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

日本銀行決済機構局

2022 年 4 月

目次

１．概念実証フェーズ１の目的

日本銀行では、現時点で中央銀行デジタル通貨（CBDC）を発行する計画はないものの、決済システム全体の安定性と効率性を確保する観点から、今後の様々な環境変化に的確に対応できるよう、技術的な実証実験を含め、しっかり準備しておくことが重要と考えている。

そうした実証実験の第１段階として、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」を実施した。その目的は、CBDC システムの基盤となる「CBDC 台帳」を中心に、実験環境を構築したうえで、CBDC に関する基本的な取引（発行、払出、移転、受入、還収等）を的確に処理することができるかどうかを検証することである。こうした目的を達成するため、実験では、内外の議論も参考としながら、CBDC 台帳に関する３つの設計パターンをパブリッククラウド上に構築した[1]。そのうえで、将来、本番用システムを開発することになった場合を想定しつつ、実機検証および机上検証を通じて、システムの処理性能や信頼性、機能拡張性等についてパターンごとの比較・検証を行った。

２．検証方法

（１）台帳の設計パターン

実験用に構築した CBDC 台帳の３つの設計パターンのうち、パターン１およびパターン２は、CBDC の保有状況を、仲介機関やエンドユーザが有する口座の残高として認識する「口座型 CBDC 台帳システム」である（図表１）。このうち、パターン１は、中央銀行が全ての仲介機関とエンドユーザの口座残高を記録する台帳を管理する方法である。一方、パターン２は、中央銀行が、仲介機関（自己口・ユーザ口）の口座残高を記録する台帳を管理し、仲介機関が、それぞれ自らの顧客ユーザの口座残高を記録する台帳を管理する方法である。パターン３は、一定額面の金銭データに固有の識別子（ID）を付与し、そうした ID とユーザ ID の紐付けにより、CBDC の保有状況を認識する「トークン型 CBDC 台帳システム」である[2]。

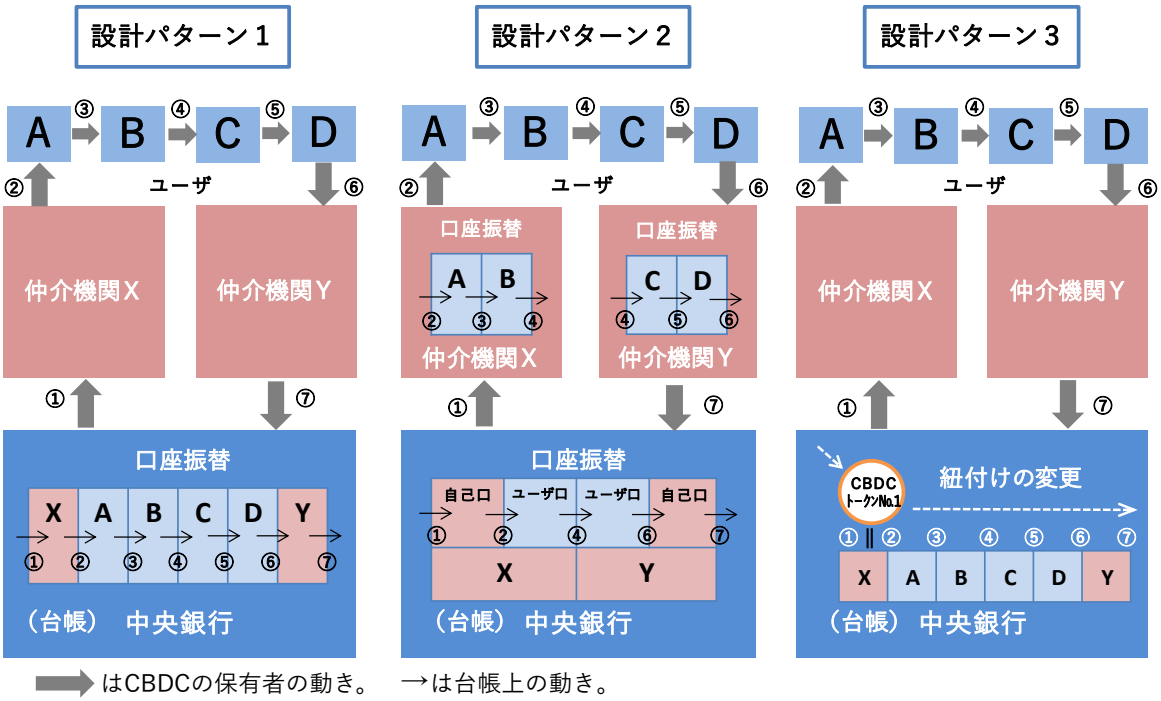
---

[1] 構築した 3 つの設計パターンの台帳システムは、いずれも、中央銀行または仲介機関がそれぞれ単独で管理する「中央管理型」であり、同一の台帳を取引の参加者が共同で管理する「分散型台帳技術」は用いていない。

[2] トークン型台帳システムの特徴については補論１を参照。なお、概念実証フェーズ１では、「トークン」や「トークン型 CBDC 台帳」を上記の意味で用いているが、「トークン」という用語に確定的な定義はなく、文脈によって様々な意味で使われていることには留意を要する。

なお、いずれの台帳システムにおいても、CBDC は、中央銀行から仲介機関に対して「発行」され、仲介機関を通じてエンドユーザに「払出」される。払出された CBDC はエンドユーザ間を「移転」する。CBDC は、仲介機関によってエンドユーザから「受入」れられ、仲介機関を通じて中央銀行に「還収」される。

（図表１）CBDC 台帳の設計パターン



（注１）①：発行、②：払出、③・⑤：同一仲介機関内の移転、④：仲介機関を跨ぐ移転、⑥：受入、⑦：還収
（注２）パターン２において、払出（②）、仲介機関を跨ぐ移転（④）、受入（⑥）は、中央銀行台帳と仲介機関台帳の口座残高が同時に増減する。同一仲介機関内の移転（③、⑤）の動きは、中央銀行台帳に反映されない。

