スタンバイ側の参照処理で発生したコンフリクトを解決したい

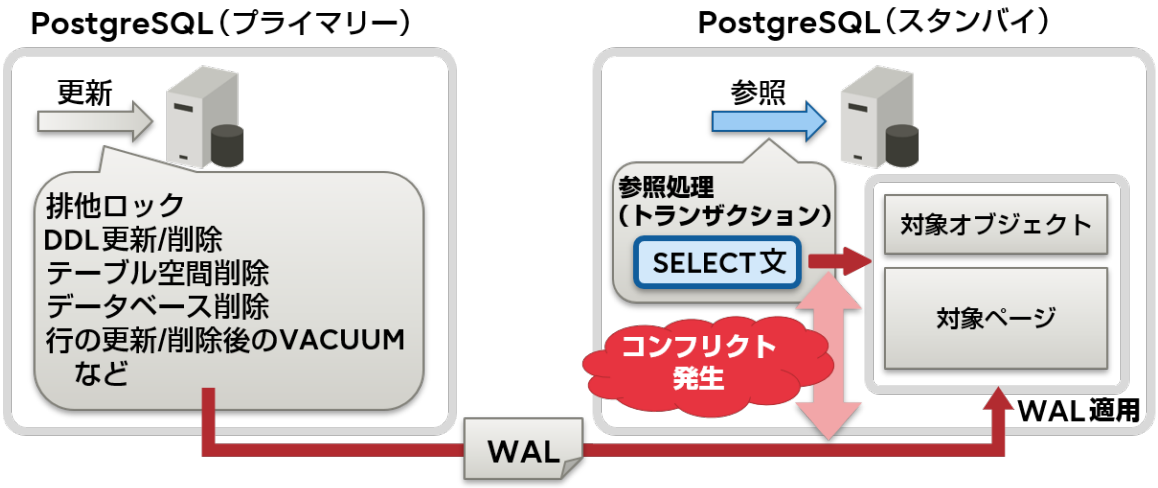
FUJITSU

技術を知る

- [x] 導入／環境設定
- [ ] 移行
- [ ] 性能
- [ ] チューニング
- [ ] バックアップ／リカバリー
- [x] 冗長化／負荷分散
- [ ] 監視
- [ ] データ連携
- [ ] 災害対策
- [x] 豆知識

実現方法

ストリーミングレプリケーション構成において、スタンバイ側で参照処理（SELECT 文による問い合わせの実行）を行う場合、プライマリー側で行われた更新処理や VACUUM 処理などによる WAL（トランザクションログ）をスタンバイ側に適用する処理が同一資源（注 1）にアクセスすることで、コンフリクト（競合）が発生することがあります。



- 注 1　アクセス対象のオブジェクト（テーブルなど）や、そのオブジェクトが格納されているページ

その際、以下のどちらかのエラーメッセージが出力され、スタンバイ側の参照処理がキャンセルされます。

エラーメッセージ

エラーによりコネクションを切断した場合のメッセージ

```
FATAL: terminating connection due to conflict with recovery
```

エラーによりクエリのみキャンセルした場合のメッセージ

```
ERROR: canceling statement due to conflict with recovery
```

PostgreSQL には、コンフリクトが発生して一定時間（デフォルトで 30 秒）が経過すると、同一資源をアクセスしている参照処理をキャンセルさせて、WAL の適用処理を優先する仕組みがあります。そのため、コンフリクトの発生を完全に無くすことは非常に難しいですが、以下の対処により解決へと近づけることができます。

- A）参照処理において、リトライ処理を入れてキャンセルされた処理を再実行する
- B）参照処理において、トランザクション処理を分割するなどして 1 回あたりの実行時間を短くする
- C）PostgreSQL の設定パラメーターを調整してコンフリクトの発生を緩和する

対処例

ここでは、設定ファイル postgresql.conf のパラメーターの調整（上記の C の対処）により、コンフリクトの発生を緩和する方法について説明します。
初めに、プライマリー側のどの処理が原因でコンフリクトが発生したのかを特定します。原因となる主な処理は以下のとおりです。詳細は、ポイントの 1 つめ、および、2 つめの項目を参考にしてください。

