Avito

Standby in production

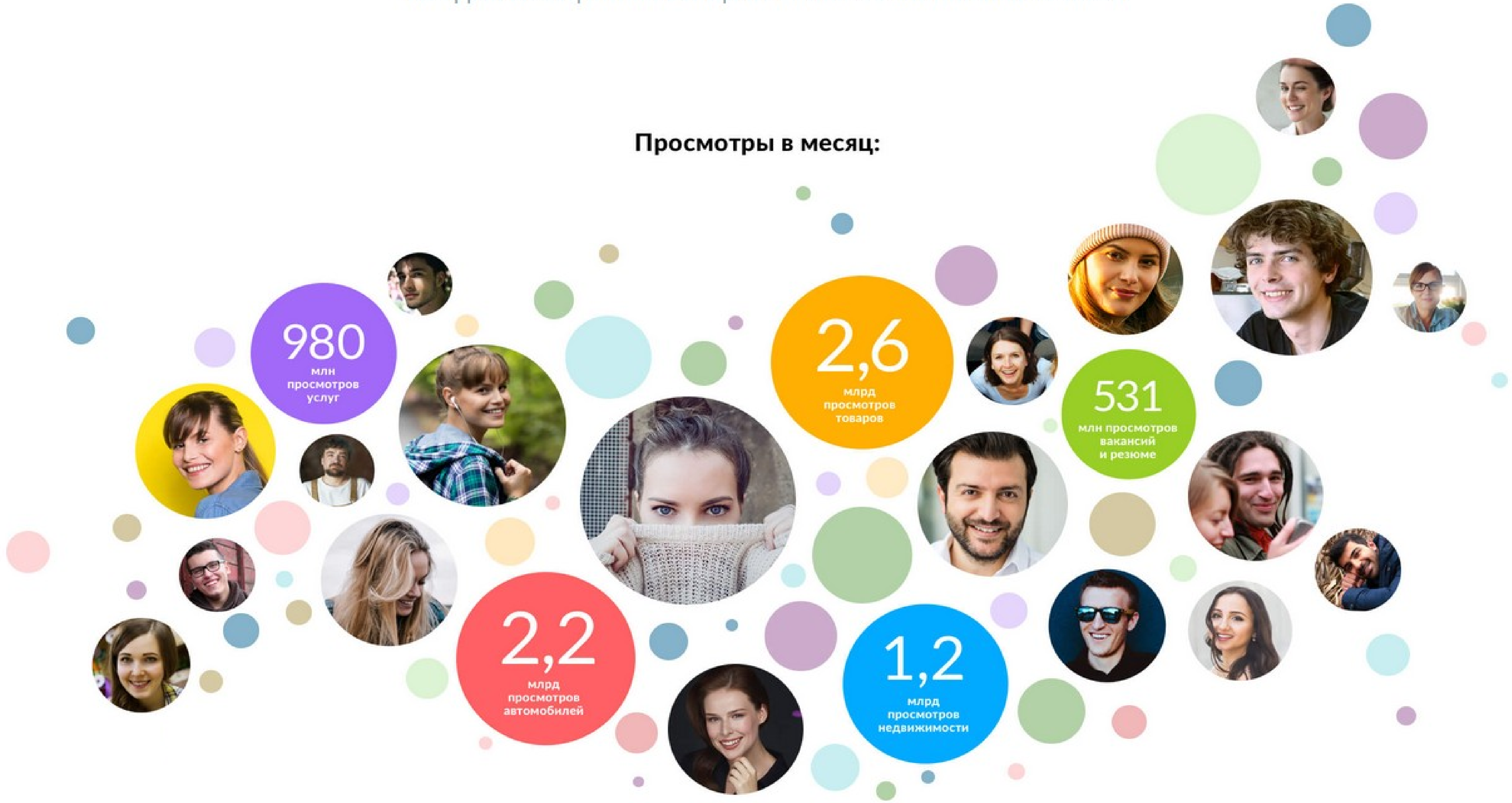
Konstantin Evteev

Moscow 2019

Avito — это аудитория размером с целую страну