設計パターン１：発行〔還収〕により、中央銀行台帳における仲介機関の口座残高が増加〔減少〕する。払出〔受入〕により、仲介機関の口座残高が減少〔増加〕し、エンドユーザの口座残高が増加〔減少〕する。移転により、送金側のエンドユーザの口座残高が減少し、受領側の口座残高が増加する。

設計パターン２：発行〔還収〕により、中央銀行台帳における仲介機関の口座残高（自己口）が増加〔減少〕する。払出〔受入〕により、中央銀行台帳における仲介機関の口座残高（自己口）が減少〔増加〕し、当該仲介機関の口座残高（ユーザ口）が増加〔減少〕する。また、仲介機関台帳におけるエンドユーザの口座残高が増加〔減少〕する。仲介機関を跨ぐ移転により、送金側の仲介機関台帳におけるエンドユーザの口座残高が減少し、受領側の仲介機関台帳におけるエンドユーザの口座残高が増加する。また、中央銀行台帳における送金側の仲介機関の口座残高（ユーザ口）が減少し、受領側の仲介機関の口座残高（ユーザ口）が増加する。同一仲介機関内の移転により、仲介機関台帳における送金側のエンドユーザの口座残高が減少し、受領側の口座残高が増加する。

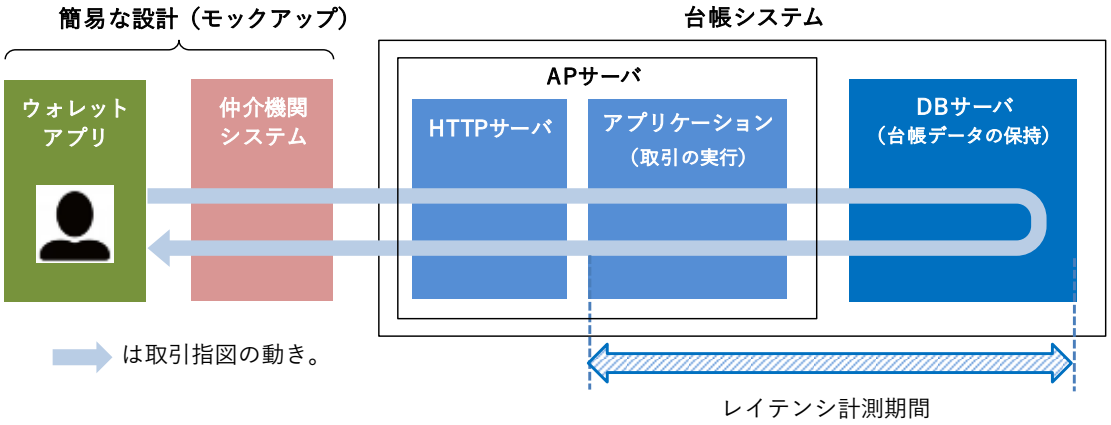
設計パターン３：発行〔還収〕により、中央銀行台帳において仲介機関と紐づくトークンが作成〔削除〕される。払出、移転、受入により、対象となるトークンと仲介機関、エンドユーザの紐付けが変更される。

（２）実機検証[3]

実験環境

実機検証では、主として、CBDC の基本的な取引に関する台帳システムの処理性能について検証した。これを実施するために実験環境用のシステムをパブリッククラウド上に構築した（図表２）。このうち、CBDC 台帳システムは、取引指図の処理を実行するためのアプリケーション（AP）サーバと、その結果を記録・保持するデータベース（DB）サーバから構成される。設計パターン１と３の台帳システムは中央銀行分のみである一方、設計パターン２では、仲介機関の台帳システムも別途構築した。台帳の外側にある仲介機関システムやエンドユーザが利用するウォレットアプリについては、取引指図を投入するだけの簡易な設計（モックアップ）とした。

（図表２）実験環境用システムの概要



基本設定

３つの設計パターン共通の前提として、エンドユーザ数は 10 万人、仲介機関は５先（大規模２、中規模１、小規模２）、各仲介機関（モックアップ）が投入する取引指図の件数はその規模に応じて設定した。取引指図の構成比は、払出５％、同一仲介機関内の移転 30％、仲介機関を跨ぐ移転 60％、受入５％とした[4]。このほか、パターン１と２については、各エンドユーザがそれぞれ１つの口座を有するとし、パターン３については、発行

[3] 実験環境用のシステムと処理性能の測定方法の詳細は補論２を参照。

[4] このほか、仲介機関から中央銀行に投入する「発行」および「還収」の取引指図は、１仲介機関当たり１分に１件とした。なお、本番環境では「残高照会」についても相応の取引指図が投入されることが想定されるが、今回の実験は払出、移転、受入等に関する性能測定が主眼であるため、１仲介機関当たり１分に１件とした。

総トークン数を 2,500 万、１回の取引指図において更新対象となるトークン数を 10 と設定した[5]。

負荷シナリオ

概念実証フェーズ１では、将来、実際に CBDC を発行することとなった場合の環境（本番環境）で求められる処理性能を、通常時スループット数万件/秒、ピーク時同 10 万件以上/秒、レイテンシ数秒以内と想定したうえで、その実現に向けて、設計パターン間の性能差やボトルネックの所在が明らかになるよう、実験用のシナリオを設定した。具体的には、いずれの設計パターンにおいても問題なく取引を処理できる水準として秒間 500 件の取引指図を投入する「通常負荷シナリオ」と、各パターンの処理性能に影響を与えうる水準として秒間 3,000 件の取引指図を投入する「高負荷シナリオ」の２つを準備した。

計測値

各パターンの処理性能は、以下の３つの計測値により評価した。

① スループット：CBDC 台帳システムが、１秒間で処理した取引指図の件数（AP サーバ上のアプリケーションの秒間正常処理件数）

② レイテンシ：取引指図１件の処理時間（アプリケーションと DB の処理時間の合計）

③ リソース使用量： AP サーバと DB サーバの CPU 使用率

（３）机上検証

机上検証では、まず、各パターンの「性能拡張性」を検証した。具体的には、実機検証における性能テストの結果を踏まえ、パターンごとに、本番環境で求められる処理性能を達成するための施策や課題を検討した。

