D-016

分散合意判定機能をもつ分散トランザクション管理ミドルウェアにおける未決着トランザクションの管理方式

A Management Scheme for Inflight Transactions in Distributed Transaction Management Middleware with Distributed Consensus Judgment Functions

坂井 秀行†
Hideyuki Sakai

1. はじめに

デジタルトランスフォーメーションの潮流に伴いマイクロサービス[1]の採用が進んでおり、複数のマイクロサービスのリソース更新からなる分散トランザクションの管理が必要となっている。多様な分散トランザクション管理ミドルウェアが提供される中、分散合意判定機能により基幹系業務に適用できるような耐障害性をもつものもあるが、ネットワーク越しのトランザクションの状態を把握することは難しく、障害発生時にはロールバックを選択するケースもある。そこで本研究では上記のようなミドルウェアにおいて、障害発生時の未決着トランザクションをロールフォワードできるケースを明確化し、トランザクションの成功率を高める方式を提案する。

2. 分散トランザクション管理ミドルウェア

本研究で検討対象とする分散トランザクション管理方式である TCC[2]および分散トランザクション管理ミドルウェア[3]について説明する。

2.1 TCC(Try, Confirm, Cancel)

複数のリソースに対し、Try フェーズでは各リソースの更新予約を行い、全てのリソースの更新予約が成功した場合には、Confirm フェーズで各リソースの更新を確定する。もしひとつでも更新予約が失敗した場合には、Cancel フェーズで成功済みの更新予約を取り消す。この手続きにより、全てのリソースを更新した状態か、全てのリソースを更新しない状態のどちらかを取ることができるようになる。

2.2 分散合意判定機能をもつミドルウェア

図 1(a)に示す、オーケストレータ、プロキシ、メディエータをもつ分散トランザクション管理ミドルウェアを考える。オーケストレータはアプリケーションから業務リクエストを受信し、各プロキシに Try リクエストを送信する。各プロキシは対応する外部サービスに Try リクエストを送信し、そのレスポンスを受信する。プロキシは Try リクエストのレスポンスをオーケストレータに返すとともに、メディエータにも Try リクエストが成功であれば Commit、失敗であれば Rollback を投票する。オーケストレータは全プロキシからレスポンスを受信したらメディエータに判定の指示を出す。メディエータは各プロキシから投票された結果を元にトランザクションを Commit するか Rollback するかを判定し、各プロキシに判定結果を送信する。各プロキシは Commit を受信したら外部サービスに Confirm リクエストを送信し、Rollback を受信したら Cancel リクエストを送信する。以上のフローにより TCC 方式の分散トランザクションの管理を実現する。

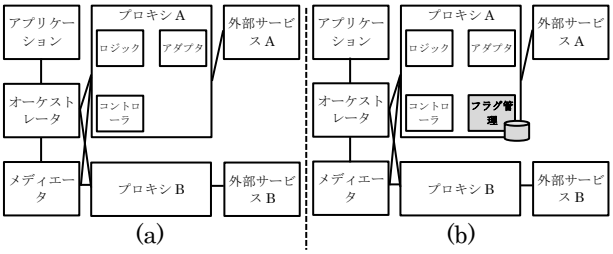

図 1　(a)分散合意判定機能をもつミドルウェア
(b)本研究で提案するミドルウェア

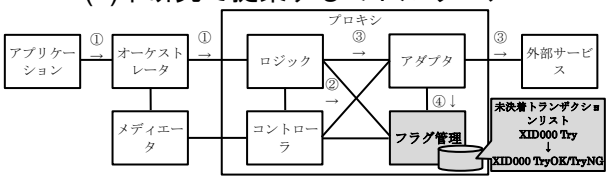

図 2　Try リクエストフェーズ

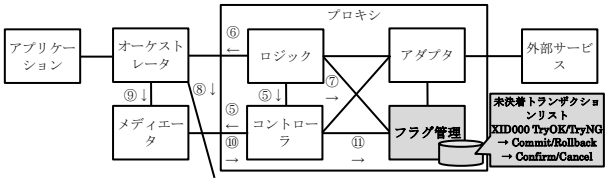

図 3　Commit/Rollback 判定フェーズ

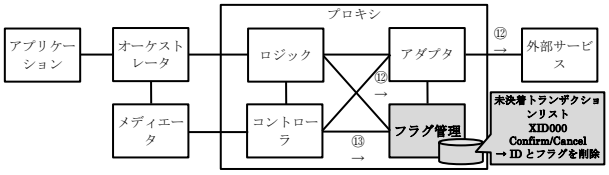

図 4　Comfirm/Cancel リクエストフェーズ

3. 未決着トランザクション管理

ミドルウェアの障害発生時にミドルウェアの再起動を行うことで、未決着トランザクションが発生する。データの整合性を保つためには、この未決着トランザクションを決着させる必要があるが、従来は全てロールバックを選択することが一般的であり、トランザクションの成功率の低下につながっていた。それに対し、以降では図 1(b)に示すようなフラグ管理機能をもつミドルウェアを考える。

3.1 TCC フローのフェーズ詳細化

本研究で考えるミドルウェアのプロキシはオーケストレータと通信するロジック部、メディエータと通信するコントローラ部、外部サービスと通信するアダプタ部、未決着トランザクションに TCC フローの進度に応じたフラグを

†株式会社日立製作所　Hitachi, Ltd.

