チェックポイント関連

WAL（アーカイブ含む）

●WAL バッファから WAL ファイルへの書き込みの契機は、トランザクションのコミット時（頻繁なコミットは I/O 負荷大なので、commit_delay 秒を設定し、他のコミットと併せて WAL ファイルへの書き込みを行うことで I/O 負荷を小さくする等も必要になる。）又は WAL バッファがあふれた時

●固定長である 16MB に達した WAL ファイルは、archive_mode が on で archive_command が指定されていれば、アーカイブ化される。そして、アーカイブ化された元の WAL ファイルは不要となり、チェックポイントを契機に当該 WAL ファイルは再利用される。また、16MB に達しなくても、pg_start_backup 実行時の WAL スイッチによっても、アーカイブ化される。

●max_wal_size について

①pg_wal フォルダ内の WAL ファイルの総ファイルサイズの上限は、max_wal_size パラメータの値によって決定

②pg_wal フォルダ内の WAL ファイルを全て再利用するためには、min_wal_size パラメータ（デフォルト値は 80MB（WAL ファイル 5 個分））は、max_wal_size パラメータと同じにしておけばいいが、min<max とすると、max-min 分の WAL ファイルは、チェックポイント契機に削除される。

●WAL ファイルの再利用によりあふれ問題はなくなるが、アーカイブ化された WAL は再利用できない。よって、アーカイブの削除を怠りアーカイブ領域が上限まで達すると WAL ファイルのアーカイブ化が止まり再利用できなくなる。そして再利用できない WAL ファイルが max_wal_size を超えて、PANIC を起こし、SQL サーバが停止する。

●アーカイブの削除の契機としては、ベースバックアップ取得時（チェックポイント処理によりバッファ上のデータはディスクに反映されている）以前のアーカイブは削除しても問題はない。**必要なのは待避した WAL ファイルとベースバックアップ取得以降のアーカイブだからである。**

●バックアップデータの世代管理している場合には、一番古い世代のベースバックアップより前のアーカイブは不要（ベースバックアップがあれば、当該アーカイブは反映済みのため）なので、pg_archivecleanup コマンドを使用することで、任意の時点より前のアーカイブを自動削除することができる。

チェックポイント関連（その１）

●チェックポイントの契機は、WAL ファイルが max_wal_size（デフォルトは 1GB）で設定した量に達した時又は checkpoint_timeout（デフォルトは 5min）で設定した時間（チェックポイント間隔）が経過した時点である。

●頻繁に max_wal_size パラメータの値に達することで、頻繁にチェックポイントが発生する場合において、それが、checkpoint_warning パラメータのデフォルト値 30 秒以下である場合には、その旨を示す LOG レベルのログとその解決のため HINT（max_wal_size の増量）が出力される。

●checkpoint_timeout について

①データの更新が少ないときは、チェックポイントの頻度を少なくしても影響ないため、デフォルトの 5 分より大きい値を設定したほうがいい。逆にデータの更新が多いときに小さくすると、チェックポイント間隔内に書き込みが終わらない可能性もあるため、書き込み時間も重要となる。

②チェックポイント間隔の間に分散して書き込みすることで、I/O 負荷の調整を行う必要があり、チェックポイント間隔の半分の時間で書き込みを終了させたい時は、checkpoint_completion_target を 0.5 に設定する。デフォルトは 0.9 なので、ほぼチェックポイント間隔と書き込み時間は同じになっている。

●full_page_writes（デフォルトは on）が on の時は、チェックポイント直後において、ダーティページのディスクへの書き込み失敗に備えて、更新対象ページを順番に WAL バッファに書き出しておくことができる。

チェックポイント（その2）

●トランザクション処理は、ログを WAL バッファに記録すること、及び共有バッファ上のデータに反映することが同時に行われる。そしてコミットされると、WAL バッファ上のログは WAL ファイルに書き出されるが、共有バッファ上のデータはチェックポイント等が来るまではディスクに書き出されない。

●チェックポイントでは、全てのダーティページデータはディスクに書き出され、特殊なチェックポイントレコードがログファイルに書き込まれる。コミットされ WAL ファイルに存在する WAL の役割は終わる（削除しても OK）。

fsync パラメータ（デフォは on）

●fsync（）は、指定したファイルのデータの永続性を保証するために利用されるシステムコール。file descriptor（ファイルディスクリプタ：ファイル記述子といい、**プログラムからファイルを操作する際、操作対象のファイルを識別・同定するために割り当てられる番号。OS** にファイルの操作を依頼する際に用いられる値）で参照されるファイルのメモリ内で存在する修正されたデータ（ダーティバッファ）を、ディスクデバイス（その他の永続ストレージデバイスを含む）に転送（flush）することで、システム障害や再起動された後も、変更された全ての情報が取り出せるようになる。このシステムコールは転送が終わったとデバイスが報告するまでブロックする。

●ファイルディスクリプタを指定しない場合は sync（）という。

●ポスグレで fsync パラメータが on であれば、fsync をコールして全てのバッファをディスクに転送するか、wal_sync_method で WAL バッファから WAL ファイルへの転送が強制できる。後者は 5 つ程度のメソッドがあるため、pg_test_fsync プログラム（速度テスト）の結果をもって決定する。

●上記の措置により、OS やハードウエアの クラッシュ（DB サーバ自体のクラッシュは除く）後の一貫性は確保されるが、ポスグレの性能は落ちることになる。

●信頼性の高いハードウエアやソフトウエアを使用している場合やバックアップ体制が確実等の場合は、fsync を off にする選択もある。その際は、full_page_writes も off にすることで、ポスグレの性能は上がる。

●逆に off から on にした時にその効果を発揮するためには、initdb --sync-only によって、全ての DB ファイルを安全にディスクに書き出し終了。この時、通常の initdb の操作は行わない。また、再起動しても効果は発揮する。

pg_resetwal

●WAL ファイルが破損したことでサーバが起動できない場合の最終手段として、WAL ファイル内を消去し、さらにオプションで pg_control ファイル内に保存された制御情報の一部を初期化することで、サーバが起動できるようになるはずである。

●起動できたとしても、不完全にコミットされたトランザクションが原因でデータベースのデータに矛盾が起こる可能性があるため、リストア等は必要である。

●pg_control ファイルは、起動した時に最初に読み込まれるファイルで、ポスグレのインスタンスの状態や、前回シャットダウンしたときの各種 ID の値、といった内部の情報が保存されているファイルである。