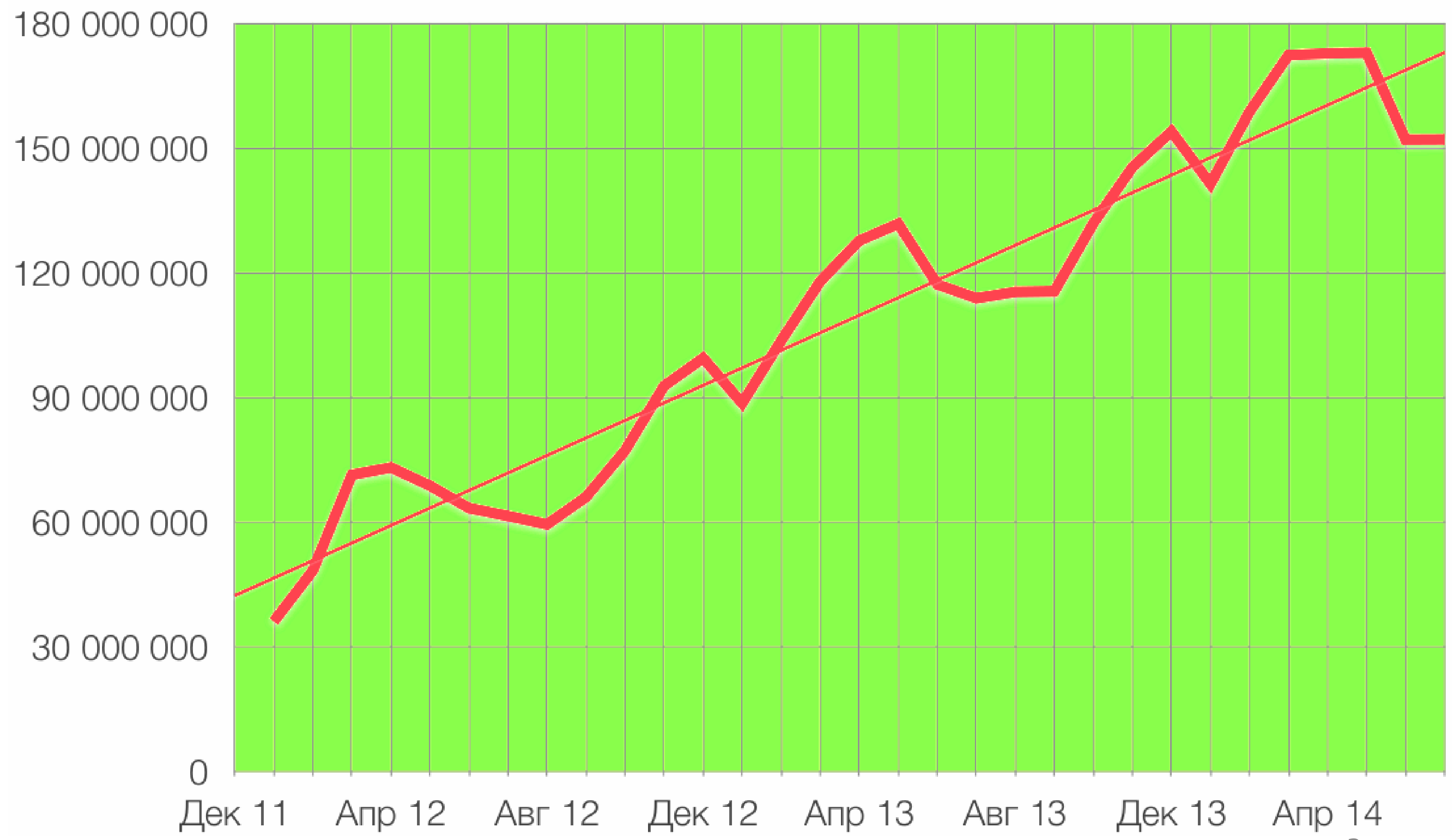
Каждый месяц Avito посещают более 32 миллионов человек.

Просмотры в месяц:



данные за февраль 2018 г.

