その他

タイムアウト関連

●以下のパラメータのデフォルトは 0 で無効。log_min_error_statement を ERROR 以下に設定すると当該 SQL がログされる。特定セッションでの適用となるよう設定ファイルではなく SET 文が推奨される。

①statement_timeout：SQL がクライアントからサーバに届いた瞬間からスタートして、その SQL が実行完了するまでの時間を設定。当該パラメータを越えた SQL は停止。

②lock_timeoiut：SQL によるオブジェクト（テーブル、インデックス、タプル等）に対するロック獲得を試みている時間が当該パラメータを越えると SQL は停止。statement_timeout ＜ lock_timeoiut は全く意味をなさない。

●idel_in_transaction_session_timeout（デフォルトは 0 で無効）：開いているトランザクションが当該パラメータを越えてアイドル（クライアント待ち）状態の時、セッションを終了。

●archive_timeout （デフォルトは 60s）：固定長 16MB に達するまで時間がかかる場合は未アーカイブが増加するため、当該パラでセグメントを強制的に切り替える時間を設定する。この措置等が面倒ならば、pg_receivewal を使用することで、指定したスタンバイのフォルダにセグメントに関係なく WAL をアーカイブ化する。

パラレルクエリのチューニング

●複数の CPU を活用し、クエリの実行を並行化することで、性能向上を図る仕組み

●パラレルクエリ（PQ）の仕組みは以下のとおり

①前提として、バックグラウンドワーカプロセス（BGWP）を説明する。BGWP はユーザが独自の WP を実装してポスグレに組み込むことができるフレームワーク。

②クライアントから接続要求を受けた時に生成されるバックエンドプロセス（BEP）はクエリの実行を行う。

③BEP はパラレルワーカプロセス（PWP）を起動して PQ を実行する。PWP は BGWP の仕組みを利用している。

●max_worker_processes はシステムがサポートする BGWP の上限値、max_parallel_workers は、システムがサポートする PWP の上限値、max_parallel_workers_per_gather は単一クエリで使用できる PWP の最大値であることから、max_worker_processes≧max_parallel_workers≧max_parallel_workers_per_gather となる。

●max_parallel_maintenance_workers は、CREATE INDEX（B-tree のみ）と VACUUM で使用できる PWP の最大値

●実行計画で PQ が選択された場合に PQ が実行されるため、管理者ができることは、PQ が選択されるようにパラメータのチューニングをするしかない。

●PQ が実行できる処理は、〇〇スキャン、〇〇ジョイン、SELECT 文、CREATE 文等である。その他、gather、aggregate、append 等の用語にも注意**（←もっと調べる）**

ロジカルレプリケーション

●pg_basebackup は、稼動中の DB クラスタのベースバックアップを取るために使用される。データベースへの他のクライアントに影響することなく（非排他的）、バックアップを得ることができる。またこのバックアップは、ポイントインタイムリカバリ（PITR）、ストリーミングレプリケーションのスタンバイの開始点（pg_basebackup -R とすることで、スタンバイの設定ファイルの設定や standby.signal ファイル（recovery.signal ではない）の作成・設置等も実行）として使用できる。

●ロジカルレプリケーションのパブリッシャにおける WAL は、pgoutput モジュールで外部システムが処理できる形に変換して walsender プロセスによりサブスクライバに転送される。それをサブスクライバは bgworker プロセス（logical_replication_worker）で受けてテーブルに適用する。

●ロジカルレプリケーションの場合は、サブスクライバではレプリカしたい DB 及びテーブル等を手動で複製して、パブリッシャでは当該 DB やテーブル等を設定ファイルで指定（CREATE PUBLICATION パブリッシャ名 FOR ALL TABLES WITH (publish = 'insert,update');）で全てのテーブルの更新と挿入だけをレプリカする。これは相互の操作が更新と挿入に限定されることも意味する。WITH がない場合、つまりデフォルトは更新、削除、挿入、TRUNCATE の全てとなる。）するため、pg_basebackup や pg_start_backup＆pg_stop_backup は使用しない。

