PostgreSQL monitoring

introduction

Aleš Zelený

About… dirty blocks cleaning

PostgreSQL monitoring

- Nezbytná služba pro DBA k zajištění provozu
  - místo na svazcích
  - replikace
- Zabbix, HP Openview, Munin, Grafana...
- Nástroj zpětné vazby pro vývojáře
  - neefektivní dotazy
  - velikost tabulek a indexů
  - počty transakcí
  - ...

Zdroje informací

- Operační systém
- Interní statistiky PostgreSQL a LOG soubor instance
- Rozšíření v contributed balíčcích
  - pg_stat_statements („must have“)
- Ostatní rozšíření/extenze
  - pg_stat_kcache
  - powa
- Externí nástroje
  - check_postgres
  - pgBadger

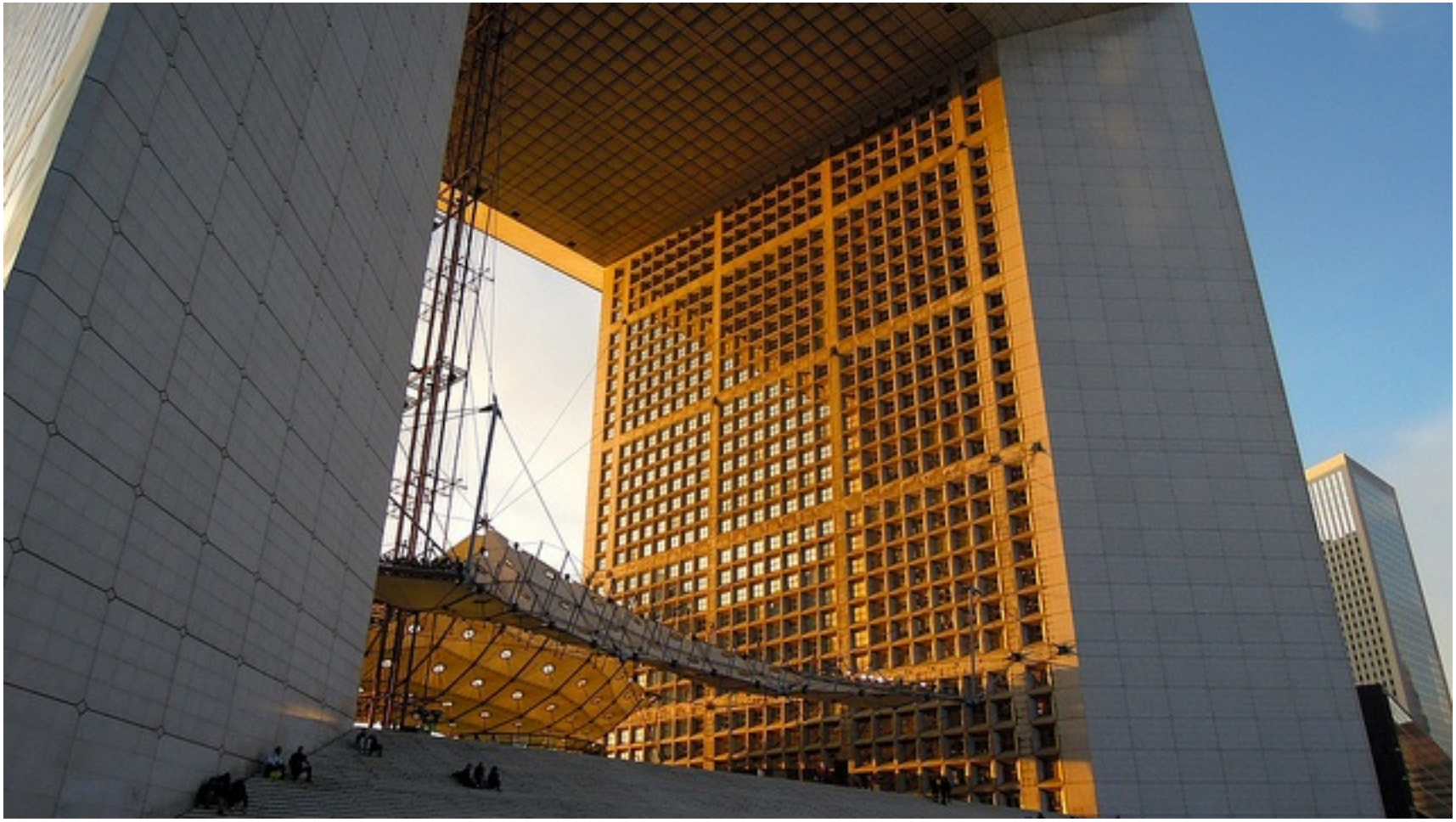
Data z operačního systému

- Procesor
  - dedikovaný DB server ?
  - user, system, IO wait, Steal Time (AWS Tx instancces)
- Paměť
  - used, buffered, cached, swap
- IO
  - reads, writes, Bytes, IO time
- Síť
- sadc (sar), vmstat, dstat ...

Kam s ním?

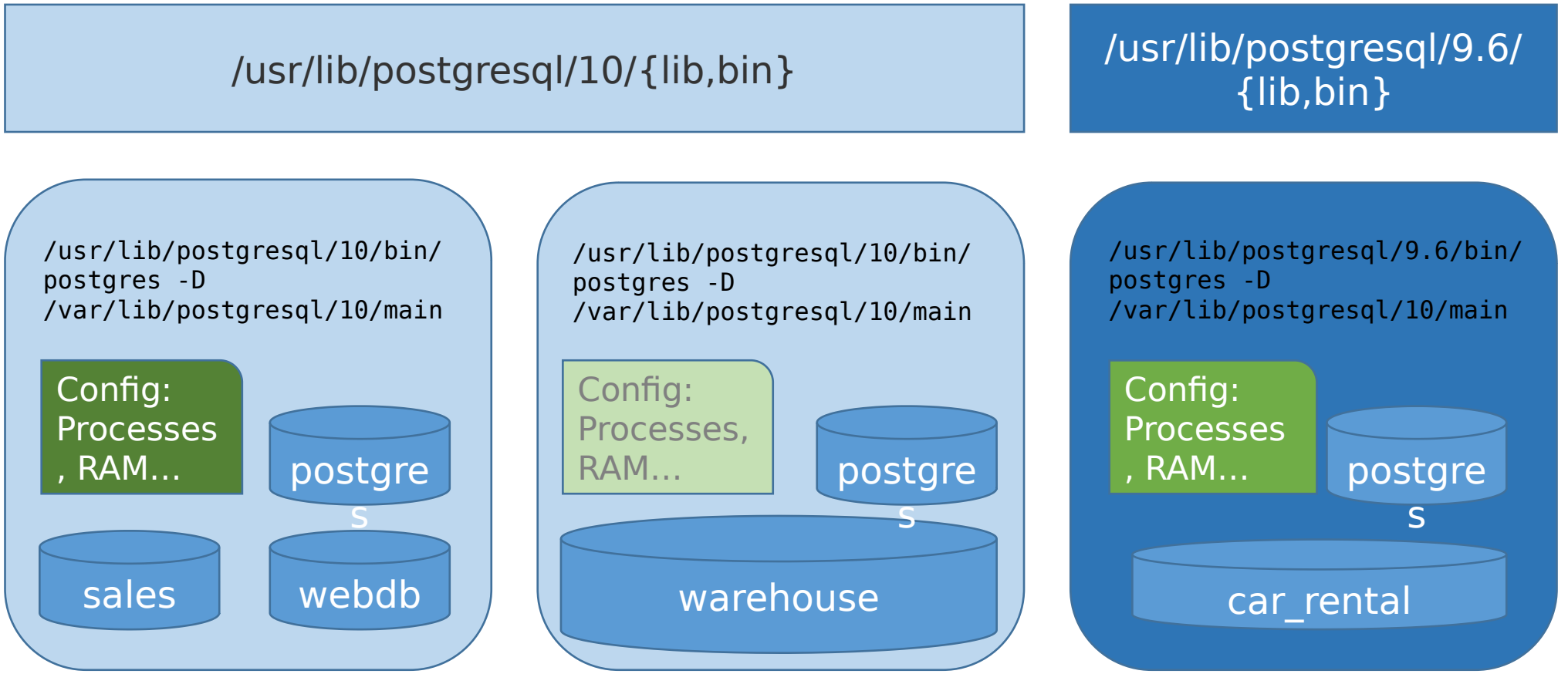
- PostgreSQL & Grafana
  - Extenze Timescale
- Prometheus & Grafana (častá implementace)
  - Postgres Exportes
  - Node Exporter
  - SQL exporter
  - ...
- InfluxDB & Grafana
  - Telegraf jako agent pro sběr dat
  - InfluxDB Continuous Queries
    - Historizace dat

Architektura

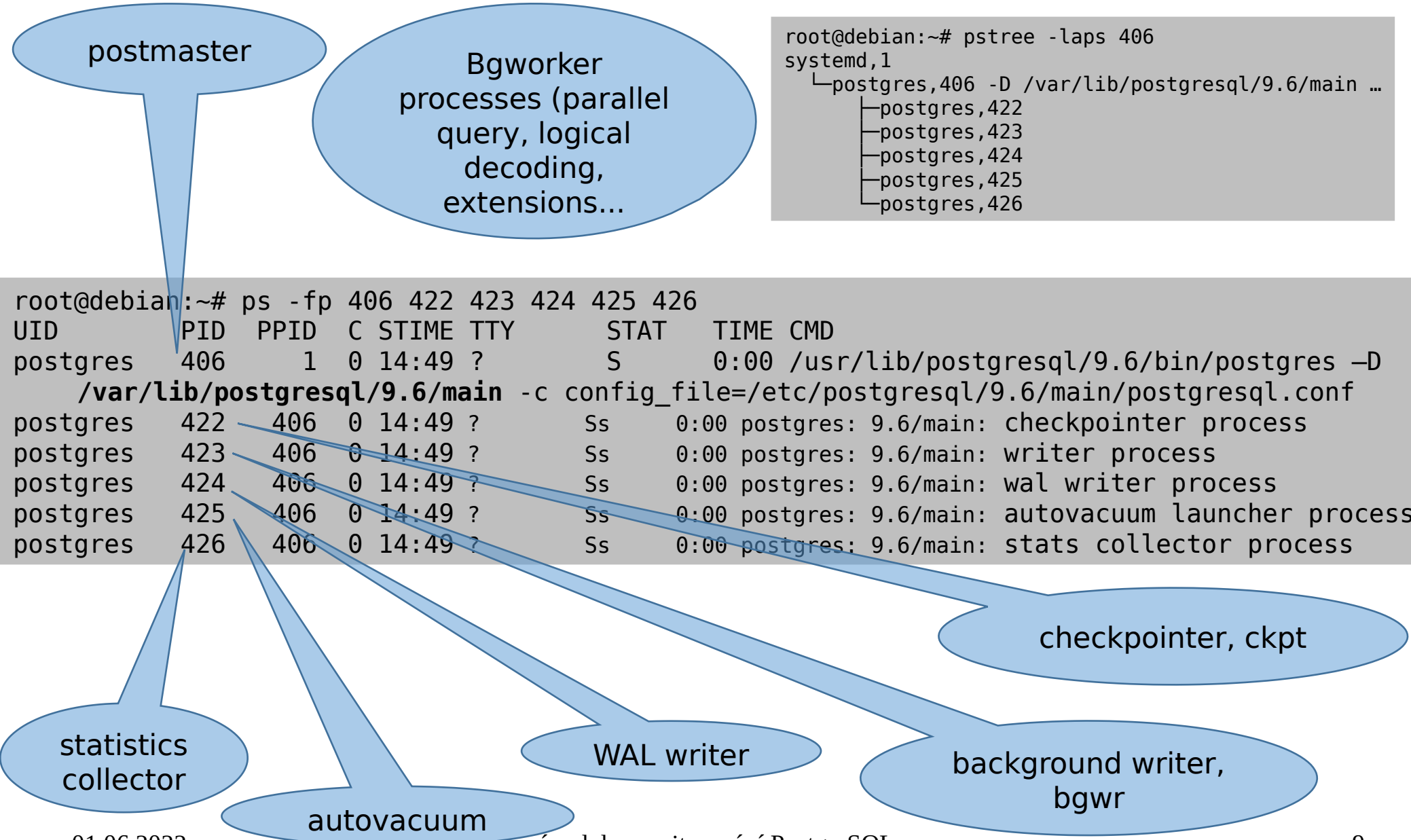


PostgreSQL v OS

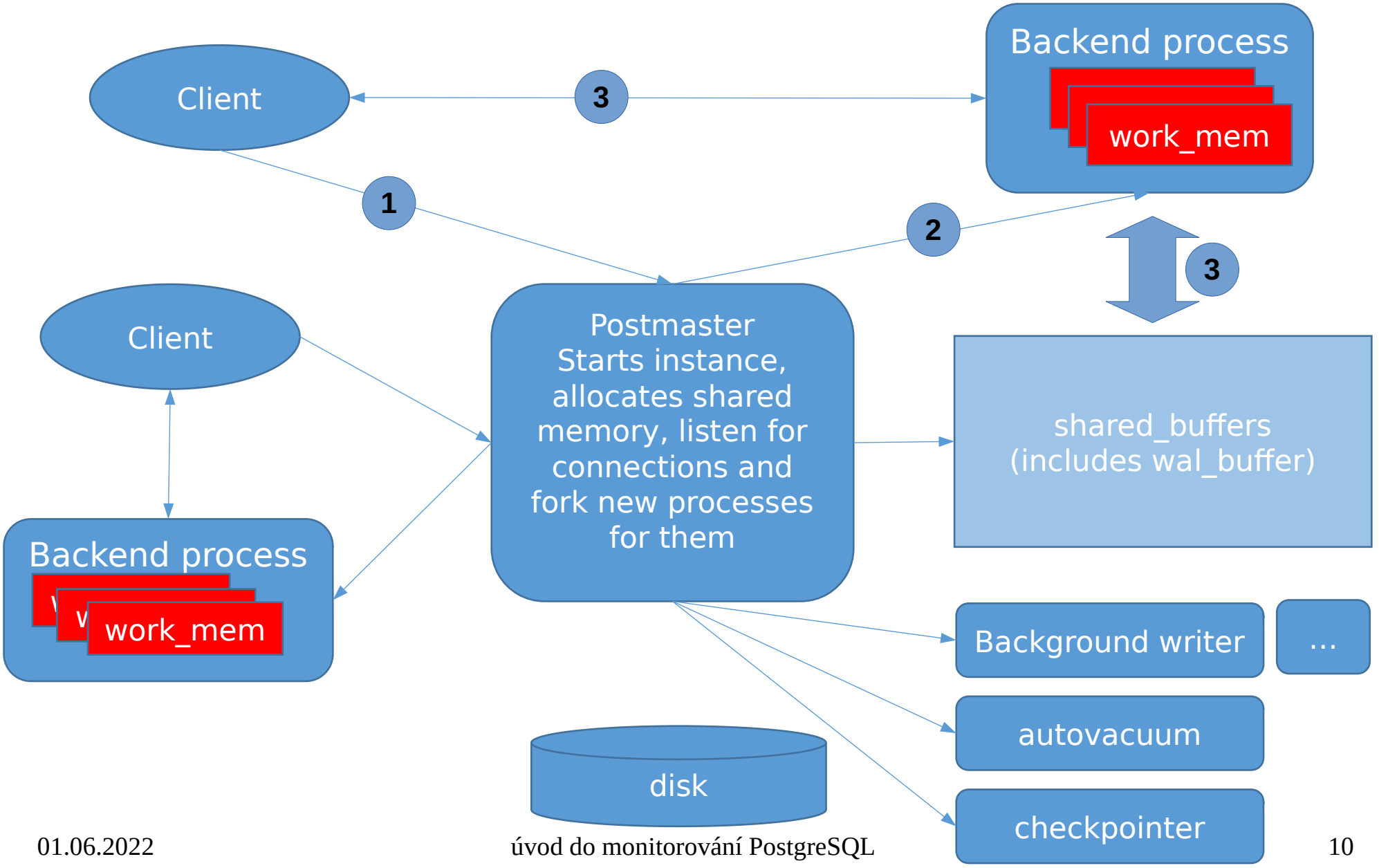
- RDBMS SW instalace → Postgres instance → databáze spravované instancemi.



