Coincheck

昨今のデータモデルと分散台帳のCBDCへの利用可能性

2024.03.13

コインチェック株式会社

杉本 秀和　迫田 晃祐　金 栄洙　赤根谷 哲雄

目次

AGENDA

目次

AGENDA

1. データモデルの種類

CBDCのサーバーサイドのデータモデルを考える際、データモデルの全体感の認識を揃えられるよう、ここでは既存（データベース技術）と新規（ブロックチェーン技術）のデータモデルを分類する。
また次のページ以降でデータモデルの具体的な内容を説明する。

	リレーショナル データベースモデル	NoSQL データベースモデル	NewSQL データベースモデル	分散台帳 データモデル
説明	1970年にコッド氏によって提唱されたデータベースモデルが発端となって普及した。 ACIDトランザクション※1を実装し、データの整合性と信頼性を担保する。垂直拡張※2は行えるが、水平拡張（分散）には適していない。 2024年3月現在では、多くの会社で採用されている。	リレーショナルモデルが普及した後に、リレーショナルモデルでは対応しきれない大規模なデータ群を処理に対応するためにNoSQLが普及した。 ACIDトランザクション非対応※3だが、非構造化データや半構造化データを取り扱うことができる。 モデルによっては水平拡張できるものがある。	さらなるリレーショナルモデルの拡張性を求め、リレーショナルモデルを用いながら水平拡張できるNewSQLが現在普及が進んでいる。 ACIDトランザクションに対応した拡張性のあるデータベースモデルであり、コンセンサスアルゴリズム(Paxos, Raftなど)を採用している。	2008年のSatoshi Nakamotoの論文を発端として、Bitcoinが一般に話題となり、Ethereumや分散台帳が暗号資産（仮想通貨）として利用されている。 コンセンサスアルゴリズム（PoW, PoSなど※4）を採用している。
データモデル詳細	• リレーショナルモデル & SQL & 垂直拡張	• キー・バリューストア • カラムストア • ドキュメント指向 • グラフベース	• リレーショナルモデル & SQL & 垂直・水平拡張	• UTXOモデル • アカウントモデル • UTXO/アカウントのハイブリットモデル • DAGモデル

※1 Atomicity（原子性）、Consistency（一貫性）、Isolation（独立性）、Durability（永続性）を指す
※2 サーバーのCPUやメモリなどのリソースを増やして処理能力を向上させる
※3 多くのNoSQLではBASE（Basically Available, Soft state, Eventually consistent）モデルを採用しており、最終的な一貫性を担保している
※4 正確なアルゴリズムはNakamoto Consensus等が正しい表現だが、ここでは掘り下げない

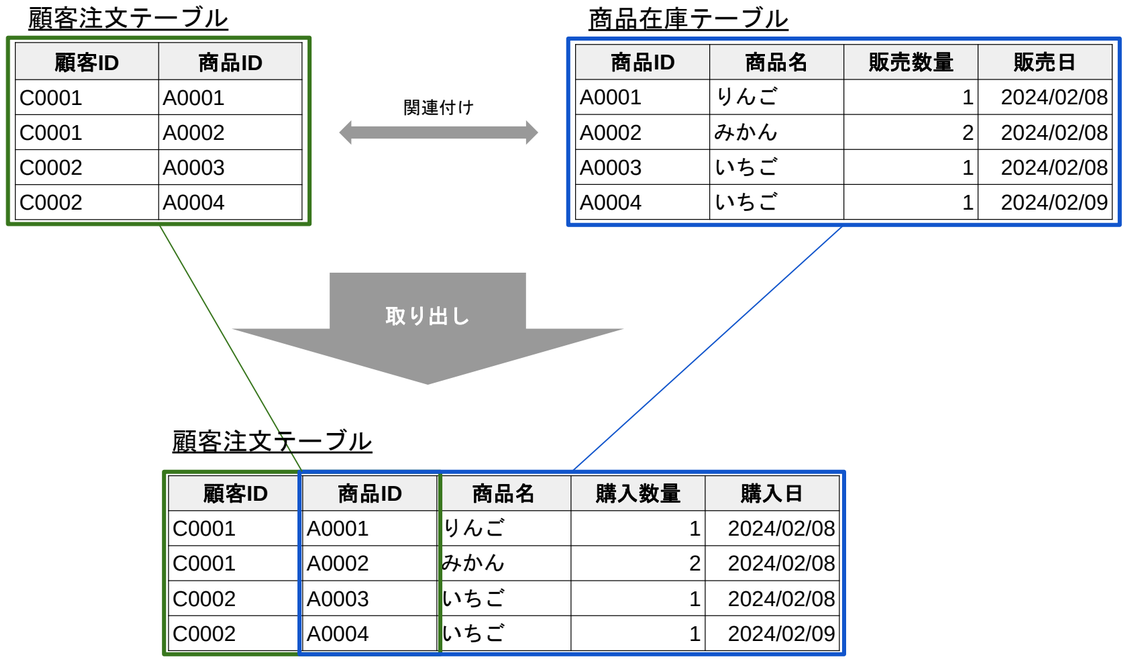
目次

AGENDA

1-a. データベース技術の特徴：リレーショナルデータベースモデル

リレーショナルデータベースモデルの特徴は、以下のとおり。

- 表型式（行・列）でデータを持ち、データ間で関係性を定義するデータモデル
- SQL（Structured Query Language）で複雑なクエリを実行可能
- 代表的なデータベースは、「MySQL」、「PostgreSQL」など



メリット

- ACID特性を備えており、データの整合性と信頼性が高い
- 複雑なクエリとトランザクションが利用できる
- 垂直拡張（サーバーのリソース増強、例えばサーバーCPUやメモリなど）に優れている
- 歴史があり、多くのシステムで利用されており、検索で多くの確立された技術を拾うことができる

デメリット

- 巨大なデータセットや高いスループットが要求される場合ではパフォーマンスが低下する可能性がある
  - また関係の深い情報の探索などもパフォーマンスが低下する可能性がある
- 水平拡張（複数のサーバーにデータ分散）が難しく、垂直拡張に頼らざるを得ない

