around gas consumption in tracing.
- We are entering V2 of our Database (reth db version

), bringing exciting improvements, at the cost of breaking the format and requiring a resync. We do not expect more of these in the near future and we appreciate users' patience.

- We released Static Files and ETL for better separation of mutable/immutable data and faster initial sync.
- Initial sync time reduced by 4-5 hours, getting close to the sub-48h mark.
- Access speed to historical data increased by up to 2x-10x.
- Disk usage for historical data reduced by up to 10%.
- Ability to offload the historical static files to a separate read-only drive and serve it to multiple nodes.
- Initial sync time reduced by 4-5 hours, getting close to the sub-48h mark.
- Access speed to historical data increased by up to 2x-10x.
- Disk usage for historical data reduced by up to 10%.
- Ability to offload the historical static files to a separate read-only drive and serve it to multiple nodes.
- We made database schema improvements:
- Switch to Roaring Bitmaps encoding for historical access indices resulted in almost 50% disk space reduction (10% of total database size) and improved querying speed.
- Nicer database interface (better table names, removal of redundant wrappers).
- Switch to Roaring Bitmaps encoding for historical access indices resulted in almost 50% disk space reduction (10% of total database size) and improved querying speed.
- Nicer database interface (better table names, removal of redundant wrappers).
- Initial sync time reduced by 4-5 hours, getting close to the sub-48h mark.
- Access speed to historical data increased by up to 2x-10x.
- Disk usage for historical data reduced by up to 10%.
- Ability to offload the historical static files to a separate read-only drive and serve it to multiple nodes.

- Switch to Roaring Bitmaps encoding for historical access indices resulted in almost 50% disk space reduction (10% of total database size) and improved querying speed.
- Nicer database interface (better table names, removal of redundant wrappers).
- We reworked the networking interface, which has speeded up Reth's responsiveness to I/O, made it more DoS resistant, and improved the performance of the transaction pool when deleting blobs or evicting transactions.
- We fixed a lot of resource leakages, such as amemory leak in our Transaction Pool.
- We addressed all our pain points around MDBX's freelist such as leaky and timing out long-running DB read-transactions that caused uncontrolled freelist growth. The freelist's size also no longer affects the performance of write transactions, prioritizing the validator performance.

Performance Update

We benchmark Reth (beta.1 & alpha.21), Geth (v1.14.0) and Nethermind (v1.25.4) nodes onLatitude c3.large bare metal instances. We invite people to help us extend our benchmarks further to other node configurations and hardware. We are excited to share preliminary results below from our most recent benchmarking adventure.

How much gas and transactions per second (tps) can Reth support?

The MegaETH Labs team profiled Reth alpha.13 (performance has improved since then) recently on large machines1

, and they show an encouraging >1-2K MGas/s (>10K TPS @ 100K gas/tx) result on historical sync in last 1M blocks (i.e. infrequent trie updates) and >100-200 MGas/s (>1K TPS) result on live sync

. For perspective, Ethereum mainnet does ~12.5 TPS, Polygon ~70 TPS, Solana ~700 TPS (source). Find their full presentation from ETHDenver on this here for a complete picture.

We want to make it extremely clear that the historical sync numbers represent pure execution time during "backfills", not including senders recovery or merklization. While this is not representative of the load at tip, it is important for bootstrapping archive nodes quickly and for catching up to the chain between node restarts.

The second diagram is produced by running the live sync logic on recent Ethereum blocks, doing senders recovery, execution and merklization. Interestingly enough, the Merkle Patricia Trie takes 80-90% of time. This means that there is still a lot of room for improvement on our performance at the tip due to the merklization overhead! It also poses an interesting question around how much we should tune Ethereum & other blockchains' state commitments for performance. What will that overhead look like with Verkle Tries for example? We don't know, but we are excited to find out.

We acknowledge that Reth's JSON RPC subscription latency for the block hash at the chain's tip (eth_subscribe

- newHeads

) is still not state of the art. We are working on improving this in the next beta releases withparallel/async storage root calculations, as well as faster chain head events.

How fast does it sync and how big is the database?

The static files & ETL releases let us improve on both the sync time and the space required by the node. Since alpha.1, we also released support for configurable pruning, with the most common configuration across non-archival Ethereum nodes exposed behind the --full

flag. This flag still stores all history (headers/transactions/receipts) but prunes any historical changesets and their corresponding indices.

One notable improvement here is the reduction of the time to execute the TransactionLookup

stage from multiple hours to 40 minutes, using theETL technique pioneered by Erigon, which minimizes write amplification. This is on bleeding edge NVMe drives, and we expect the performance improvement to be even higher in lower end hardware where TransactionLookup

used to take >1 day. We intend to apply the same technique in other stages (e.gIndexStorage/AccountHistory) and expect similar improvements in the next beta releases.

How fast is the RPC?

We used Cryo, a high performance tool we built to ETL data out of blockchains, as a reference of what performance

improvement a user of Reth would experience in their day-to-day. We benchmarked over localhost on various block ranges, illustrated in the tables below.