このほか、CBDC 台帳システムの「信頼性」（セキュリティリスクへの耐性、構造面の障害耐性、可用性）や「機能拡張性」（周辺機能の実装容易性等）について、パターンごとに比較・検証した。

---

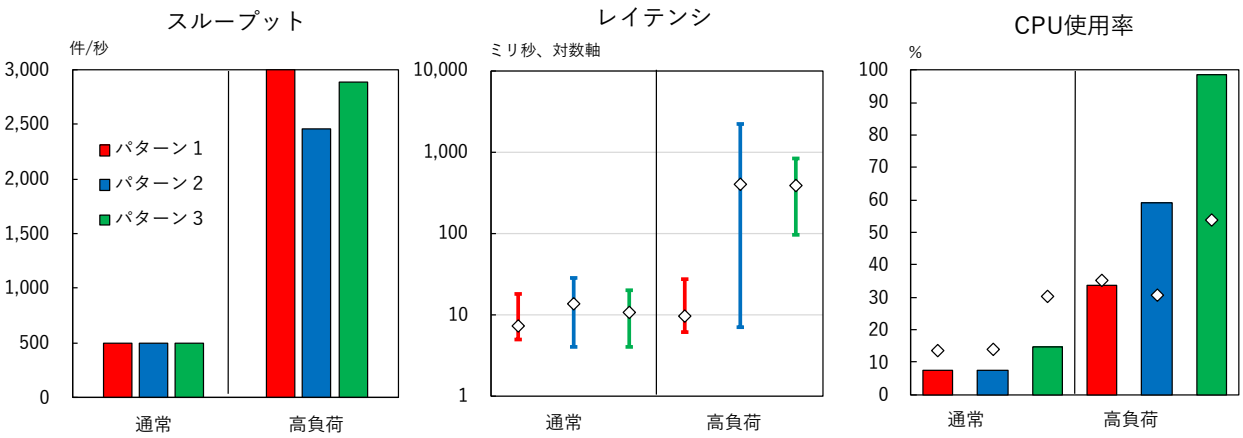
[5] パターン３では、エンドユーザが移転額に一致するトークン群を保有していない場合に対応するため、仲介機関が手元のトークンを用いてエンドユーザとの間で両替を行うこととした。そのうえで、事前のシミュレーション結果を踏まえ、１回の取引指図において更新対象となるトークン数を 10、取引指図の 50％で両替処理が発生すると仮定した。この仮定に基づき、15 分間の性能テスト中に移転や両替を繰り返してもエンドユーザや仲介機関においてトークンが枯渇しないよう、エンドユーザがそれぞれ 200 のトークン、仲介機関が５先合計で 500 万のトークンを保有している状態、すなわちトークンの発行総数が 2,500 万の状態から実験を開始した（トークン型台帳システムの特徴に関する詳細は補論１を参照）。

３．検証結果

（１）性能テストの結果

実機検証の通常負荷シナリオにおいては、全ての設計パターンで、事前の設定どおり秒間 500 件のスループットを達成し、レイテンシは約 10 ミリ秒、CPU 使用率は 10～30%となった（図表３）。高負荷シナリオにおいては、設計パターン１は、取引指図の投入件数と同じ秒間 3,000 件のスループットを達成し、レイテンシは通常負荷時とほぼ不変であった。一方、設計パターン２と３のスループットは取引指図の投入件数に比べてそれぞれ 18%、４%低下し、レイテンシは 99%タイル値でみるとそれぞれ 2,000 ミリ秒、800 ミリ秒程度まで増大[6]した。AP サーバと DB サーバの CPU 使用率は、いずれのパターンも通常負荷時より上昇したが、パターン３の DB サーバの CPU 使用率は上限の 100%近くまで上昇した。

（図表３）性能テストの結果



（注）いずれも測定評価期間平均値。レイテンシは1%～99%タイル値のレンジ、◇は平均値。CPU使用率は中央銀行台帳分、棒線はDB分、◇はAP分。

以下では、パターン２と３において性能低下をもたらしたボトルネックの所在について検証する。

パターン２の性能ボトルネック

パターン２の DB サーバの処理状況を仔細にみると、パターン１や３ではみられない「口座残高データに対する処理の集中」が多数発生している。今回構築した実験環境では、口座残高データ（レコード）を更新する複数の取引指図が行われた場合、先着の処理が完

[6] 図表中のレイテンシはアプリケーションと DB の処理時間の合計であるが、仔細にみるとほぼ DB 分で増大している。

了するまで後続の処理が実行されないよう、処理中のレコードがロックされ（レコードロック）、その後、先着の処理が完了した時点で、レコードロックの解除および後続の処理が実行される。パターン２の場合、特に、仲介機関を跨ぐ移転（図表１の④）の指図によって当該仲介機関のユーザ口に多くの処理が集中している。こうしたレコードロックに伴う処理遅延により、パターン２のレイテンシはパターン１に比べ増大した[7]。

レコードロックの影響は、高負荷シナリオにおける DB サーバの CPU 使用率を高める要因の一つにもなっている。すなわち、レコードロックが発生している間は、DB サーバ上に実行待ちの処理が滞留することになるが、こうした後続の処理が、レコードロックの解除後該当レコードにすぐにアクセスできるよう、常時当該レコードのステータスを監視する必要がある。このため、同一レコードに対する滞留件数が多いほど、CPU 使用率は上昇する。

パターン３の性能ボトルネック

パターン３については、高負荷シナリオにおける DB サーバの CPU 使用率が 100%近くまで上昇しているため、処理に必要なリソースが枯渇していることが分かる。これは、取引指図１件ごとに、複数のトークンについて保有者 ID の更新処理が行われることや、一定の割合で両替処理が行われることにより、他のパターンと比べて処理件数が大きく増加していることが主な要因である。こうしたリソース制約が、スループットの低下やレイテンシの増大にも繋がっている。

