PostgreSQL インサイド

論理レプリケーションにおける通信の改善

PostgreSQL 15 でコミットされた機能の先行紹介

技術者 Blog

Wang Wei
南京富士通南大軟件技術有限公司
開発事業本部デジタルソリューション事業部

はじめに

このブログでは、論理レプリケーションの通信における次の 2 つの問題を解決するために、PostgreSQL 15 で実施された改善について説明します。

- トランザクション内のすべての DML がパブリケーションのフィルター条件に従ってパブリッシュされない場合、walsender は空のトランザクションを送信します。これにより、CPU / メモリー / ネットワークなどのリソースが無駄になります。
- 大規模なトランザクションを処理する際、walsender がトランザクション内のパブリッシュされない DML の処理でビジー状態の場合、長時間にわたって walreceiver と通信できなくなることがあります。これにより、walsender が正しく動作していても、予期せぬタイムアウトエラーが発生することがあります。

論理レプリケーションにおける通信の仕組み

まずは、論理レプリケーションにおける通信に関する 2 つの概念、「通信メッセージの種類」と「フィルタリング」について簡単に紹介します。

通信メッセージの種類

論理レプリケーションでは、walsender から walreceiver に送信されるメッセージには 2 種類あります。

- **キープアライブ（keepalive）メッセージ**

  このメッセージは、walsender が正しく動作していることを walreceiver に伝えるために使用されます。
  ユーザーは、wal_receiver_timeout パラメーター（デフォルトは 60 秒）を使って、walreceiver のタイムアウトの時間を設定できます。wal_receiver_timeout の時間内に、walreceiver が walsender からいずれの種類のメッセージも受信しない場合、タイムアウトエラーで終了します。

- **論理レプリケーションのプロトコルメッセージ**

  論理レプリケーションのプロトコルメッセージには多くの種類がありますが、このブログではそのうちの 2 つだけを取り上げます。

  - INSERT、UPDATE、DELETE、および TRUNCATE を含む DML メッセージ
  - BEGIN メッセージや COMMIT メッセージなどのトランザクションの開始と終了を定義するメッセージ

参考

論理レプリケーションの全プロトコル一覧は、PostgreSQL の Web サイトで確認できます。

- 55.5. Logical Streaming Replication Protocol（PostgreSQL オフィシャルのページへ）
  https://www.postgresql.org/docs/15/protocol-logical-replication.html

フィルタリング

トランザクション内のすべての DML が walreceiver に送信されるわけではありません。これは、パブリケーションの作成時に、テーブル、行、または操作の種類に対してフィルターを指定できるためです。フィルター条件を満たすと、一部の DML は送信されません。なお、PostgreSQL 15 の行フィルターについては、「論理レプリケーションにおけるパブリケーションの行フィルター」というブログ記事を参照してください。

改善の概要

では、このブログの冒頭で述べた 2 つの問題点がどのように改善・修正されたかをご紹介します。

空のトランザクションに対する改善

トランザクション内のすべての DML がデコード時にフィルタリング（除外）される場合、このトランザクションを「空のトランザクション」と呼びます。PostgreSQL 14 以前では、標準のロジカルデコードプラグインである pgoutput（PostgreSQL にデフォルトで使用されている論理レプリケーションプラグイン）は、すべてのトランザクションを walreceiver に送信していました。空のトランザクションの場合でも、DML 関連のメッセージは送信されませんが、トランザクションの開始と終了を定義するメッセージは依然として送信されました（図 1 参照）。このような空のトランザクションを構築・送信することは、CPU サイクルとネットワーク帯域幅の無駄遣いでした。

