PostgreSQL 16
Add pg_stat_io view, providing more detailed IO statistics

PostgreSQL Unconferecnce #41 (2023-04-24)

自己紹介

- ぬこ@横浜，@nuko_yokohama
- にゃーん
- 趣味でポスグレをやってる者だ
- postgres is watching you!

pg_stat_io 統計情報ビュー

Add pg_stat_io view

- PostgreSQL データベース全体の I/O に関する統計情報を表示するビュー
- 2022-02-11 に commit された改善項目。
  - https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=a9c70b46dbe152e094f137f7e6ba9cd3a638ee25
- このスライドは 2023-04-22 にコミットされた版 (commit 0ec0e2095579ca16fb4f3aee086e57fd1a7533eb) を使って検証しています。

pg_stat_io が持っている列

- pg_stat_io の列 (1/2)

列名	型	意味
backend_type	text	バックエンドのタイプ（例：バックグラウンドワーカー、 autovacuum ワーカー）。詳細は pg_stat_activity を参照。一部の backend_types は I/O 操作の統計情報を蓄積しないため、ビューに含まれない
object	text	I/O 操作の対象オブジェクト。値域は以下。 relation temp relation
context	text	I/O 操作のコンテキスト。値域は normal, vacuum, bulkread, bulkwrite 。 詳細は、 **context の意味を参照。**
reads	bigint	op_bytes で指定されたサイズの読み出し操作の数。
read_time	double precision	読み取り操作に費やした時間（ミリ秒） track_io_timing が有効な場合のみ設定される。無効の場合は 0 になる。
writes	bigint	op_bytes で指定されたサイズの書き込み操作の数。
write_time	double precision	書き込み操作に費やした時間（ミリ秒） track_io_timing が有効な場合のみ設定される。無効の場合は 0 になる。

pg_stat_io が持っている列

- pg_stat_io の列 (2/2)

列名	型	意味
extends	bigint	op_bytes で指定されたリレーションのサイズ拡張操作の数。
extend_time	double precision	拡張操作に費やした時間（ミリ秒）。 track_io_timing が有効な場合のみ設定される。無効の場合は 0 になる。
op_bytes bigint	bigint	読み出し、書き込み、拡張操作 I/O の単位あたりのバイト数。 関係データの読み取り、書き込み、拡張は、構築時パラメータ BLCKSZ から得られる block_size 単位で行われ、デフォルトでは 8192 である。
hits	bigint	共有バッファで目的のブロックが見つかった回数。
evictions	bigint	ブロックが別の用途に利用できるようにするために、共有バッファまたはローカルバッファから書き出された回数。 context により数値の意味が異なる。詳細は、 **evictions の意味を参照。**
reuses	bigint	共有バッファ以外のサイズ制限のあるリングバッファの既存のバッファが、 bulkread 、 bulkwrite 、または vacuum io_contexts で I/O 操作の一部として再使用された回数。
fsyncs	bigint	fsync コールの数。これらは io_context が normal の場合のみ追跡される。
fsync_time	double precision	fsync 操作に費やした時間（ミリ秒）。
stats_reset	timestamp with time zone	統計が最後にリセットされた時刻。

pg_stat_io が持っている列

- context の意味

値	意味
normal	標準の context 。 例えば、デフォルトでは、関係データは共有バッファに読み込まれ、共有バッファから書き出される。したがって、共有バッファへのリレーションデータの読み出しと書き込みは、 context normal で追跡される。
vacuum	パーマネントリレーションのバキュームと分析中に共有バッファの外で行われる I/O 操作。一時的なテーブルのバキュームは、他の一時的なテーブル IO 操作と同じローカルバッファプールを使用し、 context normal で追跡される。
bulkread	共有バッファの外で行われる大きな読み取り I/O 操作（例えば、大きなテーブルの順次スキャンなど）。
bulkwrite	COPY など、共有バッファの外で行われる大きな書き込み I/O 操作。

pg_stat_io が持っている列

- evictions の意味

値	意味
normal	ブロックがバッファから追い出され、別のブロックに置き換えられた回数をカウントする。
vacuum bulkread bulkwrite	バルク I/O オペレーションで使用するために、共有バッファをサイズ制限のある別のリングバッファに追加するために、ブロックが共有バッファから退避した回数をカウントする。