Procesy instance

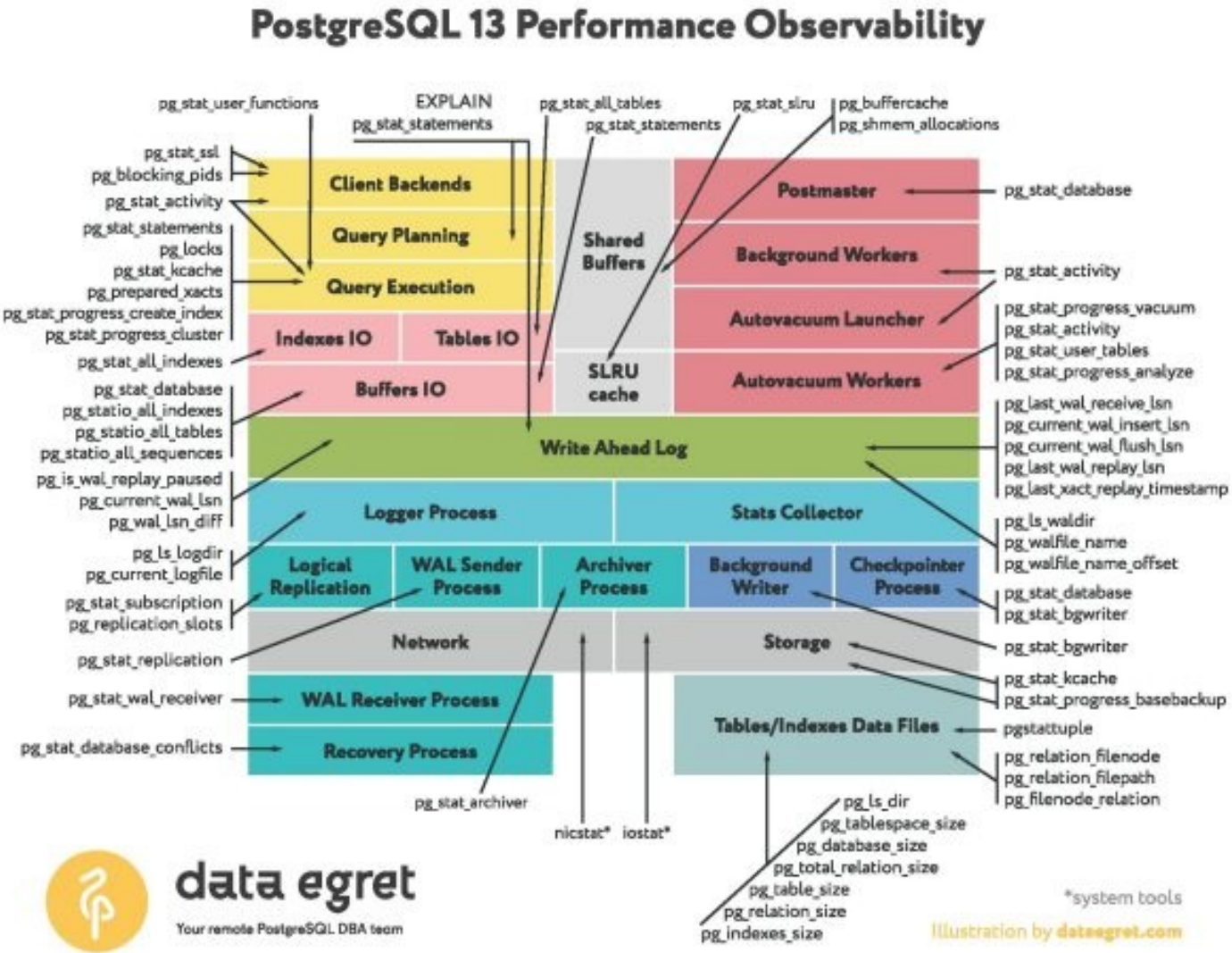


Klienti a komponenty serveru



Zdroje informací – DB instance

- Understand internal views and processes
  - PostgreSQL workings in one picture by AlexeyLesovsky, © DataEgret.com | All rights reserved
- Dokumentace
- Nástroje OS
- PostgreSQL log



Interní statistiky

- Monitoring Database Activity - dokumentace
  - Dynamické (okamžitá hodnota, progres operace)
  - Kumulativní
- Globální informace na úrovni instance
  - pg_stat_activity
  - pg_stat_database
  - pg_stat_progress_vacuum
  - ...
- Lokální informace v rámci připojené databáze
  - pg_stat_user_tables
  - ...

Konfigurace pro monitoring

- CPU - rychlost dotazu na aktuální čas
- pg_test_timing

```
postgres$ /usr/pgsql-9.4/bin/pg_test_timing
Testing timing overhead for 3 seconds.
Per loop time including overhead: 41.79 nsec
Histogram of timing durations:
< usec   % of total      count
     1     95.84044   68798574
     2      4.15628    2983562
     4      0.00242       1734
     8      0.00046        328
    16      0.00037        263
    32      0.00000          3
    64      0.00000          0
   128      0.00000          1
   256      0.00000          1
   512      0.00003         25
```

Konfigurace pro monitoring

```
# - Kernel Resource Usage -
shared_preload_libraries = 'pg_stat_statements'

# - When to Log -
log_min_duration_statement = 500 # [ms] upravit tak, aby log nebyl zahlcen
log_min_duration_sample = 5 # [ms] → min. Trvání dotazu pro sampling # PG14
log_statement_sample_rate = 0.05 # 0 – 1.0, default 1.0 → vše # PG14

# - What to Log -
log_checkpoints = on
log_connections = on
log_disconnections = on
log_line_prefix = '%t %p %c %l %r %d %u %Q %x %v %a %i [%e]:'
log_lock_waits = on
log_statement = 'ddl'
log_temp_files = 0 # 2048… based on details needed, logging impact

# - Query/Index Statistics Collector -
track_activities = on # information on the currently executing command
track_counts = on # statistics on database activity – potřebuje autovacuum
track_io_timing = on # timing of database I/O calls
track_functions = all
track_activity_query_size = 8192 # default 1024
# AUTOVACUUM PARAMETERS

log_autovacuum_min_duration = 0 # for log parsers, like pg_badger
```

Nastavení pro perf. test analýzu

- POWA - PostgreSQL Workload Analyzer
  - Lze použít i profilery
    - plpgsql_check
    - plProfiler
- Pro běžný provoz by byl log rychle narůstal
  - Intenzivní zápis do logu má dopad na performance
- pg_badger „perftest“ nastavení

```
# - What to Log -
log_duration = on
log_min_duration_statement = 0
log_statement = 'all' # pozor, po testu vypnout
log_temp_files = 0 # podle četnosti lze ponechat
```

pgbadger

- parsing logů, generuje HTML report
  - podporuje **inkrementální** zpracování
  - parallelizace
  - čtení logů přes ssh
  - zpracuje komprimované gzip logy (logrotate)
  - reporty za časové okno (v týdnech)

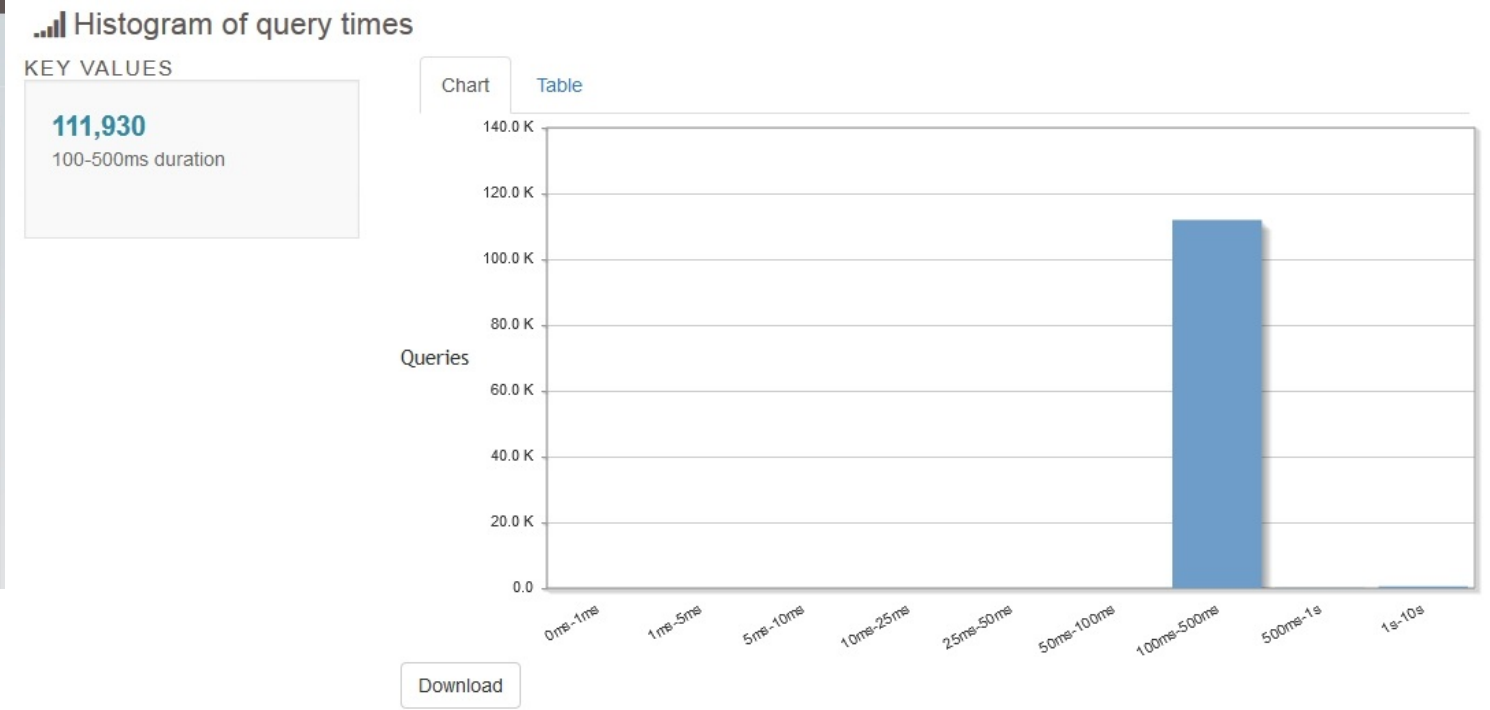
```
pgbadger -I -R 4 -j 2 -J 4 -f stderr –start-monday --ssh-option -q \
-r cosi.kdesi.net --prefix %t %p %c %l %r %d %u %x %v %a %i [%e]: \
/var/lib/pgsql/data/pg_log/postgresql.log*.gz \
-O /var/lib/pgreports/my_cluster_name
```

pgbadger

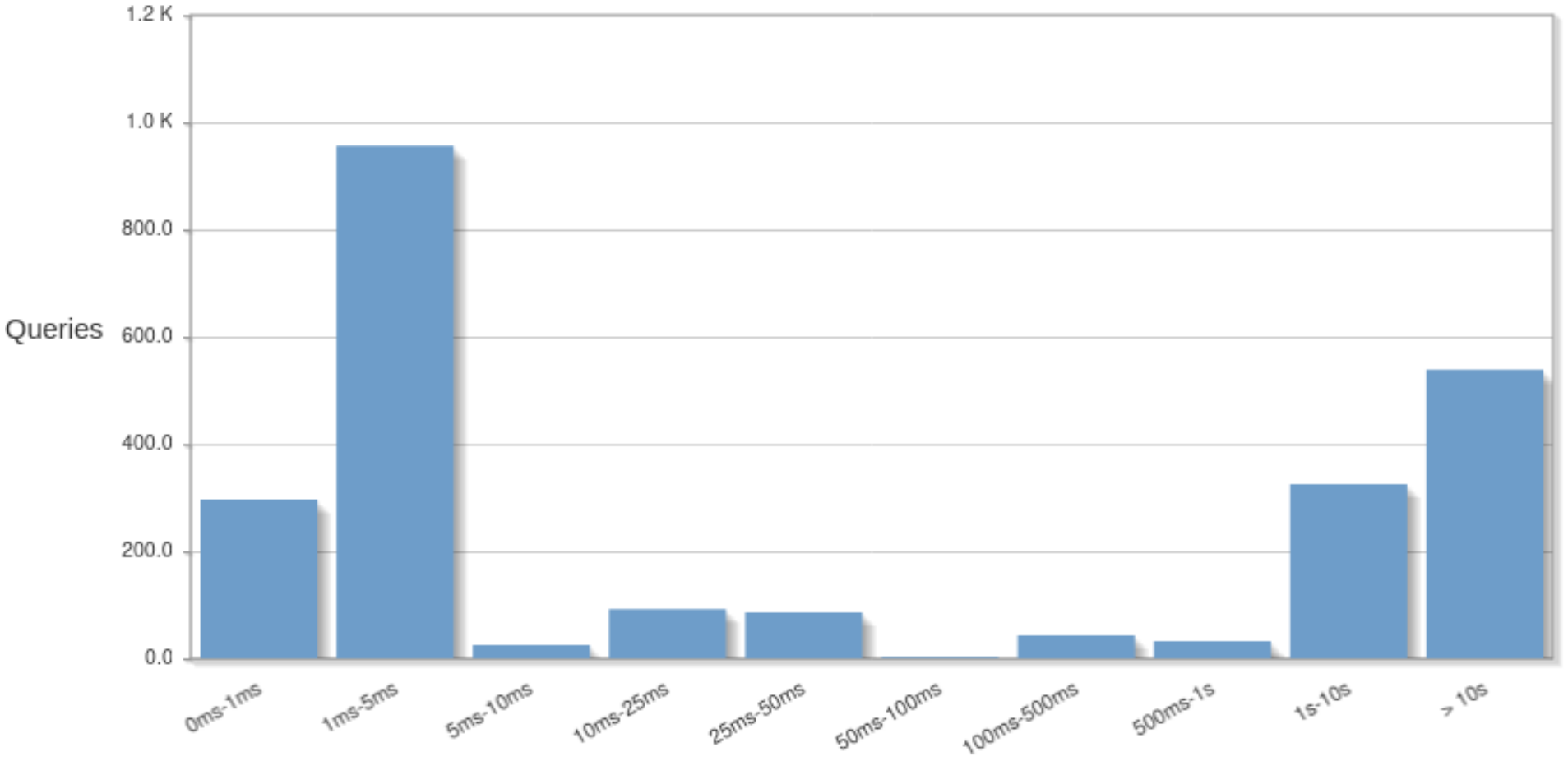
- Přehledný manuál
- denně generované reporty jsou přístupné vývojářům a aplikační podpoře přes web
- Kontrola po nasazení nové verze, či revize průběhu výkonnostních testů
- Reporty nepředávejte třetí straně, obsahují hodně informací, které mohou být důvěrné
  - Jména objektů, uživatelů
  - Vzorky hodnot vázaných proměnných...

