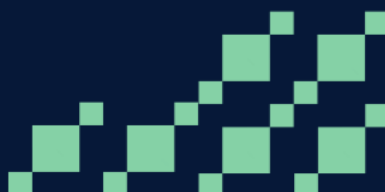
AMARU PROJECT: 2025 RETROSPECTIVE

30 DECEMBER 2025



Contents

Context

This document covers the 2025 retrospective about the initial objectives set for the Amaru project in Q1 2025.

2025 Retrospective

In 2025, we accomplished significant advances in consensus, ledger state management, simulation testing, governance, and network architecture. At the same time, we deviated from our original plan (https://pragma.io/projects/amaru/#journey) in a few areas (UTxO-RPC, P2P networking, ...).

In this section, we reflect upon what was planned and what has been done, following the previously presented roadmap. We'll highlight items that have been delivered : 🟢, those underway or partially done : 🟠, those abandoned/cancelled : 🟪 , and items that are behind schedule and require attention: 🔴. We will also highlight a few extra items: ⭐ that were unplanned but happened regardless.

Relay Node (initial target: Q2 2025)

🟢 Ledger: On-disk State Tracking

- **Full on-disk ledger state:** On-disk snapshots with minimal in-memory footprint, including notably:
    a. (e)UTxOs
    b. Stake pools and stake credential registrations/deregistrations
    c. Accounts, consensus rewards calculations, and distribution
    d. DReps and constitutional committee members
    e. Protocol parameters
    f. Governance proposals and votes

Ledger: (almost) Full Validation

- **Partial Phase-1 validations**: all transaction validation rules are mostly implemented, but a few straightforward checks must still be implemented. More hardening is also required to ensure 100% behavioral compatibility with the Haskell node.
- **Partial Phase-2 validations**: new Plutus Virtual Machine integrated with the ledger, with phase-2 evaluation implemented, yet still under testing and refinement.

Ledger: Governance Rules Implementation

- **Governance ledger state**: Governance, including voting stake distribution(s) and ratification/enactment of all governance actions
- **Proposals forest**: Tracking of governance proposals as an optimized forest of interdependent proposals
- **Hard forks**: Protocol upgrades (hard forks) and support for multi-version ledger and dynamic protocol parameters.

⭐ Ledger: Conformance Testing & Blueprint

- **Conformance testing**: creation of various conformance test scenarios from the reference Haskell implementation and/or specification, including snapshot tests from mainnet/preprod/preview data.
- **Cardano Blueprint**: contributions to the Cardano blueprint (https://github.com/cardano-scaling/cardano-blueprint) to extend foundational knowledge about Cardano.

Consensus: Conformance & Simulation Testing

- **Pure-stage actor and effects library**: Complete transition from an initial proof-of-concept to a robust event processing pipeline architecture:
    a. abstracts external effects for isolation and testing, including disk I/O, network, and time
    b. supports both a deterministic scheduler for precise control of thread interleaving and standard non-deterministic parallel scheduling for production

- **Simulation Framework:** Development of a deterministic simulation testing system inspired by Jepsen and IOG's io-sim. At the moment, simulations cover mainly chain selection from a node under test connected to multiple upstream peers.
- **Data Generation**: Creation of synthetic block trees to test arbitrary consensus scenarios, including randomization of message delivery.
- **Visualization**: - HTML tools to visualize execution traces and troubleshoot intricate distributed scenarios.
- **Reproducibility:** Support for seeds and full trace replay for test reproducibility.
- **Antithesis Testing:** Integration with the [Antithesis](https://antithesis.com) testing platform. We've been mostly experimenting with the Haskell cardano-node, identifying and reporting subtle issues as we pave the way for Amaru to be used alongside Haskell nodes in Antithesis clusters.