1-a. データベース技術の特徴：NoSQL - キー・バリュー

NoSQL キー・バリューストアの特徴は、以下のとおり。

- １つの「キー」に対して１つの「バリュー」を格納し、この2つの要素を組み合わせて構築するデータモデル
- データアクセスの高速化を実現するためにメモリやSSDにデータを格納する場合が多い
- キャッシュサーバーやログの蓄積に向いている
- 代表的なデータベースは、「Redis」、「DynamoDB」など

キー	バリュー
tag-1	りんご, 青森, 佐藤
tag-2	みかん, 和歌山, 鈴木
tag-3	いちご, 栃木, 田中

りんご, 青森, 佐藤

メリット

- 構造がシンプルなため、容量が少なく、非常に高速な読み書きが可能
- スキーマレスであるため、柔軟なデータモデルを持つ
- 分散システムでのスケーラビリティとパフォーマンスに優れている

デメリット

- データ構造がシンプルすぎて、複雑なクエリやデータの関係性の表現には不向き
- 検索機能が限られており、複雑なクエリには適していない

1-a. データベース技術の特徴：NoSQL - カラムストア

NoSQL カラムストアの特徴は、以下のとおり。

- 「キー・バリュー型」にカラム（列）を加え、列ごとに圧縮やパフォーマンスの最適化を行ったデータモデル
- 列方向の操作が容易にでき、大規模データの処理に向いている。
- ログの蓄積や分析処理に向いている
- 代表的なデータベースは、「Cassandra」、「BigTable」など

行キー	種類1	種類2	種類3	種類4	種類5
果物	もも	すいか	りんご	みかん	いちご
生産地	山梨	熊本	青森	和歌山	栃木
生産者	高橋	田中	佐藤	鈴木	
魚	マグロ	タイ			

“生産地”で取り出し

種類1	種類2	種類3	種類4	種類5
山梨	熊本	青森	和歌山	栃木

メリット

- 大量のデータに対して高速な読み込みが可能
- カラム指向型は列方向にデータを処理するため、任意の列をまとめて処理するのに適している
  - 特定の列を抜き出して行うような集計処理が得意
- 列ごとの圧縮・最適化を行うため、ストレージの効率が良い

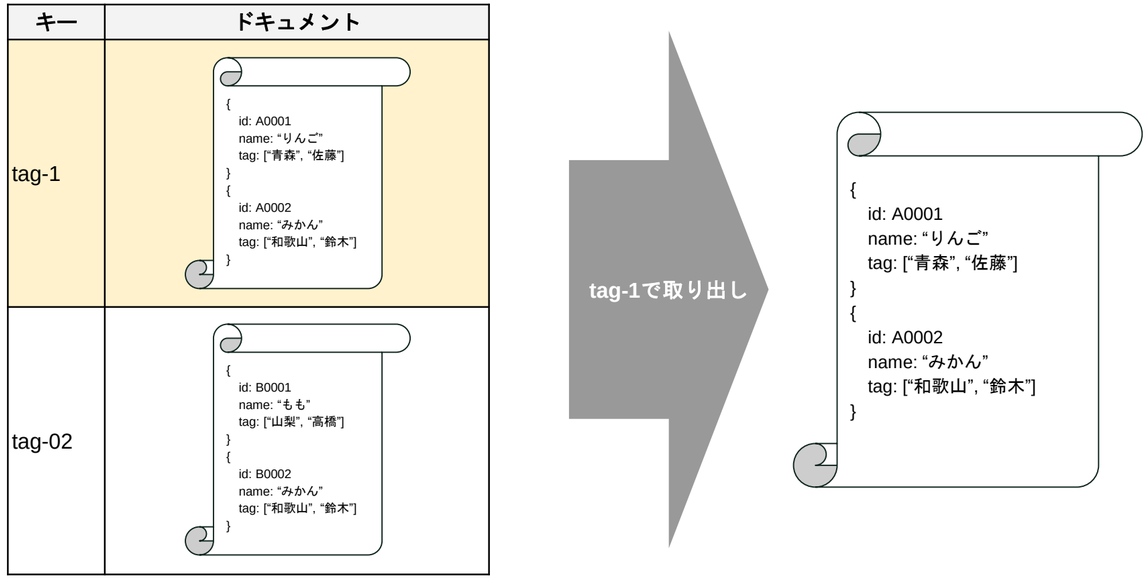
デメリット

- 列単位のデータ格納と圧縮と最適化を行っているため、書き込み処理が遅くなる
- 複雑なトランザクションには向いていない
  - リレーショナルデータベースモデルほどの柔軟性がない

1-a. データベース技術の特徴：NoSQL - ドキュメント指向

NoSQL ドキュメント指向の特徴は、以下のとおり。

- XMLやJSONなどのドキュメント形式でデータを持つデータモデル
- スキーマレスでデータ構造の定義を持たないため、複雑なデータ設計が不要
- Webシステムや口グ分析などが向いている
- 代表的なデータベースは、「MongoDB」、「Couchbase」など



メリット

- データ構造が柔軟で、様々な形式のデータを扱いやすい
  - 柔軟なデータモデルを持ち、非構造化データや半構造化データの管理に適している
- アプリケーションの開発が容易に行える

