



北京海量数据技术股份有限公司

2020年8月

**PostgreSQL数据库常用监控指标**

**目 录**

[1 文档说明 4](#_Toc54972973)

[2 监控指标 4](#_Toc54972974)

[2.1 会话连接数 4](#_Toc54972975)

[2.2 缓存命中率 4](#_Toc54972976)

[2.3 每秒事务数和DML元组数 5](#_Toc54972977)

[2.4 等待事件 6](#_Toc54972978)

[2.5 不记录log的对象 6](#_Toc54972979)

[2.6 密码到期时间 7](#_Toc54972980)

[2.7 长事务 7](#_Toc54972981)

[2.8 当前执行的SQL 7](#_Toc54972982)

[2.9 数据库空间大小 8](#_Toc54972983)

[2.10 表空间使用量 9](#_Toc54972984)

[2.11 大表监控 9](#_Toc54972985)

[2.12 大索引监控 10](#_Toc54972986)

[2.13 表碎片 10](#_Toc54972987)

[2.14 索引碎片 14](#_Toc54972988)

[2.15 流复制延迟 17](#_Toc54972989)

[3 统计信息表 18](#_Toc54972990)

[3.1 统计信息概览 18](#_Toc54972991)

[3.2 pg\_stat\_database 18](#_Toc54972992)

[3.3 pg\_stat\_user\_tables 19](#_Toc54972993)

[3.4 pg\_stat\_user\_indexes 20](#_Toc54972994)

[3.5 pg\_statio\_user\_tables 20](#_Toc54972995)

[3.6 pg\_stat\_bgwriter 20](#_Toc54972996)

[3.7 pg\_stat\_replication 21](#_Toc54972997)

[3.8 pg\_stat\_statements 21](#_Toc54972998)

[3.9 pg\_stat\_replication 22](#_Toc54972999)

[4 监控相关插件 23](#_Toc54973000)

[4.1 pg\_wait\_sampling 23](#_Toc54973001)

[4.2 pg\_stat\_kcache 23](#_Toc54973002)

[4.3 pgcenter 23](#_Toc54973003)

[4.4 pg\_activity 23](#_Toc54973004)

# 文档说明

本文收录PostgreSQL多种监控指标，旨在为运维人员提供便利。

# 监控指标

(1)将exBase 安装包上传，并解压到根目录；

unzip exbase\_environment\_20190812.zip -d /exbase\_environment

## 会话连接数

查看当前连接到数据库会话数，与运行连接的最大会话数

SELECT count(\*) as current\_conns,

current\_setting('max\_connections') as max\_conns

FROM pg\_stat\_activity;

查看当前连接到数据库活跃会话与空闲会话数量，state='idle' 空闲会话，state in ('active','background') 活跃会话。

SELECT case when state is null then 'background' else state end,

count(\*)

FROM pg\_stat\_activity group by state;

## 缓存命中率

数据库缓存命中率，一般而言都能达到90%以上，现在一般不作为数据库是否优化的参考指标。只有在命中率较低的情况下，才考虑是否缓存大小设置的不合理，或是业务模式导致数据共享率不高。

SELECT a.datname,

CAST(pg\_database\_size(a.datid)/1024/1024 AS NUMERIC(5, 2)) AS size\_Mb,

CAST(blks\_hit / (blks\_read + blks\_hit + 0.000001) \* 100.0 AS NUMERIC(5, 2)) AS buffer\_hint,

CAST(xact\_commit / (xact\_rollback + xact\_commit + 0.000001) \* 100.0 AS NUMERIC(5, 2)) AS success\_trans

FROM pg\_stat\_database a

where datname not in ('template0','template1')

ORDER BY pg\_database\_size(a.datid) DESC;

## 每秒事务数和DML元组数

统计每个数据库中每秒增、删、改、查数据量。

tps:每个库的tps；

xact\_commit:每个库每秒的提交数；

xact\_rollback:每个库每秒的回滚数；

tup\_retuen 对于表来说是全表扫描的行数，对于索引是通过索引⽅方法返回的索引行数，如果这个值数量明显⼤于tup\_fetched，说明当前数据库存在⼤大量全表扫描的情况。 接近于1最好，表示走索引多；

blk\_read\_time 数据库中花费在读取⽂文件的时间，这个值较⾼高说明内存较⼩小，需要频繁的从磁盘中读⼊入数据⽂文件， 单位毫秒；

blk\_write\_time 数据库中花费在写数据⽂文件的时间，pg中脏⻚页一般都写⼊入page cache，如果这个值较⾼高，说明page cache较⼩小，操作系统的page cache需要更更积极的写⼊，单位毫秒。

select datname,xact\_commit,xact\_rollback,(xact\_commit+xact\_rollback) as tps,tup\_returned,tup\_fetched,case when tup\_fetched+tup\_returned=0 then 0 else round(tup\_fetched::numeric/(tup\_fetched+tup\_returned),2) end as tablefullscan,tup\_inserted,tup\_updated,tup\_deleted, blk\_read\_time,blk\_write\_time

from pg\_stat\_database

where datname not in ('template0','template1');