pbadger – query times

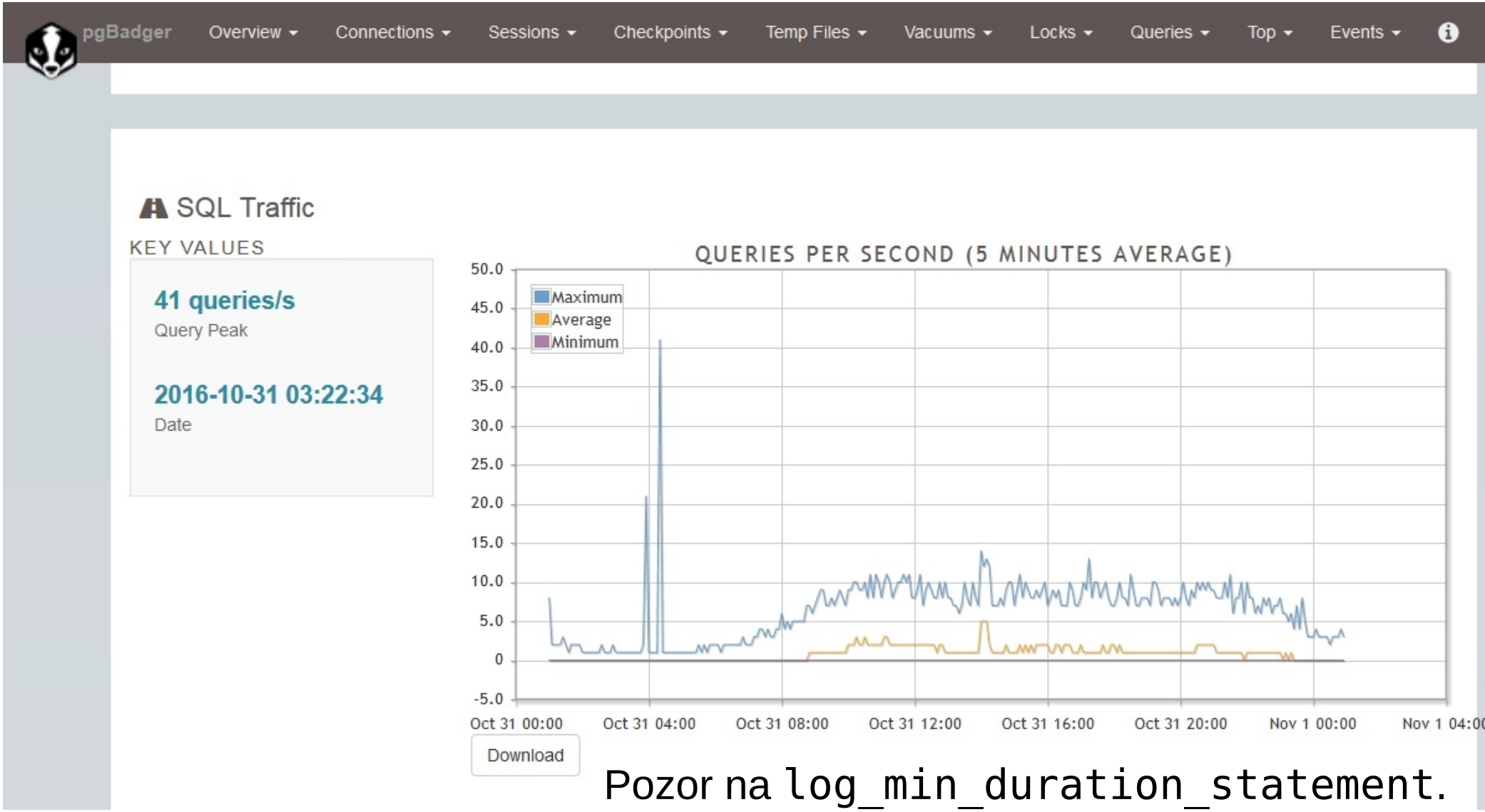
Pozor na `log_min_duration_statement` histogram pak může na první pohled vypadat zvláštně – pro aplikační podporu či vývojáře je to rychlá informace, zda se neobjevil podezřele pomalý dotaz.



PgB Query dur. PG 14 - sample rate



pbadger – sql traffic



Pozor na `log_min_duration_statement`.

pgbadger – pomalé dotazy

Slowest individual queries

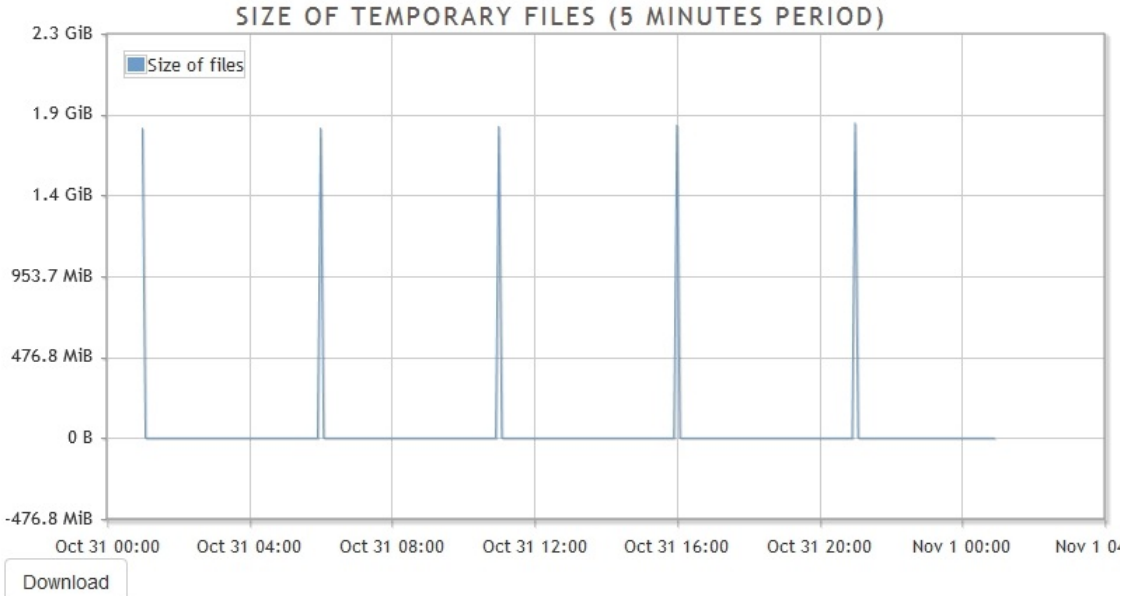
Rank	Duration	Query
1	1m56s	DELETE FROM a[illegible]t WHERE ctid IN (SELECT ctid FROM [illegible] WHERE created < '2016-08-01 05:01:41.366' LIMIT 10000); [Date: 2016-10-31 05:03:37 - Database: [illegible] - User: [illegible]_app - Remote: 10[illegible]0 - Application: [unknown] - Bind query: yes]
2	1m9s	DELETE FROM [illegible] WHERE ctid IN (SELECT ctid FROM [illegible] WHERE created < '2016-08-01 20:00:59.582' LIMIT 10000); [Date: 2016-10-31 20:02:08 - Database: [illegible] - User: [illegible]_app - Remote: 10[illegible] - Application: [unknown] - Bind query: yes]

Size of temporary files

KEY VALUES

601.54 MiB
Temp Files size Peak

2016-10-31 20:02:08
Date



pgbadger - deadlock

2	125 Details	`LOG: process ... still waiting for ShareLock on transaction ... after ... ms` Examples
3	56 Details	`LOG: process ... still waiting for ExclusiveLock on tuple (...) of relation ... of database ... after ... ms` Examples
4	27 Details	`ERROR: deadlock detected` Examples

ERROR: deadlock detected
Detail: Process 16851 waits for ShareLock on transaction 144602509; blocked by process 35447. Process 35447 waits for ShareLock on transaction 144676444; blocked by process 16851. Process 16851: DELETE FROM J[illegible] WHERE [illegible] = $1 AND [illegible] = $2 Process 35447: UPDATE [illegible] SET [illegible] = NULL, [illegible] = CURRENT_TIMESTAMP WHERE [illegible] IS NOT NULL AND CURRENT_TIMESTAMP - ($1 * '1 minute' :: INTERVAL) > next_process
Context: while deleting tuple (5,36) in relation "[illegible]"
Hint: See server log for query details.
Statement: DELETE FROM J[illegible] WHERE id = $1 AND [illegible] = $2

```
Date: 2016-11-16 03:35:04
Database: [illegible]
Application: [unknown]
User: [illegible]
Remote: 1[illegible]
```

Statistické pohledy

- aktuální stav (Dynamic Statistics Views)
  - pg_stat_activity
  - pg_stat_replication
  - pg_stat_progress_create_index
  - ...
- Kumulativní (Collected Statistics Views) – v rozsahu instance
  - Lze vynulovat funkcí pg_stat_clear_snapshot()
  - pg_stat_database
  - …
- Kumulativní – v rozsahu připojené databáze
  - pg_stat_all_tables
  - ...

Instance: pg_stat_activity

- Přehled aktuálně připojených sessions (backend procesy)
  - **datname** – kam je proces připojen
  - age(now(), query_start) – trvání dotazu
  - age(now(), xact_start) – trvání transakce
  - **state** – stav session (active, idle, *idle in transaction*…)
  - **waiting** (změna v 9.6+ – wait_event_type, wait_event)
  - **backend_type** (10+ – client backend, background worker, walwriter...)

Instance: pg_stat_activity

- Informace o transakci (xid)
  - backend_xmin
  - vacuum – log:

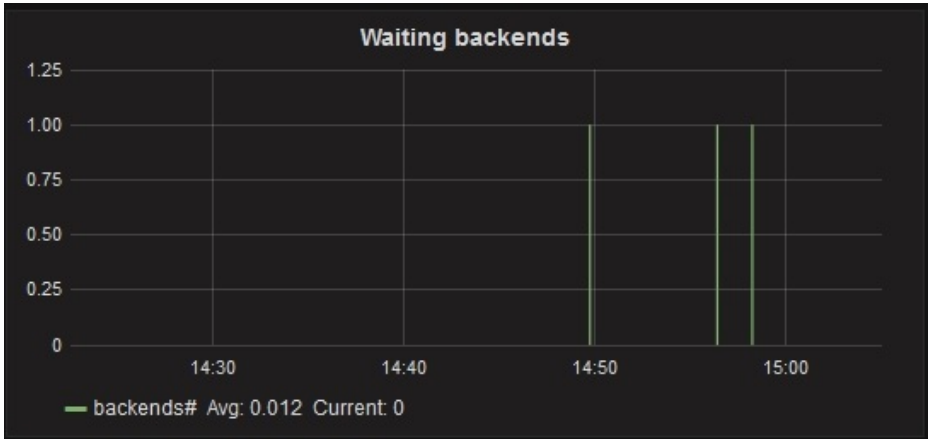
```
DETAIL:  0 dead row versions cannot be removed yet, oldest xmin: 2274177403
There were 946 unused item pointers.
Skipped 0 pages due to buffer pins, 2414342 frozen pages.
0 pages are entirely empty.
```

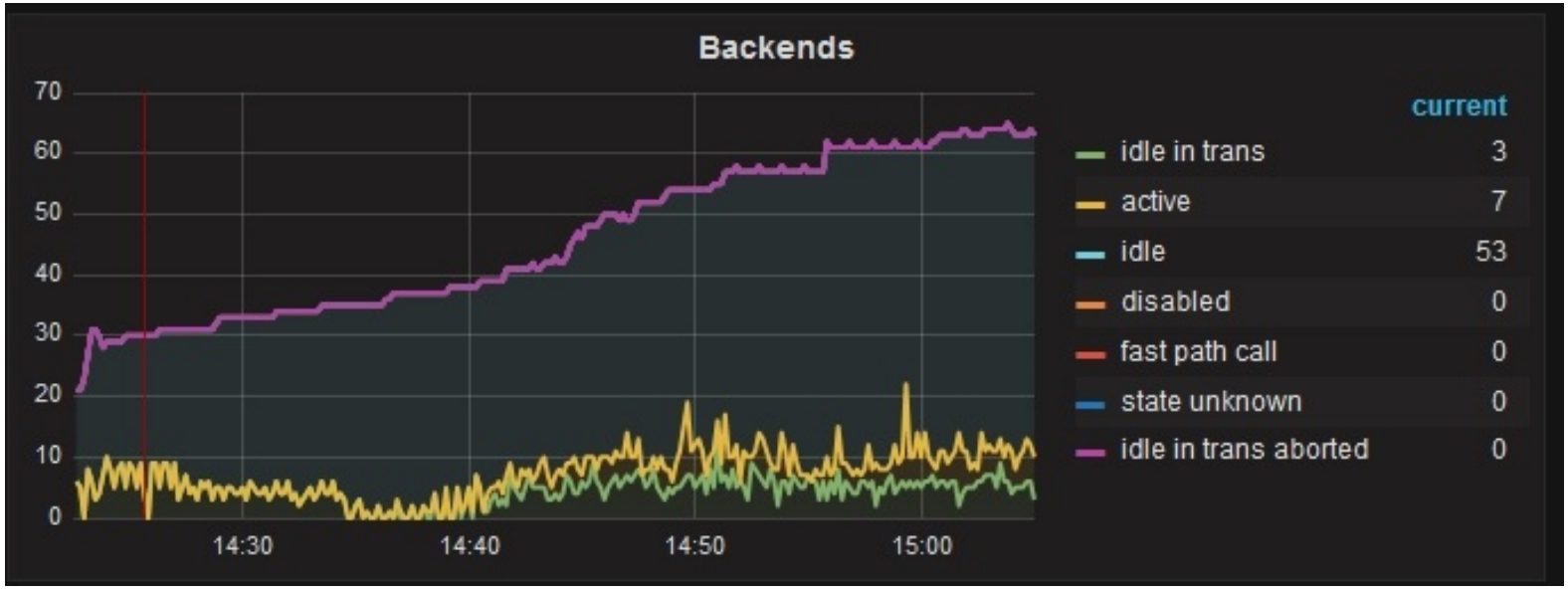
- Sloupce xact_start, backend_xmin
- Nebo dotaz na čas transakce
  - Vyžaduje nastavení `track_commit_timestamp`

