Become a PG_STAT_* (Star)

Chirag Dave

Pr. PostgreSQL SA
AWS

Sami Imseih

Senior Database Engineer
AWS

Typical PostgreSQL challenges

Poor query performance or higher query latency?

Higher I/O wait times?

Slow vacuums/table bloat?

Inconsistent query run time?

High DML latency?

What is PG_STAT_?

- *pg_stat* is a prefix for collection of server activity views
- *(NOT) pg_stats or pg_statistics*
    - Updated by ANALYZE, used by optimizer
    - Not server activity related

- https://www.postgresql.org/docs/current/monitoring-stats.html
- https://wiki.postgresql.org/wiki/Monitoring

Dynamic views

- *PID* column to identify a process
- An entry per connection
- Entry disappears when connection closes

```
pg_stat_activity
pg_stat_gssapi
pg_stat_replication
pg_stat_ssl
pg_stat_subscription
pg_stat_wal_receiver
```

```
pg_stat_progress_analyze
pg_stat_progress_basebackup
pg_stat_progress_cluster
pg_stat_progress_copy
pg_stat_progress_create_index
pg_stat_progress_vacuum
```

Dynamic views

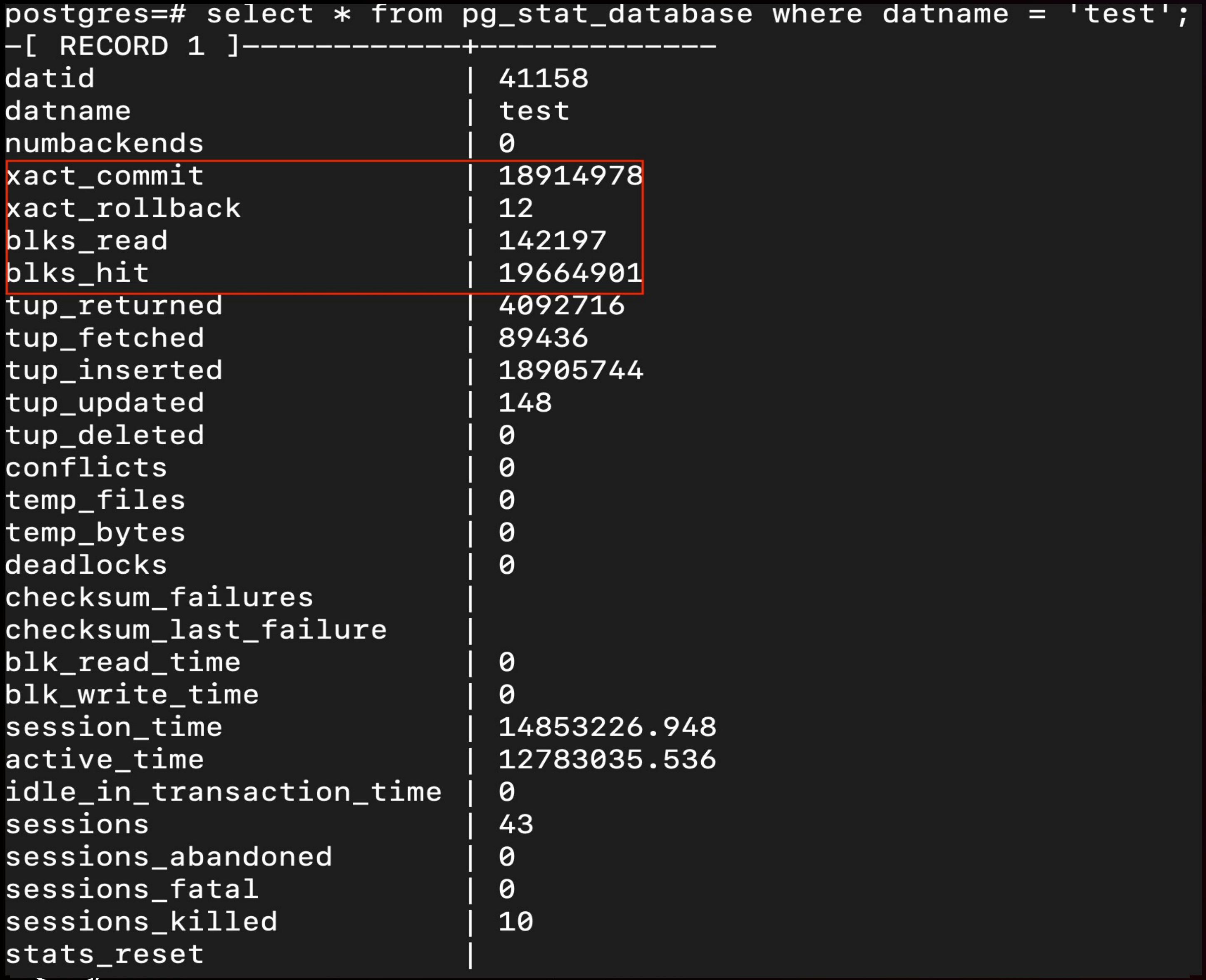
```
SELECT * FROM pg_stat_activity
WHERE state = 'active'
AND pid <> pg_backend_pid();

-[ RECORD 1 ]----+---------------------------------------
datid            | 5
datname          | postgres
pid              | 19182
leader_pid       |
usesysid         | 10
usename          | postgres
application_name | psql
client_addr      |
client_hostname  |
client_port      | -1
backend_start    | 2024-04-08 01:51:57.306027+00
xact_start       | 2024-04-08 01:52:25.990124+00
query_start      | 2024-04-08 01:52:25.990124+00
state_change     | 2024-04-08 01:52:25.990128+00
wait_event_type  | IO
wait_event       | DataFileRead
state            | active
backend_xid      |
backend_xmin     | 759
query_id         | 6214167543274274 50
query            | select count(*) from demo a, demo b;
backend_type     | client backend
```

```
SELECT * FROM pg_stat_progress_vacuum;

-[ RECORD 1 ]--------+--------------
pid                  | 19782
datid                | 5
datname              | postgres
relid                | 16407
phase                | scanning heap
heap_blks_total      | 192308
heap_blks_scanned    | 6
heap_blks_vacuumed   | 0
index_vacuum_count   | 1
max_dead_tuple_bytes | 67108864
dead_tuple_bytes     | 0
indexes_total        | 5
indexes_processed    | 3
```