- 明示的な排他ロック
- DDL による更新 / 削除
- テーブル空間の削除
- データベースの削除
- 行データの更新 / 削除後の VACUUM 処理（HOT 処理も含む）：通常このケースが多い

次に、コンフリクトを緩和するための 2 種類の方法を示します。コンフリクトの原因に応じて、必要な対処を行ってください。

スタンバイ側での WAL 適用の待ち時間を延長する

この方法は、コンフリクトを発生させるすべての原因において有効な対処です。参照処理（トランザクション）の実行に必要な時間がある程度わかっている場合には、WAL 適用の待ち時間を長くすることで、参照処理がキャンセルされるまでの時間を延長します。
そのためには、スタンバイ側の設定ファイル postgresql.conf の以下のパラメーターを調整します。

max_standby_streaming_delay

ストリーミングレプリケーションから WAL データを受け取って適用する処理が待たされた際、スタンバイサーバーの参照処理をキャンセルするまでの待ち時間を設定します（デフォルトは 30 秒）。

max_standby_archive_delay

WAL アーカイブから WAL データを読み込んで適用する処理が待たされた際、スタンバイサーバーの参照処理をキャンセルするまでの待ち時間を設定します（デフォルトは 30 秒）。

【例】WAL 適用の待ち時間を、デフォルトの 30 秒から 60 秒に変更します。

```
max_standby_streaming_delay = 60s
max_standby_archive_delay = 60s
```

ただし、スタンバイ側でコンフリクトが発生したときの WAL 適用の開始が遅れることで、以下の影響があります。そのため、システムの要件に合わせて設定値を調整する必要があります。

- プライマリー側とスタンバイ側でデータの状態が乖離する
- スタンバイ側で適用すべき WAL が蓄積されることで、ディスクの圧迫やフェイルオーバーの時間が長くなる

プライマリー側の VACUUM 処理を延期させる

この方法は、プライマリー側の VACUUM 処理に起因するコンフリクトを緩和します。特に、テーブルデータの更新頻度が高い場合に効果があります。VACUUM 処理を遅らせるためのパラメーターは次の 2 種類があります。

vacuum_defer_cleanup_age

プライマリー側の設定ファイル postgresql.conf のパラメーターです。プライマリーサーバーにおいて、VACUUM および HOT 更新が不要になった行データを回収する際、指定されたトランザクションの数だけ遅延させます（デフォルトは 0）。

【例 1】行データの VACUUM 処理を 20 トランザクション分、遅らせるよう設定します。

```
vacuum_defer_cleanup_age = 20
hot_standby_feedback
```

スタンバイ側の設定ファイル postgresql.conf のパラメーターです。on に設定することで、スタンバイ側で現在処理を行っているトランザクションの参照処理に関する情報が、プライマリー側（または上位サーバー）にフィードバックされます。すると、プライマリー側において、スタンバイ側でアクセス中の行データについての VACUUM 処理を延期させることができます。これにより、コンフリクトの原因となる WAL の転送がすぐに行われなくなり、コンフリクトが緩和されます（デフォルトは off）。

【例 2】プライマリー側にフィードバックを送るよう設定します。

```
hot_standby_feedback = on
```

ただし、これらの設定を行うことで、プライマリー側の VACUUM 処理において回収できない行データが増え、テーブルサイズが大きくなる可能性があります。特に、例 2 においては、スタンバイ側でトランザクションが終了せずに残存すると、プライマリー側に不要な行データが急増してしまうため、注意が必要です。また、トランザクション ID 周回による問題 にもつながることがあります。

ポイント

コンフリクトの課題を解決する上で参考になる情報を示します。

コンフリクトが発生したときの DETAIL メッセージについて

コンフリクトが発生すると、FATAL/ERROR メッセージとともに DETAIL メッセージが出力され、その内容からコンフリクトの原因を特定することができます。以下にコンフリクトの原因と DETAIL メッセージの対応表を示します。

コンフリクトの原因	説明	DETAIL メッセージ
プライマリー側で獲得されたアクセス排他ロック	スタンバイ側の問い合わせにおけるテーブルアクセスとコンフリクトする。明示的な LOCK コマンドおよび各種 DDL 操作を含む。	User was holding a relation lock for too long.
プライマリー側でテーブル空間を削除する	一時作業ファイル用にそのテーブル空間を使用するスタンバイ側の問い合わせとコンフリクトする。	User was or might have been using tablespace that must be dropped.
プライマリー側でデータベースを削除する	スタンバイ側でそのデータベースに接続するセッションとコンフリクトする。	User was connected to a database that must be dropped.
WAL からのバキュームクリーンアップレ	その適用により削除される行のどれか 1 つでも「見る」ことができるスナップショット（注 3）を持つ	User query might have needed to see row versions that must be removed.

