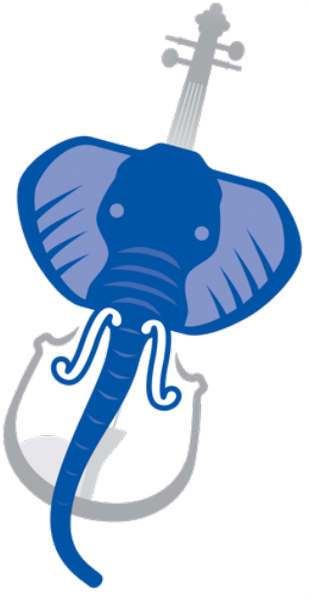
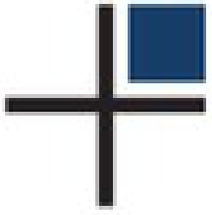
2ndQuadrant
Professional PostgreSQL



The Importance of Query Profiling

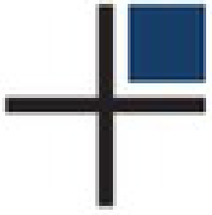
Gianni Ciolli

PGconf.EU
Vienna, 27-30 October 2015



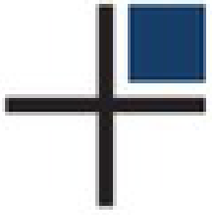
What is Query Profiling?

- Quick (and *tranchant*):
- break down execution time by query



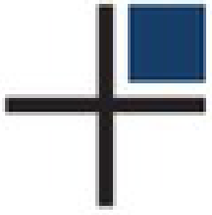
Why Query Profiling?

- Allocate time wisely
- Optimise a query that takes 10 hours per week
- Don’t touch a query that takes 10 seconds per week



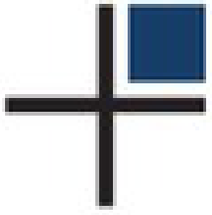
Real-time v Log Analysis

- Profiling data can be:
  - Collected real-time in catalog tables
  - Written to log for later analysis



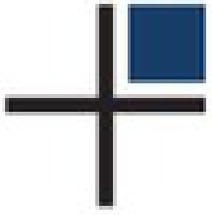
Real-time

- Real-time examples:
  - `auto_explain`
  - `pg_stat_statements`
    - `pg_stat_plans`
  - `pg_stat_*_functions`



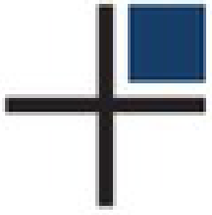
Log Analysis

- Log analysis examples:
  - grep, awk
    - don’t do it!
  - custom script
    - don’t do it either!
  - pgBadger
    - This is OK



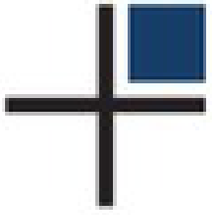
Log Analysis Issues

- Overhead of parsing
  - X% of query execution
  - Y bytes/second
- Time lag
  - Weekly/daily parsing
  - Large amount of data (see above!)



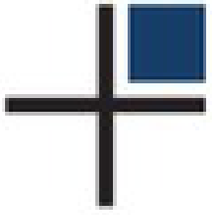
Estimating Overhead

- Intensive test
- Composed by two parts
  - Part 1: 30s
  - Part 2: 30s
- Four variations
  - With or Without `pg_stat_statements`
  - With or Without `auto_explain`



Intensive test

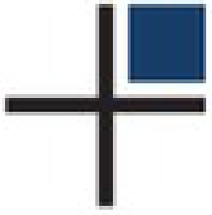
	Part 1	Part 2
Duration	30 s	30 s
Composition		
- Default pgbench's tx	90%	1%
- As in pgbench -N	9%	9%
- As in pgbench -S	1%	90%



Intensive test

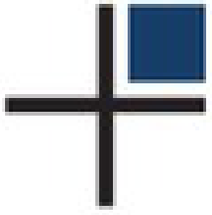
- pgBadger cost

	clean	pss	ae	ae + pss
Queries	1.28 M	1.25 M	1.00 M	0.98 M
Log size	406 MB	395 MB	973 MB	958 MB
Log size/query)	330 B	330 B	1020 B	1020 B
Analysis time	165 s	165 s	215 s	205 s
Factor	275%	275%	358%	342%



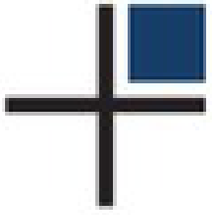
SQL is a declarative language

- The user:
  - specifies the goal
  - that's it!



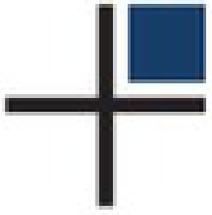
SQL is a declarative language

- PostgreSQL:
  - considers many plans for the same goal
  - estimates cost for each plan, using:
    - *statistics* (1)
    - *settings*
    - *metadata* (the "schema")
  - executes plan with smallest estimated cost
  - optional: records statistics (2) on the plan being executed
    - (1) and (2) are not the same thing!