Cumulative Statistics



```
postgres=# select * from pg_stat_database where datname = 'test';
-[ RECORD 1 ]------------+--------------
datid                    | 41158
datname                  | test
numbackends              | 0
xact_commit              | 18914978
xact_rollback            | 12
blks_read                | 142197
blks_hit                 | 19664901
tup_returned             | 4092716
tup_fetched              | 89436
tup_inserted             | 18905744
tup_updated              | 148
tup_deleted              | 0
conflicts                | 0
temp_files               | 0
temp_bytes               | 0
deadlocks                | 0
checksum_failures        |
checksum_last_failure    |
blk_read_time            | 0
blk_write_time           | 0
session_time             | 14853226.948
active_time              | 12783035.536
idle_in_transaction_time | 0
sessions                 | 43
sessions_abandoned       | 0
sessions_fatal           | 0
sessions_killed          | 10
stats_reset              |
```

- All backends increment the values
- Values constantly increasing

Cumulative Statistics

Cluster-wide

8.3+	pg_stat_bgwriter
12+	pg_stat_archiver
13+	pg_stat_slru
14+	pg_stat_wal
16+	pg_stat_io
16+	pg_stat_replication_slots
16+	pg_stat_recovery_prefetch
16+	pg_stat_subscription_stats
17+	pg_stat_checkpointer

Per-Relation

7.2+	pg_stat_all_tables
7.2 +	pg_stat_all_indexes
7.2 +	pg_statio_all_tables
7.2 +	pg_statio_all_indexes
7.2 +	pg_statio_all_sequences

