PoWA 3

June, 28 2016 - 5432... Meet us!

DALIBO
L'expertise PostgreSQL

Authors

- Ronan Dunklau
  - DBA @ Dalibo
  - Open-Source: Multicorn...
  - Some PostgreSQL contributions (IMPORT FOREIGN SCHEMA...)
- Julien Rouhaud
  - DBA @ Dalibo
  - Open-Source: HypoPG, OPM...
  - Some PostgreSQL contributions
- But also...

DALIBO
L'expertise PostgreSQL

WHAT IS POWA

[t]

2cm

5cm

PRESENTATION

- pg_stat_statements
- github.com/dalibo/pg_stat_kcache
- github.com/dalibo/pg_qualstats
- github.com/dalibo/powa-archivist
- github.com/dalibo/powa-web

pg_stat_statements

PRESENTATION

- Official PostgreSQL contrib
- Normalized queries
- Cumulative counters (buffers, execution time...), by
  - user
  - database
  - query

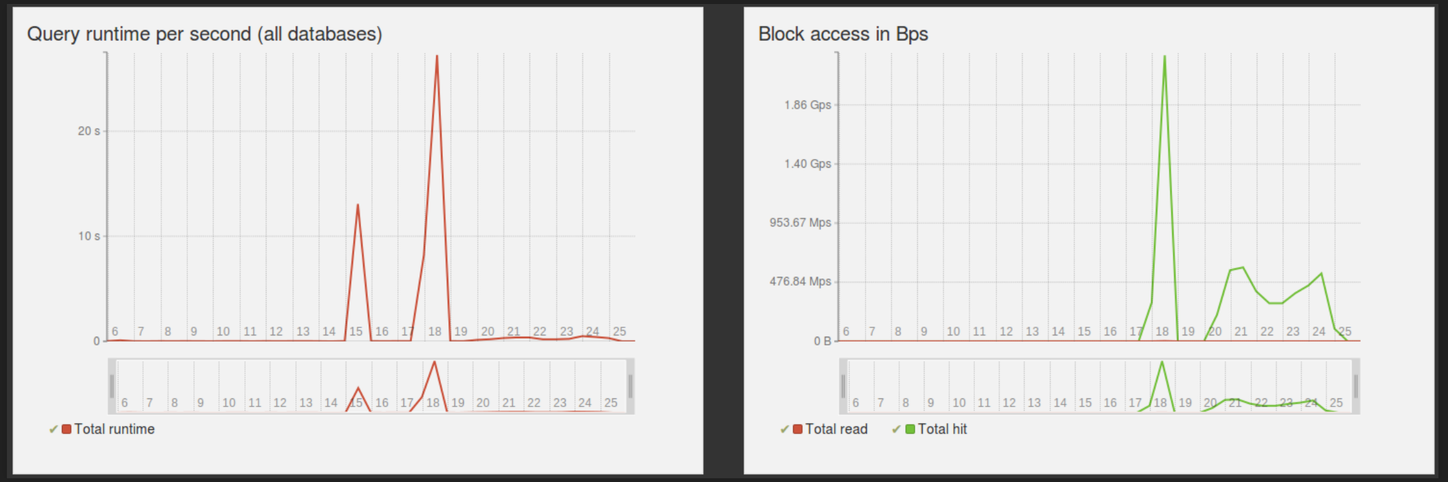
pg_stat_statements

USEFUL INDICATORS

- Number of execution per normalized query
- Average execution time
- Temporary file creation
- Blocks access from or outside **PostgreSQL**’s cache

pg_stat_statements

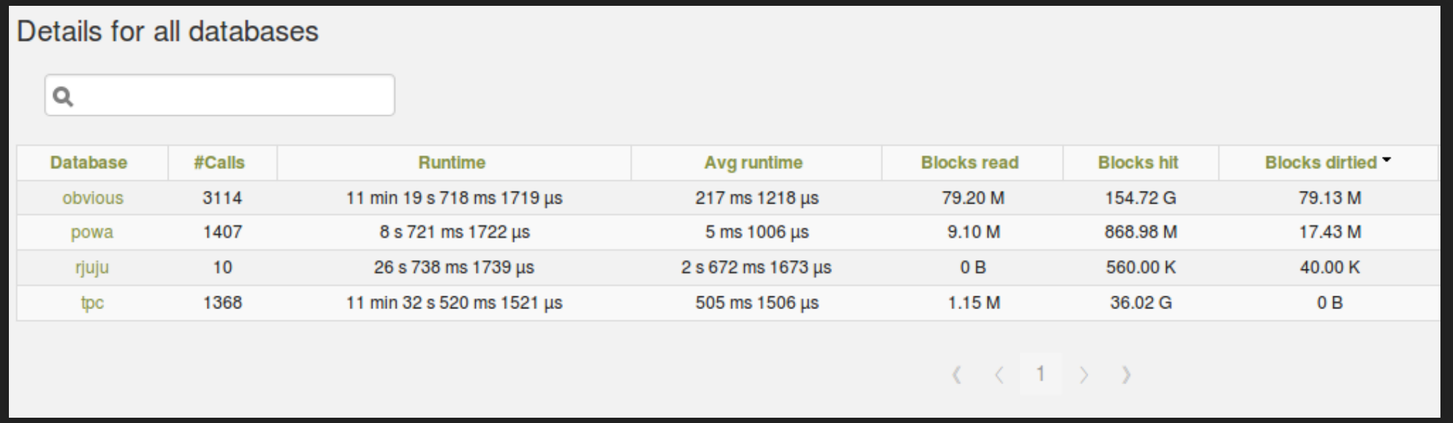
IN ACTION 1



image

pg_stat_statements

IN ACTION 2



image

pg_stat_kcache

PRESENTATION

- Collects system metrics, by normalized queries
  - Physical disk access
  - CPU usage