UTxO RPC Implementation

This work item was dropped from the roadmap early on, as we've decided to limit the number of integrated APIs in Amaru to avoid duplication of effort across the ecosystem. Our strategy is to rely on middleware (software sitting between Amaru and client applications) to provide those functionalities. In the case of UTxO-RPC, that means, for example, software like TxPipe's Dolos or Blink Labs' Dingo, which can provide client-level interfaces while relying on Amaru as a trusted data source.

Moreover, we've been deprioritizing these integrations to better focus on core development, stability, and the conformance of Amaru's internals with the Haskell implementation.

SPO Involvement

We've made significant progress toward getting Amaru ready for external consumption by SPOs by providing **multi-architecture Docker images & static executables** through our continuous integration pipeline.

However, we've only recently started reaching out to and engaging with SPOs. A few brave souls responded positively and are already deploying and monitoring Amaru in their environments. This is a workstream that needs to be continued, especially as we finalize the networking stack and the work around the mempool.

🟠 Networking: Inbound & outbound discovery

- **Full Node-to-Node Protocols**: Implementation of Cardano's handshake and node-to-node protocols:
  - 🟢 chain-sync (initiator & responder)
  - 🟢 block-fetch (initiator & responder)
  - 🟢 keep-alive (initiator & responder)
  - 🟠 tx-submission (initiator); the responder side is under development.
  - 🔴 peer-sharing not implemented yet

- **Static Network Topology**: Amaru currently supports only a static network topology (albeit from multiple peers) and no peer selection. This is sufficient for a variety of local use cases (e.g., powering a DApp), but not for qualifying as a proper relay.

Overall, the networking stack has experienced unforeseen delays due to a lack of resources, mainly because everyone was busy with other tasks. We have been focusing on aspects that could make Amaru usable, while leaving dynamic peer-to-peer management and peer sharing for later.

We are also currently hardening network protocols by implementing buffer limits and strict CBOR parsing rules, as per the networking specifications produced by the Haskell team.

Block Producing Node (initial target: Q3 2025)

🟢 Consensus: Chain validation

- **Chain Selection**: Implementation of the Ouroboros chain selection algorithm using an optimized data structure for header storage and on-disk navigation.
- **Full Header Validation**: Complete support of VRF/KES cryptographic verifications and checks, including the stake-based leader schedule of Cardano.

Ledger: Full Validations

- **Remaining Ledger Rules**: As mentioned above, some (straightforward) ledger rules remain. We have de-risked the most critical rules to ensure no “unknown unknowns” remain, but we still need to address the simple checks we left behind for time's sake.

- **Ledger Store Refinement**: We have been implementing Amaru’s ledger store based on an initial design that has shown some shortcomings over time. We’ve been waiting to get the full picture before attempting to re-design some of it to address those shortcomings. We now have a clear path for the changes we require, but they haven’t happened yet.

UTxO RPC: Use case implementation

As previously indicated, the UTxO RPC work item has been deprioritized and eventually dropped. We want to achieve similar outcomes by integrating with middlewares that provide a variety of functionalities, including UTxO-RPC support.

Public Testnet Validation

- **Continuous e2e tests**: We have been continuously synchronizing the Amaru chain across Haskell peer instances on both testnets (Preview and Preprod), as well as private node clusters. Every header is validated, and the state at the end of every epoch is compared with that of a Haskell node.

- **Real-World Deployments**: Many team members are running (so far, read-only) Amaru nodes connected to Mainnet, Preview, and Preprod. Some SPOs are also integrating Amaru into their stacks, connected to production-grade monitoring systems. The Pi examples at the summit were another occurrence of such real-world deployments.

Mempool: Complex mempool implementation

We’re running late on our mempool implementation. One of the key reasons is the various discussions happening around Ouroboros Leios. A large chunk of the work about Leios revolves around the mempool and the rapid re-validation of transactions.

Since the design of (Linear) Leios is still an ongoing discussion, we've been de-prioritizing this chunk of work as well, so that we can focus on delivering a Leios-ready mempool from the start.

