The Aptos Blockchain: Safe, Scalable, and Upgradeable Web3 Infrastructure

August 11, 2022

v1.0

概要

ブロックチェーンは新しいインターネットのインフラとして急速に台頭しています。開発者達は数十万もの分散型アプリケーションを展開し、その速度は急速に成長しています。しかし、トランザクションの頻繁な遅延や中断、高いトランザクション費用、低いスループットの制限、そしてセキュリティ上の懸念などから、ブロックチェーンの一般的な普及はまだ進んでいません。Web3時代を一般に浸透させるためには、ブロックチェーンインフラは信頼性のある、スケーラブルで、コスト効率の良い、絶えず進化し続ける「プラットフォーム」としての役割を果たす必要があります。

このホワイトペーパーでは、これらの課題を克服するために、私たちが「スケーラビリティ」、「安全性」、「信頼性」、「アップグレードの可能性」という主要な四大原則を採用した『Aptosブロックチェーン』をご紹介します。Aptosブロックチェーンは、過去3年間で全世界の350人以上の開発者によって開発されました[1]。これには、コンセンサス、スマートコントラクトの設計、システムセキュリティ、性能、そして分散化に関する画期的なイノベーションが含まれています。

• 1. Aptosブロックチェーンは、「高速」で「安全」なトランザクションの実行のために、「Move言語」をネイティブ言語として統合し、内部で使用しています[2]。"Move prover" は、Move言語で書かれたスマートコントラクトの正式な「検証者」であり、コントラクト(= 契約) の不変性と動作に対して追加の保護を提供します。このセキュリティに重点を置くことで、各開発者が自身のソフトウェアを悪意ある第三者から保護することが可能となります。

• 2.Aptosのデータモデルは、柔軟な「キー管理」とハイブリッドな「保管オプション」を可能にします。これらに加えて、署名前のトランザクションを実行する際の「透明性」と「実用的」なライトクライアントプロトコルが、より安全かつ信頼性のあるユーザーエクスペリエンスを提供します。

• 3. Aptosは、トランザクションを処理する際の主要なステージのためのパイプライン化されたモジュラーなアプローチを採用します。これにより、高いスループットと低遅延を達成します。具体的には、「トランザクションの拡散」、「ブロックメタデータの順序付け」、「並行トランザクションの実行」、「バッチストレージ」、および「台帳認証」の各ステージが同時に動作します。

• 4. Aptosは、開発者がデータの読み書きの前知識を必要とするトランザクションの原子性を破る他の並行実行エンジンとは異なり、このような制限を設けません。

• 5. Aptosのモジュラー型アーキテクチャ設計は、クライアントの柔軟性をサポートし、頻繁かつ即時のアップグレードが可能です。さらに、新しい技術革新を迅速に導入し、新しいWeb3のユースケースをサポートするため、Aptosブロックチェーンは埋め込み型のオンチェーン変更管理プロトコルを提供します。

• 6. Aptosブロックチェーンは、個々のバリデーターの性能を超えてスケールアップするための将来の取り組みを試験的に実施しています。そのモジュラー設計と並列実行エンジンは、バリデーターの内部シャーディングをサポートしており、均一なステートシャーディングは、ノードオペレー

ターの追加の複雑さを増やすことなく、水平スループットのスケーラビリティの可能性を提供します。

【免責条項】
このホワイトペーパー及びその内容はいかなるトークンの売買の申込み又は勧誘をするものではありません。我々はこのホワイトペーパーを皆様からのフィードバックやご意見を受けることのみを目的として公表しています。本書面のいかなる内容も、Aptosブロックチェーン又はそのトークン（それがあれば）がどのように発展するのか、利用されるのか、又は価値を有するようになるのかについて、保証又は約束するものと判断又は解釈されるべきものではありません。Aptosは現状の計画の概要を説明しているに過ぎず、当該計画はAptosの裁量で変化しうるものであり、それが成功するかは、Aptosのコントロール外の様々な要因に依ります。そのような将来についての言及は必然的に既知又は未知のリスクを含むものであり、そのようなリスクは、将来、実際の成果や結果を、我々が本ホワイトペーパーにおいて記載し又は意味するものとは大きく異なるものにする可能性があります。Aptosはその計画のアップデートをする義務を負いません。実際の結果や将来の事象は大きく変わりうるため、本ホワイトペーパーのいかなる記載も、将来、正確であると判明する保証はありません。将来についての言及に過度の信頼を置かないようにお願いいたします。

1 イントロダクション

Web2のインターネットでは、多くのサービス（メッセージング、ソーシャルメディア、金融、ゲーム、ショッピング、オーディオ/ビデオストリーミングなど）が、中央集権的な企業（例: Google、Amazon、Apple、Meta）によって制御されています。これらの企業はアプリケーション固有のソフトウェアとクラウドインフラを使用して、サービスを数十億のユーザーに展開しています。しかし、Web2のモデルは、ユーザーが中央集権的なエンティティを信頼する必要がある点で、社会的な懸念が増加しています。
この懸念事項を解決するために、新しいインターネットのパラダイムとしてWeb3が出現しました。Web3では、「ブロックチェーン」が中心となり、中央の仲介者を必要とせずに、ユーザー間の安全で信頼性のある相互作用を可能にします。ブロックチェーンは、分散型アプリケーションの基盤として、数十億のユーザーへの展開をサポートすることが期待されています。
しかし、多くのブロックチェーン技術はまだ成熟しておらず、信頼性、スループット、取引手数料、セキュリティなどの課題が残されており、クラウドインフラがweb2サービスを数十億人に到達させる方法と比較して、ブロックチェーンはまだWeb3アプリケーションを同様にすることができていません[3]。

2 Aptosのビジョン

Aptosの目的は、Web3の普及を促進し、分散型アプリケーションのエコシステムを強化することです。私たちは、信頼性、安全性、そして性能の面で最先端のブロックチェーンアーキテクチャを提供することを目指しています。このアーキテクチャは、技術の進化に迅速に対応し、新しいユースケースをサポートするために、柔軟かつモジュラーであるべきです。

私たちのビジョンは、コミュニティによって統治され、運営する分散型、安全でスケーラブルなネットワークを築くことです。インフラとしての需要が増えると、ブロックチェーンの計算リソースは水平および垂直にスケーリングアップし、ニーズを満たしていく必要があります。新しいユースケースや技術的進歩に適応するため、ネットワークは頻繁に、ユーザーの中断無しにシームレスにアップグレードされる必要があります。インフラに関する懸念は、背景に徐々に薄れていくと考

えます。開発者とユーザーは、キーの回復やデータモデリング、スマートコントラクトの標準、リソース使用のトレードオフ、プライバシー、そして組み合わせのためのオプションに簡単にアクセスできるようになります。ユーザーは、資産が安全で常時アクセス可能であることを確信し、信頼性を持って、世界中の取引を行うことができます。結果として、ブロックチェーンはクラウドインフラと同じくらい普及することが期待されます。

このビジョンを実現するためには、顕著な技術的な進歩が求められます。過去3年にわたりDiemブロックチェーン（Aptosブロックチェーンの前身）の構築、開発、拡張、そして展開を行ってきた経験は、ネットワークがそのプロトコルを継続的にアップグレードしても、クライアントを混乱させることなく進められることを示しています[4]。

Diemメインネットは2020年初めに12以上のノードオペレーターと複数のウォレットプロバイダーにデプロイされました。その翌年には、私たちのチームは、コンセンサスプロトコルとコアフレームワークを変更する2つの主要なアップグレードを行いました。両アップグレードは、ユーザーのダウンタイム無しに完了させることが出来ました。Aptosブロックチェーンでは、Diemブロックチェーンに触発された安全で透明で頻繁なアップグレードをコア機能として組み込むだけでなく、技術スタックに一連の急進的な改良を加えました。特に、トランザクション処理の新しい方法（セクション7で説明）および分散化とネットワークガバナンスへの新しいアプローチを強調します。

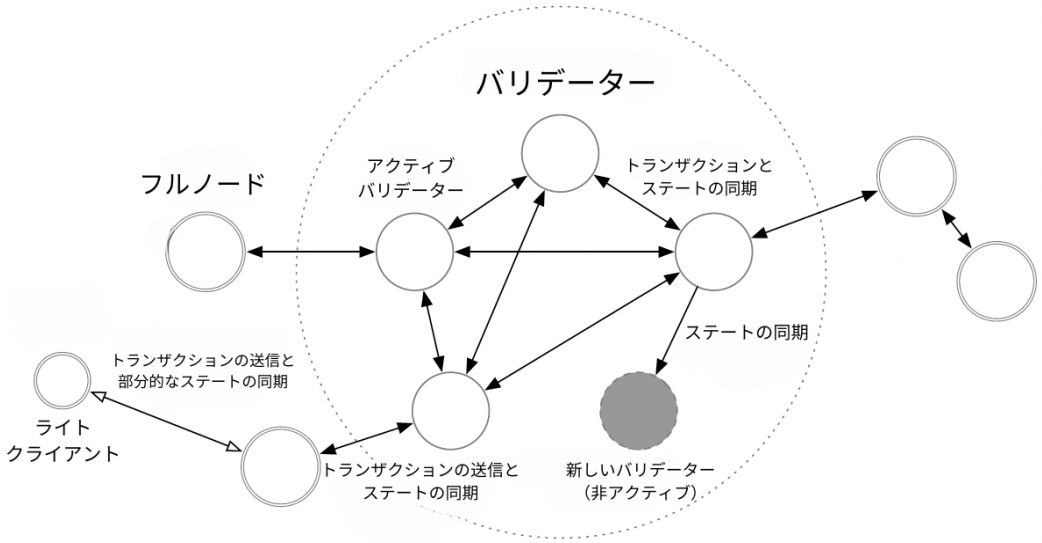

図1：Aptosエコシステムの構成要素

今後、これからもAptosブロックチェーンがさらなる進化を遂げていくにあたり、私たちはこのホワイトペーパーを定期的に更新し、プロトコルとデザインの最新の進展を共有していきます。このドキュメントの以降の部分では、Aptosブロックチェーンの現状と将来の展望について詳細に触れます。