## 等待事件

记录每个数据库中的等待事件。

select datname,wait\_event\_type||'\_'||wait\_event,count(\*)

from pg\_stat\_activity

where state in ('active', 'fastpath function call')

and pid<>pg\_backend\_pid()

group by datname,wait\_event\_type||'\_'||wait\_event;

## 不记录log的对象

unlogged table和hash index不记录XLOG, 无法使用流复制或者log shipping的方式复制到standby节点，如果在standby节点执行某些SQL，可能导致报错或查不到数据。 在数据库CRASH后无法修复unlogged table和hash index, 不建议使用。 PITR对unlogged table和hash index也不起作用。

select current\_database(),t3.rolname,t2.nspname,t1.relname,pg\_get\_indexdef(t1.oid)

from pg\_class t1,pg\_namespace t2,pg\_authid t3

where t1.relnamespace=t2.oid

and t1.relowner=t3.oid

and (t1.relpersistence=$$u$$ or(t1.relkind=$$i$$ and pg\_get\_indexdef(t1.oid) ~ $$USING hash$$));

## 密码到期时间

到期后, 用户将无法登陆, 记得修改密码, 同时将密码到期时间延长到某个时间或无限时间, alter role ... VALID UNTIL 'timestamp' .

select rolname,rolvaliduntil from pg\_authid order by rolvaliduntil;

## 长事务

长事务过程中产生的垃圾, 无法回收, 建议不要在数据库中运行LONG SQL, 或者错开DML高峰时间去运行LONG SQL。

SELECT datname, usename, query, xact\_start,

now() - xact\_start xact\_duration, query\_start, now() - query\_start query\_duration, STATE

FROM pg\_stat\_activity WHERE STATE <>'idle' and now()-xact\_start > interval '30 SECOND' ORDER BY xact\_start;

## 当前执行的SQL

SELECT

procpid,

START,

now() - START AS lap,

current\_query

FROM

(

SELECT

backendid,

pg\_stat\_get\_backend\_pid (S.backendid) AS procpid,

pg\_stat\_get\_backend\_activity\_start (S.backendid) AS START,

pg\_stat\_get\_backend\_activity (S.backendid) AS current\_query

FROM

(

SELECT

pg\_stat\_get\_backend\_idset () AS backendid

) AS S

) AS S

WHERE

current\_query <> '<IDLE>'

ORDER BY

lap DESC;

procpid：[进程id](http://www.baidu.com/s?wd=%E8%BF%9B%E7%A8%8Bid&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLrycvuyn3mWmvrjTLnhf10ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPjnzrHD4PWc1P1ckrHTsnW0Y)  
start：进程开始时间  
lap：经过时间  
current\_query：执行中的sql  
取消正在执行的sql：SELECT pg\_cancel\_backend([进程id](http://www.baidu.com/s?wd=%E8%BF%9B%E7%A8%8Bid&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLrycvuyn3mWmvrjTLnhf10ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPjnzrHD4PWc1P1ckrHTsnW0Y));

## 数据库空间大小

监控各个数据库占用磁盘空间的大小。

select datname,pg\_database\_size(datname)/1024/1024 as size\_mb

from pg\_database

where datname not in ('template0','template1')

group by datname;

## 表空间使用量

select pg\_tablespace\_size(spcname)/1024/1024 as "tablespace\_total\_usedmsize",spcname,pg\_tablespace\_location(oid)

from pg\_tablespace where pg\_tablespace\_location(oid) != ''

select pg\_tablespace\_size(spcname)/1024/1024 as "tablespace\_total\_usedmsize",spcname,

case when pg\_tablespace\_location(oid) = '' then (select setting from pg\_settings where name = 'data\_directory')

else pg\_tablespace\_location(oid) end from pg\_tablespace

## 大表监控

纪录大表占用空间的历史纪录，因为mvcc多版本模式需要考虑空间问题，确定是否需要vacuum。

select current\_database(),table\_name,pg\_relation\_size(table\_schema||'.'||table\_name) as size\_bytes

from information\_schema.tables

where table\_schema not in ('pg\_catalog', 'information\_schema')

order by size\_bytes desc

LIMIT 10;