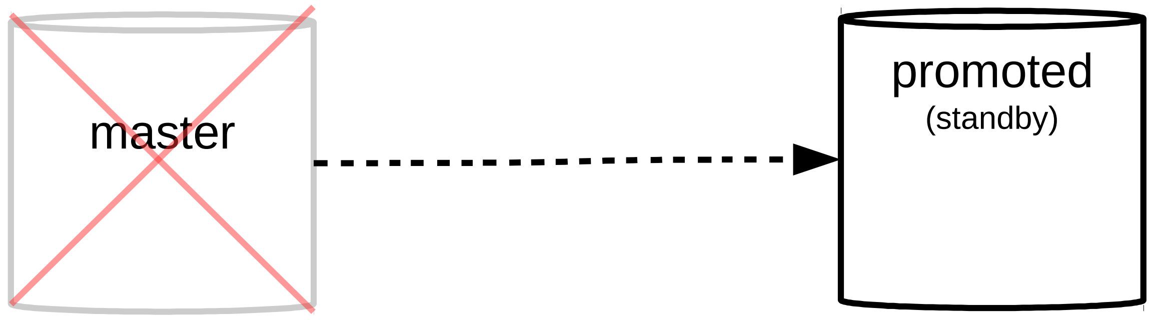
pageviews



Avito

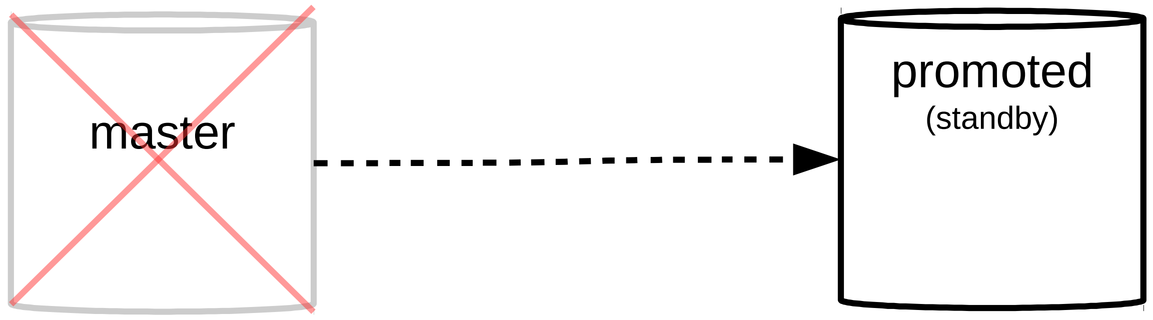
Standby

1 High Availability

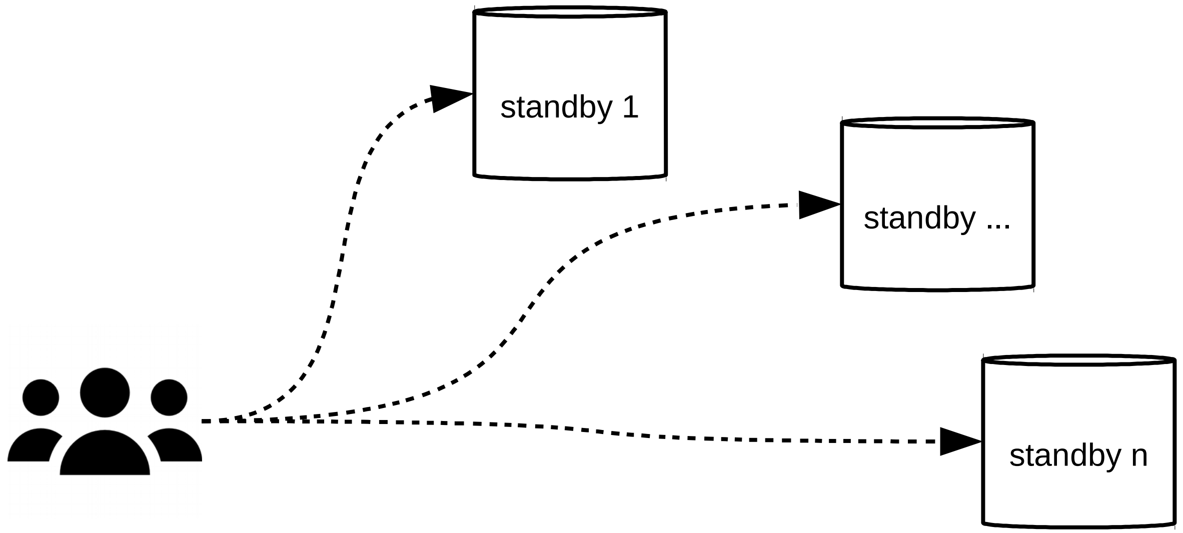


Standby

1 High Availability



2 Scaling



History

2000: Rep script

History

2000: Rep script

2001: PostgreSQL 7.1: write-ahead log

History

2000: Rep script

2001: PostgreSQL 7.1: write-ahead log

2004: Slony

History

2000: Rep script