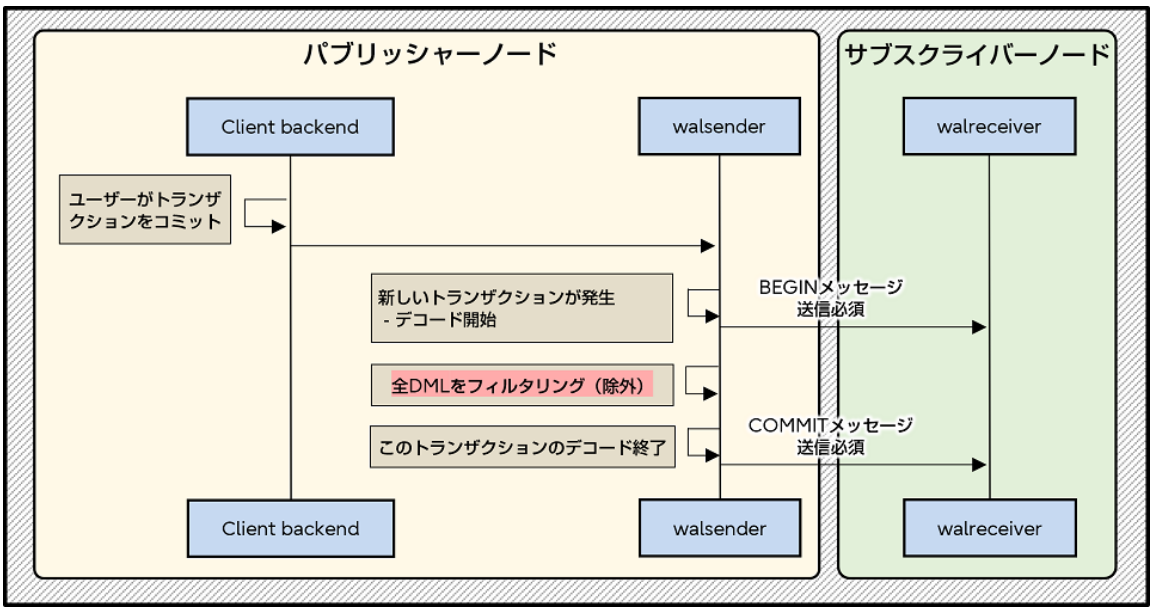

図 1：PostgreSQL 14 以前における空のトランザクションの扱い

この問題を解決するために、最初の DML メッセージが送信されるまで BEGIN メッセージを延期するように walsender を改善しました。デコード終了時に、BEGIN メッセージが送信されていなければ、COMMIT メッセージも送信されないようにしました。詳細な開発情報については、GitHub を参照してください。

- Skip empty transactions for logical replication（GitHub のページへ）
  https://github.com/postgres/postgres/commit/d5a9d86d8f

また、空でないトランザクションについても、メッセージを送信するタイミングが変更されます。この PostgreSQL 14 と PostgreSQL 15 の違いを例で見てみましょう。次のようなトランザクション T1 があるとします。

```
postgres=# BEGIN;
BEGIN
```



```
postgres=*# INSERT INTO tab_not_publish VALUES (1);  -- このDMLはフィルタリング（除外）される
INSERT 0 1
postgres=*# INSERT INTO tab_publish VALUES (1);  -- このDMLはパブリッシュされる
INSERT 0 1
postgres=*# COMMIT;
COMMIT
```