## 大索引监控

SELECT current\_database(),A.rolname,nspname,relname,

round(100 \* pg\_relation\_size ( indexrelid ) / pg\_relation\_size ( indrelid )) / 100 AS index\_ratio,

pg\_size\_pretty (pg\_relation\_size ( indexrelid )) AS index\_size,

pg\_size\_pretty (pg\_relation\_size ( indrelid )) AS table\_size

FROM pg\_index I

LEFT JOIN pg\_class C ON ( C.oid = I.indexrelid )

LEFT JOIN pg\_namespace N ON ( N.oid = C.relnamespace )

LEFT JOIN pg\_authid A ON ( C.relowner = A.oid )

WHERE nspname NOT IN ( 'pg\_catalog', 'information\_schema', 'pg\_toast' )AND C.relkind = 'i'

AND pg\_relation\_size ( indrelid ) > 0

order by pg\_relation\_size(indexrelid) desc limit 10;

## 表碎片

根据浪费的字节数，设置合适的autovacuum\_vacuum\_scale\_factor，大表如果频繁的有更新或删除和插入操作，建议设置较小的autovacuum\_vacuum\_scale\_factor来降低浪费空间。同时还需要打开autovacuum, 根据服务器的内存大小，CPU核数，设置足够大的autovacuum\_work\_mem 或 autovacuum\_max\_workers 或 maintenance\_work\_mem, 以及足够小的 autovacuum\_naptime。

同时还需要分析是否对大数据库使用了逻辑备份pg\_dump，系统中是否经常有长SQL，长事务。这些都有可能导致膨胀。使用pg\_reorg或者vacuum full可以回收膨胀的空间。

otta评估出的表实际需要页数, iotta评估出的索引实际需要页数。bs数据库的块大小，tbloat表膨胀倍数， ibloat索引膨胀倍数,，wastedpages表浪费了多少个数据块,，wastedipages索引浪费了多少个数据块；wastedbytes表浪费了多少字节， wastedibytes索引浪费了多少字节；

SELECT

current\_database() AS db, schemaname, tablename, reltuples::bigint AS tups, relpages::bigint AS pages, otta,

ROUND(CASE WHEN otta=0 OR sml.relpages=0 OR sml.relpages=otta THEN 0.0 ELSE sml.relpages/otta::numeric END,1) AS tbloat,

CASE WHEN relpages < otta THEN 0 ELSE relpages::bigint - otta END AS wastedpages,

CASE WHEN relpages < otta THEN 0 ELSE bs\*(sml.relpages-otta)::bigint END AS wastedbytes,

CASE WHEN relpages < otta THEN $$0 bytes$$::text ELSE (bs\*(relpages-otta))::bigint || $$ bytes$$ END AS wastedsize,

iname, ituples::bigint AS itups, ipages::bigint AS ipages, iotta,

ROUND(CASE WHEN iotta=0 OR ipages=0 OR ipages=iotta THEN 0.0 ELSE ipages/iotta::numeric END,1) AS ibloat,

CASE WHEN ipages < iotta THEN 0 ELSE ipages::bigint - iotta END AS wastedipages,

CASE WHEN ipages < iotta THEN 0 ELSE bs\*(ipages-iotta) END AS wastedibytes,

CASE WHEN ipages < iotta THEN $$0 bytes$$ ELSE (bs\*(ipages-iotta))::bigint || $$ bytes$$ END AS wastedisize,

CASE WHEN relpages < otta THEN

CASE WHEN ipages < iotta THEN 0 ELSE bs\*(ipages-iotta::bigint) END

ELSE CASE WHEN ipages < iotta THEN bs\*(relpages-otta::bigint)

ELSE bs\*(relpages-otta::bigint + ipages-iotta::bigint) END

END AS totalwastedbytes

FROM (

SELECT

nn.nspname AS schemaname,

cc.relname AS tablename,

COALESCE(cc.reltuples,0) AS reltuples,

COALESCE(cc.relpages,0) AS relpages,

COALESCE(bs,0) AS bs,

COALESCE(CEIL((cc.reltuples\*((datahdr+ma-

(CASE WHEN datahdr%ma=0 THEN ma ELSE datahdr%ma END))+nullhdr2+4))/(bs-20::float)),0) AS otta,

COALESCE(c2.relname,$$?$$) AS iname, COALESCE(c2.reltuples,0) AS ituples, COALESCE(c2.relpages,0) AS ipages,

COALESCE(CEIL((c2.reltuples\*(datahdr-12))/(bs-20::float)),0) AS iotta -- very rough approximation, assumes all cols

FROM

pg\_class cc

JOIN pg\_namespace nn ON cc.relnamespace = nn.oid AND nn.nspname <> $$information\_schema$$

LEFT JOIN

(

SELECT

ma,bs,foo.nspname,foo.relname,

(datawidth+(hdr+ma-(case when hdr%ma=0 THEN ma ELSE hdr%ma END)))::numeric AS datahdr,

