

Que faire de pg_stat_monitor ?

Meetup PostgreSQL Lille

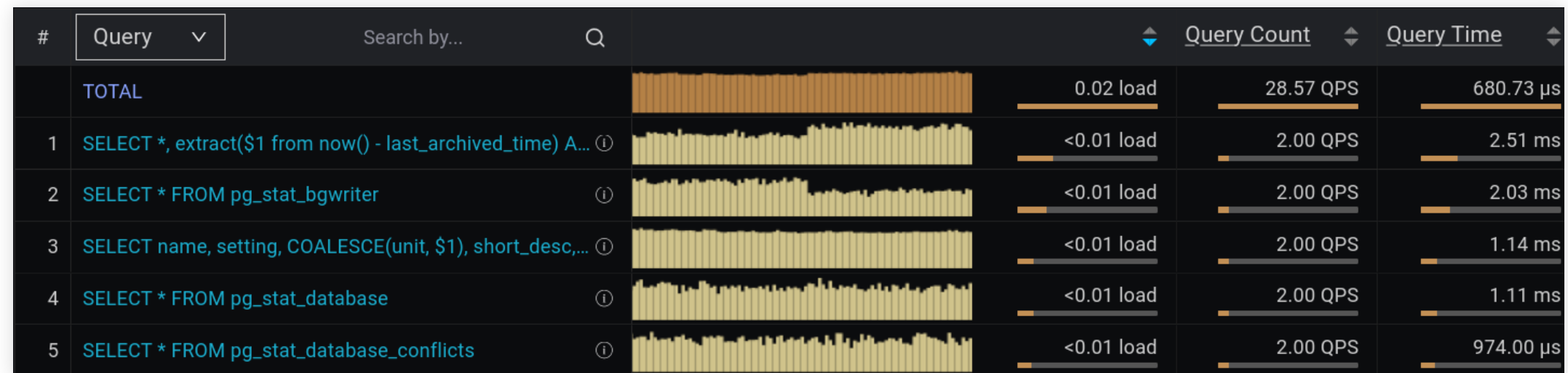
Florent Jardin

(1^{er} juin 2022)

ANNONCE DE PERCONA

`pg_stat_monitor` est disponible en version 1.0

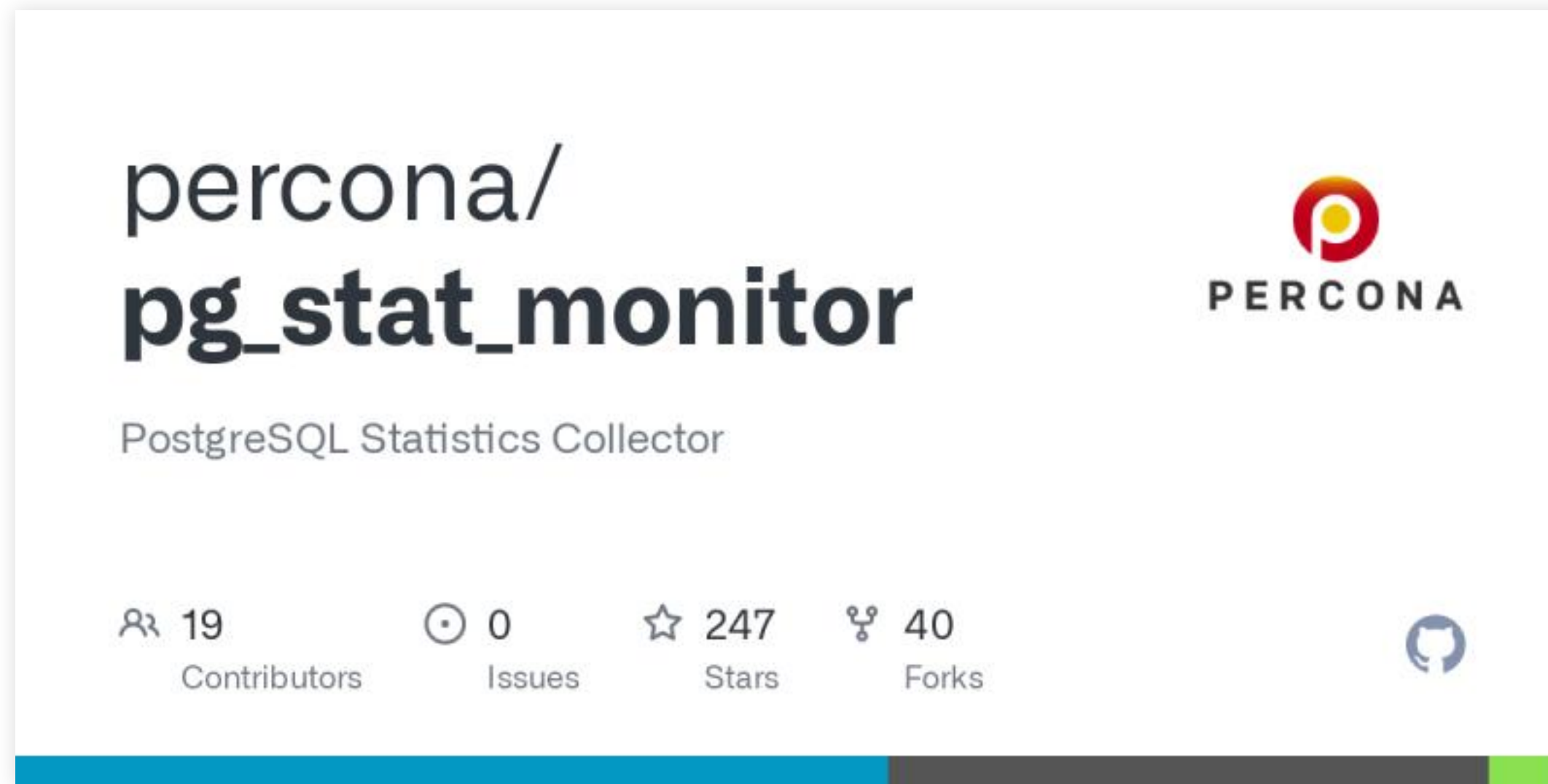
- Fork de `pg_stat_statements` (et `auto_explain`)
- Composant de supervision pour la solution PMM