Per-Function

8.4+	pg_stat_user_functions

Per-Database

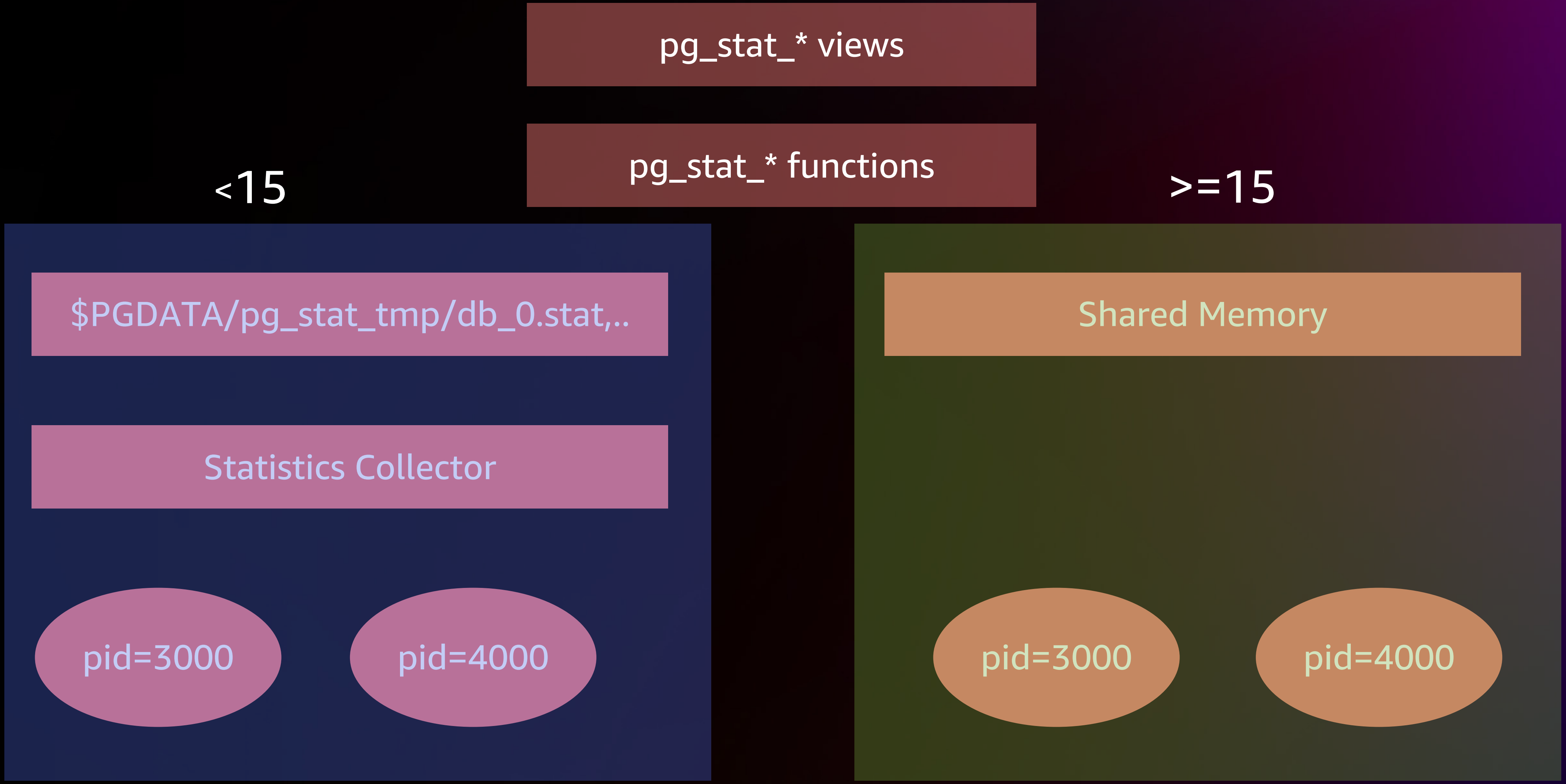
7.2+	pg_stat_database
9.1+	pg_stat_database_conflicts

Per-Statement

8.4+	pg_stat_statements

* Not Cumulative Statistics System/Core Postgres
* Add to *shared_preload_libraries*
* CREATE EXTENSION pg_stat_statements

Cumulative Statistics



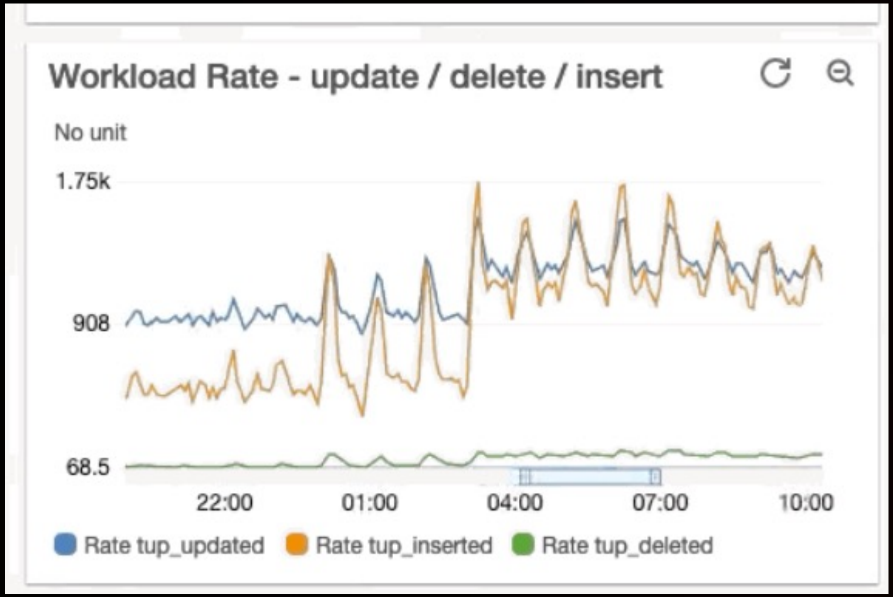
Cumulative Statistics

- Delta Metric to find changes during a time interval
- Calculates the rate of change
- 13022 – 3016 = 10006 rows ever 10 seconds
- 10006 / 10 ≈ 1000 rows per second

```
SELECT now() timestamp, tup_inserted from
pg_stat_database where datname = 'postgres’;
```

```
          timestamp              | tup_inserted
---------------------------------+--------------
 2024-04-08 02:52:28.50213+00 |         3016
(1 row)
```

```
          timestamp               | tup_inserted
----------------------------------+--------------
 2024-04-08 02:52:38.502136+00 |        13022
(1 row)
```