（２）本番環境の実現に向けた性能面での課題

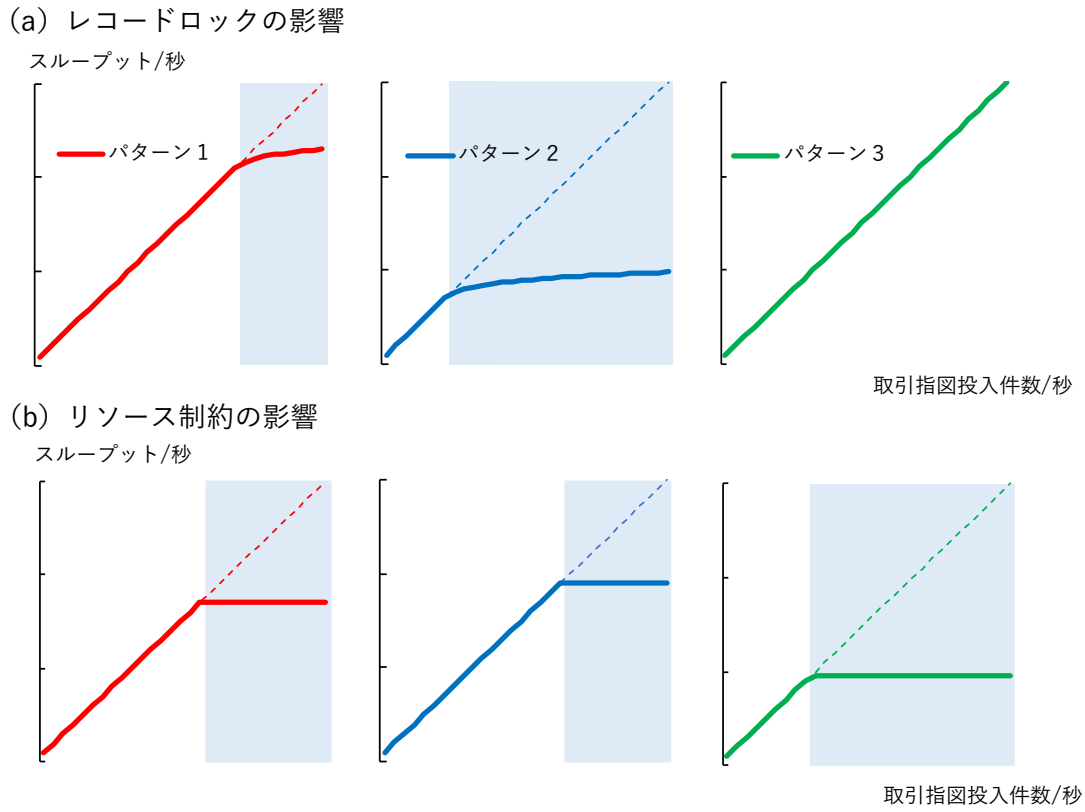
（１）の性能テストの結果を踏まえ、本番環境で求められる処理性能を達成するために必要となる施策や課題について検討した。

性能テストの結果から、パターン２ではレコードロックの影響が、パターン３ではリソースの不足が、それぞれ性能面でのボトルネックであることが分かった（図表４）。今回のテストでは、パターン１についてレコードロックの問題はみられなかったが、口座型 CBDC 台帳である以上、取引指図の投入件数次第では同様の問題が生じる可能性がある。また、リソース制約の影響は、今回、パターン３において顕著に現れたが、パターン１、２についても、取引指図の投入件数の増加とともに、いずれ同様の問題に直面し得る。本番環境の実現に向けては、こうしたボトルネックを取り除く施策を講じる必要がある。

---

[7] レコードロックの影響の詳細は補論３を参照。

（図表４）取引指図投入件数とスループットの関係



（注1）取引指図の投入件数とスループットの関係を示したイメージ図。パターン２は中央銀行台帳分。
（注2）シャドーは、ボトルネックの存在によりスループットが取引指図の投入件数より低下
している状況、破線は、そうしたボトルネックが解消した状況を示している。
（注3）(a)(b)はそれぞれ他方の影響を除いた状況を示している。

まず、レコードロックの影響については、これを軽減させるための施策として、CBDC台帳に記録される口座残高データを複数に分ける「レコード分割」が考えられる。例えば、パターン２において、仲介機関の「ユーザ口」の口座残高データを複数に分割し、それに対応する形で当該仲介機関に属するエンドユーザをグループに分けることで、１つのユーザ口に処理が集中することを軽減できる[8]。また、業務処理フローの見直しにより対応する方法も選択肢となり得る。例えば、パターン２においては、仲介機関を跨ぐ CBDC の移転（図表１の④）の指図を処理する際、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同時に行う設計としているが、こうした同時性を緩和し、別々のタイミングで口座振替を行うことを許容すれば、少なくとも技術的には、レコードロックの影響を軽減できると考えられる。

[8] ただし、口座残高のレコード分割数が多くなると、当該口座の残高を計算する場合などにおいて足し上げるべきレコード数が増えるため、その分、処理時間が長くなる可能性がある。このように、レコード分割の方法等によっては性能向上効果が低下する可能性がある点には留意する必要がある。

リソース制約の問題については、大別して、スケールアップ（サーバの処理性能の向上）とスケールアウト（サーバの増設）の２つの対応策が考えられる。前述のレコードロックに伴う性能面への影響が解消されたことを前提に、スケールアウトのみで本番環境（スループット 10 万件/秒と仮定）を構築する場合に必要となる DB サーバの台数（中央銀行台帳分）を試算すると、パターン１と２はほぼ同じ、パターン３は、その２～３倍程度であることがわかった。なお、DB サーバの増設を行う際には、各 DB サーバが管理するレコードが同一ではないことを踏まえ、特定のレコードに取引指図が偏りスループットやレイテンシが悪化しないよう、DB の設計や配置について工夫する必要がある。

本番環境では、今回の実験で確認された上記以外の箇所（ネットワークの帯域幅やストレージの I/O 性能など）がボトルネックとなる可能性もあるほか、今回の実験用のシナリオとは異なり、一時的かつ局所的に非常に高い負荷がかかるケース（例えば、特定のエンドユーザに対して大量の送金が行われたり、取引指図の構成比が大きく偏ったりするなど）も想定される。さらに、より複雑な周辺機能の実装や、セキュリティリスクへの十分な対応といった各種の追加的な要請を満たすことも求められる。本番環境の実現にあたっては、こうした様々な要素を十分に勘案したうえで、適切なシステムを設計・構築していく必要がある。