#	Query	Search by...	Q	Query Count	Query Time	
	TOTAL			0.02 load	28.57 QPS	680.73 µs
1	SELECT *, extract(\$1 from now() - last_archived_time) A...			<0.01 load	2.00 QPS	2.51 ms
2	SELECT * FROM pg_stat_bgwriter			<0.01 load	2.00 QPS	2.03 ms
3	SELECT name, setting, COALESCE(unit, \$1), short_desc,...			<0.01 load	2.00 QPS	1.14 ms
4	SELECT * FROM pg_stat_database			<0.01 load	2.00 QPS	1.11 ms
5	SELECT * FROM pg_stat_database_conflicts			<0.01 load	2.00 QPS	974.00 µs

NOUVELLES FONCTIONNALITÉS

(par rapport à `pg_stat_statements` ...)



percona/
pg_stat_monitor

PostgreSQL Statistics Collector

PERCONA

19 Contributors 0 Issues 247 Stars 40 Forks

REGROUPEMENT DES REQUÊTES EN *TIME SERIES BUCKETS*

- Par défaut, un *bucket* toutes les 60 secondes
 - `pg_stat_monitor.pgsm_max_buckets` (max: 10)
 - `pg_stat_monitor.pgsm_bucket_time` (min: 1sec)

bucket	bucket_start_time	query	calls	mean_exec_time
4	2022-05-11 16:44:00	SELECT abalance FROM pgbench_accounts WHERE aid = \$1	55628	0.0105
5	2022-05-11 16:45:00	SELECT abalance FROM pgbench_accounts WHERE aid = \$1	93491	0.0082
6	2022-05-11 16:46:00	SELECT abalance FROM pgbench_accounts WHERE aid = \$1	87153	0.0091
7	2022-05-11 16:47:00	SELECT abalance FROM pgbench_accounts WHERE aid = \$1	94469	0.0081
8	2022-05-11 16:48:00	SELECT abalance FROM pgbench_accounts WHERE aid = \$1	47375	0.0081

(5 rows)

RELATIONS DE LA REQUÊTES

- Champ `relations`
 - Liste les tables rattachées aux requêtes
 - Parcours la définition des vues

```
-[ RECORD 1 ]-----  
query      | SELECT * FROM pgbench_abalance_view LIMIT $1  
relations  | {public.pgbench_abalance_view*,public.pgbench_accounts,public.pgbench_branches}  
-[ RECORD 2 ]-----  
query      | SELECT * FROM pgbench_accounts JOIN pgbench_branches USING (bid)  
relations  | {public.pgbench_accounts,public.pgbench_branches}  
-[ RECORD 3 ]-----  
query      | SELECT * FROM pgbench_tellers JOIN pgbench_branches USING (bid)  
relations  | {public.pgbench_tellers,public.pgbench_branches}
```

TYPES DES REQUÊTES

- Catégorise les requêtes selon leur type
 - `SELECT` , `INSERT` , `UPDATE` , `DELETE`
 - `(empty)` , `UTILITY` , `NOTHING`
- Champs `cmd_type` et `cmd_type_text`
 - Fonction `get_cmd_type(integer)`

```
cmd_type_text | calls | total_exec_time | rows_retrieved
-----+-----+-----+-----
              | 114553 | 234.65 | 100000
INSERT        | 57277 | 414.03 | 57277
SELECT        | 57270 | 487.75 | 57290
UPDATE        | 171798 | 3242.97 | 171798
(4 rows)
```

