Pro-active performance analysis in PostgreSQL

«Before the migration from Oracle to PostgreSQL everything was faster! ☹ »

Dirk Krautschick

Prague PostgreSQL Developer Day
02. June 2022

trivadis
Part of Accenture

ABOUT ME...

3 HALLO, GRÜEZI, HI!

@d33_k4y

DIRK KRAUTSCHICK

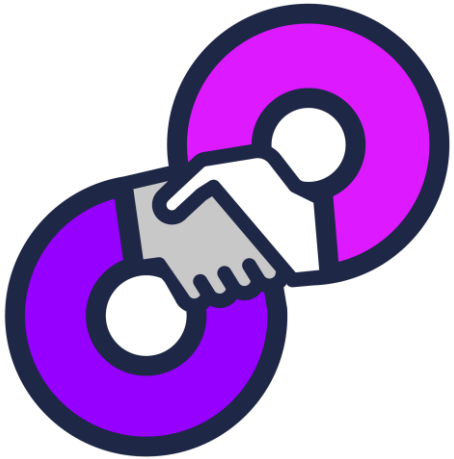
- since 03/2019 @Trivadis, Germany, Düsseldorf
- PostgreSQL & Oracle, Trainer
- 13 years DBA & Consulting

- Married, 2 Junior DBAs
- Mountainbike, swimming, movies, music, hifi/home cinema, 8bit computing

ABOUT US...

trivadis
Part of Accenture

5 TRIVADIS & ACCENTURE: #1 FOR DATA & AI



- Together we are 1500 specialists at 34 locations in Switzerland, Germany and Austria with a focus on **Data & Applied Intelligence**.
- Together we support you in the **intelligent end-to-end use of your data.**
- We cover the entire spectrum: from the **development** and **operation** of **data platforms** and **solutions**, to the refinement of data as well as **consulting** and **training**.
- We achieve this through the unique combination of Trivadis' **technological expertise** and Accenture's **strategic know-how** in the field of data.

trivadis
Part of **Accenture**

WARM UP

WARM UP
THE PERFORMANCE PROBLEM

- Usual daily task of basically every DBA
- Several variants of problem descriptions by users
- Challenging task
- Depending of user based details, like
  - What is exactly slow?
  - When was it slow?
  - What did happen before or during the problem?

WARM UP
THE PERFORMANCE PROBLEM

- Usual reasons
  - Bad sizing
  - Data diversification
  - Application changes
- Special reasons
  - New database releases, -features
  - Migrations

WARM UP
POST-MIGRATION SITUATION

- Different behaviour
  - Plan decisions
  - Use of Indexes
- Being spoiled by tooling from commercial databases
  - Example: Oracle Diagnostic Pack
    - Automatic Workload Repository (AWR)
    - Automatic Database Diagnostic Monitor (ADDM)
    - Active Session History (ASH)

WARM UP
PREVENTION OF PERFORMANCE PROBLEMS?

- KIWI (Kill it with Iron..or just Idle)?
- Invest in a detailed initialization/testing phase
- Experience and cooperation with
  - Developement
  - Application-Owner
  - System Engineers
  - Database Administrators
- Adaptive.../Autonomous...blablabla....
  - Not with PostgreSQL....yet!

WARM UP
THE PROBLEM REPORT DILEMMA

- *«The system is currently slow!»* ….bad
- *«The system was slow!»* ….even worse

The Craftmanship Statement Based Analysis

THE CRAFTMANSHIP
STATEMENT BASED ANALYSIS

- Most granular point of investigation
- Slow statement is exactly identified
  - One time problem
  - General runtime problem of the statement
- Reproducable?
- Time or data situation related?

Digging in explain plans!