（３）本番環境の実現に向けた性能面以外の課題

信頼性

CBDC 台帳システムに求められる「信頼性」に関し、セキュリティリスクへの耐性、構造面の障害耐性、可用性の３つの項目について、各設計パターンの比較を行った。検証の結果、構造面の障害耐性を中心に、設計パターン１および３と、パターン２の間には差異があることを確認した。もっとも、いずれも長短があるほか、本番環境の実現に向けて検討すべき共通の課題も少なくない。以下、項目ごとに述べる。

第１に、「セキュリティリスクへの耐性」（サイバー攻撃等への耐性）については、特定のセキュリティ脅威にとどまらない一般化した評価を行うことができるよう、DDoS 攻撃のようなサイバー攻撃のほか、盗聴や情報改竄などを含む代表的な脅威を洗い出したうえで必要な施策を検討した。検討の結果、システム面において、設計パターンの違いによる大きな差異はないと考えられる。すなわち、CBDC 台帳の構成要素、すなわち AP/DB サーバ等の各システム、およびこれらのシステム間のネットワークに対して求められる個別のセキュリティ施策は、設計パターンによって違いはないと想定される。なお、台帳の管理主体が分かれる設計パターン２については、中央銀行だけでなく各仲介機関も、CBDC 台帳について必要なセキュリティ施策を講じる必要がある点には留意を要する。

CBDC を安心して使えるものとするためには、いずれの設計パターンであっても十分なセキュリティ水準を確保することが求められるが、その場合、システムの処理性能が低下したり、導入や運用にかかるコストが増加する可能性がある。本番環境の実現に向けては、安定的・効率的な決済システムを構築するため、こうしたトレードオフの問題に適切に対処してくことが必要となる。

第２に、「構造面の障害耐性」（障害が発生し得る箇所の数とその影響範囲）については、台帳の管理主体の違いにより、設計パターン間で差異があると考えられる。すなわち、仮に中央銀行が管理する台帳上で障害が発生した場合、設計パターン２では、同じ仲介機関に口座を持つエンドユーザ間の CBDC の移転は可能であると考えられ、障害の影響範囲は相対的に小さいと想定される。一方で、中央銀行と各仲介機関の台帳が分かれているために、障害が発生し得る箇所は設計パターン１、３より多い。また、復旧時のデータ完全性（各台帳上の残高等の変化が互いに整合的であること）に問題が生じやすいと考えられる。

本番環境の実現に向けては、CBDC 台帳のみならず、CBDC システム全体でみた障害耐性について検討する必要がある。例えば、設計パターン１や３においても、エンドユーザへのインターフェースの提供や中央銀行との間の電文授受など、各仲介機関がシステムの運用・管理を担う部分がある。本番時の設計パターン間の差異については、こうした点も踏まえたうえで、総合的に判断する必要がある。

第３に、「可用性」（システム停止時間の有無・長短）については、システム面において、設計パターンの違いによる大きな差異はないと考えられる。すなわち、障害発生時に備えて、マルチサイト等を事前に構築することを想定した場合、その実現容易性は設計パターン間で変わらないと思われる。なお、設計パターン２においては、中央銀行だけでなく各仲介機関も、自らが管理する台帳の可用性を高める施策を講じる必要がある点には留意を要する。

本番環境の実現に向けては、いずれの設計パターンにおいても、障害時のシステム切替えに要するラグをどのように回避・低減するか、あるいはシステムメンテナンスのための計画的な停止を許容するかどうかといった共通の論点がある。CBDC 台帳システムに高い可用性を求める場合には、１サイト２システムの構成やマルチサイトによる両現用構成が必要となるが、その場合、準備や運用に要するコストとのバランスが検討課題となる。

機能拡張性

概念実証フェーズ１では、CBDC の基本的な取引を対象に実機検証を実施したが、本番環境では、これに加えて、様々な周辺機能を実装することになる。そこで、各設計パターンについて、周辺機能を実装していく際の課題を机上で検証した。検証の結果、設計パターン２においては、例えば、異なる仲介機関の台帳に記録されている CBDC 保有額等を相互に参照、合算することを前提とした周辺機能を実装する場合には、そのための処理を追加的に構築する必要があることを確認した。また、設計パターン３においては、例えば、エンドユーザごとの CBDC 保有額を取引の都度取得することを前提とした周辺機能を実装する場合には、各エンドユーザが保有するトークンの額の合計値を算出する機能が追加的に必要になるとの認識に至った。

「CBDC の保有額に対する制限」を例として、この点を説明する。当該機能を導入する場合、CBDC の移転指図が行われる都度、受領者の CBDC 保有額が上限に抵触しないかどうかを確認する仕組みを実装する可能性がある。その場合、設計パターン２については、各仲介機関の台帳システム間で残高の情報を連携するための機能が追加的に必要となる。なぜなら、複数の仲介機関が台帳を分担して管理するパターン２において、仲介機関を跨ぐ移転指図が行われた場合には、送金者側の仲介機関が、受領者側の仲介機関に対して、受領者の CBDC 保有額が上限に抵触するか否かを確認することが想定されるためである。また、トークン型の設計パターン３においては、台帳モデル上、個々のトークンとエンドユーザの紐付けのみが記録され、エンドユーザごとの CBDC の合計保有額は記録されない。このため、移転指図の都度、受領者の CBDC 保有額が上限に抵触しないかどうかを確認するための計算処理が必要となる。

このように、周辺機能の実装容易性という点で、設計パターン１に比べ、設計パターン２や３が劣後する面はある。もっとも、上述した仲介機関間の情報連携のための機能については、CBDC 移転時の相手先確認を目的として設計パターン１に実装することも考えられる。その意味では、パターン２に固有の課題とは言えない。また、設計パターン２では、各仲介機関に台帳設計上の自由度を持たせることができるため、例えば、自らの勘定系システムの仕様に合わせた CBDC 台帳を構築したり、エンドユーザのニーズに即した追加サービスの新規開発を効率的に進めることができるといったメリットが期待される。設計パターン３についても、トークンに対して何らかの情報を付加するような機能やサービスを想定すれば、同パターンのメリットを活かせる可能性がある。