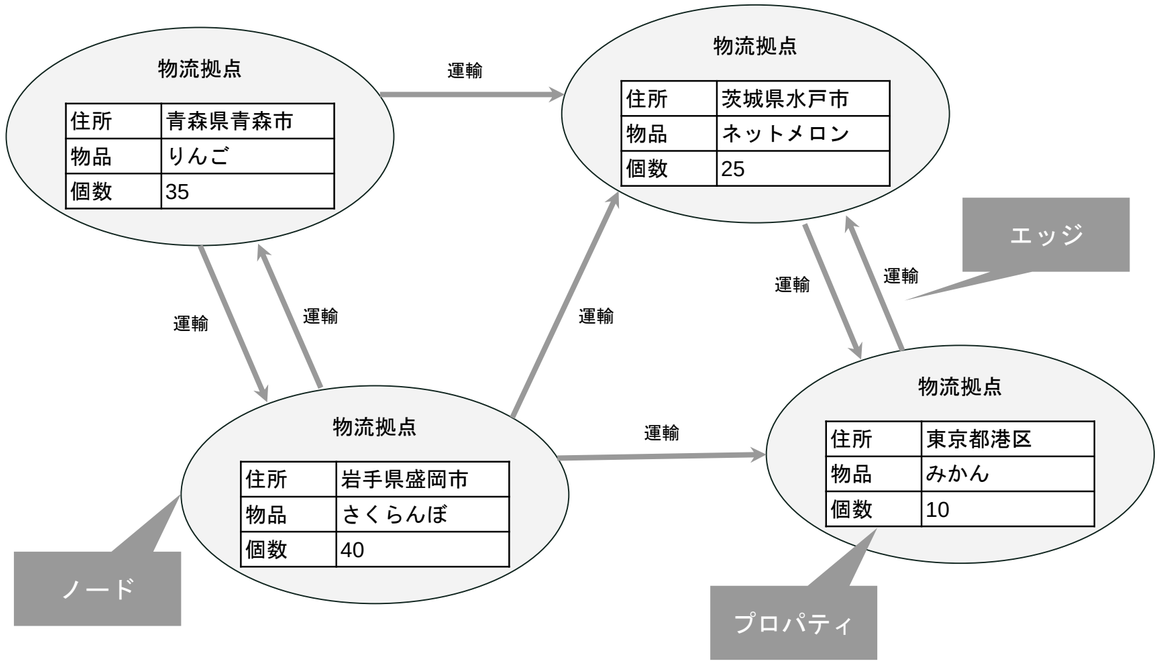
デメリット

- スキーマの不一致が発生しやすい
- データの整合性を保つのが難しい場合がある
- 非構造化や半構造化データを持つため、複雑なクエリの実行速度が遅くなることがある
  - 複雑なクエリには最適化が必要

1-a. データベース技術の特徴：NoSQL - グラフベース

NoSQL グラフベースの特徴は、以下のとおり。

- グラフ理論に基づいたデータモデル
  - 「ノード」：データの実態、「エッジ」：ノード同士の関係性、「プロパティ」：ノードやエッジの属性情報で表現する
- 複雑な関係性を持つデータの分析に適している
- 代表的なデータベースは、「Neo4j」、「InfiniteGraph」など



メリット

- 複雑な関係性（関係性の深いデータ等）を持つデータのクエリが高速に行える
  - 複雑なネットワーク分析が可能
- データ間の関係性を直感的に理解しやすい

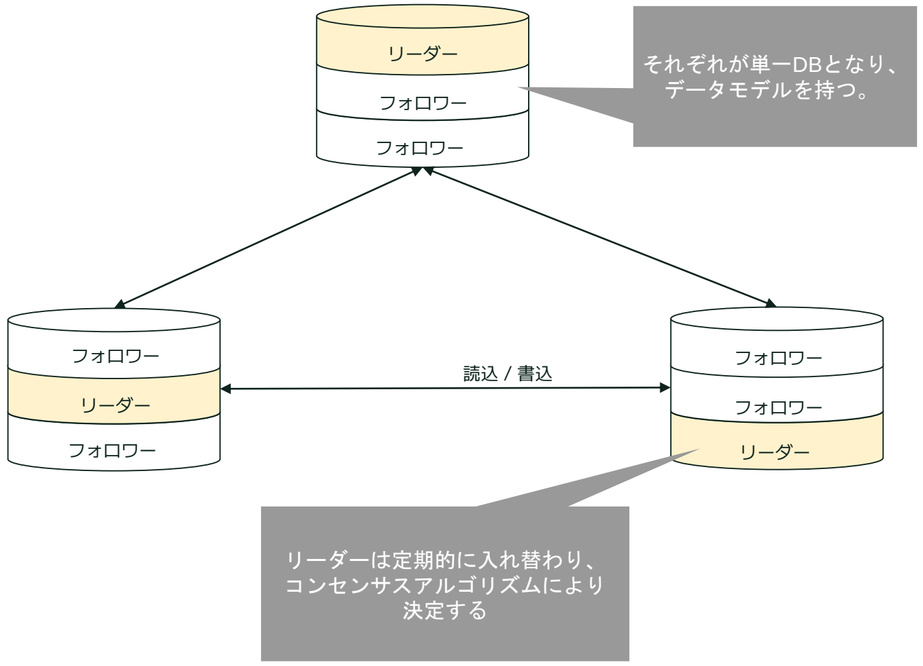
デメリット

- グラフ特有のクエリ言語を学ぶ必要がある
  - 大規模なグラフの管理と分析には専門知識が必要となる
- スケーリングが他のNoSQLモデルに比べて複雑
- グラフDBはデータ全体からの統計情報を分析する手続きなどは苦手

1-a. データベース技術の特徴：NewSQL

NewSQLの特徴は、以下のとおり。

- RDBMSが持つACID特性を保持しつつ、水平拡張（複数サーバーへの分散）をコンセンサスアルゴリズムで実現したデータモデル
  - 高い整合性と水平拡張、分散SQLを実現。Paxos, Raftなどのアルゴリズムが用いられている
- 代表的なデータベースは、「Spanner」、「YugabyteDB」、「TiDB」、「CockroachDB」など



各ノードが読書を実施する。
Raftアルゴリズムによりリーダーが入れ替わり負荷分散される

メリット

- 大規模トランザクション処理が可能でありながら、リレーショナルデータベースモデルの利点を保持している
- 高い整合性とスケーラビリティを備えている

デメリット

- 新しい技術であるため、確立された技術やサポートの面で不安がある
- 実装や運用に関する専門知識が必要になる場合がある
- 分析クエリが不向き
  - HTAPを付加しインメモリでトランザクション処理と分析処理を実現している製品がある

