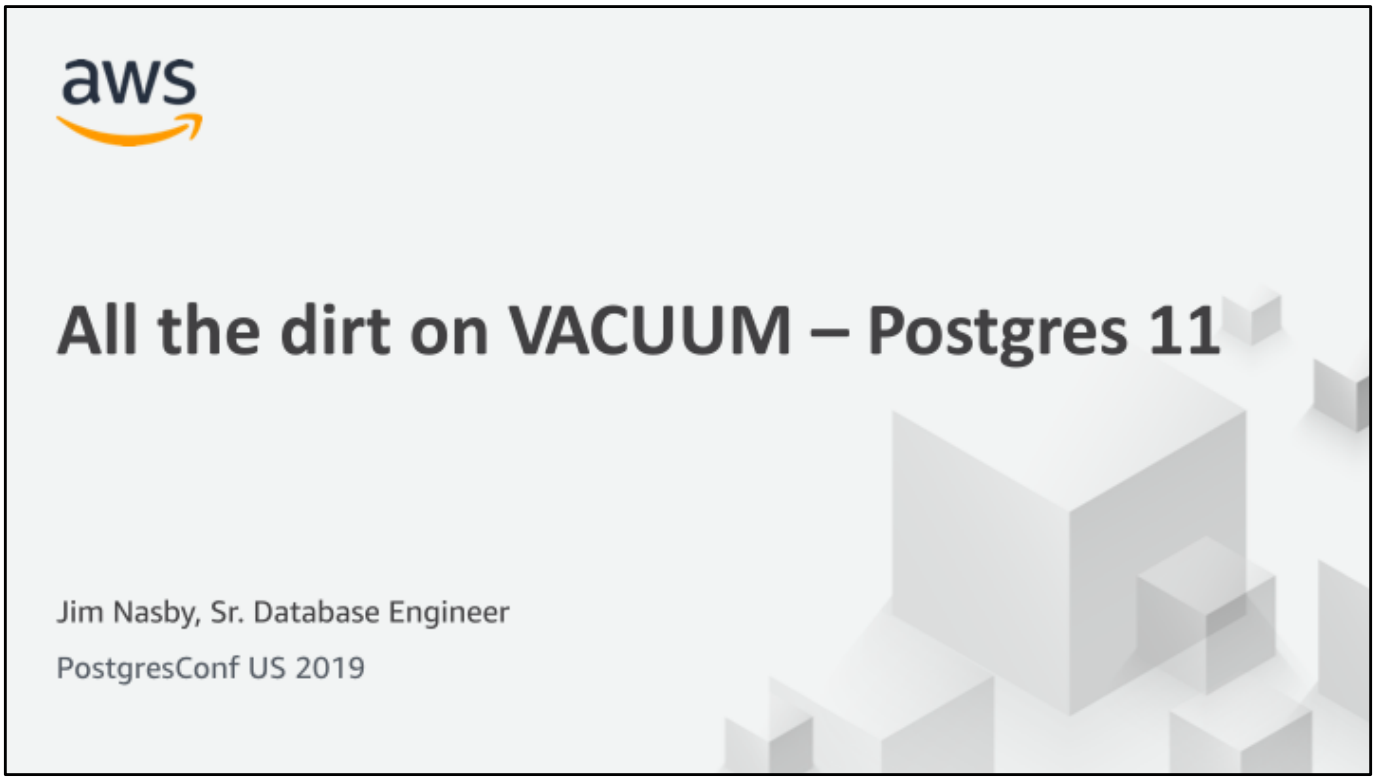

aws
All the dirt on VACUUM – Postgres 11
Jim Nasby, Sr. Database Engineer
PostgresConf US 2019

Intro

In-depth talk

But… here's some quick tips

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Intro

Postgres Multi-Version Concurrency Control is like a credit card

Every UPDATE, DELETE and ROLLBACK leaves “debt” that must be repaid

Not paying off credit cards leads to bankruptcy

Not vacuuming leads to a “death spiral”

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Intro

Postgres Multi-Version Concurrency Control is like a credit card

Every UPDATE, DELETE and ROLLBACK leaves “debt” that must be repaid

Vacuum is how this debt is repaid

You **WANT** vacuum running in your database

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Intro

I LOVE vacuum! How can I get more of it?

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Intro

By default, autovacuum limited to 4MB/s write
Increase `vacuum_cost_limit` from 200 to 2000

Ensure `maintenance_work_mem` is as close to 1GB as possible

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Heap Only Tuples

Applies when an update does not change any index values* and the new tuple will fit on the same page

Allows for updating just the heap, without touching indexes

Dead HOT tuples can be removed at any time, without need for vacuum

HOT is like paying cash instead of using a credit card

See Grant McAlister's talk from yesterday

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

* Indexed values means any column referenced anywhere in an index, including predicates and functions. See src/backend/access/heap/README.HOT.

Intro

Beware of:

- Long-running transactions (including idle in transaction)
- Prepared transactions (best to set `max_prepared_transactions = 0`)
- Stuck replicas

Targeted manual vacuums help a lot

- Vacuum small, frequently modified tables once a minute
- Vacuum the entire instance once a day / week
  (possibly with `vacuum_cost_delay` > 0)

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

VACUUM

`VACUUM FULL`: completely rebuilds table and indexes

`VACUUM FREEZE`: sets freeze table age limits to 0

`VACUUM`: regular manual vacuum

`VACUUM ANALYZE`: also runs analyze after vacuum

`VACUUM VERBOSE`: provides status and stats

See also `vacuumdb` shell command

Autovacuum: built-in background automatic vacuum process

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

VACUUM FULL

Rebuilds table and indexes from scratch, similar to CLUSTER

Takes an exclusive lock on the table

Since it's a table rebuild, doesn't actually vacuum anything

https://github.com/reorg/pg_repack is another alternative

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

`vacuum()`

Can not be run in a transaction (or function/procedure)

For each table, call `vacuum_rel()` (and `analyze_rel()` if requested)

Update `datfrozenxmin` and `datminmxid`

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

vacuum_rel()

`vacuum()` > `vacuum_rel()`

Vacuums a single relation

Non-aggressive autovac will skip relation if locked

Does a bunch of mundane stuff then calls either `cluster_rel()` (VACUUM FULL) or `lazy_vacuum_rel()`

If not autovac, call itself to vacuum the TOAST table

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

lazy_vacuum_rel()

vacuum() > vacuum_rel() > lazy_vacuum_rel()

Does the real work of vacuuming

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

There was a bug in some old versions where relfrozenxid and relminmxid were updated even if the whole table hadn't been scanned, potentially resulting in data loss.

