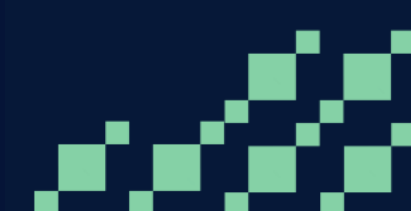
Amaru

アマルプロジェクト:
2025年の回顧

2025年12月30日



目次

コンテキスト

この文書は、2025年第1四半期に設定されたAmaruプロジェクトの初期目標に関する2025年の回顧録です。

2025年の回顧

2025年、私たちはコンセンサス、台帳状態管理、シミュレーションテスト、ガバナンス、およびネットワークアーキテクチャにおいて大きな進展を遂げました。同時に、いくつかの領域（UTxO-RPC、P2Pネットワーキングなど）で当初の計画（https://pragma.io/projects/amaru/#journey）から逸脱しました。

このセクションでは、以前に提示されたロードマップに従い、計画された内容と実行された内容を振り返ります。完了した項目には🟢、進行中または部分的に完了した項目には🟠、中止/キャンセルされた項目には🟪、そしてスケジュールが遅れており注意が必要な項目に🔴を付けます。また、計画外でしたが実行されたいくつかの追加項目には⭐を付けて強調します。

リレーノード（初期目標：2025年第2四半期）

🟢台帳: ディスク上の状態追跡

- 完全なディスク上の台帳状態: メモリフットプリントを最小限に抑えたディスク上のスナップショット。特に以下を含みます。
    1. (e)UTxOs
    2. ステークプールとステーク資格情報の登録/登録解除
    3. アカウント、コンセンサス報酬の計算と分配
    4. DRepsと憲法委員会のメンバー
    5. プロトコルパラメーター
    6. ガバナンスの提案と投票

🟢台帳: （ほぼ）完全な検証

- 部分的なフェーズ1検証: すべてのトランザクション検証ルールはほぼ実装されていますが、いくつかの簡単なチェックはまだ実装する必要があります。Haskellノードとの100%の動作互換性を確保するためには、さらなる強化が必要です。
- 部分的なフェーズ2検証: 新しいPlutus仮想マシンが台帳に統合され、フェーズ2の評価が実装されていますが、まだテストと改善の途中にあります。

🟢台帳: ガバナンスルールの実装

- ガバナンス台帳状態: 投票ステークの分配(s)と、すべてのガバナンスアクションの批准/制定を含むガバナンス。
- 提案フォレスト: 相互依存する提案の最適化されたフォレストとしてのガバナンス提案の追跡。
- ハードフォーク: プロトコルアップグレード（ハードフォーク）と、マルチバージョン台帳および動的プロトコルパラメーターのサポート。

⭐台帳: 適合性テストとブループリント

- 適合性テスト: 参照Haskell実装および/または仕様からのさまざまな適合性テストシナリオの作成。これには、メインネット/プレプロド/プレビューデータからのスナップショットテストが含まれます。
- **Cardano**ブループリント: Cardanoブループリント（https://github.com/cardano-scaling/cardano-blueprint）への貢献により、Cardanoに関する基礎知識を拡張。

🟢コンセンサス: 適合性とシミュレーションテスト

- 純粋なステージのアクターとエフェクトライブラリ: 初期概念実証から、堅牢なイベント処理パイプラインアーキテクチャへの完全な移行。
    1. ディスクI/O、ネットワーク、時間を含む外部エフェクトを分離とテストのために抽象化。
    2. スレッドのインターリーブを正確に制御するための決定論的スケジューラと、本番用の標準的な非決定論的並列スケジューリングの両方をサポート。
- シミュレーションフレームワーク: JepsenおよびIOGのio-simに触発された決定論的シミュレーションテストシステムの開発。現時点では、シミュレーションは主に、テスト対象ノードが複数のアップストリームピアに接続されている場合のチェーン選択をカバーしています。
- データ生成: メッセージ配信のランダム化を含む、任意のコンセンサスシナリオをテストするための合成ブロックツリーの作成。
- 可視化: 実行トレースを可視化し、複雑な分散シナリオのトラブルシューティングを行うためのHTMLツール。
- 再現性: テストの再現性のためのシードと完全なトレースリプレイのサポート。
- **Antithesis**テスト: [Antithesis]テストプラットフォームとの統合。AmaruがAntithesisクラスターでHaskellノードと並行して使用される道を開くために、主にHaskellのcardano-nodeで実験を行い、微妙な問題を特定し報告しています。