THE CRAFTMANSHIP
EXPLAIN PLAN

- Detailed description how PostgreSQL performs a statement
- Result of the „query planner“ AKA optimizer based on
  - Statistics
  - Settings
  - Meta data

THE CRAFTMANSHIP
EXPLAIN PLAN

- Viewing explain plan with **`explain`** (dry run)

```
scott_db=# explain select e.name, d.name from scott.dept d,scott.emp e where d.deptno=e.deptno;

                            QUERY PLAN
------------------------------------------------------------------
 Hash Join  (cost=1.09..2.28 rows=14 width=84)
   Hash Cond: (e.deptno = d.deptno)
   ->  Seq Scan on emp e  (cost=0.00..1.14 rows=14 width=42)
   ->  Hash  (cost=1.04..1.04 rows=4 width=50)
         ->  Seq Scan on dept d  (cost=0.00..1.04 rows=4 width=50)
(5 rows)
```

- More runtime details with **`explain analyze`** (while executing the statement!)

THE CRAFTMANSHIP
EXPLAIN PLAN – TRY THE OPTIONS

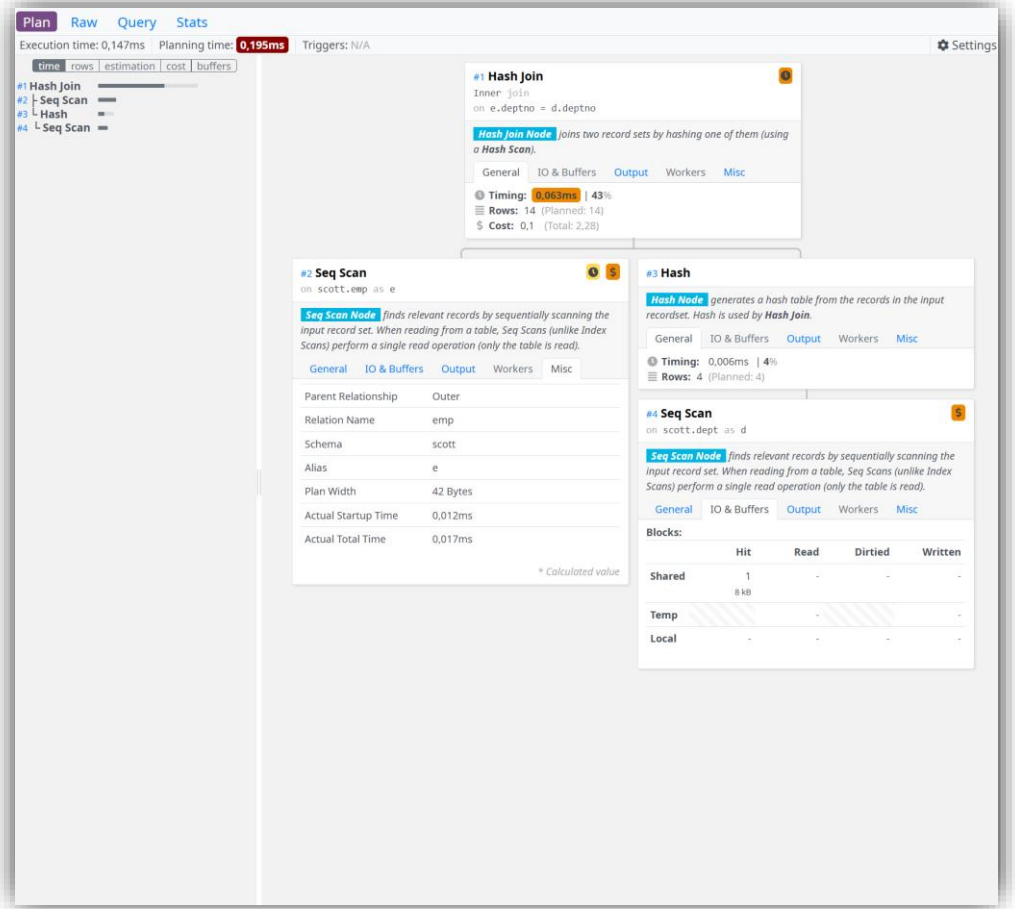
OPTION	MEANING	DEFAULT
ANALYZE	SHOW RUN TIME AND OTHER STATISTICS	FALSE
VERBOSE	MORE DETAILLED INFORMATION	FALSE
COSTS	COST INFORMATION FOR EACH STEP	TRUE
SETTINGS	INFORMATION ABOUT RELATED CONFIGURATION PARAMETERS	FALSE
BUFFERS	REPORTS BUFFER USAGE	FALSE
WAL	INFORMATION ABOUT WAL RECORD GENERATION	FALSE
TIMING	REPORTS MORE DETAILS ABOUT TIME SPEND IN EACH NODE	TRUE
SUMMARY	REPORTS OVERALL TIME (PLANNING AND EXECUTION)	TRUE
FORMAT (TEXT|XML|YAML|JSON)	OUTPUT FORMAT	TEXT