See vacuum_set_xid_limits()

Vacuum Process (`lazy_vacuum_rel()`)

- Acquire locks, set limits
- Loop through heap, possibly skipping pages
- Per-page activity
- *Serially* loop through all indexes
- Remove dead tuples from heap
- Vacuum FreeSpaceMap
- Index cleanup
- Attempt relation truncation
- Update `pg_class` info

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Vacuum Process (`lazy_vacuum_rel()`)

- Acquire locks, set limits

SETUP

- Loop through heap, possibly skipping pages
- Per-page activity
- *Serially* loop through all indexes
- Remove dead tuples from heap
- Vacuum FreeSpaceMap

MAIN LOOP

- Index cleanup
- Attempt relation truncation
- Update `pg_class` info

CLEANUP

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

What tuples can be vacuumed?

Only rows that are not visible to currently running transactions

Generally* limited by the **oldest running transaction** in the database

Can be changed by `old_snapshot_threshold`

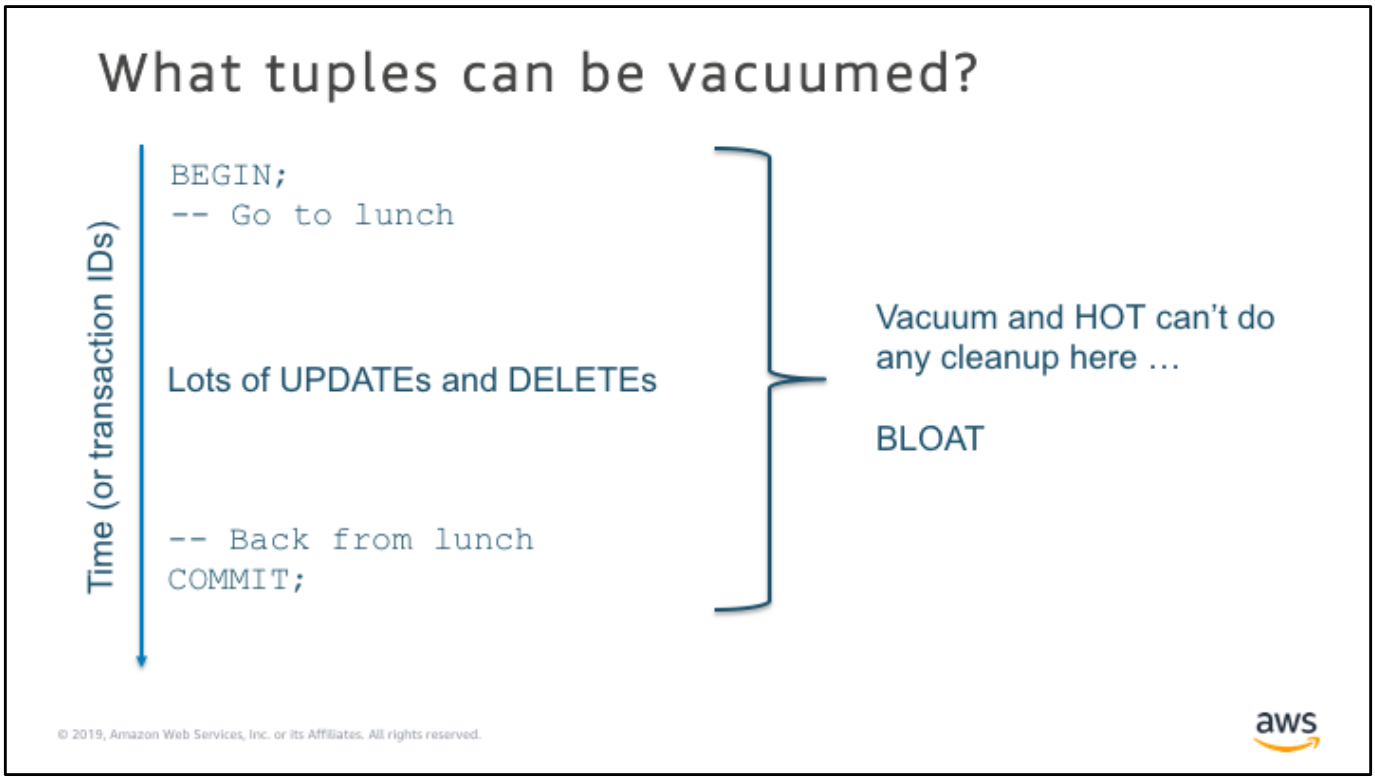
Streaming replication (`vacuum_defer_cleanup_age`, `hot_standby_feedback`), prepared transactions, and logical decoding can also affect it

