**PG\_BUFFERCACHE**

The pg\_buffercache module provides a means for examining what's happening in the shared buffer cache in real time.

 use is restricted to superusers and roles with privileges of the pg\_monitor role. Access may be granted to others using GRANT.

**Test Cases:**

**How to install pg\_buffercache: (Linux user please install contrib module)**

1. \dx
2. Create extension pg\_buffercache;

**Check database buffercache for all cache blocks in each database:**

SELECT CASE WHEN c.reldatabase IS NULL THEN ''

            WHEN c.reldatabase = 0 THEN ''

            ELSE d.datname

       END AS database,

       count(\*) AS cached\_blocks

FROM  pg\_buffercache AS c

      LEFT JOIN pg\_database AS d

           ON c.reldatabase = d.oid

GROUP BY d.datname, c.reldatabase

ORDER BY d.datname, c.reldatabase;

**Check how many blocks are empty/dirty/clean:**

SELECT buffer\_status, sum(count) AS count

  FROM (SELECT CASE isdirty

                 WHEN true THEN 'dirty'

                 WHEN false THEN 'clean'

                 ELSE 'empty'

               END AS buffer\_status,

               count(\*) AS count

          FROM pg\_buffercache

          GROUP BY buffer\_status

        UNION ALL

          SELECT \* FROM (VALUES ('dirty', 0), ('clean', 0), ('empty', 0)) AS tab2 (buffer\_status,count)) tab1

  GROUP BY buffer\_status;

**Issue Checkpoint:** (Run the above query again and check how many pages are dirty)

Checkpoint;

**In the current database how many table are cache and how many buffer used.**

SELECT n.nspname, c.relname, count(\*) AS buffers

FROM pg\_buffercache b JOIN pg\_class c

ON b.relfilenode = pg\_relation\_filenode(c.oid) AND

b.reldatabase IN (0, (SELECT oid FROM pg\_database

WHERE datname = current\_database()))

JOIN pg\_namespace n ON n.oid = c.relnamespace

GROUP BY n.nspname, c.relname

ORDER BY 3 DESC

LIMIT 10;

**Inspect Individual table in buffer cache.**

SELECT \* FROM pg\_buffercache WHERE relfilenode = pg\_relation\_filenode('pgbench\_history');

**Inspect buffer cache for tables and indexes which are cache:**

SELECT c.relname, c.relkind, count(\*)

       FROM   pg\_database AS a, pg\_buffercache AS b, pg\_class AS c

       WHERE  c.relfilenode = b.relfilenode

              AND b.reldatabase = a.oid

              AND c.oid >= 16384

              AND a.datname = 'postgres'

       GROUP BY 1, 2

       ORDER BY 3 DESC, 1;

**Inspect buffer cache to know how much portion of table/index is buffered, in percentage and in terms of relation:**

SELECT

c.relname,

pg\_size\_pretty(count(\*) \* 8192) as buffered,

round(100.0 \* count(\*) /

(SELECT setting FROM pg\_settings

WHERE name='shared\_buffers')::integer,1)

AS buffers\_percent,

round(100.0 \* count(\*) \* 8192 /

pg\_table\_size(c.oid),1)

AS percent\_of\_relation

FROM pg\_class c

INNER JOIN pg\_buffercache b

ON b.relfilenode = c.relfilenode

INNER JOIN pg\_database d

ON (b.reldatabase = d.oid AND d.datname = current\_database())

GROUP BY c.oid,c.relname

ORDER BY 3 DESC

LIMIT 10;

**Find all blocks and their usage count:**

SELECT

c.relname, count(\*) AS buffers,usagecount

FROM pg\_class c

INNER JOIN pg\_buffercache b

ON b.relfilenode = c.relfilenode

INNER JOIN pg\_database d

ON (b.reldatabase = d.oid AND d.datname = current\_database())

GROUP BY c.relname,usagecount

ORDER BY c.relname,usagecount;

**Distribution of blocks based on usage\_count:**

SELECT usagecount, count(\*)

FROM pg\_buffercache

GROUP BY usagecount

ORDER BY usagecount;

**How much percentage of table/index is cache and how much % is hot:**

SELECT c.relname,

count(\*) blocks,

round( 100.0 \* 8192 \* count(\*) / pg\_table\_size(c.oid) ) "% of rel",

round( 100.0 \* 8192 \* count(\*) FILTER (WHERE b.usagecount > 3) / pg\_table\_size(c.oid) ) "% hot"

FROM pg\_buffercache b

JOIN pg\_class c ON pg\_relation\_filenode(c.oid) = b.relfilenode

WHERE b.reldatabase IN (

0, (SELECT oid FROM pg\_database WHERE datname = current\_database())

)

AND b.usagecount is not null

GROUP BY c.relname, c.oid

ORDER BY 2 DESC

LIMIT 10;