🟪UTxO RPCの実装

エコシステム全体で作業の重複を避けるために、Amaruに統合されるAPIの数を制限することを決定したため、この作業項目はロードマップから早期に削除されました。私たちの戦略は、これらの機能を提供するために、Amaruとクライアントアプリケーションの間に位置するミドルウェア（ソフトウェア）に依存することです。UTxO-RPCの場合、これは例えば、TxPipeのDolosやBlink LabsのDingoのようなソフトウェアを意味し、Amaruを信頼できるデータソースとして利用しながら、クライアントレベルのインターフェースを提供できます。

さらに、コア開発、安定性、およびAmaruの内部構造とHaskell実装との適合性に焦点を当てるために、これらの統合の優先順位を下げています。

🔴SPOの関与

継続的インテグレーションパイプラインを通じて、マルチアーキテクチャの**Docker**イメージと静的実行可能ファイルを提供することで、AmaruをSPOによる外部利用に備える上で大きな進歩を遂げました。

ただし、SPOへの働きかけと関与はごく最近開始したばかりです。数人の勇敢な人々が肯定的に反応し、すでにAmaruを彼らの環境にデプロイし、監視しています。これは、ネットワーキングスタックとメムプールの作業を完了するにつれて、継続する必要がある作業の流れです。

🔴ネットワーキング: インバウンドおよびアウトバウンドの検出

- 完全なノード間プロトコル: Cardanoのハンドシェイクおよびノード間プロトコルの実装。
  - 🟢 チェーン同期（イニシエーターとレスポンダー）
  - 🟢 ブロックフェッチ（イニシエーターとレスポンダー）
  - 🟢 キープアライブ（イニシエーターとレスポンダー）
  - 🔴 tx-submission（イニシエーター）。レスポンダー側は開発中です。
  - 🔴 ピア共有はまだ実装されていません。
- 静的ネットワークトポロジ: Amaruは現在、静的ネットワークトポロジ（複数のピアからであっても）のみをサポートしており、ピア選択はありません。これは、さまざまなローカルユースケース（DAppの動作など）には十分ですが、適切なリレーとしての資格を得るには不十分です。

全体として、ネットワーキングスタックは、主に全員が他のタスクで忙しかったために、リソース不足による予期せぬ遅延を経験しました。Amaruを使用可能にする側面に焦点を当て、動的なピアツーピア管理とピア共有を後回しにしました。

また、Haskellチームによって作成されたネットワーキング仕様に従い、バッファ制限と厳格なCBORパースルールを実装することで、ネットワークプロトコルを現在強化しています。

ブロック生成ノード（初期目標：2025年第3四半期）

🟢コンセンサス: チェーン検証

- チェーン選択: ヘッダー保存とディスク上のナビゲーションのための最適化されたデータ構造を使用したOuroborosチェーン選択アルゴリズムの実装。
- 完全なヘッダー検証: Cardanoのステークベースのリーダーシップスケジュールを含む、VRF/KES暗号検証とチェックの完全なサポート。

🔴台帳: 完全な検証

- 残りの台帳ルール: 上述の通り、いくつかの（簡単な）台帳ルールが残っています。私たちは、「未知の未知」が残らないように最も重要なルールはリスクを低減しましたが、時間の都合で残した簡単なチェックにはまだ取り組む必要があります。
- 台帳ストアの改善: Amaruの台帳ストアを、時間の経過とともにいくつかの欠点を示す初期設計に基づいて実装してきました。それらの欠点に対処するために再設計を試みる前に、全体像を把握するのを待っていました。現在、必要な変更の明確な道筋がありますが、まだ実行されていません。

UTxO RPC: ユースケースの実装

前述のとおり、UTxO RPCの作業項目は優先順位が下げられ、最終的に削除されました。UTxO-RPCサポートを含む、さまざまな機能を提供するミドルウェアと統合することで、同様の結果を達成したいと考えています。

公開テストネット検証

- 継続的な**e2e**テスト: テストネット（PreviewとPreprod）とプライベートノードクラスターの両方で、Amaruチェーンを Haskellピアインスタンス間で継続的に同期しています。すべてのヘッダーが検証され、各エポック終了時の状態がHaskellノードの状態と比較されます。
- 実世界での展開: 多くのチームメンバーが（これまでのところ、読み取り専用の）Amaruノードをメインネット、Preview、Preprodに接続して実行しています。一部のSPOも、本番レベルの監視システムに接続されたAmaruを彼らのスタックに統合しています。サミットでのPiの例も、そのような実世界での展開の一例でした。

メムプール: 複雑なメムプールの実装

メムプールの実装が遅れています。主な理由の1つは、Ouroboros Leiosをめぐるさまざまな議論があることです。Leiosに関する作業の大部分は、メムプールとトランザクションの迅速な再検証に関連しています。