3 概要

Aptos ブロックチェーンは、図1に示すように、バイザンチンフォールトトレランス（*BFT*）証明のステーク合意メカニズムを活用しており、一連の検証者がユーザーからのトランザクションを共同で受信・処理します。トークン所有者は特定の検証者にトークンをステークし、その検証者の投票の重みはステークされた金額に比例します。アクティブな検証者は合意に参加することができますが、十分なステーク量を保有していない / 過去のパフォーマンスが不十分であると合意プロトコ

ルによって判断された場合などの条件に該当する場合は、非アクティブ化される場合もあります。

クライアントは、トランザクションの送信やブロックチェーンの状態・履歴の照会などに対してシステムの一部として関与し、データ検証者による署名された証明をダウンロードして検証することを選択することができます。「フルノード」はトランザクションやブロックチェーンの状態を複製し、必要に応じてデータのプルーニングを行うことができるクライアントです。一方、「ライトクライアント」は現行の検証者のセットを持ち、部分的なブロックチェーン情報の照会をフルノードから行います。このライトクライアントの代表的な例としてウォレットがあります。

Aptos ブロックチェーンは、既存の課題を解決するためのWeb3インフラの要求 (安全 / 迅速 / 信頼性 / アップグレード可能)を満たすための以下の原則に基づいて構築されています:

• 新しいスマートコントラクトプログラミング言語、Move [5] を通じた高速かつ安全な実行、シンプルな監査可能性、および機械的な解析性。

• 極めて効率の良いデータ処理能力と低遅延を実現するためのバッチ処理、パイプライン処理、並列処理のアプローチ。

• Block-STMを用いた新しい並列トランザクション処理：データの読み書きの先知的知識を必要とする既存の並列実行エンジンとは異なり、任意に複雑なトランザクションで原子性を効率的にサポートする新しい並列トランザクション処理。

• 急速なステークウェイト検証者セットのローテーションと評判追跡を通じてのパフォーマンスと分散化のための最適化。

• 新しいユースケースと最新の技術を受け入れるための第一級の設計原則としてのアップグレード可能性と設定可能性。

• 厳格なコンポーネントレベルのテスト、適切な脅威モデリング、モジュラーデザインを通じた安全で信頼性のある運用

• シャーディング (ユーザーやプログラミングおよびデータモデルに対して第一級の概念として露出) を活用した分散化を維持しながらの水平スループットのスケーラビリティ。

以下は、各セクションの詳細です。

- セクション4 - Move言語
  開発者がAptosブロックチェーンでMoveを用いて、どのように対話するかを説明します。
- セクション5 - 論理的データモデル
  論理データモデルについて説明します。
- セクション6 - 安全なユーザーエクスペリエンス
  Aptosブロックチェーンが強力な検証方法を通じて安全なユーザーエクスペリエンスを可能にする方法を詳しく説明します。
- セクション7 - パイプライン化、バッチ処理、並列処理
  パイプライン化、バッチ処理、並列処理に関する主要なパフォーマンス革新について説明します。

- セクション8 - 状態同期
  異なるクライアントが他のノードとの状態を同期するためのさまざまなオプションについて詳しく説明します。
- セクション9 - コミュニティ所有
  コミュニティの所有権とガバナンスに関する私たちの計画について説明します。
- セクション10 - 性能
  分散化を維持しながらの将来のパフォーマンスの方向性について議論します。

4 Move言語

Moveは、Aptos ブロックチェーンにおいて重要な役割を果たすスマートコントラクトプログラミング言語で、その設計は「安全性」と「柔軟性」を重視しています。この言語は、台帳の状態を表すためのオブジェクトモデルと、状態遷移のルールをエンコードするためのMoveコード（モジュール）の両方でAptos ブロックチェーンに利用されています（セクション5.5 参照）。ユーザーは、Moveを通じてトランザクションを提出でき、新しいモジュールの公開や既存のモジュールのアップグレード、さらにはモジュールのエントリ関数の実行や公開インターフェースとの対話が可能です。

Moveエコシステムは、コンパイラ、仮想マシン、そして多くの開発者ツールを含んでおり、Rustプログラミング言語からインスピレーションを受けて設計されました。Moveの特徴は、リソースの希少性、保存、およびアクセス制御を中心に、リニアタイプなどの概念を通じてデータの所有権を明示的に表現することです。この言語では、Coinのようなリソースが適切な資格無しで生成、二重使用、または失われることがないように、各リソースの寿命やストレージ、アクセスパターンが定義されています。

安全性の面では、Moveはバイトコード検証者を通じて、不信頼なコードの中での型とメモリの安全性を保証します。更に、Move Prover[6]という正式な検証ツールが組み込まれており、これを使うことでMoveプログラムの仕様に対する正確性を検証することができます。この検証方法は、Moveに統合された仕様言語で定式化されています。

台帳の状態に関連して、Aptos ブロックチェーンはユーザーアカウントだけでなく、ネットワークの設定情報も含めて保持します。これには、アクティブな検証者のセットやステーキングの属性、Aptos ブロックチェーンのサービス設定などが含まれています。Moveの柔軟性により、Aptos ブロックチェーン自体の設定変更やアップグレードが容易に行え、実際にこれらのアップグレードはプライベートメインネット上で複数回、ダウンタイム無しで行われています。

AptosチームはMoveの機能を拡張し、より多くのWeb3ユースケースをサポートする取り組みを進めています。セクション5.5では、Aptos ブロックチェーンが微細なリソース制御をどのように実現しているかが説明されており、これによりデータへのアクセスや変更のコストが固定化され、実行の並列化がサポートされます。さらに、Aptosは「テーブルサポート」を提供しており、これにより大規模なデータセットもアカウント内での保持が可能になります。Aptosの持つ機能の中でも特筆すべきは、オンチェーンで表される共有や自律的なアカウントのサポートです。これにより、分散型自律組織（= DAO）のような組織がリソースを共有し、協力して利用することができるようになります。

5 論理的データモデル

Aptosブロックチェーンの台帳は、全てのアカウントの状態を反映しています。この台帳は、システムで実行されたトランザクションの総数に基づいて、符号無しの64ビット整数でバージョン管理されています。誰でもAptosブロックチェーンにトランザクションを提出し、台帳の状態に変更を加えることができます。トランザクションが実行されると、その結果としてトランザクションの出力が生成されます。この出力には、台帳の状態に対する操作（*write set*）、結果イベントのベクトル（セ

クション5.1.1参照）、使用されたガス量、そしてトランザクションの実行ステータスが含まれています。

5.1 トランザクション

署名付きのトランザクションには以下の情報が含まれます。

• トランザクション認証子: 送信者は1つまたは複数のデジタル署名を含むトランザクション認証子を使用して、トランザクションが認証されていることを確認します。

• 送信者のアドレス: 送信者のアカウントアドレスです。

• ペイロード: ペイロードは、オンチェーンの既存のエントリ関数を参照するか、またはインラインのバイトコードとして実行される関数（スクリプト）を含みます。さらに、入力引数のセットがバイト配列でエンコードされます。peer-to-peerのトランザクションの場合、入力には受信者の情報と彼らへの送金額が含まれます。

• ガス価格**(**指定された通貨*/*ガス単位**)**: トランザクションの実行に必要なガスの単位あたりの料金です。これは、コンピューティング、ネットワーク、およびストレージの費用をカバーするためのものです。

• 最大ガス量: トランザクションが消費できるガスの最大量。これを超えるとトランザクションは中止されます。アカウントは最低でもガス価格に最大ガス量を掛けたものを持っていなければならず、そうでない場合は検証中にトランザクションは破棄されます。

• シーケンス番号**:** トランザクションの順序を追跡するための番号。これにより、トランザクションの一貫性とリプレイ攻撃の防止が実現されます。

• 有効期限: トランザクションが受け入れられる最終的な時間を示します。この時間を過ぎると、トランザクションは処理されません。

• チェーン**ID**: トランザクションが適用されるブロックチェーンを指定します。これは、トランザクションの正確さを確保するためのものです。

各バージョン*i*において、状態変更はタプル(*Ti,Oi, Si*)で表され、それぞれトランザクション、トランザクション出力、結果としての台帳の状態を含んでいます。決定的な関数Applyを使用して、台帳の状態*Si−1*でのトランザクション*Ti*の実行は、トランザクション出力*Oi*と新しい台帳の状態*Si*を生成します。つまり、Apply(*Si−1, Ti*) → ⟨*Oi, Si*⟩です。

