Autovacuum, explained for engineers

PGCon 2016, Ottawa

Ilya Kosmodemiansky
ik@postgresql-consulting.com

PostgreSQL-Consulting.com

Outline

- What is it and why is it so important?
- Aggressiveness of autovacuum
- What else important can autovacuum daemon do
- Some issues with schedulling autovacuum
- Autovacuum and replication
- How to remove bloat

Two most common problems we meet in our practice

- autovacuum = off
- Autovacuum settings are default

Two most common problems we meet in our practice

- autovacuum = off
- Autovacuum settings are default
- **That means there is a lot we can do about improving performance of this particular database**

What is autovacuum?

Modern (classical) databases must deal with two fundamental problems:

- **Concurrent operations**
  For that they can transactions, ACID transactions
- **Failures**
  For that they can recover to the last successful transaction using WAL

What is autovacuum?

Technically that means

- There is a combination of locking and MVCC algorithms that provides transactions support
- **Undo** and **Redo** information is stored somewhere to make recovery possible

What is autovacuum?

In PostgreSQL

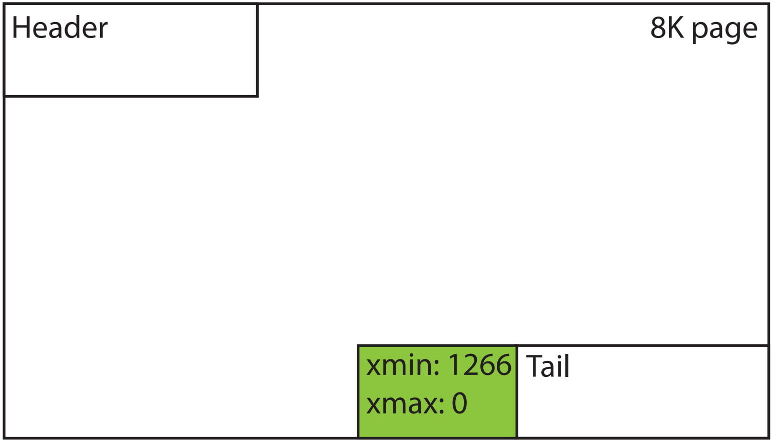
- Redo - in WAL
- Undo - directly in datafiles
- UPDATE = INSERT + DELETE
- DELETE is just marking tuple as invisible

xmin

```
tt=# INSERT into test(id) values(5);
INSERT 0 1
tt=# select *,xmin,xmax from test;
 id | xmin | xmax
----+------+------
  5 | 1266 |    0
(5 rows)

tt=# select txid_current();
 txid_current
--------------
         1267
(1 row)
```

INSERT



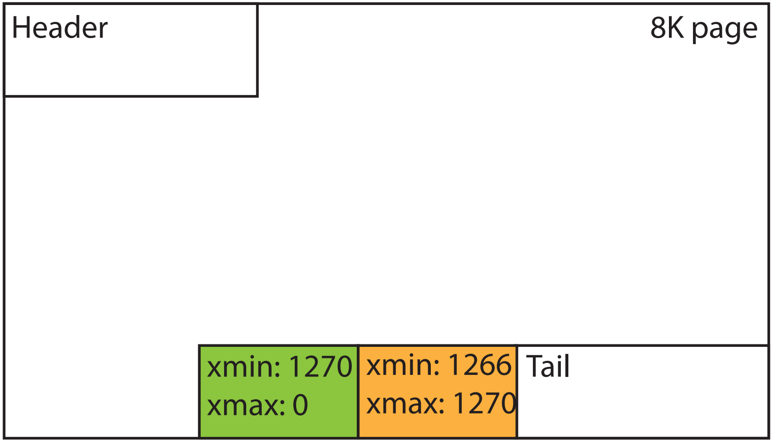
xmax

```
tt=# begin;
BEGIN
tt=# UPDATE test set id=5 where  id=4;
UPDATE 1

In another session:

tt=# select *,xmin,xmax from test;
 id | xmin | xmax
----+------+------
  4 | 1264 | 1270
(3 rows)
```

UPDATE



Some garbage collection is required

Tuples that are not visible to any running transaction should be removed

- Otherwise fragmentation increases and you run into bloat aka Big Data
- autovacuum workers do that, table by table
- Old-fashioned VACUUM is a bad choice

Beside that, autovacuum workers

- Collect statistics for the optimizer
- Perform wraparound for txid[1]

[1] I will not cover wraparound, for details please see a talk by Masahiko Sawada http://goo.gl/15YZNX

Some garbage collection is required

Tuples that are not visible to any running transaction should be removed

- Otherwise fragmentation increases and you run into bloat aka Big Data
- autovacuum workers do that, table by table
- Old-fashioned VACUUM is a bad choice

Beside that, autovacuum workers

- Collect statistics for the optimizer
- Perform wraparound for txid[1]

You do not want to turn autovacuum off!

