Опыт использования Postgres в Avito

Murat Kabilov

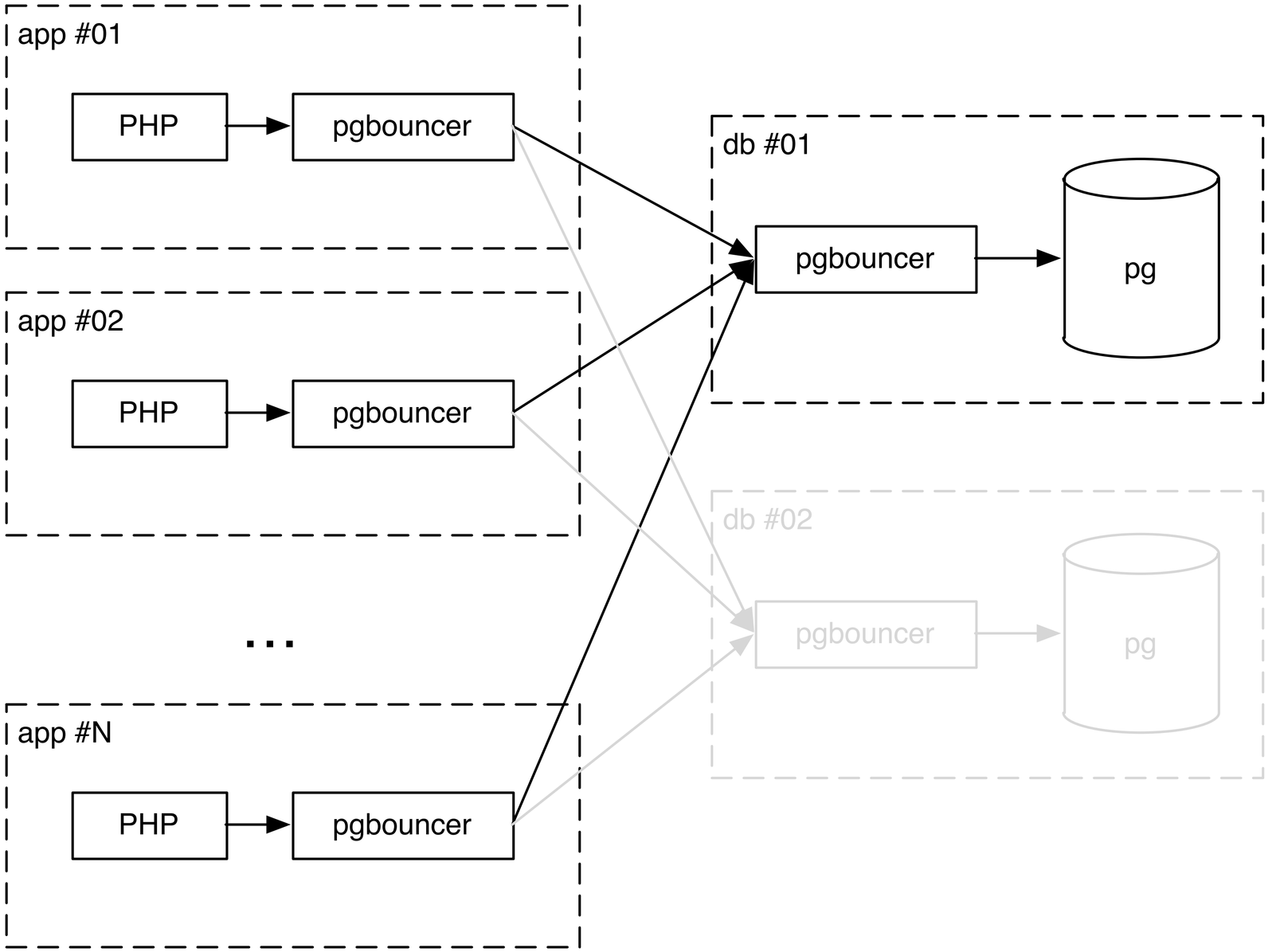
skytools

skytools

- pgbouncer
- plproxy
- pgq
- londiste

pgbouncer

- каскадные
  - ограничение количества соединений
- transaction pool mode
- connect_query='select prepare_plans(<log on/off>);'
- профилирование
- client_idle_timeout & php pgq daemon