2001: PostgreSQL 7.1: write-ahead log

2004: Slony

2005: PostgreSQL 8.0: point-in-time recovery

History

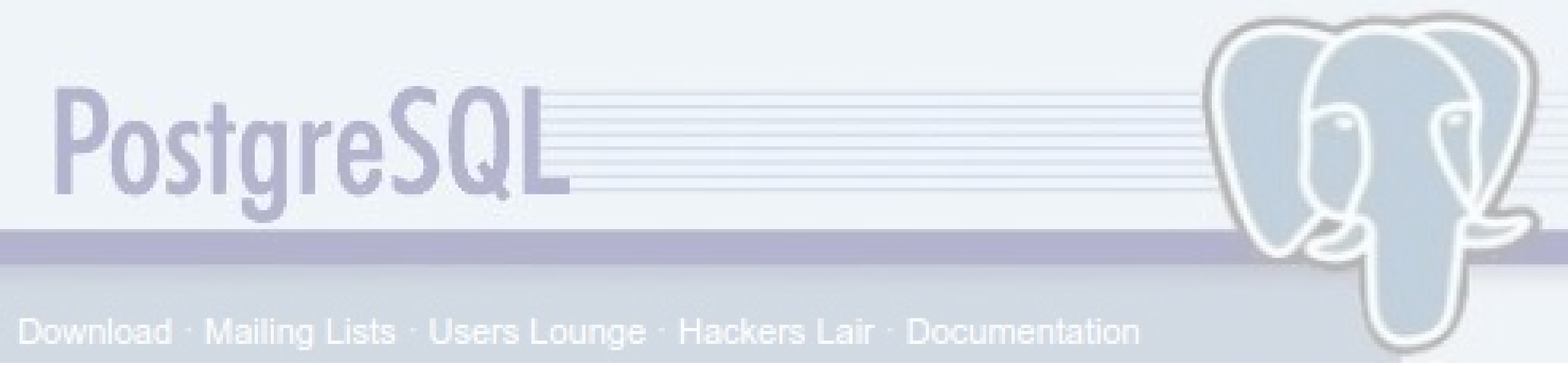
2000: Rep script

2001: PostgreSQL 7.1: write-ahead log

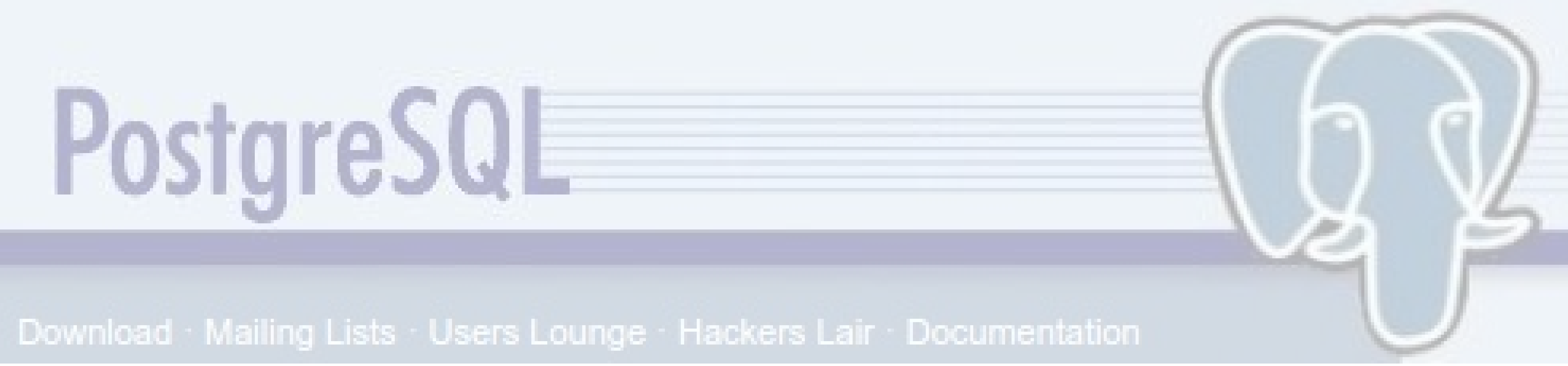
2004: Slony

2005: PostgreSQL 8.0: point-in-time recovery

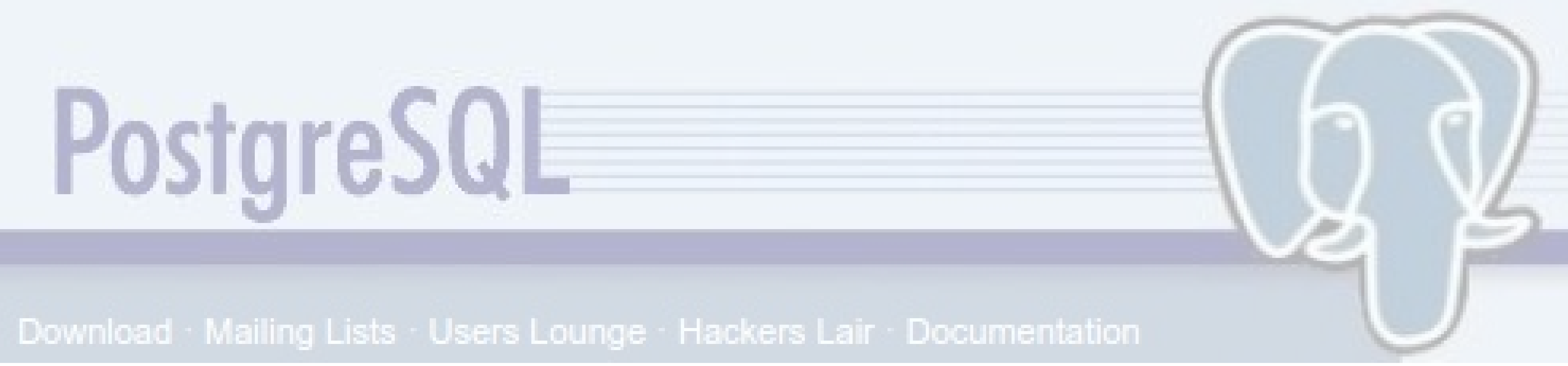
2008: 8.3 standby


PostgreSQL
Download · Mailing Lists · Users Lounge · Hackers Lair · Documentation