Special handling for current XIDs and locks

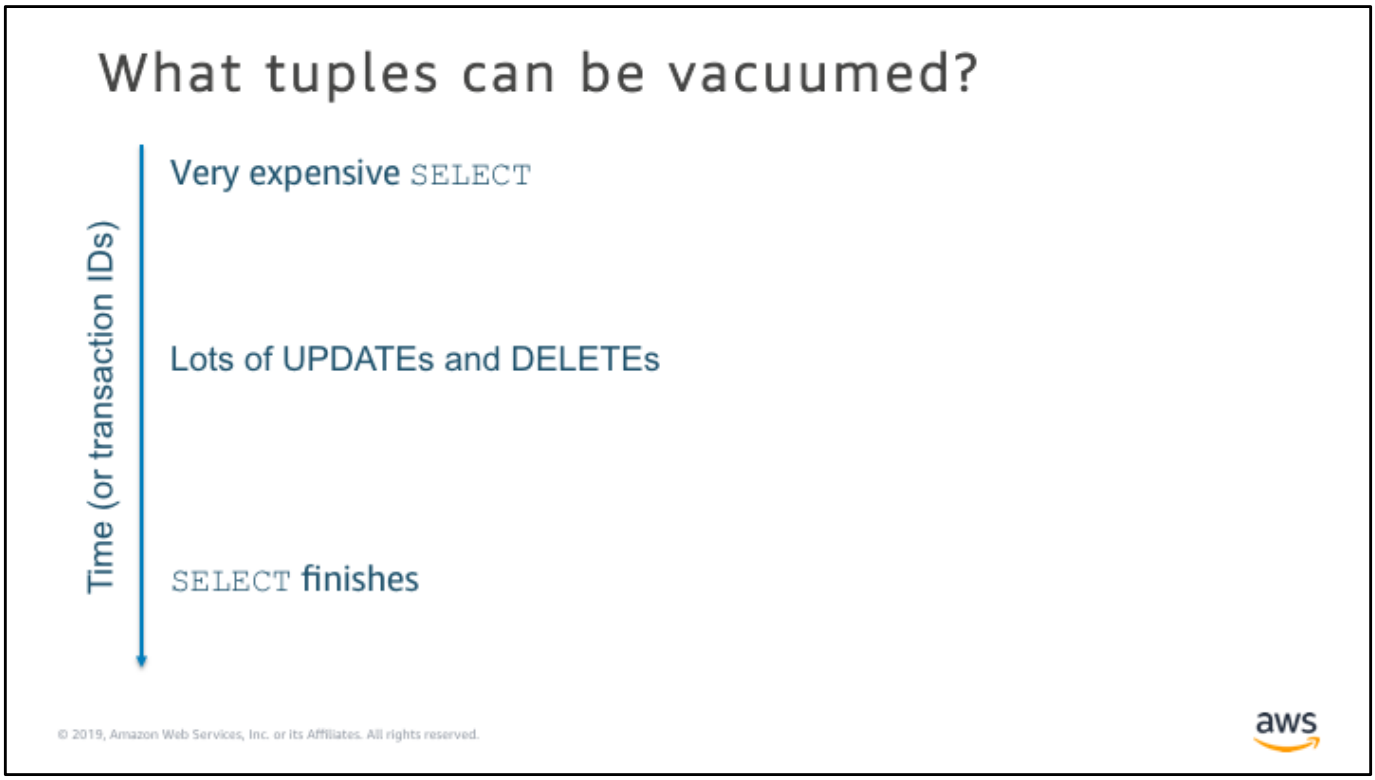
© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

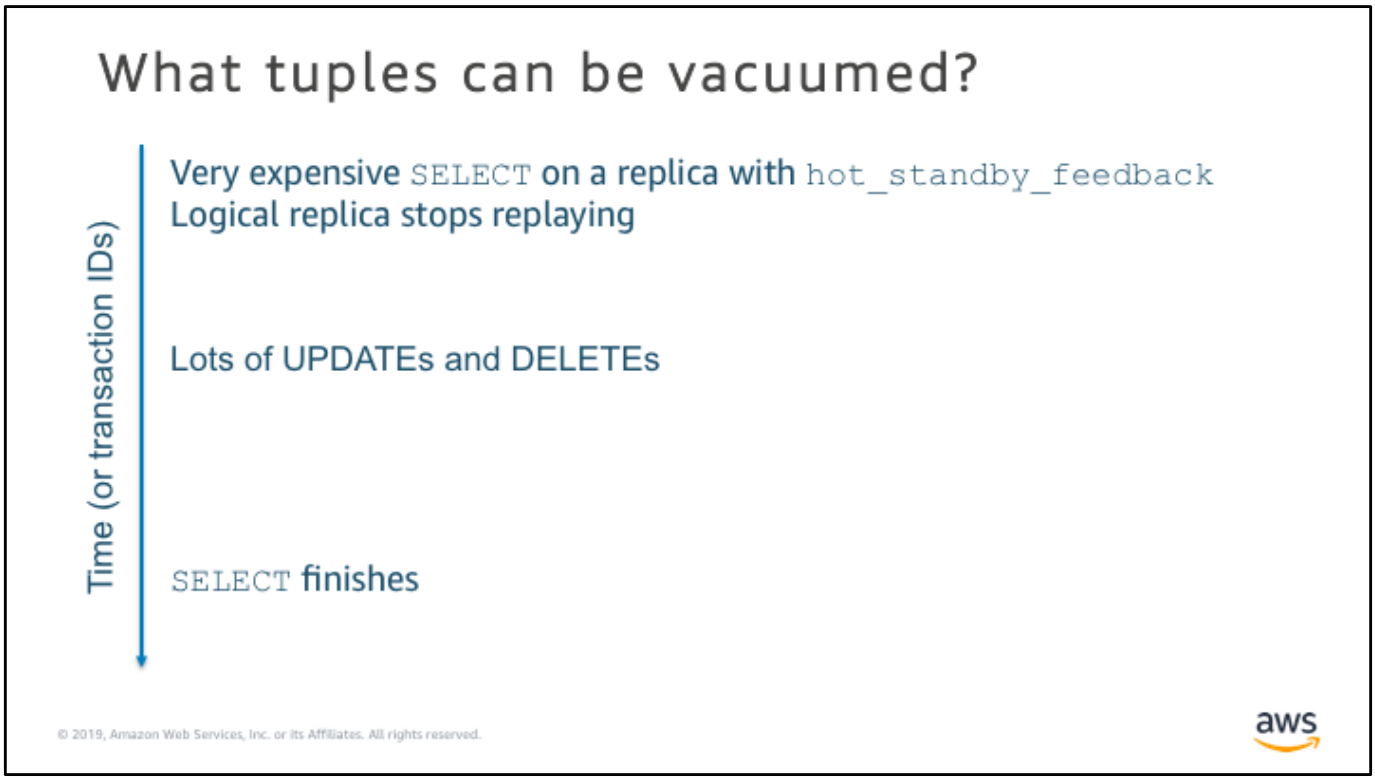
See HeapTupleSatisfiesVacuum() for details.



See HeapTupleSatisfiesVacuum() for details.



See HeapTupleSatisfiesVacuum() for details.



See HeapTupleSatisfiesVacuum() for details.

Setting limits: Freezing

Transaction ID (XID) and MultiXact ID (MXID) values are limited to 31 effective bits*

Allowing these values to roll over would result in data loss

Old values must be “frozen”

An “aggressive” vacuum is run when a table contains XIDs or MXIDs in need of freezing, as determined by `relfrozenxid` and `relminmxid` in `pg_class`.