Block production

Without an adequate mempool and without peer management at the networking level, the block production makes little sense. Given the delays we've had on the networking side, we've been postponing the implementation of the block-production capabilities while focusing on other elements further down the path (performances, observability, etc.).

Also, given recent events on Mainnet, we think it is important that we only allow block production once we are extremely confident that the risk of Mainnet's operation is as low as possible. We have identified areas that require more testing and hardening. It's also important to keep in mind that often, hardening benefits both Amaru and the Haskell cardano-node (we have discovered and reported bugs in the past that could have impacted Mainnet).

Soft-forks & performance improvements (initial target: Q4 2025)

Soft forks

There have been several discussions around that topic during the node diversity workshops, and a genuine motivation from all node implementors to coordinate and collaborate on future updates.

Yet there is still no clear plan for the form this will take. So for now, this remains a work in progress at the exploratory phase.

Performances

- **Continuous Performance Analysis**: We continuously monitor Amaru's internal performance (heap analysis, resident memory usage, CPU usage, I/O reads & writes, ...) and seek ways to reduce bottlenecks and further optimize operations.

- ⭐ **Amaru on Pi**: We attended the 2025 Cardano Summit and set a challenge to ourselves: make people feel what low hardware requirements mean by deploying Amaru on constrained hardware:
    - **Low memory usage**: we deployed Amaru on a variety of Raspberry Pis, down to the Pi Zero (with only ~400 MB of available RAM).
    - **Low energy**: we showcased Amaru's energy efficiency by powering it for hours using small batteries (~5000 mAh).
    - **First-class ARM**: We demonstrated how Amaru can run on native ARM – an architecture often desired by SPOs.

- 🟢 **UPLC Benchmarks**: Validating Amaru's Virtual Machine performances
    - **Plutus benchmarks:** Implemented the benchmarks suite from the Haskell Plutus team, comparing results with the Haskell implementation, ensuring better or similar performances.
    - **"Turbo" benchmarks**: Implemented the benchmarks suite created as part of the TurboCardano project, obtained by sampling real-world smart contracts on mainnet across all Plutus versions.

Truth is, we are already satisfied with Amaru's current performances. While we haven't yet deeply explored more performance-intensive paths, the (conscious) design choices we've made along the way have led to good-enough performance out of the box.

In 2026, we want to dedicate time to look even further into it, while preserving data integrity and without ever cutting on security considerations.

🟢 Better Operator Interface

We have already implemented an interesting set of capable monitoring tools that integrate nicely with industry-grade observability stacks:

- 🟢 **Structured Tracing**: Integration of structured tracing, logging, and metrics exported using OpenTelemetry.

- 🟢 **Metrics**: Collection of detailed metrics for all stages compatible with existing tooling (e.g. `gLiveView`).
- 🟢 **JSON logs**: Support for structured logs for analysis and debugging.
- 🟢 **Distributed Spans**: Tracing of operations across different stages.
- ⭐ **Amaru doctor**: A TUI for inspecting Amaru's internals and visualizing execution.

🔴 However, the work around metrics and tracing will only truly be completed after several rounds of feedback with SPOs running and monitoring the live system. Based on conversations we had with SPOs, we also intend to deliver more SPO-specific command-line tools (such as ways to craft on-chain updates to pool parameters or on-chain votes on governance proposals).

Workflow

🟢 Transparent Development & Treasury Management

- **Entirely open source**: Amaru is an open-source project whose source code currently lives on GitHub (https://github.com/pragma-org/amaru-treasury) with an active community conversing on PRAGMA's Discord (https://discord.gg/q5XZMZhrUY). We also have a development log (https://github.com/pragma-org/amaru/wiki) that records daily activities and explorations continuously.

- **Public meetings**: bi-weekly (public) maintainer committee meetings for alignment, progress report and Q&A.

