PostgreSQL Vacuum

Vacuum background process, purpose and insights.

Author: RadixTrie Open-Source Team

Purpose

Vacuum is a background maintenance process that facilitates the persistent operation of PostgreSQL. Its two main tasks are removing **dead tuples** and the freezing transaction IDs. Freezing is essential for "Concurrency Control", a topic for another day.

Dead Tuple: Whenever rows in a table are deleted, the row or tuple is marked as dead, during an update (UPDATE operations = DELETE + INSERT), it marks corresponding exiting tuple as dead and inserts a new tuple.

The space occupied by these dead tuples is called Bloating (fragmentation). The vacuum process reclaims this space. This space is not released to the filesystem unless the dead tuples are beyond the high water mark. It is a defragmentation of the database page to be reused for inserts by the same table.

PostgreSQL maintains both the past image and the latest image of a row in its own Table. UNDO is maintained within each table and this is done through versioning.

Every row of PostgreSQL table has a version number and these versions are maintained within each table with hidden columns of a table (example xmin, xmax etc.). If xmax is > 0 the row is deleted and the VACUUM process can reclaim the space.

Vacuum

The Vacuum processes needs to be monitored and carefully configured. This could have a negative impact on performance. Vacuum, autovacuum and analyse are the three main areas to look at. Never switch off autovacuum. The vacuum and analyse processes can be configured with parameters on global and table level. You might need to vacuum high-hit and high-grow tables with lots of updates and or deletes more frequently than static tables.

The VACUUM operation performs the following series of operation:

- Scan all pages of all tables (or specified table) of the database to get all dead tuples
- Freeze old tuples if required
- Remove the index tuple pointing to the respective DEAD tuples
- Remove the DEAD tuples of a page corresponding to a specific table and reallocate the live tuples in the page
- Update Free space Map (FSM) and Visibility Map (VM)
- Truncate the last page if possible (if there were DEAD tuples which got freed)
- Update all corresponding system tables

Parameters to consider (located in ***postgresql.conf*** file):

autovacuum (boolean)
autovacuum_max_workers (integer)
autovacuum_naptime (integer)
autovacuum_vacuum_threshold (integer)
autovacuum_vacuum_insert_threshold (integer) (+v13)
autovacuum_analyze_threshold (integer)
autovacuum_vacuum_scale_factor (floating point)
autovacuum_vacuum_insert_scale_factor (floating point) (+v13)
autovacuum_analyze_scale_factor (floating point)
autovacuum_freeze_max_age (integer)
autovacuum_multixact_freeze_max_age (integer)
autovacuum_vacuum_cost_delay (floating point)
autovacuum_vacuum_cost_limit (integer)
autovacuum_work_mem (integer)

vacuum_cost_delay (real)
vacuum_cost_limit (integer)
vacuum_cost_page_dirty (integer)
vacuum_cost_page_hit (integer)
vacuum_cost_page_miss (integer)
vacuum_defer_cleanup_age (integer)
vacuum_failsafe_age (integer) (+v14)
vacuum_freeze_min_age (integer)
vacuum_freeze_table_age (integer)

vacuum_multixact_failsafe_age (integer) (+v14)
vacuum_multixact_freeze_min_age (integer) (+v14)
vacuum_multixact_freeze_table_age (integer) (+v14)

Monitoring

Before adjusting any of the above-mentioned parameters you will need to understand the database, the objects, and the rate of change per object. Altering a parameter on global level may cause performance and stability problems. All the information is in the PostgreSQL information schema.
Every database is different in terms of its size, traffic pattern, and rate of transactions etc.

We recommend that you start by gathering enough information about each database before changing the parameters or rolling out a manual vacuum/analyse. Collect the following information:

- Number of rows in each table
- Number of dead tuples in each table
- The time of the last vacuum for each table
- The time of last analyse for each table
- The rate of data insert/update/delete in each table
- The time taken by autovacuum for each table
- Warnings about tables not being vacuumed
- Current performance of most critical queries and the tables they access
- Performance of the same queries after a manual vacuum/analyse

Most of the above-mentioned parameters can be monitored and calculated to find the best global settings for the bulk of the tables, and per table for individual settings as required. The below mentioned "information schema" tables can be used to monitor and calculate these parameter settings:

- pg_stat_progress_vacuum,
- pg_stat_all_tables,
- pg_stat_all_indexes,
- pg_stat_activity,
- pg_statio_all_tables,
- pg_statio_all_indexes

Long running queries can also be a result of mismatched or lack of proper vacuum and analysing on tables.

Summary

Please be cautious, VACUUM FULL is not an ONLINE operation. It's a blocking operation. You cannot read from or write to the table while VACUUM FULL is in progress.

Increase the autovacuum to run more often, analyse table activity to determine if global or table level vacuum is required. By tweaking the *vacuum* parameters and making sure that autovacuum have enough time to run every day, you will be able to reduce the row count and disk space of the database and have better query performance.