1-a. データベース技術の比較

既存のデータベース技術で利用されているデータモデルを一覧にし、各それぞれの強み、弱みを比較すると以下となる。

※ 評価は実際には製品の特徴により、変わる。

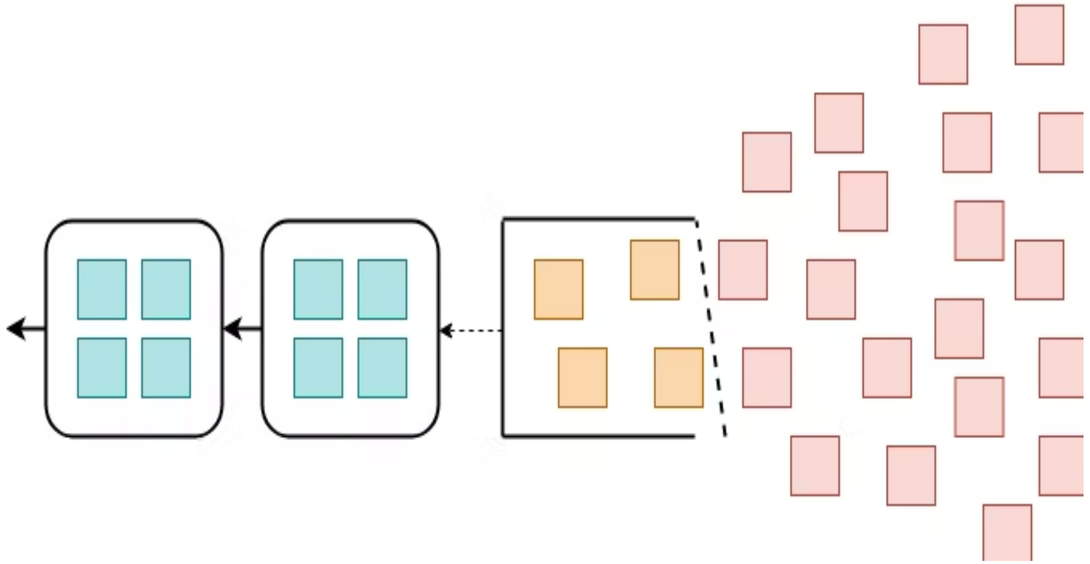
	リレーショナルデータベースモデル	NoSQLモデル				NewSQLデータモデル
		キー・バリューストア	カラムストア	ドキュメント指向	グラフベース	
データの扱いやすさ	◎ 構造化データに強い	△ シンプルで高速だが、複雑なデータ構造には不向き	○ 分析向けに最適化されたデータ構造	◎ 柔軟なデータ構造で複雑なデータを扱いやすい	○ 関係性の表現に優れている	◎ RDB同様、構造化データに強い
パフォーマンス（レイテンシー）	○ 一般的な用途では良好だが、大規模データでは限界がある	◎ 単純な構造で読み書きが非常に高速	◎ 単純な構造で読み書きが非常に高速	○ クエリによってはパフォーマンスが落ちることがある	○ 関係性のクエリにおいて高性能だが、一般的な用途ではオーバーヘッドが大きい	○ 高いスケーラビリティ実現により、レイテンシーの優位性が低くなる
スケーラビリティ（スループット）	○ 垂直拡張には優れているが、水平拡張（特に大規模な分散システムの構築）には限界がある	◎ シンプルな構造がスケールアウト（水平拡張）を容易にし、分散システムでのパフォーマンス向上に有効	◎ 分析処理の高速化と大規模データセットの効率的な管理を目的とした設計が、スケールアウトに適している	○ 柔軟なデータモデルとシャーディング（データの分割と分散）により、一定の水平拡張は実現できるが、リレーショナルやキーバリューほどではない	○ 高度に連結されたデータの複雑な関係性を扱うため、大規模な分散処理においてスケーラビリティの課題を抱えることがある	◎ リレーショナルデータベースのACID特性を維持しつつ、分散システムを通じた水平拡張に優れている
プログラマビリティ（複雑性）	○ SQLの学習が必要だが、広く使われているため資料が豊富	○ シンプルで直感的なデータアクセスが可能	○ 特定のクエリには最適だが、一般的な用途では複雑性が増す	◎ 柔軟なデータモデルで直感的な開発が可能	△ 特殊なクエリ言語を必要とすることが多く、学習コストが高い	△ リレーショナルと同じくSQLを使用するが、分散環境の管理が加わる
運用負荷（保守性）	◎ 成熟した技術とツール、広いコミュニティ支援	○ シンプルな構造で運用は容易だが、機能制限	○ 運用は比較的容易だが、最適化が必要な場合	○ 柔軟性が高いが、スキーマの変更管理が課題になることがある	△ 特殊な構造とクエリのため、運用と最適化に専門知識が必要	△ リレーショナルデータベースの運用経験が活かせるが、分散システムの複雑さが加わる

目次

AGENDA

1-b. ブロックチェーン技術の特徴：分散台帳のデータ構造

分散型台帳のデータ構造としてブロックチェーンを紹介する。



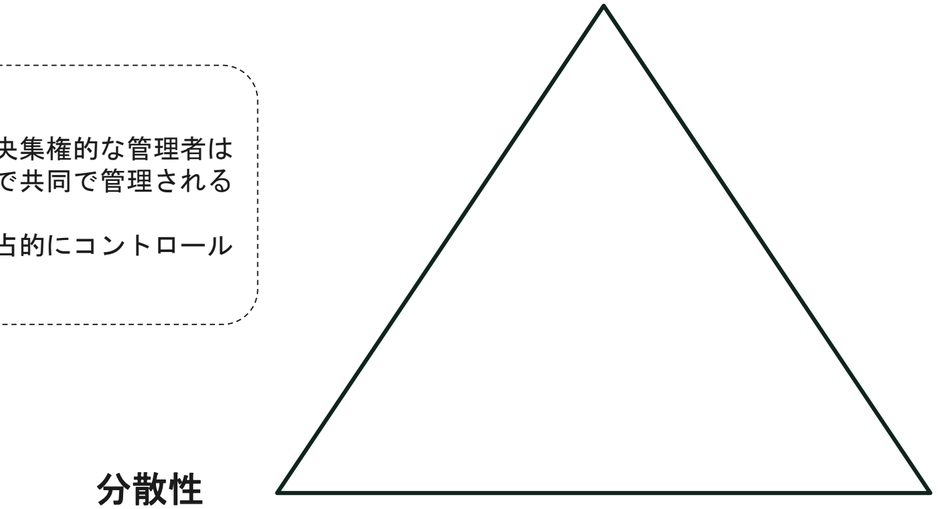
https://qiita.com/abmushi/items/82824f26398af76c35f7

- トランザクションは指定されたブロック容量に応じてブロックに格納される
- 各ブロックは時系列順に直列に繋がっていく

1. ブロックチェーン技術の特徴：ブロックチェーンのトリレンマ

パブリックブロックチェーンにおける重要な要素として、スケーラビリティ・セキュリティ・分散性の3つが挙げられる。

この3要素を同時に満たすことは難しいとされており、この問題は「ブロックチェーンのトリレンマ」と呼ばれている。



分散性（Decentralization）
・パブリックブロックチェーンでは中央集権的な管理者は存在せず、ネットワークは参加者全員で共同で管理されることが望ましい
・分散性が高いほどネットワークを独占的にコントロールすることが難しくなる