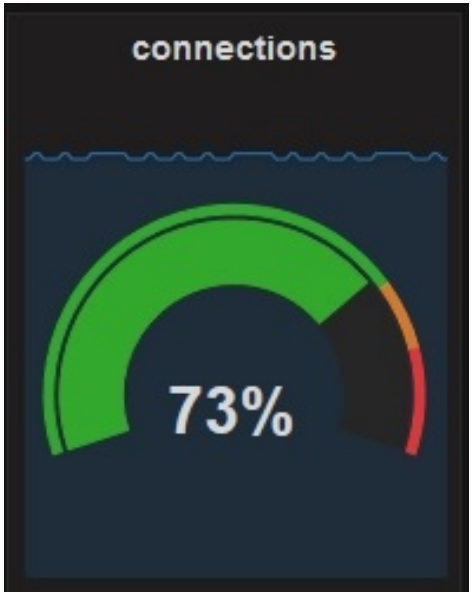
```
select pg_xact_commit_timestamp(backend_xmin), psa.* from pg_stat_activity psa
order by 1;

select pg_xact_commit_timestamp('2274177403'::xid);
```

pg_stat_acticity

- idle in transaction
- využití max_sessions
- čekající sessions







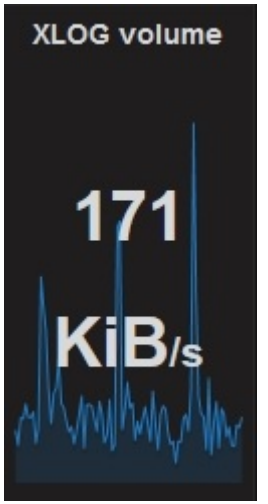
Instance: pg_stat_archiver

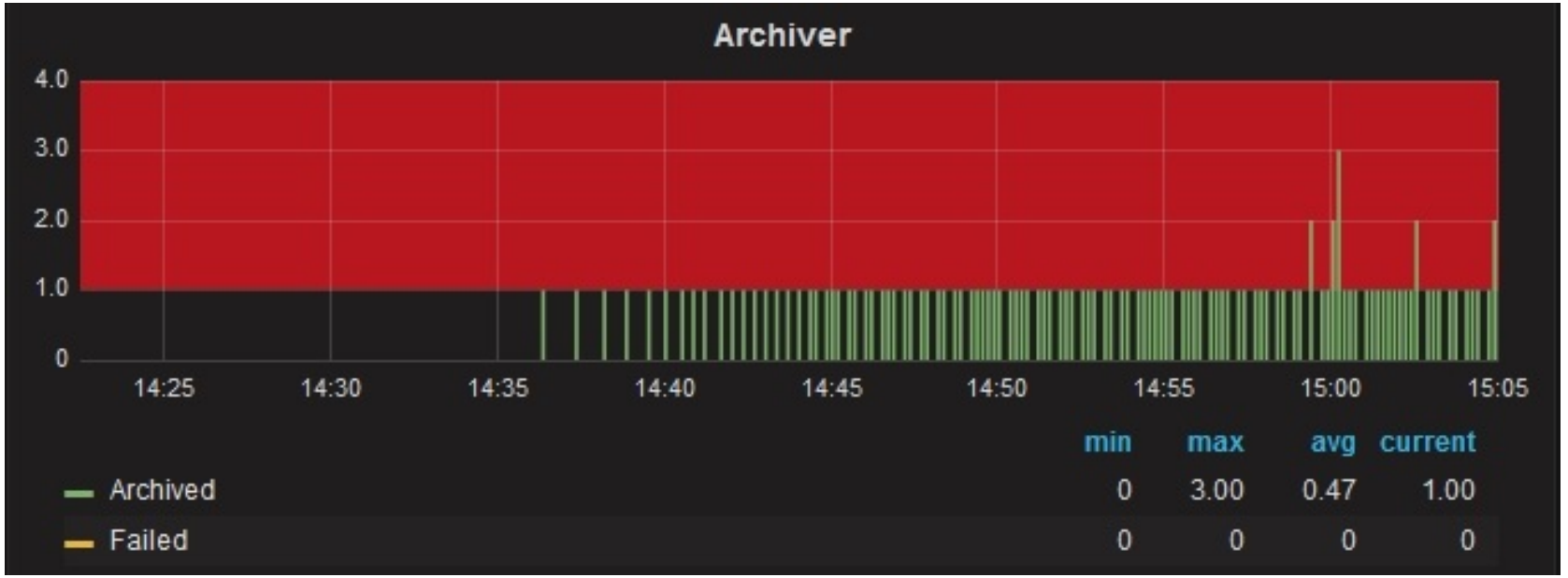
- **archived_count**
- last_archived_time
  - `age(now() - last_archived_time)` -- interval
  - `extract( epoch from age(now(), last_archived_time)) as arch_age_sec` -- integer
- **failed_count**
- last_failed_time
- Lze přidat objem transakčních logů

```
select pg_xlog_location_diff(pg_current_xlog_location(),
'0/00000000'::pg_lsn) as xlog_volume;
```

pg_stat_archiver

- četnost přepínání xlog segmentů
  - Vizuální kontrola zda je nastaven `archive_timeout`
- Četnost selhání `archive_command`



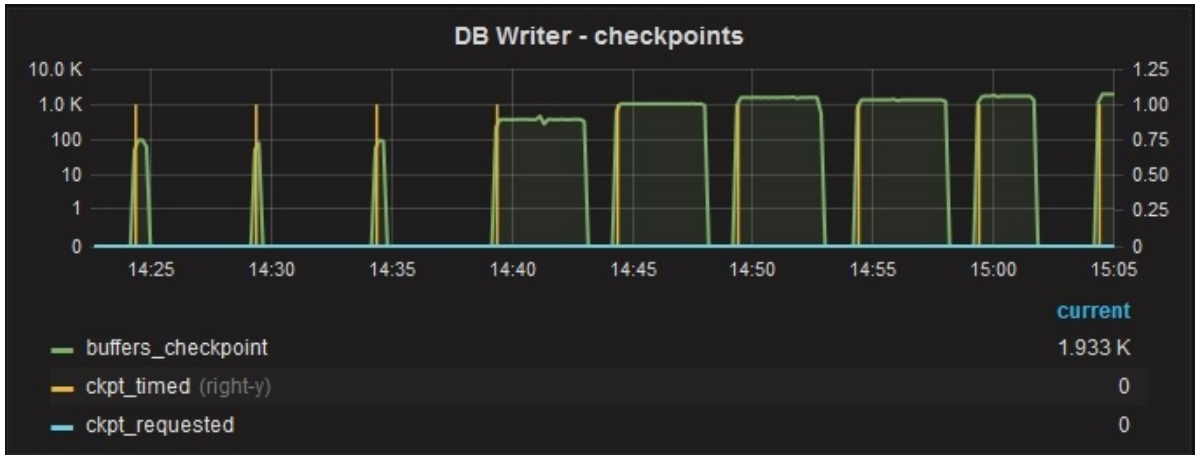


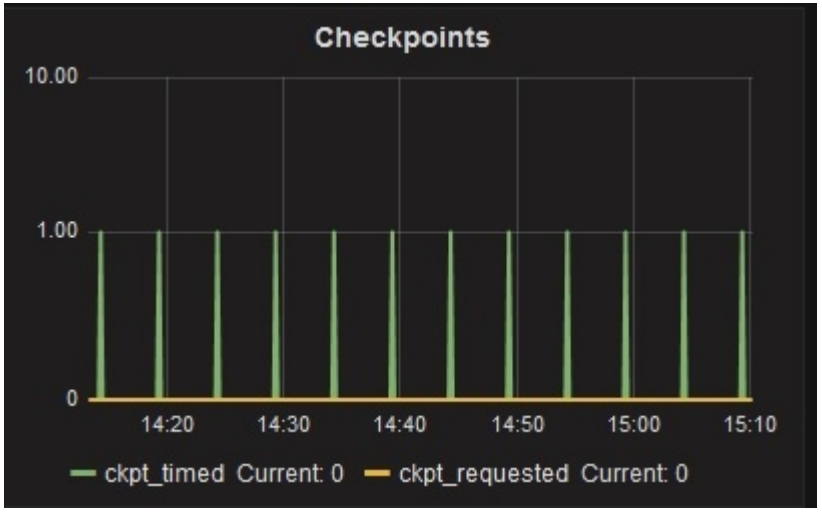
Instance: pg_stat_bgwriter

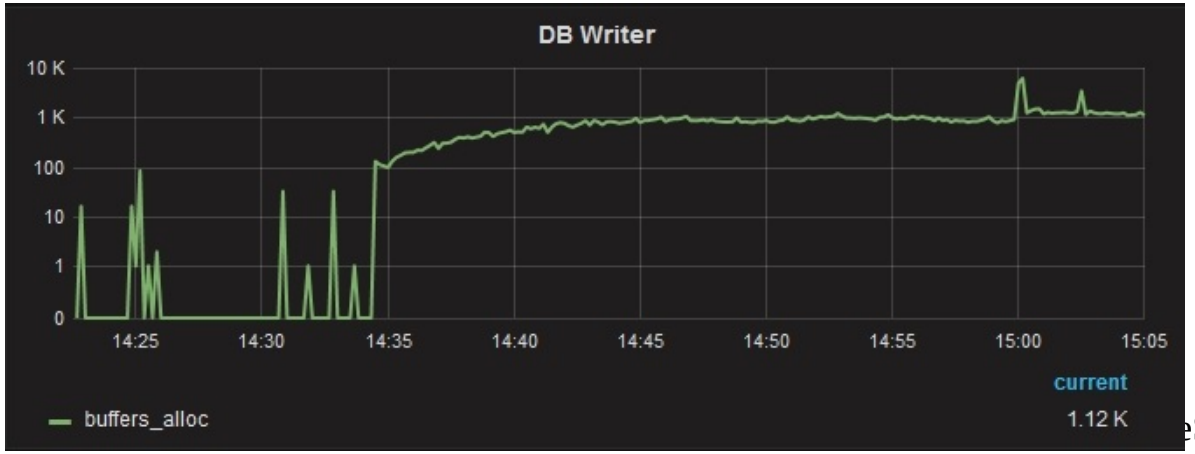
- **checkpoints_timed**
- **checkpoints_req**
- **buffers_checkpoint**
- buffers_clean – buffery zapsané bgwriter procesem (asynchronní, to chceme)
- buffers_backend – buffery zapsané backend procesy přímo (nebylo místo, zdržení)
- buffers_alloc – alokované buffery

pg_stat_bgwriter

- Vizuální kontrola checkpoint-ů a checkpoint_completion_target





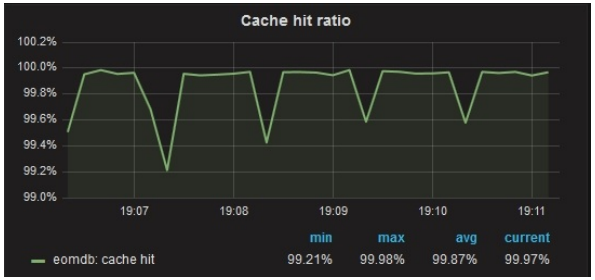


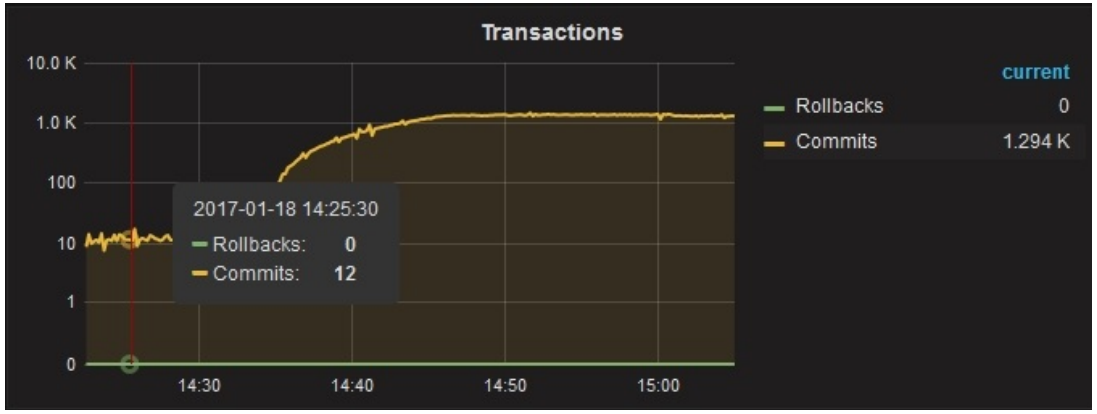
Instance: pg_stat_database

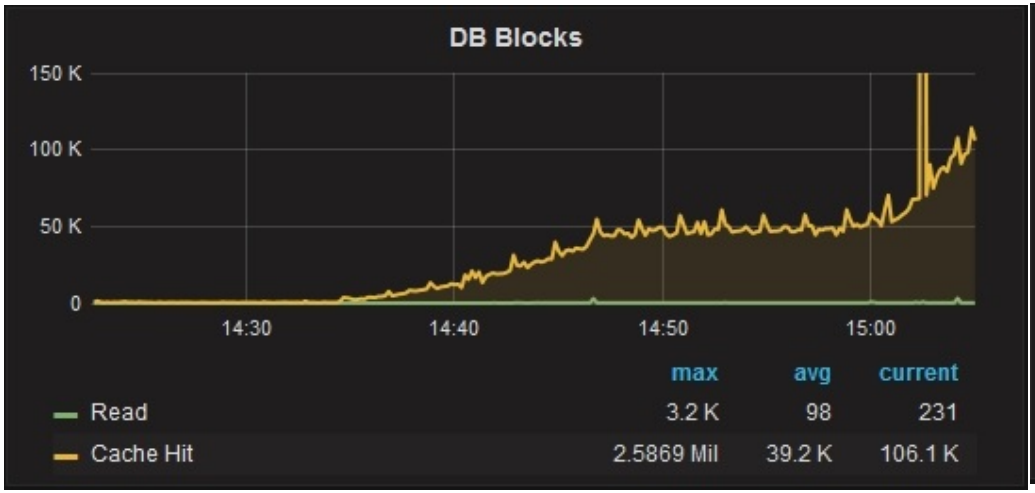
- jeden řádek pro každou DB, kromě numbackends jsou statistiky kumulativní
- datname
- **numbackends** – aktuální hodnota
- **xact_commit, xact_rollback** – kumulativní sledovat poměr commit/rollback
- **blks_read, blks_hit** – efektivita buffer cache
- temp_files, temp_bytes
- **deadlocks**
- tup_% statistiky pro řádky (return, fetch, ins, udp, del)
- blk_read_time, blk_write_time – čas strávený backendy na IO – efektivita cache na úrovni OS

