中央銀行デジタル通貨に関する実証実験

「概念実証フェーズ 2」結果報告書

日本銀行決済機構局

2023 年 4 月

目次

1 目的

日本銀行決済機構局は、2021 年 4 月から 2022 年 3 月までの間、「概念実証フェーズ 1」を実施し、中央銀行デジタル通貨（Central Bank Digital Currency, CBDC）システムの基盤となる「CBDC 台帳」に関して、いくつかの設計パターンを用いて実験環境を構築し、CBDC の基本機能を的確に実行することができるかを検証した[1]。

フェーズ 1 に続き、2022 年 4 月から 2023 年 3 月に実施した「概念実証フェーズ 2」では、フェーズ 1 で検証した CBDC 台帳を中心とする基本機能に、技術的な課題を早めに確認しておくことが望ましいと考えられるいくつかの周辺機能を付加した上で、処理性能や技術的な実現可能性を検証した。この他、データモデルやデータベースに関して、フェーズ 1 では扱わなかった新たな技術の活用可能性を検討した。

わが国で CBDC を導入するかどうかは、現時点では決定しておらず、今後の国民的な議論の中で決まってくる。日本銀行としては、こうした議論に資する観点からも、今後の様々な環境変化に的確に対応できるよう、引き続き技術的な実証実験を含め準備しておくことが重要だと考えている。

[1] 「概念実証フェーズ 1」の結果は、以下の文献を参照。日本銀行決済機構局、「中央銀行デジタル通貨に関する実証実験『概念実証フェーズ 1』結果報告書」、2022 年 4 月。

2 検証スコープ

フェーズ 2 では、CBDC の周辺機能を 3 つのブロックに分けて検証した。これらの機能は、仮に社会実装する場合に採用することを前提としている訳ではないが、今後の判断に資するよう、技術的な課題を早めに確認しておくことが望ましいと考えられる（図表 1 参照）。

図表 1 フェーズ 2 で検証した周辺機能

経済的な設計 （**金融システムの安定確保のためのセーフガード**）	● 保有額に対する制限 ● 取引額・取引回数に対する制限 ● 上限超過時やユーザ属性に応じたスウィング機能（上限超過分や送金受領分を銀行預金等に自動受入）の適用 ● 保有額に対する利息の適用
決済の利便性向上	● ユーザによる送金指図の予約 ● ユーザの依頼による一括送金、逆引送金
仲介機関間・外部システムとの連携	● １ユーザへの複数口座の提供 ● 上記を前提とした場合の、保有額、取引額・取引回数に対するユーザ単位での制限 ● 外部システムとの接続方法

すなわち、経済的な設計の周辺機能は、銀行預金から CBDC への急激なシフトを招かないようにするための、各種セーフガードについて取り上げた。決済の利便性向上のための周辺機能は、ユーザの利便性向上に繋がるようないくつかの機能を取り上げた。仲介機関間の連携や CBDC システムと外部システムとの連携のための機能は、1 ユーザへの複数口座の提供や、既存の外部システムとの連携を想定した場合に必要となる接続方法について検証を行った。

この他、フェーズ 1 では検証しなかった新しい技術として、変動額面方式トークンや NoSQL データベースの活用可能性についても検討を行った（図表 2 参照）。

図表 2 フェーズ 2 で検討した新たな技術

変動額面方式のトークン型台帳	● フェーズ1では、価値（額面）が固定され必要に応じ両替を行う固定額面方式を検証 ● フェーズ2では、価値（額面）が変動し必要に応じトークンの統合・分割を行う変動額面方式を検討
NoSQL データベース	● フェーズ1以降、伝統的なリレーショナルデータベース（RDB）を利用 ● フェーズ2では、RDB以外のデータベースとして普及が進むNoSQL（Not only SQL）データベースの活用余地も検討

トークン型のデータモデルについては確定した定義はないものの、概念実証においては、価値（額面）が付された金額データに固有の識別子（ID）を付与し、この ID とユーザ ID の紐づけにより CBDC の保有状況を認識できるデータモデルをトークン型としている。フェーズ 1 では、現金のように額面が固定されており、送金時に必要に応じて小額トークンへの両替を行った上で、既存トークンの紐づけ対象となるユーザを変更していくトークン型（固定額面方式）

を検証した。同方式の性能傾向については既にフェーズ 1 で確認したことを踏まえ、フェーズ 2 では、固定額面方式とは異なるトークン型として、額面は固定せず、送金時に必要に応じて既存トークンの統合・分割を行い、それにより新たに生成されたトークンとユーザを紐づけるトークン型（変動額面方式）について検討した[2]。

また、実験環境用システムのデータベースには、フェーズ 1 以降、伝統的なリレーショナルデータベース（Relational Database, RDB）を採用してきたが、CBDC システムには極めて高い処理性能が求められるため、フェーズ 2 では、処理速度や性能拡張性が高いと言われるものもある RDB 以外のデータベース、すなわち NoSQL（Not only SQL）データベースの活用可能性について検討した。

なお、フェーズ 2 で検証対象とした台帳の設計パターンは、台帳管理のシステムアーキテクチャに関して「中央銀行による中央管理か、中央銀行と仲介機関での分担管理か」、データモデルに関して「口座型か、トークン型か」の組合せの 4 種類とした。フェーズ 1 ではパターン 1～3 を検証対象とし、フェーズ 2 ではそれらに加えパターン 4 も検証対象とした（図表 3 参照）。 なお、パターン 3 については、フェーズ 1 では固定額面方式トークンを検証していたが、フェーズ 2 では変動額面方式トークンを検証した。

図表 3 フェーズ 2 で検証した台帳設計パターン

	中央管理	分担管理
口座型	パターン1	パターン2
トークン型	パターン3	パターン4

＝フェーズ1検証対象
＋ ＝フェーズ2検証対象

[2] トークン型台帳システムにおける固定額面方式と変動額面方式の違いの詳細は、「中央銀行デジタル通貨に関する実証実験『概念実証フェーズ 1』結果報告書」の補論 1 を参照。

3 検証結果

以下では、周辺機能に関する検証結果を 3.1 で、その他の新たな技術に関する検討結果を 3.2 で述べる。

3.1 周辺機能

3.1.1 経済的な設計/利便性向上のための周辺機能

経済的な設計のための周辺機能として、具体的には以下の機能を検証した。

①保有額に対する上限値の設定（保有額制限）

②都度ないし一定期間内の取引額や取引回数に対する上限値の設定（取引額・回数制限）

③保有額上限超過分を銀行預金等の民間マネーへ自動変換したり、法人・個人などのユーザ属性に応じて、送金受領分を銀行預金等の民間マネーへ自動変換したりする機能（スウィング）

④保有額に対する利息の適用（利息適用）

利便性向上のための周辺機能として、具体的には以下の機能を検証した。

①予約送金

②複数送金の一括実行（一括送金）

③受領側ユーザの依頼が起点となる送金の実行（逆引送金）[3]

実験環境用システム

実機検証は、パブリッククラウド上に実験環境用システムを構築した上で、時間的制約のなかで効率的に検証を行えるよう、最も構築が容易なパターン 1 を用いて行った。同じ口座型データモデルであるパターン 2 に関しては、パターン 1 の測定結果を用いて机上検証を行った[4]。

周辺機能の実装にあたっては、複数のシステムを構築し、API（Application Programming Interface）により、システム間連携を行う方式を採用した。具体的には、CBDC システム内にはフェーズ 1 で構築した台帳システムに加え、取引額・回数制限にて参照する取引履歴管理システムや、ユーザが予約した送金指図を自動起動する予約管理システムを追加した。また、取

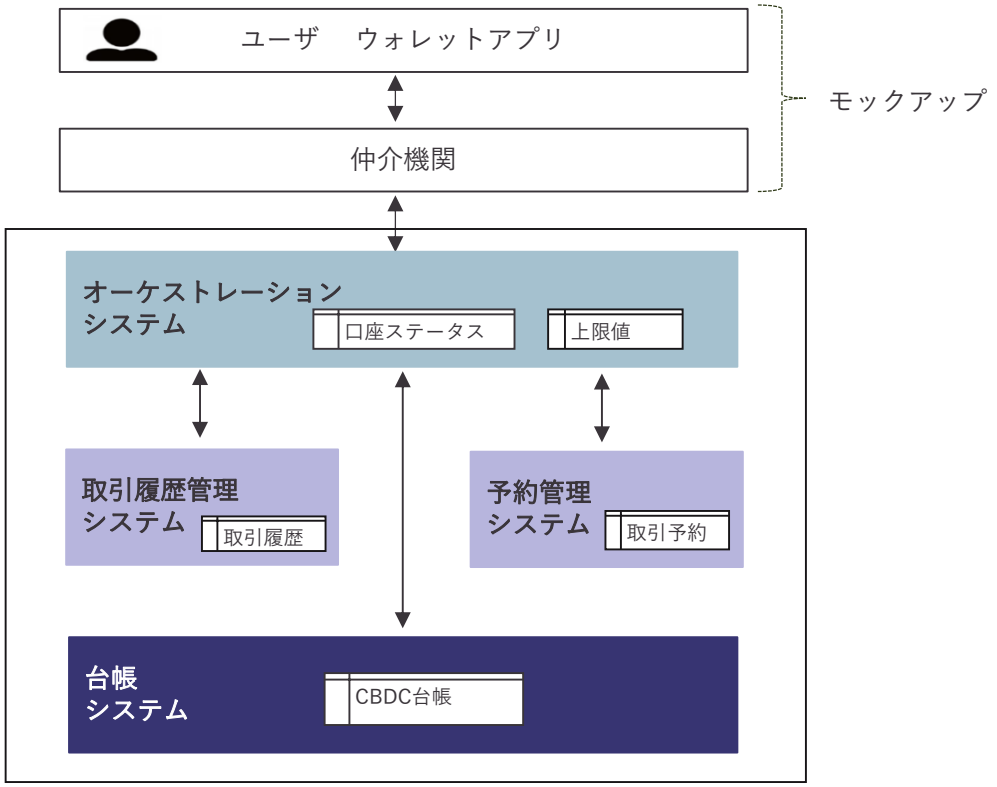
[3] 既存の逆引送金の具体例は、銀行口座における口座振替などにみられる。

[4] データモデルが変動額面方式トークン型のパターン 3 と 4 に関しては、基本機能から検証を行う必要があるため、3.2 にて、別枠で検証結果を記載した。

引指図を受ける窓口機能を持ち、システム間でのデータ整合性を確保するため、各システムにおける口座の処理状況を管理し、ある取引実行中には他取引の処理が原則実行されないよう排他制御を行うオーケストレーションシステム[5]を追加した。オーケストレーションシステムには、各種制限の上限値を管理するデータベースを持たせることで、上限値に抵触したか否かを判定する処理も実装した。なお、台帳システム上に取引履歴管理などの機能を直接持たせる構成[6]としなかったのは、社会実装する場合、高い更新負荷が台帳システムに生じる可能性を踏まえ、台帳システムには残高更新以外の処理を可能な限り行わせない方が適当と考えたためである。