5.1.1 イベント

イベントは、トランザクションの実行中に発行されるもので、このメカニズムはAptosの中核的部分です。特定のMoveモジュールごとに、独自のイベントが定義され、そのモジュール内での操作に応じて適切なタイミングでイベントが発行されます。たとえば、コインの転送時、送信者のアカウントは*SentEvent*を、受信者のアカウントは*ReceivedEvent*を発行します。これらのイベント情報は台帳に保存され、Aptosノードを介して後からクエリすることが可能です。

さらに、各イベントには一意のキーが付与されており、このキーを使用して特定のイベントの詳細情報を取得することができます。同じイベントキーで発行されたイベント群は、イベントストリー

ムとしてまとまり、それぞれのエントリには連番、タイプ、そして関連データが記録されます。イベントのタイプは、それぞれのイベントを特定・区別するために重要です。同じ、または似たようなタイプを持つ複数のイベントが存在することもあります。特に、ジェネリクスの概念を使用した際にこのような状況が生まれることがあります。イベントに関連するデータは、トランザクションの実行に伴う変更を理解するための基盤となります。Moveモジュールの開発者は、トランザクションが行ったデータ変更や、それに伴って発行されるイベントに、変更内容を正確に理解するための十分な情報を含めるべきです。

最後に、トランザクションはイベントを発行することはできますが、一度発行されたイベントを直接読み取ることはできません。これはAptosの設計上の決定であり、この設計によって、トランザクションの実行は現在の状態とトランザクションの入力のみに依存し、歴史的な情報に依存することがなくなります。

5.2 アカウント

各アカウントは、一意の256ビットの値であるアカウントアドレスによって識別されます。新しいアカウントの作成は、既存のアカウントから発行されたトランザクションがcreate_account(addr) Move関数を呼び出す際に、台帳の状態（セクション5.5参照）に記録されます。これは、まだ作成されていないアカウントアドレスにAptosトークンを送信する意図があるときに発生します。しかし、便宜上、Aptosはtransfer(from, to, amount)関数を通じて、既存しないアカウントへの送信を試みる際に新しいアカウントを作成する機能も提供しています。

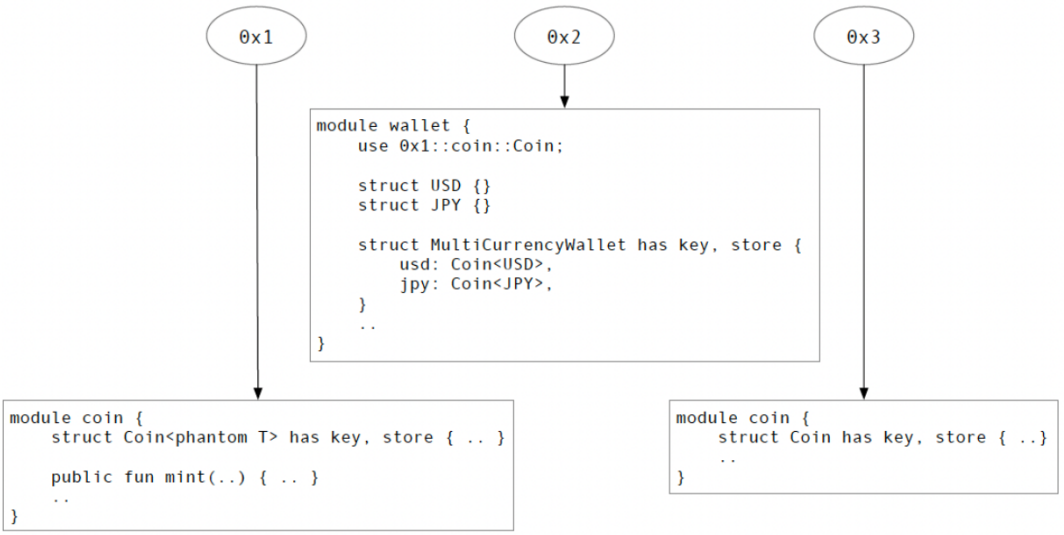

図2：オンチェーン Move モジュールの例

アカウントの作成を始めるために、ユーザーは最初に署名キーペア(*vk, sk*)を生成する必要があります。公開検証キー*vk*と署名スキーム識別子(*ssid*)の組み合わせから、指定された署名スキームに基づく新しいアカウントアドレスが、暗号ハッシュ*H*を用いて導出されます。具体的には、*addr = H(vk, ssid)*という関係になります。

この新しいアカウントアドレスaddrが作成された後、ユーザーはこのアドレスからのトランザクションに署名するために、プライベート署名キー*sk*を使用することができます。必要に応じて、セキュリティ上の理由や他の目的で、ユーザーは*sk*を変更やローテート (= *rotate*)させることが可

能ですが、それによってアカウントアドレスが変わることはありません。アカウントアドレスは、初めて生成された公開検証キーを基に、一度だけ作成されます。
Aptos ブロックチェーンの特徴として、アカウントは特定の実際のアイデンティティに直接リンクされることはありません。ユーザーは複数のキーペアを生成し、それによって複数のアカウントを作成することが出来ます。同じユーザーが管理する複数のアカウント間には固有のリンクは存在しないものの、一つのウォレット内で複数のアカウントを統一的に管理することは可能です。この設計は、ユーザーの匿名性を確保しつつ、プライバシー保護のための新しい手法を試みる実験的提供です。また、単一のユーザーやユーザーグループが持つ複数のアカウントは、トランザクションの同時実行を増加させるためのチャネルも提供します。この点については、セクション7.4で詳しく説明されています。

5.3 Moveモジュール

Moveモジュールは、データタイプ（structs）と手続きを宣言するMoveバイトコードを内包しています。モジュールの識別には、モジュールが宣言されたアカウントのアドレスとモジュール名が使用されます。例として、図2に示される最初の通貨モジュールの識別子は0x1::coinとなっています。

モジュールは他のオンチェーンモジュールに依存することができます。この機能は、図2にあるウォレットモジュールで確認することができ、これによりコードの再利用が促進されます。重要な点として、モジュールはアカウント内で一意である必要があり、1つのアカウントは特定の名前で1つのモジュールのみを宣言できます。例を挙げると、図2にあるアドレス0x1のアカウントでは「coin」という名前のモジュールを再度宣言することはできません。しかし、アドレス0x3のアカウントは「coin」という名前で新しいモジュールを宣言可能で、その際の識別子は0x3::coinとなります。また、0x1::coin::Coinと0x3::coin::Coinは全く異なるタイプです。これらは交換や共通のモジュールコードの共有はできません。しかし、0x1::coin::Coin<0x2::wallet::USD>と0x1::coin::Coin<0x2::wallet::JPY>は同じジェネリックタイプを持つ異なるインスタンスであり、これらは交換できないものの、同じモジュールコードを共有することができます。
これらのモジュールは、同じアドレス内のパッケージでまとめられます。該当アドレスの所有者は、バイトコードとパッケージのメタデータを合わせて、オンチェーンに公開します。このメタデータは、パッケージがアップグレード可能か、もしくは不変かを示します。アップグレード可能なパッケージでは、実際にアップグレードが行われる前に、互換性の検証が行われます。この検証では、既存のエントリポイント関数が維持され、リソースはメモリに保存されませんが、新しい関数やリソースの追加は許可されます。
最後に、Aptosフレームワークは、Aptosブロックチェーンのコアライブラリや設定として、標準のアップグレード可能なモジュールのパッケージとして定義されている点を指摘しておきます（詳細はセクション9.2を参照）。

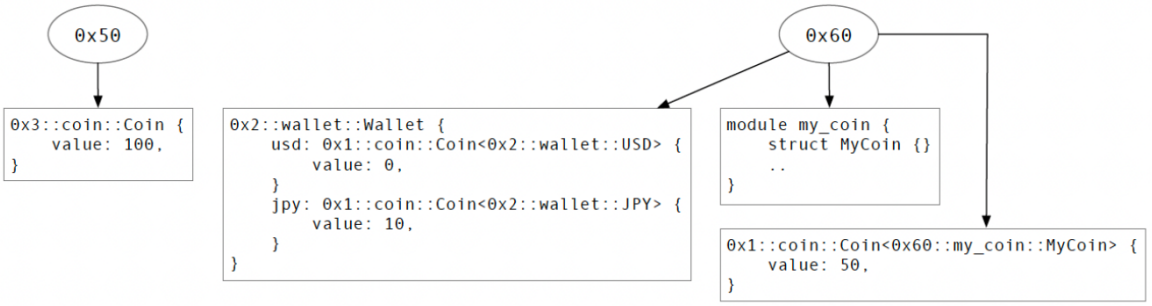

図3 : オンチェーンデータの例

5.4 リソース

モジュールと並行して、アカウントアドレスは関連するデータ値も持つことが可能です。具体的には、各アカウントアドレス内の値は、そのタイプに基づいてキー化されており、特定のタイプのデータ値がそれぞれのアカウントに関連付けられています。図3にこの仕組みの例が示されています。例として、アドレス0x50が保持する単一のデータ値は、0x3::coin::Coinという完全修飾型 (= *fully-qualified type*)で表現されています。ここで、0x3はcoinモジュールが格納されているアドレス、coinはモジュール名、そしてCoinはデータタイプの名前を指します。さらに、ジェネリックタイプのデータ値も許容され、それぞれのインスタンスは独自のタイプとして識別されます。この仕組みはシステムの拡張性を強化し、異なるジェネリックインスタンスが同じ機能コードを共有することを許容します。