(maxfracsum\*(nullhdr+ma-(case when nullhdr%ma=0 THEN ma ELSE nullhdr%ma END))) AS nullhdr2

FROM (

SELECT

ns.nspname, tbl.relname, hdr, ma, bs,

SUM((1-coalesce(null\_frac,0))\*coalesce(avg\_width, 2048)) AS datawidth,

MAX(coalesce(null\_frac,0)) AS maxfracsum,

hdr+(

SELECT 1+count(\*)/8

FROM pg\_stats s2

WHERE null\_frac<>0 AND s2.schemaname = ns.nspname AND s2.tablename = tbl.relname

) AS nullhdr

FROM pg\_attribute att

JOIN pg\_class tbl ON att.attrelid = tbl.oid

JOIN pg\_namespace ns ON ns.oid = tbl.relnamespace

LEFT JOIN pg\_stats s ON s.schemaname=ns.nspname

AND s.tablename = tbl.relname

AND s.inherited=false

AND s.attname=att.attname,

(

SELECT

(SELECT current\_setting($$block\_size$$)::numeric) AS bs,

CASE WHEN SUBSTRING(SPLIT\_PART(v, $$ $$, 2) FROM $$#"[0-9]+.[0-9]+#"%$$ for $$#$$)

IN ($$8.0$$,$$8.1$$,$$8.2$$) THEN 27 ELSE 23 END AS hdr,

CASE WHEN v ~ $$mingw32$$ OR v ~ $$64-bit$$ THEN 8 ELSE 4 END AS ma

FROM (SELECT version() AS v) AS foo

) AS constants

WHERE att.attnum > 0 AND tbl.relkind=$$r$$

GROUP BY 1,2,3,4,5

) AS foo

) AS rs

ON cc.relname = rs.relname AND nn.nspname = rs.nspname

LEFT JOIN pg\_index i ON indrelid = cc.oid

LEFT JOIN pg\_class c2 ON c2.oid = i.indexrelid

) AS sml order by wastedbytes desc limit 5;

## 索引碎片

如果索引膨胀太大, 会影响性能, 建议重建索引, create index CONCURRENTLY。

SELECT current\_database() AS db, schemaname, tablename, reltuples::bigint AS tups, relpages::bigint AS pages, otta,

ROUND(CASE WHEN otta=0 OR sml.relpages=0 OR sml.relpages=otta THEN 0.0 ELSE sml.relpages/otta::numeric END,1) AS tbloat,

CASE WHEN relpages < otta THEN 0 ELSE relpages::bigint - otta END AS wastedpages,

CASE WHEN relpages < otta THEN 0 ELSE bs\*(sml.relpages-otta)::bigint END AS wastedbytes,

CASE WHEN relpages < otta THEN $$0 bytes$$::text ELSE (bs\*(relpages-otta))::bigint || $$ bytes$$ END AS wastedsize,

iname, ituples::bigint AS itups, ipages::bigint AS ipages, iotta,

ROUND(CASE WHEN iotta=0 OR ipages=0 OR ipages=iotta THEN 0.0 ELSE ipages/iotta::numeric END,1) AS ibloat,

CASE WHEN ipages < iotta THEN 0 ELSE ipages::bigint - iotta END AS wastedipages,

CASE WHEN ipages < iotta THEN 0 ELSE bs\*(ipages-iotta) END AS wastedibytes,

CASE WHEN ipages < iotta THEN $$0 bytes$$ ELSE (bs\*(ipages-iotta))::bigint || $$ bytes$$ END AS wastedisize,

CASE WHEN relpages < otta THEN

CASE WHEN ipages < iotta THEN 0 ELSE bs\*(ipages-iotta::bigint) END

ELSE CASE WHEN ipages < iotta THEN bs\*(relpages-otta::bigint)

ELSE bs\*(relpages-otta::bigint + ipages-iotta::bigint) END

END AS totalwastedbytes

FROM (

SELECT

nn.nspname AS schemaname,

cc.relname AS tablename,

COALESCE(cc.reltuples,0) AS reltuples,

COALESCE(cc.relpages,0) AS relpages,

COALESCE(bs,0) AS bs,

COALESCE(CEIL((cc.reltuples\*((datahdr+ma-

(CASE WHEN datahdr%ma=0 THEN ma ELSE datahdr%ma END))+nullhdr2+4))/(bs-20::float)),0) AS otta,

COALESCE(c2.relname,$$?$$) AS iname, COALESCE(c2.reltuples,0) AS ituples, COALESCE(c2.relpages,0) AS ipages,

COALESCE(CEIL((c2.reltuples\*(datahdr-12))/(bs-20::float)),0) AS iotta -- very rough approximation, assumes all cols