仲介機関システムやユーザが利用するウォレットアプリについては、フェーズ 1 同様、取引指図を投入するだけの簡易な設計（モックアップ）とし、性能評価の対象は、モックアップから取引指図を受けた後のシステム処理の開始から完了までとした（図表 4 参照、使用したサーバのスペック等の詳細は補論 2 を参照）。

図表 4 パターン 1 の実験環境用システム



[5] ある取引の処理においては、原則として同じ口座を対象とする他の処理を待機させた上で、取引額・回数制限判定、保有額制限判定、残高更新、取引履歴更新の順で処理が実行されるよう制御を行う。ただし、性能向上のため、各種制限処理の整合性が確保可能な際は他の処理を進める仕組み（例えば払出と送金のように、一方が増額で他方が減額となる業務の組合せは、片方の処理を待機させることなく処理を進めることが可能）を採用した。

[6] こうしたオルタナティブなシステム構成については、補論 1 を参照。

周辺機能の実装方法

上記システム構成のもと、各周辺機能は以下のとおり実装した。

各種制限

オーケストレーションシステムが、他のシステムが管理する判定に必要な情報（台帳システムが管理する CBDC 残高、取引履歴管理システムが管理する取引履歴）を参照しながら実施する。判定処理後は、フェーズ 1 で実装した基本機能と同じ送金処理が台帳上で行われ、台帳の更新後、取引履歴が更新される。

スウィング

オーケストレーションシステムにおいて、保有額上限超過分の計算や法人・個人などのユーザ属性の識別を行った上で、対象分を銀行預金等に変換する自動受入の指図を、台帳システムおよびモックアップである仲介機関に送信する。

利息適用

利息の受払いは、取引額・回数制限の対象外とした。したがって、取引額・回数制限への抵触を判定せず利息の受払いを実行する設定とした上で、後述の予約送金の仕組みを用いて実装した[7]。

予約送金

予約管理システムには、将来実行される予定の送金指図が事前に登録されている前提とし、指定時刻に予約管理システムからオーケストレーションシステムに対して送金指図を 1 件ずつ送り[8]、その後は通常の送金と同じ処理内容となる実装とした。

一括送金

オーケストレーションシステムで、複数一括の送金指図に対し纏めて上限値を取得した上で、1 件ずつ分割し、その後は通常の送金と同じ処理内容となる実装とした。

逆引送金

送金側ユーザによる事前の同意を前提として、受領側ユーザが送金指図を実行し、その後は通常の送金と同じ処理内容となる実装とした。

性能テスト

周辺機能の追加がシステムの性能に与える影響を検証するため、性能テストの基本的な設定はフェーズ 1 と同じとした。すなわち、ユーザ数は 10 万人、仲介機関は 5 先とし、基本機能の取引指図の負荷量は通常負荷シナリオとして秒間 500 件、高負荷シナリオとして秒間 3,000 件とした。

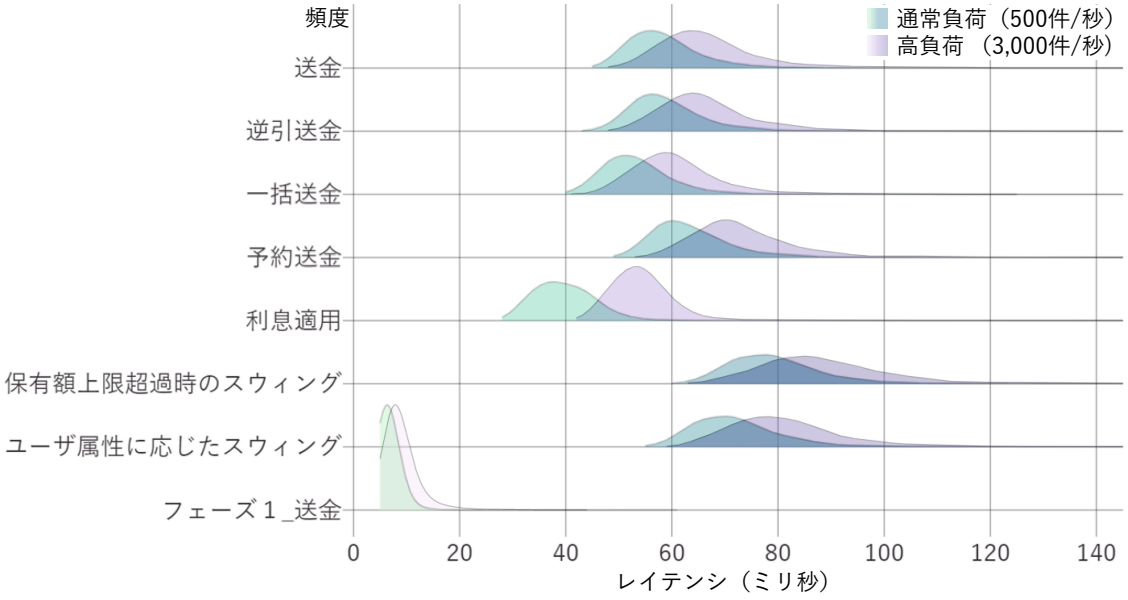
---

[7] 利息適用の実装にあたっては、各口座に適用される利息額は一定期間内における各口座の CBDC 残高と所定の利率に基づいて計算されたものとの前提を置く。その上で、中央銀行の CBDC 口座を便宜的に設け、その口座とユーザの口座との間で利息額の CBDC の受払いを行う（システム上は口座間の送金と同様に扱う）方式をとった。この他にも、中央銀行の口座は設けず、ユーザ口座の CBDC 残高を直接に増減させる（システム上は各口座への CBDC の発行や還収と同様に扱う）方式なども考えられる。

[8] 複数送金を実行する予約送金や一括送金では、他の業務処理への影響を極力抑えられるよう、バッチによって一括処理する方式はとらず、1 件ずつ分割し処理していく方式をとった。

テストの結果、スループット（1 秒間あたりの処理件数）は、秒間のリクエスト投入件数と同じ件数を達成し、リソースを含め性能上のボトルネックは生じなかったことが確認できた。レイテンシ（1 件の処理時間）は、フェーズ 1 対比で処理内容や経由するシステム数が増加したことを背景に、幾分増加し、分布の裾野もやや拡大したものの、高負荷時であっても概ね 200 ミリ秒以下、すなわち 0.2 秒以下となっており、大きな性能劣化とはなっていない（図表 5 参照、測定方法や結果詳細は補論 2 参照）。

図表 5 業務別レイテンシの結果[9]



上記レイテンシの結果を仔細にみると、以下のとおり。

送金

レイテンシは 60 ミリ秒程度と、フェーズ 1 の送金対比、周辺機能追加による処理内容やシステム数の増加を背景に幾分増加したが、大きな性能劣化とはならなかった。

逆引送金

指図の起点が逆（受領側）であるものの、処理内容は送金と同じであるため、レイテンシは送金とほぼ同じとなることが確認された。

一括送金

テスト期間内に測定値を複数セット取得できるよう、600 件を一括とした送金指図に対し、保有額や取引額・回数の上限値を纏めて取得した上で、1 件ずつの送金指図に分割して処理をしていく実装をとった。このため、上記表上の分割後 1 件あたりのレイテンシは、上限値を纏めて取得した分、通常送金より数ミリ秒速くなることが確認された。

[9] レイテンシは、1 業務の取引指図に関するアプリケーション処理の開始から終了までの時間を計測。各周辺機能を付加した場合のレイテンシの大まかな傾向を可視化するため、図表の縦軸は、レイテンシの頻度、カーネル確率密度推定量を用いた（Silverman の検定結果を参考にバンド幅を設定して算出）。

予約送金

指定時刻になったら 1 件ずつ予約管理システムからオーケストレーションシステムに指図が送信され、その後は通常送金と同じ処理が実行されるため、予約管理システムで処理が起動される分、通常送金よりレイテンシが数ミリ秒遅くなることが確認された。

利息適用

予約送金の仕組みを用いた実装となっているが、取引額・回数制限の対象外とした分、予約送金よりもレイテンシが幾分速くなっていることが確認できた。

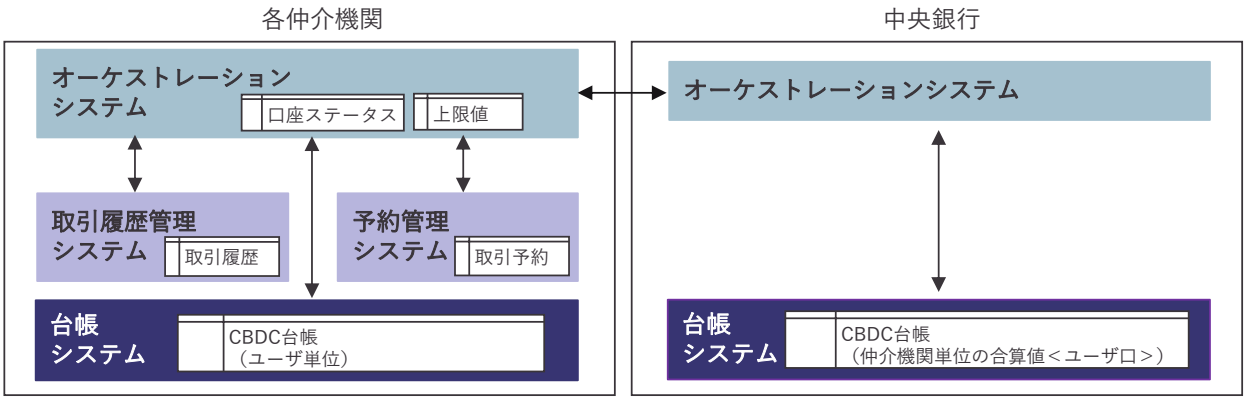
スウィング

銀行預金等への受入業務が追加された分、通常送金よりもレイテンシが幾分増加することが確認できた[10]。

パターン 2 の性能予測

パターン 2 の場合、ユーザの口座管理は各仲介機関の台帳で行うデザインであることから、各種上限判定処理に必要なユーザの上限値や取引履歴の管理も、各仲介機関が行うシステム構成が考えられる[11]（図表 6 参照）。この場合、パターン 1 の中央銀行システムで行っていた各種上限判定処理は、パターン 2 においては各仲介機関のシステム内で行うことが想定される[12]。

図表 6 パターン 2 のシステム構成例



[10] なお、ユーザ属性に応じたスウィングのレイテンシは、上限超過分の計算処理がない分、ごく僅かながら保有額上限超過時のスウィングより速くなっていることが確認できた。