BEGIN メッセージと COMMIT メッセージの送信タイミングを、次の図 2 に示します。

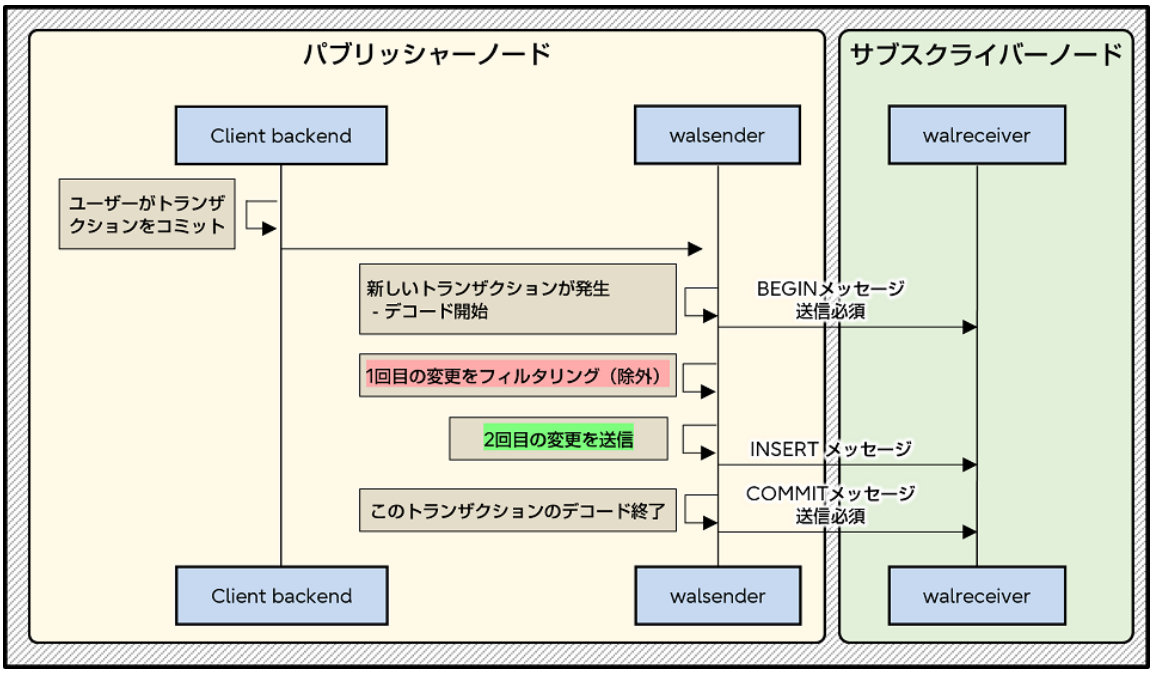

図 2：BEGIN メッセージと COMMIT メッセージの送信タイミング - 修正前

図 2 からわかるように、修正前は walsender から walreceiver に送信できるのは、テーブル tab_publish の INSERT メッセージだけです。
修正後の BEGIN メッセージと COMMIT メッセージの送信タイミングを、次の図 3 に示します。

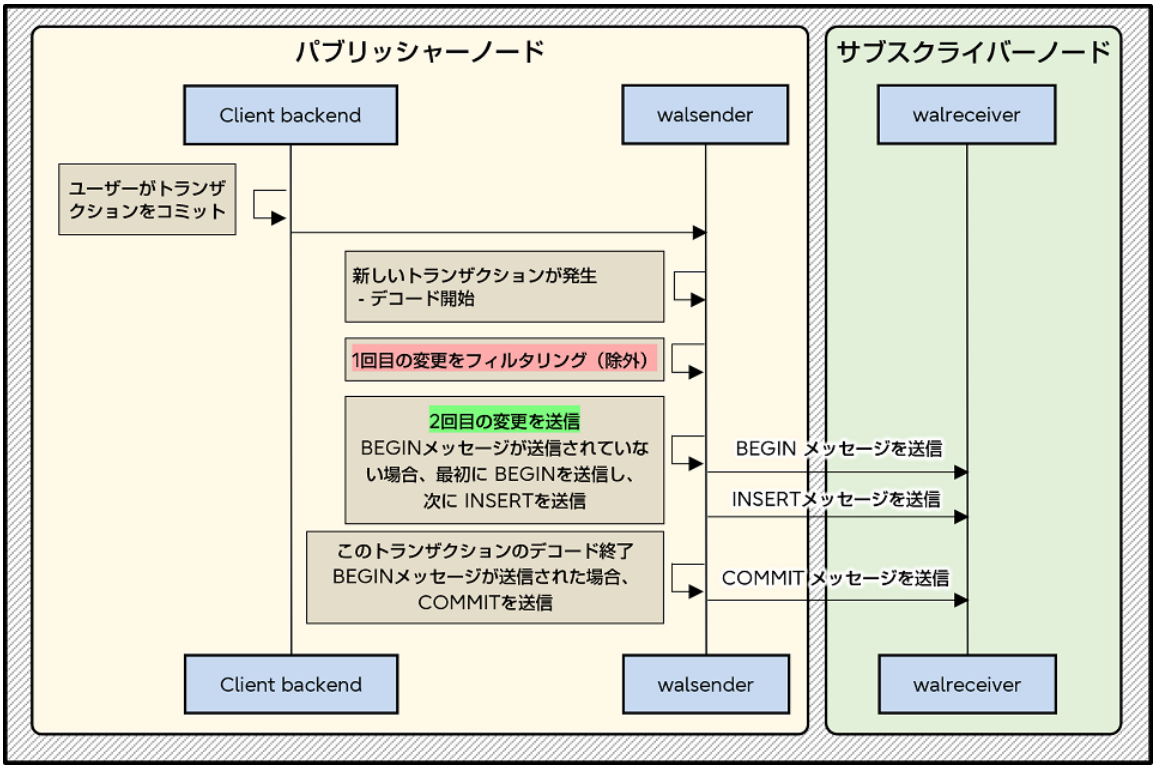

図 3：BEGIN メッセージと COMMIT メッセージの送信タイミング - 修正後

図 3 からわかるように、修正後では、もしトランザクション内のすべての DML がフィルタリング（除外）されると、BEGIN メッセージと COMMIT メッセージは送信されません。このようにして、walsender は空のトランザクションの送信をスキップできます。