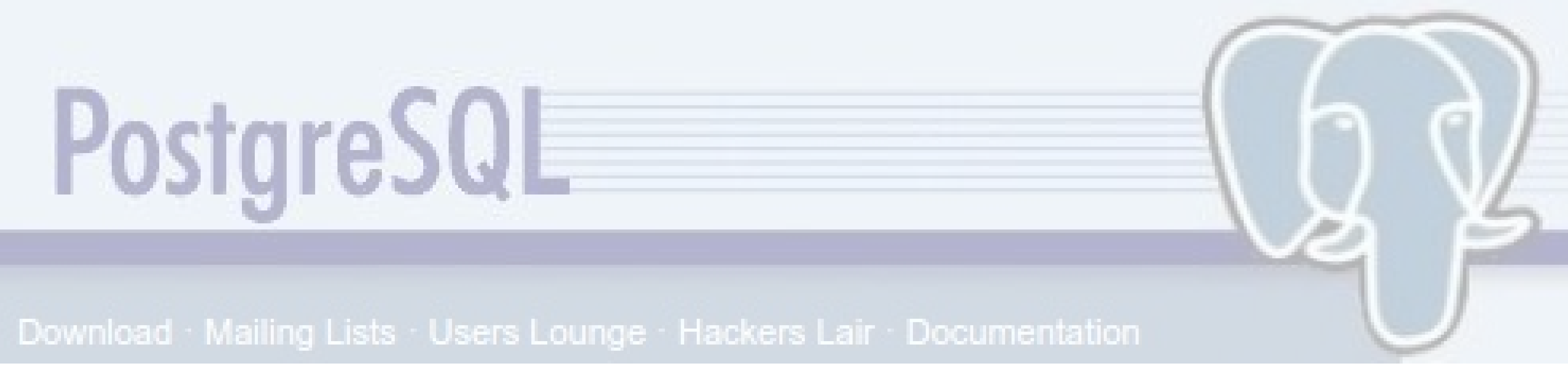
1. 2010: 9.0: hot standby, streaming replication


PostgreSQL
Download · Mailing Lists · Users Lounge · Hackers Lair · Documentation

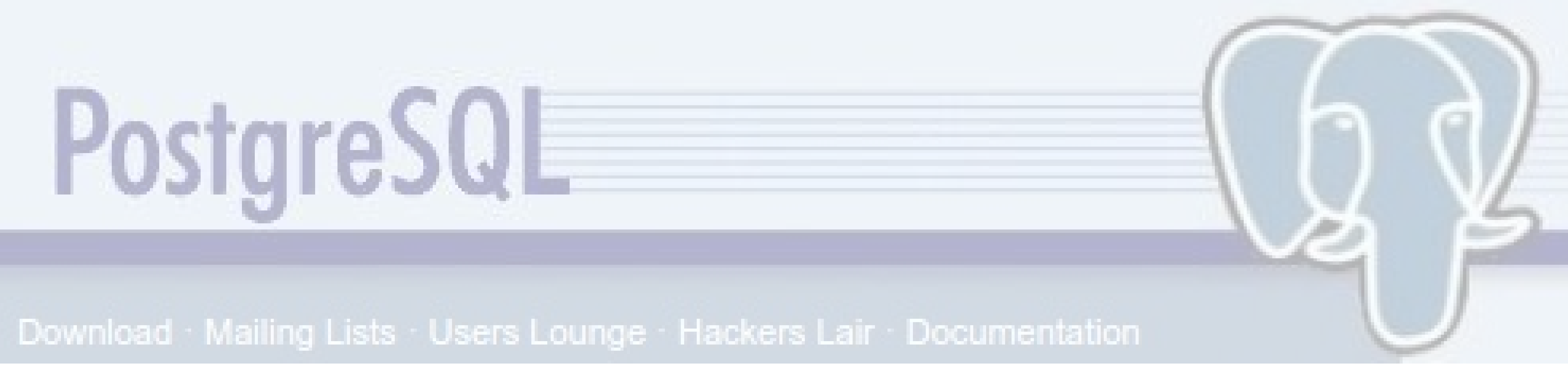
1. 2010: 9.0: hot standby, streaming replication
2. 2011: 9.1: synchronous replication


PostgreSQL
Download · Mailing Lists · Users Lounge · Hackers Lair · Documentation

1. 2010: 9.0: hot standby, streaming replication
2. 2011: 9.1: synchronous replication
3. 2013: 9.3: sb can follow timeline switch