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

XIDs and MXIDs are 32 bit unsigned values, but XIDs need to accommodate transactions that are considered to be “in the future”, which means there can’t be more than 31 effective bits. MXIDs don’t have a concept of “in the future”, but are artificially limited to 31 bits.

Freezing

A non-aggressive vacuum can also freeze tuples

`vacuum_freeze_min_age*` and `vacuum_multixact_freeze_min_age` determine how old a XID/MXID must be to be considered for freezing

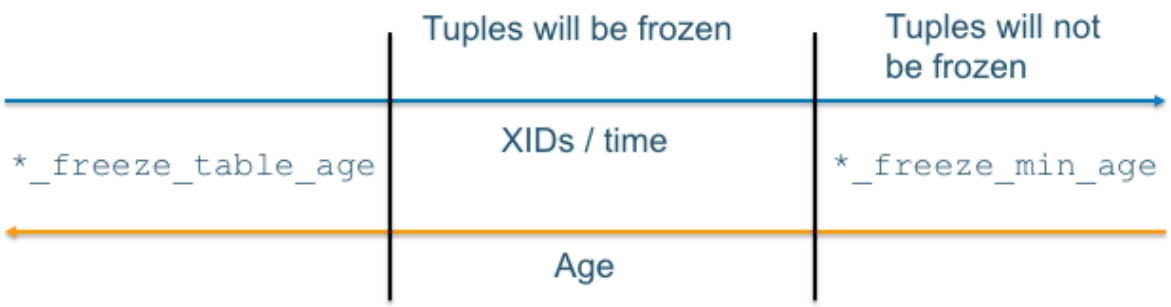
`(auto)vacuum_freeze_table_age*` and `(auto)vacuum_multixact_freeze_table_age` determine when a vacuum will become aggressive

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

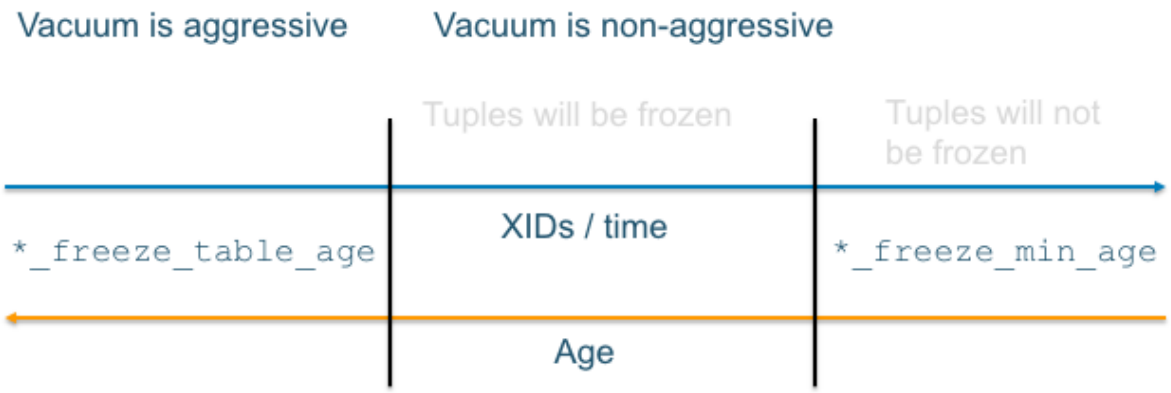
*_min_age are computed from the oldest running XID in the system.
*_table_age are computed from pg_class.relfrozenxid and pg_class.relminmxid.

Freezing



© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Freezing



© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

What is a MultiXact?

MultiXacts occur when multiple transaction IDs need to lock and invalidate a tuple, most commonly due to updates on a Foreign Key parent.

Subtransactions (from savepoints & plpgsql EXCEPTION handlers) create their own transaction IDs, so a single backend can create MultiXacts

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Vacuum Process (`lazy_vacuum_rel()`)

- Acquire locks, set limits — SETUP

- Loop through heap, possibly skipping blocks
- Per-page activity
- *Serially* loop through all indexes — MAIN LOOP
- Remove dead tuples from heap
- Vacuum FreeSpaceMap

- Index cleanup
- Attempt relation truncation — CLEANUP
- Update `pg_class` info

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Loop through heap, possibly skipping blocks

```
for (blkno = 0; blkno < nblocks; blkno++)
```

`nblocks` is determined on entry to `lazy_scan_heap()`.

Newer blocks will not be scanned.

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Block skipping

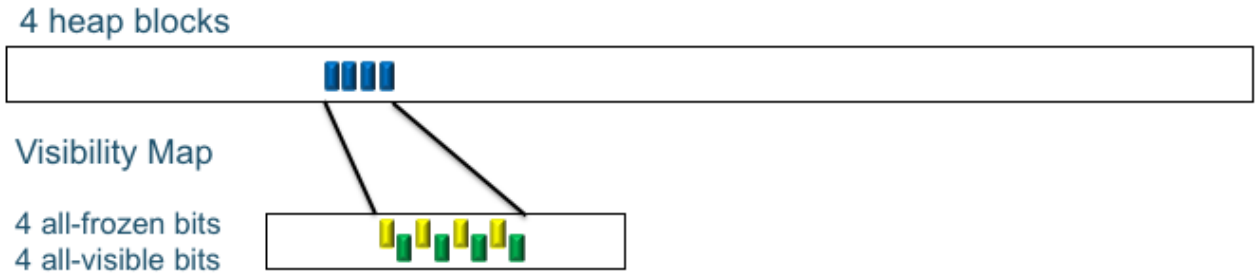
Vacuum can skip blocks that are all-visible
An aggressive vacuum can skip blocks that are all-frozen

While reading heap, vacuum will skip blocks if at least 32 blocks would be skipped

Skipping can be disabled by adding the `DISABLE_PAGE_SKIPPING` option to `VACUUM`.

