ストリーミングレプリケーション関連

ストリーミングレプリケーションのコンフリクト緩和措置について（その１）

●例えば、プライマリでの DB、テーブル、インデックスの削除処理（DROP TABLE 等）とスタンバイでの当該テーブルへの参照クエリのコンフリクトが発生した場合。※DB やインデックスの削除等も同様

①コンフリクト検出後、スタンバイ側の設定ファイルの㋐max_standby_streaming_delay（デフォルトは 30 秒）に設定された秒数だけ、参照クエリが実行でき、その後、参照クエリがキャンセルされる。

②その後、プライマリの処理が実行され、プライマリから WAL が送られて、スタンバイに適用されるため、スタンバイのデータはその分、陳腐化する。

③参照時間が短ければ㋐は短くできるが、参照時間が長いときには㋐を長くする、若しくは-1 を設定して参照処理が終わるまでとする方法もある。いずれにせよ、㋐が長ければ長いほど、スタンバイのデータは陳腐化する。

④スタンバイに適用される WAL がアーカイブ WAL である場合、㋐の streaming 部分を archive にして読み替えればいい。通常はスタンバイにストリーミング WAL のアーカイブは存在しないが、archive_mode=always とするとその WAL をアーカイブしてくれる

ストリーミングレプリケーションのコンフリクト緩和措置について（その２）

●例えば、プライマリでの VACUUM や HOT 等で削除された行データの回収処理（再利用）とスタンバイでの当該削除された行を見ることができる参照クエリのコンフリクトが発生した場合。

①プライマリ側の設定ファイルの㋑vacuum_defer_cleanup_age（デフォルトは 0、defer は延期という意味）で調整する。デフォルトの場合、実行中の全てのトランザクションから不可視の行の即時回収を行うが、㋑に値が設定された場合、**その値のトランザクションだけ遅延させる。（←説明に自信なし）**

②スタンバイ側の設定ファイルの㋒hot_standby_feedback（デフォルトは off）を on にすることで、スタンバイがプライマリに現在実施中の参照クエリの情報を wal_receiver_status_interval 間隔（デフォルトは 10s）で送ることで、プライマリの処理を遅延させることができる。

レプリケーションスロット

●上記の多くのコンフリクト緩和策により最終的に WAL 領域があふれてポスグレが停止する可能性もあるため、WAL 領域において削除されずに残り続ける領域を確保するため、プライマリ側の設定ファイルの㋓wal_keep_size に値を設定（デフォルトは 0 で無効）する必要がある。

●スタンバイが㋓の設定値を越えて遅延した場合、今後とも必要とする WAL を削除する可能性があるため、ストリーミングレプリケーションは強制終了させられる。よって、㋓は適切な大きな値が必要となるが、再利用されない領域も増えてくるので難しいところである。

●上記の方法ではなく、レプリケーションスロットを使用することで、スタンバイのリカバリに必要と判断された WAL を max_slot_wal_keep_size（デフォルトは-1 で無制限）の容量分、プライマリの pg_wal 配下に残すことができる。このスロットはストリーミングレプリケーションでは手動作成であるが、ロジカルレプリケーションでは CREATE SUBSCRIPTION を実行した時点でパブリッシャ側に自動作成される。そのため、スロット管理としては、サブスクライバが故障した場合には、pg_wal 配下に WAL が残り続けるため、手動で pg_drop_replication_slot（'サブスクライバ名'）でスロットを削除する必要がある。この際、サブスクライバに当該 WAL は永遠に反映されないことは当然である。

pg_stat_replication ビュー

●当該ビューは、プライマリに接続している walreceiver プロセスからの情報を表示する。複数のスタンバイがあれば、複数行となる。ということで、当該ビューはスタンバイではなく、プライマリで表示される。ただし、ロジカルレプリケーションではパブリッシャとサブスクライバの両方で表示できる。

●当該ビューの中で、sent_lsn（プライマリが送出した Log Sequence Number）、write_lsn（バッファ書き込みの LSN）、flush_lsn（WAL ファイル書き込みの LSN）、replay_lsn（WAL 適用の LSN）は、同期であれば一致しているが、当該ビューは、write_lsn や flush_lsn の更新、又は wal_receiver_status_interval（デフォルトは 10s）が経過したときに更新されるため、一致していない場合がある。なお、wal_receiver_status_interval は、コンフリクト緩和策で使用する hot_standby_feedback の送信間隔でも登場。

●一致していない場合は、スタンバイで pg_last_wal_receiver_location 関数や pg_last_wal_replay_location 関数でスタンバイの状況をリアルタイムで確認できる。

synchronous_commit パラメータ

●ストリーミングレプリケーションは、プライマリから送信された WAL（WAL ファイル単位ではなく、WAL レコード）をスタンバイの WAL バッファに書き込み（remote_write）、それを WAL ファイルに書き込み（on 又は flush）、そして、WAL を適用（remote_apply：ディスクからデータをバッファに読み込んで、WAL を適用して、ディスクに書き出す）する。

●上記括弧内の remote_write、on、remote_apply はプライマリ側の設定ファイルの synchronous_commit パラメータ（デフォルトは on）であり、どの時点でクライアントにコミットを通知するかを設定するものであるが、これらはプライマリとスタンバイが同期している状態である。しかし、２つのデータディスクが完全に一致するのは remote_apply だけである。

●非同期には local と off があり、local はプライマリが WAL ファイルに WAL を書き出せばスタンバイの状態は問わずにコミット完了を通知する。また、off に至ってはプライマリが WAL ファイルに WAL を書き出す前にコミット完了を通知する。

ストリーミングレプリケーションのフェイルオーバー

●プライマリが落ちたときは、スタンバイをプライマリに昇格させるためには以下の 3 つの方法がある。

①promote_trigger_file をスタンバイにぶち込んでおく。どこでもいいらしい。

②スタンバイで pg_ctl promote（pg_ctl と promote との間はスペース）を実行する。

③外部から psql -c 文で pg_promote を発行する。

④クラアントにコミットを送る前にプライマリが落ちても、上記①②③の措置が取られた後にスタンバイで WAL の適用がなされれば、コミット済みとなる。

●スタンバイが落ちたときは、同期している場合には、クライアントにコミット完了通知できなくなるため、プライマリ設定ファイルの synchronous_standby_names において、当該スタンバイを外して他のスタンバイにする、代替のスタンバイがない場合は空にする等の措置が必要となる。

その他

●ストリーミングレプリケーションのホットスタンバイの操作による XID や WAL の発行はない。DML では挿入、更新、削除は不可であるが、参照に係る BEGIN や COMMIT、カーソル操作、LOAD によるライブラリの読み込み、ロック操作、SET 等のパラメータ操作等も可能である。

●wal_sender_timeout（デフォルトは 60s）はプライマリで設定し、タイムアウト発生時には LOG メッセージが表示され、wal_receiver_timeout（デフォルトは 60s）はスタンバイで設定し、タイムアウト発生時には FATAL メッセージが表示される。いずれにしても、レプリケーション接続は停止する。

●pg_rewind の rewind は巻き戻しという意味である。激甚対策としての非同期レプリケーション（スタンバイの状況に関係なくプライマリはクラアントにコミットを送信することができる）を利用している場合、プライマリ故障時に新プライマリ（旧スタンバイ）と新スタンバイ（旧プライマリ）では WAL 適用位置は新スタンバイの方が進んでいる可能性が大きい。そこで pg_rewind を利用することで両者の差分を解消してくれる。