O-015

外部サービスと連携するマイクロサービスのトランザクション制御方法
Transaction control on microservices coordinating with external services

今木 常之†
Tsuneyuki Imaki

1. はじめに

近年、モノリシックなレガシーシステムに対する必要性が認識されているモダナイゼーションにおいては、システムを構成する機能を疎結合のマイクロサービスに分散することで、個別機能の改変や交換を容易化し、業務要件の変更・拡張に際して迅速に対応可能なシステムへと刷新することが一つの目的となる。特に、システムの外部に位置する外部サービスもマイクロサービスとして抽象化することで、外部も含めたビジネス環境の変化への追従性の向上を期待できる。一方で、分散化の代償としてシステム全体の整合性を確保することが困難になるため、その補償としてトランザクション制御が必要になる。一般に、マイクロサービスの分散運用においては、整合性を厳密に保証するトランザクションは前提とされないため、基幹系のような信頼性保証が必須となるシステムのモダナイゼーションに、既存の運用手法を適用することは困難であった。本稿では、外部サービスとの連携を伴うシステムを対象に、基幹系業務に資するトランザクションの整合性保証を目的とした、マイクロサービスの分散運用手法を提案する。

2. 外部サービスを含むトランザクションの課題

従来、RDBMS やキューシステムなどのデータリソースにおいては、リソースを複数利用するアプリケーションの開発を容易化する目的で、整合性を維持したまま複数リソースの更新を制御する、分散トランザクション技術が提供されている。同技術の一般的な実現形態では、トランザクションマネージャを制御主体とする二相コミットにより、厳密な整合性（ACID 特性）の維持を実現する。これに類するマイクロサービス向け分散トランザクションの設計様式として、トランザクションマネージャに相当するコーディネータによって分散制御する TCC[1]が知られている。

TCC では、外部サービスを含む各マイクロサービスが、業務処理を仮実行する Try、仮実行の結果を確定させる Confirm、および仮実行を取消す Cancel の 3 つの要求を処理できることを前提として、トランザクションの原子性（Atomicity）を担保する。TCC をサポートするマイクロサービス運用基盤の製品や OSS も幾つか存在する。但し、TCC では各マイクロサービスについて並行性制御、即ち、複数のトランザクションが同時並行で要求された場合の排他制御を前提としないため、トランザクション間の分離性（Isolation）を一般には保証できず、結果整合が前提となる。前記の各マイクロサービス運用基盤においても厳密な整合性の保証（強整合）は未サポートとされている。更に、外部サービス連携を伴うシステムを対象に想定する場合、一般に無改造で利用することが前提である外部サービスが Try 要求に対する Confirm/Cancel 要求のいずれかを具備しないケースには、そのままでは TCC は適用不可である。

†株式会社 日立製作所，Hitachi Ltd.

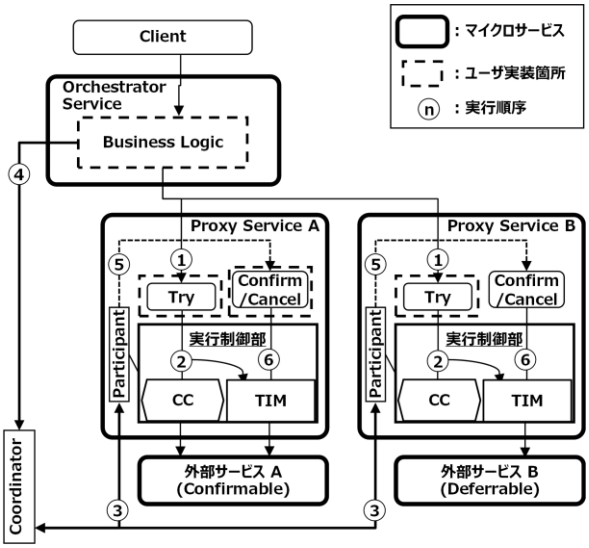

図 1　提案方式の構成

基幹系システムのマイクロサービス化においては強整合が必須、かつ銀行業務のように連携先システムの頻繁な追加や更改が想定される。本報告では、これに対応する目的で、多様な外部サービスとの連携と強整合性を担保する分散トランザクションを、効率的に構築可能とするマイクロサービス運用基盤を提案する。より具体的には、TCC をサポートする運用基盤上で動作する、以下の機能性を具備するライブラリを提供することを目的とした。

- Confirm/Cancel の両機能を具備しない外部サービスとの連携の確保
- 強整合性を担保する並行性制御の補完

3. 外部サービスのタイプ別トランザクション制御

前節に示した目的に対して提案するシステム構成の全体概要を図 1 に示す。外部サービスへの Try 処理実行のリクエスト（Try 要求）を発行することでシステム全体の業務要件を実現するユーザ実装コード（ビジネスロジック）と、各外部サービスの間に一つずつプロキシを配置し、各プロキシを TCC 運用基盤で動作するマイクロサービス（Proxy サービス）として実現する。この想定のもと、前記ビジネスロジックが動作するマイクロサービスをオーケストレータ（Orchestrator サービス）とする、オーケストレーション型のマイクロサービスシステムを構築する。

各 Proxy サービスがオーケストレータから外部サービスへの Try 要求の実行成否を捕捉し、TCC 運用基盤のコーディネータはその結果を集約して、分散トランザクションを確定か取消のいずれかに決着させる。この制御を各 Proxy サービス上で動作する「実行制御部」が、TCC 運用基盤のトランザクション参加者 API（Participant）を介して