データの操作に関して、ミューテート(変更)、削除、公開などのルールは、該当のデータタイプを定義するモジュールによってエンコードされています。Moveのセキュリティメカニズムは、他のモジュールで定義されているデータタイプのインスタンスを、不正に作成、変更、削除するような他のコードやエンティティのアクセスを防ぎます。

アドレスごとに各データタイプの最上位の値を1つしか持てないという制限があるため、初めての方には制約が強いと感じるかもしれませんが、この制限は問題とはなりません。なぜなら、プログラマは内部フィールドとして他のデータを保持する *wrapper type* を定義・作成することで、その制限を回避することができるからです。図3に示されているWallet structは、このwrapper typesの実例を示しています。

最後に、全てのデータタイプがオンチェーンに保存されるわけではありません。最上位のデータとして保存する資格があるデータタイプは、「キーの能力 (= *key ability*) 」を必要とします。*nested values* の場合は、「ストア能力 (= *store ability*) 」が必要です。これらの両方の能力を持つデータタイプは、「リソース (= *resources*)」としても認識されます

5.5 レジャー状態

各アカウントをMove仮想マシン（Move VM）の視点から見ると、キー・バリューデータ構造と連なる値から成り立っています。これらのデータ構造は、「*table entries*」と呼ばれ、バイナリカノニカルシリアル化フォーマット（BCS）によって保存されています。この特定のデータの配置は、多数のアカウントに小規模なデータをまたがる形で、あるいは少数のアカウントに大量のデータを持つ形で、スマートコントラクトを効果的に操作する際に役立ちます。Moveモジュールはアカウントデータと同様に保存されますが、それは独立した *namespace* の下に格納されています。初期のブロックチェーンの状態、すなわちgenesisレジャーの状態は、ブロックチェーンが始動する際にアカウントとその初期の状態を定義します。

Aptosブロックチェーンの起動時は、この単一の状態でレジャーが表現されます。しかし、Aptosは技術の発展と採用率の増加を背景に、将来はシャードの数を増やすことでトランザクションの処理能力を拡張する計画をしています。このシャード増加により、複数のレジャー状態が生じ、異なるシャード間での資産の移動やアクセスが可能になるでしょう。

各レジャー状態は、特定のシャードにおける全オンチェーン資産の維持に責任を持ち、キー・バリューデータストアの形を通じて、一貫したアカウントモデルとストレージへのアクセスコストを提供します。

6 セーフなユーザーエクスペリエンス

数十億のインターネットユーザーを取り込むためには、Web3のユーザーエクスペリエンスは安全性を確保しつつ、簡便にアクセス可能でなければなりません。こちらのセクションでは、この目的を達成するためのAptosブロックチェーンの革新的な機能について解説します。

6.1 トランザクションの生存性保護

トランザクションに署名をすることは、署名者がトランザクションを通してブロックチェーンに対して「コミット」および「実行」されることを許可することを意味します。時には、ユーザーがトランザクションに意図せず、または考慮せずに署名するケースがあります。このリスクを減少させるため、Aptosブロックチェーンは全てのトランザクションの生存性を制約し、署名者を無制限の有効性から保護します。現在、Aptosブロックチェーンによって提供される保護は3つあります。

- 送信者のシーケンス番号: これは、各送信者のアカウントごとに1回だけコミットが可能な番号です。これにより、送信者は現在のアカウントのシーケンス番号を確認することで、該当のトランザクションtが既にコミットされているか、またはコミットされる可能性がないことを判別できます（該当のシーケンス番号が他のトランザクションによって消費されている場合）。
- トランザクションの有効期限: ブロックチェーンの時刻は非常に高い精度で更新されます。そして、あるトランザクションtの有効期限がブロックチェーンの時刻を超えた場合、そのトランザクションtは既にコミットされているか、コミットされることはないと判断されます。（セクション7.3.1で詳細）
- 指定されたチェーン識別子: 各トランザクションには、他のブロックチェーン環境間でのリプレイを防ぐための固有の識別子が付与されています。これにより、例えばテストネットとメインネットを跨いだ悪意のある再生が防がれます。

6.2 Moveベースのキー管理

Aptosアカウントは、セクション5.2で触れられたキーのローテーション機能を持っており、これはプライベートキーの妥協や長距離攻撃、さらには暗号化アルゴリズムが将来的に破壊されるリスクを軽減する上で極めて重要です。さらに、Aptos アカウントは、新しいハイブリッドな管理モデルの実装にも対応しています。このモデルでは、ユーザーはプライベートキーの管理権をカストディアンや他の信頼できるエンティティに委任することができます。Moveモジュールは、これらのエンティティに対し、特定の状況でキーをローテートさせるポリシーを定義する能力を提供します。例として、これらのエンティティはk-out-of-nマルチシグキーの形で表現される複数の信頼パーティーによって、キーの喪失時のリカバリーサービスを提供する可能性があります。(ビットコインの20%がアクセス不能なアカウントに保管されている事例がこれを示しています。[7])

現在の多くのウォレットは様々なキーリカバリースキームをサポートしており、これらの多くはブロックチェーンのサポート無しに実装されています (プライベートキーをクラウドインフラにバックアップする、マルチパーティ計算、ソーシャルリカバリなど) 。そのため、各ウォレットは専用のキー管理インフラを構築する必要があり、このプロセスはユーザーにとって不透明になりがちです。しかし、Aptosブロックチェーンでは、キー管理機能が直接ブロックチェーンレイヤーに組み込まれているため、キーに関する全ての操作が透明になり、高度なキー管理機能を持つウォレットの実装も容易になります。

6.3 トランザクションの事前署名の透明性

現在のウォレットは、署名対象となるトランザクションに関して十分な透明性を持たないため、ユーザーは資金を盗まれるリスクが高まっています。この問題は、各トランザクションがオンチェーンデータにアクセスする際にも顕著となり、様々な攻撃からユーザーを守るための現行の対策では不十分です。この課題に対応するため、Aptosエコシステムはトランザクションの「事前実行サービス」を導入しています。このサービスは、署名の前にユーザーにトランザクションの結果を人間が読める形式で提示するものです。この機能は、過去の攻撃や悪意のあるスマートコントラクトの履歴情報と組み合わせることで、ユーザーを詐欺から守るのに役立ちます。

さらに、Aptosでは、ウォレットがトランザクションの実行に特定の制約を設けることが可能です。この制約に反する動作をした場合、トランザクションは自動的に中断されるため、ユーザーは悪意のあるアプリケーションやソーシャルエンジニアリング攻撃から追加の保護を受けることができます。

6.4 実用的な軽量クライアントプロトコル

APIプロバイダのTLS/SSL証明書だけに依存してブロックチェーンクライアントとサーバ間の信頼を確立することは、クライアントの安全性を十分に確保するものではありません。有効な証明書があっても、ウォレットやクライアントは提供されるデータの真正性や完全性を保証できません。

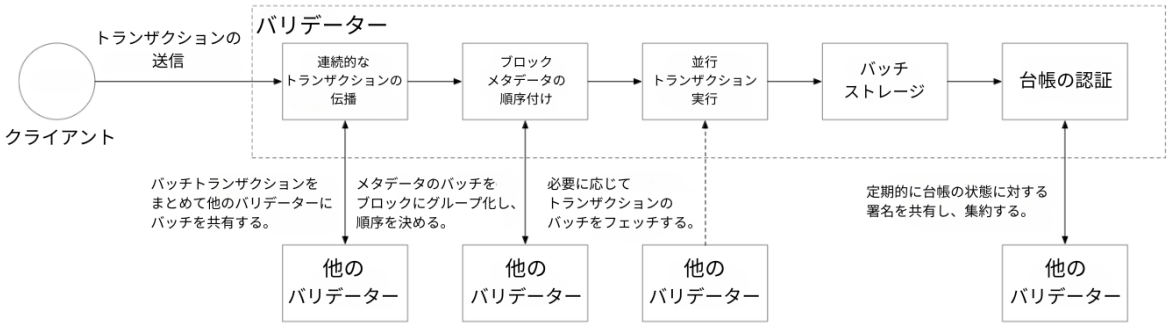

図4：トランザクション処理のライフサイクル。
全てのステージは完全に独立しており、個別に並列処理することが可能です。

APIプロバイダは誤ったデータや悪意のあるブロックチェーン情報を返し、第三者を欺くことや二重支払い攻撃を実行する可能性があります。

これを解決するために、Aptosは「ステート証明」と「軽量クライアント検証プロトコル」を導入しています。これにより、ウォレットやクライアントは信頼されていない第三者のサーバーからのデータの妥当性を確認できます。セクション7.6.2のタイムスタンプベースのステート証明を用いることで、軽量クライアントは常に最新のアカウント状態（例えば、数秒以内）を確認できます。そして、ネットワーク構成の変更（エポックの変更）を追跡するか、現在の信頼されるチェックポイント（ウェイポイント）を利用して最新の状態を維持できます[8]。