付与するフラグ管理部からなるとする。この構成における TCC フローについて説明する。

3.1.1 Try リクエストフェーズ(図 2)

① アプリケーションからオーケストレータに業務リクエストを送信。オーケストレータからロジック部に Try リクエストを送信。トランザクション ID(XID000)付与。
② ロジック部からフラグ管理部に XID000 を登録。その ID に対してフラグとして Try を登録。
③ ロジック部からアダプタ部経由で外部サービスに Try リクエストを送信。レスポンスとして OK/NG を受信。
④ アダプタ部からフラグ管理部に、Try リクエストが OK であればフラグとして TryOK を、NG であれば TryNG を、XID000 に対して登録。

3.1.2 Commit/Rollback 判定フェーズ(図 3)

⑤ ロジック部から Try リクエストの結果をコントローラ部に通知。コントローラ部はメディエータに結果が OK であれば Commit を、NG であれば Rollback を投票。
⑥ ロジック部はオーケストレータに Try リクエストのレスポンスを送信。
⑦ ロジック部からフラグ管理部に、Try リクエストが OK であればフラグとして Commit を、NG であれば Rollback を、XID000 に対して登録。
⑧ オーケストレータは次のプロキシのロジック部に Try リクエストを送信。以降、各プロキシに対して①～⑦を実施。
⑨ オーケストレータは、全てのプロキシから Try リクエストのレスポンスを受信したら、メディエータに分散合意判定実施の指示を通知。
⑩ メディエータは各プロキシからの投票内容を元にトランザクションの決着処理を判定。判定結果をコントローラ部に送信。
⑪ コントローラ部からフラグ管理部に、判定結果が Commit であればフラグとして Confirm を、Rollback であれば Cancel を、XID000 に対して登録。

3.1.3 Confirm/Cancel リクエストフェーズ(図 4)

⑫ コントローラ部からアダプタ部経由で外部サービスに、判定結果に応じて Confirm リクエストまたは Cancel リクエストを送信。
⑬ コントローラ部からフラグ管理部に、決着したトランザクションの ID とそれに対応したフラグの削除の指示を通知。

3.2 未決着トランザクションのリカバリ

ミドルウェア再起動時のリカバリフローを説明する。初期化処理としてコントローラ部は未決着トランザクションリストに管理されている全ての未決着トランザクションの ID とフラグを取得する。各未決着トランザクションをそのフラグに応じたリカバリフローに従って決着させる。

3.2.1 Try/TryOK/TryNG フラグに対するリカバリ(図 5)

Try、TryOK、TryNG のフラグを付与されたトランザクションは、ミドルウェアの障害発生時には、オーケストレータに Try リクエストのレスポンスを送信することができずにタイムアウトが発生するため、失敗という扱いとなる。

⑭ コントローラ部は未決着トランザクションリストに管理されている全ての未決着トランザクションの ID とフラグを取得。この処理は以降で説明する Commit/Rollback フラグのケースと Confirm/Cancel フラグのケースでも共通に行う初期化処理となる。
⑮ コントローラ部からフラグ管理部に、Try、TryOK、TryNG の付与された未決着トランザクションの ID とフラグの削除の指示を通知。

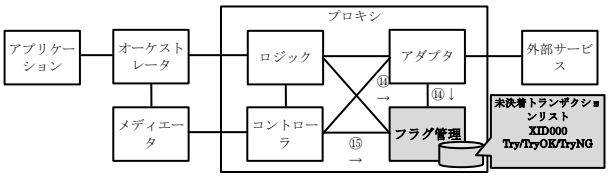

図 5　Try/TryOK/TryNG フラグに対するリカバリ

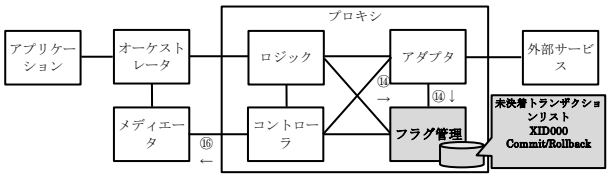

図 6　Commit/Rollback フラグに対するリカバリ

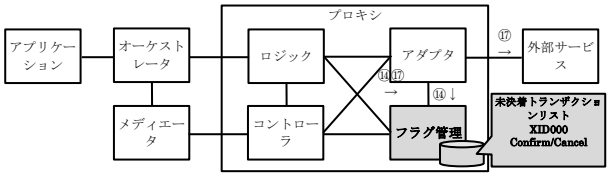

図 7　Confirm/Cancel フラグに対するリカバリ

3.2.2 Commit/Rollback フラグに対するリカバリ(図 6)

Commit、Rollback のフラグを付与されたトランザクションは、ミドルウェアの障害発生時には、メディエータに再度投票をすることができる。Commit フラグの場合は再度 Commit 投票を行うため、メディエータの判定結果が Commit になる可能性が生じ、従来はロールバックしていたトランザクションをロールフォワード(Commit)できる可能性が高まることになる。

⑯ コントローラ部はメディエータにフラグが Commit であれば Commit を、Rollback であれば Rollback を再投票。メディエータは再判定を実施(⑩～⑬を実施)。

3.2.3 Confirm/Cancel フラグに対するリカバリ(図 7)

Confirm、Cancel のフラグを付与されたトランザクションは、ミドルウェアの障害発生時には、再度 Comfirm または Cancel のリクエストを送信できる。

⑰ コントローラ部からアダプタ部経由で外部サービスに、フラグに応じて Confirm リクエストまたは Cancel リクエストを送信(⑫⑬を実施)。

4. おわりに

本方式によるトランザクション成功率の向上について、具体業務を通じて評価を行いたい。

参考文献

[1] Sam Newman, “マイクロサービスアーキテクチャ”, O’REILLY (2016).
[2] Pat Helland, “Life Beyond Distributed Transactions: An apostate’s opinion”, ACM Queue, Vol.14, Issue 5, pp.69–98 (2016).
[3] 西谷 淳平, “分散合意を用いたクラウドネイティブトランザクション Paxos Commit”, CloudNative Days Tokyo 2022 (2022).