これらの点を踏まえると、「機能拡張性」については、各パターンの特性に応じた違いはあるものの、現時点で、全体として大きな差があるとまではいえない。また、将来的に実装し得る周辺機能は、「CBDC の保有額に対する制限」のほかにも、「エンドユーザに

よる送金指図の予約」や「CBDC の取引額に対する制限」など様々である。概念実証フェーズ 2 以降の実証実験を通じて、今後は、こうした周辺機能を実装する際の技術的な実現可能性や処理性能に及ぼす影響等をより詳細に検証していく必要がある。

4．結論

概念実証フェーズ 1 では、本番環境において求められる処理性能を、通常時スループット数万件/秒、ピーク時同 10 万件以上/秒、レイテンシ数秒以内と想定したうえで、その実現に向けて、CBDC に関する基本的な取引を適切に処理できるかどうかについて設計パターンごとの比較・検証を行った。

実機検証の結果、パターン 2 は、パターン 1 に比べ、レコードロックの影響により処理性能が低下することがわかった。もっとも、取引指図の投入件数次第では、同じ口座型台帳システムであるパターン 1 にも同様の問題が生じ得る。いずれにせよ、本番環境の実現に向けては、レコード分割や業務処理フローの見直し等により、こうした問題に対処していくことが考えられる。

パターン 3 は、パターン 1 やパターン 2 に比べ、同じ取引を処理するために必要となるリソースの消費量が多い。パターン 1 や 2 についても、取引指図の投入件数が増加すれば、いずれリソース面の制約に直面する。本番環境の実現に向けては、パターン 3 を中心に、DB サーバのスケールアップやスケールアウトなど、大規模なリソースの拡充やそれに伴う追加的な工夫が必要となる。

こうした性能面の検証以外では、「セキュリティリスクへの耐性」や「可用性」については、設計パターンの違いに起因する大きな差異はないと考えられる。「構造面の障害耐性」については、パターン 1 やパターン 3 に比べ、パターン 2 は、障害発生時の影響範囲が相対的に小さいと想定される。一方で、障害が発生し得る箇所が多いほか、復旧時のデータ完全性に問題が生じやすいと考えられる。「機能拡張性」については、各パターンの特性に応じた違いはあるものの、現時点で、全体として大きな差があるとまではいえない。

上記の検証結果や設計パターンの差異に起因する課題は、今後、CBDC 台帳システムに必要な機能を追加したり、台帳以外のシステムを構築したりする過程で変化する可能性がある。このため、概念実証フェーズ 2 では、フェーズ 1 で構築した CBDC 台帳により複雑な周辺機能を実装したうえで、その技術的な実現可能性や処理性能に及ぼす影響を検証するなど、本番環境に向けた技術的な検討をさらに深めていくことが適当と考えられる。

以　上

補論１．トークン型台帳システムの特徴

トークン型台帳システムでは、一定額面の金銭データ（トークン）とその保有者を紐付ける。払出や受入、移転の処理を行うときは、トークンと保有者の紐付けを変更する。トークン型台帳システムでは、送金者が十分な数のトークンを保有している場合、口座型台帳システムのように同一レコードに処理が集中する蓋然性は低く、レコードロックの解除待ちによる処理遅延は想定しにくい。一方で、１回の取引指図において多くのトークンが更新対象となり得るため、その分リソース使用量が増大する。

トークン型台帳システムには、CBDC の移転時に既存のトークンの紐付け対象となるエンドユーザを変更する「固定額面方式」のほか、例えば、CBDC の移転時に、必要に応じて既存のトークンの分割・結合を行い、それによって新たに組成されたトークンとエンドユーザを紐付ける「変動額面方式」がある。概念実証フェーズ１におけるパターン３では、固定額面方式を採用した。固定額面方式および変動額面方式に関する概略的な説明は以下の通りである。

固定額面方式

固定額面方式においては、エンドユーザが移転額に一致するトークン群を保有していない場合、仲介機関との間で「両替」を行って小口額面のトークンを入手し、これを移転にあてることが考えられる。受領者から「お釣り」を受け取る方法も選択肢となり得るが、この場合、受領者側もお釣りの額に一致するトークン群を有している必要があるなど、追加的な考慮が必要となる。

なお、「オンライン」の決済を前提とした今回の実験と異なり、各エンドユーザのエンドポイントデバイス同士の通信により CBDC が直接移転し、その情報が各自のデバイスに独立して記録される「オフライン」の決済を想定した場合、使用されたトークンの真正性をどのように確認するかが重要なポイントとなる。この点、流通の過程でトークンの分割・結合が生じない固定額面方式のもとでは、発行時に各トークンに付された中央銀行の電子署名が、移転の都度、そのまま引き継がれることになるため、エンドユーザ間で電子署名を検証すればトークンの真正性を有効に確認できると考えられる。

（補論図表１－１）固定額面方式における A から B への 10,000 円の移転

変動額面方式

変動額面方式においては、エンドユーザが送金額に一致するトークン群を保有していない場合、大口額面トークンを小口額面トークンに分割する。この場合、分割前のトークンは廃棄され、分割後のトークンについては、移転される分が受領者に紐付けられ、残りは送金者に紐付けられる。

変動額面方式については、両替処理を必要としない分だけ固定額面方式より CBDC の移転に関する処理時間を短縮し得るほか、定期的なトークンの結合（小口額面トークン群を大口額面トークンに集約すること）によりデータベースの規模を圧縮できるといった利点があると考えられる。一方で、「オフライン」の決済を考慮した場合、エンドユーザの手許で分割されていくトークンの真正性を確保する方法などについて、さらに検討する必要がある。