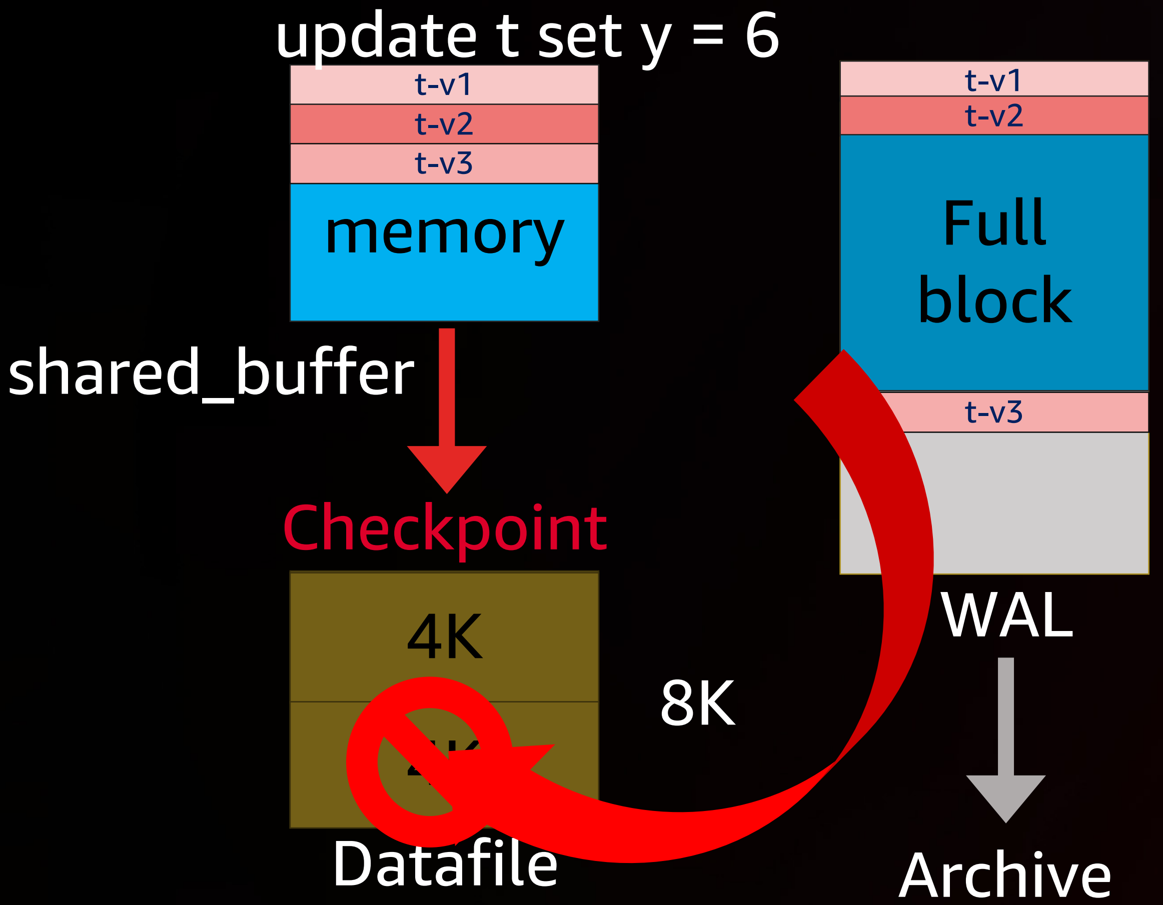


- https://github.com/awslabs/pg-counter-metrics

Cumulative Statistics System

- Background Writer, Checkpoint
- Vacuum /Autovacuum
- DML activity
- HOT Updates
- Index/Table Access
- I/O

PostgreSQL I/O



1. **Write it to the WAL log**
2. **Update Shared Buffers**
3. **Checkpoint**
4. **Write to the disk**

- Dirty Buffer Flushing
  - Checkpoint
    - Too often: More Full Page Writes
    - Too far: longer recovery times
  - Background Writer