Aptosブロックチェーンは、高頻度のタイムスタンプとミニマルでのステート証明を組み合わせることで、クライアントに対して強固なセキュリティを提供します。さらに、Aptosのノードは証明へのサブスクリプションへの特化を可能とした高性能なストレージインターフェースを提供します。このインターフェースは、フルノードの運用や大量のトランザクション処理無しに、最小の検証可能なデータを維持する軽量クライアントにとって有用です。

7 パイプライン、バッチ処理、並列トランザクション処理

Aptosブロックチェーンのトランザクション処理は、スループットの最大化、並行性の増加、およびエンジニアリングの複雑さの縮小を目的として、複数の独立した段階に分割されています。これらの段階は互いに独立しており、個別に並列実行が可能です。この設計は、現代の「スーパースカラープロセッサアーキテクチャ (= superscalar processor architectures) 」と類似しています。これにより、パフォーマンスの大幅な向上が期待されるだけでなく、Aptosブロックチェーンは新しいモードのバリデータクライアントとの相互作用をサポートします。

具体的に、以下のような相互作用が考えられます：

• クライアントは、特定のトランザクションが永続的なトランザクションバッチに含まれた際に通知を受け取ることができます。永続化された有効なトランザクションは、即座にコミットされる可能性が高まります。

• 永続化されたトランザクションのバッチが順序付けられると、クライアントは通知を受け取ることができます。これにより、クライアントはリモートのバリデータがトランザクションを完了するのを待つ代わりに、ローカルでのトランザクション実行を選択することができます。

• クライアントは、バリデータによるトランザクションの認証を待ち、認証された結果（セクション8を参照）のステート同期を実行することができます。

Aptosのモジュラー設計は開発速度を促進し、変更点を個別のモジュールに絞ることでリリースサイクルの高速化を支援します。このモジュラーデザインは、バリデータを1台以上のマシンにスケーリングするための構造化されたアプローチも提供します。これにより、追加の計算、ネットワーク、ストレージリソースへのアクセスが可能となります。図4では、トランザクションのライフサイクルを各処理段階ごとに示しています。

7.1 バッチ処理

バッチ処理とは、Aptosブロックチェーンの各操作段階において、その操作を効率化させるための重要な処理手段です。この処理がある事により、トランザクションは各バリデータによってバッチにグループ化され、その後、ブロックに結合される際のコンセンサスフェーズにおいて使用されます。このバッチ処理のアプローチは、実行、ストレージ、および台帳の認証フェーズでも採用され、組み替え、重複する計算や署名検証の削減、そして並列実行の可能性を提供します。

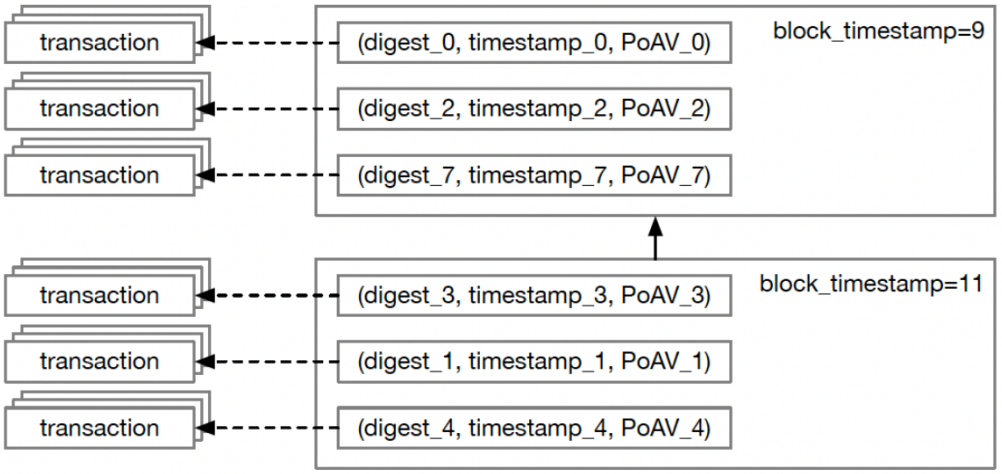

図5：ブロックメタデータの順序付けは、トランザクションの伝播とは独立して行われます。

トランザクションをバッチグループ化することで、わずかなタイムラグが発生することがあります。例えば拡散前の200ミリ秒の待機などの微小な遅れが想定されます。しかし、その最大待機期間や最大バッチサイズは簡単に設定可能で、分散ネットワークは遅延と効率のバランスを自動的に最適化することができます。さらに、バッチ処理を通じて、効率的な料金市場がトランザクションを優先し、不必要なクライアントからのDoS攻撃（＝サービス拒否攻撃）を回避することが可能となります。

7.2 連続的なトランザクションの拡散

Narwhal＆Tuskの研究[9]に基づき、Aptosブロックチェーンではトランザクションの拡散がコンセンサスとは独立して行われます。バリデータ同士はトランザクションのバッチを連続的にストリームし合い、利用可能なネットワークリソースを最大限に活用します。バリデータ v が配布する各バッチは永続化され、そのバッチのダイジェストへの署名が v に返されます。セクション7.3のコンセンサス要件に基づき、2f + 1のステーク加重された署名がバッチダイジェストに対して行われると、利用可能性の証明（PoAv）が形成されます。この証明は、f +1のステーク加重されたプロトコルとネットワークの安全性を維持するための「正直なバリデータ (= 以下、*Honest Validator* ）」が該当バッチを保存していることを確認し、その結果、全ての*Honest Validator* がその実行前にそれを取得可能であることを保証します。

しかし、トランザクションのバッチを無制限に永続化することは、ストレージの枯渇やDoS攻撃のリスクを持ち込む可能性があります。これらを回避するために、各バッチにはタイムスタンプが紐付けられ、その結果、各バリデータでの効率的なガベージコレクションが可能となります。さらに、バイザンタイン攻撃のような最も極端な状況下でも、バリデータがストレージの制限を超えないように、別のバリデーターごとに特定のクォータメカニズムが導入されています。バッチは、永続ストレージに保存される前のサイジング制約を経て検証されます。そして、トランザクションの重複を避け、キャッシュによる最適化を実施することで、ストレージコストが削減され、並列実行エンジンとの高い連携が確保されます。

7.3 ブロックメタデータの順序付け

コンセンサスがブロックチェーンのスループットや遅延の主要な制約であると捉えられ遅いと誤解されています。Aptosブロックチェーンの中心的な革新の一つは、トランザクションの拡散、トランザクションの実行/ストレージ、および台帳の認証のような非合意関連タスクをコンセンサスフェーズから切り離すことです。この分離により、ブロックメタデータと証明のみで非常に低い帯域幅（ブロックメタデータおよび証明のみ）での順序付けが可能となり、極めて効率の良いスループット、そして遅延を最小限に抑えることを実現します。

現在、AptosブロックチェーンはBTFコンセンサスプロトコルを持つ最新のDiemBFTv4[10]を採用してます。標準的な状況下では、コンセンサスは2回のネットワークのラウンドトリップのみを要求し（ 世界中でのラウンドトリップ時間は通常300ミリ秒未満 ）、リーダーの評判メカニズム[11]を通じて不良なバリデータを動的に調整します。このオンチェーンの評判メカニズムは、期間内にブロックを正常にコミットしたバリデータを昇進させ、非参加のバリデータを降格させます。この新しいメカニズムは、分散環境でのパフォーマンス性能の向上を促進し、適切なインセンティブを提供し、失敗したバリデータのスループットと遅延への影響を迅速に最小限に抑えます。

DiemBFTv4は、部分的な同期の下で生存を保証し、総バリデータのステークが≥ 3f +1で、最大fのステーク加重された欠陥のあるバリデータの下で安全性を確保します。DiemBFTv4は、2019年以降のいくつかの反復で、数十のノードオペレーターとマルチウォレットエコシステムで徹底的にテストされてきました。また、最近の研究（ 例：Bullshark[12] ）や、ブロックの履歴と関連するコミュニケーションに依存してブロックメタデータの順序付けと最終性を決定する他のプロトコルとの実験も行っています。

リーダーが提案したコンセンサスブロックと、他のバリデータが合意した提案タイムスタンプは図5に示されています。各コンセンサスブロックにはバッチのメタデータと証明のみが含まれ、実際のトランザクションは含まれていません。これは、PoAVがトランザクションのバッチを実行フェーズで利用可能とするためです。バリデータは、提案とブロックメタデータの要件が満たされていることを確認した上で、リーダーの提案に投票することができます。（例：提案タイムスタンプ≤ブロック有効期限）。

7.3.1 ブロックチェーンの時間

Aptosブロックチェーンは、全てのブロックに固有の物理的タイムスタンプを採用し、このタイムスタンプはブロック内の全てのトランザクションにも適用されます。

このタイムスタンプの採用により、様々な重要なユースケースが実現されます:

- スマートコントラクトでは時間依存のロジックを利用できます。たとえば、開発者はオークションが木曜日の正午までに入札を受け付けるようにエンコードすることができます。
- オラクルがオンチェーンデータを公開する際、リアルタイムのイベントと遅延を関連付けるために正確で信頼性のあるタイムスタンプが必要となります。
- クライアントはブロックチェーンの現在の状態を評価する際、セキュリティ上の理由でアカウントの状態が最後に更新された際の高精度のタイムスタンプにアクセスすることが推奨されます。
- 信頼性のあるタイムスタンプを活用してブロックチェーンを監査することで、法的義務としての支払いが期待どおりに実行されているかの確認が可能となります。
- トランザクションの有効期限は、最も新しい確定タイムスタンプに基づきます。これを利用して、クライアントはトランザクションの有効期限を設定することができます。

Aptosブロックチェーンは、ブロック内の全てのトランザクションに関して、次のようなタイムスタンプの保証を提供します:

- ブロックチェーン内の時間は単調に増加します。例えば、ブロックB1 < ブロックB2であれば、B1.Time < B2.Timeであると保証されます。
- タイムスタンプTでトランザクションのブロックが合意された場合、少なくともf + 1の *Honest Validator* がTがタイムスタンプを「過去」として認識します。これらのバリデータは、自らの時計がタイムスタンプT以上の場合にのみブロックへの投票を行います。
- タイムスタンプTで合意が得られたブロックは、署名のクォーラムを持ち *Honest Validator* は他のバリデータにそのブロックを伝えません。

最新のタイムスタンプは、確認された全てのブロックで更新されます。ネットワークが同期している場合、ブロックのタイムスタンプは迅速に更新され、これにより非常に正確な時計が提供されます。必要に応じて、ブロック内のトランザクションの詳細な順序を確定することもできます

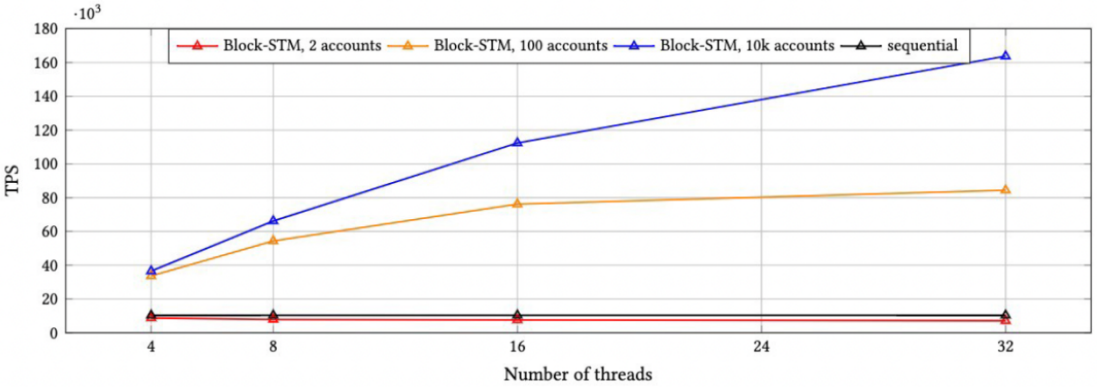

図6 : 物理コアの数と異なるコンテンションレベルを比較した
Block-STM（コンポーネントのみ）のベンチマーク。

7.4 並列トランザクション実行

コンセンサスによりブロックメタデータが順序付けられた後、トランザクションはバリデータ、フルノード、あるいはクライアントによって実行され得ます。提案されるトランザクションバッチは、少なくとも2f +1のステーク加重バリデータにより確実に保存されます。トランザクションの伝播は連続的であり、経過時間とともに、追加の *Honest Validator* がこのバッチを受け取ります。もし、*Honest Validator* が順序付けられたトランザクションバッチを実行段階までに受け取らなかった場合、少なくともf + 1のステーク加重バリデータ（ステーク加重PoAV署名者の半分以上）が正であるとして、それは2f +1のステーク加重バリデータからダウンロードすることが可能です。

ブロックチェーンの最も重要な目標は、最大限の並列実行をサポートすることです。Aptosブロックチェーンは、データモデルと実行エンジンの両方でこの目標に取り組んでいます。

7.4.1 並列データモデル

Moveのデータモデルは、データとモジュールのグローバルアドレス指定をサポートしています。競合するアクセスがない限り、トランザクションは並列で実行可能です。Aptosブロックチェーンのパイプライン設計を考慮すると、トランザクションの順序を再配置することで競合を最小化し、並行実行性を向上させることができます。

さらに、トランザクションが同じオンチェーンデータを変更する場合でも、多くの実行プロセスは並列に行われます。ここで、Aptosは「デルタライト」という新しい概念を採用します。これは、アカウントの状態を直接変更するのではなく、その変更を表すものです（例：整数を増分するのではなく、最終値を決定する）。全てのトランザクションは並列に処理され、後にデルタライトが正確な順序で適用され、結果の一貫性が確保されます。

Aptosブロックチェーンはデータモデルを継続的に強化し続け、同時実行性（例：読み書きのヒントを活用する）を向上させるとともに、開発者がオンチェーンの値を作成、変更、組み合わせるのがより自然になるように改善します。Moveは、この種の改善をサポートするための柔軟性を、言語レベルだけでなくプラットフォーム特有の機能を通じて提供しています。

7.4.2 並列実行エンジン

Block-STMの並列実行エンジンは、特定の順序付けを持つトランザクションセットの競合を検出し、管理します。これにより、最適な並行制御を用いて、特定の順序付けを考慮した最大の並列性を実現します [13]。

トランザクションはバッチ処理された後に、最適化された並列処理を経て実行され、その後、検証作業が行われます。検証に失敗した場合、再度実行されます。Block-STMは、書き込み同士の競合を防ぐためにマルチバージョンのデータ構造を使用します。同じ場所への全ての書き込みは、トランザクションIDと最適な再実行回数を含むバージョン情報とともに保存されます。

トランザクションtxがメモリの位置を読む際、マルチバージョンデータ構造から、設定順序でtxの前に実行された最新のトランザクションによってその位置に書き込まれた値と、関連するバージョン情報を取得します。

Block-STMは、Aptosブロックチェーンに既に統合されています。Block-STMの性能を最大限に引き出すため、メモリ内データベースとともに孤立した実行のみのベンチマークとして、非トリビアルなP2P Moveトランザクションでの実験を行いました。図6ではBlock-STMの実行結果を示しており、各ブロックには10,000のトランザクションが含まれています。アカウントの数により、競合と競争のレベルが決定されます。

低競合環境下で、Block-STMは32スレッドを使って逐次実行に対して16倍の高速化を達成しました。一方、高競合環境下では8倍以上の高速化を実現しました。他のブロックチェーンの並列実行エンジンとは異なり、Block-STMは任意のワークロードからの固有の並列性を動的かつ透明に抽出できます。データの位置の事前知識を必要とする他の並列実行環境と比較して、Block-STMはより複雑なトランザクションを同時に処理する能力があります。これは、より少なく効率的なトランザクションにつながり、コストとユーザーの遅延を低減します。最も重要なのは、単一の複雑な状態を持つトランザクションのセマンティクスを保持しつつ、小さなトランザクション群に分割することで、開発者は最適な環境を得られることです。

ブロックメタデータの順序付けステップは、並列実行段階でトランザクションを再度並べ替えることを妨げるものではありません。最適な並行性を目指して、トランザクションは1つ以上のブロックで順序を変更することができます。この並べ替えは、全ての正直な検証者間で一貫性が保たれる必要があります。並列実行を最適化するために並べ替えをランダム化することで、パフォーマンスが向上し、最大の利益を追求するためのトランザクションの再並べ替えを避けることができます。この設計には「Order-then-reveal」のMEV耐性戦略も組み込むことができます。

Block-STMとトランザクションの再順序付けは、実行の並列性を増強するための補完的な技術です。これらは、さらなる並行性を得るためのトランザクションの読み/書きヒントと組み合わせることができます。

7.5 バッチストレージ

並列実行フェーズの後、全てのトランザクションに対するライトセットが生成されます。これらは、最適な実行速度のためにメモリに保存され、後続のブロックやブロックセットのキャッシュとして活用できます。重複する書き込みは、安定したストレージに一度のみ書き込まれるべきであり、仮に検証者がライトセットをメモリに保存する前の段階で障害が発生した場合、検証者はブロックメタデータの順序付けフェーズから並列実行を再スタートできます。バッチストレージを並列実行から分離することで、並列処理が効率的に動作することが保証されます。要点として、ライトセットのバッチ処理は、ストレージの操作回数を削減し、効率的な大規模なI/O操作を可能にします。

キャッシュとして使用するメモリの量は、マシンごとに手動で調整でき、これによりバックプレッシャーの調整メカニズムが提供されます。並列実行ブロックとは異なり、バッチの粒度は特定のI/Oやメモリ環境に合わせてチューニングされることがあります。