ただし、PostgreSQL 14 以前の同期論理レプリケーションでは、データが同期されたことを確認するために、walreceiver は空のトランザクションの COMMIT メッセージを受信すると、ローカルデータを同期し、walsender にフィードバックメッセージを送信しています。

補足　同期論理レプリケーションについては、PostgreSQL オフィシャルのページをご覧ください。

- 27.2.8. Synchronous Replication（PostgreSQL オフィシャルのページへ）
  https://www.postgresql.org/docs/15/warm-standby.html#SYNCHRONOUS-REPLICATION

walsender は、walreceiver からフィードバックメッセージを受け取った後のみ処理を続行し、それ以外の場合、walsender はトランザクションをコミットするために Client backend をブロックします。
そこで、PostgreSQL 15 では、walsender は同期論理レプリケーションで空のトランザクションをスキップした後、walreceiver にキープアライブメッセージを送信し、フィードバックメッセージを要求するようにしました。そして、walreceiver はデータを同期し、このメッセージに従って walsender にフィードバックメッセージを送信します。このようにして、同期論理レプリケーションでは、空のトランザクションによって応答が遅延する事態を回避できます。次の図 4 と図 5 は、同期論理レプリケーションにおける通信の違いを示しています。

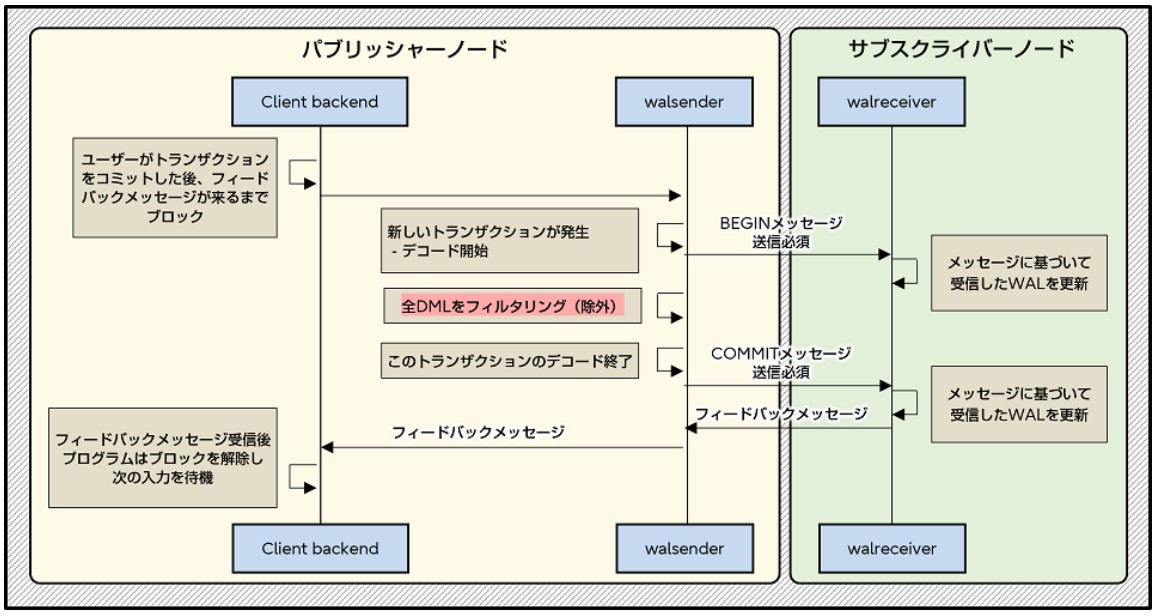

図 4：同期論理レプリケーションにおける通信 - 修正前