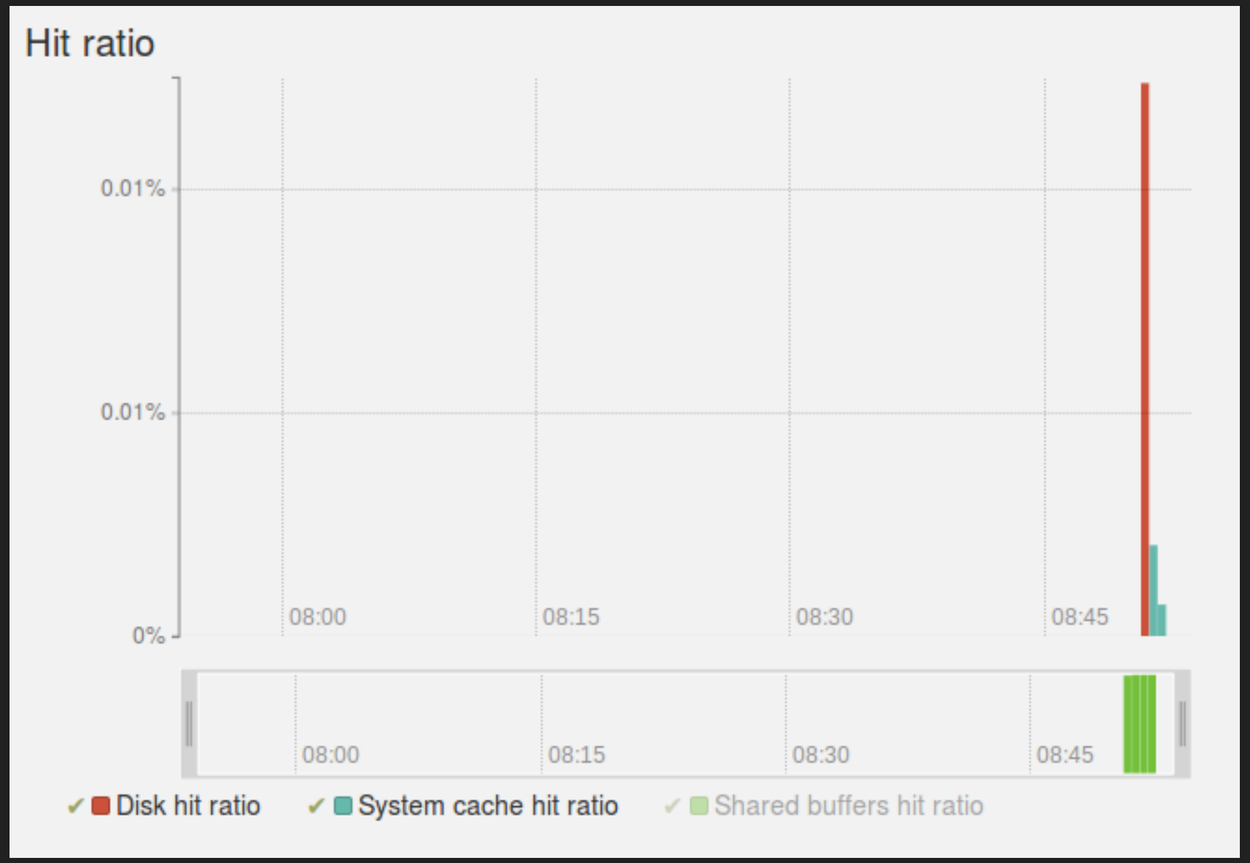
pg_stat_kcache

MEANING...

- “real” hit-ratio (PostgreSQL cache Vs system cache)
- Identify CPU bound queries

pg_stat_kcache

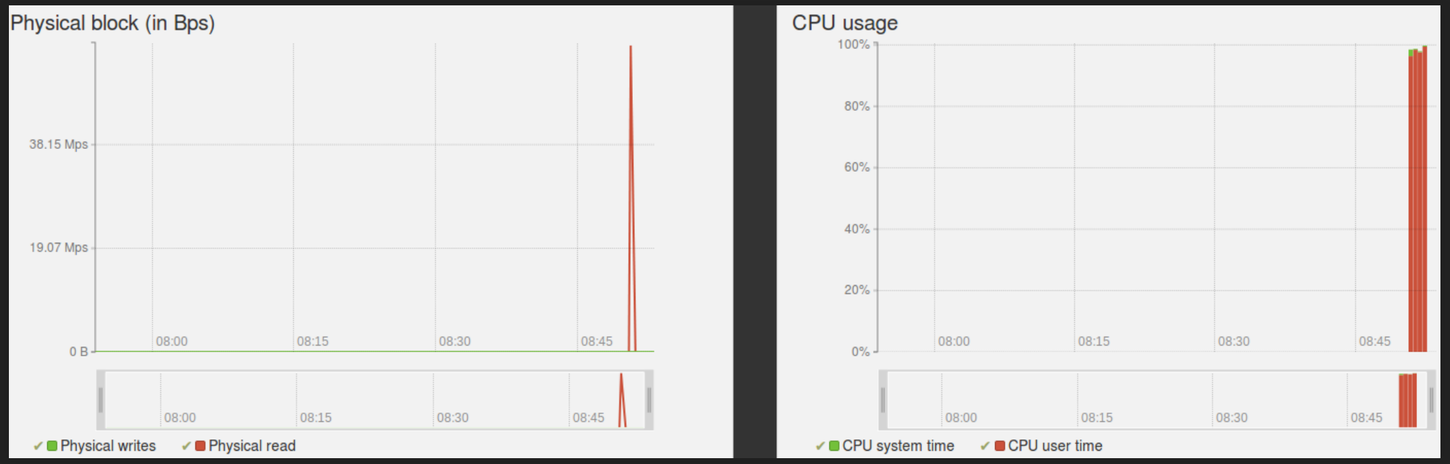
IN ACTION 1



image

pg_stat_kcache

IN ACTION 2



image

pg_qualstats

PRESENTATION

- Predicate analysis
  - WHERE clauses
  - JOIN clauses
- Collects various metrics
  - Selectivity
  - Constants sampling (most executed, most filtering...)
  - Execution count
  - Evalutation type (Index clause or post-scan filtering)

pg_qualstats

IN ACTION 1

```
SELECT
    com.id,
    sum (
        c_l.price ) AS total_price
FROM
    command com
    JOIN command_line c_l ON com.id = c_l.id_command
    JOIN client cli ON cli.id = com.id_client
WHERE
    cli.id = ?
GROUP BY
    com.id;
```

image

pg_qualstats

IN ACTION 2

Predicates used by this query

Predicate	Eval Type	Avg filter_ratio (excluding index)	Execution count (excluding index)
`WHERE command.id_client = ?`		99.99%	129,800,000.00
`WHERE client.id = ?`		0.00%	1,298.00

‹ ‹ 1 › ›

Index suggestion

- Possible indexes for attributes present in **WHERE command.id_client = ?**:

  With access method *btree*

  - **Attribute**

    command.id_client

    Data distribution

    approximately **9837** distinct values

  With access method *brin*

  - **Attribute**

    command.id_client

    Data distribution

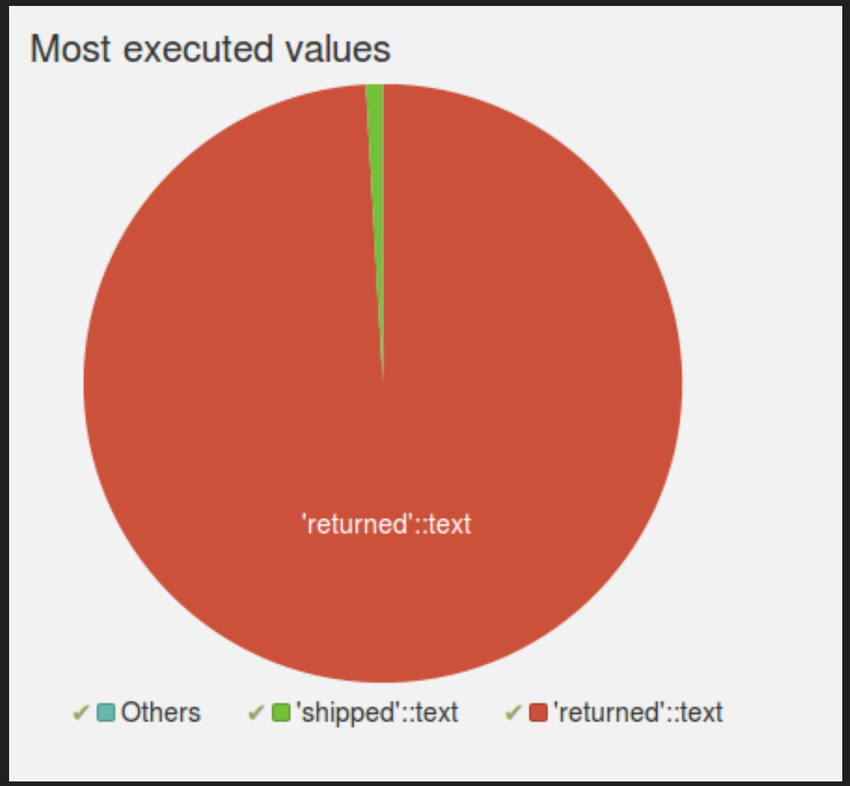
    approximately **9837** distinct values

image

pg_qualstats

DALIBO
L'expertise PostgreSQL

IN ACTION 3



image

pg_qualstats

DALIBO
L'expertise PostgreSQL

IN ACTION 4

Least Filtering values

Executed:

400000 times

Average filter ratio:

0.1%

Example plan:

```
SELECT id,dt FROM command WHERE state = 'shipped'::text;
```

```
Seq Scan on command  (cost=0.00..1986.00 rows=99907 width=12)
  Filter: (state = 'shipped'::text)
```

Most Executed values

Executed:

55000000 times

Average filter ratio:

99.9%

Example plan:

```
SELECT id,dt FROM command WHERE state = 'returned'::text;
```

```
Seq Scan on command  (cost=0.00..1986.00 rows=93 width=12)
  Filter: (state = 'returned'::text)
```

image

powa-archivist

PRESENTATION

- Archive those data sources
- Configurable (retention, frequency...)
- Extensible to other datasources

powa-archivist

WHAT TO GET

- Where / when are the bottlenecks
- For what reason
- How to fix
- Live!

Compatibility

- PostgreSQL **9.4** et later
- PoWA 1 compatible with 9.3, but much more limited

powa-web

PRESENTATION

- Web interface for PoWA
- Manage one or more PoWA instance
- Drill-down analysis

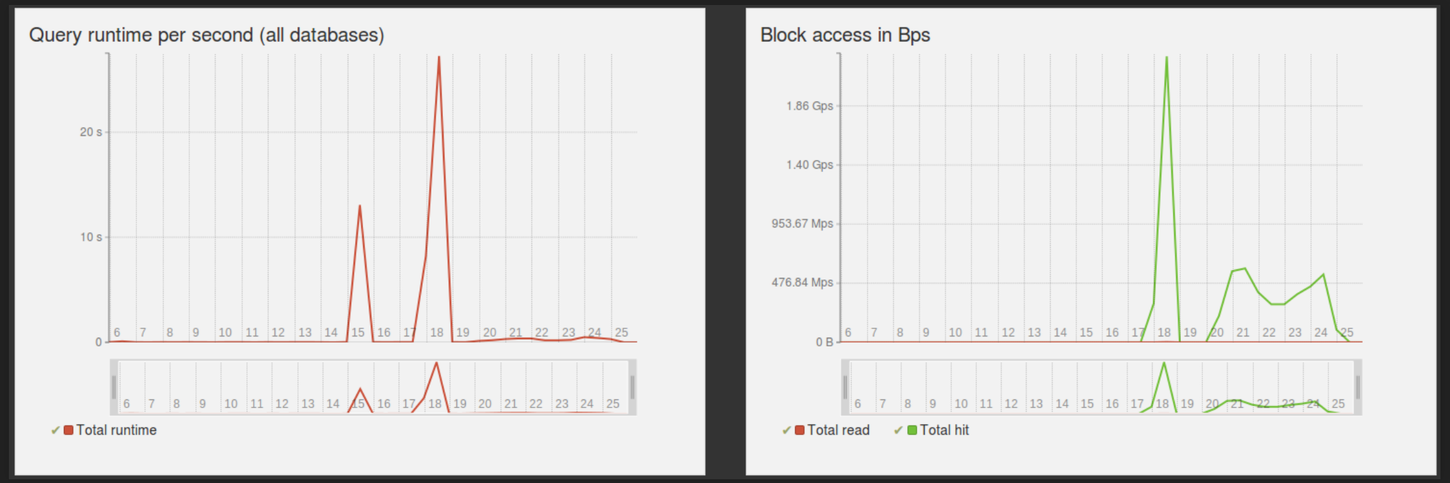
powa-web

USAGE EXAMPLE

- problem: bad performance on parts of an application
- Select an analysis period
- Identify the database