コーディネータと連携することで実現する。実行制御部はユーザコードとリンクするライブラリであり、ユーザには、同ライブラリの API に準拠して、外部サービスへの Try 要求の発行と（必要な場合に）Confirm/Cancel 要求の発行を RESTful API で定義するコードを実装することが要求される。同ライブラリは Try 要求を表現する REST リクエスト発行をインターセプトし、トランザクション制御に係わる並行性制御（Concurrency Control：CC）および、Try 要求情報管理（Try-request Information Manager：TIM）を実行する機能を備える。後者の機能は Try 要求のパラメタ（引数）、及びその実行結果（返戻）の情報を一時保存する（これらは Confirm/Cancel 要求の発行に際して、一般に必要十分な情報である）。ライブラリがこれらの機能を担うことで、外部サービスの改造、およびユーザによるトランザクション制御の論理設計を不要とする。

ここで、外部サービスが Confirm/Cancel 要求を受付けるインタフェースを具備するか否か、及びその機能性に基づいて外部サービスを 4 つのタイプに分類し（表 1）、該タイプ別に実行制御部ライブラリの動作を切換える。タイプの使い分けは、同ライブラリの API においてユーザ実装コードが継承する抽象クラスの違いによって指定可能とした。以上を、システム全体の実行フローとして纏める。

① Orchestrator サービスが各 Proxy サービスに Try 要求を発行
② 上記①の Try 要求のパラメタを TIM に保存し、同パラメタから外部サービスが管理する対象エンティティの識別子（e.g. 口座名義）を決定し、これをロックキーとして CC にて排他制御しつつ外部サービスに Try 要求を発行し、その返戻も TIM に保存
③ 各外部サービスでの Try 要求に対する処理の成否を Participant 経由でコーディネータに集約
④ ビジネスロジック完了時オーケストレータからコーディネータにトランザクションの決着を指示
⑤ 上記③の集約結果に基づきコーディネータがトランザクションの決着先を確定か取消の何れか一方に決定し、Participant 経由で各 Proxy サービスにその実行を指示
⑥ 上記②の TIM に保存した情報に基づき（必要な場合に）Confirm/Cancel 要求を外部サービスに発行

表 1 に示した分類のうち、#1 以外は外部サービスの機能が TCC の要件を満たさない場合に相当する。本提案手法では、そのような外部サービスも、強整合を担保する分散トランザクションの一部として参加することを可能とする。例えば#2 のタイプは Try 処理を（仮ではなく）本実行するが、CC による並行性制御によって、その実行結果がトランザクション決着前に他のトランザクションから観測されることを防ぐため、分離性を担保できる。なお、#4 のタイプは Try 処理の実行が、外部サービスへの副作用を伴わない処理（e.g. 参照処理）であることを想定する。

実行制御部ライブラリのプロトタイプをマイクロサービス運用基盤 HMP-PCTO[3]上に実装し、表 1 の 4 タイプ全ての外部サービスとの連携を含む単一システムを対象としてマイクロサービス化し、実行成否両方の複数トランザクションを同時に要求して、整合性の Atomicity と Isolation が担保されることを確認した。これにより、外部サービス連携を伴う基幹系システムのモダナイゼーションにおける、本提案の有効性を確認した。

4. おわりに

本報告では、モノリシックなレガシーシステムのモダナイゼーションにおける分散トランザクションをテーマとして、特に、多様な他システムとの接続と、高い信頼性の担保が要件として想定される、基幹系システムへの適用を目的とした、マイクロサービス運用基盤技術を提案した。更に、プロトタイピングと案件適用の試行により、提案手法の実現性を確認した。

マイクロサービスの分散トランザクションの設計様式としては、TCC の他に Saga パターンも広く知られている。Saga パターンは、各サービスの処理が基本的には成功することを想定する楽観的な実行制御手法であり、いずれかのサービスの処理が失敗してトランザクションを取消に決着させる場合は、各サービスの補償トランザクションを実行する。即ち、外部サービスが表 1 に示した#2 のタイプに分類される場合に相当する。Saga パターン適用の課題と解決案[2]も参考に、ユーザの要望に従って TCC と Saga の両設計様式を利用可能とする運用基盤技術の実現が、今後の検討項目として挙げられる。

参考文献

[1] Helland、 Pat. "Life beyond distributed transactions: an apostate's opinion." Queue 14.5 (2016).
[2] 野田昌太郎、 大越淳平、 樫山俊彦、 馬場恒彦、 “マイクロサービスアーキテクチャ向け分散トランザクション管理技術の提案と評価”、 FIT2021 (2021).
[3] 日立製作所、 https://www.hitachi.co.jp/Prod/comp/soft1/hmp/ 、[Accessed: 2024/6/6].

表 1　Try 要求と関連機能に基づく外部サービス API の分類

#	外部サービス API タイプ	Try 要求に対して外部サービスが備える機能	Confirm 要求のユーザ実装	Cancel 要求のユーザ実装
1	Confirmable (確認可能型)	仮実行した Try 処理の結果を確定させるか否かを、Try 要求の後に別途要求するためのリクエストを具備する	要	要
2	Offsetable (相殺可能型)	Try 処理と逆の作用を持つ更新リクエストを具備する（e.g. 入金リクエストに対する出金リクエスト）	否	要
3	Deferrable (遅延可能型)	#1, #2 に該当しない（i.e. 一度実行した Try 処理の結果は元に戻せない）が、トランザクション中で実行する必要は無い場合（Try 要求は、決着先が確定に決定したのちに発行）	否	否
4	Irrevocable (取消不能型)	#1, #2, #3 に該当しない場合（i.e. Try 処理の結果は取消せず、トランザクション中での実行も必要。外部サービスのステートを無変更のまま参照する要求 （e.g. 残高確認）も、このタイプ）	否	否