（補論図表１－２）変動額面方式における A から B への 10,000 円の移転

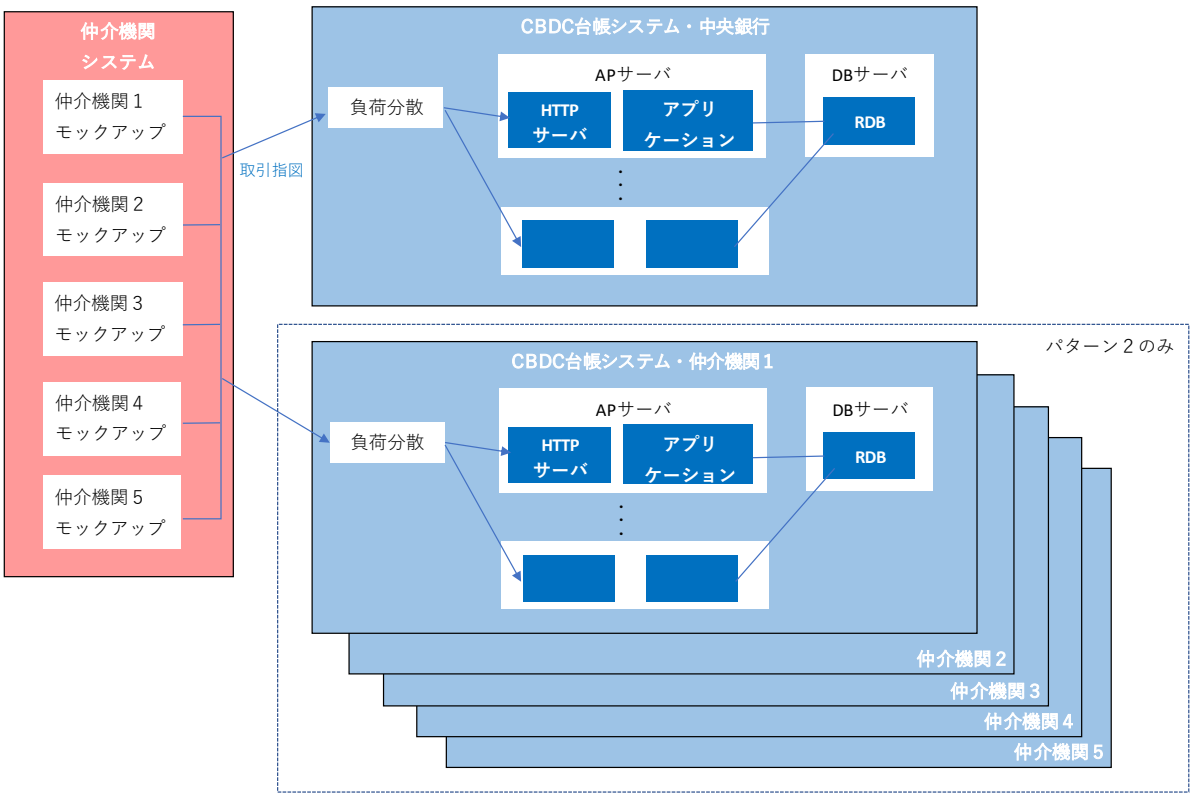
トークンID	金額（円）	保有者ID	保有者 （台帳に記録されない）
25B48BA…	40,000	88-228-504	エンドユーザA
3EF3520…	5,000	41-923-016	エンドユーザB
：	：	：	：

トークンID	金額（円）	保有者ID	保有者 （台帳に記録されない）
~~25B48BA…~~	~~40,000~~	~~88-228-504~~	~~エンドユーザA~~
3EF3520…	5,000	41-923-016	エンドユーザB
1C30EC1…	10,000	41-923-016	エンドユーザB
E72974F…	30,000	88-228-504	エンドユーザA
：	：	：	：

補論２．実験環境用のシステムと処理性能の測定方法

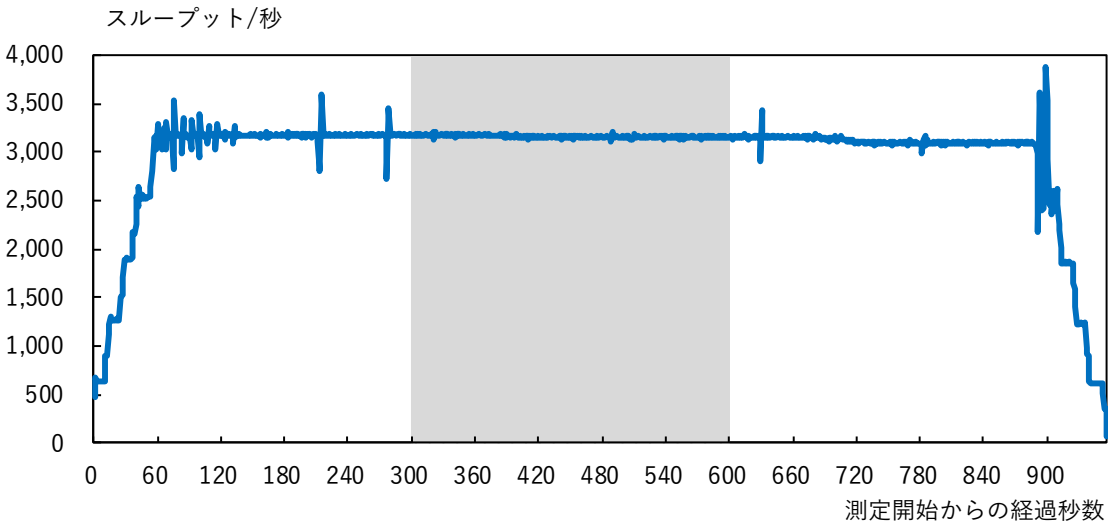
実験環境用の CBDC 台帳システムは、取引指図の処理を実行するための AP サーバと、その結果を記録・保持する DB サーバ（リレーショナルデータベース）を主要構成とした。各サーバは vCPU：８コア、メモリ：64GB、ストレージ：SSD を用い、AP サーバについては、通常負荷時には２台、高負荷時にはパターンごとに AP サーバのリソースが枯渇しない台数まで増設し、DB サーバは、全てのパターンについて、通常負荷時、高負荷時ともに１台とした。パターン２では、中央銀行に加えて、各仲介機関も同様の CBDC 台帳システムを管理する構成とした。仲介機関システムは、台帳システムの外側から取引指図を投入するだけのモックアップとした。