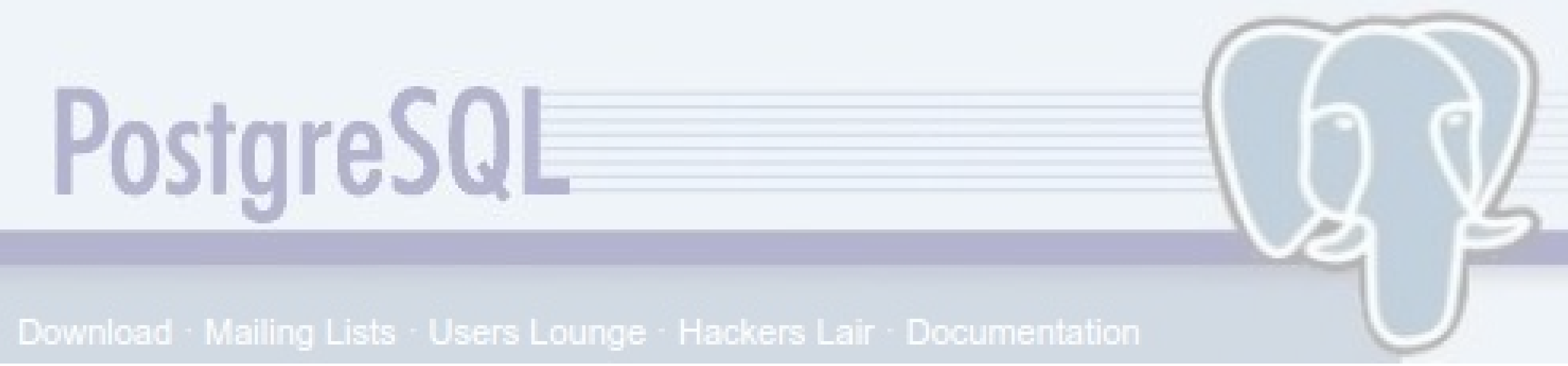

PostgreSQL
Download · Mailing Lists · Users Lounge · Hackers Lair · Documentation

1. 2010: 9.0: hot standby, streaming replication
2. 2011: 9.1: synchronous replication
3. 2013: 9.3: sb can follow timeline switch
4. 2014: 9.4: replication slots, logical decoding


PostgreSQL
Download · Mailing Lists · Users Lounge · Hackers Lair · Documentation

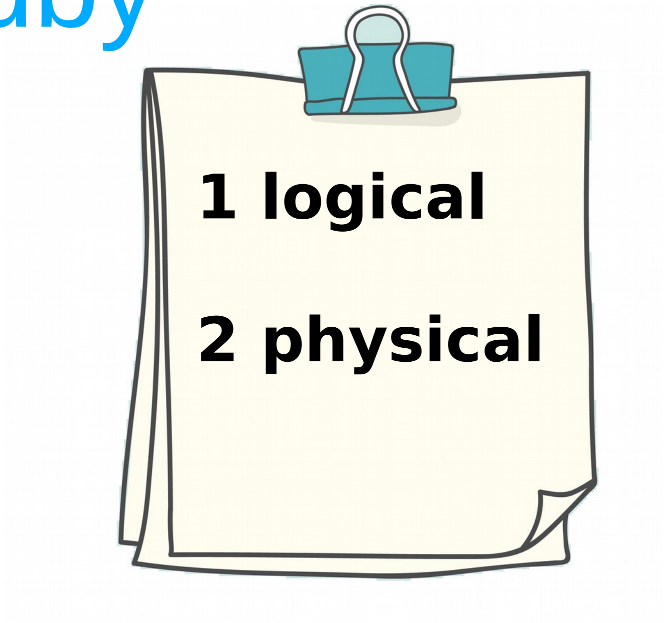
1. 2010: 9.0: hot standby, streaming replication
2. 2011: 9.1: synchronous replication
3. 2013: 9.3: sb can follow timeline switch
4. 2014: 9.4: replication slots, logical decoding
5. 2016: 9.6 multiple synchronous standbys, remote_apply

Avito

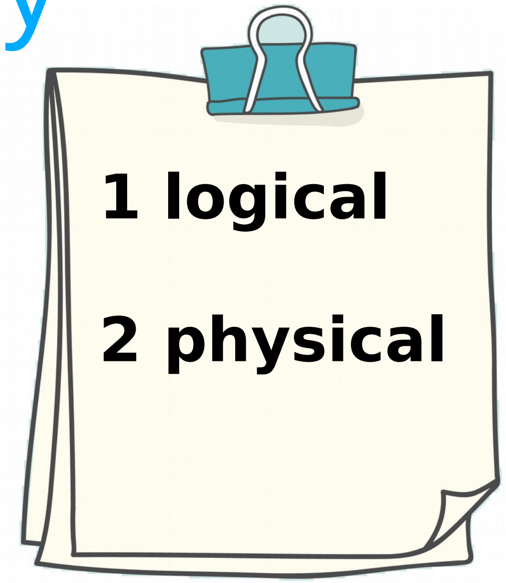

PostgreSQL
Download · Mailing Lists · Users Lounge · Hackers Lair · Documentation

1. 2010: 9.0: hot standby, streaming replication
2. 2011: 9.1: synchronous replication
3. 2013: 9.3: sb can follow timeline switch
4. 2014: 9.4: replication slots, logical decoding
5. 2016: 9.6 multiple synchronous standbys, remote_apply
6. 2017: 10: logical replication