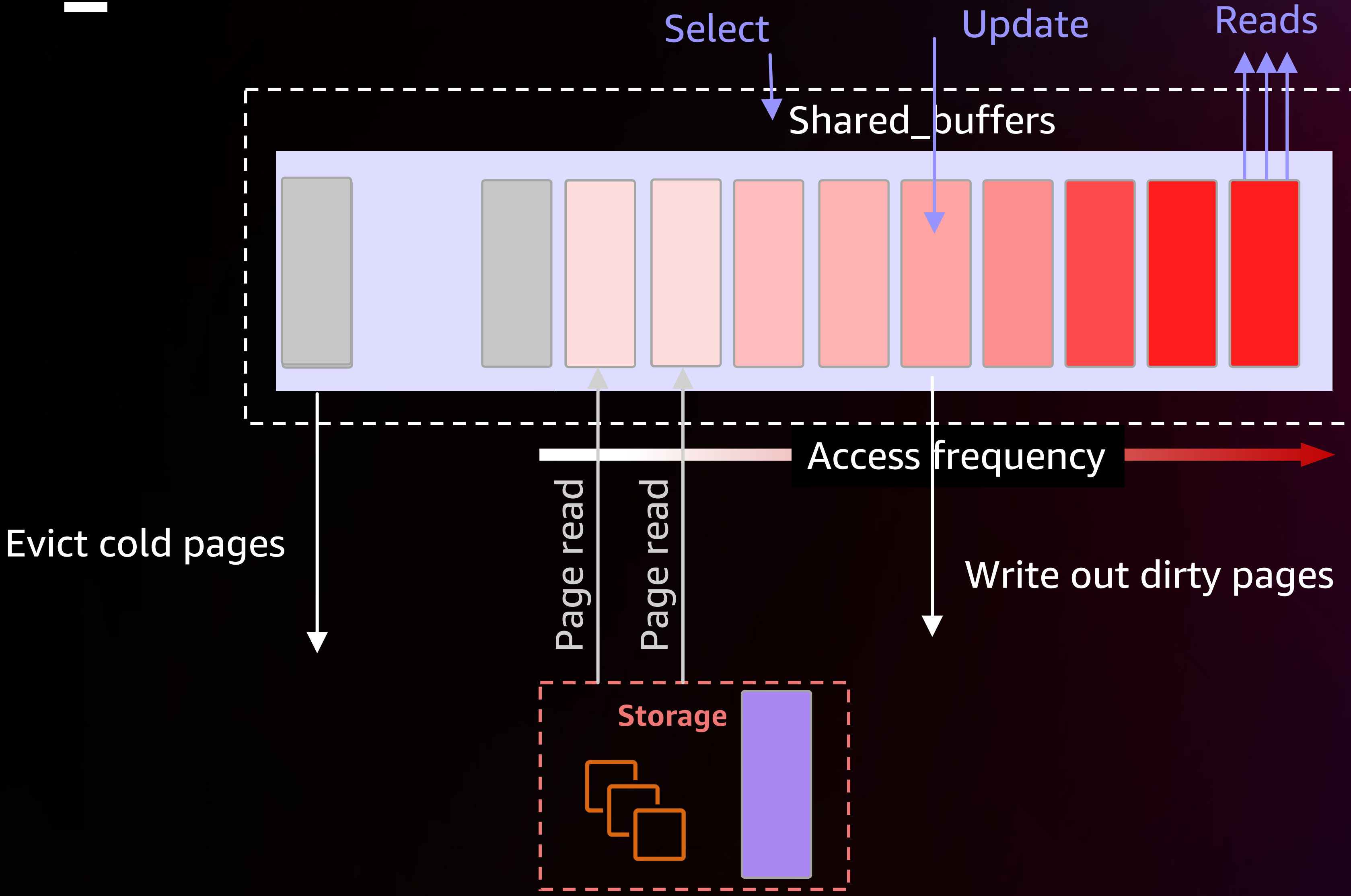
pg_stat_wal (PG14)

```
postgres=# select * from pg_stat_wal;
-[ RECORD 1 ]----+------------------------------
wal_records      | 414
wal_fpi          | 31
wal_bytes        | 207099
wal_buffers_full | 0
wal_write        | 27
wal_sync         | 27
wal_write_time   | 0
wal_sync_time    | 0
stats_reset      | 2024-04-14 02:17:21.429386+00
```

wal_fpi → WAL records due to a checkpoint

wal_buffers_full -→ wal_buffers setting is set too low

shared_buffers



background writer

checkpointer

Ring Buffers - Buffer Access Strategy

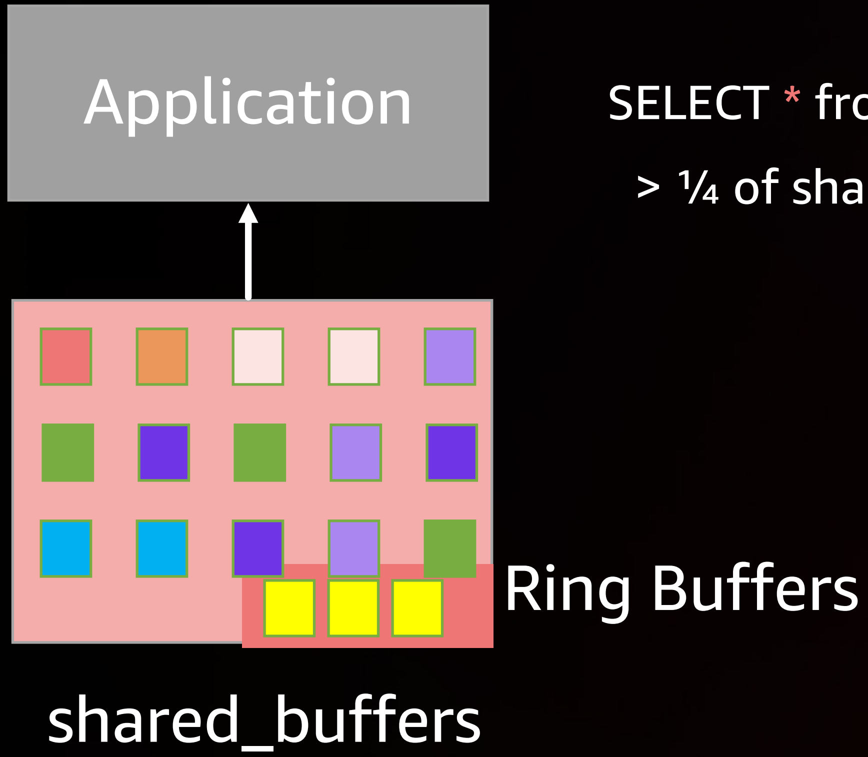
Prevent **cache thrashing and maintains a higher cache hit ratio**

