PostgreSQL Monitoring Day

with Zabbix & Postgres Professional

ONLINE

New Monitoring-Related Features in PostgreSQL 13

7 PM CET

Anastasia Lubennikova
Sr. Database Developer (Postgres Professional), PostgreSQL contributor
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
```

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Time</th>
<th>Logfile Size</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>log_min_duration_statement = 0</code></td>
<td>5m 10 s</td>
<td>1 GB</td>
</tr>
<tr>
<td><code>log_min_duration_statement = 1000</code></td>
<td>4m 50 s</td>
<td>87 MB</td>
</tr>
<tr>
<td><code>log_min_duration_sample = 0</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>log_statement_sample_rate = 0.1</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>log_min_duration_statement = 1000</code></td>
<td>2m 40 s</td>
<td>2 KB</td>
</tr>
</tbody>
</table>
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
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
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`

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

<table>
<thead>
<tr>
<th>query</th>
<th>explain (analyze, wal)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>insert into test select i, i::text</td>
</tr>
<tr>
<td></td>
<td>from generate_series(1,1000) as i</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>wal_records</th>
<th>1001</th>
</tr>
</thead>
<tbody>
<tr>
<td>wal_fpi</td>
<td>1</td>
</tr>
<tr>
<td>wal_bytes</td>
<td>1071241</td>
</tr>
</tbody>
</table>
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;
```

<table>
<thead>
<tr>
<th>query</th>
<th>select from test t1 inner join test t2 on t1.id = t2.id</th>
</tr>
</thead>
<tbody>
<tr>
<td>total_plan_time</td>
<td>1.280523</td>
</tr>
<tr>
<td>min_plan_time</td>
<td>0.098761</td>
</tr>
<tr>
<td>max_plan_time</td>
<td>0.250417</td>
</tr>
<tr>
<td>mean_plan_time</td>
<td>0.2134205</td>
</tr>
</tbody>
</table>
### Leader pid in pg_stat_activity

```sql
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;
```

<table>
<thead>
<tr>
<th>query</th>
<th>select * from test;</th>
</tr>
</thead>
<tbody>
<tr>
<td>leader_pid</td>
<td>31630</td>
</tr>
<tr>
<td>members</td>
<td>{32269,32268}</td>
</tr>
</tbody>
</table>

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.

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

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

- Parallel VACUUM resource consumption

- Disk based hash aggregation
  - hash_mem_multiplier