[11] この場合、中央銀行のオーケストレーションシステムには、システム間の接続インターフェースとしての窓口機能のみが残る。また、中央銀行の台帳システムでは、フェーズ 1 同様、主に仲介機関を跨ぐユーザ間の取引によって台帳上の仲介機関のユーザ口（ユーザの CBDC 残高合算値）が変動する場合に更新処理が発生する（同一仲介機関内の送金の場合には、台帳上の仲介機関のユーザ口が変動しないため、更新処理が発生しない）。

[12] 異なる仲介機関間の送金の場合には、送金元/送金先の各仲介機関がそれぞれで取引額・回数上限判定処理を行い、送金先の仲介機関でのみ保有額上限判定処理を行う。

パターン 2 の異なる仲介機関間の送金に関するレイテンシを、性能テストで得られた測定結果を用い、上記システム構成を前提に一定の仮定のもと机上計算[13]すると、パターン 1 対比、数十ミリ秒程度増加したが、大きな性能劣化とはならなかった。数十ミリ秒程度の増加は、パターン 1 の場合には台帳更新が中央銀行台帳における減額・増額のみで処理が完了していたが、パターン 2 の場合には送金元の仲介機関と送金先の仲介機関のそれぞれの台帳の更新が追加されることが影響している。

パターン 2 の中央銀行の台帳では、仲介機関単位でのユーザの CBDC 残高合算値が、ユーザ口として記録されるため、フェーズ 1 で指摘したとおり、仲介機関を跨ぐ移転の負荷量が増えた場合、更新処理が集中し、レコードロックの影響に伴う性能劣化が生じる可能性がある。こうした性能劣化は、仲介機関のユーザ口のレコード分割や、中央銀行台帳上の口座振替と仲介機関台帳上の口座振替を別々のタイミングで行うようにする[14]ことなどで、回避・緩和できると考えられる。

3.1.2 仲介機関間・外部システムとの連携のための周辺機能

複数仲介機関による CBDC 口座の提供

これまでの概念実証では、1 ユーザは 1 つの仲介機関に 1 つの CBDC 口座のみ保有する前提（単一口座）で進めていたが、ユーザの利便性向上の観点から、1 ユーザが複数の仲介機関にそれぞれ 1 つの CBDC 口座を保有できる仕組み（複数口座）も考えられる。こうした 1 ユーザへの複数仲介機関による CBDC 口座の提供を前提とした場合、3.1.1 で扱ったような各種制限処理をユーザ単位（あるユーザの複数口座分の合算ベース）で実現する必要性が生じる可能性がある。こうした可能性に対応できるようなシステム間の連携方法を検討した。

パターン 1 の場合は、中央銀行におけるシステム内で、ユーザの口座残高や取引履歴が一元管理されることから、基本的にはそれらの情報を管理する粒度を、口座単位からユーザ単位に

---

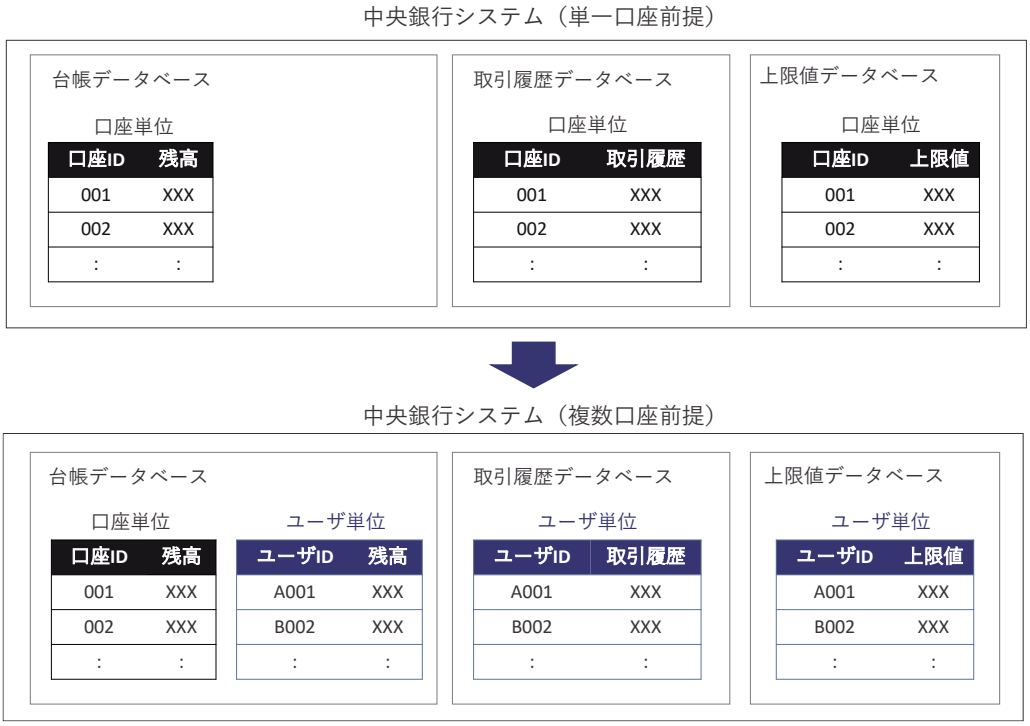
[13] 計算方法や仮定は以下のとおり。

一連の送金業務フローを、取引額・回数上限判定、送金側口座からの CBDC 引落し、受領側口座への CBDC 入金、といった個別処理単位に分解し、各個別処理単位の処理時間の確率分布を、パターン 1 についての性能テストで取得した類似処理の処理時間の確率分布を参照することで推定する。次に、各処理が順序だって行われる部分には、複数の確率変数の和の分布を計算し、複数の処理が並列で行われる部分では、それら確率変数の最大値の分布を計算する。最後に、これらの処理時間の分布を組み合わせることで、業務フロー全体のレイテンシの確率分布を算出する。計算を行う上では、各個別処理の処理時間は互いに独立であり、システム間通信時間は一定であることなどの仮定・前提を置いた。

[14] なお、上記を行うにあたっては、中央銀行台帳と仲介機関台帳が同期処理されることで実現できていた、取引のアトミック性等を維持する工夫を検討する必要がある。

変更することによって、ユーザ単位での各種制限の実装が可能となる（図表 7 参照）。この場合、処理内容は、単一口座を前提とした場合とほぼ変わらなくなる[15]。

図表 7 パターン 1 における情報管理粒度の変更



パターン 2 の場合は、各仲介機関の台帳で、ユーザの各口座の残高や取引履歴は別々に分担管理されるため、ユーザが複数の口座を持つことを前提に、ユーザ単位で各種制限を行おうとすると、上限判定処理を行う仲介機関において、他の仲介機関が保有する当該ユーザの残高情報等の収集が必要となり得る。

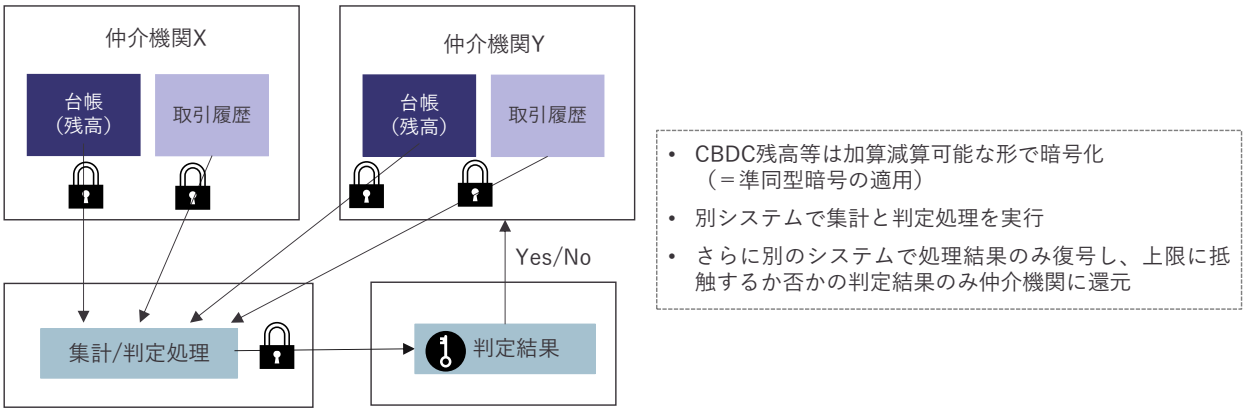
プライバシーの観点からは、各仲介機関が保有するユーザの残高情報等が他の仲介機関に共有されるのは望ましくないという考え方もあり、こうした点も配慮した連携方法の 1 つとしては、上限判定処理に必要な情報収集を行うシステムを別途準備する構成が考えられる[16]。この場合、例えば暗号化した状態でデータの演算処理が可能な準同型暗号を用いることで、情報が秘匿された状態を維持しながら別システムにおいて情報収集と判定処理を可能とするような、

[15] 図表 7 にもあるとおり、台帳データベースのみ、ユーザ単位だけでなく口座単位のデータテーブルも維持し、両者を更新する必要があるため、その分、ごく僅かな処理時間が追加される見込み。

[16] この他、情報収集用の別システムは設けず、マイクロサービスの考え方に基づき、各仲介機関で上限判定処理を完結させるようなシステム構成も考えられる。詳細は補論 1 を参照。

追加的工夫を取り入れることも考えられる[17]（図表 8 参照）。仮にこのような構成をとった場合、シミュレーション結果からは、送金に係るデータの暗号化や復号の処理時間は 50 ミリ秒程度となることが確認された[18]。

図表 8 パターン 2 における準同型暗号活用例



上記から、パターン 2 で複数口座を前提とし、プライバシーに一定の配慮をしつつ、ユーザ単位で各種制限を行うことが、送金レイテンシを大きく劣化させることなく可能であることが示唆された。もっとも、送金時に行う処理が増加することで、障害が発生し得る箇所は増えるほか、潜在的なデータ不整合の発生確率は高くなる。このため、マクロ的な CBDC 流通量をコントロールする方策として、上記のような方法ではなく、シンプルに 1 ユーザの保有口座数に上限を設け、かつ 1 口座あたりの保有額や取引額・回数に上限を設けることで、各口座の残高情報等の合算を不要とする方法なども考えられる。

---

[17] この他、口座残高や取引履歴といった金額情報以外のユーザデータ（ユーザ ID や口座 ID 等）についても、暗号化したままで検索することが求められる場合には、検索可能暗号の利用も考えられるほか、データを分割し複数サーバで演算処理をする MPC（Multi-Party Computation）や、隔離されたハードウェア領域で処理を行う TEE（Trusted Execution Environment）といった情報秘匿技術を活用できる可能性もある。各技術の詳細は、以下の文献を参照。日本銀行決済機構局、「プライバシー保護技術とデジタル社会の決済・金融サービス」、決済システムレポート別冊シリーズ、2022 年 9 月。