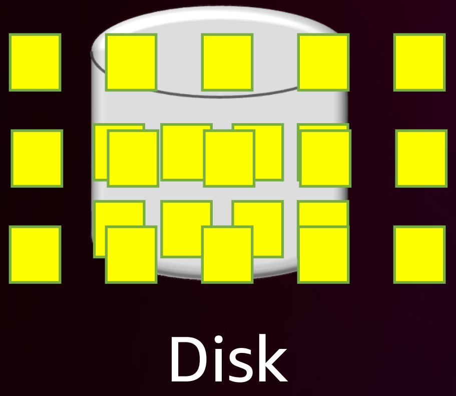
bulk-reading	**256 KB**
bulk-writing	**16 MB**
vacuum-processing	**256 KB**

New in PG16: **vacuum_buffer_usage_limit**

SELECT * from large_table;

> ¼ of shared_buffers





Background Writer, Checkpoint

```
postgres=# select * from pg_stat_bgwriter;
-[ RECORD 1 ]---------+------------------------------
checkpoints_timed     | 0     > checkpoint_timeout
checkpoints_req       | 6     > max_wal_size
checkpoint_write_time | 72441
checkpoint_sync_time  | 3838
buffers_checkpoint    | 6380  dirty buffers written by checkpointer
buffers_clean         | 16248 dirty buffers written by background writer
maxwritten_clean      | 161
buffers_backend       | 520106 dirty buffers written by backend
buffers_backend_fsync | 0
buffers_alloc         | 465918
stats_reset           | 2024-04-09 13:54:18.683023-05
```

- Keep buffers_backend close to 0 as possible
- 17+, this info will be spread between *pg_stat_bgwriter*, *pg_stat_checkpointer*, *pg_stat_io*

DML Activity

```
select * from pg_stat_all_tables
where relname = 'pgbench_accounts';
```

- Tracks # of inserts/updates/deletes per table

```
n_tup_ins                | 0
n_tup_upd                | 288202
n_tup_del                | 0
```

HOT Updates

- Maximize n_tup_hot_update
- Minimize n_tup_newpage_upd (16+)
- Reduce FILLFACTOR for heavily updated tables
- Drop unused Indexes

```
n_tup_ins            | 0
n_tup_upd            | 288202
n_tup_del            | 0
n_tup_hot_upd        | 288159
n_tup_newpage_upd    | 43
```

Vacuum

- track_counts = ON (DEFAULT)
  - pg_stat_all_indexes
  - pg_stat_all_sequences

- Autovacuum/vacuum metrics

```
select * from pg_stat_all_tables where relname =
'pgbench_accounts';
-[ RECORD 1 ]-------+------------------------------
relid               | 16391
schemaname          | public
relname             | pgbench_accounts
seq_scan            | 0
last_seq_scan       |
seq_tup_read        | 0
idx_scan            | 8253719
last_idx_scan       | 2024-04-07 14:46:15.527926+00
idx_tup_fetch       | 8253719
n_tup_ins           | 0
n_tup_upd           | 4126862
n_tup_del           | 0
n_tup_hot_upd       | 4064855
n_tup_newpage_upd   | 62007
n_live_tup          | 999905
n_dead_tup          | 134650
n_mod_since_analyze | 80551
n_ins_since_vacuum  | 0
last_vacuum         |
last_autovacuum     | 2024-04-07 14:45:50.843147+00
last_analyze        |
last_autoanalyze    | 2024-04-07 14:45:53.843147+00
vacuum_count        | 0
autovacuum_count    | 0
analyze_count       | 0
autoanalyze_count   | 18
```

Vacuum

- Metrics used directly by autovacuum launcher
- Crash recovery/pg_stat_reset
  - Wipes out the data
  - May delay autovacuum/autoanalyze

```
/*
 * If we found stats for the table, and autovacuum is currently enabled,
 * make a threshold-based decision whether to vacuum and/or analyze.  If
 * autovacuum is currently disabled, we must be here for anti-wraparound
 * vacuuming only, so don't vacuum (or analyze) anything that's not being
 * forced.
 */
if (PointerIsValid(tabentry) && AutoVacuumingActive())
{
        reltuples = classForm->reltuples;
        vactuples = tabentry->dead_tuples;
        instuples = tabentry->ins_since_vacuum;
        anltuples = tabentry->mod_since_analyze;

        /* If the table hasn't yet been vacuumed, take reltuples as zero */
        if (reltuples < 0)
                reltuples = 0;

        vacthresh = (float4) vac_base_thresh + vac_scale_factor * reltuples;
        vacinsthresh = (float4) vac_ins_base_thresh + vac_ins_scale_factor * reltuples;
        anlthresh = (float4) anl_base_thresh + anl_scale_factor * reltuples;
```