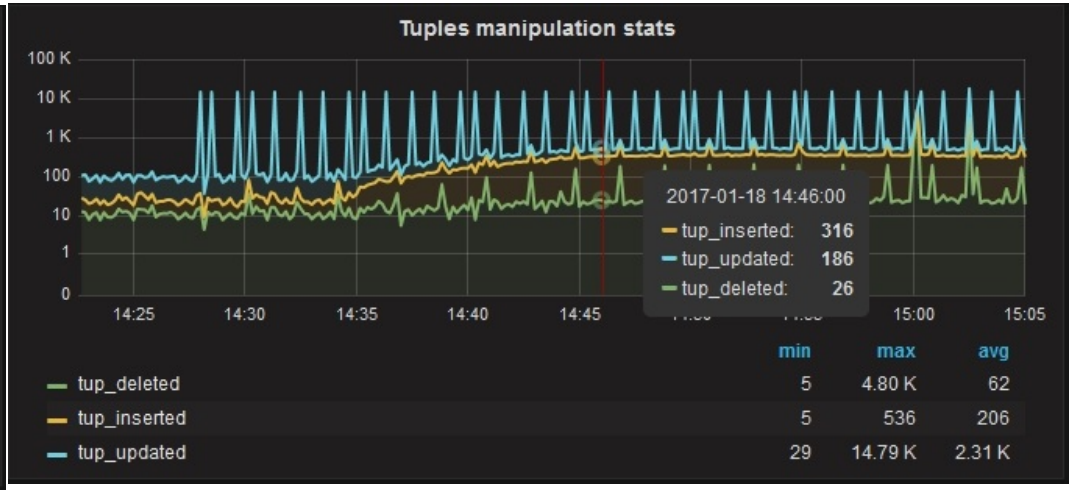
pg_stat_database

- úspěšnost buffer cache
- DML statistiky
- transakce









DB: pg_stat_all_tables

- pg_stat_sys_tables
- pg_stat_user_tables
- `seq_scan`, seq_tup_read/seq_scan
  - Kolikrát se tabulka četla a **kolik řádek bylo vráceno na jedno čtení** – nechybí index ?
- `n_tup_upd/n_tup_hot_upd`
  - Potenciální kandidát na změnu FILLFACTOR

- schemaname, relname…
- **seq_scan**
- seq_tup_read
- **idx_scan**
- idx_tup_fetch
- n_tup_upd
- n_tup_hot_upd
- autovacuum_count
- autoanalyze_count

pg_stat_user_tables

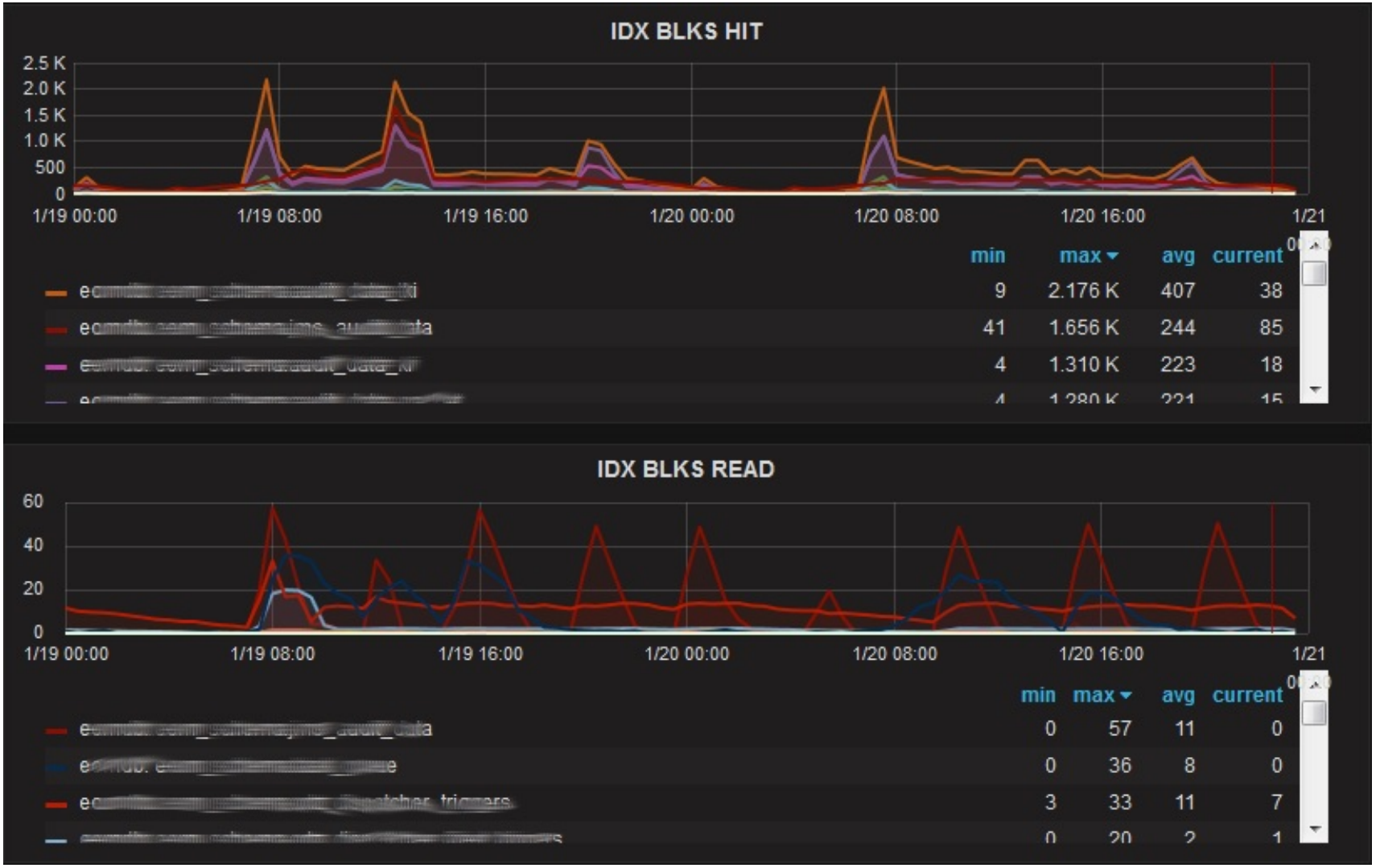
DB: pg_stat_all_indexes

- pg_stat_sys_indexes
- pg_stat_user_indexes
- `idx_scan`
  - používá se index ?
  - Pozor na časový úsek, za který data vyhodnocujeme (měsíční zpracování..)

- schemaname, relname, indexrelname…
- **idx_scan**
- idx_tup_read
- idx_tup_fetch

pg_buffercache extenze (nepouštět dotaz moc často, potřebuje přístup k shared buffers) umožňuje ověřit, zda bloky často používaných indexů zůstávají v shared buffers.

pg_stat_user_indexes



DB: pg_statio_all_tables

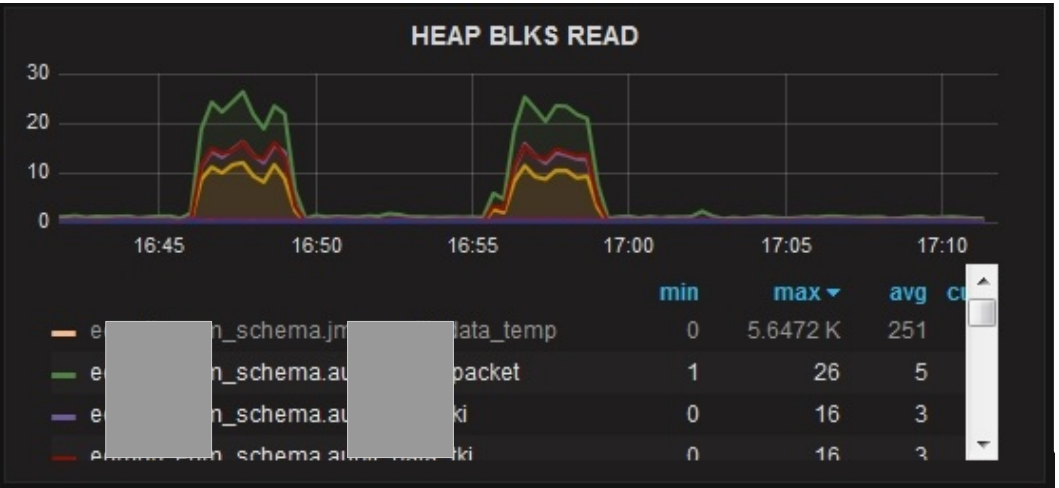
- pg_statio_sys_tables
- pg_statio_user_tables

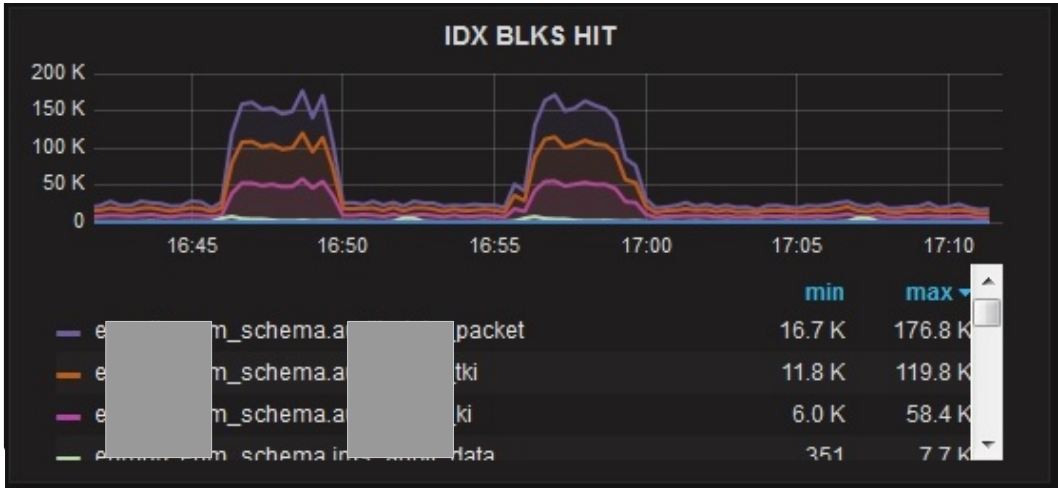
```
heap_blks_hit/
(heap_blks_hit+heap_blks_read)
```

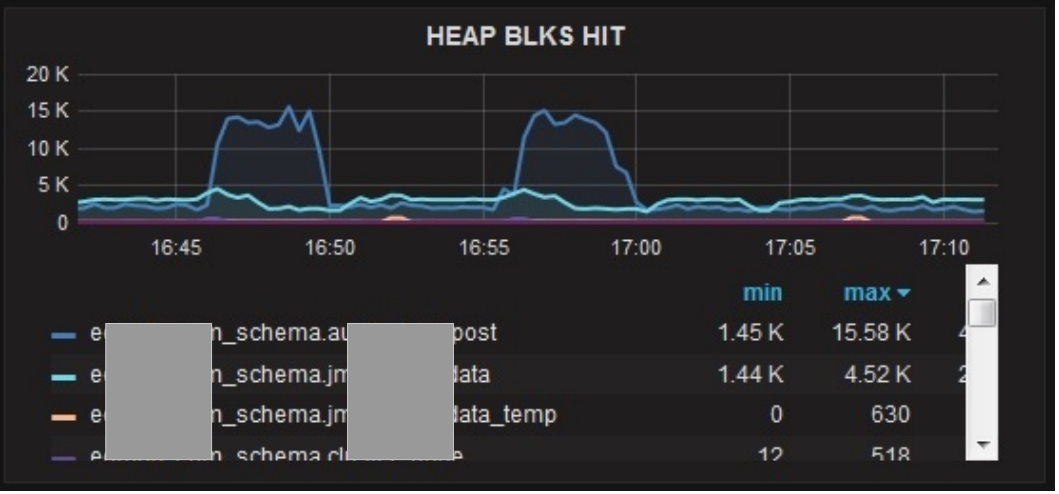
  - efektivita buffer cache
  - fyzické čtení nemusí být problém, pokud dobře funguje cache OS, viz pg_stat_database.blk_read_time