[18] 準同型暗号処理を課した場合の処理時間を計算するにあたっては、実用例が比較的多い Paillier 暗号を用い、鍵長にいくつかバリエーションを持たせ、実験環境と同スペックの別環境にて、暗号化、加算、減算、復号を 10,000 回ずつ行うシミュレーションを実施した。その結果、現時点で安全性が高いと言われる 2,048 ビットの鍵長であっても、暗号化・加算・減算・復号あわせて 46 ミリ秒となることが確認された。

なお、準同型暗号には Paillier 暗号の他、ElGamal 暗号や RSA 暗号などがあり、種類によって可能な演算処理や処理スピードに違いがある。Paillier 暗号は、加算と減算のみが可能であるが、相対的に高速な処理が可能なため、社会実装例が多いと言われている。

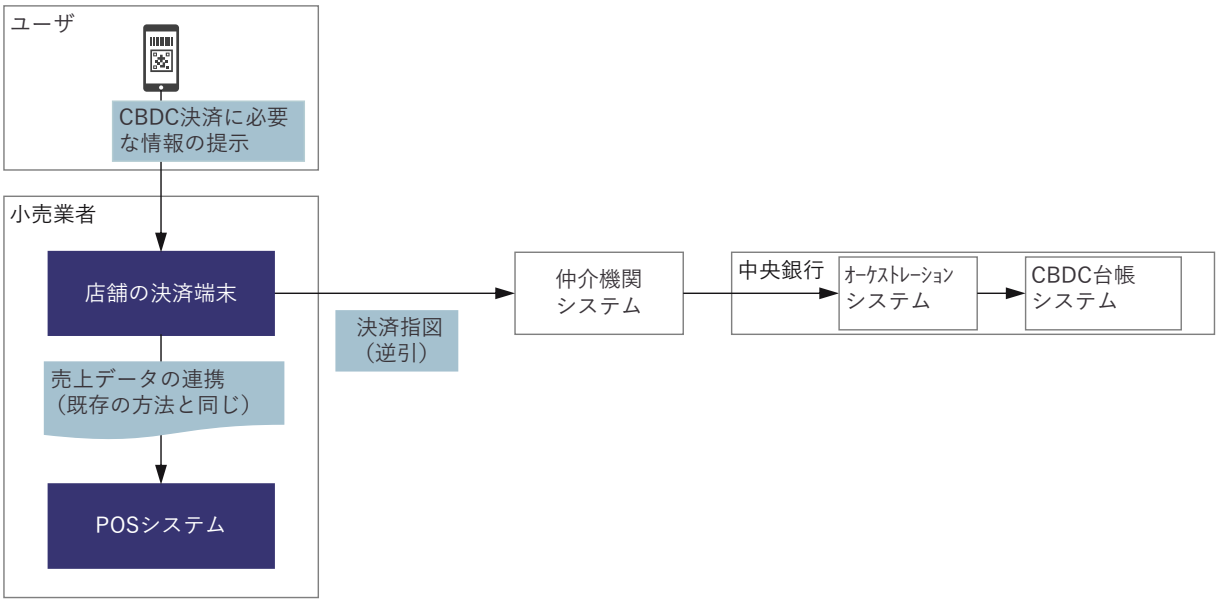
外部システムとの連携

CBDC システムの連携の対象となる可能性のある外部システムとしては、様々なものが考えられる。例えば、店頭決済における POS システムとの連携を考えてみると、店舗の決済端末と POS システムとの間に存在する既存の連携方法を活用できる可能性がある。

現在、百貨店やスーパーマーケット、コンビニエンスストアといった小売業の多くでは、日々の売上高や販売した商品をデータ化して管理する POS システムが導入されており、現金や電子マネー等によって支払いが行われる都度、店舗の決済端末では、受けた支払いの処理を行うとともに、発生した売上に係るデータ（商品や個数等）を POS システムへ自動的に連携する状況がみられる。このような自動的な連携は、将来、小売店などで CBDC を用いた支払いが行われるケースにおいても活用可能と考えられ、その活用のために、店舗の決済端末を、決済情報の秘匿性も確保しながら、CBDC による支払いに対応するよう改修することが想定される。

POS システムとの連携例として、ユーザが店舗で CBDC を用いて支払うとき、店舗の決済端末（CBDC 対応）が CBDC による逆引送金を指図するとともに、POS システムに対して売上データを連携するというケースが想定できる。当該ケースは、下図のように示せる（図表 9 参照）。

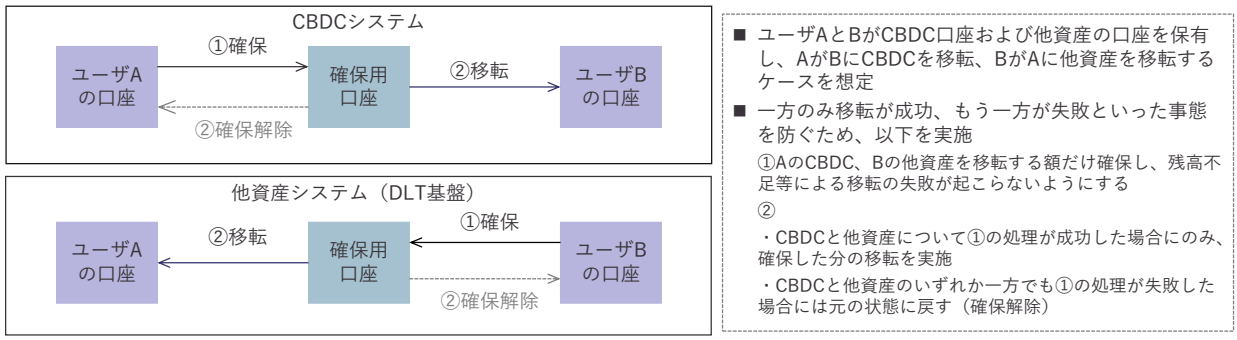
図表 9 店頭決済における POS システムとの連携例



連携する外部のシステムとしては、既存の技術に基づくものだけではなく、例えば、DLT（Distributed Ledger Technology、分散型台帳技術）のような新たな技術を基にしたものとなる可能性も考えられる。諸外国の中央銀行の CBDC の実証実験においては、資産をトークン化して DLT を活用した基盤上で流通させる、いわゆる「アセットトークナイゼーション」を意

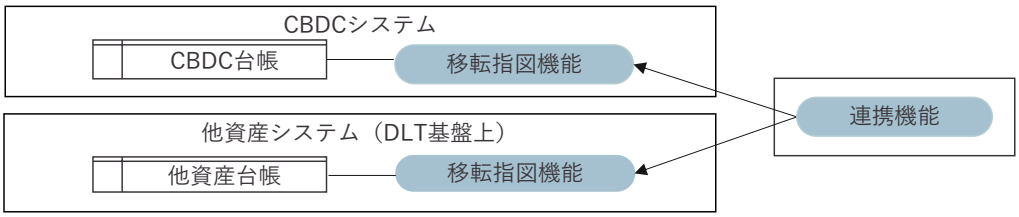
識して、CBDC システムと DLT 基盤との連携を扱った事例も多い[19]。そのため、概念実証では、CBDC と DLT 基盤上の資産の交換を検討するにあたっては、既存の DvP 決済（Delivery versus Payment、証券と資金の同時決済）や PvP 決済（Payment versus Payment、多通貨同時決済）の事例も参考に、交換される資産を各システムで一旦確保し、両システムで確保が成功したら最終的な交換を行うことで、エスクロー（取引保全）を行う処理連動を想定した（図表 10 参照）。

図表 10 処理連動の例



処理連動の実装にあたっては、交換される資産を管理するシステムに、資産の移転指図を出すアプリケーションを持たせ、各システムとは独立した連携用のシステム経由でそれらを間接的に接続する（連携用のシステムが移転指図を適切なタイミングでキックするアプリケーションを持つ）方法が一案となる（図表 11 参照）。なお、連携機能を担う各システムのアプリケーション（図表 11 の移転指図機能と連携機能の部分）については、連携機能を持つシステムに移転指図機能を担わせることもできるほか、DLT 基盤の構成や特性は様々であることから、これ以外の様々な連携方法が考えられる。

図表 11 システム構成の例



こうした外部システムとの相互運用性に関しては、既存のシステムインフラや技術的な連携方法を上手く活用することも選択肢の 1 つとなり得る。また、その性質に応じ、競争領域に属するもの、非競争領域に属するもの、いずれも想定し得ることから、制度設計を含む多様な観点からの検討が重要と考えられる。

[19] DLT を活用した海外中銀の CBDC 実証実験の例は、以下の文献を参照。杉江次郎・鳩貝淳一郎、「分散型台帳技術を活用した決済の改善の取り組み―各国のホールセール型 CBDC の実証実験を中心に―」、日銀レビューシリーズ、2022-J-16、2022 年 11 月。

BOX オフライン決済を検討するに当たってのポイント

これまでの概念実証では、ウォレットアプリや仲介機関・中央銀行の各システム等がネットワークを介してオンラインで常時接続されている状況を前提としてきた。他方、CBDC の将来的なユースケースとしては、一時的であれ、認証や台帳サーバなどへの接続を伴わないオフライン環境において、ユーザ端末間（P2P 接続）で CBDC の移転が行われる決済も想定し得る。こうしたオフライン決済においては、「不正・不整合を検知する」機能が重要となる。想定される不正・不整合としては、例えば以下のような 3 つのケースが考えられ、それぞれ検討すべき課題があることを確認した（本検討の前提として、オフライン CBDC のデータモデルが固定額面方式または変動額面方式のトークンの場合を想定した）。

① オフライン CBDC が不正に生成されるケース

：オフライン CBDC を生成する権限を持たないステークホルダーによって、オフライン CBDC が生成される場合

- 端末上に保蔵されるオフライン CBDC に中央銀行の電子署名を付与しておき、決済時に受領側のユーザ端末で署名の検証を行うことが一案。
- この点について、特に変動額面方式のトークン型のように移転の都度トークンの統合・分割が発生する方式の場合には、ユーザの手元で生成されるトークンの正当性を確認する方法等について、別途検討する必要。

② 異なる端末にオフライン CBDC が複製されるケース

：あるユーザ端末上の正当なオフライン CBDC が、別の端末上に複製される場合

- 例えば複製された CBDC について移転が指図されたときに、これを検知するためにどのような設計が考えられるか検討する必要。

③ 使用済みのオフライン CBDC が端末に残存して二重使用されるケース

：あるユーザ端末上の正当なオフライン CBDC が、オフライン送金後も未使用状態として同じ端末上に残り、別のオフライン送金時に再び利用される場合

- 台帳サーバその他オンライン環境への接続を一切行わない前提のもとでは、受領側のユーザ端末でオフライン決済の都度、二重使用を検知することは技術的に困難。諸外国の実証実験の例においても同様の指摘がある。
- この点について、例えば、各端末を定期的にオンライン環境に接続して必要な情報を収集し突合することで、二重使用がなかったか事後的に確認するなど、別途の方法を検討する余地がある。