スケーラビリティ（Scalability）
・パブリックブロックチェーンではネットワーク参加者が増加すると、トランザクション手数料の高騰や取引遅延が発生する場合がある
・スケーラビリティが高いほど、多くの参加者の取引を迅速に処理することができ、ユーザーエクスペリエンスを損なわない

セキュリティ（Security）
・パブリックブロックチェーンは悪意ある攻撃者から攻撃を受ける場合がある
・セキュリティが高いほど、ハッキング、改ざん、詐欺などに対し堅牢な仕組みを持つといえる

1-b. ブロックチェーン技術の特徴：パブリックブロックチェーンのアプローチ

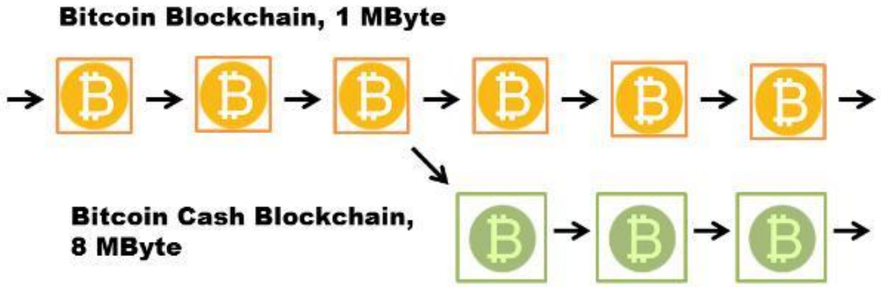
パブリックブロックチェーンにおける機能改善に向けたアプローチをプロジェクト別に紹介する。

	時価総額	仕様	特徴
Bitcoin（BTC）	約152兆円（1位）	コンセンサスアルゴリズム：PoW（Proof of Work） ブロック生成間隔：約10分 TPS（Transaction per second）：約5	● 最も分散的で堅牢なネットワークを有する暗号資産 ● ブロックサイズ：1MB
Bitcoin Cash（BCH）	約7,900億円（20位）	コンセンサスアルゴリズム：PoW（Proof of Work） ブロック生成間隔：約10分 TPS（Transaction per second）：約120	● Bitcoinから派生した暗号資産 ● ブロックサイズ：**32MB** ● **ブロックサイズを拡張**することで、処理可能なトランザクション量を増加させ、スケーラビリティを向上させている ● ノード運用に対する技術的要求がBitcoinと比べて高く、ノード数減少による**分散性低下の懸念**がある
Ethereum（ETH）	約55兆円（2位）	コンセンサスアルゴリズム：PoS（Proof of Stake） ブロック生成間隔：約12秒 TPS（Transaction per second）：約10-15	● スマートコントラクトと分散アプリケーション（DApps）のためのプラットフォーム ● レイヤー2のソリューションとシャーディングの導入によりスケーラビリティ向上を目指している
Avalanche（AVAX）	約2兆円（9位）	コンセンサスアルゴリズム：PoS（Proof of Stake） ブロック生成間隔：約2秒 TPS（Transaction per second）：約70	● 高速でスケーラブルなトランザクションを実現する分散アプリケーション（DApps）のためのプラットフォーム ● PoSを基盤とする**統計学的なコンセンサスアルゴリズム**を採用することで、高い分散性を保ちながら取引の高速化を実現している
Solana（SOL）	約7兆円（5位）	コンセンサスアルゴリズム：PoS（Proof of Stake） ブロック生成間隔：約400ミリ秒 TPS（Transaction per second）：約3000	● 高性能ブロックチェーンとして設計されたブロックチェーン ● PoSを基盤とする独自のコンセンサスアルゴリズムを採用することで、**レイヤー1の基盤で**高速なトランザクションを処理することを目指している
Sui（SUI）	約500億円（59位）	コンセンサスアルゴリズム：PoS（Proof of Stake） ブロック生成間隔：約50-100ミリ秒 TPS（Transaction per second）：約100	● 元Libra（Diem）の元メンバーが開発するレイヤー1ブロックチェーン ● Move言語を使いNFTやトークンをオブジェクトとして扱う事を特徴とする

※ Suiは部分的にDAG（有向非巡回グラフ）の技術を採用しているが、メインネットに対して採用されている訳ではない
メインネットにDAGを採用しているプロジェクトは現時点で少なく、マーケットで十分な信用を得る状況には至っていないと考えている

1-b. ブロックチェーン技術の特徴：ビットコインキャッシュの事例

ビットコインキャッシュはビットコインのブロックチェーンがハードフォーク（分岐）することによって誕生した暗号資産である。



https://www.cryptoambit.com/blog/2017/12/20/what-the-fork

ビットコインキャッシュのブロックサイズは、ハードフォーク時点で
8MBに拡張され、その後32MBまで拡張されている

メリット

- 処理速度の向上
  - ブロックに取り込み可能なトランザクション数が増加することで、
    トランザクションの処理速度が向上する
- トランザクション手数料の低下
  - ブロックに格納可能なトランザクションは有限であるため、ブロックサイズが小さい場合、手数料競争が発生することがある
  - ブロックサイズが増加に伴い、手数料競争は抑制され、ユーザーは低い手数料で送金することができる
- スケーラビリティの向上

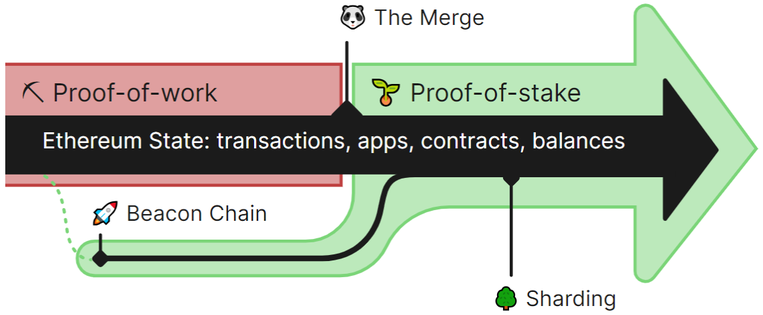
デメリット

- 分散性低下の懸念
  - ブロックサイズ増加に伴い、フルノードの運営に必要なストレージも増加しノード運営のコストが高まる
  - 資金力の少ないノード運営者（主に個人）は撤退を余儀なくされ、その結果、分散性が損なわれる可能性がある

