Do you vacuum everyday?

“HINT: Stop the postmaster and vacuum that database in single-user mode.”

07. Apr. 2022

Speaker introduction

Hannu Krosing
Cloud SQL / PostgreSQL
hannuk@google.com

Working with PostgreSQL since it was called `Postgres95` (and also played around with `Postgres 4.2` - without the "SQL" - a little before that).

My oldest *surviving* post on `postgresql-hackers@` mailing list archives is from January 1998, proposing using index for fast ORDER BY queries with LIMIT.

The first DBA at Skype, where I wrote patches for making **VACUUM** able t**o work on more than one table in parallel** and invented the sharding and remote call language **`pl/proxy`** to make it easy to use PostgreSQL in an infinitely scalable way.

Have written books, **PostgreSQL 9 Admin Cookbook** and **PostgreSQL Server Programming**

After Skype I did 10+ years of PostgreSQL consulting all over the world as part of 2ndQuadrant.

For last three years he has been a PostgreSQL Database Engineer at Google working mostly with Cloud SQL.

Table of Contents

01

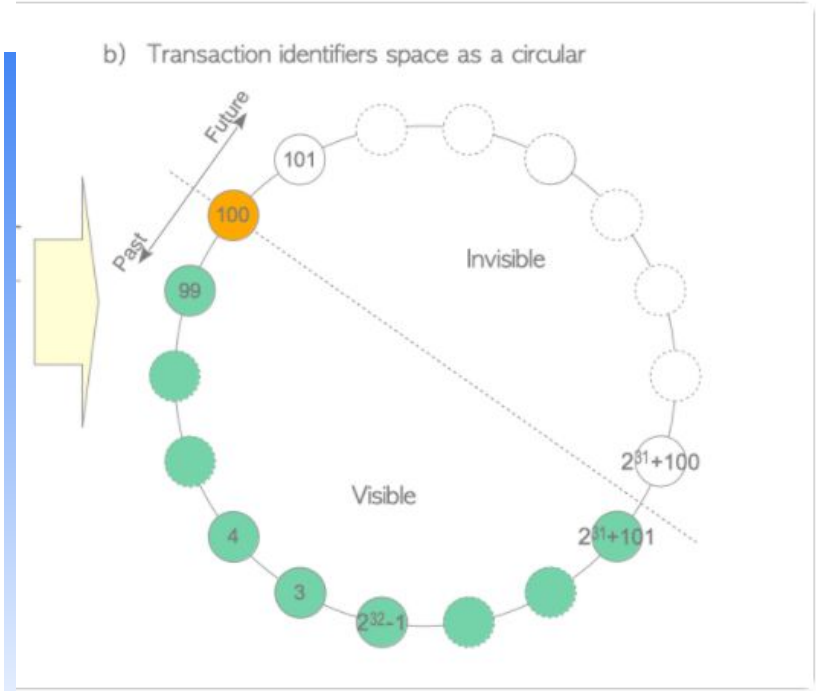
VACUUM and MVCC

- What is MVCC
- History, origins
- What does VACUUM do
- When things can go wrong - BLOAT and WRAPAROUND
- Why PostgreSQL MVCC is still awesome

MVCC the PostgreSQL way

- Tuples have “lifetime”
- All data stays in same Heap file
- Each transaction sees different set of tuples
- DEAD tuples cleaned up once invisible for all
- RECENTLY_DEAD tuples cause bloat
- Basic architecture is very clean

	t_xmin	t_xmax	t_cid	t_ctid	user data
Tuple_1	99	100	0	(0,2)	'A'
Tuple_2	100	0	0	(0,2)	'B'



https://www.interdb.jp/pg/pgsql05.html

Quick History of MVCC

- Started as full-history database
- Nowadays called TEMPORAL TABLES
- Had (tmin, tmax) instead of (xmin, xmax)
- So just a single write transaction
- Was changed to current way in 6.3
- Incremental improvements
- Like HOT updates, mini-vacuum, ...