Standby

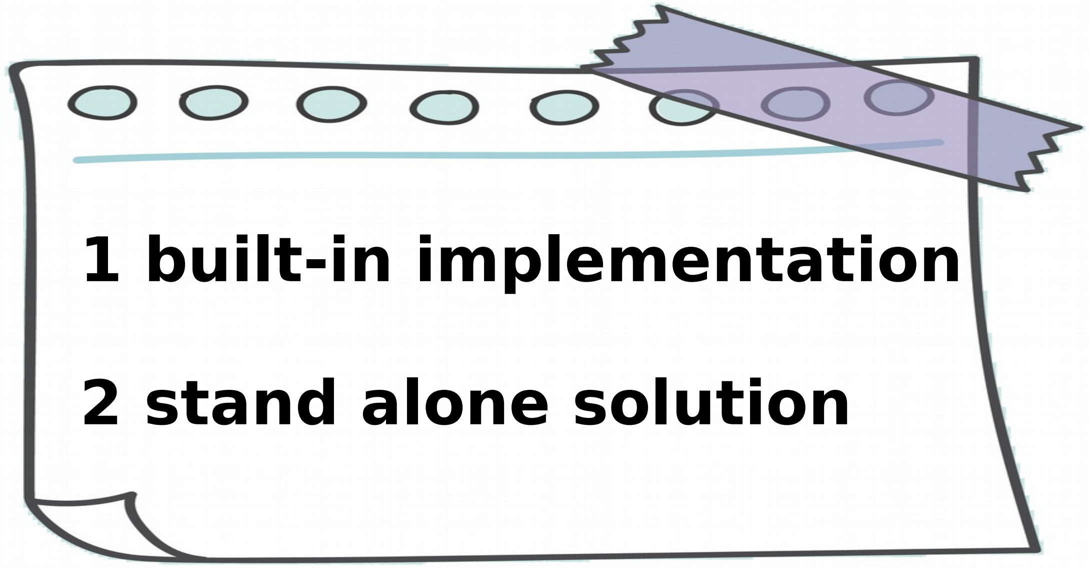


Standby



1 logical

2 physical



1 built-in implementation

2 stand alone solution

Avito

Standby

1 logical

2 physical

1 built-in implementation

2 stand alone solution

1 read only

2 not only read only

Standby

1 logical

2 physical

1 built-in implementation

2 stand alone solution

1 read only

2 not only read only

sync

async

...