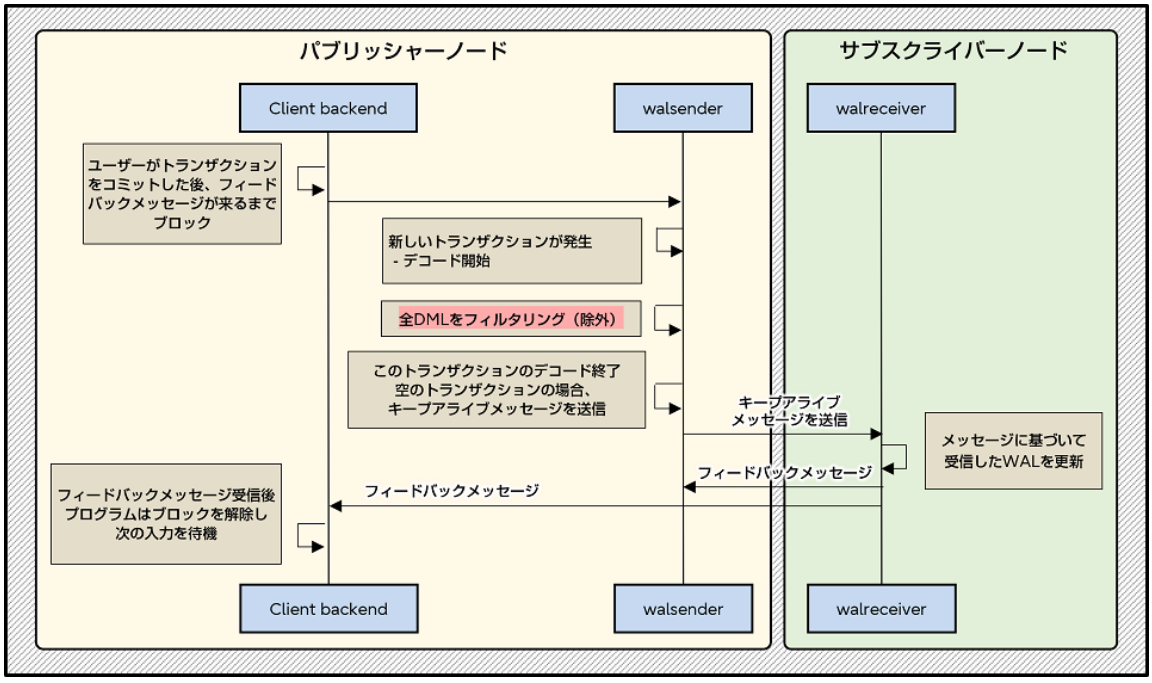

図 5：同期論理レプリケーションにおける通信 - 修正後

予期せぬタイムアウトエラーの修正

PostgreSQL 14 以前では、トランザクションにパブリッシュされない連続した DML が多数ある場合、walsender はこれらのパブリッシュされない DML のデコード処理でビジー状態になるため、長時間にわたって walreceiver と通信できなくなります。この場合、walsender が正しく動作していても、walreceiver は指定されたタイムアウト内に walsender からメッセージを受信しないため、walreceiver に予期せぬタイムアウトエラーが発生する原因となっていました。

PostgreSQL 15 では、このエラーを回避するために、walsender は walreceiver と定期的に通信し続けます。そのため、walsender は DML の処理数が特定の閾値に達すると（それらの DML がパブリッシュされているかどうかに関わらず）、必要に応じて通信を維持するために walreceiver にキープアライブメッセージを送信しようと試みます。この開発に関する詳細な情報については、GitHub で参照できます。

- Fix the logical replication timeout during large transactions（GitHub のページへ）
  https://github.com/postgres/postgres/commit/f95d53eded

次のようなたくさんの DML があり、どれもパブリッシュされないトランザクションがあったとします。次の図 6 と図 7 でわかるように、トランザクションが処理されるときの、walsender と walreceiver の間の通信が変更されました。