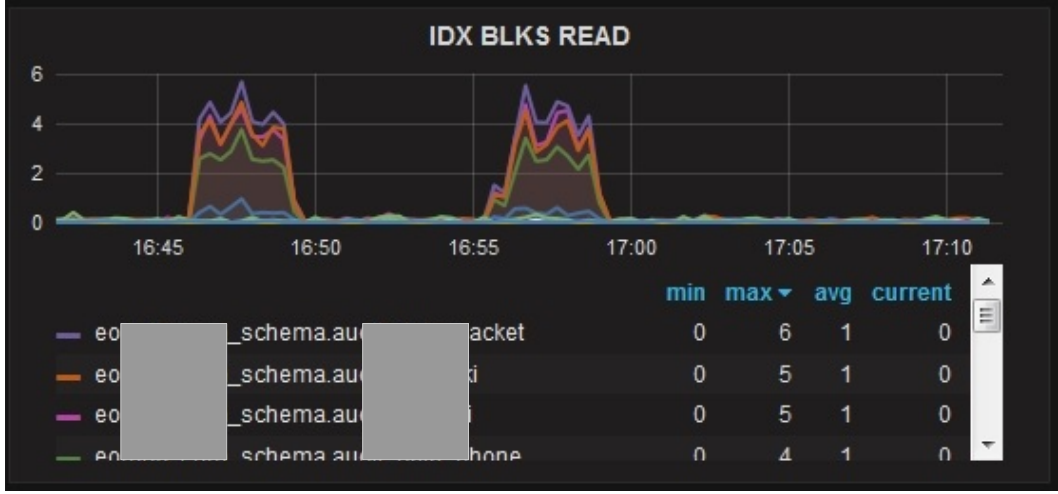
- schemaname, relname
- heap_blks_read
- heap_blks_hit
- **idx_blks_read**
- **idx_blks_hit**
- Toast_..., tidx_...

pg_statio_user_tables









DB: pg_statio_all_indexes

- pg_statio_sys_indexes
- pg_statio_user_indexes

```
idx_blks_hit/
(idx_blks_hit+idx_blks_read)
```

  - efektivita buffer cache
  - Informace na úrovni konkrétních indexů

- schemaname, relname, indexrelname
- **idx_blks_read**
- **idx_blks_hit**

DB: pg_statio_all_sequences

- pg_statio_sys_sequences
- pg_statio_user_sequences

- Nezaznamenali jsme žádný problém či důvod k systematickému sledování (ale každé prostředí je jiné)

- relname, schemaname, blks_read, blks_hit

DB: pg_stat_user_functions

- Zjištění prodlužujících se časů na jednotlivé volání
- ne / lineární vztah k objemu dat

- schemaname, funcname
- **calls**
- **total_time**
- **self_time**

instance: vacuum progress

- pg_stat_progress_vacuum
- od verze 9.6
- online progress reporting
- neukládáme do monitorovací DB
- vacuum FULL neposkytuje informace do „vacuum“ pohledu (jsou v pg_stat_progress_cluster)
- ANALYZE, CREATE INDEX, CLUSTER, COPY, base backup

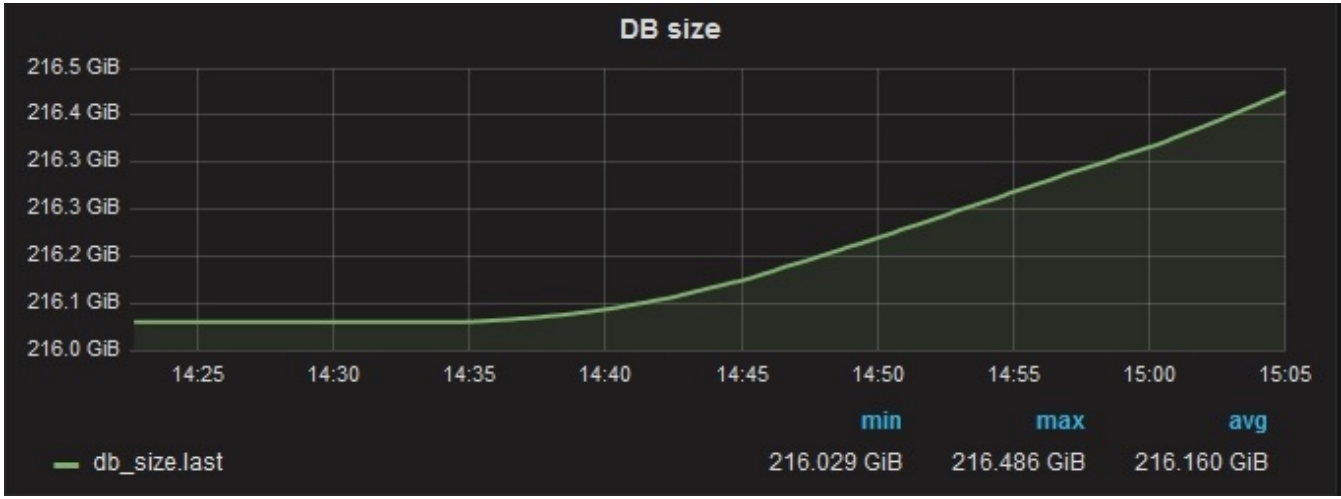
Vlastní dotazy

- velikost databází
- velikost jednotlivých relací (tabulky, indexy, marterializované pohledy, TOAST tabulky)
- blokující session
- objem WAL záznamů – přidali jsme k archiveru
- ne / povolené autovacuum nad tabulkami
- využití max_sessions (viz fce `current_setting()`)
- autovacuum threshold podle počtu řádek tabulky
- ...

instance: DB size

```
SELECT d.datname,
   CASE
      WHEN pg_catalog.has_database_privilege(d.datname,
'CONNECT') THEN
         pg_catalog.pg_database_size(d.datname)
      ELSE
         -1
   END as size
FROM pg_catalog.pg_database d;
```

```
  datname   |   size
------------+---------
 template0  |  6513156
 postgres   |  6631452
 hibernate  | 25325596
 jackrabbit | 22327324
 quartz     |  7209476
 template1  |  6521348
(6 rows)
```



DB: relation_size – 9.1

```
SELECT current_database() as datname, a.schemaname, a.relation_name,
a.relation_kind,
a.relation_persistence, a.row_estimate, a.total_bytes, a.index_bytes,
total_bytes-index_bytes AS relation_bytes
 FROM
( SELECT nspname AS schemaname, relname AS relation_name ,
    (CASE
        WHEN c.relkind = 'r' THEN 'table'
        when c.relkind = 'i' then 'index'
        when c.relkind = 'm' then 'materialized view'
        when c.relkind = 't' then 'TOAST table'
        ELSE 'other'
    END) as relation_kind ,
    (case
        when c.relpersistence = 'p' then 'permanent'
        when c.relpersistence = 'u' then 'unlogged'
        when c.relpersistence = 't' then 'temporary'
    END) as relation_persistence,
    c.reltuples AS row_estimate,
    pg_total_relation_size(c.oid) AS total_bytes,
    pg_indexes_size(c.oid) AS index_bytes
    FROM pg_class c LEFT JOIN
        pg_namespace n ON n.oid = c.relnamespace
    WHERE relkind in ('r', 'i', 'm', 't' )
) a;
```

DB: relation_size – 9.2+

```
SELECT current_database() as datname, a.schemaname, a.relation_name,
a.relation_kind,
a.relation_persistence, a.row_estimate, a.total_bytes, a.index_bytes,
a.toast_bytes, total_bytes-index_bytes-COALESCE(toast_bytes,0) AS
relation_bytes FROM
( SELECT nspname AS schemaname, relname AS relation_name ,
    (CASE
        WHEN c.relkind = 'r' THEN 'table'
        when c.relkind = 'i' then 'index'
        when c.relkind = 'm' then 'materialized view'
        when c.relkind = 't' then 'TOAST table'
        ELSE 'other'
    END) as relation_kind ,
    (case
        when c.relpersistence = 'p' then 'permanent'
        when c.relpersistence = 'u' then 'unlogged'
        when c.relpersistence = 't' then 'temporary'
    END) as relation_persistence,
    c.reltuples AS row_estimate,
    pg_total_relation_size(c.oid) AS total_bytes,
    pg_indexes_size(c.oid) AS index_bytes ,
    pg_total_relation_size(reltoastrelid) AS toast_bytes
    FROM pg_class c LEFT JOIN
        pg_namespace n ON n.oid = c.relnamespace
    WHERE relkind in ('r', 'i', 'm', 't')
) a;
```

instance: blocking sessions – 9.2 – 9.6

Lock Monitoring - vhodné příklady, přidat trvání blokovaného dotazu – threshold, grafy

– zdroj postgresql wiki…

```
SELECT a.datname AS db,
    kl.pid AS blocking_pid,
    ka.usename AS blocking_user,
    ka.query AS blocking_query,
    bl.pid AS blocked_pid,
    a.usename AS blocked_user,
    a.query AS blocked_query,
    extract( epoch from age(now(), a.query_start)) as age_sec,
    to_char(age(now(), a.query_start), 'HH24h:MIm:SSs'::text) AS age
   FROM pg_locks bl
     JOIN pg_stat_activity a ON bl.pid = a.pid
     JOIN pg_locks kl ON bl.locktype = kl.locktype AND NOT bl.database IS
     DISTINCT FROM kl.database AND NOT bl.relation IS DISTINCT FROM kl.relation
     AND NOT bl.page IS DISTINCT FROM kl.page AND NOT bl.tuple IS DISTINCT FROM
     kl.tuple AND NOT bl.virtualxid IS DISTINCT FROM kl.virtualxid AND NOT
     bl.transactionid IS DISTINCT FROM kl.transactionid AND NOT bl.classid IS
     DISTINCT FROM kl.classid AND NOT bl.objid IS DISTINCT FROM kl.objid AND
      NOT bl.objsubid IS DISTINCT FROM kl.objsubid AND bl.pid <> kl.pid
     JOIN pg_stat_activity ka ON kl.pid = ka.pid
  WHERE kl.granted AND NOT bl.granted
  ORDER BY a.query_start;
```

DB: autovacuum=on

- Ve skutečnosti je zajímavé, zda některá tabulka nemá **off**

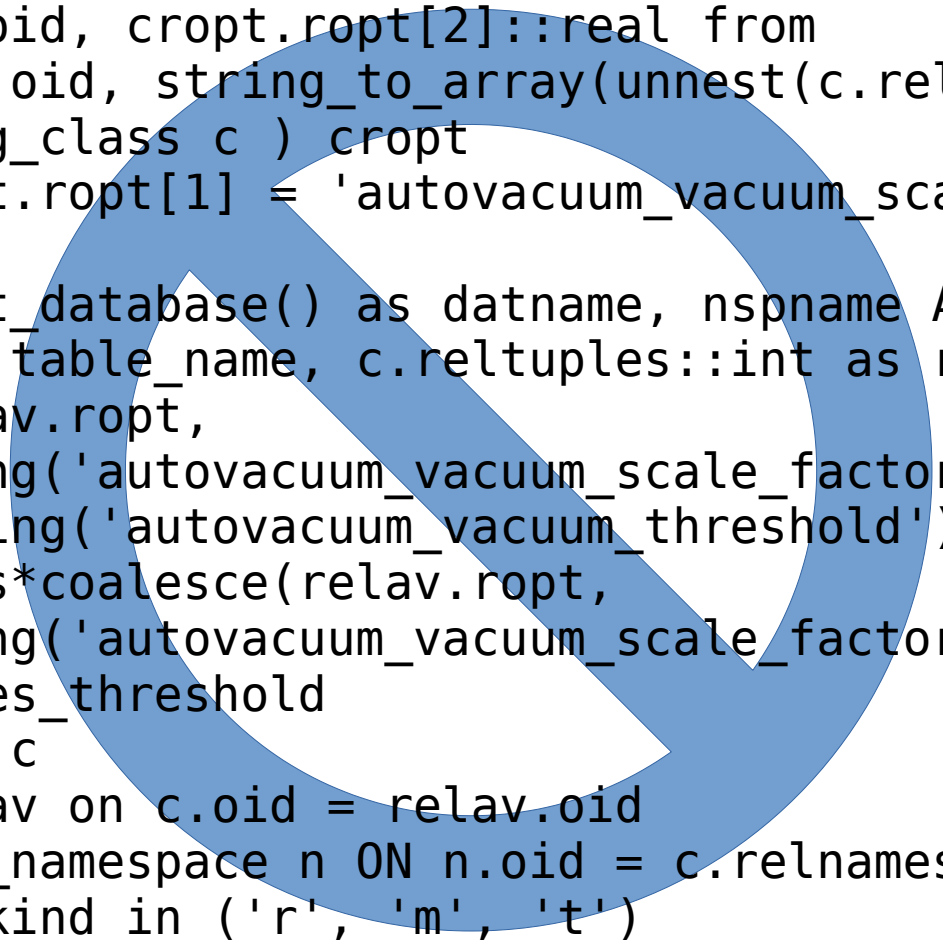
```
with relav as (
    select cropt.oid, cropt.ropt[2]::boolean from
        ( SELECT c.oid, string_to_array(unnest(c.reloptions), '=') as
ropt
          FROM pg_class c
        ) cropt
    where cropt.ropt[1] = 'autovacuum_enabled'
)
select current_database() as datname, nspname AS schemaname,
  c.relname as table_name,
  coalesce(relav.ropt, current_setting('autovacuum')::boolean)
    as autovacuum_enabled
from pg_class c
  left join relav on c.oid = relav.oid
  LEFT JOIN  pg_namespace n ON n.oid = c.relnamespace
WHERE c.relkind IN ('r', 'm', 't');

 datname  | schemaname |  table_name  | autovacuum_enabled
----------+------------+--------------+--------------------
 postgres | pg_catalog | pg_statistic | t
 postgres | pg_catalog | pg_type      | t
 postgres | pg_catalog | pg_authid    | t
 postgres | pg_catalog | pg_proc      | t
 postgres | pg_catalog | pg_class     | t
```

Vypnuté autovacuum

```
--  jen jinak napsaný dotaz, nebojte se vlastní tvorby

Select rvac.datname , rvac.relowner, rvac.schemaname, rvac.relname, rvac.relnspnametoast
    , rvac.relnametoast, rvac.autovacuum_enabled::text
from (
      select current_database()::text as datname
        , c.relnamespace::regnamespace::text AS schemaname
        , pg_get_userbyid(c.relowner) as relowner
        , c.relname::text as relname, c.relkind as relation_kind
        , c.reltoastrelid::regclass::text as toastrelname
        , tr.relnspnametoast, tr.relnametoast
        , coalesce(
             max(po.option_value) filter (where po.option_name = 'autovacuum_enabled')
           , current_setting('autovacuum')
        )::boolean as autovacuum_enabled
      from pg_class c
      LEFT JOIN pg_options_to_table(c.reloptions) po ON true -- lateral is implicit
      LEFT JOIN LATERAL (
        select relnamespace::regnamespace as relnspnametoast, relname as relnametoast
        from pg_class tc where tc.reltoastrelid = c.oid
      ) tr ON true
      where
        c.relkind = ANY ('{r,m,t}'::char[])
      group by
          datname, c.relnamespace, c.relowner, c.relname, c.relkind
        , c.reltoastrelid, relnspnametoast, tr.relnametoast
) rvac WHERE not autovacuum_enabled
order by
      rvac.datname
    , rvac.schemaname
    , rvac.relname;
```

DB: autovacuum threshold

- Pro velké tabulky může být výchozí autovacuum_vacuum_scale_factor příliš vysoký