（補論図表２－１）実験環境の構成図



実機検証における性能テストでは、通常負荷シナリオ、高負荷シナリオともに、取引指図（秒間 500 件、3,000 件）を 15 分間以上投入し続け、安定した計測値が取得できる開始後５分から 10 分の間を評価対象時間とした。

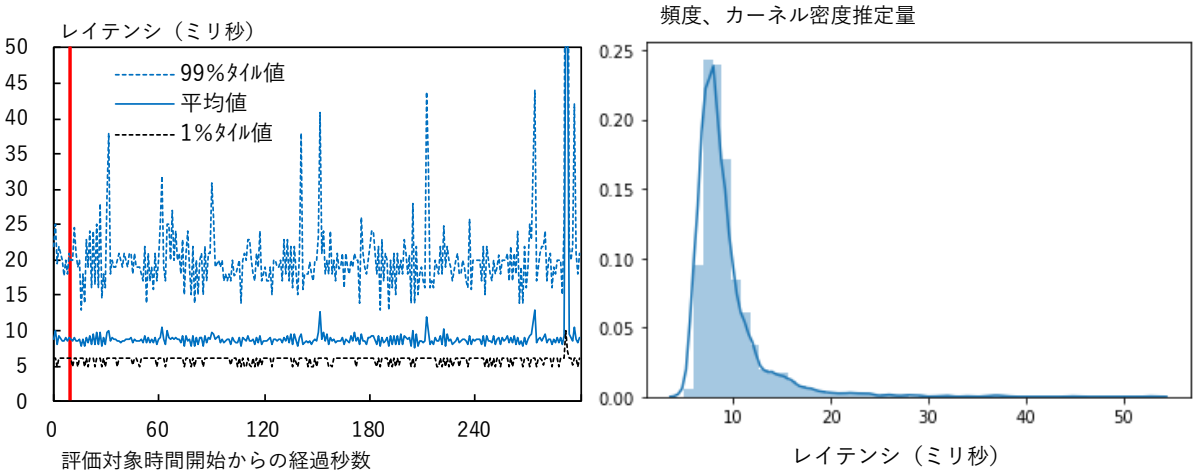
（補論図表２－２）スループット：高負荷シナリオ・パターン１



（注）シャドーは評価対象時間。

取引指図１件の処理時間であるレイテンシは、一般的に、平均値付近に実測値の大部分が集中するものの、まれに処理時間が非常に長くなる事象が生じるロングテール分布となる。このため、評価対象時間内の平均値に加え、１％タイル値と 99％タイル値を計測した。

（補論図表２－３）レイテンシ：高負荷シナリオ・パターン１

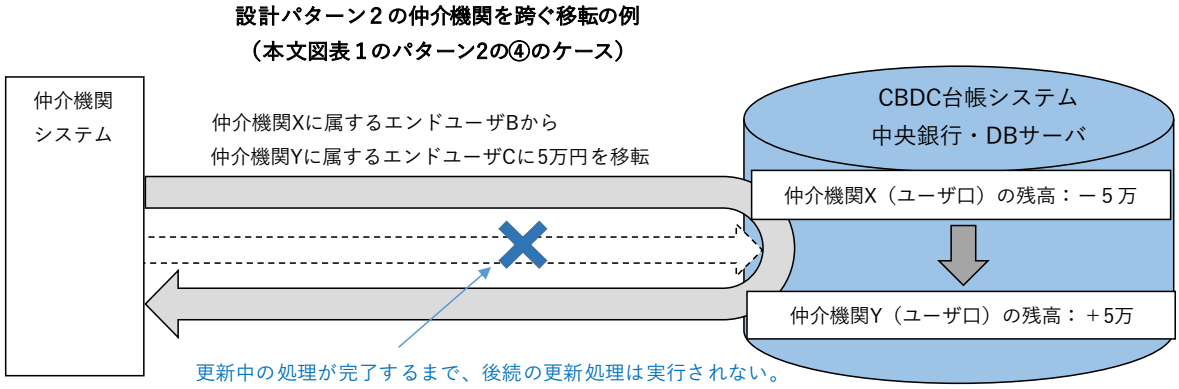


（注）右図は評価対象時間開始 20 秒後（左図の赤線）時点の取引指図件数分のレイテンシを表す。

補論３．レコードロックの影響

パターン１や２のような口座型 CBDC 台帳システムにおいては、口座残高データ（レコード）を更新する複数の取引指図が行われた場合、先着の処理が完了するまで後続の処理が実行されないよう、処理中のレコード、すなわち送金者の口座残高データと受領者の口座残高データがロックされる（レコードロック）。仮にこうした機能がなければ、複数の DB 更新処理が同時に行われ正確な残高管理の実現が難しくなる。

（補論図表３）レコードロックのイメージ



このように、決済の順序性を担保するためにレコードロックの機能は必要であるが、一方で、同一レコードに多くの処理が集中した場合には、システムの処理性能に負の影響を及ぼす。特に設計パターン２では、中央銀行台帳における各仲介機関の口座（ユーザ口）に顧客ユーザの取引状況や残高変動が全て反映されることになるため、仲介機関を跨ぐ移転の指図等が当該口座に集中しやすく、その結果、レコードロックの解除待ちに伴う処理の遅延が発生する。

レコードロックの影響を軽減するためには、本論３．（２）で示した「レコード分割」や「業務処理フローの見直し」のほか、「トランザクション分割」も対応策として考えられる。これは、CBDC の移転指図を、送金者の口座残高の減額処理（引落）と受領者の口座残高の増額処理（入金）の２つに分割したうえで、それぞれ独立して処理するものであり、システム上の処理件数は全体として増加するものの、レコードロックに繋がる同一レコードへの処理の集中度合を緩和することができる。なお、今回の実験では、通常負荷シナリオにおいて秒間 500 件の取引指図を円滑に処理できるよう、パターン２において、あらかじめ必要最低限のレコード分割とトランザクション分割を実装している。

トランザクション分割は、レコードロックの影響を軽減する効果がある一方、データの不整合などが発生し、同一の取引指図の処理の一貫性を損なう要因となり得るため、本番環境の実現に向けては、こうしたメリット、デメリットの双方を考慮して検討を進めていく必要がある。