なお、本検討の前提として、ここで指摘するような不正・不整合な CBDC がそもそも発生しないよう、端末や端末上のアプリケーションを堅固に構築することでユーザによる不正等を排除する必要がある。もっとも、端末等が堅固に構築されることを前提としても、それらに潜在し得る欠陥や未知の脆弱性の可能性は完全には払拭されない。そのため、不正・不整合な CBDC が仮に発生したとしても、それがオフライン決済に利用されようとしたときに検知できる仕組みについて、検討しておく必要がある。

今後も、想定される状況や前提を整理し、オフライン決済に関する技術的検証を続けることが重要である。また、各施策の検討を進めるにあたっては、技術的検証だけでなくプライバシーの観点からも検討を続ける必要がある。

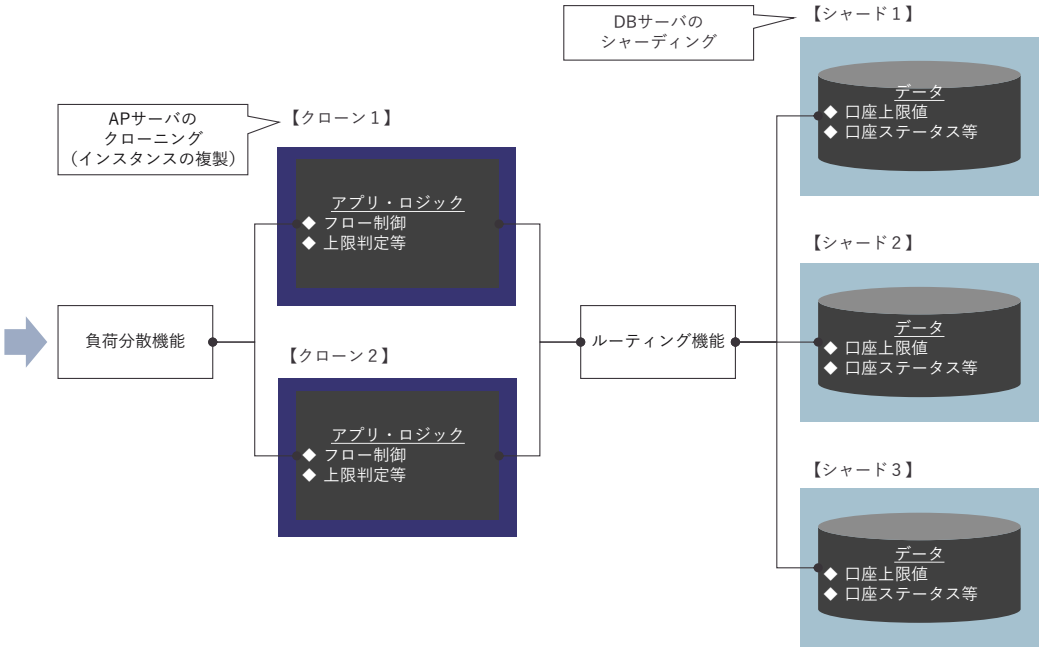
3.1.3 社会実装する場合の課題

拡張性

仮に周辺機能の社会実装をする場合には、フェーズ 1 で想定したとおり、秒間数万～十万件といった大規模高頻度処理への対応や、膨大な口座情報管理を可能とするような性能拡張性が必要となる。

実験環境用システムの性能拡張について、取引指図の処理を実行するアプリケーション（AP）サーバに関しては、同じサーバを複数配置して負荷分散および並列処理を行うクローニング（インスタンスの複製）が考えられる。また、指図の処理結果を記録・保持するデータベース（DB）サーバに関しては、まずはスケールアップが考えられるが、スケールアップはサーバのハードウェアのスペックに限界があるため、より拡張性を考慮すれば、水平分割（シャーディング）[20]により負荷を低減させることが対応の方向性となる（図表 12 参照）。すなわち、処理実行のためのアプリロジックを実装する AP サーバは、指図をサーバごとに独立に処理するように構成できるため、台数を単純に増やすことにより一定の性能拡張が可能となる。一方、膨大な数のレコードを管理する必要がある DB サーバは、水平分割を行うことで各シャードのデータ量やアクセス量を抑制し、これにより処理性能を確保することが考えられる。なお、水平分割を行う際には、複数のサーバ間で整合性のとれた処理を実現する必要があるほか、サーバ内部に保持するデータ量やアクセス量に偏りが出ないよう、定期的なチューニングが必要になるなど、システム運行上の課題がある。

図表 12 性能拡張方法の考え方＜オーケストレーションシステムの例＞



[20] 1 つの DB サーバ上に格納された 1 つのテーブルデータをレコード単位で複数のサーバに分散して保持させる方法。

周辺機能の追加や改修を想定した場合の機能拡張性については、今回の設計ではオーケストレーションシステムに改修の影響が集中するため、当該部分の処理のモジュール化（開発・改修効率向上に繋がる処理の部品化）を検討することが重要である。

信頼性

高い可用性の実現に向けては、障害耐性を高めるため、システムの冗長化や、災害対策のための DR（Disaster Recovery）サイトの準備などが重要となる。パターン 2 の場合には、仮に中央銀行システムに障害が起きても、仲介機関システムだけで完結する処理への影響は限定的となる一方、障害発生箇所の数は多くなる可能性がある。また、複数仲介機関による口座の提供とユーザ単位での各種制限を想定する場合には、残高情報等を一元的に集約するシステムで障害が発生すると、送金元や送金先の仲介機関だけでなく、当該ユーザの口座を保有する他の仲介機関まで影響が広がる可能性がある。このため、前述のように口座数に上限を設け口座毎に各種制限をかけていくことで、残高情報等の合算を不要にする制度面からのアプローチを含め、対応方法を検討していく余地がある。

社会実装する場合に必要なセキュリティ施策は、フェーズ 1 と同様である。すなわち、各サーバやシステム間ネットワークといった CBDC システムの構成要素ごとに施策を講じる必要がある。求めるセキュリティの水準を高めるほど、施策の導入・運用コストが上昇し、処理性能も劣化するというトレードオフの関係に留意しつつ、適切なバランスを検討していくことが重要である。

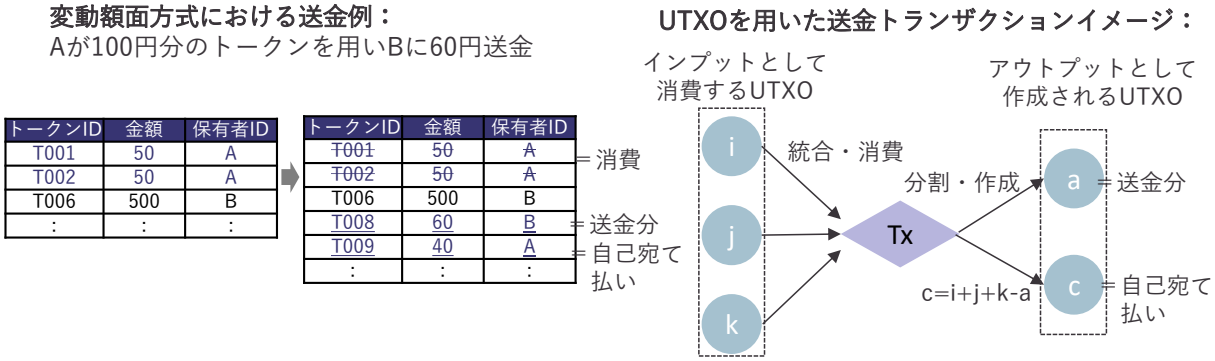
3.2 新たな技術の活用可能性

3.2.1 変動額面方式のトークン型台帳

データモデルとしてのトークン型の定義に確定的なものはないものの、概念実証では、価値（額面）が付された金額データに固有の識別子（ID）を付し、この ID とユーザ ID の紐づけにより CBDC の保有状況を認識するデータモデルと定義し、トークン型の検討を続けている。

台帳設計については前掲図表 3 でも示したとおり、フェーズ 1 では、現金のように額面を固定し、送金時に必要に応じて小額トークンに両替を行った上で、既存のトークンの紐づけ対象となるユーザを変更する「固定額面方式」のトークン型を検証対象とした。フェーズ 2 では、額面を固定せず、送金時に必要に応じて既存のトークンの統合・分割を行い、それにより、新たに生成されたトークンとユーザを紐づける「変動額面方式」のトークン型を検討対象とした。変動額面方式では、既存のトークンはインプットとして統合・消費され、新たなアウトプットトークンとして送金分とお釣り（自己宛て払い）の 2 つに分割・作成され、その新規トークンが次の送金時のインプットとして使用される。こうした送金時のトークンの利用のされ方を考慮すると、変動額面方式は、UTXO（Unspent Transaction Output）モデルに類似したデータモデルといえる[21]（図表 13 参照）。

図表 13 変動額面方式トークンと UTXO を用いた送金例[22]



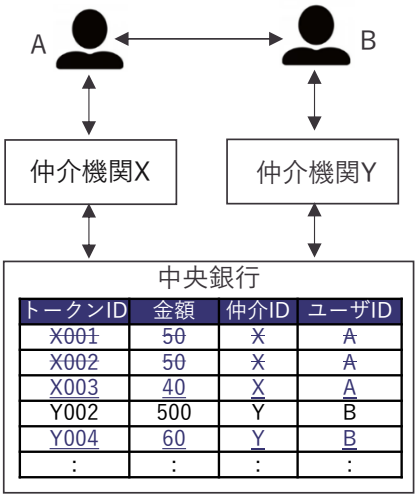
変動額面方式トークンの実装方法について、まず台帳管理を中央銀行のみで行うパターン 3 の場合を考えると、送金元の仲介機関が、送金に使用するトークンを選定し、お釣りを算出し、新規トークンを生成した上で、中央銀行に送金依頼を実施し、中央銀行が台帳を更新する方法が考えられる（図表 14 参照）。

---

[21] UTXO モデルでは、過去のアウトプットに関し、消費（スペント）されたものもインプットとの紐づけがされた形で情報が記録されるケースが多いが、今回の変動額面方式では、アンスペントなアウトプットトークンの情報のみ記録される。

[22] UTXO を用いた送金トランザクションのイメージは、ECB「Documents for the digital euro prototyping exercise」（2022 年 12 月）Annex 1 を参考に作成した。