●相互操作において、挿入は特に問題ないが、更新や削除は REPLICA IDENTITY（ALTER TABLE テーブル名 REPLICA IDENTITY <key>;）により相互のカラムに主キーやユニークキーを付ける必要がある。

postgres_fdw

●fdw は foreign data wrapper で外部データラッパ、つまり外部データ参照機能を表す。

●contrib モジュールであり、外部のポスグレサーバだけが対象である。設定や使用方法等は日本 PostgreSQL ユーザ会の HP の該当項目を参照。

●CREATE FOREIGN TABLE で作成された外部テーブルも一時テーブルと同様に autovacuum 対象外

pg_cancel_backend 関数及び pg_terminate_backend 関数

●長時間実行状態にある SQL を取り消す場合、引数に当該 PID を指定した pg_cancel_backend 関数を使用することで、制御用シグナル（SIGINT）をサーバのバックエンドプロセスに送る。

●長時間待ち状態にあるセッションを切断する場合、引数に当該 PID を指定した pg_terminate_backend 関数を使用することで、制御用シグナル（SIGTERM）をサーバのバックエンドプロセスに送る。

●当該 PID は pg_stat_activity ビューや OS の ps コマンドで確認できる。

●両関数はスーパーユーザのみが実行可能であるが、GRANT 文で他のユーザに権限を与えることができる。

●制御用シグナルをメインのポスグレプロセスに送ってしまうとシャットダウンしてしまうので注意。以下参考

①pg_ctl kill TERM PID = pg_ctl stop -m smart

②pg_ctl kill INT PID = pg_ctl stop -m fast

③pg_ctl kill QUIT PID = pg_ctl stop -m immediate

④pg_ctl kill HUP PID = pg_ctl reload

障害対応

●システム上の制限を設定するために 1 つのプロセスが同時に開くことができるファイル数の上限として max_files_per_process（デフォルトは 1000）がある。「Too many open files」等のエラーが出ない限り、デフォルトのままで OK

●大量のパーティションを取り扱う際、共有ロックテーブルのエントリー数に引っ掛かり、「out of shared memory」等のエラーが出た時は、max_locks_per_transaction（デフォルトは 64）を少なくともパーティション数よりも大きくする。

●restart_after_crash（デフォルトは on）：バックエンドクラッシュ後の自動再初期化

●PANIC は全てのセッションの強制切断によるポスグレ停止。FATAL は特定セッションの強制切断。exit_on_error（デフォルトは off）を on にすると ERROR で特定セッションの強制切断

postgresql.conf

●システムビューである pg_settings では、postgresql.conf の詳細な情報を入手（SHOW）できる。また、挿入と削除はできないが更新（SET）はできる。context 列を見ると、設定項目が反映される以下のタイミングがわかる。

①postmaster：再起動

②user 又は superuser：SET コマンド実行時

③sighup 又は backend 等：SIGHUP シグナル受信時（リロード時）

●ALTER SYSTEM SET 又は RESET コマンド（スーパーユーザのみ実行）によって、スーパーユーザがリモートから㋐postgresql.auto.conf の内容を変更できる。そして、再起動又はリロードで㋑postgresql.conf に反映される。㋐と㋑で同一項目が変更された場合は、再起動又はリロード時に㋐が優先される。

その他のその他

●VACUUM の実行権限がないもの VACUUM を実行すると、ERROR にはならないが、WARNING となりスキップ

●接続要求時は、狭めた接続元と緩い認証（trust や reject）から広めた接続元と厳しい認証の順

●DML はパーサ、リライタ、プランナ、エグゼキュータを遵守するが、DDL と DCL はパーサ、リライタの後にプランナは経由するが、プランがないため、そのまま通り抜けで、エグゼキュータは経由せずに実行。