pg_stat_io の行

- pg_stat_io の行は PostgreSQL Document を見ると、以下の組み合わせごとに 1 つの行が生成される
  - backend_type
  - object
  - context
- データベースクラスタ生成直後でも 30 行存在している。

pg_stat_io の行

- データベースクラスタ生成→起動直後の pg_stat_io ビュー

```
$ psql -p 16001 -c "TABLE pg_stat_io"
Null display is "(null)".
    backend_type     |    object     |  context  | reads  | read_time | writes | write_time | extends |
extend_time | op_bytes |  hits  | evictions | reuses | fsyncs | fsync_time |          stats_reset
--------------------+--------------+-----------+--------+-----------+--------+------------+---------
+------------+----------+--------+-----------+--------+--------+-----------+-------------------------------
 autovacuum launcher | relation     | bulkread  |      0 |         0 |      0 |          0 |  (null) |
(null) |     8192 |      0 |         0 |      0 | (null) |    (null) | 2023-04-23 16:31:28.382048+09
 (中略)
 client backend      | relation     | bulkread  |      0 |         0 |      0 |          0 |  (null) |
(null) |     8192 |      0 |         0 |      0 | (null) |    (null) | 2023-04-23 16:31:28.382048+09
 client backend      | relation     | bulkwrite |      0 |         0 |      0 |          0 |       0 |
     0 |     8192 |      0 |         0 |      0 | (null) |    (null) | 2023-04-23 16:31:28.382048+09
 client backend      | relation     | normal    |     77 |         0 |      0 |          0 |       0 |
     0 |     8192 |   1593 |         0 | (null) |      0 |         0 | 2023-04-23 16:31:28.382048+09
 client backend      | relation     | vacuum    |      0 |         0 |      0 |          0 |       0 |
     0 |     8192 |      0 |         0 |      0 | (null) |    (null) | 2023-04-23 16:31:28.382048+09
 client backend      | temp relation | normal    |      0 |         0 |      0 |          0 |       0 |
     0 |     8192 |      0 |         0 | (null) | (null) |    (null) | 2023-04-23 16:31:28.382048+09
 (中略)
(30 rows)
```

よく使いそうな検索パターン

- クエリの処理を行うバックエンドプロセスを対象にする場合には、 backend_type = ‘client backend’ を条件に指定する。
- 良く参照しそうな列
  - reads, writes あたり？

```
$ psql -p 16001 postgres -c "SELECT object, context, reads, writes FROM  pg_stat_io WHERE backend_type =
'client backend'"
Null display is "(null)".
    object     |  context  | reads | writes
---------------+-----------+-------+--------
 relation      | bulkread  |     0 |      0
 relation      | bulkwrite |     0 |      0
 relation      | normal    |   116 |      0
 relation      | vacuum    |     0 |      0
 temp relation | normal    |     0 |      0
(5 rows)
```

pgbench 初期化前後の pg_stat_io

- pgbench の初期化背景では COPY, VACUUM, INDEX 生成が動作する

```
$ createdb -p 16001 testdb
$ pgbench -p 16001 -i -s 10 -q testdb
dropping old tables...
NOTICE:  table "pgbench_accounts" does not exist, skipping
 (中略)
$
```

- context が bulkread/bulkwrite/vacuum の値が変化する

```
$ psql -p 16001 postgres -c "SELECT object, context, reads, writes FROM  pg_stat_io WHERE backend_type =
'client backend'"
Null display is "(null)".
    object     |  context  | reads | writes
---------------+-----------+-------+--------
 relation      | bulkread  |  7835 |      0
 relation      | bulkwrite |     0 |  14346
 relation      | normal    |   127 |      0
 relation      | vacuum    | 14346 |      0
 temp relation | normal    |     0 |      0
(5 rows)
```

pgbench 実行後の pg_stat_io

- pgbench -c 2 -t 5000 testdb

```
$ pgbench -p 16001 -c 2 -t 5000 testdb
pgbench (16devel)
starting vacuum...end.
 (中略)
tps = 904.397252 (without initial connection time)
$
```