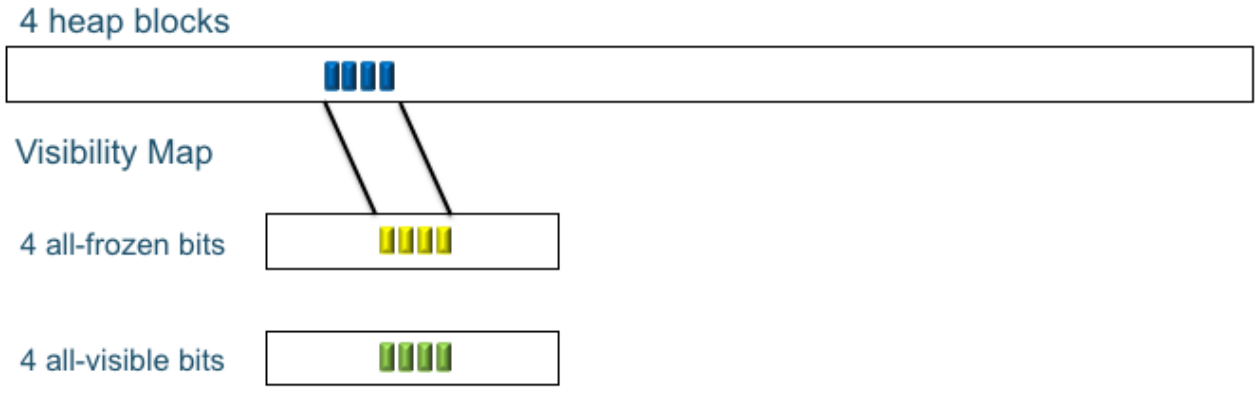
© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Block skipping



© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Block skipping



© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

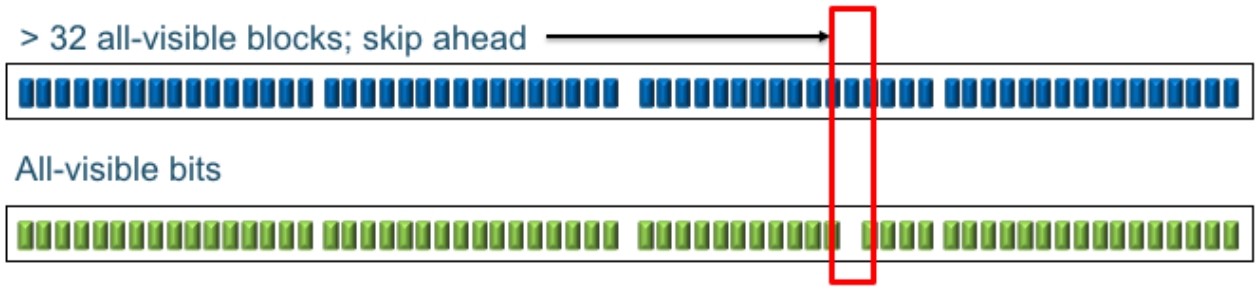
Block skipping

64 heap blocks

All-visible bits

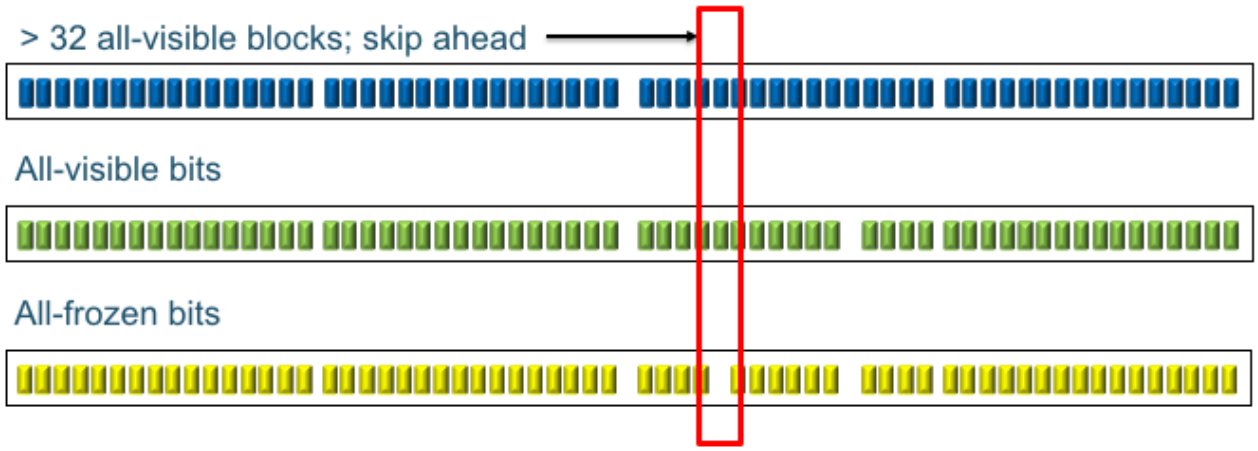
© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Block skipping



© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Block skipping

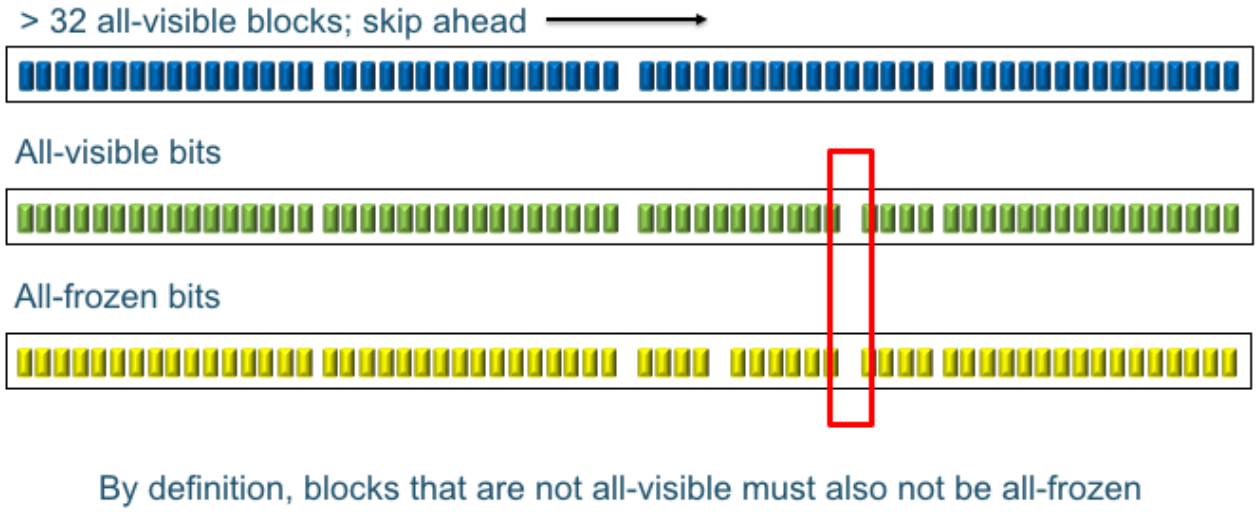


An aggressive vacuum can not skip blocks that are not all-frozen

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

Block skipping



By definition, blocks that are not all-visible must also not be all-frozen

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

Vacuum Process (`lazy_vacuum_rel()`)