17 THE CRAFTMANSHIP
EXPLAIN PLAN

```
scott_db=# explain (analyze, verbose, buffers true, wal true, settings true, format text)
           select e.name, d.name from scott.dept d,scott.emp e where d.deptno=e.deptno;
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=1.09..2.28 rows=14 width=84) (actual time=1.168..1.197 rows=14 loops=1)
   Output: e.name, d.name
   Inner Unique: true
   Hash Cond: (e.deptno = d.deptno)
   Buffers: shared hit=3 read=2
   I/O Timings: read=1.036
   ->  Seq Scan on scott.emp e  (cost=0.00..1.14 rows=14 width=42) (actual time=0.714..0.720 rows=14 loops=1)
         Output: e.empno, e.name, e.job, e.manager, e.hiredate, e.salary, e.comm, e.deptno
         Buffers: shared read=1
         I/O Timings: read=0.692
   ->  Hash  (cost=1.04..1.04 rows=4 width=50) (actual time=0.373..0.374 rows=4 loops=1)
         Output: d.name, d.deptno
         Buckets: 1024  Batches: 1  Memory Usage: 9kB
         Buffers: shared read=1
         I/O Timings: read=0.344
         ->  Seq Scan on scott.dept d  (cost=0.00..1.04 rows=4 width=50) (actual time=0.357..0.359 rows=4 loops=1)
               Output: d.name, d.deptno
               Buffers: shared read=1
               I/O Timings: read=0.344
 Planning:
   Buffers: shared hit=91 read=20
   I/O Timings: read=14.577
 Planning Time: 23.734 ms
 Execution Time: 3.090 ms
```

trivadis
Part of Accenture

THE CRAFTMANSHIP
EXPLAIN PLAN



- Explain plan reading could be hard
- Try block based

- Visualizing tools are highly recommended
  - https://explain.dalibo.com/
  - https://tatiyants.com/pev

