

# PostgreSQL Monitoring Day

with Zabbix &  
Postgres Professional



**Anastasia Lubennikova**

Sr. Database Developer  
(Postgres Professional),  
PostgreSQL contributor



**ZABBIX**

PostgresPro

## New Monitoring- Related Features in PostgreSQL 13

7 PM CET

# About me

- PostgreSQL contributor since 2015
  - Index-only scan for GiST
  - Microvacuum for GiST
  - B-tree INCLUDE clause
  - B-tree deduplication
  - pg\_probackup co-maintainer
- Tier 3 support for PostgreSQL and PostgresPro solutions
- Education and mentoring

# Agenda for today's talk

- Query sampling
- WAL usage statistics
- Progress reporting
- New system views

# Query sampling

- `log_statement`
- `log_min_duration_statement`

New in Pg 13:

- `log_statement_sample_rate`
- `log_min_duration_sample`

Always set `log_min_duration_sample < log_min_duration_statement`.

# Query sampling (example)

```
pgbench -i -s 100 postgres
```

```
pgbench -c 10 -t 100000 postgres
```

	time	logfile size
<code>log_min_duration_statement = 0</code>	5m 10 s	1 GB
<code>log_min_duration_statement = 1000</code> <code>log_min_duration_sample = 0</code> <b><code>log_statement_sample_rate = 0.1</code></b>	4m 50 s	87 MB
<code>log_min_duration_statement = 1000</code>	2m 40 s	2 KB

# Prepared statement parameter logging

- `log_parameter_max_length`
- `log_parameter_max_length_on_error`

```
ERROR: division by zero
```

```
STATEMENT: SELECT 1/$1
```

```
SET log_parameter_max_length_on_error = 1024
```

```
ERROR: division by zero
```

```
CONTEXT: extended query with parameters: $1 = '0'
```

```
STATEMENT: SELECT 1/$1
```

# WAL statistics in EXPLAIN and auto\_explain

```
CREATE TABLE test (id int, data char(1000));
```

```
EXPLAIN (ANALYZE, WAL)
```

```
INSERT INTO test SELECT i, i::text FROM generate_series(1,1000) as i;
```

```
Insert on test (cost=0.00..17.50 rows=1000 width=4008)
```

```
(actual time=8.219..8.221 rows=0 loops=1)
```

```
WAL: records=1001 fpi=1 bytes=1071241
```

```
-> Function Scan on generate_series i
```

```
(cost=0.00..17.50 rows=1000 width=4008)
```

```
(actual time=0.234..1.027 rows=1000 loops=1)
```

```
Planning Time: 0.096 ms
```

```
Execution Time: 8.284 ms
```

# WAL statistics in autovacuum log

```
SET log_autovacuum_min_duration = 0
```

```
automatic vacuum of table "postgres.public.test": index scans: 0  
pages: 3428 removed, 0 remain, 0 skipped due to pins, 0 skipped frozen  
tuples: 6001 removed, 0 remain, 0 are dead but not yet removable,  
oldest xmin: 6210665  
buffer usage: 10295 hits, 2 misses, 3 dirtied  
avg read rate: 0.167 MB/s, avg write rate: 0.251 MB/s  
system usage: CPU: user: 0.01 s, system: 0.01 s, elapsed: 0.09 s  
WAL usage: 7719 records, 3 full page images, 518517 bytes
```



# WAL statistics in pg\_stat\_statements

```
SELECT query, wal_records, wal_fpi, wal_bytes  
FROM pg_stat_statements;
```

```
query          | explain (analyze, wal)  
               | insert into test select i, i::text  
               | from generate_series(1,1000) as i  
  
wal_records | 1001  
  
wal_fpi    | 1  
  
wal_bytes  | 1071241
```

# Planning time in pg\_stat\_statements

Breaks backward compatibility!

```
SET pg_stat_statements.track_planning TO true;
```

```
SELECT query, total_plan_time, min_plan_time, max_plan_time,  
mean_plan_time FROM pg_stat_statements;
```

```
query | select from test t1 inner join test t2 on t1.id = t2.id
```

```
total_plan_time | 1.280523
```

```
min_plan_time | 0.098761
```

```
max_plan_time | 0.250417
```

```
mean_plan_time | 0.2134205
```

# Leader pid in pg\_stat\_activity

```
SELECT query, leader_pid,  
       array_agg(pid) filter(WHERE leader_pid != pid) AS members  
FROM pg_stat_activity WHERE leader_pid IS NOT NULL  
GROUP BY query, leader_pid;
```

**query** | select \* from test;

**leader\_pid** | 31630

**members** | {32269,32268}

[https://rjuju.github.io/postgresql/2020/02/06/new-in-pg13-leader\\_pid.html](https://rjuju.github.io/postgresql/2020/02/06/new-in-pg13-leader_pid.html)

# Progress reporting

- [pg\\_stat\\_progress\\_basebackup](#)
- [pg\\_stat\\_progress\\_analyze](#)

# pg\_shmem\_allocations

- [pg\\_shmem\\_allocations](#)
- [pg\\_stat\\_slru](#)

# Wait events changes

Breaks backward compatibility!

Wait events renamed to improve consistency:

Hash/Batch/Allocating → HashBatchAllocate

ControlFileLock → ControlFile

clog → XactBuffer

AsyncCtlLock → NotifySLRU

New wait events:

VacuumDelay

BackupWaitWalArchive, RecoveryPause

RecoveryConflictSnapshot, RecoveryConflictTablespace

# track\_activity\_query\_size

- Updated upper limit to 1MB

The 'max\_connections \* track\_activity\_query\_size' amount of memory is allocated at database startup time.

[https://www.cybertec-postgresql.com/en/fixing-track\\_activity\\_query\\_size-in-postgresql-conf/](https://www.cybertec-postgresql.com/en/fixing-track_activity_query_size-in-postgresql-conf/)

# New in PostgreSQL 13

1. Query sampling
2. CONTEXT for failure of parameterized queries
3. EXPLAIN WAL statistics
4. autovacuum WAL statistics
5. Per-statement WAL statistics
6. pg\_stat\_statements: Planning Time
7. pg\_stat\_activity: leader\_pid for Parallel Query
8. pg\_stat\_progress\_basebackup
9. pg\_stat\_progress\_analyze
10. pg\_shmem\_allocations
11. pg\_stat\_slru
12. Additional & renamed wait events
13. track\_activity\_query\_size limit increase



Thank you for attention!

[a.lubennikova@postgrespro.com](mailto:a.lubennikova@postgrespro.com)

 **PostgresPro**

<https://postgrespro.com/>

# New things to monitor

- B-tree index bloat

- Previously it was an important thing to monitor.

New optimization is intended to eliminate bloat.

So, this metric is not relevant anymore:

[https://wiki.postgresql.org/wiki/Show\\_database\\_bloat](https://wiki.postgresql.org/wiki/Show_database_bloat)

- Parallel VACUUM resource consumption

- Disk based hash aggregation

- hash\_mem\_multiplier