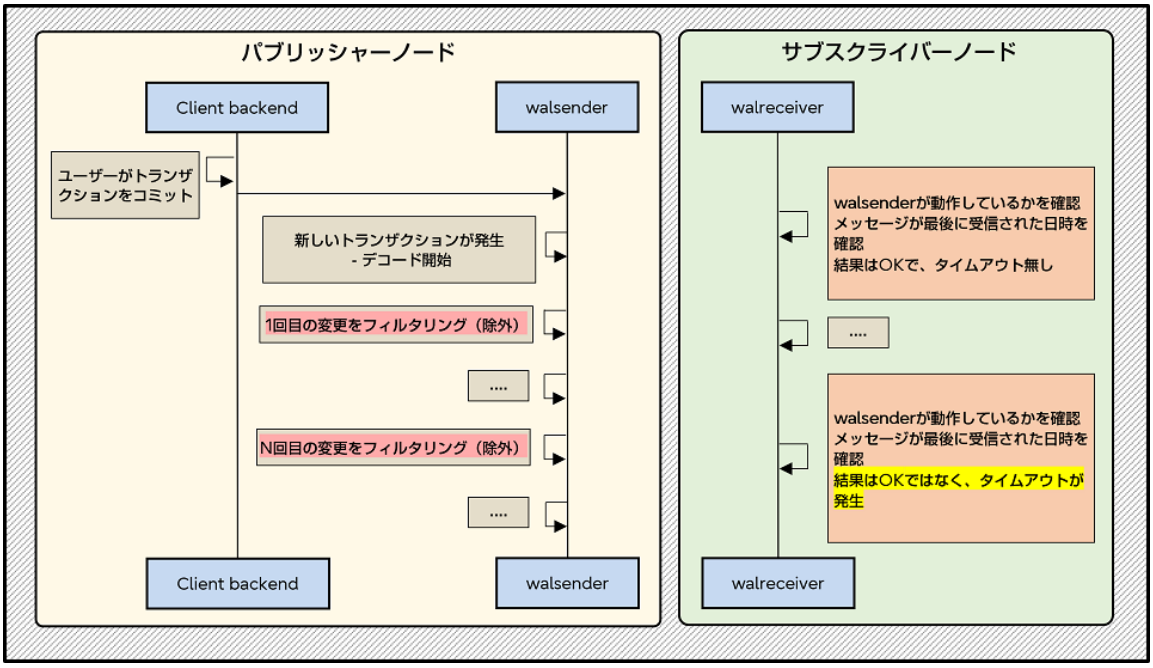

図 6：walsender と walreceiver の間の通信 - 修正前

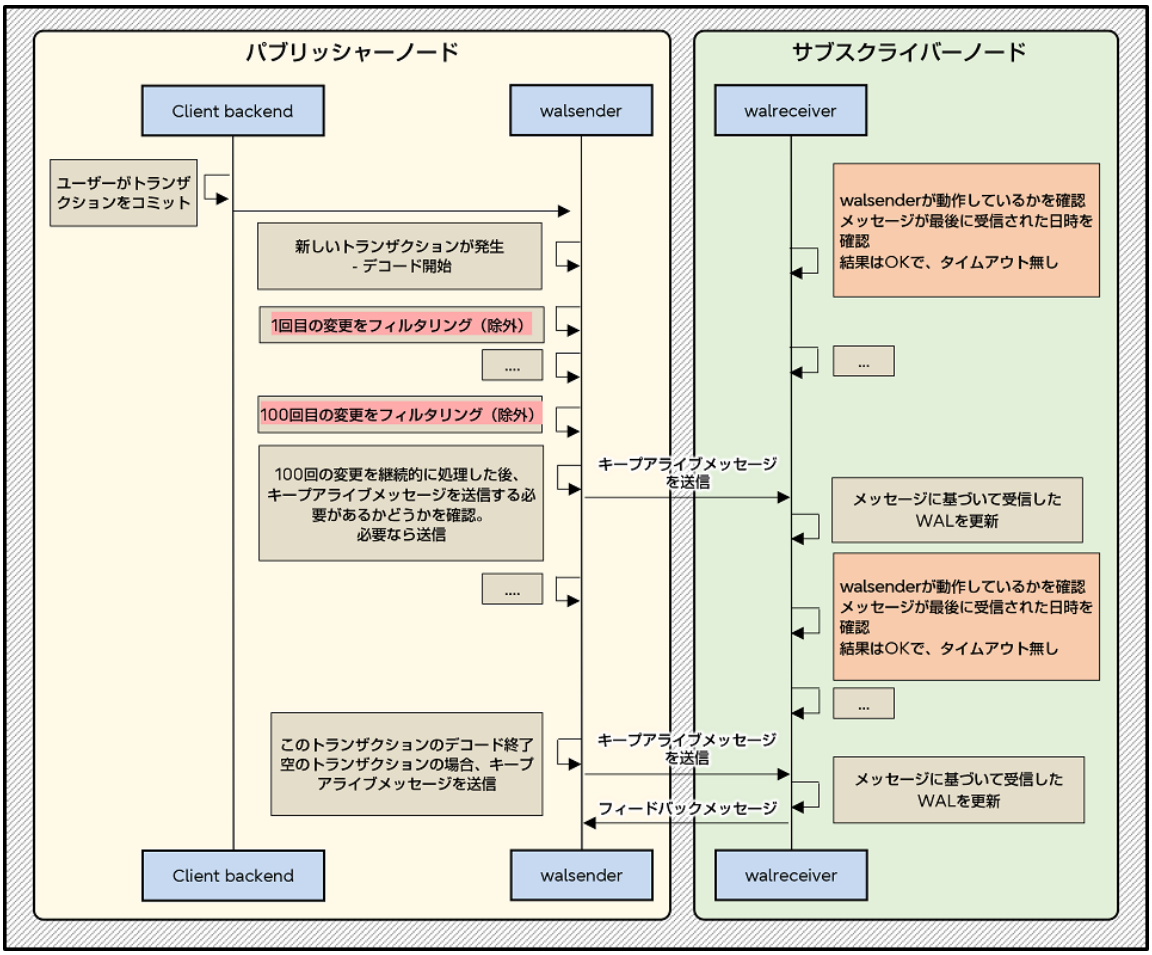

図 7：walsender と walreceiver の間の通信 - 修正後

上記からわかるように、PostgreSQL カーネルでは閾値を 100 に設定しています（性能テストの結果、この閾値によってこのタイムアウトエラーを解決でき、性能が低下しないことが確認されています）。こうすることで、walsender が正しく動作している場合、walreceiver に予期しないタイムアウトエラーは発生しません。

補足　閾値については、PostgreSQL オフィシャルのページをご覧ください。

- RE: Logical replication timeout problem（PostgreSQL オフィシャルのページへ）
  https://www.postgresql.org/message-id/OS3PR01MB6275C67F14954E05CE5D04399E139%40OS3PR01MB6275.jpnprd01.prod.outlook.com

性能への影響

最後に、空のトランザクションに対する改善に関する性能テストの結果を共有します。