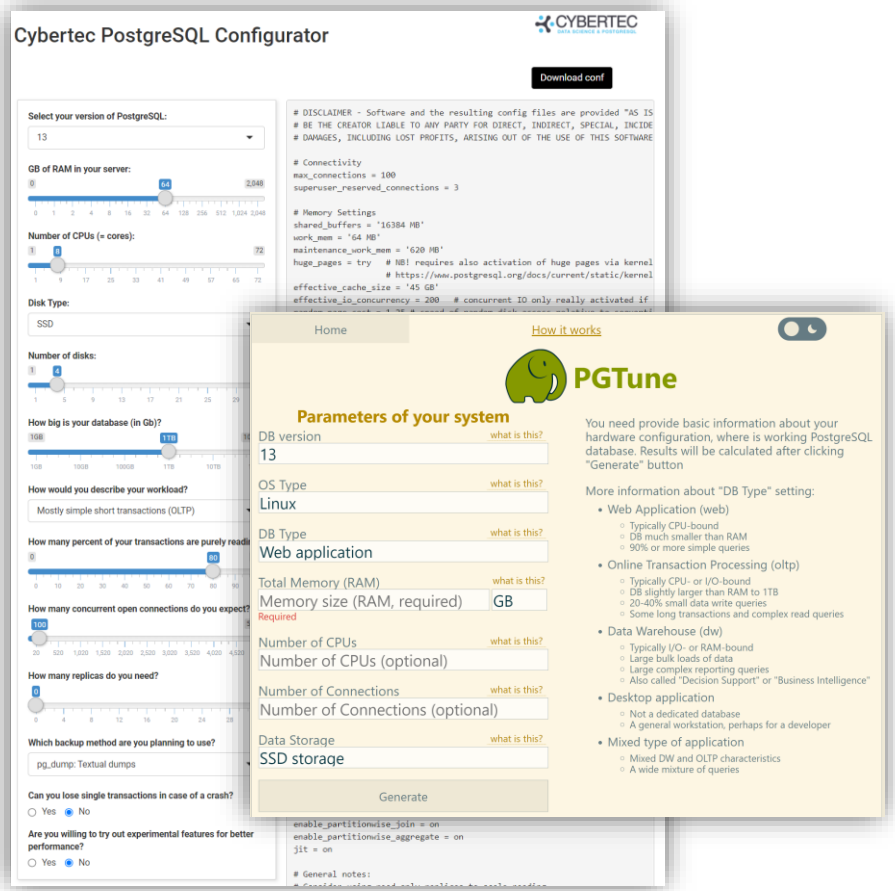
trivadis
Part of Accenture

THE CRAFTMANSHIP
USUAL RECOGNITIONS

- Not optimal access methods
  - Wrong or missing index strategy
  - Poorly written SQL
  - Query Planner settings
- Variant estimations
  - Old or useless object statistics
  - VACUUM configuration

Getting Pro-Active

21 GETTING PRO-ACTIVE
FIRST OF ALL…SIZING



- PostgreSQL defaults are pretty …lightweight
- Frequently review of configuration
- Consider parallelization
- For simple initial help
  - http://pgconfigurator.cybertec.at
  - https://pgtune.leopard.in.ua/#/

trivadis
Part of Accenture

GETTING PRO-ACTIVE
THE IDEA

- Recognizing slow performance before the users
- Possibility to check past performance situations
- Permanent tracking of
  - Statement details
  - Explain plans
  - Load statistics
- No need of reproducing scenarios

23 GETTING PRO-ACTIVE MONITORING

- For sure, monitoring is essential
- „The fact that it's slow or what is exactly slow?"
- Necessary PostgreSQL insights
  - e.g. pg_stat_* views
- Helpful Tools
  - https://pg_top.gitlab.io/
  - https://github.com/dalibo/pg_activity
  - Not pro-active but still helpful
- Example
  - https://pgwatch.com/

trivadis
Part of Accenture

24 GETTING PRO-ACTIVE
MASSIVE LOGGING

```
log_line_prefix = '%t [%p]: user=%u,db=%d,app=%a,client=%h ,
…
log_parser_stats = off
log_planner_stats = off
log_executor_stats = off
log_statement_stats = on
…
log_checkpoints = on
log_connections = on
log_disconnections = on
log_lock_waits = on
log_temp_files = 0
log_autovacuum_min_duration = 0
log_error_verbosity = default
…
log_min_messages = debug5
log_min_error_statement = debug5

log_min_duration_statement = 0
log_min_duration_sample = 0
…
log_statement = 'all'
```

- Exhaustive logging possibilities
- Straight and easy configuration
- Be aware of storage and load
  - High maintenance
  - Evaluate Logging strategies
- postgresql.org/docs/13/runtime-config-logging.html