FROM

pg\_class cc

JOIN pg\_namespace nn ON cc.relnamespace = nn.oid AND nn.nspname <> $$information\_schema$$

LEFT JOIN

(

SELECT

ma,bs,foo.nspname,foo.relname,

(datawidth+(hdr+ma-(case when hdr%ma=0 THEN ma ELSE hdr%ma END)))::numeric AS datahdr,

(maxfracsum\*(nullhdr+ma-(case when nullhdr%ma=0 THEN ma ELSE nullhdr%ma END))) AS nullhdr2

FROM (

SELECT

ns.nspname, tbl.relname, hdr, ma, bs,

SUM((1-coalesce(null\_frac,0))\*coalesce(avg\_width, 2048)) AS datawidth,

MAX(coalesce(null\_frac,0)) AS maxfracsum,

hdr+(

SELECT 1+count(\*)/8

FROM pg\_stats s2

WHERE null\_frac<>0 AND s2.schemaname = ns.nspname AND s2.tablename = tbl.relname

) AS nullhdr

FROM pg\_attribute att

JOIN pg\_class tbl ON att.attrelid = tbl.oid

JOIN pg\_namespace ns ON ns.oid = tbl.relnamespace

LEFT JOIN pg\_stats s ON s.schemaname=ns.nspname

AND s.tablename = tbl.relname

AND s.inherited=false

AND s.attname=att.attname,

(

SELECT

(SELECT current\_setting($$block\_size$$)::numeric) AS bs,

CASE WHEN SUBSTRING(SPLIT\_PART(v, $$ $$, 2) FROM $$#"[0-9]+.[0-9]+#"%$$ for $$#$$)

IN ($$8.0$$,$$8.1$$,$$8.2$$) THEN 27 ELSE 23 END AS hdr,

CASE WHEN v ~ $$mingw32$$ OR v ~ $$64-bit$$ THEN 8 ELSE 4 END AS ma

FROM (SELECT version() AS v) AS foo

) AS constants

WHERE att.attnum > 0 AND tbl.relkind=$$r$$

GROUP BY 1,2,3,4,5

) AS foo

) AS rs

ON cc.relname = rs.relname AND nn.nspname = rs.nspname

LEFT JOIN pg\_index i ON indrelid = cc.oid

LEFT JOIN pg\_class c2 ON c2.oid = i.indexrelid

) AS sml order by wastedibytes desc limit 5;

## 流复制延迟

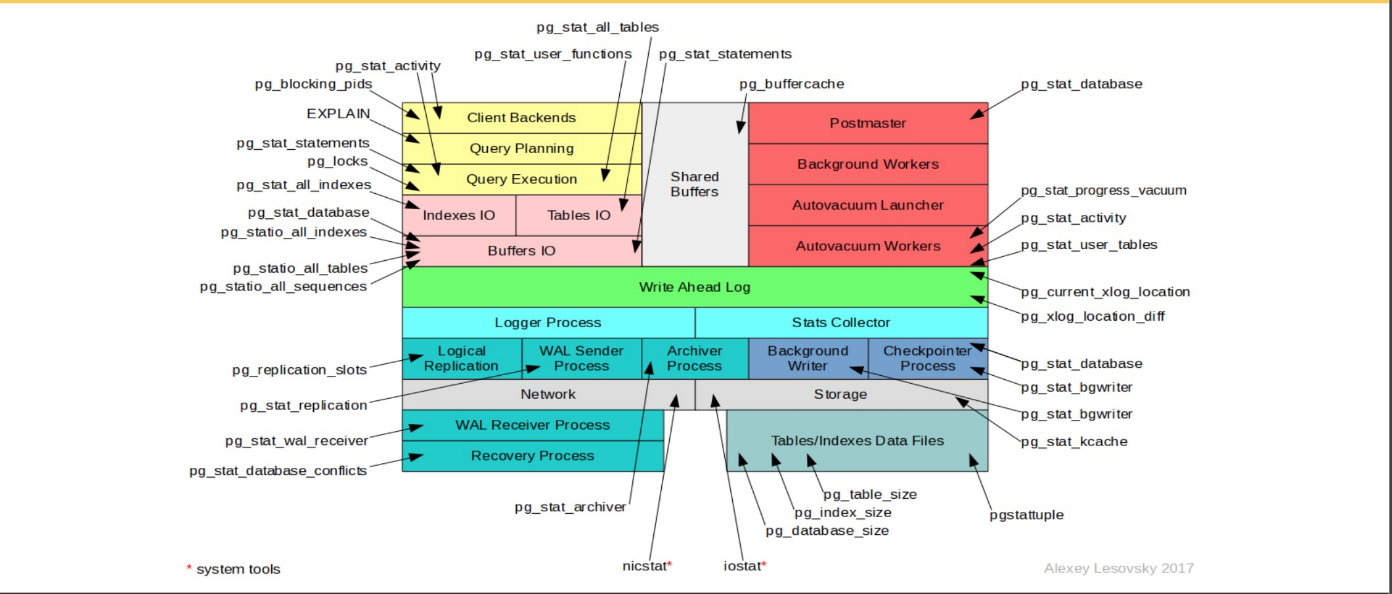
关注流复制的延迟, 如果延迟非常大, 建议排查网络带宽, 以及本地读xlog的性能, 远程写xlog的性能。