Vacuum

- pg_stat_progress_vacuum.index_vacuum_count increases every ”vacuum index cleanup” cycle

- In Postgres 16 and below:
    - Scan the table for dead rows, and store the dead rows in autovacuum_work_mem/maintenance_work_mem (179 million dead rows max)
    - Vacuum the indexes
    - Repeat
    - Super expensive if multi-index vacuum cycles are required.

- In Postgres17:
    - Multi-index cycles will become less likely thanks to https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=30e144287a

Index/Table Access

- last_seq_scan and last_idx_scan (16+)
- For OLTP, seq_scan should be close to 0

```
postgres=# select * from pg_stat_all_tables where relname =
'demo';
-[ RECORD 1 ]-------+-------------------------------
relid               | 65572
schemaname          | public
relname             | demo
seq_scan            | 0
last_seq_scan       | 2024-04-07 17:08:22.454168+00
seq_tup_read        | 0
idx_scan            | 5
last_idx_scan       | 2024-04-07 17:09:22.454168+00
idx_tup_fetch       | 37199829
```

```
postgres=# select * from pg_stat_all_indexes where
indexrelid = 'demo_id'::regclass::oid;
-[ RECORD 1 ]-+-----------------------------
relid         | 65572
indexrelid    | 65575
schemaname    | public
relname       | demo
indexrelname  | demo_id
idx_scan      | 5
last_idx_scan | 2024-04-07 17:09:22.454168+00
idx_tup_read  | 61999715
idx_tup_fetch | 37199829
```

Index/Table Access

- idx_tup_fetch
  - Index scan visits a heap
  - Projecting columns not in index

```
postgres=# select * from pg_stat_all_tables where relname =
'demo';
-[ RECORD 1 ]-------+--------------------------------
relid               | 65572
schemaname          | public
relname             | demo
seq_scan            | 0
last_seq_scan       |
seq_tup_read        | 0
idx_scan            | 5
last_idx_scan       | 2024-04-07 17:09:22.454168+00
idx_tup_fetch       | 37199829
```

```
EXPLAIN (ANALYZE) SELECT id FROM demo WHERE id = 'c40d8806-
c87e-4942-8d35-ce79819ba68c';
                             QUERY PLAN

---------------------------------------------------------------------
Index Only Scan using demo_pkey on demo  (cost=0.43..4.45 rows=1
width=16) (actual time=0.045..0.047 rows
=1 loops=1)
   Index Cond: (id = 'c40d8806-c87e-4942-8d35-ce79819ba68c'::uuid)
   Heap Fetches: 0
 Planning Time: 0.123 ms
 Execution Time: 0.077 ms
(5 rows)
```

Index/Table Access

- Index-only scans minimize heap fetches
    - If idx_tup_fetch high, VACUUM more aggressive
    - Visibility Map not up-to-date

```
postgres=# select * from pg_stat_all_indexes where
indexrelid = 'demo_id'::regclass::oid;
-[ RECORD 1 ]-+------------------------------
relid         | 65572
indexrelid    | 65575
schemaname    | public
relname       | demo
indexrelname  | demo_id
idx_scan      | 5
last_idx_scan | 2024-04-07 17:09:22.454168+00
idx_tup_read  | 61999715
idx_tup_fetch | 37199829
```

```
SELECT tup_returned, tup_fetched from
pg_stat_database where datname =
'postgres';
-[ RECORD 1 ]+---------
tup_returned | 62284143
tup_fetched  | 37204480
```

I/O

Pre 16, Cumulative Statistics did not make a distinction for Buffer Access Strategies