コンフリクトの原因	説明	DETAIL メッセージ
コードの適用（注 2）	スタンバイ側でのトランザクションとコンフリクトする。	
WAL からのバキュームクリーンアップレコード	消去されるデータが可視か否かに関係なく、スタンバイ側で対象ページ（注 4）にアクセスする問い合わせとコンフリクトする。	User was holding shared buffer pin for too long. または User transaction caused buffer deadlock with recovery.

- 注 2　行データの不要領域を回収すること。その結果、対象の行データは完全に削除される。
- 注 3　実行されるトランザクションのトランザクション ID(XID)などを記録したもの。複数のトランザクションの同時実行を実現するため、実行されるトランザクションは、自身の XID と行データに記録されている XID を比較するなどして、その行データが可視（アクセス可能）であるか、不可視（アクセス不可）であるかを判断する。
- 注 4　テーブルやインデックスを格納するときの 8,192 バイト単位のブロック領域。テーブルやインデックスにアクセスする際には、このページ単位で共有メモリー上にキャッシュされる。

pg_stat_database_conflicts ビューについて

スタンバイ側で pg_stat_database_conflicts ビューを確認することで、コンフリクトのためにキャンセルされた問い合わせの回数を調べることができます。以下に、確認結果の例を示します。

```
postgres=# ¥x
Expanded display is on.
postgres=# select * from pg_stat_database_conflicts where datname = 'postgres';
-[ RECORD 1 ]----+---------
datid            | 13892
datname          | postgres
confl_tablespace | 0     → テーブル空間が削除されたことによりキャンセルされた問い合わせの回数
confl_lock       | 0     → ロック時間切れのためにキャンセルされた問い合わせの回数
confl_snapshot   | 74    → 古いスナップショットのためにキャンセルされた問い合わせの回数
confl_bufferpin  | 0     → バッファに対する PIN(ロック)のためにキャンセルされた問い合わせの回数
confl_deadlock   | 0     → デッドロックのためにキャンセルされた問い合わせの回数
```

スタンバイ側での参照処理を優先する方法について

スタンバイ側において max_standby_streaming_delay や max_standby_archive_delay に-1 を指定することで、参照処理はコンフリクトが発生してもキャンセルされず優先して処理されるようになります。その反面、WAL の適用処理が待たされるため、スタンバイ側の WAL があふれる可能性があるため注意が必要になります。

レプリケーションスロットの利用について

スタンバイ側で必要とする WAL がプライマリー側で消失しないよう保持するための機構として、レプリケーションスロットがあります。この機構は、スタンバイ側での WAL 適用を遅らせる効果もあるため、上記の例 2 の代わりに利用することができます。ただし、レプリケーションスロットの作成／削除関数を使ってレプリケーションスロットの管理を行う必要性があり、かつ、プライマリー側で WAL ファイルを大量に保持することでディスクを圧迫することにつながるため、利用に際しては注意が必要です。

参考

PostgreSQL 14 文書

- Documentation（PostgreSQL オフィシャル）
  https://www.postgresql.org/docs/
  - III. Server Administration
    - 27. High Availability, Load Balancing, and Replication
      - 27.2. Log-Shipping Standby Servers
        - 27.2.6. Replication Slots
      - 27.4. Hot Standby
        - 27.4.2. Handling Query Conflicts
- PostgreSQL 日本語ドキュメント（日本 PostgreSQL ユーザ会）
  https://www.postgresql.jp/document/
  - III. サーバの管理
    - 27. 高可用性、負荷分散およびレプリケーション
      - 27.2. ログシッピングスタンバイサーバ
        - 27.2.6. レプリケーションスロット
      - 27.4. ホットスタンバイ
        - 27.4.2. 問い合わせコンフリクトの処理

2022 年 1 月 28 日