- Acquire locks, set limits

SETUP

- Loop through heap, possibly skipping blocks
- Per-page activity
- *Serially* loop through all indexes
- Remove dead tuples from heap
- Vacuum FreeSpaceMap

MAIN LOOP

- Index cleanup
- Attempt relation truncation
- Update `pg_class` info

CLEANUP

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Per-page activity

- Attempt to lock page
  If aggressive vacuum and any tuples need freezing, wait for lock
- Perform HOT pruning (`heap_page_prune()`)
- Scan items on page, deciding how to handle each tuple
- Freeze items (if any)
- If no indexes, vacuum page (`lazy_vacuum_page()`)
- Update visibility map if needed (all-visible & all-frozen)

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Locking

Vacuum requires a cleanup lock on a buffer (`LockBufferForCleanup()`)

Normally multiple backends can hold pins* on multiple buffers for a long time

Vacuum needs exclusive access to the buffer

Non-aggressive vacuums skip any pages they can't lock

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

Every time a backend takes a reference to a buffer, in gets a "pin". See src/backend/storage/buffer/README.

Vacuum Process (`lazy_vacuum_rel()`)

- Acquire locks, set limits

SETUP

- Loop through heap, possibly skipping blocks
- Per-page activity
- *Serially* loop through all indexes
- Remove dead tuples from heap
- Vacuum FreeSpaceMap

MAIN LOOP

- Index cleanup
- Attempt relation truncation
- Update `pg_class` info

CLEANUP

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Loop through all indexes

Index lookups can involve user-defined code
Postgres does not trust that for something as critical as vacuum
Lots of index probes could also be quite expensive

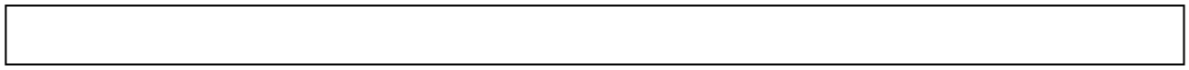
Instead of probing indexes as each tuple to be removed is discovered, vacuum remembers each TupleID

Each index method implements it's own routine for scanning the index, checking each index tuple against the list of remembered heap TupleIDs

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Loop through all indexes

`autovacuum_work_mem`
`maintenance_work_mem`



Scan items on page

Find another page that needs to be vacuumed, remember its TIDs

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Loop through all indexes

autovacuum_work_mem
maintenance_work_mem

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Loop through all indexes

For each index, scan through entire index checking each TID against the list of remembered TIDs (`lazy_vacuum_index()`)

Is TID 2300 in list?
Is TID 45 in list?
Is TID 301 in list?

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Vacuum Process (`lazy_vacuum_rel()`)

- Acquire locks, set limits

SETUP

- Loop through heap, possibly skipping blocks
- Per-page activity
- *Serially* loop through all indexes
- Remove dead tuples from heap
- Vacuum FreeSpaceMap

MAIN LOOP

- Index cleanup
- Attempt relation truncation
- Update `pg_class` info

CLEANUP

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Remove dead tuples from heap

`lazy_vacuum_heap()`

Using the list of TIDs

- Go to each block with tuples to be vacuumed
- Remove the tuples from the block
- Repair page fragmentation
- Update visibility map
- Update FreeSpaceMap

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Vacuum FreeSpaceMap

Update non-leaf data in the FreeSpaceMap

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Vacuum Process (`lazy_vacuum_rel()`)

- Acquire locks, set limits

SETUP

- Loop through heap, possibly skipping blocks
- Per-page activity
- *Serially* loop through all indexes
- Remove dead tuples from heap
- Vacuum FreeSpaceMap

MAIN LOOP

- Index cleanup
- Attempt relation truncation
- Update `pg_class` info

CLEANUP

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Vacuum Process (`lazy_vacuum_rel()`)

- Loop through heap, possibly skipping blocks
- Per-page activity
- *Serially* loop through all indexes
- Remove dead tuples from heap
- Vacuum FreeSpaceMap

MAIN LOOP

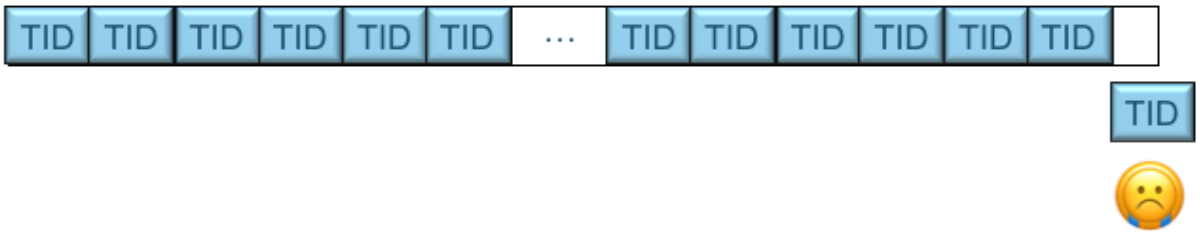
© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

Running out of memory for TIDs

`autovacuum_work_mem`
`maintenance_work_mem`

(values capped at 1GB, TIDs are 6 bytes)



© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Vacuum Process (`lazy_vacuum_rel()`)

- Loop through heap, possibly skipping blocks
- If about to run out of TID memory:
  - *Serially* loop through all indexes
  - Remove dead tuples from heap
  - Vacuum FreeSpaceMap
- Per-page activity
- *Serially* loop through all indexes
- Remove dead tuples from heap
- Vacuum FreeSpaceMap

MAIN LOOP

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

Vacuum Process (`lazy_vacuum_rel()`)

- Loop through heap, possibly skipping blocks
- If about to run out of TID memory:
  - *Serially* loop through all indexes
  - Remove dead tuples from heap
  - Vacuum FreeSpaceMap
- Per-page activity
- *Serially* loop through all indexes
- Remove dead tuples from heap
- Vacuum FreeSpaceMap

OUCH!