First, we benchmarked the time required to extract blocks from a node, without their transactions (eth_getBlock

with full = false

).

Then, we investigated the performance of also extracting the transactions (eth_getBlock

with full = true

) using both point receipt queries (eth_getTransactionReceipt

) and batch queries (eth_getBlockReceipts

).

We investigated long eth_getLogs

range queries, as these are pertinent for all indexing use cases.

Finally, we benchmarked the debug_traceBlockByNumber

endpoint using the preStateTracer

for the code_reads

workload and the callTracer

for the geth_calls

workload:

At the time of the benchmark, Reth outperforms the tested clients in all RPC queries. We want to appropriately caveat that our benchmarks are representative of our experiments and may vary across hardware. We also expect other node implementations to continuously improve, so we want to be explicit about our numbers being a 'point in time' measurement. Benchmarking is hard, and we are working towards reproducible cross-client nightly benchmarks on CI. Please reach out if you'd like to work together on this.

We contacted the Nethermind team regarding the tracing performance and we attributed it to their usage of the Javascript tracer instead of native implementations like in Reth and Geth. This is known and part of their backlog, and we will be updating our numbers when this is done. We did not provision Erigon nodes for this benchmark, we intend to do so in the coming weeks to stress test the performance of the trace_

endpoints, but Erigon node operator feedback and preliminary benchmarks from a few alpha releases ago have been encouraging.

The gains from the static files are remarkable in range queries, as static files were optimized for such access patterns, whereas our random access performance is ~unchanged due to usage of offset-based indices in the static files (the lacking support of which prevented us from using parquet here).

We also want to give a big shoutout to theTokio team for building a high performance and reliable substrate to build all our async infrastructure on, and is doing a lot of the heavy lifting here.

We are encouraged by our performance, and are keen to keep benchmarking and uncover areas for further optimization in all Ethereum clients.

When production-ready?

Reth has been undergoing rigorous battle-testing over the last months using our own security tooling, third parties running Reth in production, as well as being integrated in ecosystem tooling such as Hive, goevmlab, Kurtosis, and assertoor, among others. We are always interested in building more tools for testing and fuzzing (e.g. for networking), please reach out if you're interested in collaborating.

As a result, we actively recommend professional node operators to switch to Reth in production for performance and cost reasons in use cases where high performance with great margins is required such as Rollups, RPC, MEV, Indexing, Simulations, and P2P activities.

While we are aware of parties running Reth staking nodes in production, we do not encourage usage in production staking

environments by non-professionals yet, but are available to support without warranty or liability.

We have contracted Sigma Prime, the authors of Lighthouse (Rust Consensus Layer), to perform an audit on Reth's critical components. We acknowledge it is hard to make security claims about Ethereum nodes, given the large surface area of the codebases. We also believe that expert reviews on the most important parts of the codebase such as the Engine API, Merkle Patricia Trie or the P2P Network are a requirement for convincing ourselves that our implementation of the Ethereum protocol is sound. The audit is underway.

We also are collaborating with Guido Vranken, #1 on the Ethereum Bug Bounty leaderboard, on a community-driven audit of Revm. Revm is the Rust EVM behind Reth and we view it as critical infrastructure for Ethereum. We are excited for this review to surface potential worst-case scenarios for Reth's performance and stability, and hope that its results strengthen the entire ecosystem. The audit is underway, and there will be a separate announcement for this later in the year.

Once both audits are complete and we have addressed all the finds, we intend to release Reth 1.0 by the end of Q2, our first production-ready release which will be recommended for usage in staking operations.

Post 1.0 Reth releases will follow the semver convention.

What's the future of Reth?

On Layer 1, Reth is meant to contribute towards client diversity and help experiment with new technologies for the Ethereum protocol. We have executed on this so far with the Cancun upgrade, and are excited to continue this with the Prague/Electra hardfork – see our view on the EIPs considered for inclusion.

But what we are most excited about is about Reth's modularity and its proliferation in EVM-centric infrastructure. We are already seeing early signs of that with usage of Reth in the Paradigm portfolio as a way to provide faster, cheaper, more resilient services than what conventional libraries offer. We are thrilled to see Reth adoption on Layer 2s such as OP Stack, for example, companies are already adopting Reth RPCs in production for Base.

We are still early in our execution of our ambitious roadmap, and we cannot wait to share more about the future of Reth as an SDK. The L1/L2 nodes are just the first application.

If this sounds exciting to you, please reach out to georgios@paradigm.xyz. We're looking for the best engineers in the world to execute on this together. Until then, see you on Github.

Footnotes

1. Intel® Xeon W5-2465X 16 core @ 3.1GHz, Samsung 512G DDR5-4800, Intel SSD D7-P5510 7.68 TB NVMe PCIe 4.0 ↩

Intel® Xeon W5-2465X 16 core @ 3.1GHz, Samsung 512G DDR5-4800, Intel SSD D7-P5510 7.68 TB NVMe PCIe 4.0↩