REQUÊTES EN ERREUR

- Capte les requêtes en erreur
- Champs `state` , `elevel` , `sqlcode` , `message`

state	count	elevel	sqlcode	message
ACTIVE	1	0		
FINISHED	30	0		
FINISHED WITH ERROR	1	21	22012	division by zero
FINISHED WITH ERROR	1	21	42703	column "cid" does not exist

(4 rows)

PLANS D'ÉXECUTION

- Champs `planid` et `query_plan`
 - Affecte les performances de l'instance
 - `pg_stat_monitor.pgsm_enable_query_plan` (`no`)
- Équivalent de `auto_explain` mais en mémoire
 - pas d'options `EXPLAIN` supplémentaires

CONSOMMATION CPU

- Champs `cpu_user_time` et `cpu_sys_time`
 - Consommation CPU du tracking de requêtes
 - S'appuient sur la fonction `getrusage()`
 - Décorrélés de la valeur `total_exec_time`

bucket	query	calls	total_exec_time	cpu_user_time	cpu_sys_time	cpu_sys_ratio
4	UPDATE pgbench_tellers ...	55473	858.4205	1116.26	280.96	0.20
5	UPDATE pgbench_tellers ...	54042	858.6594	1113.96	283.03	0.20
6	UPDATE pgbench_tellers ...	56046	853.4157	1098.79	292.26	0.21
7	UPDATE pgbench_tellers ...	53425	858.5944	1118.82	284.16	0.20
8	UPDATE pgbench_tellers ...	53493	861.9183	1124.73	285.60	0.20

(5 rows)

MÉTADONNÉES DE REQUÊTE

- Spécification « Sqlcommenter » de Google
- Extrait le bloc de commentaire
 - et maintient le `queryid` intact

```
application_name |      queryid      |      comments
-----+-----+-----
pgbench          | 28DB385168F3A689 |
psql             | 28DB385168F3A689 | /* writer='florent' */
(2 rows)
```

REQUÊTES DÉNORMALISÉES

- Désactiver la normalisation des requêtes
 - Afficher les valeurs réelles
 - ... Seule la première occurrence est tracée
- Facilite l'analyse des performances d'une requête
 - `pg_stat_monitor.pgsm_normalized_query`

```
bucket | bucket_start_time | query | calls
-----+-----+-----+-----
4 | 2022-05-11 17:04:00 | INSERT INTO pgbench_hist ... VALUES (3, 1, 36263, 3963, CURRENT_TIMESTAMP) | 68033
7 | 2022-05-11 17:07:00 | INSERT INTO pgbench_hist ... VALUES (3, 1, 36263, 3963, CURRENT_TIMESTAMP) | 102925
8 | 2022-05-11 17:08:00 | INSERT INTO pgbench_hist ... VALUES (3, 1, 36263, 3963, CURRENT_TIMESTAMP) | 102921
9 | 2022-05-11 17:09:00 | INSERT INTO pgbench_hist ... VALUES (3, 1, 36263, 3963, CURRENT_TIMESTAMP) | 107886
0 | 2022-05-11 17:10:00 | INSERT INTO pgbench_hist ... VALUES (3, 1, 36263, 3963, CURRENT_TIMESTAMP) | 91386
1 | 2022-05-11 17:11:00 | INSERT INTO pgbench_hist ... VALUES (3, 1, 36263, 3963, CURRENT_TIMESTAMP) | 28927
(6 rows)
```

DIFFÉRENCES MINEURES

- `queryid` de type TEXT au lieu de BIGINT
- `userid` de type REGROLE au lieu de OID
- `datname` de type NAME au lieu de `dboid` de type OID
- `bucket_start_time` de type TEXT au lieu de TIMESTAMPTZ dans la documentation
- `rows_retrieved` au lieu de `rows`
- Colonnes inédites `application_name`, `client_ip`

LIMITES ACTUELLES

- Cohabitation difficile entre `pgss` et `pgsm`
 - `pgss` doit être chargé avant `pgsm`
 - En version 14, `compute_query_id = true`
- Les statistiques ne sont pas conservées après un redémarrage
- Un *bucket* n'est pas limité en nombre de requêtes distinctes
 - `pg_stat_monitor.pgsm_max` est exprimé en *MB* et non en quantité de requêtes

DÉMONSTRATIONS

Conclusion

- `pg_stat_monitor` est fortement couplé à PMM
- De bonnes idées pour `pg_stat_statements`
- Si vous ne connaissiez pas, essayez `pg_stat_statements`

Questions