powa-web

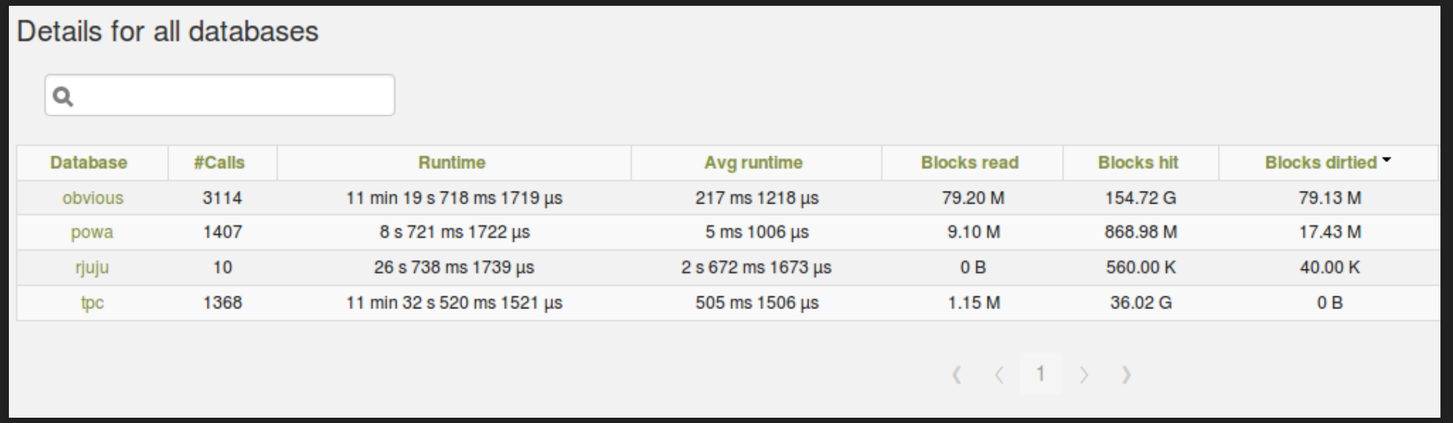
CLUSTER VIEW - 1



image

powa-web

CLUSTER VIEW - 2



image

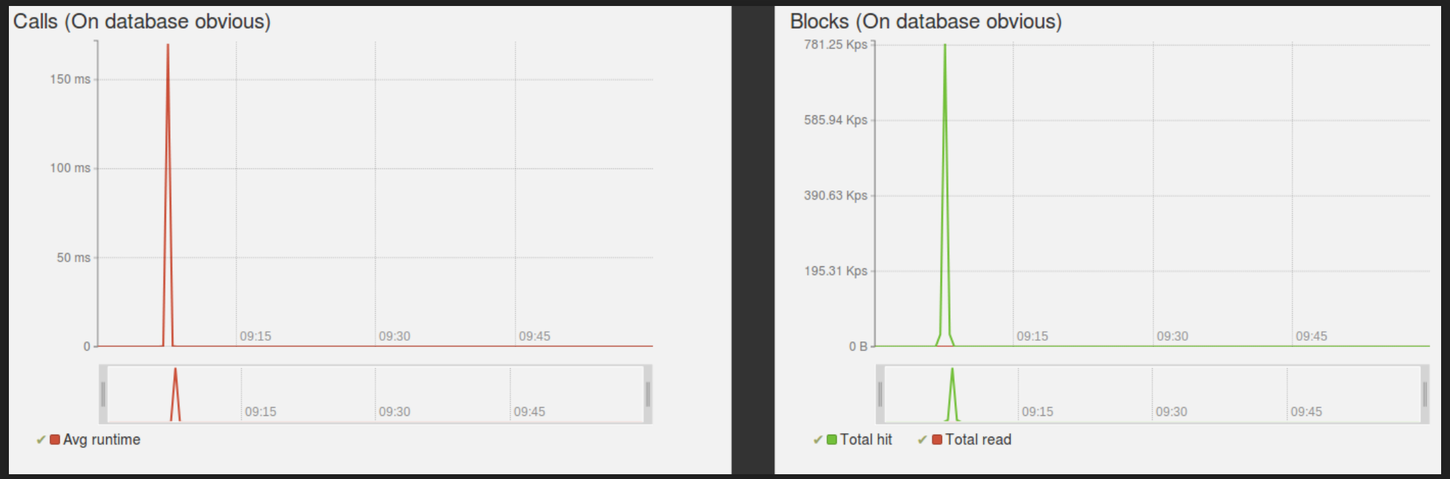
powa-web

DATABASE VIEW

- Problematic database has been identified...
- let’s drill down to the query level!

powa-web

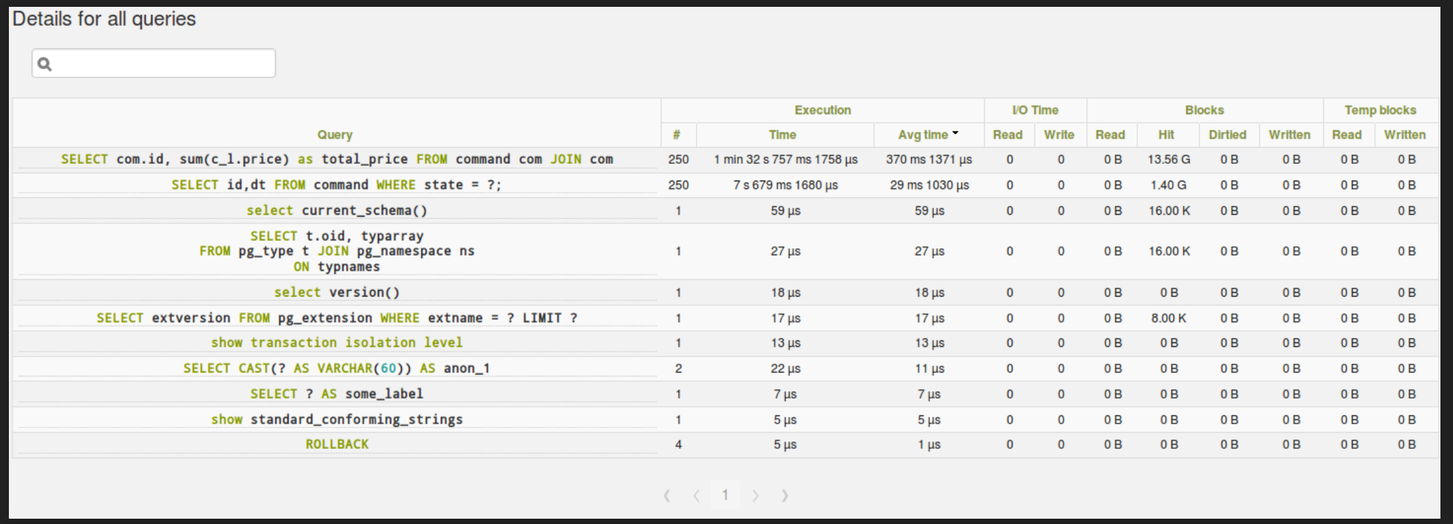
DATABASE VIEW - 1



image

powa-web

DATABASE VIEW - 2



Details for all queries

Query	Execution			I/O Time		Blocks				Temp blocks	
	#	Time	Avg time	Read	Write	Read	Hit	Dirtied	Written	Read	Written
SELECT com.id, sum(c_l.price) as total_price FROM command com JOIN com	250	1 min 32 s 757 ms 1758 µs	370 ms 1371 µs	0	0	0 B	13.56 G	0 B	0 B	0 B	0 B
SELECT id,dt FROM command WHERE state = ?;	250	7 s 679 ms 1680 µs	29 ms 1030 µs	0	0	0 B	1.40 G	0 B	0 B	0 B	0 B
select current_schema()	1	59 µs	59 µs	0	0	0 B	16.00 K	0 B	0 B	0 B	0 B
SELECT t.oid, typarray FROM pg_type t JOIN pg_namespace ns ON typnames	1	27 µs	27 µs	0	0	0 B	16.00 K	0 B	0 B	0 B	0 B
select version()	1	18 µs	18 µs	0	0	0 B	0 B	0 B	0 B	0 B	0 B
SELECT extversion FROM pg_extension WHERE extname = ? LIMIT ?	1	17 µs	17 µs	0	0	0 B	8.00 K	0 B	0 B	0 B	0 B
show transaction isolation level	1	13 µs	13 µs	0	0	0 B	0 B	0 B	0 B	0 B	0 B
SELECT CAST(? AS VARCHAR(60)) AS anon_1	2	22 µs	11 µs	0	0	0 B	0 B	0 B	0 B	0 B	0 B
SELECT ? AS some_label	1	7 µs	7 µs	0	0	0 B	0 B	0 B	0 B	0 B	0 B
show standard_conforming_strings	1	5 µs	5 µs	0	0	0 B	0 B	0 B	0 B	0 B	0 B
ROLLBACK	4	5 µs	1 µs	0	0	0 B	0 B	0 B	0 B	0 B	0 B

1

image

powa-web

DALIBO
L'expertise PostgreSQL

DATABASE VIEW - 3

Query	Execution		
	#	Time	Avg time
`SELECT com.id, sum(c_l.price) as total_price FROM command com JOIN com`	250	1 min 32 s 757 ms 1758 µs	370 ms 1371 µs
`SELECT id,dt FROM command WHERE state = ?;`	250	7 s 679 ms 1680 µs	29 ms 1030 µs
`select current_schema()`	1	59 µs	59 µs
`SELECT t.oid, typarray FROM pg_type t JOIN pg_namespace ns ON typnames`	1	27 µs	27 µs
`select version()`	1	18 µs	18 µs

image

powa-web

DALIBO
L'expertise PostgreSQL

QUERY VIEW

- 2 problematic queries
- Drill down on each of them

[fragile]