1-b. ブロックチェーン技術の特徴：イーサリアムの事例

イーサリアムでは様々なアプローチで機能改善が測られている。

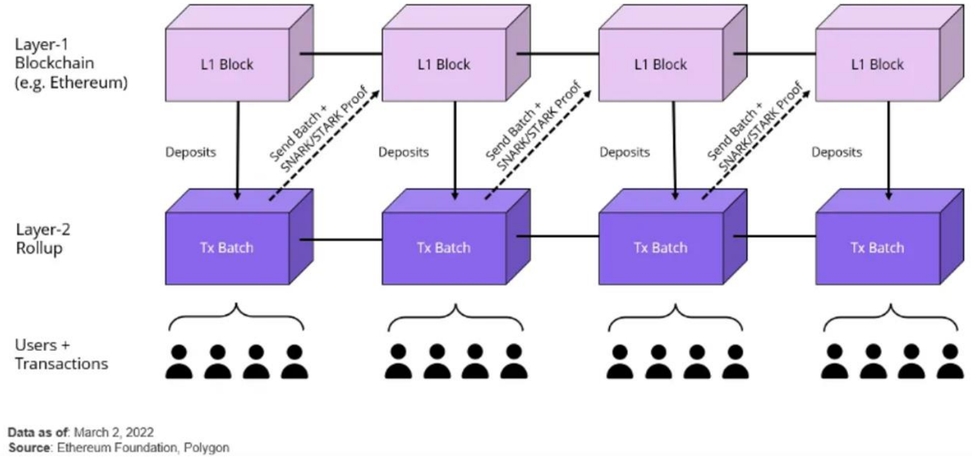
①コンセンサスアルゴリズムの変更



https://ethereum.org/roadmap/merge

- 2022年のアップデート「The Merge」によって、コンセンサスアルゴリズムがPoW（Proof of Work）からPoS（Proof of Stake）に変更された
  - エネルギー消費量が大幅に減少した
  - 分散性の改善（ASIC販売会社への依存が無くなった）

②レイヤー2との統合



https://messari.io/report/polygon-a-multi-sided-approach-to-zk-scaling

- メインチェーン（レイヤー1）の取引の一部をオフチェーンで処理する
- 総体として処理速度が向上し、スケーラビリティを高めることに繋がる

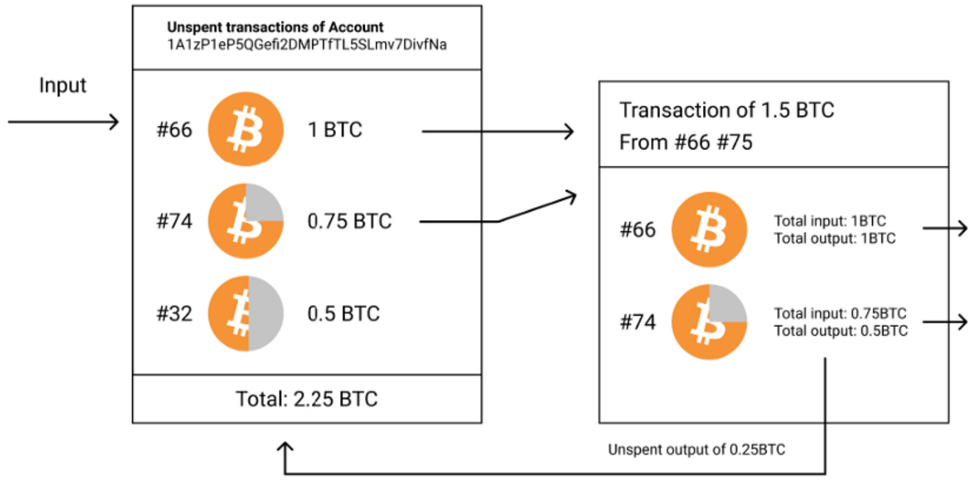
1-b. ブロックチェーン技術の特徴：UTXO型

ビットコインも採用しているアカウントモデル

UTXOとは、「未使用のトランザクションのアウトプット（Unspent Transaction Outputs）」のリストを合算することで「残高」を管理するブロックチェーンのアカウントモデルの一種であり、ビットコインをはじめとするブロックチェーンで採用されている。

（厳密には特定のアドレス宛に送られたUTXOの総和を、アプリケーション側で算出することによって、残高が表示されている。）

UTXOは紙幣の様に分割して消費することができず、「おつり」を自分宛ての新たなUTXOとして作成している



ビットコインにおけるトランザクションの仕組み（※1）

メリット

- プライバシー
  - 各トランザクションが新しいoutputを生成し、ユーザーの資産はバラバラのUTXOとして保持されるため、トランザクションの追跡が難しい
- スケーラビリティ
  - UTXOは独立して存在するため、トランザクションの検証を並列で行うことができ、トランザクション処理速度が高い
- セキュリティ
  - UTXOは消費される度に生成されるため、二重支払いなどの不正行為を防ぐことができる

デメリット

- 複雑性
  - アカウントバランスのトラッキングが直感的でなく、開発者やユーザーにとって理解しにくい場合がある
- データサイズ
  - 大量のトランザクションが発生するとUTXOsのサイズが膨大になり、フルノードのストレージ要件が増大する
- 送金手数料
  - トランザクションのサイズが大きくなるほど、処理にかかる手数料が増加する

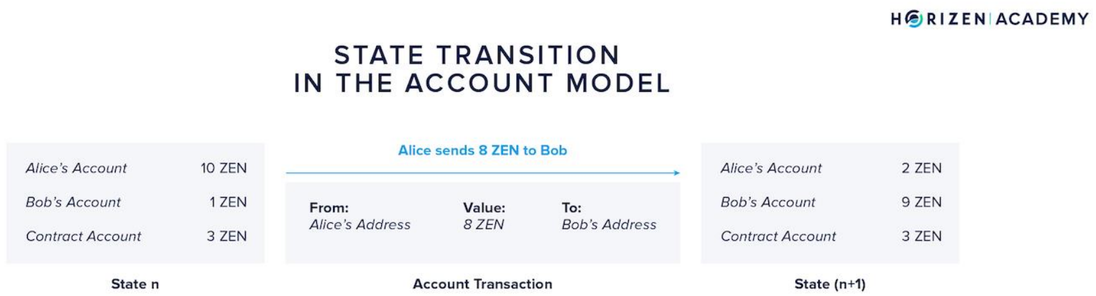
（※1）出典：SafePal Knowledge Base UTXO - What is it and how to use it