select pg\_xlog\_location\_diff(pg\_current\_xlog\_location(),flush\_location), \* from pg\_stat\_replication;

# 统计信息表

## 统计信息概览

原文地址：<https://yq.aliyun.com/articles/697692>

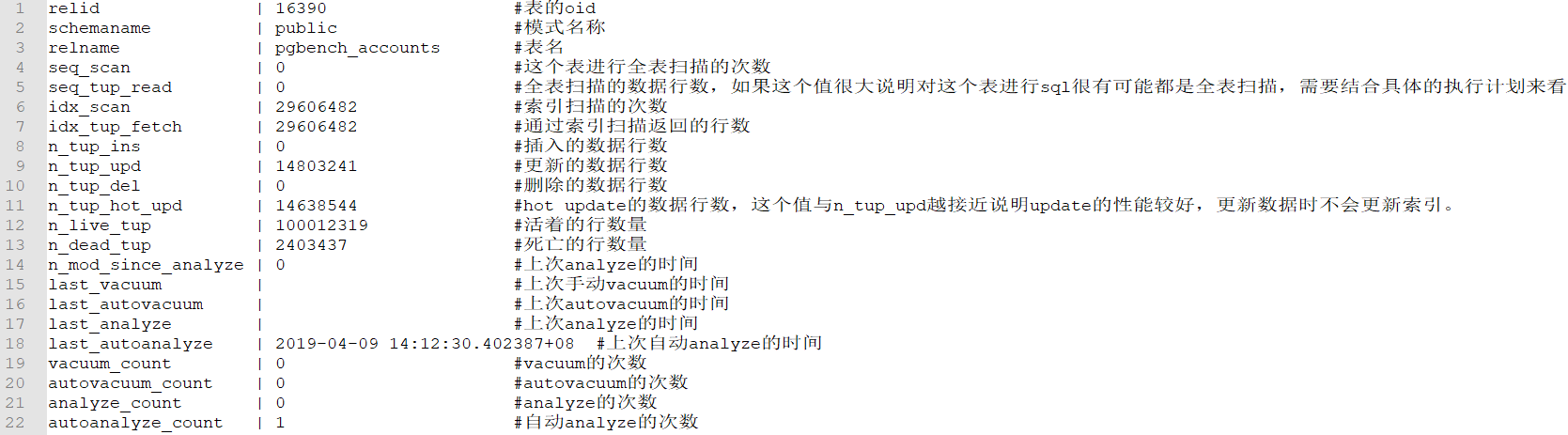


## pg\_stat\_database



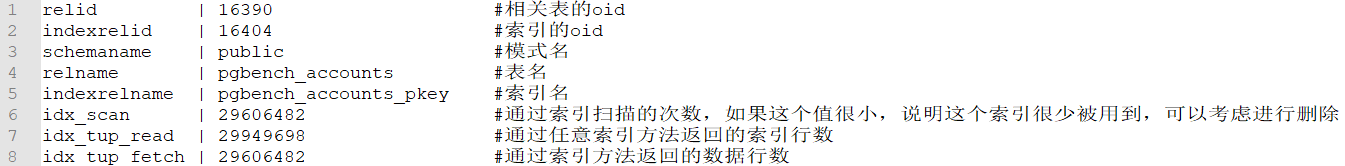
通过pg\_stat\_database我们就可以大概了解数据库的历史情况，比如看到tup\_returned值远大于tup\_fetched，说明数据库历史执行的sql很多都是全表扫描，说明存在很多没有走索引的sql，这时候可以结合pg\_stat\_statments来查找慢sql，也可以通过pg\_stat\_user\_tables找到全表扫描次数和行数最多的表。通过看到tup\_updated很高说明数据库有很频繁的更新，这个时候就需要关注一下vacuum相关的指标和长事务，如果没有及时进行垃圾回收会造成数据膨胀的比较厉害。如果temp\_files较高的话说明存在很多的排序，hash，或者聚合这种操作，可以通过增大work\_mem减少临时文件的产生，并且同时这些操作的性能也会有较大的提升。