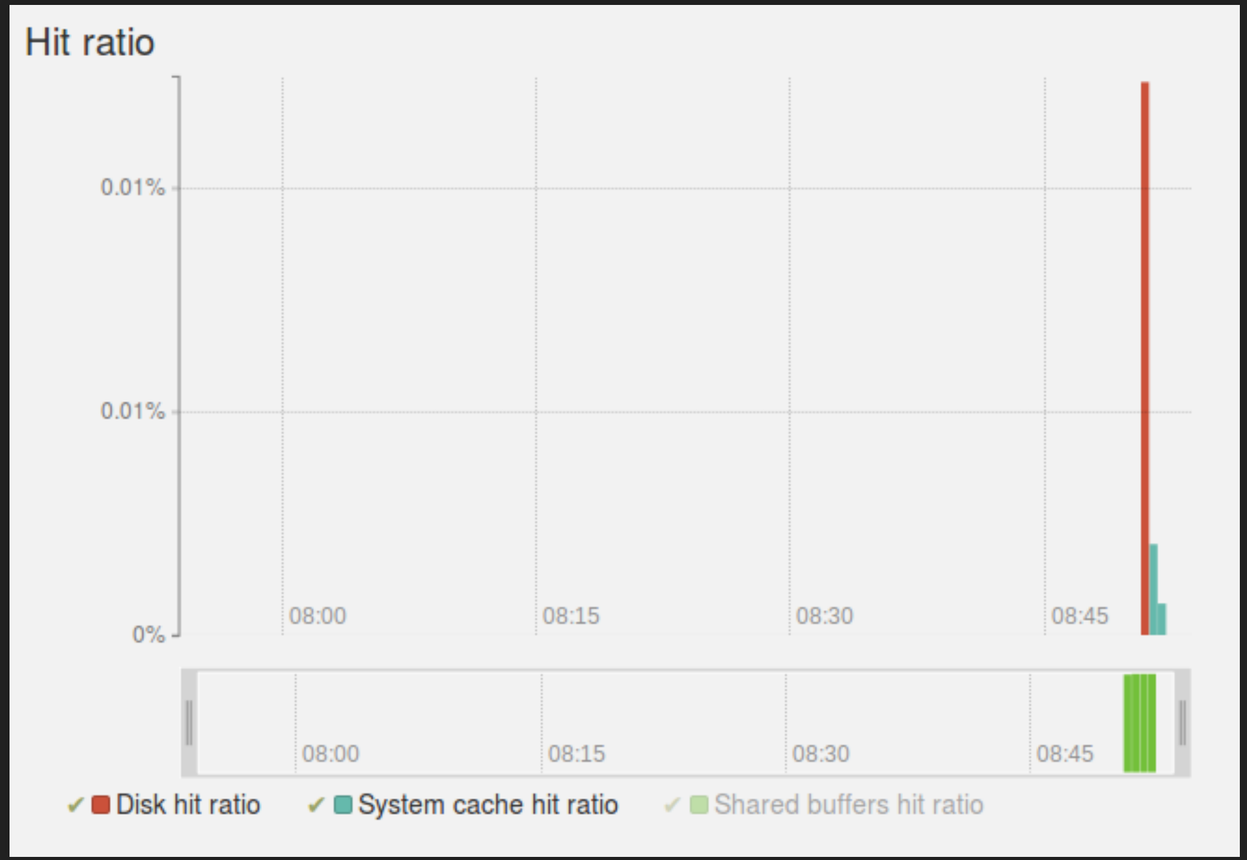
POWA-WEB

FIRST QUERY - SQL

[mathescape, numbersep=5pt, gobble=2, frame=lines, framesep=2mm]sql SELECT com.id, sum(cl.pric) AS total_price FROM command com JOIN commandline cl ON com.id = cl.id_command JOIN client cli ON cli.id = com.id_client WHERE cli.id = ? GROUP BY com.id