- object=relation, context=normal の reads の値が変化する

```
$ psql -p 16001 postgres -c "SELECT object, context, reads, writes FROM  pg_stat_io WHERE backend_type =
'client backend'"
Null display is "(null)".
    object     |  context  | reads | writes
---------------+-----------+-------+--------
 relation      | bulkread  |  7835 |      0
 relation      | bulkwrite |     0 |  14346
 relation      | normal    |  9307 |      0
 relation      | vacuum    | 14346 |      0
 temp relation | normal    |     0 |      0
(5 rows)
```

更新は共有バッファのみで行われているので writes は増えてない？

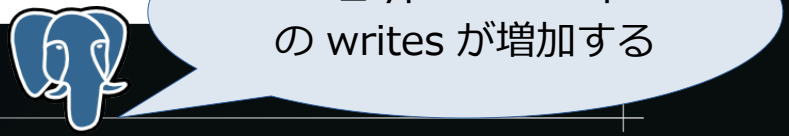
CHECKPOINT 実行後の pg_stat_io

- CHECKPOINT 前

```
$ psql -p 16001 postgres -c "SELECT context, reads, writes FROM  pg_stat_io WHERE backend_type =
'checkpointer'"
Null display is "(null)".
 context | reads  | writes
---------+--------+--------
 normal  | (null) |      0
(1 row)
```

- CHECKPOINT 後

```
$ psql -p 16001 postgres -c "SELECT context, reads, writes FROM  pg_stat_io WHERE backend_type =
'checkpointer'"
Null display is "(null)".
 context | reads  | writes
---------+--------+--------
 normal  | (null) |  12393
(1 row)
```



pg_stat_io のリセット

リセット方法については
@fujii_masao さんに
教えてもらいました

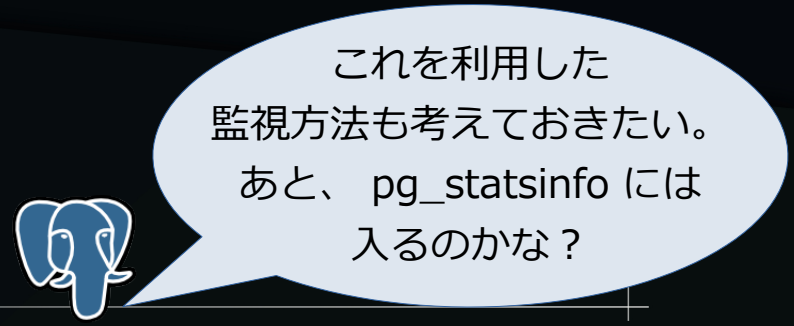
- pg_stat_i のリセットは pg_stat_reset() ではリセットできない。
- pg_stat_reset_shared('io') を使う。

```
$ psql -p 16001 postgres -c "SELECT pg_stat_reset_shared('io')"
Null display is "(null)".
 pg_stat_reset_shared
----------------------


(1 row)

$ psql -p 16001 postgres -c "SELECT context, reads, writes, stats_reset FROM  pg_stat_io WHERE backend_type
= 'client backend'"
Null display is "(null)".
  context  | reads | writes |          stats_reset
-----------+-------+--------+-------------------------------
 bulkread  |     0 |      0 | 2023-04-23 18:57:48.408184+09
 bulkwrite |     0 |      0 | 2023-04-23 18:57:48.408184+09
 normal    |     7 |      0 | 2023-04-23 18:57:48.408184+09
 vacuum    |     0 |      0 | 2023-04-23 18:57:48.408184+09
 normal    |     0 |      0 | 2023-04-23 18:57:48.408184+09
(5 rows)
```

backend_type='checkpointer'
の writes が増加する

まとめ

- PostgreSQL 16 から pg_stat_io を参照して、データベースクラスタ全体の I/O 関連の情報が収集できる。
- backend_type や context で情報の絞り込みが可能。
- リセットする場合は pg_stat_reset_shared('io') を使う。



おしまい