前述のとおり、空のトランザクションの処理を改善した結果、walsender は BEGIN と COMMIT メッセージのみを含む空のトランザクションについては walreceiver に送信しなくなりました。これにより、ネットワークトラフィックが減少し、性能が向上します。私のテストでは、改善後、空のトランザクションをデコードする場合、非同期論理レプリケーションでは walsender の送信量は 97 バイト減少しました。また、同期論理レプリケーションでは walsender がキープアライブメッセージを追加送信するものの、合計送信量は 79 バイト減少しました。次に、ネットワーク帯域消費における性能（パフォーマンス）の向上について、テスト結果を通して見てみましょう。

walsender が送信するトランザクションに占める空のトランザクションの割合がテスト結果に影響するため、空のトランザクションの割合を 5 段階に分けてテストしました。また、改善後の同期論理レプリケーションでは、walsender が空のトランザクションをスキップした場合、キープアライブメッセージを追加送信するため、同期論理レプリケーションと非同期論理レプリケーションの両方でテストを実施しました。次の図 8 と図 9 に示すように、非同期論理レプリケーションと同期論理レプリケーションの両方で性能が向上していることがわかります（横軸は送信トランザクションに占める空のトランザクションの割合を表します。縦軸は walsender から walreceiver に転送されたデータの総量をバイトで表しています。）