powa-web

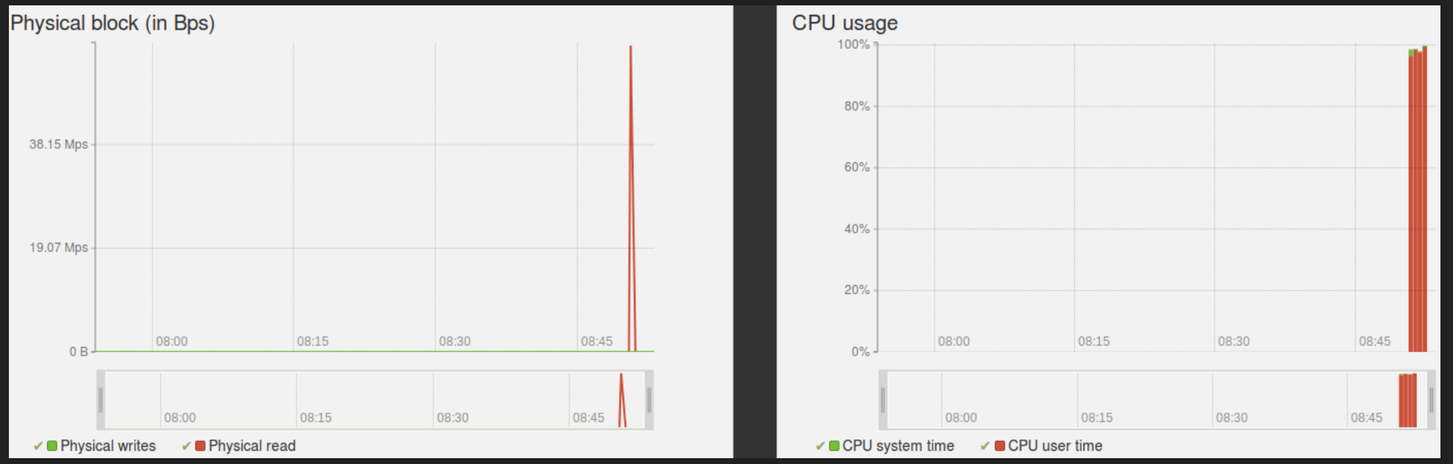
FIRST QUERY - CACHE



image

powa-web

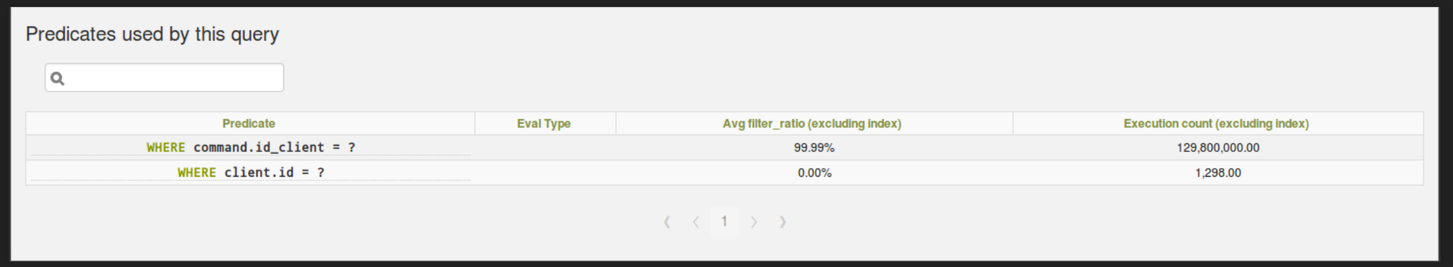
FIRST QUERY - CPU



image

powa-web

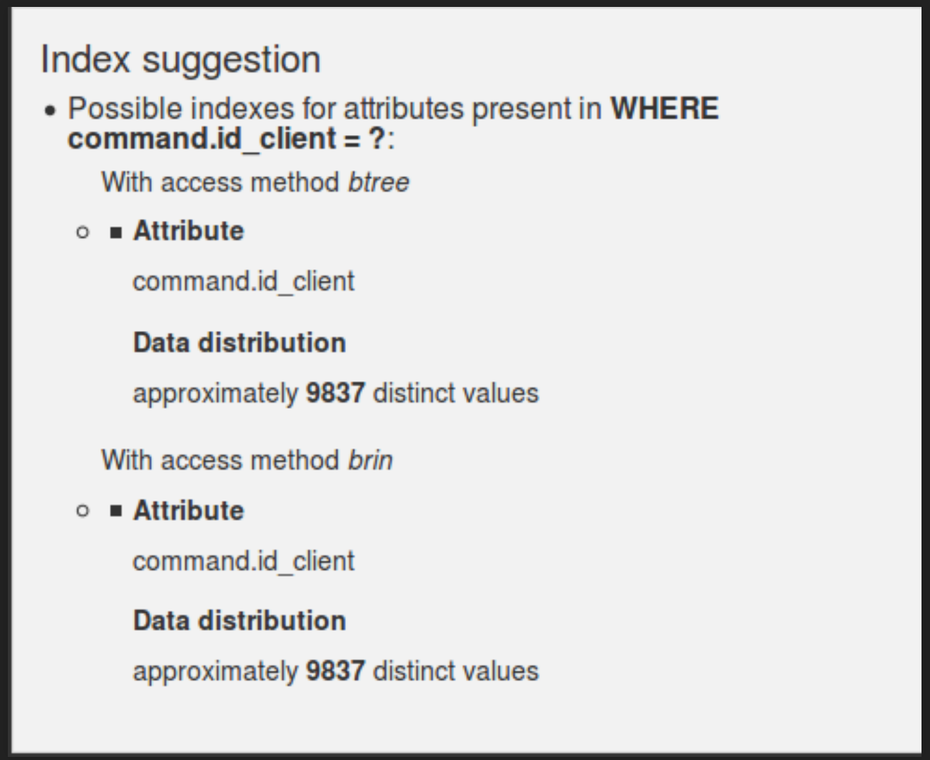
FIRST QUERY - PREDICATES



image

powa-web

FIRST QUERY - INDEX



image

[fragile]powa-web

SECOND QUERY - SQL

[mathescape, numbersep=5pt, gobble=2, frame=lines, framesep=2mm]sql SELECT id, dt FROM command WHERE state = ?

powa-web

SECOND QUERY - EXPLAIN

Least Filtering values

Executed:

300000 times

Average filter ratio:

0.1%

Example plan:

```
SELECT id,dt FROM command WHERE state = 'shipped'::text;
```

```
Seq Scan on command  (cost=0.00..1986.00 rows=99907 width=12)
  Filter: (state = 'shipped'::text)
```

Most Executed values

Executed:

20000000 times

Average filter ratio:

99.9%

Example plan:

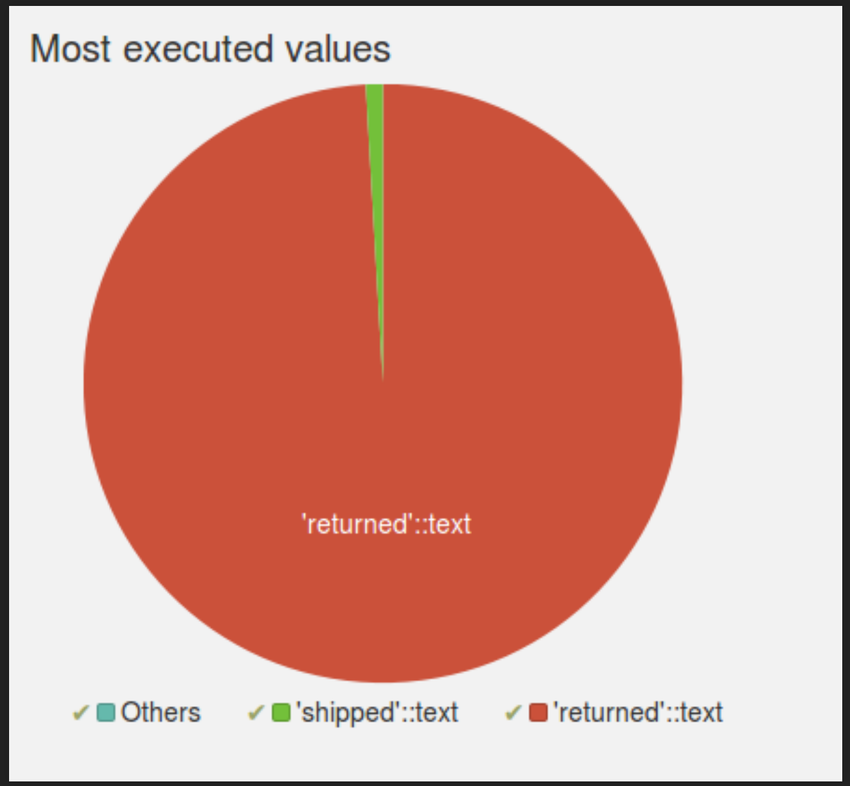
```
SELECT id,dt FROM command WHERE state = 'returned'::text;
```

```
Seq Scan on command  (cost=0.00..1986.00 rows=93 width=12)
  Filter: (state = 'returned'::text)
```

image

powa-web

SECOND QUERY - DISTRIBUTION



image

powa-web

VIDEO

•

powa-web

WHAT'S NEW IN VERSION 3

- github.com/dalibo/HypoPG extension support
- Global index suggestion

HypoPG

PRESENTATION

- Allow for hypothetical indexes creation
- Instant creation, no impact on resources and no lock
- Only used in EXPLAIN statements

[fragile]HypoPG

EXAMPLE

[mathescape, numbersep=5pt, gobble=2, frame=lines, framesep=2mm]sql rjuju=# EXPLAIN SELECT * FROM t1 WHERE id = 3 ; QUERY PLAN ——————————————————————— Seq Scan on t1 (cost=0.00..1693.00 rows=1 width=4) Filter: (id = 3) (2 rows)

[fragile]HypoPG

EXAMPLE

[mathescape, numbersep=5pt, gobble=2, frame=lines, framesep=2mm]sql # SELECT hypopg_create_index('CREATE INDEX ON t1(id)') ; hypopg_create_index ———————————— (77523,<77523>btree_t1_id) (1 row)

rjuju=# EXPLAIN SELECT * FROM t1 WHERE id = 3 ; QUERY PLAN ———————————————————————————— Index Only Scan using <77523>btree_t1_id on t1 (0.04..8.06 rows=1 width=4) Index Cond: (id = 3) (2 rows)

HypoPG

WHAT IS IT USEFUL FOR

- Will PostgreSQL use such an index
- What size can I expect it to be
- How useful can it be

HypoPG

IN ACTION

The following indexes would be used:

```
CREATE INDEX ON "public"."command"(state)
```