- **Public demos**: bi-monthly (public) touchpoints with presentations and live demos from developers to showcase visible deliveries:
    - 2025/11 | Simulating executions and pre-summit preparation (https://pragma.io/projects/amaru/#Deterministic%20simulations%2C%20ledger%20rules%26%20%20P2P%20Networking)
    - 2025/09 | More ledger rules, distributed traces and conformance testing (https://pragma.io/projects/amaru#More%20ledger%20rules%2C%20distributed%20traces%20and%20conformance%20testing)
    - 2025/06 | Ledger rules, time travelling & node-to-node conversations (https://pragma.io/projects/amaru#Ledger%20rules%2C%20time%20travelling%20%26%20node-to-node%20conversations)

  - 2025/03 | Deterministic simulations, ledger rules & P2P Networking (https://pragma.io/projects/amaru#Deterministic%20simulations%2C%20ledger%20rules%26%20P2P%20Networking)
  - 2025/02 | Stake distribution, multi-chain consensus & simulation testing (https://pragma.io/projects/amaru#Stake%20distribution%2C%20multi-chain%20consensus%20%26%20simulation%20testing)
  - 2024/12 | On-disk Ledger State & Observability (https://pragma.io/projects/amaru#On-disk%20Ledger%20State%20%26%20Observability)
  - 2024/10 | First steps: Simple peer-to-peer network with a single block producer (https://pragma.io/projects/amaru/#First%20Steps)
- **Treasury smart contract**: A fully audited (https://github.com/pragma-org/amaru-treasury/blob/265918afa7dfc1e4daeb40b1b1ecfb2f4c995607/audit-report-2025-08-05-txpipe-shop.pdf), deeply tested (https://github.com/pragma-org/amaru-treasury/blob/265918afa7dfc1e4daeb40b1b1ecfb2f4c995607/validators/permissions.test.ak) and open-source treasury smart contract.
- **Public financial journal**: A public journal (https://github.com/pragma-org/amaru-treasury/tree/265918afa7dfc1e4daeb40b1b1ecfb2f4c995607/journal#journal) updated regularly with payouts and delivered milestones from contributors.
- **Website**: Information, documentation and progress on Amaru are now available at (https://amaru.global).

Legal Framework

- **Framework**: a legal framework (https://ipfs.io/ipfs/bafkreiabxyva5lfm6zztg7tnktxvvbbucljrce7hlrp4p6hropqzfaip3y) and contract setup that allows the Amaru Maintainer Committee to use ADA withdrawn from the treasury to pay contributors in fiat currencies.
- **Milestone-based**: every contract has a milestone-based and deliverable-oriented schedule, to map invoices explicitly to the agreed budget and timelines;
- **Self-administered**: payouts are initiated by each scope owner, then reviewed and accepted by the maintainer committee; Everything is assisted by the on-chain treasury smart contract.
- **Transparency**: full transparency of every payout, visible on-chain and recorded in the (public) financial journal (https://github.com/pragma-org/amaru-treasury/tree/265918afa7dfc1e4daeb40b1b1ecfb2f4c995607/journal#journal).

⭐ Node Diversity Workshop(s)

We organised and fully funded the second Node diversity workshop in Toulouse (https://forum.cardano.org/t/cardano-node-diversity-workshop-2-toulouse-22-23-september-2025/150304). With it, we continued building a constructive dynamic to make Cardano more resilient and to create a common, collaborative environment for node implementors.

Conclusion

We are immensely proud of what our team has accomplished and delivered in a short timeframe, and we remain conscious that we had to make adjustments to the initially projected roadmap. We spent more time than anticipated on coordination and establishing a work environment for our project contributors.

With these necessary prerequisites now out of the way and thanks to the massive progress we've made, mainly in the ledger and consensus areas, we're confident in our ability to deliver on the next steps. We firmly believe that Amaru demonstrates not only that an alternative Cardano node implementation is possible, but also that it can deliver significant performance, testability, and maintainability improvements.

There's no more doubt that Amaru contributes to the decentralization and resilience of the Cardano ecosystem.