## pg\_stat\_user\_tables



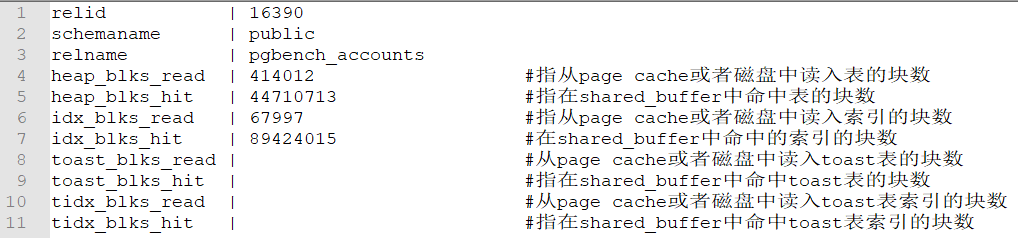
通过查询pg\_stat\_user\_tables，可以基本清楚哪些表的全表扫描的次数较多，表中是插入还是更新，删除比较多。也可以了解当前表中垃圾数据的数量。

## pg\_stat\_user\_indexes



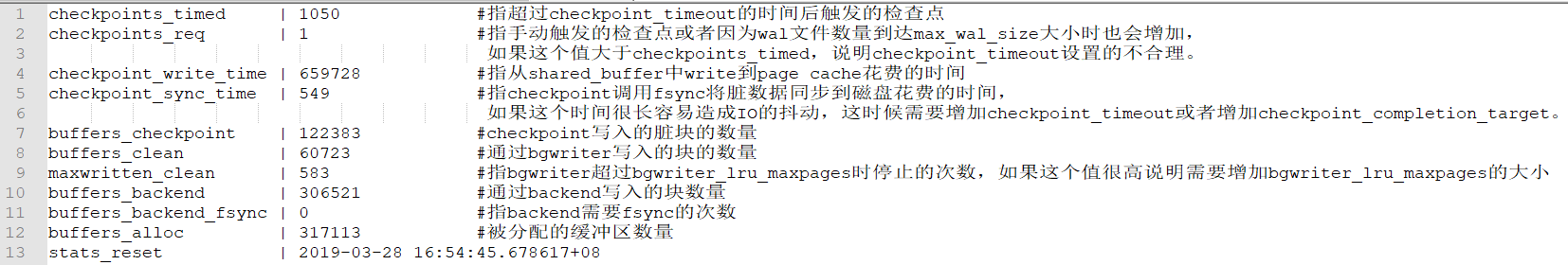
通过pg\_stat\_user\_indexes可以知道当前数据库中哪些是用的很频繁的索引，哪些是无效索引，无效索引可以进行删除，可以减少磁盘空间的使用和提升insert，update，delete性能。

## pg\_statio\_user\_tables



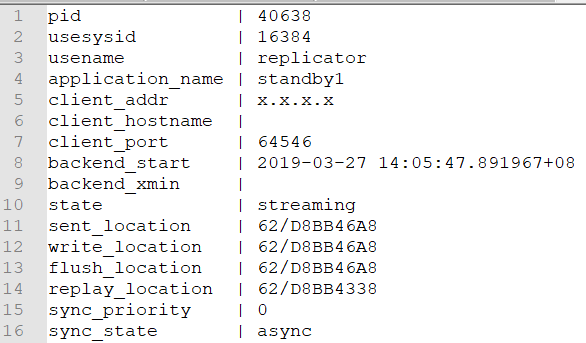
如果heap\_blks\_read，idx\_blks\_read很高说明shared\_buffer较小，存在频繁需要从磁盘或者page cache读取到shared\_buffer中。

## pg\_stat\_bgwriter



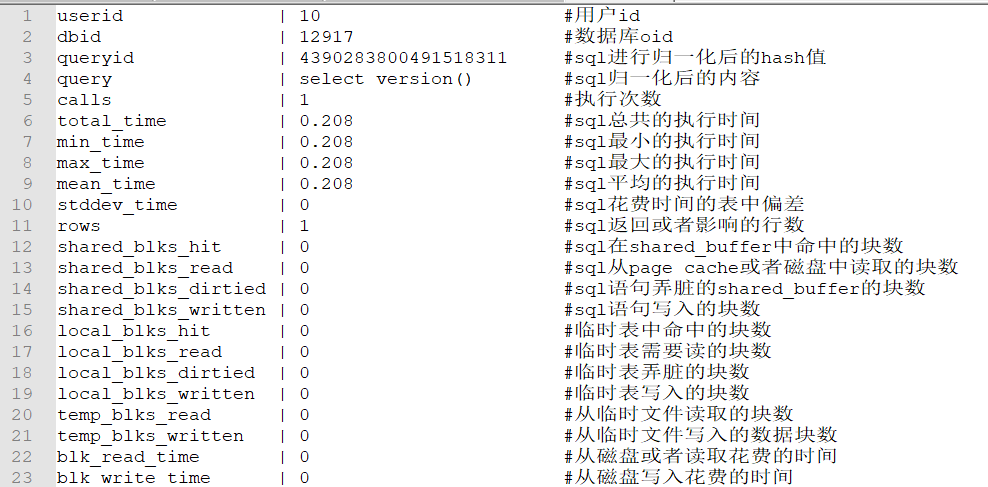
通过这个视图我们可以判断checkpoint以及max\_wal\_size的相关参数是否合理。也可以判断bgwriter相关的参数是否合理。