EXPLAIN plan **without** suggested indexes:

```
Seq Scan on command  (cost=0.00..1986.00
rows=120 width=12)
  Filter: (state = 'returned'::text)
```

EXPLAIN plan **with** suggested index

```
Index Scan using <28731>btree_command_state
on command  (cost=0.04..12.11 rows=120
width=12)
  Index Cond: (state = 'returned'::text)
```

Query cost gain factor with hypothetical index: **99.39 %**

image

Global optimization

PRESENTATION

- Find the optimal set of index to add
  - Helping every queries
  - Minimum set of indexes
  - Privileging multi-column indexes

Global optimization

ALGORITHM - 1

- Fetch the predicates that need optimization (pg_qualstats)
  - Predicates filtering more than X lines out
  - Predicates filtering more than X% of lines out
  - Predicates used as part of a Seq Scan

Global optimization

ALGORITHM - 2

- Group predicates by supported access methods
  - ***Hint:*** *Think about btree_gist and btree_gin*
- Build a list of predicates “contained” by each predicates
  - WHERE id = ? AND label = ?
  - WHERE id = ?
  - WHERE label = ?
- For each node, attribute a “score” to it (currently, number of columns)

Global optimization

ALGORITHM - 3

- For each node, compute a path containing all included node
- Score it (sum of individual nodes scores)
- Starting with the highest scoring path, generate the index definition for it
- Delete any other path made obsolete by this one
- Loop until no path is left unoptimized

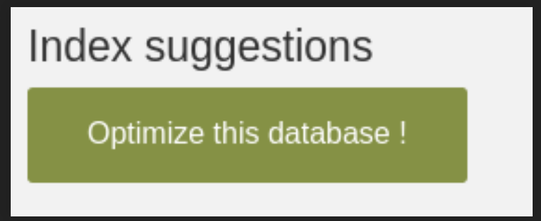
Global optimization

VALIDATION

IN ACTION

Query	#
SELECT pg_sleep(?);	59
SELECT * FROM contacts;	10
SELECT numero_commande, etat_commande FROM commandes WHERE client_id =	10
SELECT * FROM clients cl JOIN contacts co ON co.contact_id = cl.contac	10
SELECT COUNT(*) FROM commandes WHERE date_commande BETWEEN (? || ?)::d	10
SELECT count(*) FROM commandes cmd JOIN lignes_commandes lc ON lc.nume	10
SELECT COUNT(*) FROM pays p JOIN contacts con ON con.code_pays = p.cod	10
SELECT numero_commande, etat_commande FROM commandes WHERE client_id =	10
SELECT count(*) FROM commandes cmd JOIN lignes_commandes lc ON lc.nume	10
SELECT co.nom FROM clients cl JOIN contacts co ON co.contact_id = cl.c	10
SELECT con.nom || ? || code_pays || ? FROM clients cli JOIN contacts c	10
SELECT COUNT(*) FROM pays p JOIN contacts con ON con.code_pays = p.cod	10
SELECT region_id FROM regions WHERE nom_region = ?;	10
SELECT COUNT(*) FROM pieces_fournisseurs WHERE cout_piece >= ?	10
SELECT COUNT(*) FROM commandes WHERE date_commande BETWEEN (? || ?)::d	10
SELECT numero_commande, etat_commande FROM commandes WHERE client_id =	10
SELECT nom FROM contacts c JOIN pays p ON p.code_pays = c.code_pays WH	10
SELECT COUNT(*) FROM commandes WHERE client_id = ? AND priorite_comman	10
SELECT numero_commande, etat_commande FROM commandes WHERE client_id =	10
SELECT COUNT(*) FROM pieces_fournisseurs WHERE quantite_disponible < ?	10

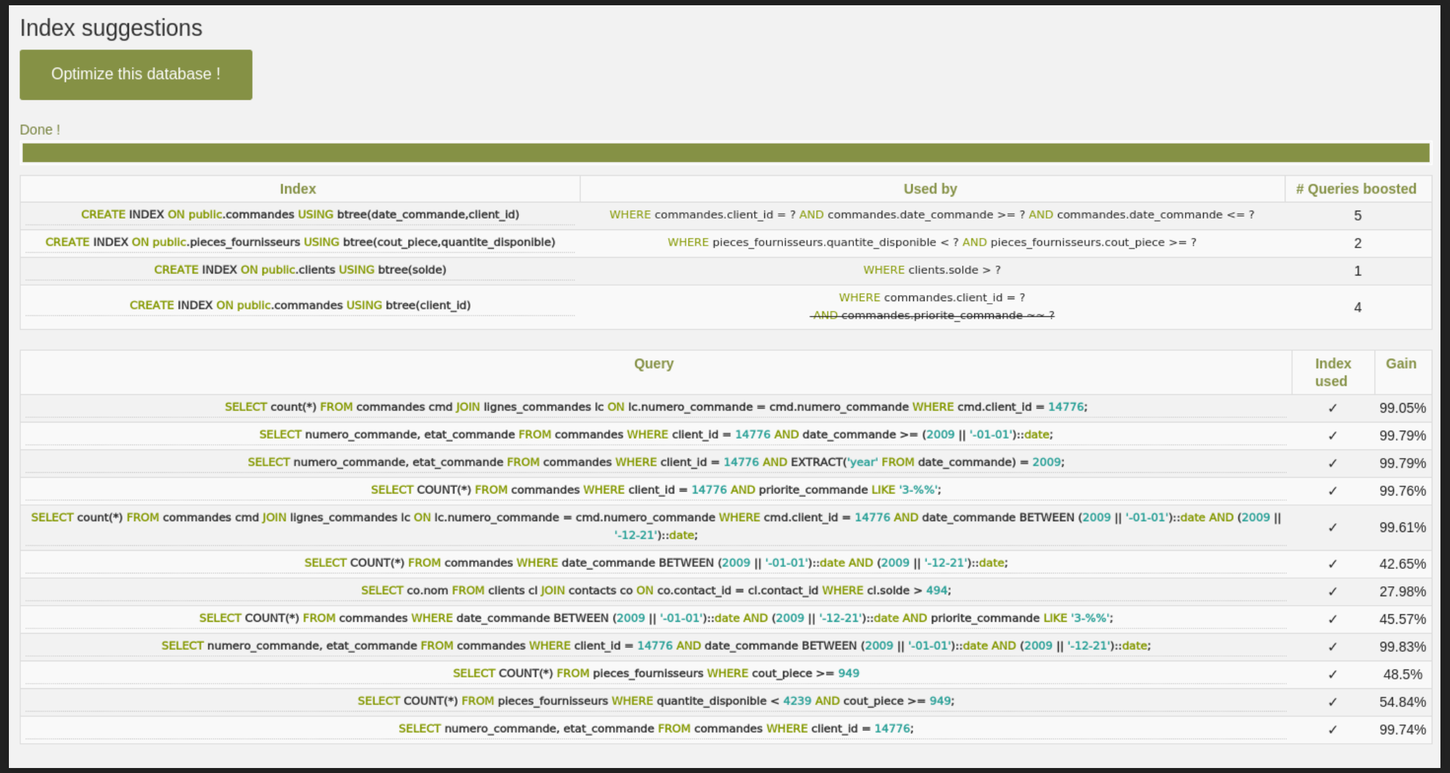
IN ACTION



image

Global optimization

IN ACTION



Index suggestions

Optimize this database !