GETTING PRO-ACTIVE
MASSIVE LOGGING – AUTO EXPLAIN

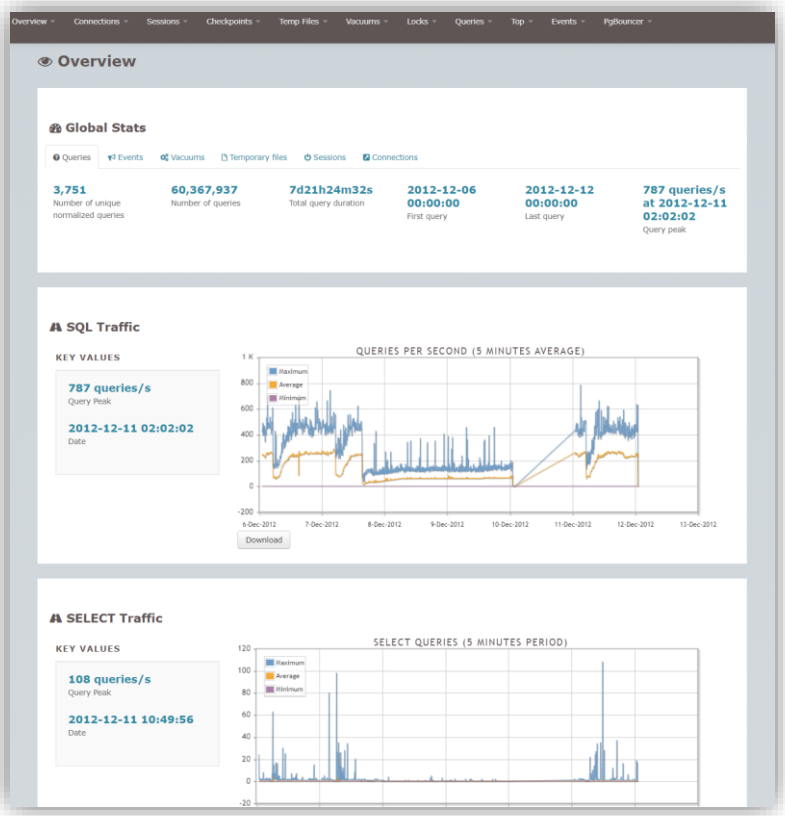
```
LOAD 'auto_explain';

auto_explain.log_min_duration = '3s'

set auto_explain.log_analyze  = true;
set auto_explain.log_buffers  = true;
set auto_explain.log_wal      = true;
set auto_explain.log_timing   = true;
set auto_explain.log_verbose  = true;
set auto_explain.log_settings = true;
set auto_explain.log_format   = 'text';
...
```

- Load the Auto Explain module
- Or add in parameter `session_preload_libraries`
- Configuration like usual Explain Analyze
- Same here
  - High maintenance
- postgresql.org/docs/13/auto-explain.html

26 GETTING PRO-ACTIVE
PGBADGER



- Advanced logging analysis reporting
  - https://pgbadger.darold.net/
- Graphical reports based on logfiles
- Incremential daily reports
- Cumulative reports per week

trivadis
Part of Accenture

GETTING PRO-ACTIVE

ALL GOOD, BUT WE NEED MORE...

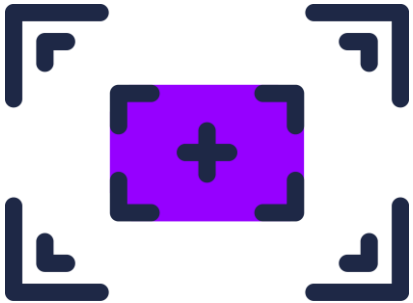
- What exactly is going on...
- Runtime data of overall statements...
- Where is the waste of performance...
- Detailed load analysis....

- Somehow it is all in the catalogues, but...
  - ...not so easy to find!
  - ...not so easy to keep!

The Power of Extensibility

THE POWER OF EXTENSIBILITY
PG_STAT_STATEMENTS

- No pointed reason for slowness
- Need of an overview of runtime statistics of all statements
- Period view
- Finding the long running queries



THE POWER OF EXTENSIBILITY
PG_STAT_STATEMENTS

- Statement level statistics
- Required by several monitoring tools
- Statement based collection of e.g.
  - Executions
  - Execution times (min, max, average)
  - Rows
  - Blocks read/write

THE POWER OF EXTENSIBILITY
PG_STAT_STATEMENTS

```
# show shared_preload_libraries;
  shared_preload_libraries | pg_stat_statements

# create extension pg_stat_statements;

# \d pg_stat_statements
                    View "public.pg_stat_statements"
       Column        |       Type       | ...
---------------------+------------------+-...
 userid              | oid              | ...
 dbid                | oid              | ...
 queryid             | bigint           | ...
 query               | text             | ...
 total_plan_time     | double precision | ...
…
 calls               | bigint           | ...
 total_exec_time     | double precision | ...
 min_exec_time       | double precision | ...
 max_exec_time       | double precision | ...
 mean_exec_time      | double precision | ...
 stddev_exec_time    | double precision | ...
 rows                | bigint           | ...
…                                       ...
 blk_read_time       | double precision | ...
 blk_write_time      | double precision | ...
…
```