1-b. ブロックチェーン技術の特徴：アカウントステート型（Ethereum）

従来の銀行口座の概念をもったアカウントモデル

アカウントステート型は、各ユーザーは銀行口座と同様にアカウントを持ち資産を口座内の残高として表す。

残高は、その時点での保有する暗号資産を表示する。



イーサリアムにおけるトランザクションの仕組み（※1）

メリット

- シンプルな設計
  - 各ユーザーの資産が単一のアカウントバランスとして保持される
- スマートコントラクトの統合
  - スマートコントラクトと連携するために最適化されている
- ガスの計算が容易
  - 「ガス」を使用するため、トランザクションのコストをより明確に把握しやすい

デメリット

- プライバシー保護の課題
  - トランザクションが特定のアカウントにリンクされるため、ユーザーのトランザクションヒストリーが追跡しやすい
- 並列処理の制限
  - アカウントの状態を更新する必要があるため、同時に多数のトランザクションを処理することが難しい
- セキュリティリスク
  - アカウントの状態が変化するため、コントラクト間の相互作用や状態の変化により、セキュリティ上の脆弱性リスクがある

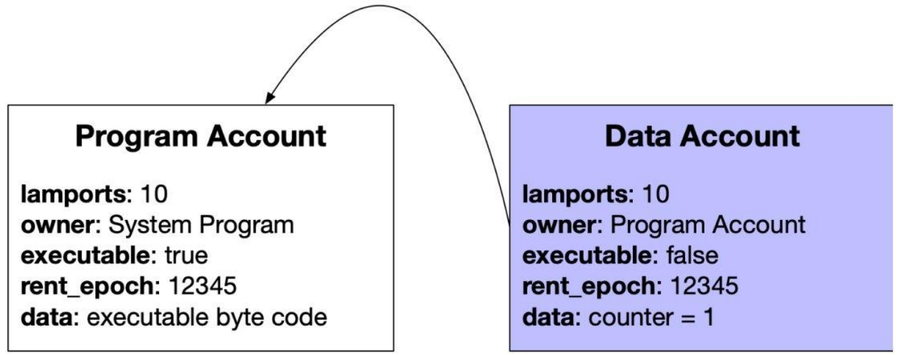
（※1）出典：HORIZON ACADEMY The Account Model

1-b. ブロックチェーン技術の特徴：アカウントステート型（Solana）

各アカウントが独自の状態を持つ、高速なトランザクションを実現するシステム

各アカウントは一意のアドレスに関連付けられており、アカウントにはその残高、所有者、およびスマートコントラクトなどのアカウントに関連するデータやプログラムが保持される。

アカウントは自己完結型であり、トークンの数、プログラム（スマートコントラクト）の状態、アカウントに必要なメタデータなどの状態情報を維持する。



メリット

- シンプルなアカウント管理
  - 各ユーザーやスマートコントラクトが持つ一意のアドレスに資金を結び付けることで、複数の資産の管理をシンプル化する
- スマートコントラクトの効率
  - スマートコントラクトとの相互作用が頻繁に行われる用途のために設計されており、アカウントモデルはステートフルなスマートコントラクトの実行に適している
- 高速なトランザクション処理
  - Proof of History (PoH) と組み合わせることにより、非常に高速なトランザクション処理を可能にする

デメリット

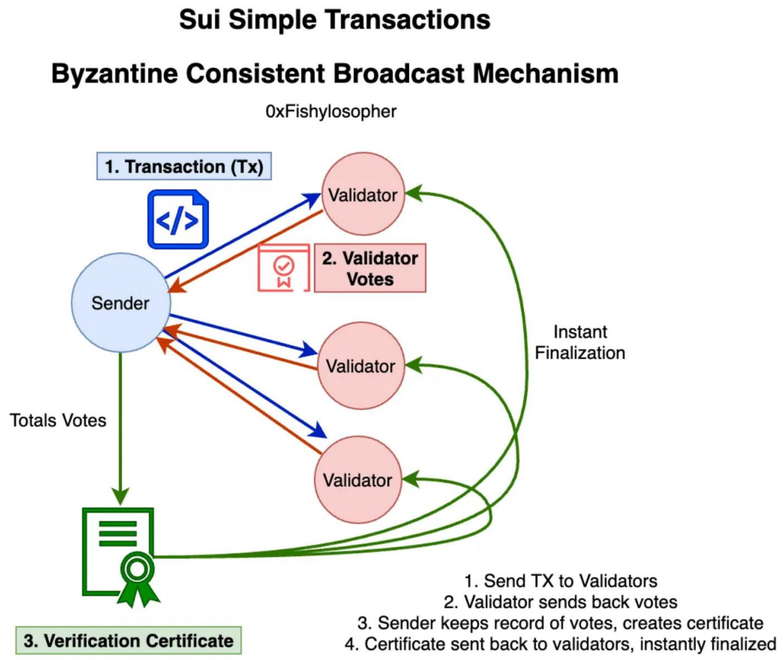
- 状態の複雑性
  - アカウントの状態をブロックチェーン上で頻繁に更新する必要があるため、複合的なトランザクションやスマートコントラクトの操作については状態管理が複雑になる可能性がある
- ネットワークの負荷
  - スループットを保ちながら、同時に発生する可能性のある大量のトランザクションを処理するためには、ネットワークのノードにとって大きな負荷がかかる
- ステートの膨張
  - ステートフルなトランザクションとアカウントの継続的な更新により、時間とともにブロックチェーンの状態が拡大し、データストレージの要件が増大する

（※1）出典：Solana Cookbook

1-b. ブロックチェーン技術の特徴：アカウントステート型（Sui）

オブジェクト指向で表現する革新的な設計

ユーザーの残高は、彼らが所有するオブジェクトの集合として管理されるため、より細やかな所有権の管理と効率的なトランザクションの処理が可能である。アカウントステートモデルのように、アカウントの「状態」を更新するのではなく、所有権が移動することによってオブジェクトの状態が変化する。