app #01
PHP
pgbouncer
app #02
PHP
pgbouncer
...
app #N
PHP
pgbouncer
db #01
pgbouncer
pg
db #02
pgbouncer
pg

pl/proxy

- распределенное хранилище
- «правильная» хеш-функция
  - md5($1)::bit(32)::int
- резервирование
- расчет статистики

pgq

- «транзакционность»
- относительная простота настройки
- php консьюмер
- rpc
- def procs

londiste

- табличная репликация поверх pgq
- мат. представление на отдельный сервер
- надежная
- размер буфера
  512*1024 -> 512*1024*1024

avito

avito

- статистика по seq, idx сканы
- pg_stat_statements, track_activity_query_size(int), queryid & 9.4
- pgbadger ~~pgfouine~~
- дамп ХП в git
- inline кода

avito

- билд словарей в php (массивы)
- не используем foreign key, постпроверки
- tmpfs tablespace
- DDL: deadlock_timeout & statement_timeout
- таймзоны и restart
- sql f(), plpgsql f(), prepared statements

хранимые процедуры

```
CREATE OR REPLACE FUNCTION core.x_advert_status(i_status_name text) RETURNS smallint
 LANGUAGE plpgsql IMMUTABLE STRICT
AS $function$
  declare
    v_status_name constant text := lower(btrim(i_status_name));
  begin
    case v_status_name
      when 'active'    then return 1;
      when 'suspended' then return 2;
      when 'rejected'  then return 3;
      when 'blocked'   then return 4;
      when 'removed'   then return 5;
    else
        raise 'status "%" is not found', v_status_name;
    end case;
  end;
$function$;
```

хранимые процедуры

```
CREATE OR REPLACE FUNCTION core.advert_save(
  i_advert_id integer,
  i_category_id integer,
  i_params_hst hstore,
  i_title text,
  i_description text,
  ...
  OUT error_code integer,
  OUT o_advert_id integer)
 RETURNS integer
 LANGUAGE plpgsql
AS $function$
declare
  v_advert record;
  v_microcategory_id integer;
  v_campaign record;
begin
  -- some stuff

  return;
end;
$function$;
```

cte

```
select a.title, a.status, a.price, a.contacts, u.login, u.email, l.location_name,
  c.category_name, a.create_time, s.shop_name
from adverts a
  left join users u on u.user_id = a.user_id
  left join locations l on l.location_id = a.location_id
  left join categories c on c.category_id = a.category_id
  left join shops s on s.user_id = a.user_id
...
where a.user_id in (...)
order by a.create_time desc
limit 100
```

cte

```
with a as (
    select * from adverts where a.user_id in (...)
    order by a.create_time desc
    limit 100
)
select a.title, a.status, a.price, a.contacts, u.login, u.email, l.location_name,
  c.category_name, a.create_time, s.shop_name
from a
  left join users u on u.user_id = a.user_id
  left join locations l on l.location_id = a.location_id
  left join categories c on c.category_id = a.category_id
  left join shops s on s.user_id = a.user_id
...
order by a.create_time
limit 100
```

deferred triggers

B before trigger: NEW.last_update_txtime = NOW();

```
create table adverts(
  last_update_txtime timestamp with time zone,
  ...
)

B deferred trigger:
if TG_OP = 'UPDATE' then
  if OLD.last_update_txtime is distinct from NEW.last_update_txtime then
    execute 'select hstore(a) from adverts a where a.advert_id = $1'
      into data using NEW.advert_id;
  end if;
end if;
```

vacuum & toast

- 100% cpu при vacuum
- toast-колонки и их распаковка

```
ALTER TABLE adverts ALTER COLUMN
description SET STATISTICS 0;
```

функциональные индексы

```
test=# show autovacuum;
 autovacuum
------------
 off
(1 row)

test=# create table my_table as select i from generate_series(1, 100000) i;
SELECT 100000
test=# analyze my_table;
ANALYZE
```