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Vacuum Process (`lazy_vacuum_rel()`)

- Acquire locks, set limits

SETUP

- Loop through heap, possibly skipping blocks
- Per-page activity
- *Serially* loop through all indexes
- Remove dead tuples from heap
- Vacuum FreeSpaceMap

MAIN LOOP

- Index cleanup
- Attempt relation truncation
- Update `pg_class` info

CLEANUP

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Index Cleanup (`lazy_cleanup_index()`)

Call index-specific cleanup method

For B-tree, simply cleans up index free space map

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Attempt Truncation

See if truncation would save enough space to be worth-while (`should_attempt_truncation()`)

- Must be at least 1,000 empty blocks
- Number of empty blocks must be >= 16% of heap
- Truncation is not possible if `old_snapshot_threshold` is set

Attempt truncation (`lazy_truncate_heap()`)

1. Abort if new pages added since vacuum started
2. Try to exclusive-lock table (up to 5 seconds)
3. Scan backwards to find last non-empty page. If our lock is blocking someone, go back to step 1
4. If pages were found, actually truncate relation

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Update `pg_class` info (`vac_update_relstats()`)

If `relpages`, `reltuples`, `relallvisible`, `relfrozenxid` or `relminmxid` have changed, then update them in `pg_class`

If this is a vacuum (and not just an analyze), also update `relhasindex`, `relhasrules` and `relhastriggers`

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Vacuum Process (`lazy_vacuum_rel()`)

- Acquire locks, set limits — SETUP

- Loop through heap, possibly skipping blocks
- Per-page activity
- *Serially* loop through all indexes — MAIN LOOP
- Remove dead tuples from heap
- Vacuum FreeSpaceMap

- Index cleanup
- Attempt relation truncation — CLEANUP
- Update `pg_class` info

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

vac_update_datfrozenxid()

vacuum() > vac_update_datfrozenxid()

Update datfrozenxid and datminmxid in pg_database

If new values for either:
- Truncate Commit LOG files (pg_xact/)
- Update internal frozen XID and MXID info

MultiXact files (pg_multixact) are truncated during checkpoint

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

pg_stat_progress_vacuum

```
View "pg_catalog.pg_stat_progress_vacuum"
       Column       |  Type   | Collation | Nullable | Default
--------------------+---------+-----------+----------+---------
 pid                | integer |           |          |
 datid              | oid     |           |          |
 datname            | name    |           |          |
 relid              | oid     |           |          |
 phase              | text    |           |          |
 heap_blks_total    | bigint  |           |          |
 heap_blks_scanned  | bigint  |           |          |
 heap_blks_vacuumed | bigint  |           |          |
 index_vacuum_count | bigint  |           |          |
 max_dead_tuples    | bigint  |           |          |
 num_dead_tuples    | bigint  |           |          |
```

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

pg_stat_progress_vacuum

```
View "pg_catalog.pg_stat_progress_vacuum"
       Column        |  Type   | Collation | Nullable | Default
---------------------+---------+-----------+----------+---------
 pid                 | integer |           |          |
 datid               | oid     |           |          |
 datname             | name    |           |          |
 relid               | oid     |           |          |
 phase               | text    |           |          |
 heap_blks_total     | bigint  |           |          |
 heap_blks_scanned   | bigint  |           |          |
 heap_blks_vacuumed  | bigint  |           |          |
 index_vacuum_count  | bigint  |           |          |
 max_dead_tuples     | bigint  |           |          |
 num_dead_tuples     | bigint  |           |          |
```

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

pg_stat_progress_vacuum

```
View "pg_catalog.pg_stat_progress_vacuum"
      Column       |  Type   | Collation | Nullable | Default
-------------------+---------+-----------+----------+---------
 pid               | integer |           |          |
 datid             | oid     |           |          |
 datname           | name    |           |          |
 relid             | oid     |           |          |
 phase             | text    |           |          |
 heap_blks_total   | bigint  |           |          |
 heap_blks_scanned | bigint  |           |          |
 heap_blks_vacuumed| bigint  |           |          |
 index_vacuum_count| bigint  |           |          |
 max_dead_tuples   | bigint  |           |          |
 num_dead_tuples   | bigint  |           |          |
```

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

pg_stat_progress_vacuum

7 phases for a vacuum

```
initializing
scanning heap
vacuuming indexes
vacuuming heap
cleaning up indexes
truncating heap
performing final cleanup
```

} MAIN LOOP

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

pg_stat_progress_vacuum

- Loop through heap, possibly skipping pages
- Per-page activity
- *Serially* loop through all indexes
- Remove dead tuples from heap
- Vacuum FreeSpaceMap

MAIN LOOP

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

pg_stat_progress_vacuum

• Loop through heap, possibly skipping pages	`scanning heap`
• Per-page activity	`scanning heap`
• *Serially* loop through all indexes	`vacuuming indexes`
• Remove dead tuples from heap	`vacuuming heap`
• Vacuum FreeSpaceMap	`vacuuming heap`

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Vacuum Process (`lazy_vacuum_rel()`)

- Loop through heap, possibly skipping blocks
- If about to run out of TID memory:
  - *Serially* loop through all indexes
  - Remove dead tuples from heap
  - Vacuum FreeSpaceMap
- Per-page activity
- *Serially* loop through all indexes
- Remove dead tuples from heap
- Vacuum FreeSpaceMap

OUCH

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

Vacuum Process (`lazy_vacuum_rel()`)

• Loop through heap, possibly skipping blocks	`scanning heap`
• If about to run out of TID memory:	`scanning heap`
❖ *Serially* loop through all indexes	`vacuuming indexes`
❖ Remove dead tuples from heap	`vacuuming heap`
❖ Vacuum FreeSpaceMap	`vacuuming heap`
• Per-page activity	`scanning heap`
• *Serially* loop through all indexes	`vacuuming indexes`
• Remove dead tuples from heap	`vacuuming heap`
• Vacuum FreeSpaceMap	`vacuuming heap`