- Installation
  - Install contrib package from repository
  - Add shared_preload_libraries entry
  - Restart cluster
  - Create extension…
- Optional configuration and functions
  - `pg_stat_statements_reset()`
- postgresql.org/docs/13/pgstatstatements.html

THE POWER OF EXTENSIBILITY
PG_STAT_STATEMENTS - EXAMPLE

```
# select
      substring(query, 1, 50) as short_query,
      round(total_exec_time) as total_exec_time, calls,
      round(mean_exec_time) as mean_exec_time,
      round(100 * total_exec_time / (select sum(total_exec_time) from pg_stat_statements)) as percentage
  from
      pg_stat_statements
  order by percentage desc;
```

```
 short_query                                        | total_exec_time | calls | mean_exec_time | percentage
----------------------------------------------------+-----------------+-------+----------------+------------
 UPDATE pgbench_branches SET bbalance = bbalance +  |            7114 |  1500 |              5 |         63
 UPDATE pgbench_tellers SET tbalance = tbalance + $ |            2506 |  1500 |              2 |         22
 copy pgbench_accounts from stdin                   |             664 |     1 |            664 |          6
 UPDATE pgbench_accounts SET abalance = abalance +  |             194 |  1500 |              0 |          2
 alter table pgbench_accounts add primary key (aid) |             193 |     1 |            193 |          2
 vacuum analyze pgbench_accounts                    |             138 |     1 |            138 |          1
…
```

THE POWER OF EXTENSIBILITY
PG_WAIT_SAMPLING

```
INSERT INTO ... (
SELECT DISTINCT ...,
    CASE
        WHEN COUNT(...) OVER (PARTITION BY ...) >= 1
            THEN 2
        WHEN COUNT(...) OVER (PARTITION BY ...) = 0
            THEN 3
    END
FROM ...
...)
/* lots of other JOINS and whatsovever.....
JOIN (
    SELECT ..., COUNT (*) AS ...
    FROM ...
    JOIN (
      SELECT DISTINCT ...
      FROM ...
    ) ... ON ... = ...
    LEFT JOIN ... ON ... = ...
    LEFT JOIN ... ON ... = ...
    WHERE ... = '...'
        AND ... IN ('...', 'Rev...iew')
        AND ... BETWEEN ... AND ...
        AND ... IS NOT NULL
        AND ... IS NOT NULL
        AND ... = '...'
        AND ... IS NOT NULL
    GROUP BY ...
) ... ON ... = ...AND ... = ...)
);
```

- Customer after migration from Oracle DB
- Bad performance in benchmark scripts
- Customer was pointing on
  - Wrong indexing
  - Lots of full table scans
- Where is the waste of time?

THE POWER OF EXTENSIBILITY
PG_WAIT_SAMPLING

- Sampling based statistics of wait events
- Combination with pg_stat_statements possible
- github.com/postgrespro/pg_wait_sampling

THE POWER OF EXTENSIBILITY
PG_WAIT_SAMPLING

```
$ git clone github.com/postgrespro/pg_wait_sampling.git
$ cd pg_wait_sampling
$ make USE_PGXS=1
$ sudo make USE_PGXS=1 install
$ make USE_PGXS=1 installcheck

$ psql DB -c "CREATE EXTENSION pg_wait_sampling;"
```

- Installation
  - repositories or compilation (see left)
  - Add shared_preload_libraries entry
  - Restart cluster
  - Create extension…