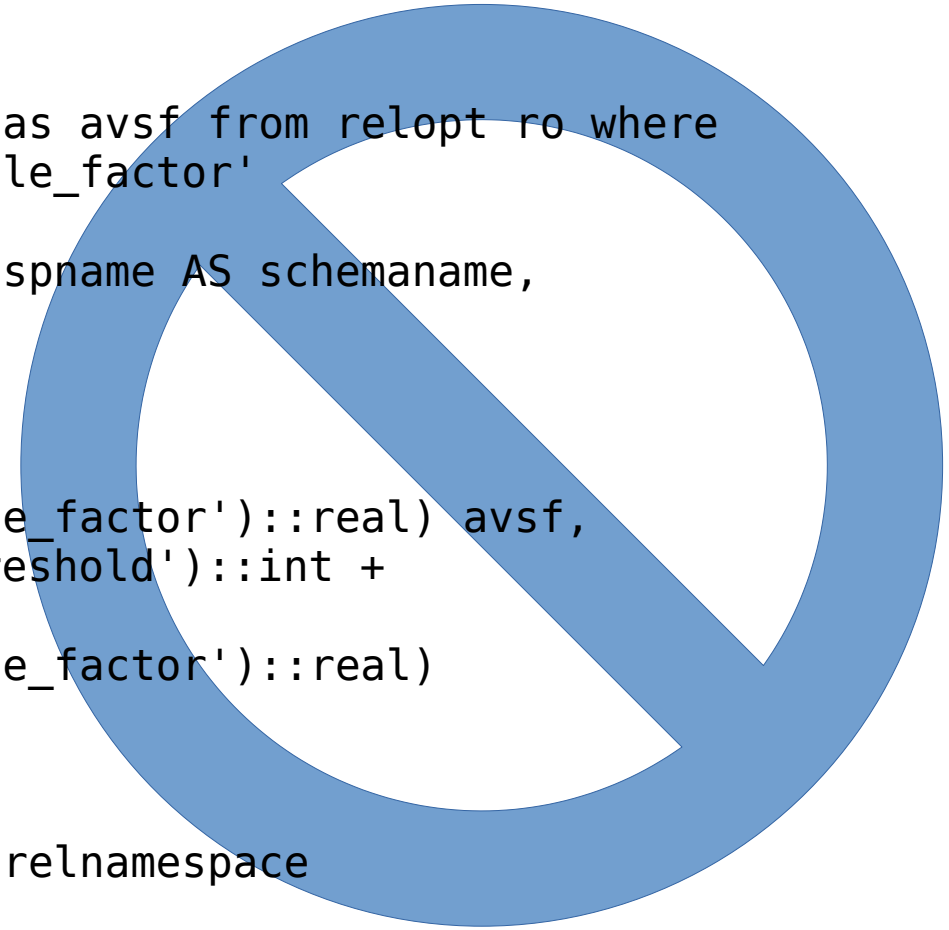


```
with relav as (
select cropt.oid, cropt.ropt[2]::real from
   ( SELECT c.oid, string_to_array(unnest(c.reloptions), '=') as
ropt FROM   pg_class c ) cropt
   where cropt.ropt[1] = 'autovacuum_vacuum_scale_factor'
)
select current_database() as datname, nspname AS schemaname,
 c.relname as table_name, c.reltuples::int as row_estimate,
 coalesce(relav.ropt,
current_setting('autovacuum_vacuum_scale_factor')::real) avsf,
 current_setting('autovacuum_vacuum_threshold')::int +
   c.reltuples*coalesce(relav.ropt,
current_setting('autovacuum_vacuum_scale_factor')::real)
   as av_tuples_threshold
from pg_class c
left join relav on c.oid = relav.oid
LEFT JOIN  pg_namespace n ON n.oid = c.relnamespace
where --c.relkind in ('r', 'm', 't')
   relkind = ANY ('{r,m,t}'::char[])
order by av_tuples_threshold desc;
```

DB: autovacuum threshold

- Alternativa téhož dotazu

```
WITH relopt AS (
  select OID, (pg_options_to_table(reloptions)).option_name,
  (pg_options_to_table(reloptions)).option_value
  from pg_class c
),
relav as (
  select ro.oid, ro.option_value::real as avsf from relopt ro where
ro.option_name = 'autovacuum_vacuum_scale_factor'
)
select current_database() as datname, nspname AS schemaname,
 c.relname as table_name,
 c.reltuples::int as row_estimate,
 relav.avsf,
 coalesce(relav.avsf,
current_setting('autovacuum_vacuum_scale_factor')::real) avsf,
 current_setting('autovacuum_vacuum_threshold')::int +
   c.reltuples*coalesce(relav.avsf,
current_setting('autovacuum_vacuum_scale_factor')::real)
   as av_tuples_threshold
from pg_class c
left join relav on c.oid = relav.oid
LEFT JOIN  pg_namespace n ON n.oid = c.relnamespace
where --c.relkind in ('r', 'm', 't')
c.relkind = ANY ('{r,m,t}'::char[])
order by av_tuples_threshold desc limit 5;
```



Autovacuum – už zase...

- Testujte, předchozí dotazy neuměly zpracovat autovacuum_vacuum_scale_factor = 0

```
select
      rvac.datname, rvac.schemaname, rvac.relname, rvac.relnspnametoast, rvac.relnametoast, rvac.row_estimate
    , rvac.autovacuum_enabled::boolean::text, rvac.autovacuum_analyze_scale_factor, rvac.autovacuum_vacuum_scale_factor
    , rvac.autovacuum_vacuum_threshold, rvac.autovacuum_analyze_threshold
    , (rvac.autovacuum_analyze_threshold + rvac.row_estimate * autovacuum_analyze_scale_factor)::int8 as
eff_autovacuum_analyze_threshold
    , (rvac.autovacuum_vacuum_threshold + rvac.row_estimate * autovacuum_vacuum_scale_factor)::int8 as eff_autovacuum_vacuum_threshold
from (
      select current_database()::text as datname, c.relnamespace::regnamespace::text AS schemaname,
      c.relname::text as relname,
      c.relkind as relation_kind,
      tr.relnspnametoast,
      tr.relnametoast,
      max(c.reltuples)::int8 as row_estimate,
      coalesce(
          max(po.option_value) filter (where po.option_name = 'autovacuum_enabled'), current_setting('autovacuum')
      ) as autovacuum_enabled,
      coalesce(
          max(po.option_value) filter (where po.option_name = 'autovacuum_analyze_scale_factor')
        , current_setting('autovacuum_analyze_scale_factor'))::float as autovacuum_analyze_scale_factor,
      coalesce(
          max(po.option_value) filter (where po.option_name = 'autovacuum_vacuum_scale_factor')
        , current_setting('autovacuum_vacuum_scale_factor'))::float as autovacuum_vacuum_scale_factor,
      coalesce(
          max(po.option_value) filter (where po.option_name = 'autovacuum_vacuum_threshold')
        , current_setting('autovacuum_vacuum_threshold'))::int8 as autovacuum_vacuum_threshold,
      coalesce(
         max(po.option_value) filter (where po.option_name = 'autovacuum_analyze_threshold')
       , current_setting('autovacuum_analyze_threshold'))::int8 as autovacuum_analyze_threshold
      from pg_class c
      LEFT JOIN pg_options_to_table(c.reloptions) po ON true -- lateral is implicit
      LEFT JOIN LATERAL (
        select relnamespace::regnamespace as relnspnametoast, relname as relnametoast from pg_class tc where tc.reltoastrelid =
c.oid) tr ON true
      Where c.relkind = ANY ('{r,m,t}'::char[])
      group by datname, c.relnamespace, c.relname, c.relkind, c.reltoastrelid, tr.relnspnametoast, tr.relnametoast
) rvac
order by rvac.row_estimate desc;
```

Autovacuum – znovu???

- Domácí úkol – report tabulek s vypnutým autovacuum na toast table…
  - Nápověda: `toast.autovacuum_enabled = true`

DB: autovacuum threshold

```
    table_name    | row_estimate | avsf | av_tuples_threshold
------------------+--------------+------+---------------------
 measurement_mid  |     43656168 |  0.2 |             8731284
 measurement_high |    491057280 | 0.01 |             4910622
 measurement_low  |      7307218 |  0.2 |             1461494
 blmeasurements   |       958449 |  0.2 |              191740


...

 ac_data          |          152 |  0.2 |                  80
 pg_aggregate     |          133 |  0.2 |                  77

...

 pg_toast_2606    |            0 |  0.2 |                  50
 pg_toast_1255    |            0 |  0.2 |                  50
 pg_toast_2620    |            0 |  0.2 |                  50
```

Instance: wraparound

- Dokumentace
- The maximum time that a table can go unvacuumed is two billion transactions minus the vacuum_freeze_min_age value **at the time of** the last aggressive vacuum.
- uvedená kontrola *předpokládá*, že se vacuum_freeze_min_age neměnilo

```
SELECT datname, age(datfrozenxid) as age,
(
    age(datfrozenxid)/(2*10^9-current_setting('vacuum_freeze_min_age')::int)
)::real as pct_to_wraparound
FROM pg_database;

  datname   |  age  | pct_to_wraparound
------------+-------+-------------------
 template0  | 64573 |       3.31144e-05
 postgres   | 64573 |       3.31144e-05
 hibernate  | 64573 |       3.31144e-05
```

DB: wraparound – table level

```
with relafma as(
select cropt.oid, cropt.ropt[2]::int from
   ( SELECT c.oid, string_to_array(unnest(c.reloptions), '=') as ropt FROM   pg_class c ) cropt
   where cropt.ropt[1] ilike 'autovacuum_freeze_max_age'
   ),
relvfma as(
select cropt.oid, cropt.ropt[2]::int from
   ( SELECT c.oid, string_to_array(unnest(c.reloptions), '=') as ropt FROM   pg_class c ) cropt
   where cropt.ropt[1] ilike 'vacuum_freeze_min_age'
   ),
pgcl as ( select c.oid, c.relnamespace, c.relkind,
       age(c.relfrozenxid) as tbl_age,
       age(t.relfrozenxid) as toast_age,
       greatest(age(c.relfrozenxid),age(t.relfrozenxid)) as age,
       coalesce(relvfma.ropt, current_setting('vacuum_freeze_min_age')::int) as tbl_vfma,
       coalesce(relafma.ropt, current_setting('autovacuum_freeze_max_age')::int) as tbl_afma
FROM pg_class c
LEFT JOIN pg_class t ON c.reltoastrelid = t.oid
left join relafma on c.oid = relafma.oid
left join relvfma on c.oid = relvfma.oid
)
  SELECT current_database() as datname, nspname AS schemaname,
       c.oid::regclass::text as table_name,
       c.relkind,
       c.tbl_age,
       c.toast_age,
       c.age,
       (c.age/(2*10^9 - c.tbl_vfma))::real as ptc_to_tbl_wraparound,
       (c.age / (least(c.tbl_afma, 2*10^9)))::real AS pct_to_tbl_aggressive_vacuum
FROM pgcl c
LEFT JOIN  pg_namespace n ON n.oid = c.relnamespace
WHERE c.relkind IN ('r', 'm')
order by age desc;
```

DB: wraparound

- pct_to_tbl_wraparound
  - lze najít která/é konkrétní tabulky jsou nejstarší
- pct_to_tbl_aggresive_vacuum
  - all-visible but not all-frozen pages are scanned

table_name	relkind	age	ptc_to_tbl_wraparound	pct_to_tbl_aggressive_vacuum
pg_type	r	64573	3.31144e-05	0.000322865
pg_authid	r	64573	3.31144e-05	0.000322865
pg_proc	r	64573	3.31144e-05	0.000322865
pg_class	r	64573	3.31144e-05	0.000322865
pg_statistic	r	64573	3.31144e-05	0.000322865

Instance: pg_stat_statements

- „must have“ extenze
- pro interaktivní práci VŽDY seřadit a pracovat jen s nejnáročnějšími dotazy
  - calls, total_time, rows
  - shared_blks_hit, shared_blks_read, temp_blks_read, temp_blks_written, blk_read_time, blk_write_time
  - min_time, max_time, mean_time, stddev_time
- pro monitoring sbírat s rozumnou periodou
  - vyhodnocujeme pak rozdíly za daný časový úsek
  - InfluxDB má pro tento účel fci non_egative_derivative()

Bloat monitoring

- MVCC vytváří verze řádek (update → insert + update (xmax))
- Vacuum označí již nepotřebné verze (dead tuples) za prázdné a přidá je do free space map → místo může být použito pro nová data
- Extenze: pgstattuple (exact/approx)
  - Interně ji používá například pg_bloat_check
- Select z check_postgres nebo pgsql-bloat-estimation - odhad z datata dictionary