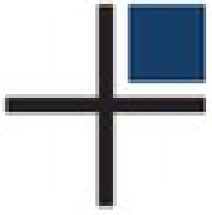
Planner Inputs

- Database statistics
  - Distribution of data
  - Table and index size
- Planner settings
  - Estimated cost of each operation
  - Resource limits
  - Other limitations e.g. `enable_*`
- Metadata
  - Availability of an index
  - Database constraints



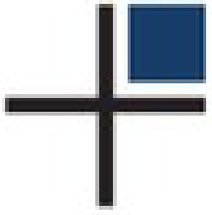
Plan Instability

- Plan Instability
  - Sounds like a bug/problem. . .
- Adaptive Planning
  - Sounds like a great feature. . .
- The same concept described in two ways!
- Other examples:
  - "With Greater Power Comes Greater Responsibility"
  - "Every Crisis Is Also An Opportunity"
  - (etc.)



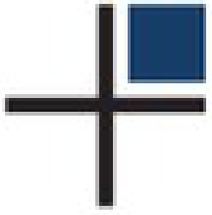
Builtin v External

- Builtin v External
  - something missing. . .



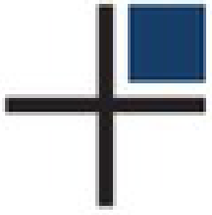
Builtin v Extension v External

- Builtin v Extension v External (better!)
- PostgreSQL has *hooks*
- Implicitely documented "by example" in contrib
- Extensions
  - *plug in* on a live server
  - Cannot "unplug" :-(
- Extension can use hooks to inspect the Query Execution Workflow



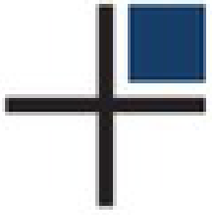
What is a Plan?

- Tree structure
- A leave represents a *data source*
- A node (inner) represents a *data transformation*
- The root represent the *query output*



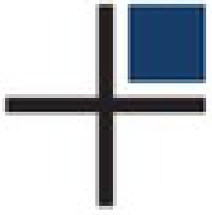
Plan Node Information

- Each plan node can carry:
  - Number of rows
    - Both *expected* and *actual*
  - *Expected* cost
  - *Actual* timing
  - Other information
    - Both *generic* and *specific*



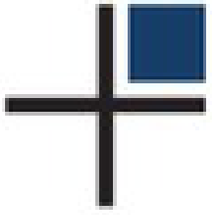
Plan Node Information

```
Hash Join
    (cost=2.36..14142.36 rows=1000000 width=813)
    (actual time=0.061..242.253 rows=1000000 loops=1)
  Hash Cond: (a.bid = b.bid)
  ->  Seq Scan on pgbench_accounts a
        (cost=0.00..2640.00 rows=100000 width=97)
        (actual time=0.014..8.392 rows=100000 loops=1)
  ->  Hash
        (cost=2.24..2.24 rows=10 width=716)
        (actual time=0.033..0.033 rows=10 loops=1)
      Buckets: 1024  Batches: 1  Memory Usage: 1kB
      ->  Nested Loop
            (cost=0.00..2.24 rows=10 width=716)
            (actual time=0.014..0.024 rows=10 loops=1)
          Join Filter: (b.bid = t.bid)
          ->  Seq Scan on pgbench_branches b
                (cost=0.00..1.01 rows=1 width=364)
                (actual time=0.002..0.002 rows=1 loops=1)
          ->  Seq Scan on pgbench_tellers t
                (cost=0.00..1.10 rows=10 width=352)
                (actual time=0.002..0.004 rows=10 loops=1)
```



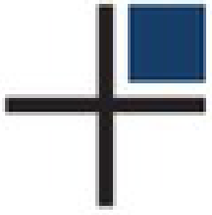
auto_explain

- Contrib module
- Print EXPLAIN of each query
  - (slower than a given threshold)
- Serializing to text file
- Can be ~~hacked~~ augmented to insert plans in a dedicated table
  - JSON plans, JSONB column
  - Beware of the overhead...



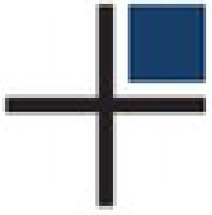
pg_stat_statements

- Contrib module
- Collect execution statistics for each query
- Statistics available in a catalogue view



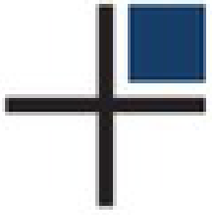
pg_stat_plans

- Hosted on `github`
- Variant of `pg_stat_statements`
- Collect execution statistics for each plan
- Statistics available in a catalogue view
- Currently working with 9.2 and 9.3



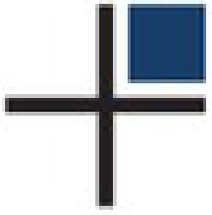
Special case

- Function-based abstraction
- Good for Query Profiling:
  - `track_functions=on`
  - `pg_stat_user_functions`
  - No need to normalise queries
  - Separation between App and Maintenance
  - Still need to investigate Plan Stability
  - Extra metadata is available:
    - queries are anonymous, functions are not



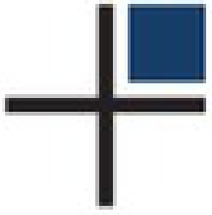
Special case

- Good for other reasons:
  - Sharding-friendly (e.g. PL/Proxy)
  - Portable
  - "Functional" permissions
    - as opposed to "data-centric"
    - Probably closer to the use case. . .



And now. . .

Questions?



And then...

Thank you!

postgresql.eu/events/feedback/pgconfeu2015

Licence