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

pg_stat_progress_vacuum

```
View "pg_catalog.pg_stat_progress_vacuum"
       Column        |  Type   | Collation | Nullable | Default
---------------------+---------+-----------+----------+---------
 pid                 | integer |           |          |
 datid               | oid     |           |          |
 datname             | name    |           |          |
 relid               | oid     |           |          |
 phase               | text    |           |          |
 heap_blks_total     | bigint  |           |          |
 heap_blks_scanned   | bigint  |           |          |
 heap_blks_vacuumed  | bigint  |           |          |
 index_vacuum_count  | bigint  |           |          |
 max_dead_tuples     | bigint  |           |          |
 num_dead_tuples     | bigint  |           |          |
```

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

pg_stat_progress_vacuum

heap_blks_total	bigint	# of blocks in table at start of vacuum
heap_blks_scanned	bigint	# of table blocks scanned
heap_blks_vacuumed	bigint	# of table blocks vacuumed

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

pg_stat_progress_vacuum

```
heap_blks_total    | bigint
heap_blks_scanned  | bigint   scanning heap
heap_blks_vacuumed | bigint   vacuuming heap
```

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

pg_stat_progress_vacuum

`index_vacuum_count | bigint`

# of times indexes have been looped through (`vacuuming indexes` phase)

If `index_vacuum_count > 0` and `phase = 'scanning heap'` 😭

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

pg_stat_progress_vacuum

```
View "pg_catalog.pg_stat_progress_vacuum"
       Column        |  Type   | Collation | Nullable | Default
---------------------+---------+-----------+----------+---------
 pid                 | integer |           |          |
 datid               | oid     |           |          |
 datname             | name    |           |          |
 relid               | oid     |           |          |
 phase               | text    |           |          |
 heap_blks_total     | bigint  |           |          |
 heap_blks_scanned   | bigint  |           |          |
 heap_blks_vacuumed  | bigint  |           |          |
 index_vacuum_count  | bigint  |           |          |
 max_dead_tuples     | bigint  |           |          |
 num_dead_tuples     | bigint  |           |          |
```

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

pg_stat_progress_vacuum

```
max_dead_tuples     | bigint
num_dead_tuples     | bigint
```

Size of array (max # of TIDs)

Number of remembered TIDs

autovacuum_work_mem
maintenance_work_mem

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

pg_stat_progress_vacuum

max_dead_tuples ~= num_dead_tuples

autovacuum_work_mem
maintenance_work_mem

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

Autovacuum

Two parts: launcher & worker

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Autovacuum Launcher

Launcher wakes every `autovacuum_naptime` seconds

Prioritizes databases by
- Most in need of XID freeze
- Most in need of MXID freeze
- Least recently autovacuumed, skipping any database less than `autovacuum_naptime` ago

Multiple workers can work on a database

Check count of running autovacuums vs `autovacuum_max_workers`

On RDS check MaximumUsedTransactionIDs in CloudWatch

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Autovacuum Worker

Get list of heap tables & materialized views that need vacuum or analyze

Get list of TOAST tables that need vacuuming

TEMP tables are ignored (or removed)

List of tables is not prioritized in any fashion

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Autovacuum Worker

For each relation
- attempt to get lock
- skip if unavailable (unless freeze is needed)
- call `vacuum()`. `vacuum()` will terminate if it blocks another process, unless aggressive.

Process work items (currently only BRIN summarize)

`vac_update_datfrozenxid()`

exit

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Vacuum Cost Delay

Well documented

Simple explanation: once `(auto)vacuum_cost_limit` is hit, sleep for `(auto)vacuum_cost_delay`. Increasing `_limit` speeds vacuum; increasing `_delay` slows vacuum.

Autovacuum default: 4-8MB/s

Don't slow vacuum too much

On systems where writing is cheaper than reading, set `vacuum_cost_page_dirty` lower than `vacuum_cost_page_miss`

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Adaptive Autovacuum

Makes autovacuum settings more aggressive when maximum transaction ID age gets too high. Resets settings once age drops.

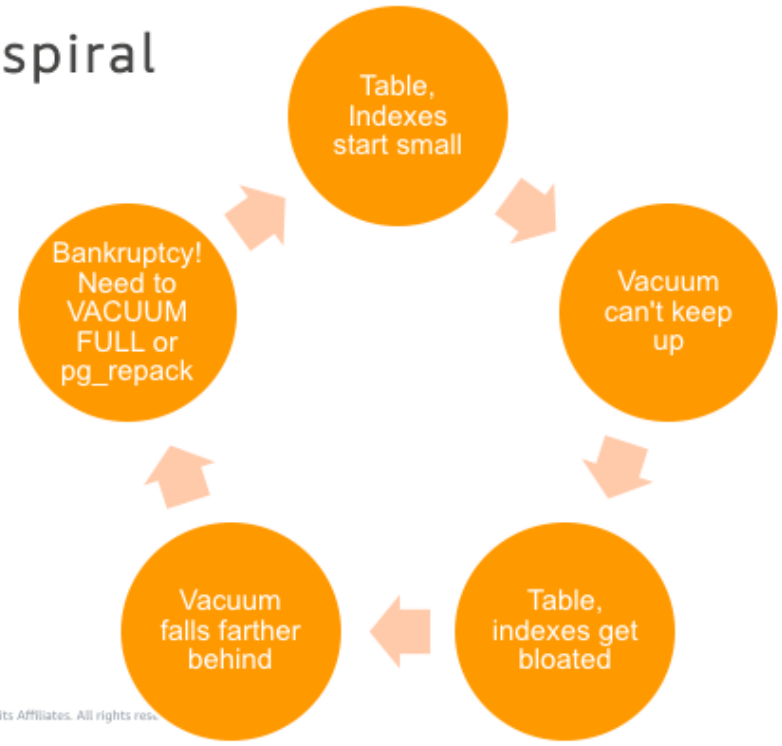
You can monitor via events on the instance.

Reduces risk of instance going read-only to prevent wraparound

Available in RDS Postgres 9.4+

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Death-spiral



© 2019, Amazon Web Services, Inc. or its Affiliates. All rights res

aws