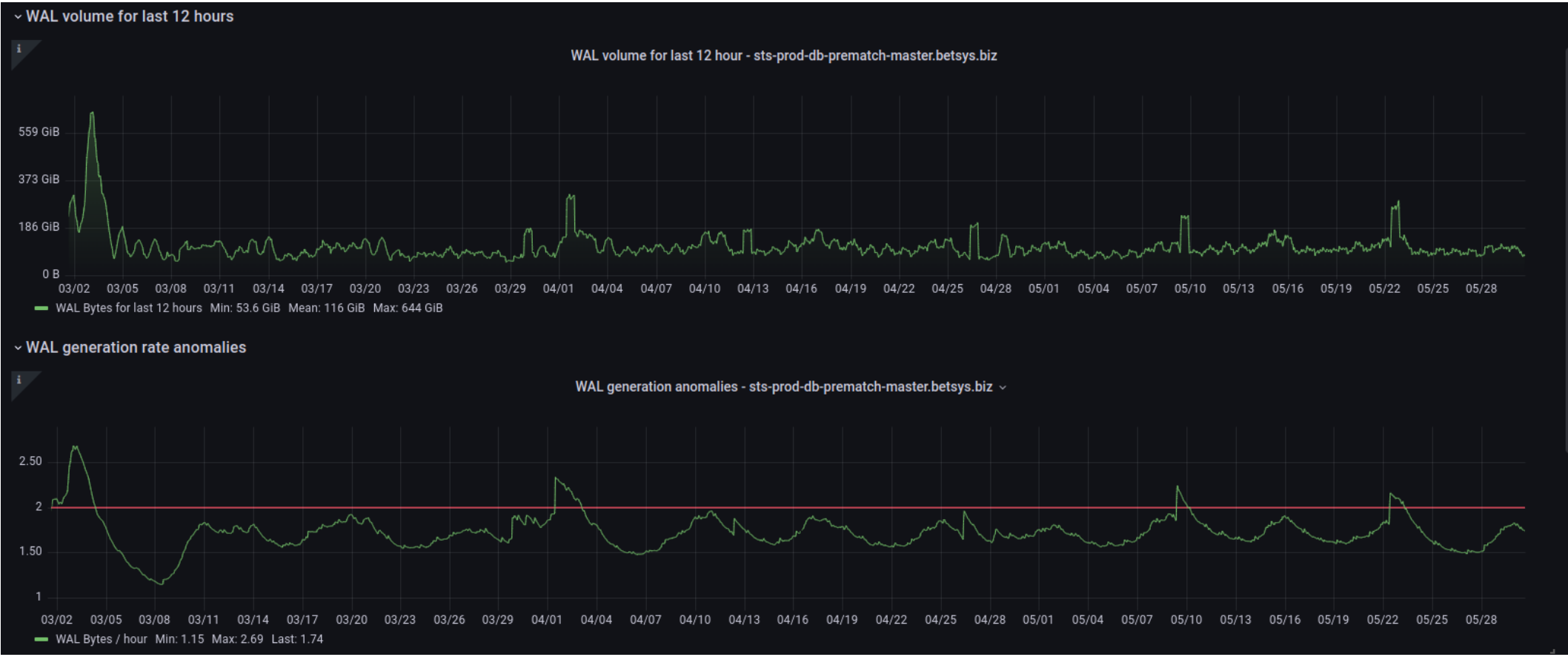
```
select pg_size_pretty(table_len) as table_len, scanned_percent, approx_tuple_count,
pg_size_pretty(approx_tuple_len) as approx_tuple_len, approx_tuple_percent, dead_tuple_count,
pg_size_pretty(dead_tuple_len) as dead_tuple_len, dead_tuple_percent, pg_size_pretty(approx_free_space) as
approx_free_space, approx_free_percent from pgstattuple_approx('public.mytable'::regclass);

 table_len | approx_tuple_count | approx_tuple_len | approx_tuple_percent | dead_tuple_count |
dead_tuple_len |  dead_tuple_percent  | approx_free_space | approx_free_percent
-----------+--------------------+------------------+----------------------+------------------
+----------------
 51 GB     |          259732969 | 21 GB            |     40.8085152577027 |              999 | 69 kB
 |

|  dead_tuple_percent  | approx_free_space | approx_free_percent
+----------------------+-------------------+---------------------
| 0.000130408286945782 | 30 GB             |    59.1854516900596
```

Pg-exporter, Promethehus, Grafana

- Postgres-exporter
  - Vestavěný sběr klíčových metrik
  - Možnost rozšiřujících dotazů
- Pometheus
  - Egektivní uložení časových řad
  - PromQL
- Grafana
  - Integrace s Promethus, vizualizace

Pometheus & Grafana

- Objem WAL (replika – recovery_min_apply_delay = 12h)
- Z score (standardní skóre)



Exporter bloat estimate query

```
sql_exporter_bloat_interval: 2h
sql_exporter_bloat_queries:
  - name: "table_bloat"
    help: "Monitoring tables bloat using estimates from catalog"
    labels:
      - "datname"
      - "schemaname"
      - "tblname"
      - "is_na"
    values:
      - "real_size"
      - "bloat_size"
      - "bloat_ratio"
      - "last_analyzed"
      - "last_analyzed_age"
    query: |
      /* https://github.com/ioguix/pgsql-bloat-estimation/blob/master/table/table_bloat.sql
       * WARNING executed with a non-superuser role, the query inspect only tables and materialized view (9.3+) you are
granted to read.
       * This query is compatible with PostgreSQL 9.0 and more
       */
      SELECT current_database() AS datname, s3.schemaname, s3.tblname
        --, CASE WHEN is_na THEN 'NaN'::float ELSE s3.bs*s3.tblpages END AS real_size,
        , s3.bs*s3.tblpages AS real_size,
...
```

Dotazy na WAL volume a Zscore

- PromQL:

```
delta(pg_stat_archiver_archived_count{instance=~"$instance:
$pg_exp_port"}[12h])*16*1048576
```

- PromQL:

```
(
   pg_stat_archiver_archived_count{instance=~"$instance"}
   -
avg_over_time(pg_stat_archiver_archived_count{instance=~"$in
stance"}[1w])
)
/
stddev_over_time(pg_stat_archiver_archived_count{instance=~"
$instance"}[1w])
```

Bloat monitoring

Extennze pg_stat_statements

- Konfigurace:
  - shared_preload_libraries
  - pg_stat_statements.track (top, all, none) all – sleduje statistiky i pro vnořené dotazy ve funkcích
- `log_line_prefix %Q` – `queryid` (může logovat automaticky, pokud detekuje extenzi)
- Samples (POWA…)
- Nemá smysl `select *`
- Příklady: Cybertec blog
  - Cybertec: PgWatch2 – předpřipravené řešení pro monitoring

Vacuum progress

```
SELECT pcr.oid::regclass as vacuumed_rel,
    CASE pcr.reltoastrelid
        WHEN 0 THEN
            pcr.oid::regclass
        ELSE
            pcr.reltoastrelid::regclass
    END AS table_name
    , pssv.*
FROM pg_stat_progress_vacuum pssv
LEFT JOIN pg_class pcr ON ( pssv.relid = pcr.oid
OR pssv.relid = pcr.reltoastrelid );
```

Index creation progress

```
SELECT
    pci.pid
  , pci.relid::regclass
  , pci.index_relid::regclass
  , pci.command
  , pci.phase
  , pci.lockers_total
  , round(pci.lockers_done::numeric/nullif(pci.lockers_total, 0) *100, 2)
AS PCT_lockers
  , pci.current_locker_pid
  , round( blocks_done::numeric/nullif(blocks_total, 0) *100, 2) as
PCT_BLOCKS
  , round(tuples_done::numeric/nullif(tuples_total, 0) *100, 2) as
PCT_TUPLES
  , round(partitions_done::numeric/nullif(partitions_total, 0) *100 , 2) as
PCT_PARTITIONS
FROM pg_stat_progress_create_index pci;
```

Table size monitoring

- Velké tabulky/malé/všechny ?
  - Obvykle je více než velikost důležitá změna v čase

```
select bucket, count(*), pg_size_pretty(sum(relbytes)), pg_size_pretty(min(relbytes)) as bucket_min,
pg_size_pretty(max(relbytes)) as bucket_max
from
(
select c.relnamespace::regnamespace, c.relname, pg_total_relation_size(c.oid) relbytes,
ntile(10) over W as bucket
from pg_class c
WHERE
c.relkind = 'r'
window W AS (order by pg_total_relation_size(c.oid))
) rs
group by bucket
order by bucket;
```

Table size monitoring

- Fun queries

```
select bucket, count(*), pg_size_pretty(sum(relbytes)), pg_size_pretty(min(relbytes)) as bucket_min,
pg_size_pretty(max(relbytes)) as bucket_max
from
(
select c.relnamespace::regnamespace, c.relname, pg_total_relation_size(c.oid) relbytes,
ntile(10) over W as bucket
from pg_class c
WHERE
c.relkind = 'r'
window W AS (order by pg_total_relation_size(c.oid))
) rs
group by bucket
order by bucket;
```

```
 bucket | count | pg_size_pretty | bucket_min | bucket_max
--------+-------+----------------+------------+------------
      1 |   219 | 1368 kB        | 0 bytes    | 16 kB
      2 |   219 | 3504 kB        | 16 kB      | 16 kB
      3 |   219 | 4864 kB        | 16 kB      | 24 kB
      4 |   219 | 5600 kB        | 24 kB      | 32 kB
      5 |   219 | 8032 kB        | 32 kB      | 48 kB
      6 |   219 | 15 MB          | 48 kB      | 176 kB
      7 |   219 | 876 MB         | 200 kB     | 7640 kB
      8 |   219 | 26 GB          | 7648 kB    | 253 MB
      9 |   219 | 77 GB          | 253 MB     | 444 MB
     10 |   219 | 1555 GB        | 445 MB     | 338 GB
(10 rows)
```

Table size monitoring

- Fun queries...

```
select bucket, count(*), pg_size_pretty(sum(relbytes)), pg_size_pretty(min(relbytes)) as bucket_min,
pg_size_pretty(max(relbytes)) as bucket_max
from (
select c.relnamespace::regnamespace, c.relname, pg_total_relation_size(c.oid) relbytes,
dense_rank() over W as bucket
from pg_class c
WHERE
c.relkind = 'r'
and pg_total_relation_size(c.oid) > 0
window W AS (order by round(log(pg_total_relation_size(c.oid))))
) rs
group by bucket
order by bucket;
```

```
 bucket | count | pg_size_pretty | bucket_min | bucket_max
--------+-------+----------------+------------+------------
      1 |   708 | 14 MB          | 8192 bytes | 24 kB
      2 |   497 | 28 MB          | 32 kB      | 304 kB
      3 |    50 | 61 MB          | 344 kB     | 2512 kB
      4 |   177 | 1091 MB        | 3248 kB    | 29 MB
      5 |   245 | 39 GB          | 32 MB      | 295 MB
      6 |   352 | 255 GB         | 304 MB     | 2899 MB
      7 |    27 | 379 GB         | 3874 MB    | 29 GB
      8 |     8 | 646 GB         | 38 GB      | 144 GB
      9 |     1 | 338 GB         | 338 GB     | 338 GB
(9 rows)
```

Dotazy ?

Ukázky dotazů pro jednotlivé verze jsou na dalších stranách

pg_stat_activity 9.1

```
SELECT datid, datname, usesysid, usename, application_name,
client_addr, client_hostname, client_port, backend_start,
extract(epoch from current_timestamp - xact_start) as
xact_duration,
CASE
   WHEN current_query = '<IDLE>' THEN 'idle'
   WHEN current_query = '<IDLE> in transaction' THEN 'idle in
transaction'
   ELSE  'unknown'
END as state,
CASE
   WHEN current_query = '<IDLE>' THEN null
   ELSE extract(epoch from current_timestamp - query_start)
END as query_duration,
waiting,
CASE
   WHEN current_query = '<IDLE>' THEN null
   ELSE current_query
END as query_text
FROM pg_stat_activity;
```


-- InfluxDB tags jsou kurzívou

pg_stat_activity 9.2 – 9.3

```
SELECT datid, datname, pid, usesysid, usename, application_name,
client_addr, client_hostname, client_port, backend_start,
extract(epoch from current_timestamp - xact_start) as
xact_duration, state_change, waiting, state,
CASE
   WHEN state = 'idle' THEN null
   ELSE extract(epoch from current_timestamp - query_start)
END as query_duration,
CASE state
   when 'idle' THEN null
   ELSE query
END as query_text
  FROM pg_stat_activity;
```

pg_stat_activity 9.4 – 9.5

```
SELECT datid, datname, pid, usesysid, usename, application_name,
client_addr, client_hostname, client_port, backend_start,
extract(epoch from current_timestamp - xact_start) as
xact_xact_duration,
state_change, waiting, state,
CASE
   WHEN state = 'idle' THEN null
   ELSE extract(epoch from current_timestamp - query_start)
END as query_duration,
backend_xid, backend_xmin,
CASE state
   when 'idle' THEN null
   ELSE query
END as query_text
  FROM pg_stat_activity;
```

pg_stat_activity 9.6

```
SELECT datid, datname, pid, usesysid, usename, application_name,
client_addr, client_hostname, client_port, backend_start,
extract(epoch from current_timestamp - xact_start) as
xact_duration,
CASE
   WHEN state = 'idle' THEN null
   ELSE extract(epoch from current_timestamp - query_start) END
as query_duration   ,
state_change,
CASE
   WHEN wait_event_type is null THEN false
   ELSE true
END as waiting,
wait_event_type, wait_event, state, backend_xid, backend_xmin,
CASE
   when state = 'idle' THEN null
   ELSE query
END as query_text
FROM pg_stat_activity;
```

Pg_stat_archiver 9.4 – 9.6

```
select
    archived_count, last_archived_wal,
    extract(epoch from current_timestamp - last_archived_time)::int as last_arch_sec_age,
    failed_count, last_failed_wal,
    case when (last_failed_wal IS NULL OR last_failed_wal <= last_archived_wal) then
        null
    else
        extract(epoch from current_timestamp - last_failed_time)::int
    end as last_failed_sec_age,
    stats_reset,
    (
        current_setting('archive_mode')::BOOLEAN
        AND    (   last_failed_wal IS NULL
               OR
                   last_failed_wal <= last_archived_wal
               )
     )AS is_archiving,
    -- „xlog volume appendix“
    pg_xlog_location_diff(
        pg_current_xlog_location(), '0/00000000'::pg_lsn
    ) as xlog_volume
from pg_stat_archiver;
```

pg_stat_bgwriter

- 9.1

```
SELECT checkpoints_timed, checkpoints_req, buffers_checkpoint,
buffers_clean, maxwritten_clean, buffers_backend,
buffers_backend_fsync, buffers_alloc, stats_reset
    FROM pg_stat_bgwriter;
```

- 9.2 – 9.6

```
SELECT
    checkpoints_timed,
    checkpoints_req,
    checkpoint_write_time,
    checkpoint_sync_time,
    buffers_checkpoint,
    buffers_clean,
    maxwritten_clean,
    buffers_backend,
    buffers_backend_fsync,
    buffers_alloc,
    stats_reset
FROM pg_stat_bgwriter;
```

pg_stat_database

- 9.1

```
SELECT datid, datname, numbackends, xact_commit, xact_rollback,
blks_read, blks_hit, tup_returned, tup_fetched, tup_inserted,
tup_updated, tup_deleted, conflicts, stats_reset
  FROM pg_stat_database;
```

- 9.2 – 9.6

```
SELECT
    datid, datname,
    numbackends,
    xact_commit, xact_rollback,
    blks_read, blks_hit,
    tup_returned, tup_fetched, tup_inserted,
    tup_updated, tup_deleted,
    conflicts,
    temp_files, temp_bytes,
    deadlocks,
    blk_read_time, blk_write_time,
    stats_reset
FROM pg_stat_database;
```

pg_stat_user_tables

- 9.1 – 9.6
  - data pro připojenou DB
  - obdobné dotazy pro indexy, funkce, sekvence...

```
select
    current_database() as datname,
    relid, schemaname, relname,
    seq_scan, seq_tup_read,
    idx_scan, idx_tup_fetch,
    n_tup_ins, n_tup_upd, n_tup_del,
    n_tup_hot_upd, n_live_tup, n_dead_tup,
    last_vacuum, last_autovacuum,
    last_analyze, last_autoanalyze,
    vacuum_count, autovacuum_count,
    analyze_count, autoanalyze_count
from pg_stat_user_tables;
```

~~~ definitivní konec presentace ~~~
~~~