Done !

Index	Used by	# Queries boosted
CREATE INDEX ON public.commandes USING btree(date_commande,client_id)	WHERE commandes.client_id = ? AND commandes.date_commande >= ? AND commandes.date_commande <= ?	5
CREATE INDEX ON public.pieces_fournisseurs USING btree(cout_piece,quantite_disponible)	WHERE pieces_fournisseurs.quantite_disponible < ? AND pieces_fournisseurs.cout_piece >= ?	2
CREATE INDEX ON public.clients USING btree(solde)	WHERE clients.solde > ?	1
CREATE INDEX ON public.commandes USING btree(client_id)	WHERE commandes.client_id = ? ~~AND commandes.priorite_commande ~~ ?	4

Query	Index used	Gain
SELECT count(*) FROM commandes cmd JOIN lignes_commandes lc ON lc.numero_commande = cmd.numero_commande WHERE cmd.client_id = 14776;	✓	99.05%
SELECT numero_commande, etat_commande FROM commandes WHERE client_id = 14776 AND date_commande >= (2009 || '-01-01')::date;	✓	99.79%
SELECT numero_commande, etat_commande FROM commandes WHERE client_id = 14776 AND EXTRACT('year' FROM date_commande) = 2009;	✓	99.79%
SELECT COUNT(*) FROM commandes WHERE client_id = 14776 AND priorite_commande LIKE '3-%%';	✓	99.76%
SELECT count(*) FROM commandes cmd JOIN lignes_commandes lc ON lc.numero_commande = cmd.numero_commande WHERE cmd.client_id = 14776 AND date_commande BETWEEN (2009 || '-01-01')::date AND (2009 || '-12-21')::date;	✓	99.61%
SELECT COUNT(*) FROM commandes WHERE date_commande BETWEEN (2009 || '-01-01')::date AND (2009 || '-12-21')::date;	✓	42.65%
SELECT co.nom FROM clients cl JOIN contacts co ON co.contact_id = cl.contact_id WHERE cl.solde > 494;	✓	27.98%
SELECT COUNT(*) FROM commandes WHERE date_commande BETWEEN (2009 || '-01-01')::date AND (2009 || '-12-21')::date AND priorite_commande LIKE '3-%%';	✓	45.57%
SELECT numero_commande, etat_commande FROM commandes WHERE client_id = 14776 AND date_commande BETWEEN (2009 || '-01-01')::date AND (2009 || '-12-21')::date;	✓	99.83%
SELECT COUNT(*) FROM pieces_fournisseurs WHERE cout_piece >= 949	✓	48.5%
SELECT COUNT(*) FROM pieces_fournisseurs WHERE quantite_disponible < 4239 AND cout_piece >= 949;	✓	54.84%
SELECT numero_commande, etat_commande FROM commandes WHERE client_id = 14776;	✓	99.74%

image

Global optimization

DALIBO
L'expertise PostgreSQL

IN ACTION

Index	Used by	# Queries boosted
`CREATE INDEX ON public.pieces_fournisseurs USING btree(cout_piece,quantite_disponible)`	`WHERE pieces_fournisseurs.quantite_disponible < ? AND pieces_fournisseurs.cout_piece >= ?` `WHERE pieces_fournisseurs.cout_piece >= ?`	2

`CREATE INDEX ON public.commandes USING btree(date_commande,client_id)`	`WHERE commandes.client_id = ? AND commandes.date_commande >= ? AND commandes.date_commande <= ?`
	`WHERE commandes.date_commande <= ? AND commandes.date_commande >= ?`
	`WHERE commandes.client_id = ?`
		5

`CREATE INDEX ON public.clients USING btree(solde)`	`WHERE clients.solde > ?`	2
`CREATE INDEX ON public.commandes USING btree(client_id)`	`WHERE commandes.client_id = ?`	4

image

Global optimization

IN ACTION

Query	Index used	Gain
`SELECT co.nom FROM clients cl JOIN contacts co ON co.contact_id = cl.contact_id WHERE cl.solde > 444;`	✓	15.19%
`SELECT COUNT(*) FROM commandes WHERE date_commande BETWEEN (2011 || '-01-01')::date AND (2011 || '-12-21')::date;`	✓	84.14%
`SELECT numero_commande, etat_commande FROM commandes WHERE client_id = 14608 AND EXTRACT('year' FROM date_commande) = 2011;`	✓	99.83%
`SELECT COUNT(*) FROM pieces_fournisseurs WHERE quantite_disponible < 7126 AND cout_piece >= 956;`	✓	93.98%
`SELECT COUNT(*) FROM commandes WHERE date_commande BETWEEN (2011 || '-01-01')::date AND (2011 || '-12-21')::date AND priorite_commande LIKE '3-%%';`	✓	45.69%
`SELECT con.nom || ' (' || code_pays || ' )' FROM clients cli JOIN contacts con ON con.contact_id = cli.contact_id WHERE solde > 444;`	✓	14.54%
`SELECT numero_commande, etat_commande FROM commandes WHERE client_id = 14608 AND date_commande >= (2011 || '-01-01')::date;`	✓	99.83%
`SELECT numero_commande, etat_commande FROM commandes WHERE client_id = 14608 AND date_commande BETWEEN (2011 || '-01-01')::date AND (2011 || '-12-21')::date;`	✓	99.86%
`SELECT count(*) FROM commandes cmd JOIN lignes_commandes lc ON lc.numero_commande = cmd.numero_commande WHERE cmd.client_id = 14608 AND date_commande BETWEEN (2011 || '-01-01')::date AND (2011 || '-12-21')::date;`	✓	27.09%
`SELECT count(*) FROM commandes cmd JOIN lignes_commandes lc ON lc.numero_commande = cmd.numero_commande WHERE cmd.client_id = 14608;`	✓	17.35%
`SELECT numero_commande, etat_commande FROM commandes WHERE client_id = 14608;`	✓	99.78%
`SELECT COUNT(*) FROM pieces_fournisseurs WHERE cout_piece >= 956`	✓	93.25%
`SELECT COUNT(*) FROM commandes WHERE client_id = 14608 AND priorite_commande LIKE '3-%%';`	✓	99.8%

image

Global optimization

DALIBO
L'expertise PostgreSQL

IN ACTION

- vidéo

What’s next

FUTURE ENHANCEMENTS

- Find correlations, and suggest them once correlated statistics are available
  - WHERE cityname = ? AND zipcode = ? (10 rows avg)
  - WHERE cityname = ? (10 rows avg)
  - WHERE zipcode = ? (10 rows avg)
  - It means that cityname and zipcode are probably correlated
- Collect statistics on table to take DML workload into account
- Suggest partial indexes based on most-often used values

- powa-archivist
  - dalibo.github.io/powa (website)
  - github.com/dalibo/powa-archivist (repository)
- powa-web
  - github.com/dalibo/powa-web (repository)
  - demo-powa.dalibo.com (demo)
- pg_qualstats
  - github.com/dalibo/pg_qualstats (repository)
  - article on rdunklau.github.io

- contact@dalibo.com
- powa@dalibo.com
- powa.readthedocs.org