7.6 レジャーの認証

パイプラインのこの段階では、各検証者はコミットされたトランザクションブロックの新しい状態を算出しています。しかし、Aptosブロックチェーンは、軽量クライアントとステートの同期を効率的にサポートするため、レジャーの履歴と状態の認証を導入しています。Aptosブロックチェーンの特徴的な点は、レジャー認証が取引処理の中心的なルート上になく、必要に応じて独立して実行できることです。

7.6.1 レジャー履歴の認証

検証者は、トランザクションとその実行結果を一つのグローバル認証済みレジャーデータ構造に組み込みます。トランザクションの出力の一部として、Moveでアクセスできるグローバルステートへの変更を示すステートのライトセットが含まれます。このデータ構造の短い認証子は、新たに実行されたトランザクションのバッチを含むレジャー履歴への確約として機能します。トランザクションの実行と同じく、このデータ構造の生成は確定的です。

各検証者は、得られたデータベースの新版に対する短い認証子への署名を行います。検証者同士で、最近の署名済みの短い認証子を共有し、クォーラムによって署名された認証子を集約します。また、その集約された認証子をお互いに共有します。

この共同署名を用いて、クライアントはデータベースのバージョンが、プロトコルのBFTプロパティに基づき、完全で正確、かつ不変のレジャー履歴を反映していると確信できます。クライアントは、任意の検証者やデータベースの第三者レプリカ（例: フルノード）に対してデータベースの値の照会を行い、必要なデータの証拠と認証子を用いてその結果を検証します。

7.6.2 定期的な状態の認証

Moveによってアクセスできるグローバルステートは、任意の時点でのレジャー履歴の概要と同じように、短い認証子に要約できます。グローバルステートはランダムアクセスが可能なため（レジャー履歴は追加のみとは異なり）、この認証の維持コストは高いです。それにもかかわらず、大量のデータを一度に更新する際には、更新を並列で行うことができ、また、各ステート値が変わるたびの更新のオーバーラップを最大限に利用できます。Aptosブロックチェーンは、重複する更新を削減するために、グローバルステートの認証を定期的に行うように設計されています。

設定した間隔で、ネットワークは、その出力としてグローバルステートの認証子を含むステートのチェックポイントトランザクションを発行します。このようなバージョンはステートのチェックポイントとして示されます。2つのチェックポイント間の間隔が長いほど、トランザクションごとの認証データ構造の更新のコストは低下します。

ステートチェックポイントを利用すれば、グローバルステート全体を保存することなく、その中の任意のステート値を信頼性を持って読み取ることができます。この機能は、逐次的なステート同期や、検証者間での分散ストレージ、ステートレスな検証者ノード、ストレージの制約があるライトクライアントなどに役立ちます。

しかし、チェックポイントは定期的であるため、レジャーの特定のバージョンに関する証明を取得するには、不足しているステートの変更のための追加トランザクションや、認証済みのレジャー履歴からの取り込み証明が必要です。

ステートチェックポイントは、レジャー履歴の特定のトランザクションバージョンに関連づけられており、セクション7で述べられているトランザクションバッチのタイムスタンプと連動しています。タイムスタンプを利用することで、ライトクライアントは証明されたステート値の新旧を確認できます。タイムスタンプがなければ、ライトクライアントの証明は、過去のものとしての有効性しか保証できません。また、アクセスの追跡や監査、例えばトークンリザーブの時間当たりの平均バランスの計算などのために、ステートの証明のタイムスタンプは必要です。

ステートチェックポイントは、前のチェックポイントと、それ以降のトランザクション出力のステート変更に基づいて導出できます。そのため、ステートチェックポイントを安定したストレージに保存する必要は、トランザクションの処理のクリティカルパス上にはありません。また、ステートチェックポイントを保存する際には、バッチ処理の利点も得られます。最近のステートチェックポイントをメモリにキャッシュし、周期的なステートチェックポイントだけを安定したストレージに保存することで、ストレージの帯域利用を大幅に削減できます。認証データ構造の計算には影響を与えないので、これはノードごとの選択です。ノードオペレーターは、メモリの容量とストレージ帯域の間で適切なトレードオフを設定することができます。

8 状態同期

Aptosブロックチェーンは、全ての参加者に対して高スループットと低レイテンシを提供することを目的としています。このため、ブロックチェーンは、ライトクライアント、フルノード、および検証者[14] へのブロックチェーンデータの拡散、検証、保存のための効率的な状態同期プロトコルを持つ必要があります。さらに、この同期プロトコルはネットワークのリソース制約や多様性にも耐える必要があり、異なるユーザーや利用シナリオを考慮に入れる必要があります。たとえば、アーカイブフルノードはブロックチェーンの全履歴と状態を検証・保存する一方、ライトクライアントはAptosブロックチェーン状態の一部のみを効率的に追跡することが求められます。

これらの事を実現するために、Aptosブロックチェーンは、検証者やフルノードから提供される認証済みのレジャー履歴や認証済みステート証明（セクション7.6.1参照）を活用して、柔軟かつ設定可能な同期プロトコルを提供します。具体的には、ネットワークの参加者は、それぞれの用途や要件に応じて異なる同期戦略を選択することができます。

例えば、フルノードに対してAptosは始めからの全トランザクションを処理する方法や、ブロックチェーンの履歴をスキップして最新の状態のみを同期する方法など、複数の同期戦略を提供しています。ライトクライアント向けには、部分的なブロックチェーン状態の同期や、特定のアカウント情報の検証付き読み取りなどの戦略が含まれます。Aptosは、取得・処理・保持するデータの範囲や時期を参加者が自由に設定できるようにしています。

柔軟かつ設定可能な状態同期のアプローチを採用することで、Aptosは様々なクライアントの要件に適応し、今後も新しいそして更に効率的な同期戦略を提供し続けることができると考えられます。

9 コミュニティ所有

Aptosブロックチェーンは、多様性豊かなコミュニティによって所有、運営、そして管理されることを予定しています。Aptosトークンの主要な役割として、トランザクション及びネットワークの料金、ガバナンス投票、そしてプルーフ・オブ・ステークモデルを通じたブロックチェーンの保護が挙げられます。

9.1 トランザクションとネットワーク料金

Aptosの全てのトランザクションはAptosトークンで指定されたガスの単価を持っており、これにより検証者はネットワーク内で最も価値あるトランザクションを優先して処理することができます。また、パイプラインモデルの各ステージにおいて、低い価値のトランザクションを排除する機会が複数回存在します。これにより、システムがフル稼働時でも、ブロックチェーンは効率的に動作します。時間の経過とともに、ネットワークの手数料が実施されることで、Aptosブロックチェーンの使用料がハードウェア導入、維持、ノードの運営といった実際のコストに見合ったものとなりま

す。さらに、開発者たちは、計算、ストレージ、ネットワーキングの間でコストを最適に調整できるアプリケーションの設計の機会を得ることができます。

9.2 ネットワークガバナンス

Aptosブロックチェーンでの主要な機能変更や改善は、提案から実装、テスト、そしてデプロイメントの段階を経て行われます。このフローは、各関係者やステークホルダーに、フィードバックを提供したり、懸念点を共有したりする機会を設けます。

デプロイメント段階は通常、2つのステップで進行します。初めに、新しい機能を搭載したソフトウェアが各ノードに展開され、その後、該当の機能が有効化されます。例えば、機能フラグやオンチェーンの設定変数を使って実現される場合が考えられます。

ノードオペレーターによるソフトウェアの展開時には、新しいソフトウェアが以前のバージョンとの互換性を保つ必要があります。ソフトウェアの展開は、タイムゾーンの違いや外部要因を考慮して、数日間にわたって行われることがあります。

多数のノードがアップグレードされた後、新しい機能の有効化は同期ポイント（合意されたブロックの高さやエポックの変更など）で開始されます。緊急事態に際しては、手動での強制的な変更やネットワークのハードフォークによって有効化が行われることも考えられます。

他のブロックチェーンと異なり、Aptosブロックチェーンはその設定をオンチェーン上にコーディングします。全ての検証者は、ブロックチェーンの最新状態に同期する能力を持ち、オンチェーンの設定に基づき適切な設定を自動的に選択することが可能です。この特徴により、Aptosブロックチェーンのアップグレードは迅速かつスムーズに行われます。

有効化プロセスに柔軟性と設定の変更可能性を提供するために、Aptosブロックチェーンは、トークン保有者がそのステークトークンの重みに応じて投票することができるオンチェーンガバナンスをサポートします。オンチェーンの投票プロトコルは公開されており、検証可能で、瞬時に実施することができます。オンチェーンガバナンスは、ソフトウェアのデプロイメント無しで非バイナリの結果の有効化もサポートできます。例えば、オンチェーンのリーダー選出プロトコルのパラメータは、オンチェーンのガバナンスで変更可能です。一方、予め知られている同期ポイントは、全ての変更が事前に知らされていなければならないため、動的な変更処理をすることはできません。

時間の経過とともに、オンチェーンのガバナンスはアップグレード管理プロセス全体に適用されることが期待できます。例として、以下の手順が考えられます:

1. トークン保有者は新しい量子耐性の署名スキームへの移行についてオンチェーンで投票します。
2. 開発者は新しい署名スキームを実装、検証し、新しいソフトウェアをリリースします。
3. 検証者は新リリースにソフトウェアをアップグレードします。
4. トークン保有者は新しい署名スキームをオンチェーンで有効にする投票を行い、オンチェーン設定が更新され、変更が有効化されます。