функциональные индексы

```
test=# create index on my_table (md5(i::text));
CREATE INDEX

test=# explain select * from my_table where md5(i::text) =
'202cb962ac59075b964b07152d234b70';
                                  QUERY PLAN
-------------------------------------------------------------------------------
 Bitmap Heap Scan on my_table  (cost=20.30..494.39 rows=500 width=4)
   Recheck Cond: (md5((i)::text) = '202cb962ac59075b964b07152d234b70'::text)
   ->  Bitmap Index Scan on my_table_md5_idx  (cost=0.00..20.18 rows=500 width=0)
         Index Cond: (md5((i)::text) = '202cb962ac59075b964b07152d234b70'::text)
(4 rows)
```

функциональные индексы

src/include/utils/selfuncs.h

```
45    /* default number of distinct values in a table */
46    #define DEFAULT_NUM_DISTINCT  200
47
```

функциональные индексы

```
test=# analyze my_table;
ANALYZE

test=# explain select * from my_table where md5(i::text) =
'202cb962ac59075b964b07152d234b70';
                                  QUERY PLAN
-------------------------------------------------------------------------------
 Index Scan using my_table_md5_idx on my_table  (cost=0.42..8.44 rows=1 width=4)
   Index Cond: (md5((i)::text) = '202cb962ac59075b964b07152d234b70'::text)
(2 rows)
```

функциональные индексы

```
test=# select * from pg_statistic where starelid = 'my_table_md5_idx'::regclass;
-[ RECORD 1 ]--------------------------
starelid    | 204837
staattnum   | 1
stainherit  | f
stanullfrac | 0
stawidth    | 36
stadistinct | -1
stakind1    | 2
stakind2    | 3
...
```

— функциональные индексы требуют analyze после создания

rows у функций

```
select * from
each('"201"=>"1059","496"=>"3","497"=>"5","498"=>"5245","499"=>"5254","549"=>"5697","566"
=>"5828","575"=>"5929","578"=>"45","1459"=>"16179"'::hstore) p
left join category_params_values v on v.param_id = p.key::int and v.value = p.value
where p.key in ('496','501','513','514','515','516','497','502','525','526','530','745')
```

```
Hash Left Join  (cost=139.37..149.12 rows=60 width=148)
  Hash Cond: (((p.key)::integer = v.param_id) AND (p.value = (v.value)::text))
  ->  Function Scan on each p  (cost=0.00..2.80 rows=60 width=64)
        Filter: (key = ANY
('{496,501,513,514,515,516,497,502,525,526,530,745}'::text[]))
  ->  Hash  (cost=121.91..121.91 rows=10910 width=84)
        ->  Seq Scan on category_params_values v  (cost=0.00..121.91 rows=10910
width=84)
```

rows у функций

```
CREATE OR REPLACE FUNCTION each(
    IN hs hstore,
    OUT key text,
    OUT value text)
  RETURNS SETOF record AS
'$libdir/hstore', 'hstore_each'
  LANGUAGE c IMMUTABLE STRICT
  COST 1
  ROWS 1000;
```

=>

```
CREATE OR REPLACE FUNCTION each_r10(
    IN hs hstore,
    OUT key text,
    OUT value text)
  RETURNS SETOF record AS
'$libdir/hstore', 'hstore_each'
  LANGUAGE c IMMUTABLE STRICT
  COST 1
  ROWS 10;
```

rows у функций

```
select * from
each_r10('"201"=>"1059","496"=>"3","497"=>"5","498"=>"5245","499"=>"5254","549"=>"5697","
566"=>"5828","575"=>"5929","578"=>"45","1459"=>"16179"'::hstore) p
left join category_params_values v on v.param_id = p.key::int and v.value = p.value
where p.key in ('496','501','513','514','515','516','497','502','525','526','530','745')
```

```
Nested Loop Left Join  (cost=0.00..28.27 rows=7 width=148)
  ->  Function Scan on each_r10 p  (cost=0.00..0.03 rows=7 width=64)
        Filter: (key = ANY
('{496,501,513,514,515,516,497,502,525,526,530,745}'::text[]))
  ->  Index Scan using category_params_values2_param_value_idx on
category_params_values v  (cost=0.00..4.03 rows=1 width=84)
        Index Cond: ((param_id = (p.key)::integer) AND ((value)::text = p.value))
```

Спасибо!