What does VACUUM do

- Cleanup
  - Frees space used by dead rows
  - Cleans indexes
- FREEZE
  - FEEZE txids in live rows
  - Used to be xmin=2, now pair of flags
  - May be something else in the future

When things can go wrong

- If Cleanup is failing you get
  - **BLOAT**
- When FREEZE is failing you get
  - **Blocked DDL**
  - **WRAPAROUND**

Why PostgreSQL MVCC is still awesome

- keeping all housekeeping out of critical path
- Simplicity of concept, relative simplicity of optimisations
- Immediate (O(1)) commit or rollback
- No slowdown of OLAP queries when loading (updating) data in parallel with

- Moore's law is dead, so doing more things in parallel is the only way forward

02

Why things can go wrong

- Long Transactions hold back vacuum
- “Long Transaction” can also be 2PC
- Or it can be on Replica
- Or it can be a replication slot
- … all the above cause BLOAT and can lead to WRAPAROUND
- sometimes VACUUM is just not fast enough (So tune it!)

- … sometimes TEMP tables cause wraparound (new to me)

03

How things can go wrong

- Everything is slowing down because of big BLOAT

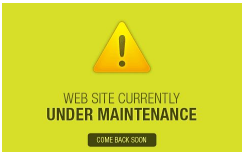
- There is big BLOAT with minor slowdown

- There is not much BLOAT but danger of WRAPAROUND

- Anti-wraparound VACUUM blocks DLL ...

- ... then DDL blocks main DML queries

- System goes into WRAPAROUND --> big outage




```
ERROR: database is not accepting commands to avoid wraparound data loss in database <oldest_datname>.
HINT: Stop the postmaster and vacuum that database in single-user mode.
      You might also need to commit or roll back old prepared transactions, or drop stale replication slots.
```

04

How to return to normality FAST

- **Do not :**
  - Run in single-user mode
  - VACUUM everything
- **Instead DO :**
  - Check for VACUUM blockers, resolve any found
  - Find the database(s) with oldest transaction ids
  - Find the tables(s) with oldest transaction ids
  - Tune Vacuum, then Vacuum the tables(s)
- once OK, tune autovacuum to keep up in the future

05

What was added to Cloud SQL for this

- Never use Single-user mode
- Skip truncate when close to wraparound (critical for <= v11)
- No need for superuser to terminate Autovacuum
- `cloudsql.enable_maintenance_mode`
- work on getting rid of 1GB limit for deleted xids

Never user Single-user mode

- You are running completely blind
  - VACUUM VERBOSE isn't
  - No way to see progress (except guessing from CPU, Memory and Disk Usage patterns)
  - **`pg_stat_progress_vacuum`** view is not available
- You need at least 2x the space as WAL is not rotated, as CHECKPOINTS are not running
- If you have replicas, then these are not updated and need to catch up after restarting back to normal mode
- Vacuuming indexes is done serially (newer versions of PostgreSQL can clean up more than one index in parallel)

Never user Single-user mode (Cloud SQL)

- Documented the mitigations (also sent a mail about this to pg hackers list)
- Changed the error message point to documentation
- Patched PostgreSQL to skip truncate when close to wraparound
  - Absolutely critical for PostgreSQL versions <= v11
  - Can be worked around manually in v12+

No need for superuser to terminate Autovacuum (Cloud SQL)

- PostgreSQL requires superuser to pg_cancel_backend() autovacuum
- We patch PostgreSQL to allow terminating VACUUM by any user with pg_signal_backend

No need for superuser to terminate Autovacuum (Cloud SQL)

- We added a special maintenance mode to avoid single-user
- Enabled via `cloudsql.enable_maintenance_mode = true`
- Can use extra 0.5 million transaction ids
- Is throttled to discourage abuse

- Only way to remove stuck temp file once in Wraparound

Work on getting rid of 1GB limit for deleted xids

- One thing holding back vacuuming HUUGE tables is 1GB limit on number of collected deleted xids
- This limit has came up in discussions, patch to increase was rejected
- There has been some discussions around this as part of general VACUUM speedup
- One way around this I have tested is collecting initially the deleted tuple ids in a file
- When testing this, it was not measurably slower than collecting in oversized memory array

Thank you.

Google Cloud