図表 14 パターン 3（中央銀行による中央管理）の例



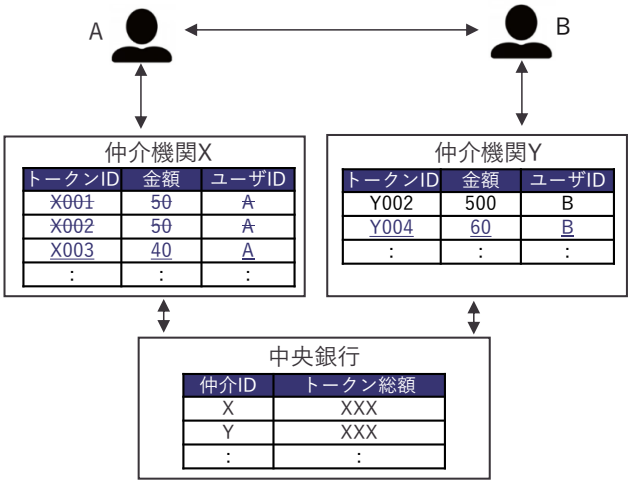
例）Aが100円分のトークンを用いてBに60円送金
- 送金元仲介機関のXは、
  - ✓ 送金に使用するトークンの選定（X001, X002）
  - ✓ 自己宛て払いの算出（40）
  - ✓ 新規トークンを生成（X003,Y004）

  を行い、中央銀行に情報連携
- 中央銀行は台帳更新

台帳管理を中央銀行と仲介機関とで分担管理するパターン 4 の場合、ユーザが保有する個別トークン情報は各仲介機関で管理されることが前提となる。この場合、仲介機関を跨ぐ送金においては、送金されるトークンは、送金元仲介機関で付番・作成され、送金後、送金先仲介機関がその管理を引き継ぐ（なお、送金元仲介機関に残存するトークンは、引き続き送金元仲介機関が管理を維持する）と想定される。

上記仲介機関の想定のもと、以下のように、中央銀行は各仲介機関単位で合算されたトークン総額を管理するデザインが考えられる（図表 15 参照）。中央銀行台帳における更新処理は集中することになるが、例えば 3.1.1 で記載した通り、中央銀行台帳のレコード分割などパターン 2 同様の工夫を施せば、仲介機関の台帳では複数リクエストを並列に処理できるトークン型の性能面での良さは維持され得る。

図表 15 パターン 4（中央銀行と仲介機関の分担管理）の例①



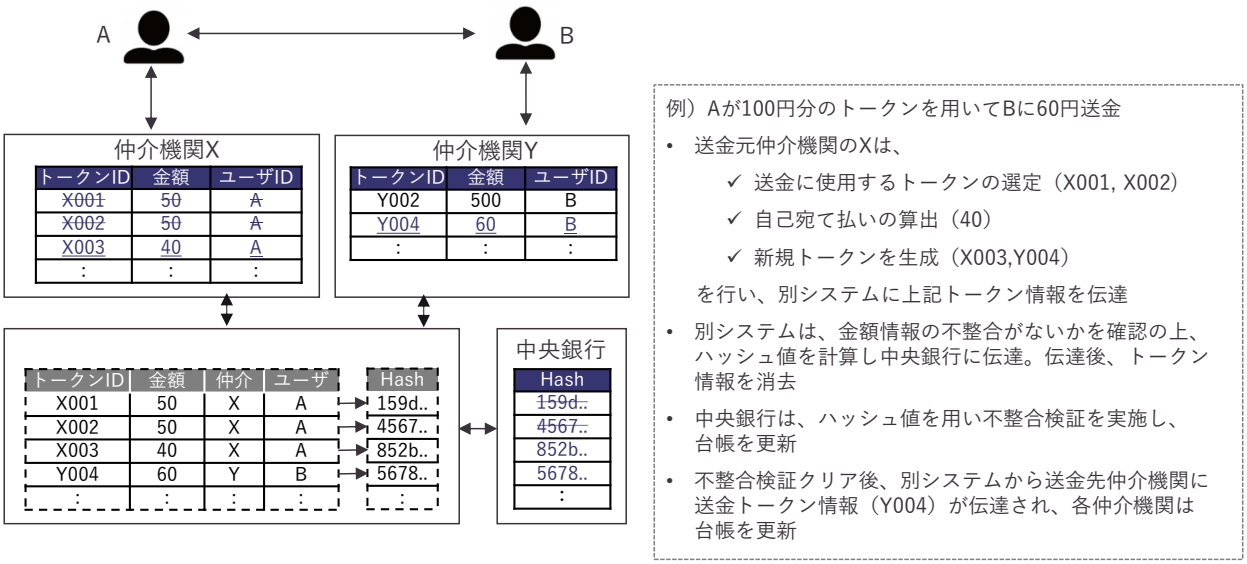
例）Aが100円分のトークンを用いてBに60円送金
- 送金元仲介機関のXは、
  - ✓ 送金に使用するトークンの選定（X001, X002）
  - ✓ 自己宛て払いの算出（40）
  - ✓ 新規トークンを生成（X003,Y004）

  を行い、中央銀行を経由し送金先仲介機関と情報連携
- 送金元・先仲介機関は個別トークン情報の台帳更新
- 中央銀行は各仲介機関のトークン総額のみ台帳更新

この他、中央銀行が、仲介機関間で流通するトークン全体の整合性維持の役割の一端を担いながら、個々のトークン情報の詳細を保持しないデザインも考えられる。その一例として、送

金元仲介機関から伝達されるトークンの金額情報について、別システムにおいて不整合がないかを確認した上でハッシュ値[23]を計算し（計算後トークン情報は消去）、そのハッシュ値のみが中央銀行へ伝達され、中央銀行でトークンの二重払い検知を行うデザインが考えられる（図表 16 参照）。

図表 16 パターン 4（中央銀行と仲介機関の分担管理）の例②



パターン 3、4 ともに、複数の実装案が考えられるが、いずれの場合においても、固定額面方式で必要であった両替処理がなくなる分、送金時に更新するトークン数は相応に減少する。そのため、変動額面方式においては、複数のリクエストを並列に処理できるトークンの性能面での特長が活かされやすくなると考えられる。ただし、統合・分割のアルゴリズム次第でその程度は変化し得るほか、口座型と比較した場合には、必要なリソース使用量は増える可能性や、性能を維持した形での保有額制限といった周辺機能の実装の難度は上がる可能性がある。

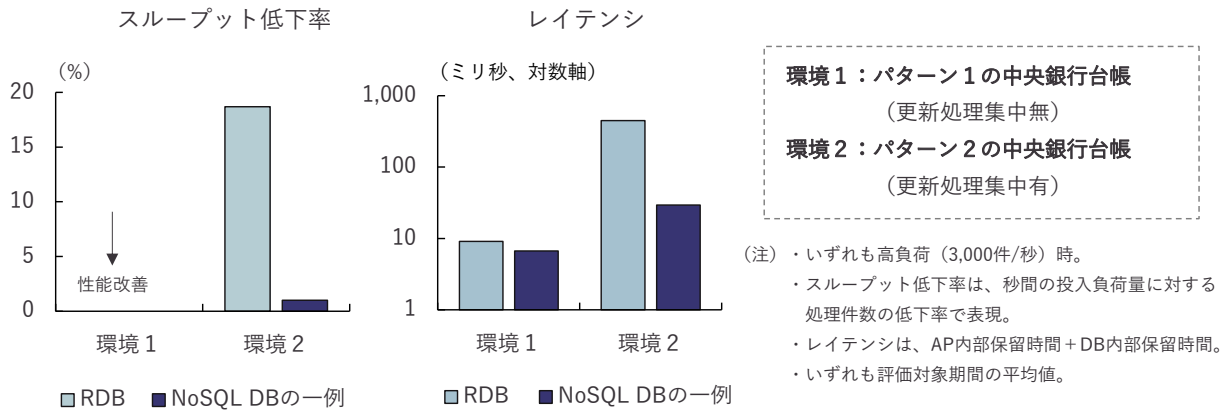
[23] ハッシュ値とは任意の入力値から規則性の無い固定長の値を得る関数（暗号学的ハッシュ関数）の出力として算出される値で、ハッシュ値を用いることで、入力情報の改ざんを纏めて検知できる（入力で用いる情報が少しでも異なれば出力のハッシュ値が変わるため）ほか、直接、ユーザのトークン情報を保有しなくてよい（入力情報が不可逆変換されたハッシュ値のみ管理すればよい）等の利点がある。

3.2.2 NoSQL データベース

実験環境用システムにおけるデータベースには、フェーズ 1 以降、伝統的な RDB を採用してきたが、CBDC システムには極めて高い処理性能が求められるため、フェーズ 2 では、処理速度や性能拡張性が高いと言われるものもある RDB 以外のデータベース、すなわち NoSQL データベースの活用可能性について検討した。

実機検証では、パブリッククラウド上に構築した口座型台帳システムの実験環境で、NoSQL データベースのうち Key-Value 型[24]のデータベースを用い、性能テストを実施した。テストに当たっては、決済業務に必要だと考えられるトランザクション機能（ここでは 1 つの取引で送金元の減額と送金先の増額の 2 つのレコードの更新を同時にコミットする機能を指す）を利用するなど、必要な整合性を確保する実装[25]を行った。処理集中の程度が異なる 2 つの環境、すなわちパターン 1 中央銀行台帳のように 10 万のユーザ口座に更新がほぼランダムに起こる環境とパターン 2 中央銀行台帳のように 5 つの仲介機関が持つユーザ口に更新が集中する環境に対して高負荷シナリオを用いた。テストの結果、いずれの NoSQL データベースも、更新処理集中時であっても必要な整合性を維持し、このうちインメモリ型のデータベース[26]においては、処理速度が向上する可能性が確認された（図表 17 参照）。

図表 17 性能テスト結果



（注）・いずれも高負荷（3,000件/秒）時。
・スループット低下率は、秒間の投入負荷量に対する処理件数の低下率で表現。
・レイテンシは、AP内部保留時間＋DB内部保留時間。
・いずれも評価対象期間の平均値。

[24] NoSQL データベースには、様々なタイプが存在し、そこで用いられるデータモデルも様々である。今回の性能テストは、口座型台帳システムの実験環境上で行うことから、データを識別できる一意の Key と、Value（値）のシンプルな組合せのデータモデルで、行方向の処理に特化した Key-Value 型の NoSQL データベースを採り上げた。

[25] トランザクション機能に加えて、例えば、複数のデータ更新処理が競合してエラーとなった場合に、リトライ処理できる機能をアプリ側に追加実装した。すなわち、今回利用した NoSQL データベースでは、RDB では一般的なレコードロック機能を使わず、データの更新状況を監視することで整合性を制御しており、デフォルトの仕様では、データ更新処理が競合すると後続処理がエラーで返され、再実行されないようになっていた。そこで、データ更新処理の競合時に後続処理がリトライとして再実行される仕組みをアプリで導入する実装を行った。