メリット

- スケーラビリティ
  - オブジェクト指向モデルは高スループットと低レイテンシーを可能にし、大量のトランザクションを迅速に処理する助けとなる
- 個別のオブジェクト管理
  - 各アセットが独自のオブジェクトとして管理されるため、複雑なスマートコントラクトの実行や、個々のアセットに特化したインタラクションがより容易
- トランザクション処理
  - オブジェクトが別々に処理されるため、複数のトランザクションが同時に進行できる

デメリット

- 新規性による未知数
  - 比較的新しいブロックチェーン技術であり、全体的なセキュリティと安定性の検証には実用化が必要
- ツールとインフラの成熟度
  - SuiやMoveに対応した開発ツール、エコシステム、インフラストラクチャは未だ発展途上にある
- エコシステムのサポート
  - 新しいアプローチが広範な開発者やユーザーコミュニティのサポートを獲得するには、時間を要する可能性がある

（※1）出典：Sui-nnovation

1-b. ブロックチェーン技術の比較

３つのアカウントモデルの比較を行う。

評価基準は、◎：優れている、◯：標準、△：劣っている、とする。（評価基準は、以下の要素を総合的に考慮して主観的に判断）

項目	UTXO型	アカウントステート型		
	ビットコイン	Ethereum	Solana	Sui
セキュリティ	◎ 二重支払い防止 改ざん耐性 透明性	◯ スマートコントラクトの脆弱性	◯ 過去のダウンタイムの発生がセキュリティ評価に影響	◎ 分散性と効率性を両立させるデザインにより将来有望
プライバシー	◎ 資金の流れの追跡が困難	△ 送受信者と取引履歴が公開 （匿名性の低さ）	△ 送受信者と取引履歴が公開されている	△ 送受信者と取引履歴が公開されている
プログラマビリティ	△ スマートコントラクトの利用が限定的 複雑なトランザクション開発が困難	◎ スマートコントラクトによる柔軟な開発 dAppsの開発ツールが充実している	◯ Rustなどの一般的な言語を使用可能	◯ Move言語を使用したオブジェクト指向アプローチがプログラマビリティを向上
ユーザー体験	△ トランザクション処理が複雑で理解が困難	◯ アカウント管理がシンプルで トランザクション処理が簡単	◯ 高速なトランザクションと低コストがユーザー体験を向上 一方、システムのダウンタイムがネガティブな体験	◯ zkLoginを採用することで大衆に受け入れられやすい
（送金の）スケーラビリティ	◯ スケーラビリティが高い	△ ネットワークが混雑すると送金手数料が高くなる	◎ 現時点で提供されている中でトップクラスのスケーラビリティ	◯ 分散型の設計アプローチが高いスケーラビリティ

目次

AGENDA

2. 当社におけるブロックチェーンデータ管理

- 問題
  - ブロックチェーンはデータベースと比較して書き込みコストが高い
    - 顧客間の売買取引を都度ブロックチェーンへ書き込むのは非経済的

- 対応策
  - ブロックチェーンにおいては、複数顧客の暗号資産を混蔵して保管する
    - 顧客資産と自己資産は異なるウォレットで保管する（分別管理）
  - 各顧客の持分は当社のデータベース（RDB）に記録し、直ちに判別できる状態で管理する
    - 顧客間の取引はデータベースへの書き込みだけで完結し、ブロックチェーンに書き込む必要がない

- 課題
  - 暗号資産の入出金が行われた際は、ブロックチェーンとデータベースの整合性をとらなければならない
    - ブロックチェーン上の入金データは正当なものしかデータベースに反映してはならない
    - データベース上の出金データはブロックチェーンに二重で書き込むことがあってはならない

2. 当社におけるブロックチェーンデータ管理：出金

- UTXO型ブロックチェーンにおけるUTXOは一度しか支払いに使用できないため二重支払いの危険性は低いが、複数顧客による出金申請を連続で処理するには十分なUTXOを確保しておく必要がある
  - お釣りのUTXOをあえて複数に分割しUTXOを確保している
  - 少額のUTXOが多すぎても送金データが大きくなりブロック生成者に支払うコストが嵩むので調整が必要

- アカウント型ブロックチェーンの場合、送金データにnonceを含めることによって二重支払いを防ぐ
  - nonceは一意かつ増加させていかなければならない
    - レースコンディションにより複数の出金のnonceが被るといずれかの出金が処理されない
    - プログラムの再実行等により誤って一つの出金申請に対しnonceが異なる複数の送金データを作成してしまうと、二重支払いが発生する

2. 当社におけるブロックチェーンデータ管理：入金

- UTXO型ブロックチェーンの送金データには成功・失敗のステータスがないため、ブロックに含まれておりブロックが覆らないと判定できればデータベースに反映して良い
  - ブロックが覆らない状態 = ファイナリティがある
  - PoWのファイナリティは確率的なので、ヒューリスティックに判定することになる
    - 51%攻撃の可能性を検知した場合は入金の反映を停止する必要がある

- アカウント型ブロックチェーンの送金データには成功・失敗のステータスがあることが多く、ファイナリティの判定に加え正当性の反映も必要
  - アカウント型ブロックチェーンの多くはスマートコントラクト等を利用した複雑な条件での支払いを可能にしており、条件が成立せず送金が失敗していることもある
  - データベースへの反映条件を細かく設定する必要がある

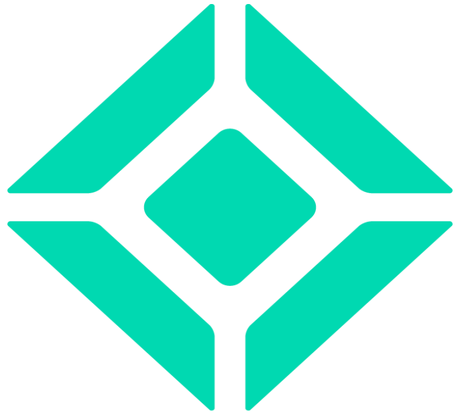
目次

AGENDA

3. ディスカッションテーマ

論点

1. CBDCに関するあらゆるデータを共通の台帳に書き込むことは現実的か
2. 事業者（CBDCと接続する仲介業者）がブロックチェーンとデータベースを併用しデータ管理を行うことを前提とすると、CBDCに求められる要件は何か
3. NoSQL、NewSQLの活用可能性はあるか



Coincheck

新しい価値交換を、もっと身近に