[1] I will not cover wraparound, for details please see a talk by Masahiko Sawada http://goo.gl/15YZNX

VACUUM vs autovacuum

- VACUUM removes all pages, which are not visible to any running transaction[2]
- You need to run it really frequently, to **prevent** bloat (VACUUM does not remove it!)
- If you don't, you will need VACUUM FULL - it rebuilds the table, that can be painful
- autovacuum automates that all in some convenient manner

[2]Things are more complicated, refer to Jim Nasby's talk http://goo.gl/K0PDtq for some details

This sort of work **must** be finally done

- If your autovacuum process runs for hours and interferes with some DDL, to simply terminate it is not an option
- Especially for OLTP, autovacuum should be configured **aggressively enough**: so it can work with small portions of data quickly

autovacuum: aggressive enough

```
postgres=# select name, setting, context  from pg_settings
where category ~ 'Autovacuum';

                name                 | setting  |  context
-------------------------------------+----------+------------
 autovacuum                          | on       | sighup
 autovacuum_analyze_scale_factor     | 0.05     | sighup
 autovacuum_analyze_threshold        | 50       | sighup
 ...
 autovacuum_max_workers              | 10       | postmaster
 ...
 autovacuum_naptime                  | 60       | sighup
 autovacuum_vacuum_cost_delay        | 10       | sighup
 autovacuum_vacuum_cost_limit        | -1       | sighup
 autovacuum_vacuum_scale_factor      | 0.01     | sighup
 autovacuum_vacuum_threshold         | 50       | sighup
(11 rows)
```

Scale factors

- autovacuum_vacuum_scale_factor = 0.01 means that at least 1% of rows (**% of table size**) in the table should be changed before autovacuum happens
- autovacuum_vacuum_threshold - alternative setting, exact number of rows
- The idea is to make autovacuum work more frequently, vacuuming tables in small portions
- Can be set per-table, but that can be some sort of pain

Autovacuum has it's own mechanism to reduce I/O ov

Autovacuum delays *autovacuum_naptime* seconds, then checks tables if they need a vacuum. It runs vacuum on a table until *autovacuum_vacuum_cost_limit* is reached, then sleeps *autovacuum_vacuum_cost_delay* milliseconds.

- It looks like this mechanism does not work like it was designed
- For example it doesn't make a difference between physical and logical IO
- I doubt if such mechanism is useful at all on modern SSD's

A good idea, if you have bad disks

in crontab:

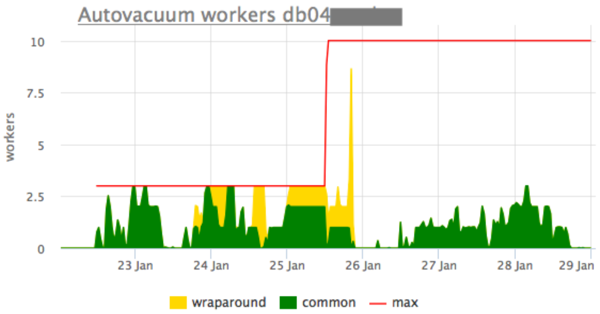
```
* * * * *  /usr/bin/pgrep -f 'postgres: autovacuum' | xargs --no-run-if-empty -I $ renice -n 20 -p $ >/dev/null 2>/dev/null
* * * * *  /usr/bin/pgrep -f 'postgres: autovacuum' | xargs --no-run-if-empty -I $ ionice -c 3 -t -p $
```

in postgresql.conf:
autovacuum_max_workers → 10-20
autovacuum_vacuum_cost_delay → 10

Keep in mind, that ionice could not work in certain cases, such as Noop scheduller,

LWM or software RAID

As a result



Autovacuum workers db04[redacted]
workers
10
7.5
5
2.5
0
23 Jan
24 Jan
25 Jan
26 Jan
27 Jan
28 Jan
29 Jan
wraparound
common
max

autovacuum_vacuum_cost_delay

- On fast SSD, autovacuum_vacuum_cost_delay should be be shorter than on slower SAS
- Is a global setting
- In some cases can be an issue (one tablespace on SSD, another on SAS)

ERROR: canceling statement due to conflict with reco

- The tuple, cleaned up by autovacuum on master, is still in use by some query on hot standby
- hot_standby_feedback = on - The safest way, in spite of some bloat on master

Before you hurry to reconfigure your PostgreSQL

- autovacuum does not remove existing bloat
- dump/restore can be an option, but...
- http://reorg.github.io/pg_repack/
- https://github.com/PostgreSQL-Consulting/pgcompacttable

Questions?

ik@postgresql-consulting.com

slides will be available at
pgcon.org