## pg\_stat\_replication

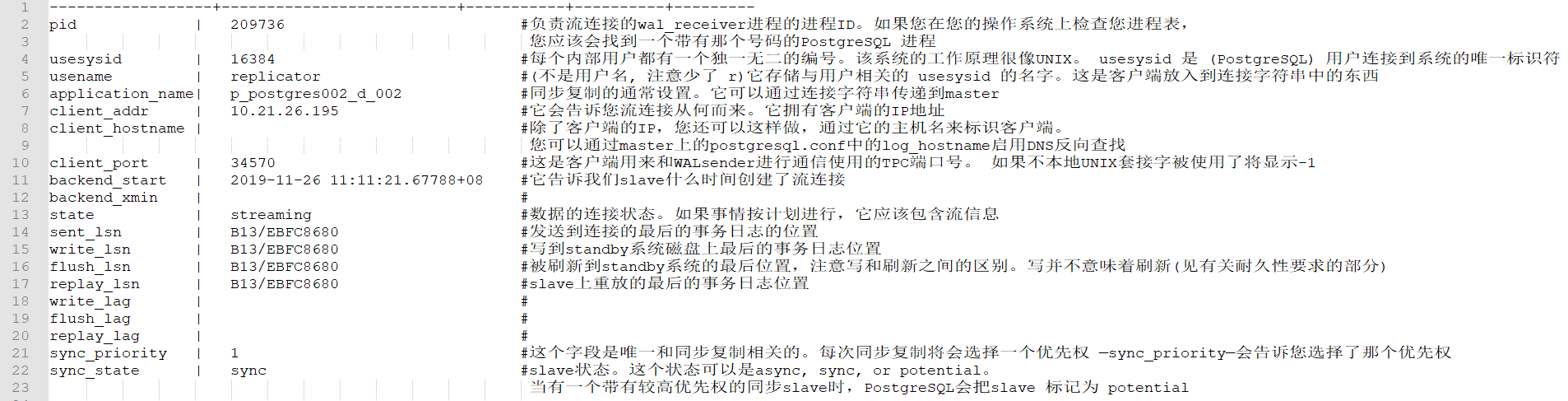


通过这个视图可以检查当前流复制的状态。检查备库replay的进度。

## pg\_stat\_statements



## pg\_stat\_replication



这个系统视图中每个记录只代表一个slave。因此，您看到谁处于连接状态，在做什么任务。pg\_stat\_replication也是检查slave是否处于连接状态的一个好方法。

# 监控相关插件

## pg\_wait\_sampling

可以收集数据库中产生的所有等待事件的信息，通过等待事件可以了解数据库的一些瓶颈，适合用于压测中发现性能瓶颈，平时也可以用于定位分析问题。

[https://github.com/postgrespro/pg\_wait\_sampling](https://yq.aliyun.com/go/articleRenderRedirect?url=https%3A%2F%2Fgithub.com%2Fpostgrespro%2Fpg_wait_sampling)

## pg\_stat\_kcache

可以获取些更底层的信息，从文件系统中读写花费的时间，cpu使用的时间等等，可以结合pg\_stat\_statements得到更多的信息。

[https://github.com/powa-team/pg\_stat\_kcache](https://yq.aliyun.com/go/articleRenderRedirect?url=https%3A%2F%2Fgithub.com%2Fpowa-team%2Fpg_stat_kcache)

## pgcenter

一个命令工具可以进行性能问题排查和分析，pgcenter结合pg内部的统计信息视图，方便快速查找和定位问题。

[https://github.com/lesovsky/pgcenter](https://yq.aliyun.com/go/articleRenderRedirect?url=https%3A%2F%2Fgithub.com%2Flesovsky%2Fpgcenter)

## pg\_activity

监控pg系统状态的命令工具 [https://github.com/julmon/pg\_activity](https://yq.aliyun.com/go/articleRenderRedirect?url=https%3A%2F%2Fgithub.com%2Fjulmon%2Fpg_activity)