[26] メモリ（主記憶装置）上でデータを管理するデータベース。ディスク上にデータを管理する従来のデータベースよりも高速なデータの読み書きを可能とする。

テスト結果を踏まえた机上検証結果からは、データベースによっては、社会実装時に想定される負荷量においても高い性能拡張性が維持される可能性も確認され、台帳システムに活用可能性があることが示唆された[27]。一方、トランザクション機能を利用すると、シャーディングに制約がかかるなど性能拡張性が阻害される可能性があり、台帳としての利用には更なる工夫が必要となるデータベースもあった。ただし、そのようなデータベースであっても、CBDC システムのうち、トランザクション機能が不要なデータ（例えば周辺機能実装のための口座ステータス管理や、残高参照用のレプリカデータなど）を扱うデータベースとしての活用可能性があることも示唆された。

今回検証した以外にも多くの種類の NoSQL データベースが存在する他、RDB であっても高性能なデータベースが出現し始めている。想定される業務量によっては、伝統的なデータベースを用いて安定的なシステムを構築することで、より多くの利点がある可能性もある。新たなデータベースの活用可能性については、性能面以外の観点（例えばインメモリ型データベースにおける障害時のデータ保全[28]など）からの検証も必要な点には留意が必要である。今後も、想定される業務量等の要件を踏まえ、幅広い観点から、進化を続けるデータベース技術について調査を続けることが重要である。

[27] 実機検証では口座型の台帳システムを対象にしたが、トークン型の台帳システムの場合を考えると、両替や統合・分割の処理がある分、社会実装時に必要なデータベースのレコード数の総量を正確に想定することは、口座型対比より難しくなる可能性がある。こうした観点からは、NoSQL データベースの性能拡張のしやすさは、口座型よりもトークン型の方が、よりメリットとなる可能性が示唆される。

[28] データ保全への一般的な対応としては、何らかの形でデータをストレージに記録する施策がとられることが多い。施策によっては処理時間が追加的にかかるなど、性能面でのトレードオフが生じ得る。

4 結論

概念実証フェーズ 2 では、仮に社会実装を想定する場合に技術的な課題を早めに確認しておくことが望ましいと考えられる周辺機能について、その処理性能や技術的な実現可能性を検証した。この他、データモデルやデータベースに関して、フェーズ 1 では検討しなかった新しい技術の活用可能性についても、追加的に検討を行った。

周辺機能の検証では、銀行預金から CBDC への急激なシフトを招かないようにするセーフガードの機能、予約・一括・逆引送金といったユーザの利便性を向上させる機能、仲介機関間の連携や外部システムとの連携機能について検証した。検証結果からは、いずれの周辺機能についても、大きな性能劣化を招くことはなく、社会実装する場合に必要な拡張性や信頼性を確保できる可能性が示唆されたが、そのための工夫や対策などが必要であることも示された。

新たな技術に関する検討では、変動額面方式のトークン型台帳について、いずれの実装案でも、並列処理可能というトークンの性能面での特長は、固定額面方式対比で活かされやすくなる可能性がある一方、口座型対比では必要なリソース使用量が増える可能性や周辺機能の実装難度が上がる可能性が確認された。NoSQL データベースについては、性能向上の観点から、台帳や台帳以外の CBDC システムのデータベースとして活用可能性がある一方、想定される業務等の要件に応じ活用可能性の程度差があるほか、性能面以外の観点からの検討も必要であることが確認された。

CBDC に関する基本的なアイディアが技術的に実現可能かどうかを確認するプロセスである「概念実証」は、所期の目的を達成したため当初予定のとおり 2022 年度に終了し、2023 年度より「パイロット実験」を実施する。パイロット実験においては、エンドツーエンドでの処理フローの確認や、外部システムとの接続に向けた課題・対応策の検討などを行っていくほか、概念実証で必要性が示された工夫や対策などについても検討していく。また、CBDC の設計を適切に進める観点から「CBDC フォーラム」を設置し、リテール決済に関わる民間事業者からアイディアや知見を募り、検討を深めていく。

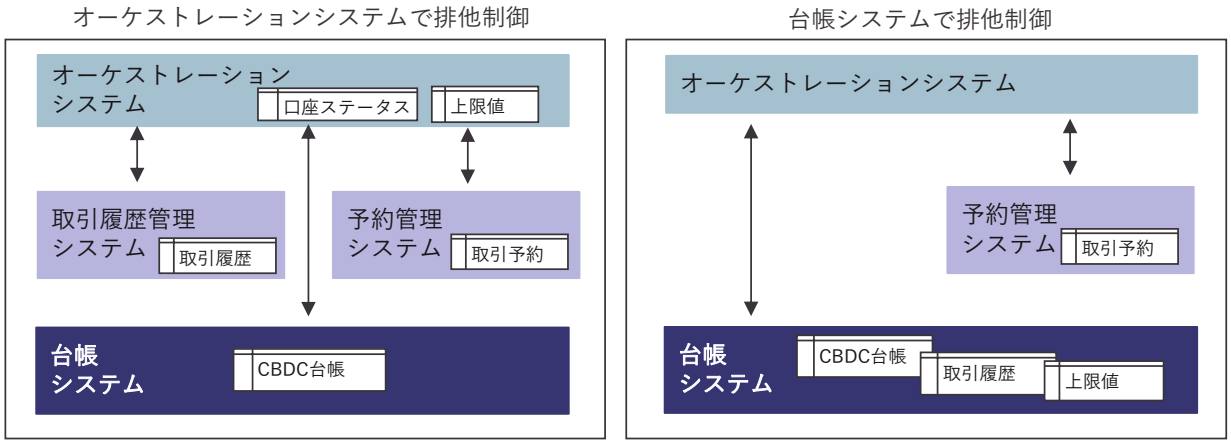
以　　上

補論 1 システム構成のオルタナティブ

周辺機能を実装するにあたり、実験環境用システムでは、CBDC 台帳にいくつかのシステムを追加し、そのうちの１つであるオーケストレーションシステムでシステム間のデータ整合性をとる構成とした。オーケストレーションシステムには、各口座の処理が更新中か否かのステータスを記録管理する DB を持たせ、更新中の場合は原則として他の業務の処理を待機させ[29]、処理の順序を制御するアプリケーションを配置した。

実験環境用システムとは異なるシステム構成としては、台帳システム上に取引履歴等の情報を持たせ、周辺機能の実装を集約する構成が考えられる。この場合、台帳 DB 単体の排他制御機能（いわゆるレコードロック機能[30]）を用いることで、処理の整合性をとることができる。具体的には、台帳 DB の別テーブルに、取引履歴や上限値など各種上限判定処理に関連性の高い情報を直接持たせることで、台帳システムに各種上限判定処理を行わせ、処理の整合性は DB の排他制御機能でとることとなる（補論図表 1.1 参照）。

補論図表 1.1 CBDC システム内の排他制御方法＜パターン 1 の例＞



仮に、実験環境でこの方式をとった場合、オーケストレーションシステム内の口座ステータス管理や取引履歴管理システムが不要となるため、性能テストの結果としては、システムを跨ぐ通信時間等が減少し、処理時間が数十ミリ秒短縮される可能性がある。また、システム構成がシンプルとなる分、障害時の復旧がしやすくなる可能性がある。ただし、台帳システム上で取引履歴や上限値などのデータを扱うことになるため、想定される当該データの量や取引指図の負荷量によっては、台帳システムへの負荷が高まる可能性には留意が必要である。

このほかの異なるシステム構成としては、関係性の深い業務単位でシステムを分割する、マイクロサービスの考えに基づくデザインも考えられる。具体的には、残高に関連する業務（残

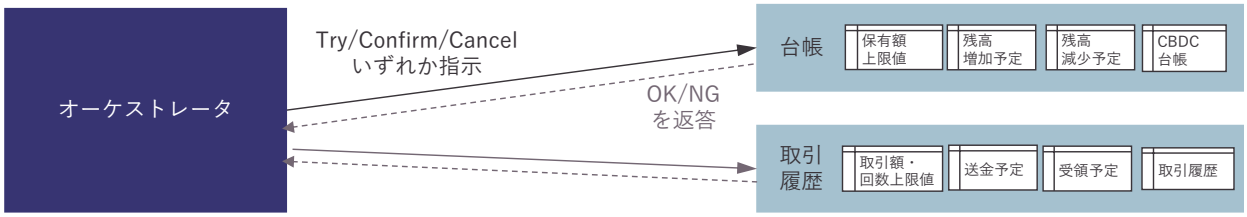
[29] 制限処理に支障が出ない一部業務の組合せは、並列処理させる仕組みをとっている。詳細は脚注 5 参照。

[30] DB のレコードロック機能の詳細は、「中央銀行デジタル通貨に関する実証実験『概念実証フェーズ 1』結果報告書」の補論 3 を参照。

高更新や保有額上限判定）は CBDC 台帳システムが、取引履歴に関連する業務（取引履歴更新や取引額・回数上限判定）は取引履歴管理システムが、それぞれ行う構成が考えられる。

具体的には、例えば以下のような構成があり得る。システム間の処理の制御はオーケストレータが担い、各システムに残高増減や送金・受領の予定の仮登録（Try）を指示し、更新可否を判定する。各システムは、更新可であれば仮登録を行いオーケストレータに OK を、更新不可であれば NG を返す。全てのシステムが OK の場合、オーケストレータは仮登録の内容の確定（Confirm）を各システムに指示し、1 つでも NG の場合は仮登録の内容の取消（Cancel）を指示する。こうした処理方式（Try/Confirm/Cancel<TCC>パターン）をとることが考えられる（補論図表 1.2 参照）。

