AUTOVACUUM&VACUUM 関連

AUTOVACUUM のチューニング

●AUTOVACUUM ランチャは制御担当、AUTOVACUUM ワーカは実行担当

●AUTOVACUUM の VACUUM は VACUUM FULL ではない。

●AUTOVACUUM ワーカは大量のタプルの更新等があったテーブルを稼働統計情報の pg_stat_all_tables ビュー等を参照して検査する。なお、検査間隔は autovacuum_naptime（デフォルトは 60s）で設定。

●そして、以下の①及び②のいずれかのトリガーで AUTOVACUUM が実行される。大きなテーブルにおいて、閾値を大きくすると、それだけ不要領域が大きくなるため、AUTOVACUUM に時間を要することから、閾値を小さくすることも検討する。

①VACUUM トリガー：不要領域回収が目的なので、更新や削除された行により閾値を算定

a：autovacuum_vacuum_threshold（デフォルトは 50）

b：autovacuum_vacuum_scale_factor（デフォルトは 0.2）

x：テーブルの全タプル数なので、pg_class.reltuples の値

y：閾値=a+bx：ここで、x を 1000 として、他はデフォルトとすると、y=250 が閾値となる。

※挿入に係るパラメータとして、autovacuum_vacuum_insert_threshold（デフォルトは 1000）と autovacuum_vacuum_insert_scale_factor（デフォルトは 0.2）があるが、ここでは考慮しない。

②ANALYZE トリガー：統計情報更新が目的なので、更新や削除に加えて挿入された行により閾値を算定

a：autovacuum_analyze_threshold（デフォルトは 50、挿入も考慮した値）

b：autovacuum_analyze_scale_factor（デフォルトは 0.1、挿入も考慮した値）

x：テーブルの全タプル数なので、pg_class.reltuples の値

y：閾値=a+bx：ここで、x を 1000 として、他はデフォルトとすると、y=150 が閾値となる。

●その後、AUTOVACUUM 実行時には Visibility Map を参照して高速化を図るといった流れになる。

●autovacuum_max_workers は、同時に実行可能な AUTOVACUUM ワーカ数であるが、通常、テーブル等はそれ以上あるため、待ち時間が発生する。

●autovacuum_vacuum_cost_limit は AUTOVACUUM ワーカによる CPU や I/O のリソース消費量を制限し、autovacuum_vacuum_cost_delay は、上記の消費量を超えたときのスリープ時間

VACUUM 対象タプル

●row version という用語は、行の個別の状態を指すものであり、行の更新毎に同一の論理的な行に対する新バージョンの行が作成される。

●Xmin と Xmax の例を以下に示す。

①XID=100（XID はタプルのシステム列に格納されるトランザクション ID）でタプルを INSERT すると、そのタプルには Xmin=100 を付与

②そのタプルを XID=200 で UPDATE すると、更新前には Xmin=100 の他に Xmax=200 を付与

③更新後には Xmin=200 を付与

④更新後を XID=300 で DELETE すると Xmin=200 の他に Xmax=300 を付与

●VACUUM 対象タプルは、Xmax を持ち、現在進行中のトランザクションの中で一番小さい XID（Oldest Xmin）が基準となる。なぜなら Oldest Xmin より小さい XID はコミット又はロールバックして確定しているためである。

●よって、VACUUM 時点で Oldest Xmin が判明すれば、VACUUM 中の DML の実行は問題ない。それゆえ、ロングトランザクションは Oldest Xmin を見極める際の阻害要因となる。

XID 周回問題（枯渇問題）

●autovacuum = off の場合、XID 周回問題（枯渇問題）を起こさないように、強制的 VACUUM（強制的凍結処理）が実行される。off に設定されたテーブルの例としては、挿入のみで不要領域があまり発生しない場合がある。

●on であっても、可能性は限りなく低いが、ロングトランザクションによって AUTOVACUUM ができない状態が続く場合やテーブルの肥大化により AUTOVACUUM 処理が完了しない場合で XID 周回問題が起こることもある。

●そして、XID 周回問題まで 100 万トランザクションを切った時点でエラー出力後、ポスグレが停止する。リカバリ方法は、シングルユーザーモードでサーバを再起動（postgres --single -D DB 名）して、VACUUM を実行する。

●とにかく、強制的 VACUUM により、デフォルトで以下のとおり現 XID から 5 千万以上前の XID を凍結処理し、その XID を再利用させることで、XID の周回問題を解決することができる。VM をすっきりさせる効果もある。なお、現 XID から 5 千万前までの XID は現役のデータであるため、凍結処理は行わない。

●強制的 VACUUM の閾値関連

①autovacuum_freeze_max_age：デフォは 2 億であり、強制的 VACUUM 実行の閾値

②vacuum_freeze_min_age：デフォは 5 千万

③vacuum_freeze_table_age：デフォは 1.5 億であり、凍結処理範囲（①－②）

④VACUUM FREEZE コマンドは②と③を 0 にすることと同じであり、現在から 2 億前までを凍結。VACUUM FULL はテーブルを書き換えるため、当然 VACUUM FREEZE している。

pg_xact フォルダ

●強制的 VACUUM 時には pg_xact フォルダの古いデータの削除も許可される。

●トランザクションコミット時に pg_xact ログ「例：XID=100 のトランザクションはコミットされた」をバッファに書き込み、コミット時ではなくバッファあふれやチェックポイント時に pg_xact フォルダに書き出す。

●並行してトランザクションコミット時には WAL が WAL フォルダに書き出されているので、pg_xact ログが消失しても問題はない。

pgstattuple モジュール

●実行権限は、定義済みロールである pg_stat_scan_tables（潜在的に長時間、テーブルの ACCESS SHARE ロック【SELECT による参照による共有ロック】を取得する可能性がある監視機能を実行する）又はスーパーユーザ

●pgstattuple（'テーブル名'）で詳細なタプルレベルの統計情報（リレーションの物理的な長さ、"不要"なタプルの割合等）を返すため、バキュームの必要性を判断する有用な情報となる。例えば、有効タプル割合が低い等。また、無効タプルがゼロならは、VACUUM 直後である等の状況がわかる。

●pgstatindex、pgstatginindex、pgstathashindex 等の引数にインデックスを指定するとそれらの情報（詳細は省略）を返す。

●ACCESS SHARE ロックは、当該テーブルやインデックスに対する更新が可能であるため、結果に影響を与える。