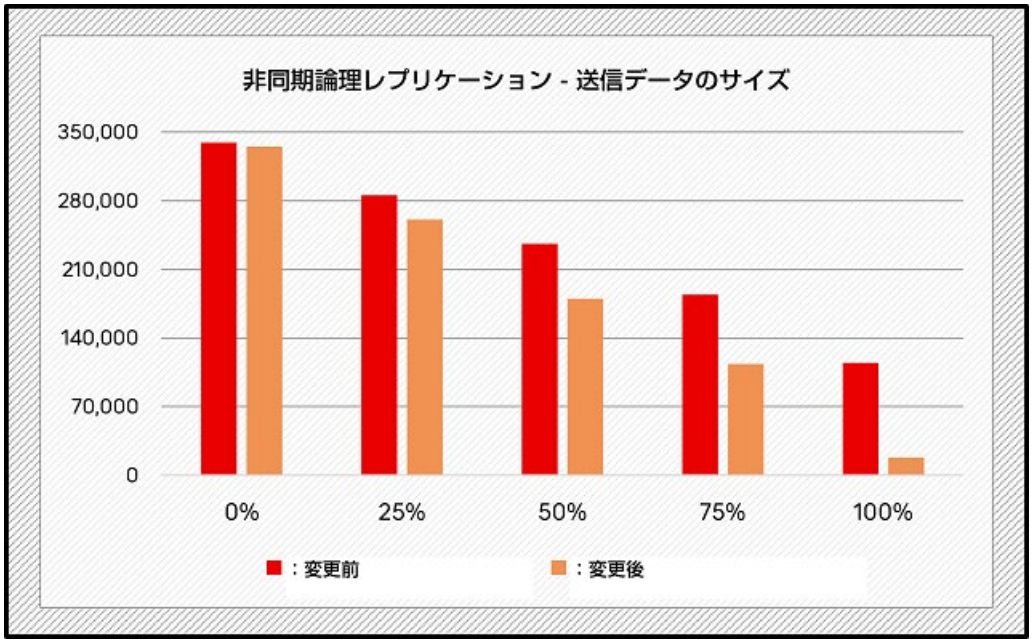

図 8：性能比較 - 非同期論理レプリケーション

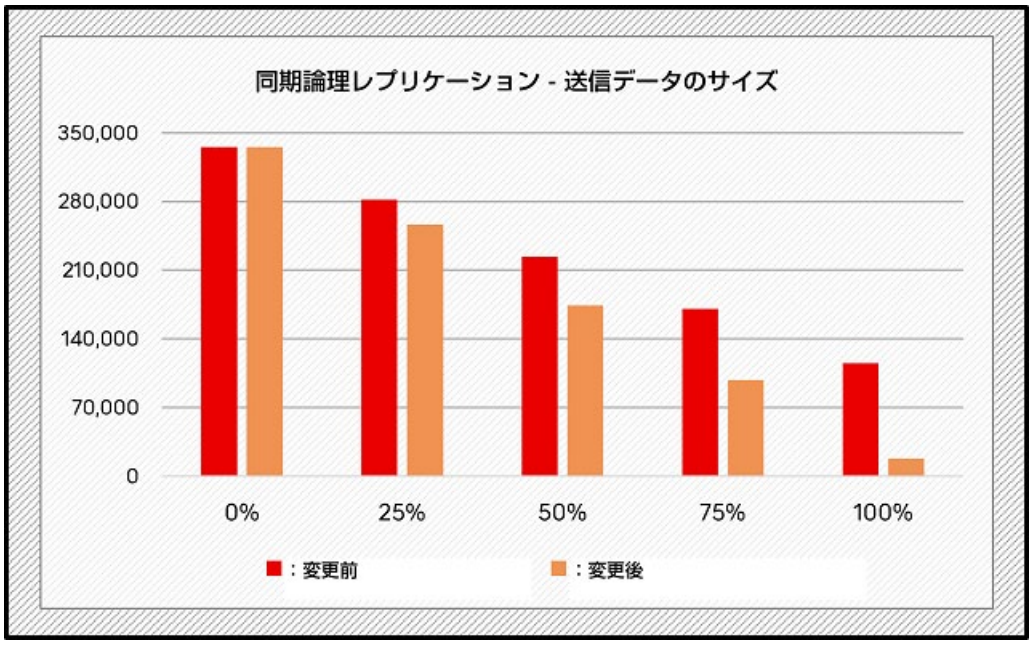

図 9：性能比較 - 同期論理レプリケーション

テスト結果から、空のトランザクションの割合が高いほど、改善効果は大きいことがわかります。空のトランザクションの割合が 25%、50%、75%、100%の場合、ネットワーク通信量はそれぞれ約 8%、22%、40%、84%削減されました。また、空のトランザクションがない場合、性能の劣化はありません。

今後に向けて

今回は、空のトランザクションのスキップや、予期せぬタイムアウトエラーの修正による論理レプリケーションの性能・機能の向上について紹介しました。しかし、空のトランザクションをスキップするメカニズムには、まだいくつかの制限があります。例えば、空のトランザクションは二相コミットではスキップされません。これは、トランザクションを準備してからプリペアドトランザクションをコミットするまでの間に walsender が再起動した場合、プリペアドトランザクションがコミット時にスキップ

されたかどうかが明確にわからないためです。進行中のトランザクションのストリーミング（subscription_parameter のパラメーター"streaming"で設定）についても、同様の問題により改善されていません。近いうちに、上記 2 種類のトランザクションのうち、空のトランザクションの処理について、より良い方法での改善を試みる可能性があります。
さらに、walreceiver における進行中のトランザクションのストリーミング適用の効率を改善するために、私たちのチームは walreceiver におけるトランザクションの並列適用に関するパッチを共有し、コミュニティーで活発に議論して、継続的に改善しています。

補足　議論の状況については、PostgreSQL オフィシャルのページをご覧ください。

- Perform streaming logical transactions by background workers and parallel apply（PostgreSQL オフィシャルのページへ）
  https://www.postgresql.org/message-id/flat/CAA4eK1%2BwyN6zpaHUkCLorEWNx75MG0xhMwcFhvjqm2KURZEAGw%40mail.gmail.com

2022 年 10 月 7 日