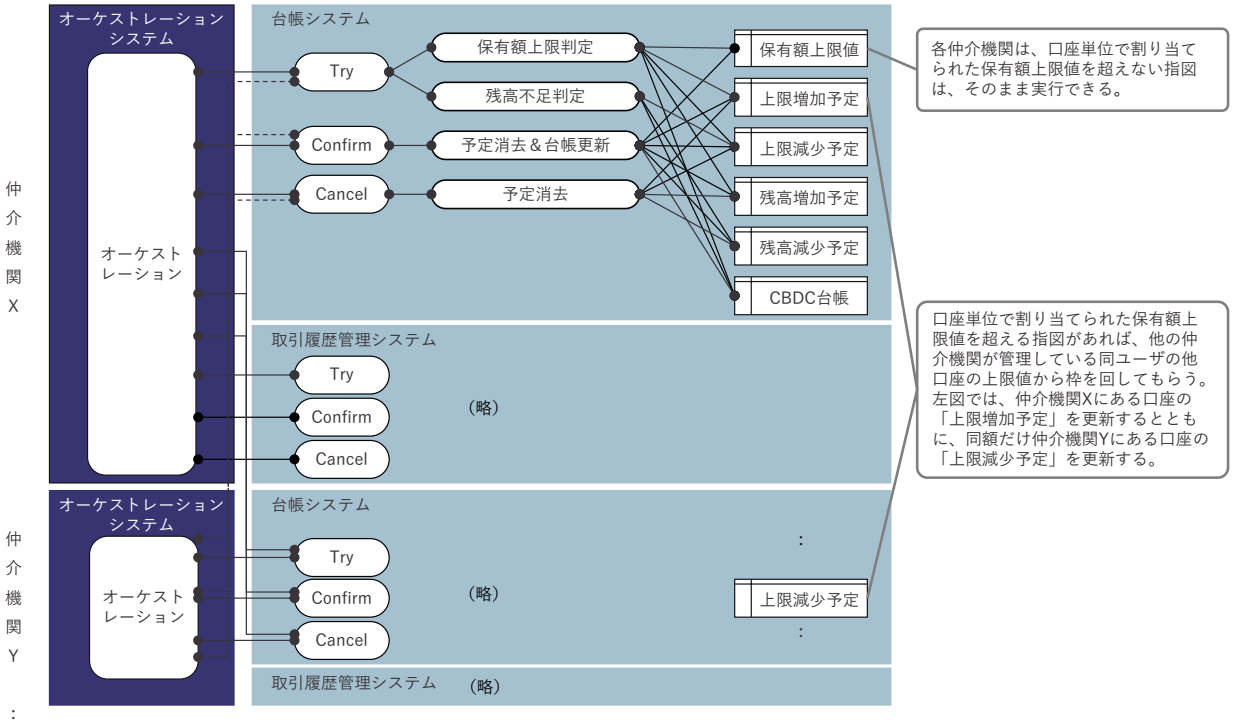
補論図表 1.2 TCC パターンの考え方



なお、パターン 2 で、複数口座の提供を想定し、ユーザ単位で各種制限を課すことを考えた場合、この方式をとると、情報集約のための別システムを不要とできる可能性がある。例えば、各仲介機関の口座にあらかじめ上限値を割り当てることで、上限を超過しない指図に関してはそのまま処理を実行し、上限を超過する指図に関しては、他の仲介機関が管理している同ユーザの別口座の上限余裕枠を融通してもらう仕組みが考えられる[31]（補論図表 1.3 参照）。このような上限を融通する仕組みをとることで、ユーザの残高情報等を別システムに集約せずとも、結果的にユーザ単位での上限判定と同等の処理を分権的に行うことが可能となる。ただし、余裕枠を融通する際に、他の仲介機関に、当該ユーザが他のいずれかの口座で上限超過が生じていることが判明する可能性がある。

[31] より詳細には、上限枠を融通できるよう、各仲介機関に上限の増加/減少予定を管理するテーブルを持たせた上で、例えば、仲介機関 X にある口座の上限増加予定を更新するとともに、同額だけ仲介機関 Y にある口座の上限減少予定も更新する。両者の Try が成功すれば、仲介機関 X の口座の上限値を超えていても、Y の上限余裕枠から融通を受けることで、送金が実行されるようになる。

補論図表 1.3 TCC・パターン 2 システム構成例



上記構成をとった場合、オーケストレーションシステム内で口座ステータス管理を行う場合と比較すると、性能面においては、上限判定処理を各システムに閉じて行える分、処理負荷が減り得る一方、残高増加予定や上限融通処理のような新たにデータを更新する箇所が増える分、処理負荷が増える可能性もあるため、どちらの要素が強く作用するかは、明らかではない。保守・運用面においては、関連する業務を同一システムでカバーしていることから、各システム内で完結するアプリケーションの修正は容易化する可能性がある。

なお、ここで考察したシステム構成は、いずれも台帳システムに追加的な機能を持たせている。この点は、社会実装する場合に高い更新負荷が生じる可能性を踏まえると、台帳システムに残高更新以外の処理を可能な限り行わせない方が適当との考え方（3.1.1 参照）とは異なるコンセプトに立つものと言える。

補論 2 周辺機能の性能測定詳細

実験環境用システムでは、台帳システムと各システム内に、取引指図の処理を実行する AP サーバとその結果を記録・保持する DB サーバ（RDB）を配置した[32]。性能テストは、周辺機能を追加した場合の性能変化に焦点をあてるため、基本的な設定[33]はフェーズ 1 から変えず、複数の負荷シナリオを用いて行った。

負荷シナリオは、以下の図表のとおり、周辺機能に関する追加負荷量にいくつかのバリエーションを持たせる 3 つとした（補論図表 2.1 参照）。シナリオ 1（S1）では、基本機能の取引に直接付随する各種制限の上限判定処理のみ（各種上限超過は生じない想定）を追加、シナリオ 2（S2）では、S1 の負荷量に加え、ユーザの条件や属性によって頻度や取引量が変動し得る、利息適用以外の周辺機能を追加した。シナリオ 3（S3）では、S1 の負荷量に加え、全ユーザを対象とするため取引量が多く、夜間等、台帳に負荷がかからない時間帯に処理を行うことが予想される利息適用のみ追加した。

補論図表 2.1 負荷シナリオ

	基本機能		ユーザ属性に応じたスウィング	逆引送金	一括送金	予約送金	利息適用
	各種上限超過無	保有額上限超過分のスウィング					
S1							
S2							
S3							

通常負荷：合計で500件/秒、高負荷：合計で3000件/秒
通常負荷：各40件/秒、高負荷：各240件/秒

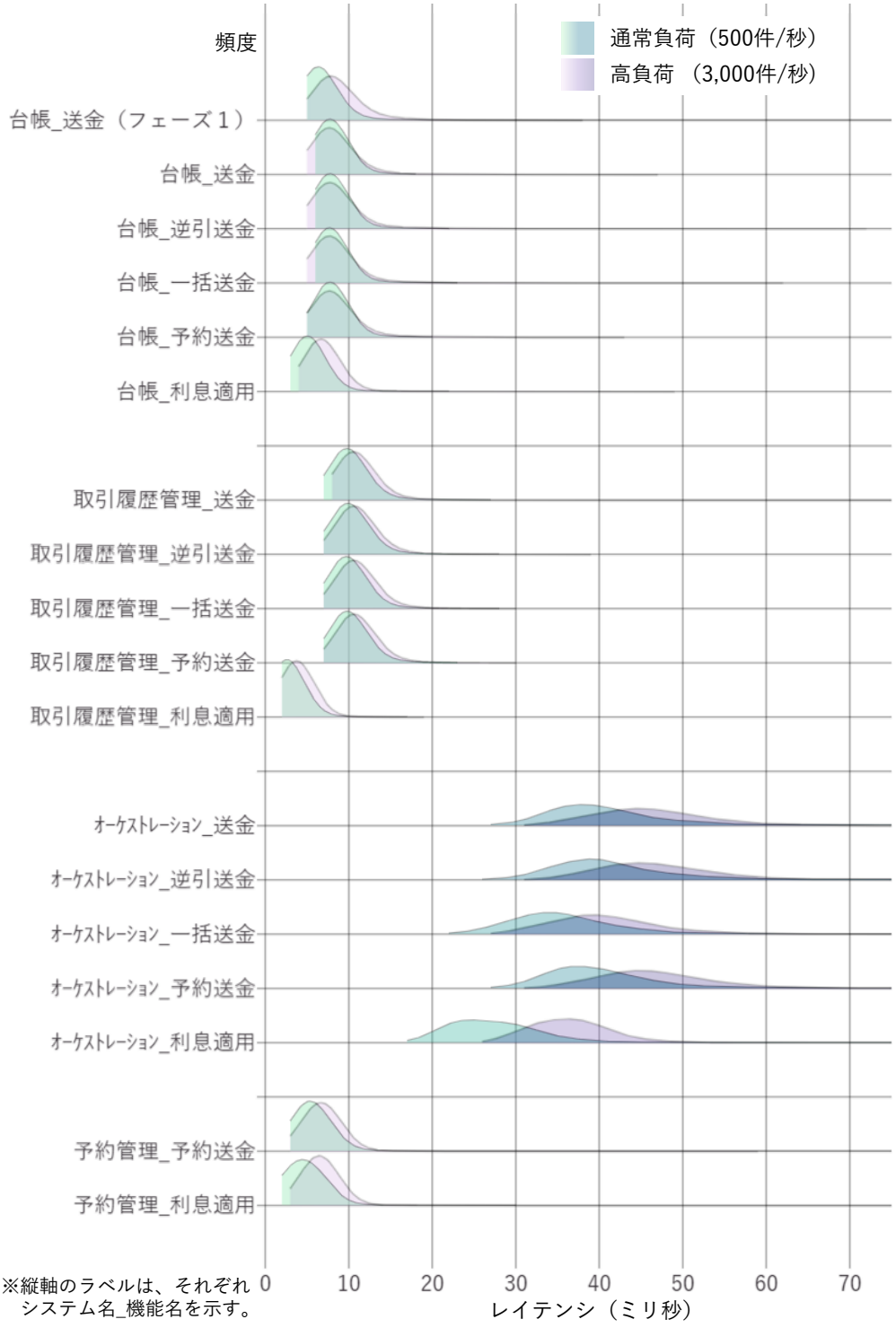
主な性能テストの結果は本文に記載のとおりであるが、レイテンシについてシステム別にみると、いずれの業務においても、オーケストレーションシステムでの水準が相対的に高く、分布の形状もより平らになっていることが確認できる（補論図表 2.2 参照）。オーケストレーシ

[32] 各システムに配置された AP サーバと DB サーバのスペックは、vCPU8 コア（オーケストレーションと取引履歴管理システムの DB サーバのみ 16 コア）、メモリ 64GB（同 128GB）、ストレージ SSD。配置した AP サーバの台数は、台帳システムが通常負荷時：3 台、高負荷時：10 台。オーケストレーションシステムが通常負荷時：3 台、高負荷時：18 台。取引履歴管理システムが通常負荷時：2 台、高負荷時：10 台。予約管理システムが通常負荷時：1 台、高負荷時：3 台。配置した DB サーバの台数は、全システムにおいて、負荷によらず 1 台。

[33] フェーズ 1 同様、ユーザ数は 10 万人、仲介機関は 5 先とし、基本機能の取引指図の負荷量は通常負荷シナリオとして秒間 500 件、高負荷シナリオとして秒間 3,000 件、基本機能の業務比率は送金 90%（同一仲介機関内の送金 30%、異なる仲介機関間の送金 60%）、払出・受入各 5%、発行・還収・残高照会各 0.08%（1 仲介機関あたり 1 分に 1 件）とした。テスト期間についても、取引指図を 15 分間投入し続け、安定した計測値が取得できる開始後 5 分から 10 分の間を評価対象期間とした。

ョンシステムのレイテンシには、各種上限判定処理が含まれているほか、相対的に変動の大きいシステム間通信等が含まれていることが影響していると考えられる。

補論図表 2.2 システム別・業務別のレイテンシ分布



※縦軸のラベルは、それぞれ
システム名_機能名を示す。