Aptosブロックチェーンはオープンソースプロジェクトとして、コミュニティからのフィードバックを大切にし、その適切な運営のためにオンチェーンのガバナンスを活用しています。特定の状況でのオフチェーンのアップグレードが必要となる場合もありますが、これは長期的には最小化される見込みです。

9.3 PoS（プルーフ・オブ・ステーク）コンセンサス

Aptosブロックチェーンでトランザクションの検証に参加するためには、バリデータは一定量のAptosトークンをステーキングする必要があります。ステークされた量は、トランザクションの伝播時の2f + 1ステーク加重PoAvや、ブロックメタデータの順序付けの際の投票やリーダー選出に影

響します。バリデータは、自分たちとそのステーカーの間での報酬分割を決定します。ステーカーは、事前に合意した報酬の分割のために、彼らのトークンをステークするバリデータを上限数無く選ぶことができます。各エポックの終わりに、バリデータとそのステーカーは、関連するオンチェーンのMoveモジュールを通じて報酬を受け取ります。 バリデータは自分とステーカーとの間の報酬分割を決めます。ステーカーは報酬分割を前もって合意して、複数のバリデータにトークンをステーキングすることができます。それぞれのエポックの後で、関連するオンチェーンMoveモジュールを通じて、バリデータとステーカーは報酬を受け取ります。

また、十分なトークンをステーキングしているバリデータは、Aptosブロックチェーンに自由に加わることができます。必要なステークの最低量などの詳細は、セクション9.2に記載のオンチェーンプロセスで調整可能です。

10 性能

Aptosブロックチェーンは、並列処理、バッチ最適化、およびモジュラートランザクション処理というパイプラインを通じて、高いスループットとハードウェア効率を達成しています。これはセクション7で詳しく取り上げられています。今後のアップグレード、例えばコンセンサスの改善、デルタライト、トランザクションヒント、クリティカルパスキャッシングなどは、パフォーマンスをさらに高める見込みです。

ブロックチェーンのスループットを「1秒あたりのトランザクション数」として評価するのは適切ではなく、これが他のシステムとの比較において不適切である可能性が指摘されています。トランザクションの種類やインフラのコストと複雑さなどが多岐にわたるため、一律の評価基準としては適切ではないからです。一部のシステムは、トランザクションを事前に分割し、より小さいものにすることを要求するケースもありますが、Aptosでは、開発者が自由に制約無しで構築できる環境を提供し、実際のユースケースを基にしたスループットとレイテンシの測定を重視しています。

ブロックチェーンの展開には、明確なトレードオフが存在します。どのブロックチェーンにおいても、並列実行の能力を持つ場合、それに応じて高性能のハードウェアが必要になるか、あるいは各バリデーターを個別のマシンクラスタとして設定し、追加の並行性を実現することが考えられます。しかしながら、バリデーターオペレーターの運用コストやその複雑さを考慮すると、グローバルなバリデーターの数には実質的な上限が存在します。また、クラウドサービスにおけるサーバーレスデータベースの普及やその高まる人気は、これら複雑な分散システムの効率的なデプロイや維持が可能なエンティティは限られていることを示唆しています。そのため、ネットワークとしてのAptosの取り組みでは、バリデーターの性能最適化だけでなく、新たなバリデーターの追加に関する研究もしています。

10.1 同種のステートシャーディング

Aptosブロックチェーンは、当初、単一のレジャーステートで始まります。しかし、時間の経過とともに、Aptosは中央集権を維持しつつ、水平スケーラビリティを追求するユニークなシャーディング手法を導入する予定です。このシャーディングの特徴として、均一なAPIの提供 (シャーディングを第一のコンセプトとして複数のシャーデッドレジャーステートを通じて行われます) や、Aptosトークンのユニバーサルな利用 (トランザクション手数料、ステーキング、およびガバナンスへの使用。) 、そしてシャード間のデータ転送を可能とする均一なブリッジが挙げられます。

ユーザーや開発者は、シャーディングの柔軟性を利用して、自分のニーズに応じた最適な方法を選択できます。さらに、異なるシャードは、異なるハードウェア特性を持つことができ、一つのシャードはSSDを持つ計算に最適化されているかもしれませんし、もう一つは低い計算特性を持つ大容量のハードドライブに最適化されているかもしれません。これにより開発者はアプリケーションの要件に合わせて最適なシャードを利用できます。

同種のステートシャーディングは、水平スループットのスケーラビリティを向上させるだけでなく、

シャードを跨いで単一の普遍的な状態でプログラムすることを開発者に許可し、シャーデッドデータの取り扱いを簡単にするウォレットの提供や、Moveスマートコントラクトプラットフォームの利用を可能にするなど、多岐にわたるメリットを持ちます。

参考文献・資料

[1] “Aptos-core,” 2022. [Online]. https://github.com/aptos-labs/aptos-core

[2] “Move,” 2022. [Online]. https://github.com/move-language/move

[3] D. Matsuoka, C. Dixon, E. Lazzarin, and R. Hackett. (2022) Introducing the 2022 state of crypto report. [Online]. https://a16z.com/tag/state-of-crypto-2022/

[4] Z. Amsden, R. Arora, S. Bano, M. Baudet, S. Blackshear, A. Bothra, G. Cabrera, C. Catalini, K. Chalkias, E. Cheng, A. Ching, A. Chursin, G. Danezis, G. D. Giacomo, D. L. Dill, H. Ding, N. Doudchenko, V. Gao, Z. Gao, F. Garillot, M. Gorven, P. Hayes, J. M. Hou, Y. Hu, K. Hurley, K. Lewi, C. Li, Z. Li, D. Malkhi, S. Margulis, B. Maurer, P. Mohassel, L. de Naurois, V. Nikolaenko, T. Nowacki, O. Orlov, D. Perelman, A. Pott, B. Proctor, S. Qadeer, Rain, D. Russi, B. Schwab, S. Sezer, A. Sonnino, H. Venter, L. Wei, N. Wernerfelt, B. Williams, Q. Wu, X. Yan, T. Zakian, and R. Zhou, “The libra blockchain,” 2019. [Online]. https://developers.diem.com/papers/the-diem-blockchain/2020-05-26.pdf

[5] S. Blackshear, E. Cheng, D. L. Dill, V. Gao, B. Maurer, T. Nowacki, A. Pott, S. Qadeer, D. R. Rain, S. Sezer, T. Zakian, and R. Zhou, “Move: A language with programmable resources,” 2019. [Online]. https://developers.diem.com/papers/diem-move-a-language-with-programmableresources/2019-06-18.pdf

[6] D. Dill, W. Grieskamp, J. Park, S. Qadeer, M. Xu, and E. Zhong, “Fast and reliable formal verification of smart contracts with the move prover,” in *Tools and Algorithms for the Construction and Analysis of Systems*, D. Fisman and G. Rosu, Eds. Cham: Springer International Publishing, 2022, pp. 183–200.

[7] N. Popper. (2021) Lost passwords lock millionaires out of their bitcoin fortunes. [Online]. https://www.nytimes.com/2021/01/12/technology/bitcoin-passwords-wallets-fortunes.html

[8] The Diem Team, “State synchronization and verification of committed information in a system with reconfigurations,” 2020. [Online]. https://github.com/aptos-labs/aptoscore/blob/main/documentation/tech-papers/lbft-verification/lbft-verification.pdf

[9] G. Danezis, L. Kokoris-Kogias, A. Sonnino, and A. Spiegelman, “Narwhal and tusk: A dag-based mempool and efficient bft consensus,” in *Proceedings of the Seventeenth European Conference on Computer Systems*, ser. EuroSys ’22. New York, NY, USA: Association for Computing Machinery, 2022, p. 34–50. [Online]. https://doi.org/10.1145/3492321.3519594

[10] The Diem Team, “Diembft v4: State machine replication in the diem blockchain,” 2021. [Online]. https://developers.diem.com/papers/diem-consensus-state-machine-replication-in-the-diemblockchain/2021-08-17.pdf

[11] S. Cohen, R. Gelashvili, L. Kokoris-Kogias, Z. Li, D. Malkhi, A. Sonnino, and A. Spiegelman, “Be aware of your leaders,” *CoRR*, vol. abs/2110.00960, 2021. [Online]. https://arxiv.org/abs/2110.00960

[12] A. Spiegelman, N. Giridharan, A. Sonnino, and L. Kokoris-Kogias, “Bullshark: Dag bft protocols made practical,” in *Proceedings of the 20th Conference on Computer and Communications Security (CCS)*, ser. CCS ’22. Los Angeles, CA, USA: Association for Computing Machinery, 2022.

[13] R. Gelashvili, A. Spiegelman, Z. Xiang, G. Danezis, Z. Li, Y. Xia, R. Zhou, and D. Malkhi, “Block-stm: Scaling blockchain execution by turning ordering curse to a performance blessing,” 2022. [Online]. https://arxiv.org/abs/2203.06871

[14] J. Lind, “The evolution of state sync: The path to 100k+ transactions per second with sub-second latency at aptos,” 2022. [Online]. https://medium.com/aptoslabs/52e25a2c6f10