（線形）Leiosの設計はまだ進行中の議論であるため、最初からLeios対応のメムプールを提供することに集中できるように、この作業も優先順位を下げています。

ブロック生成

適切なメムプールがなく、ネットワーキングレベルでのピア管理がない場合、ブロック生成はほとんど意味がありません。ネットワーキング側で遅延が発生したため、ブロック生成機能の実装を延期し、他の要素（パフォーマンス、可観測性など）に焦点を当てています。

また、メインネットでの最近の出来事を踏まえ、メインネットの運用リスクが可能な限り低いと確信できる場合にのみブロック生成を許可することが重要だと考えています。より多くのテストと強化が必要な領域を特定しました。また、多くの場合、強化はAmaruとHaskell cardano-nodeの両方に利益をもたらすこと（過去にメインネットに影響を与えかねないバグを発見し報告したことがあります）を覚えておくことも重要です。

ソフトフォークとパフォーマンス改善（初期目標：2025年第4四半期）

🔴ソフトフォーク

ノード多様性ワークショップでこのトピックに関していくつかの議論があり、すべてのノード実装者からの、将来のアップデートで連携し協力したいという真の意欲があります。

しかし、これがどのような形になるかについての明確な計画はまだありません。したがって、現時点では、これは探索段階の進行中の作業のままです。

🟢パフォーマンス

- 継続的なパフォーマンス分析: Amaruの内部パフォーマンス（ヒープ分析、常駐メモリ使用量、CPU使用率、I/O読み取り/書き込みなど）を継続的に監視し、ボトルネックを減らし、さらなる最適化の方法を模索しています。
- ⭐ **Amaru on Pi**: 2025年のCardanoサミットに参加し、制約のあるハードウェアにAmaruをデプロイすることで、低いハードウェア要件が何を意味するかを人々に感じてもらうという課題を自らに課しました。
    - 低メモリ使用量: Pi Zero（利用可能なRAMがわずか約400MB）に至るまで、さまざまなRaspberry PiにAmaruをデプロイしました。
    - 低エネルギー: 小さなバッテリー（約5000 mAh）を使用して何時間も動作させることで、Amaruのエネルギー効率を実証しました。
    - ファーストクラスの**ARM**: SPOがしばしば望むアーキテクチャであるネイティブARMでAmaruを実行できることを示しました。
- 🟢 **UPLC**ベンチマーク: Amaruの仮想マシンのパフォーマンスを検証
    - **Plutus**ベンチマーク: Haskell Plutusチームのベンチマークスイートを実装し、Haskell実装との結果を比較して、より良いか同等のパフォーマンスを確保しました。
    - 「**Turbo**」ベンチマーク: TurboCardanoプロジェクトの一部として作成されたベンチマークスイートを実装。これは、すべてのPlutusバージョンでメインネット上の実際のスマートコントラクトをサンプリングすることで取得されました。

実際、私たちはAmaruの現在のパフォーマンスにすでに満足しています。まだ、よりパフォーマンス集約的なパスを深く探索していませんが、これまでに下した（意識的な）設計上の選択により、すぐに使える十分なパフォーマンスが得られています。

2026年には、データの整合性を維持し、セキュリティ上の考慮事項を犠牲にすることなく、さらに深く掘り下げるための時間を割きたいと考えています。

🟢 より良いオペレーターインターフェース

業界グレードの可観測性スタックと適切に統合された、興味深い一連の有能な監視ツールをすでに実装しています。

- 🟢 構造化トレース: OpenTelemetryを使用した構造化トレース、ロギング、およびメトリックのエクスポートの統合。
- 🟢 メトリック: 既存のツール（例：`gLiveView`）と互換性のある、すべてのステージの詳細なメトリックの収集。
- 🟢 **JSON**ログ: 分析とデバッグのための構造化ログのサポート。
- 🟢 分散スパン: 異なるステージ間での操作のトレース。
- ⭐ **Amaru doctor**: Amaruの内部を検査し、実行を可視化するためのTUI。

🔴 ただし、メトリックとトレースに関する作業は、ライブシステムを実行および監視しているSPOからの数回のフィードバックラウンドを経て初めて真に完了します。SPOとの会話に基づいて、プールパラメーターのオンチェーン更新やガバナンス提案へのオンチェーン投票を作成する方法など、SPO固有のコマンドラインツールをさらに提供する予定です。

ワークフロー

🟢 透明性の高い開発と財務管理

- 完全にオープンソース: Amaruはオープンソースプロジェクトであり、そのソースコードは現在GitHub（https://github.com/pragma-org/amaru-treasury）で公開されており、PRAGMAのDiscord（https://discord.gg/q5XZMZhrUY）で活発なコミュニティが議論しています。また、継続的に日々の活動と探求を記録する開発ログ（https://github.com/pragma-org/amaru/wiki）もあります。
- 公開会議: アライメント、進捗レポート、質疑応答のための隔週の（公開）メンテナー委員会会議。
- 公開デモ: 開発者によるプレゼンテーションとライブデモを伴う隔月の（公開）タッチポイントで、目に見える成果を披露。