THE POWER OF EXTENSIBILITY
PG_WAIT_SAMPLING

```
postgres=# select * from pg_wait_sampling_profile order by pid, count desc;

pid  | event_type |         event         |       queryid        |  count
-----+------------+-----------------------+----------------------+---------
1683 | Activity   | CheckpointerMain      |  1511417639870010300 | 3997957
1683 | IO         | DataFileWrite         | -5761583335759906029 |      30
1683 | IO         | DataFileSync          | -5761583335759906029 |       3
1683 | IO         | DataFileFlush         | -4888004026240188267 |       1
1684 | Activity   | BgWriterHibernate     |  2862011717192834034 | 3921477
1684 | Activity   | BgWriterMain          | -4888004026240188267 |   88494
1685 | Activity   | WalWriterMain         |  2862011717192834034 | 4010099
1685 | IO         | WALSync               |  6648255685428052402 |      13
1686 | Activity   | AutoVacuumMain        | -4888004026240188267 | 4007477
1689 | Activity   | LogicalLauncherMain   |  2862011717192834034 | 4010493
3537 | IO         | DataFileRead          | -4888004026240188267 |       1
3546 | IO         | DataFileRead          |  6648255685428052402 |       3
3546 | Client     | ClientRead            | -4888004026240188267 |       3
3546 | IO         | DataFileRead          |  2862011717192834034 |       1
3720 | Client     | ClientRead            | -4888004026240188267 |    2393
...
```

- Views
  - pg_wait_sampling_current
  - pg_wait_sampling_history
  - pg_wait_sampling_profile
- Functions
  - pg_wait_sampling_get_current(pid)
  - pg_wait_sampling_reset_profile()

THE POWER OF EXTENSIBILITY
PG_WAIT_SAMPLING

```
postgres=# select
             pid,
             event_type,
             event,
             count
           from
             pg_wait_sampling_profile
           where
             pid = 3300169;
```

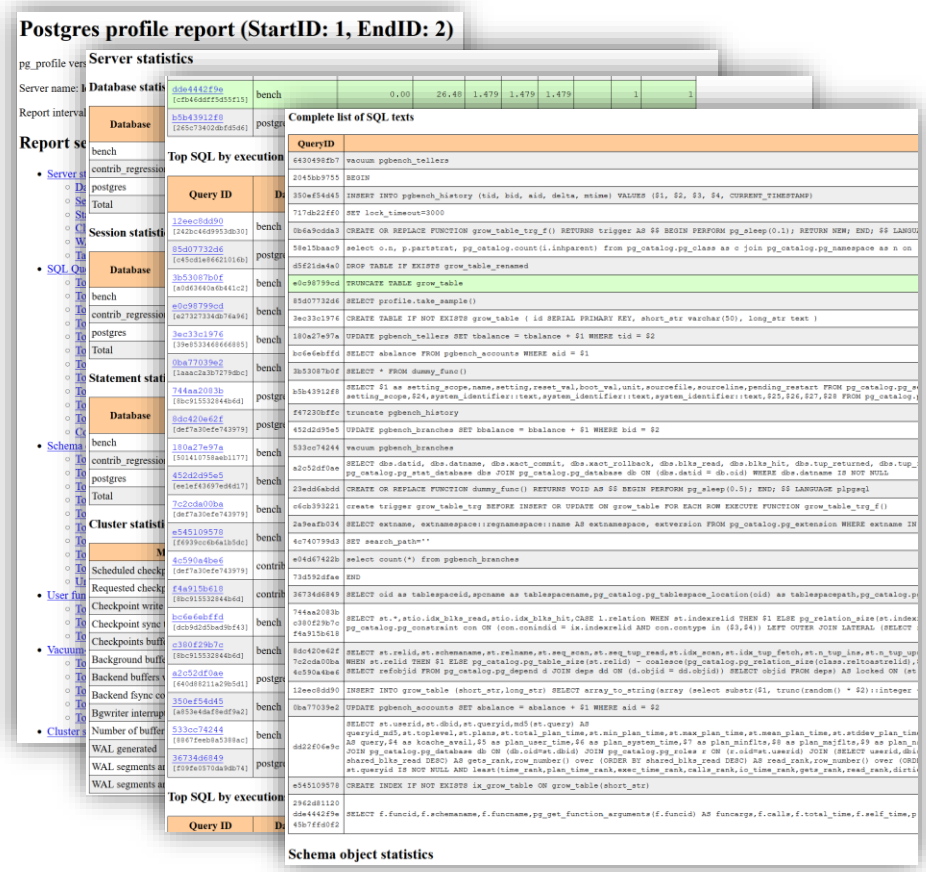
pid	event_type	event	count
3300169	IO	DataFileRead	63
3300169	IO	BufFileWrite	3620
3300169	IO	BufFileRead	1931
3300169	IO	DataFileExtend	21
3300169	LWLock	WALWrite	3
3300169	IO	DataFileWrite	1
3300169	IO	WALWrite	1
3300169	IO	WALSync	1
3300169	LWLock	WALBufMapping	1
3300169	Client	ClientRead	1