New in PG16 -→ **pg_stat_io**

- CONTEXT column = Buffer Access Strategy
- Improves cache hit ratio calculation

I/O

- Context = Normal, Bulkread, Bulkwrite, Vacuum

```
postgres=# \d pg_statio_all_tables
          View "pg_catalog.pg_statio_all_tables"
     Column       |  Type   | Collation | Nullable | Default
------------------+---------+-----------+----------+---------
 relid            | oid     |           |          |
 schemaname       | name    |           |          |
 relname          | name    |           |          |
 heap_blks_read   | bigint  |           |          |
 heap_blks_hit    | bigint  |           |          |
 idx_blks_read    | bigint  |           |          |
 idx_blks_hit     | bigint  |           |          |
 toast_blks_read  | bigint  |           |          |
 toast_blks_hit   | bigint  |           |          |
 tidx_blks_read   | bigint  |           |          |
 tidx_blks_hit    | bigint  |           |          |
```

```
postgres=# \d pg_statio_all_indexes
         View "pg_catalog.pg_statio_all_indexes"
    Column      |  Type   | Collation | Nullable | Default
----------------+---------+-----------+----------+---------
 relid          | oid     |           |          |
 indexrelid     | oid     |           |          |
 schemaname     | name    |           |          |
 relname        | name    |           |          |
 indexrelname   | name    |           |          |
 idx_blks_read  | bigint  |           |          |
 idx_blks_hit   | bigint  |           |          |
```

```
postgres=# \d pg_stat_io
                        View "pg_catalog.pg_stat_io"
     Column     |           Type           | Collation | Nullable |
Default
----------------+--------------------------+-----------+----------+---
------
 backend_type   | text                     |           |          |
 object         | text                     |           |          |
 context        | text                     |           |          |
 reads          | bigint                   |           |          |
 read_time      | double precision         |           |          |
 writes         | bigint                   |           |          |
 write_time     | double precision         |           |          |
 writebacks     | bigint                   |           |          |
 writeback_time | double precision         |           |          |
 extends        | bigint                   |           |          |
 extend_time    | double precision         |           |          |
 op_bytes       | bigint                   |           |          |
 hits           | bigint                   |           |          |
 evictions      | bigint                   |           |          |
 reuses         | bigint                   |           |          |
 fsyncs         | bigint                   |           |          |
 fsync_time     | double precision         |           |          |
 stats_reset    | timestamp with time zone |           |          |
```

```
postgres=# \d pg_stat_database
              View "pg_catalog.pg_stat_database"
     Column      |       Type       | Collation | Nullable | Default
-----------------+------------------+-----------+----------+---------
 datid           | oid              |           |          |
 datname         | name             |           |          |
....
 blk_read_time   | double precision |           |          |
 blk_write_time  | double precision |           |          |
```

I/O - Demo

```
DROP TABLE IF EXISTS demo;
CREATE TABLE demo ( id int, c1 text );
INSERT INTO demo SELECT n FROM generate_series(1, 8000000) as n;
```

```
SELECT
blks_read,
blks_hit,
ROUND(blks_hit/(blks_hit+blks_read::numeric)*100,
2)
FROM pg_stat_database
WHERE datname = 'postgres';

 blks_read | blks_hit | round
-----------+----------+-------
        82 |     1567 | 95.03
(1 row)
```

```
SELECT
backend_type,
context,
reads,
hits,
ROUND(hits/(reads+hits)::numeric * 100, 2)
FROM pg_stat_io
WHERE context = 'normal'
AND backend_type = 'client backend'
AND object = 'relation';

  backend_type  | context | reads | hits | round
----------------+---------+-------+------+-------
 client backend | normal  |    86 | 1578 | 94.83
(1 row)
```

I/O - Demo

VACUUM demo;

```
SELECT
blks_read,
blks_hit,
ROUND(
blks_hit/(blks_hit+blks_read::numeric)*100, 2)
FROM pg_stat_database
WHERE datname = 'postgres';

 blks_read    | blks_hit | round
-----------+----------+----------
        19657 |    53499 | 73.13
(1 row)
```

```
SELECT
backend_type,
context,
reads,
hits,
ROUND(hits/(reads+hits)::numeric * 100, 2)
FROM pg_stat_io
WHERE context = 'normal'
AND backend_type = 'client backend'
AND object = 'relation';

  backend_type  | context | reads | hits | round
----------------+---------+------+-----+-------
 client backend | normal  |   193 | 37438 | 99.49
(1 row)
```

I/O - Demo

SELECT COUNT(*) FROM demo;

```
select
  blks_read,
  blks_hit,
  ROUND(
  blks_hit/
  (blks_hit+blks_read::numeric)*100, 2) chr
from pg_stat_database
where datname = 'postgres';

blks_read  | blks_hit  | chr
-----------+-----------+-------
     58593 |   122667  | 67.67
(1 row)
```

```
select
  backend_type,
  context,
  reads,
  hits,
  ROUND(hits/(reads+hits)::numeric * 100, 2) chr
from
  pg_stat_io
where context = 'bulkread'
and backend_type = 'client backend'
and object = 'relation';

backend_type    | context   | reads | hits | chr
----------------+-----------+-------+------+-------
 client backend | bulkread  |  6542 | 5371 | 45.09
(1 row)
```

```
select
  backend_type,
  context,
  reads,
  hits,
  ROUND(hits/(reads+hits)::numeric * 100, 2) chr
from
  pg_stat_io
where context = 'normal'
and backend_type = 'client backend'
and object = 'relation';

backend_type    | context | reads | hits  | round
----------------+---------+-------+-------+-------
 client backend | normal  |   247 | 73745 | 99.67
(1 row)
```

Takeaways

- pg_stat_* views gives you observability into PG's workload
- Postgres 15 and 16
  - Improved stability of the system, shared memory vs background worker
  - PG_STAT_IO improves visbility into I/O contexts = better cache hit ratio calculations

Ready to upgrade!

Thank you!