    - 2025/11 | 実行のシミュレーションとサミット前の準備（https://pragma.io/projects/amaru/#Deterministic%20simulations%2C%20ledger%20rules%26%20%20P2P%20Networking）
    - 2025/09 | さらなる台帳ルール、分散トレース、適合性テスト（https://pragma.io/projects/amaru#More%20ledger%20rules%2C%20distributed%20traces%20and%20conformance%20testing）
    - 2025/06 | 台帳ルール、タイムトラベル、ノード間通信（https://pragma.io/projects/amaru#Ledger%20rules%2C%20time%20travelling%20%26%20node-to-node%20conversations）
    - 2025/03 | 決定論的シミュレーション、台帳ルール、P2Pネットワーキング（https://pragma.io/projects/amaru#Deterministic%20simulations%2C%20ledger%20rules%26%20P2P%20Networking）
    - 2025/02 | ステーク分配、マルチチェーンコンセンサス、シミュレーションテスト（https://pragma.io/projects/amaru#Stake%20distribution%2C%20multi-chain%20consensus%20%26%20simulation%20testing）
    - 2024/12 | ディスク上の台帳状態と可観測性（https://pragma.io/projects/amaru#On-disk%20Ledger%20State%20%26%20Observability）
    - 2024/10 | 最初のステップ: 単一ブロック生成者によるシンプルなピアツーピアネットワーク（https://pragma.io/projects/amaru/#First%20Steps）
- 財務スマートコントラクト: 完全に監査され（https://github.com/pragma-org/amaru-treasury/blob/265918afa7dfc1e4daeb40b1b1ecfb2f4c995607/audit-report-2025-08-05-txpipe-shop.pdf）、深くテストされ（https://github.com/pragma-org/amaru-treasury/blob/265918afa7dfc1e4daeb40b1b1ecfb2f4c995607/validators/permissions.test.ak）、オープンソースの財務スマートコントラクト。
- 公開財務日誌: 貢献者からの支払いと達成されたマイルストーンが定期的に更新される公開日誌（https://github.com/pragma-org/amaru-treasury/tree/265918afa7dfc1e4daeb40b1b1ecfb2f4c995607/journal#journal）。
- ウェブサイト: Amaruに関する情報、ドキュメント、進捗状況は、現在（https://amaru.global）で入手可能です。

法的枠組み

- 枠組み: Amaruメンテナー委員会が、財務から引き出されたADAを使用して、貢献者に法定通貨で支払うことを可能にする法的枠組み（https://ipfs.io/ipfs/bafkreiabxyva5lfm6zztg7tnktxvvbbucljrce7hlrp4p6hropqzfaip3y）と契約設定。
- マイルストーンベース: すべての契約にはマイルストーンベースの成果指向のスケジュールがあり、請求書が合意された予算とタイムラインに明確に対応するようにします。
- 自己管理: 支払いは各スコープオーナーによって開始され、メンテナー委員会によってレビューおよび承認されます。すべてはオンチェーンの財務スマートコントラクトによって支援されています。
- 透明性: すべての支払いの完全な透明性があり、オンチェーンで確認でき、（公開）財務日誌（https://github.com/pragma-org/amaru-treasury/tree/265918afa7dfc1e4daeb40b1b1ecfb2f4c995607/journal#journal）に記録されています。

⭐ノード多様性ワークショップ

私たちは、第2回ノード多様性ワークショップをトゥールーズで開催し、全額出資しました（https://forum.cardano.org/t/cardano-node-diversity-workshop-2-toulouse-22-23-september-2025/150304）。これにより、Cardanoをより強靭にするための建設的なダイナミクスを構築し続け、ノード実装者のための共通の協力的な環境を創出しました。

結論

短期間で私たちのチームが達成し、提供したことを非常に誇りに思っており、当初予定されていたロードマップに調整を加える必要があったことも認識しています。プロジェクト貢献者のための調整と作業環境の確立に、予想以上の時間を費やしました。

これらの必要な前提条件が整い、主に台帳とコンセンサスの分野で達成した大きな進歩のおかげで、私たちは次のステップを実現する能力に自信を持っています。Amaruは、代替のCardanoノード実装が可能であるだけでなく、パフォーマンス、テスト容易性、および保守性の面で大きな改善をもたらすことができることを示していると強く信じています。

AmaruがCardanoエコシステムの分散化と回復力に貢献していることに、もはや疑いの余地はありません。