- Customer Example
- Buffer File Read/Write
- Idea
  - Prevent writing temp files
- Solution
  - Increase of work_mem size
- Statement runtime from 15 min to 3 min

THE POWER OF EXTENSIBILITY
PG_PROFILE

- So far so good, but still too pointy
- Time window based workload
- Compare abilities
- Just picking a period to check the situation

39 THE POWER OF EXTENSIBILITY
PG_PROFILE



- https://github.com/zubkov-andrei/pg_profile
- Sample (snapshot) based repository
- Regular and differential reports
- Required extensions
  - pg_stat_statements
  - pg_stat_kcache (if needed)
- Like Oracle Statspack / AWR
- Integration of pg_wait_sampling pending

trivadis
Part of Accenture

40 THE POWER OF EXTENSIBILITY
PG_STAT_KCACHE

```
$ git clone github.com/powa-team/pg_stat_kcache.git
$ cd pg_stat_kcache
$ make
$ make install

$ psql DB -c "CREATE EXTENSION pg_stat_kcache;"
```

- Statstics about reads and writes on filesystem
- Statistics about CPU usage
- pg_stat_statements is required
- Installation
  - repositories or compilation (see left)
  - shared_preload_libraries entry

- github.com/powa-team/pg_stat_kcache

THE POWER OF EXTENSIBILITY
PG_STAT_KCACHE

```
postgres=# \d pg_stat_kcache_detail;
                View "public.pg_stat_kcache_detail"
      Column      |       Type       |
------------------+------------------+-
 query            | text             | ...
 top              | boolean          | ...
 datname          | name             | ...
 rolname          | name             | ...
 plan_user_time   | double precision | ...
 plan_system_time | double precision | ...
 exec_user_time   | double precision | ...
 exec_system_time | double precision | ...
 exec_reads       | bigint           | ...
 exec_reads_blks  | bigint           | ...
 exec_writes      | bigint           | ...
 exec_writes_blks | bigint           | ...
...

postgres=# \d pg_stat_kcache;
                    View "public.pg_stat_kcache"
      Column      |       Type       |
------------------+------------------+-...
 datname          | name             | ...
 plan_user_time   | double precision | ...
 plan_system_time | double precision | ...
 plan_nswaps      | numeric          | ...
 exec_user_time   | double precision | ...
 exec_system_time | double precision | ...
 exec_reads       | numeric          | ...
 exec_reads_blks  | numeric          | ...
 exec_writes      | numeric          | ...
 exec_writes_blks | numeric          | ...
...
```

- Views
  - pg_stat_kcache
  - pg_stat_kcache_detail
- Functions
  - pg_stat_kcache_reset()
  - pg_stat_kcache()

My Strategy

MY STRATEGY
BEING PREPARED

- Pre-install extension on all clusters per default
  - pg_stat_statements
  - pg_wait_sampling
  - pg_stat_kcache (if needed)
- Default configuration, just in case
- Consider the use of pg_profile

MY STRATEGY
BEING PREPARED

- Review the logging settings
  - Compromize need and log amount
  - Keep an eye on storage
- Evaluate pre-defined logging settings
- Create set scripts for those settings
- Consider analytics with pgbadger
- Use the reports for developement feedback

MY STRATEGY
LOWEST HANGING FRUITS

- Cluster tuning vs SQL tuning?
- Prevention check for not reported long running queries
- Possibility for SQL rewrite or index changes?
- Check application behaviour
- Evaluate object statistics and collection strategy

Final Words

FINAL WORDS
RECOMMENDATIONS

- Make your life as easy as possible
- Don‘t over-measure
- Don‘t focus on one potential error cause
- Take the lowest hanging fruits first
- PostgreSQL-typical: Use the potential of extensions!

trivadis

Part of **Accenture**

trivadis
Part of Accenture