Standby

1 logical

2 physical

1 built-in implementation

2 stand alone solution

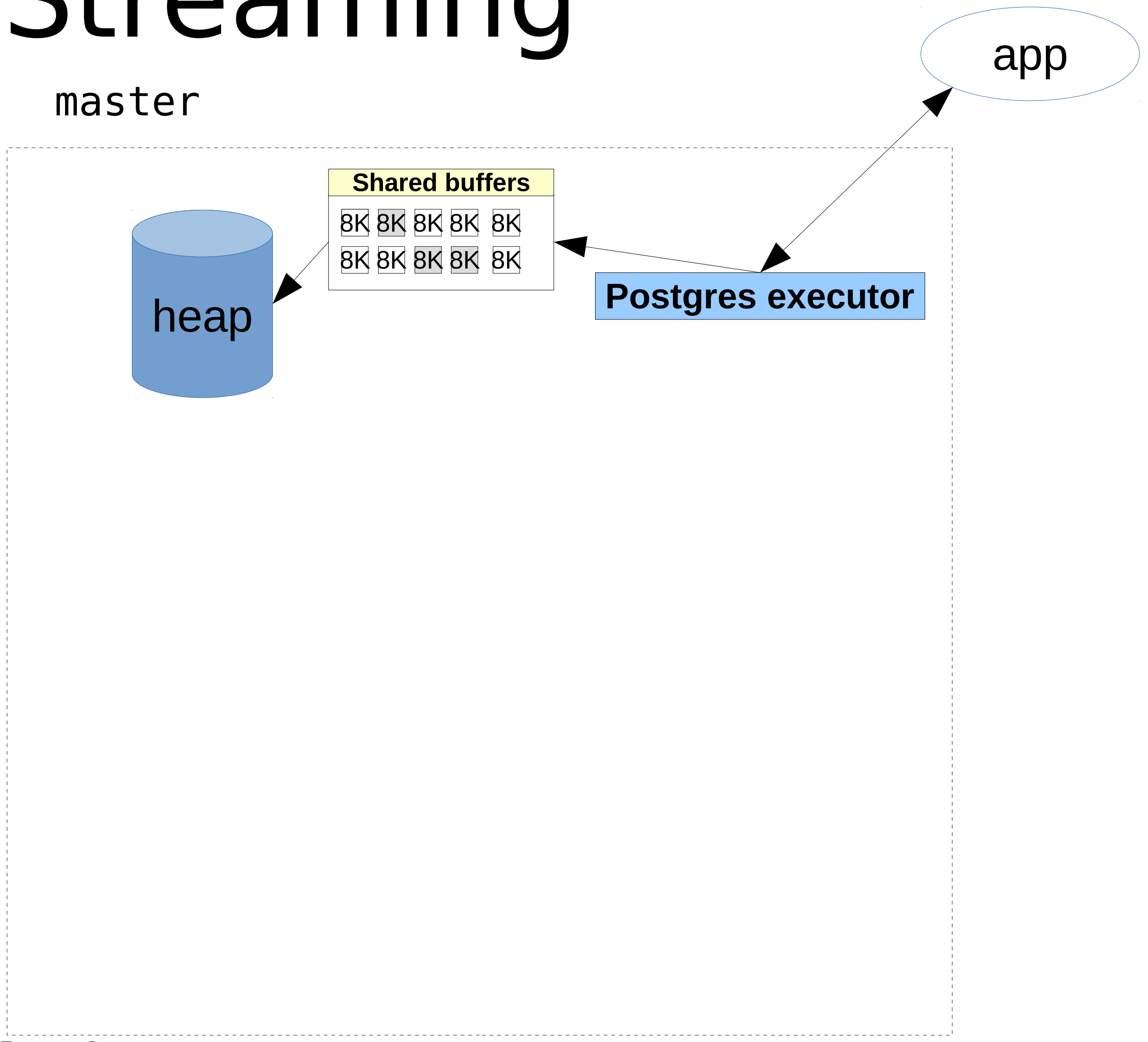
1 read only

2 not only read only

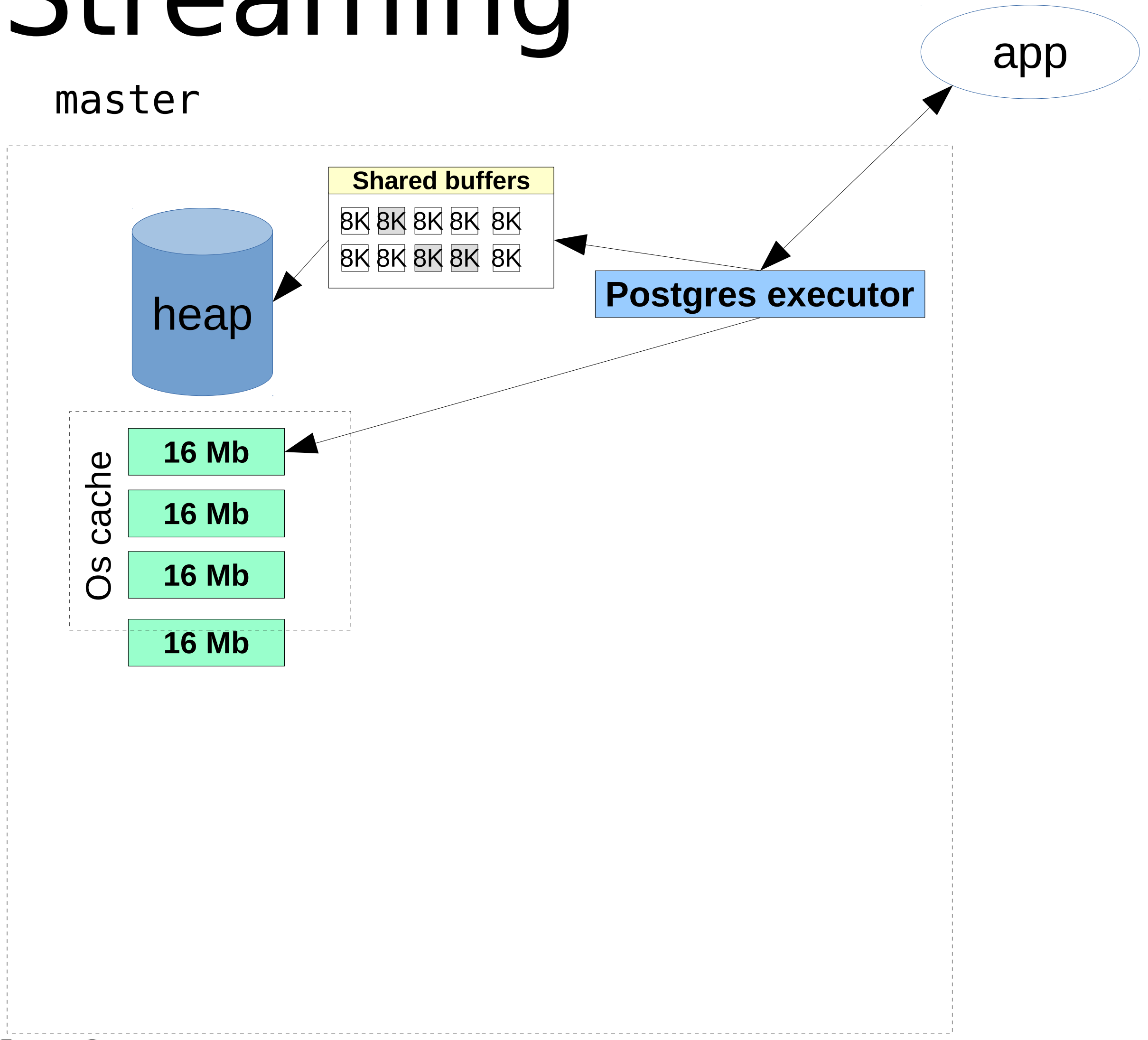
sync

async

Streaming

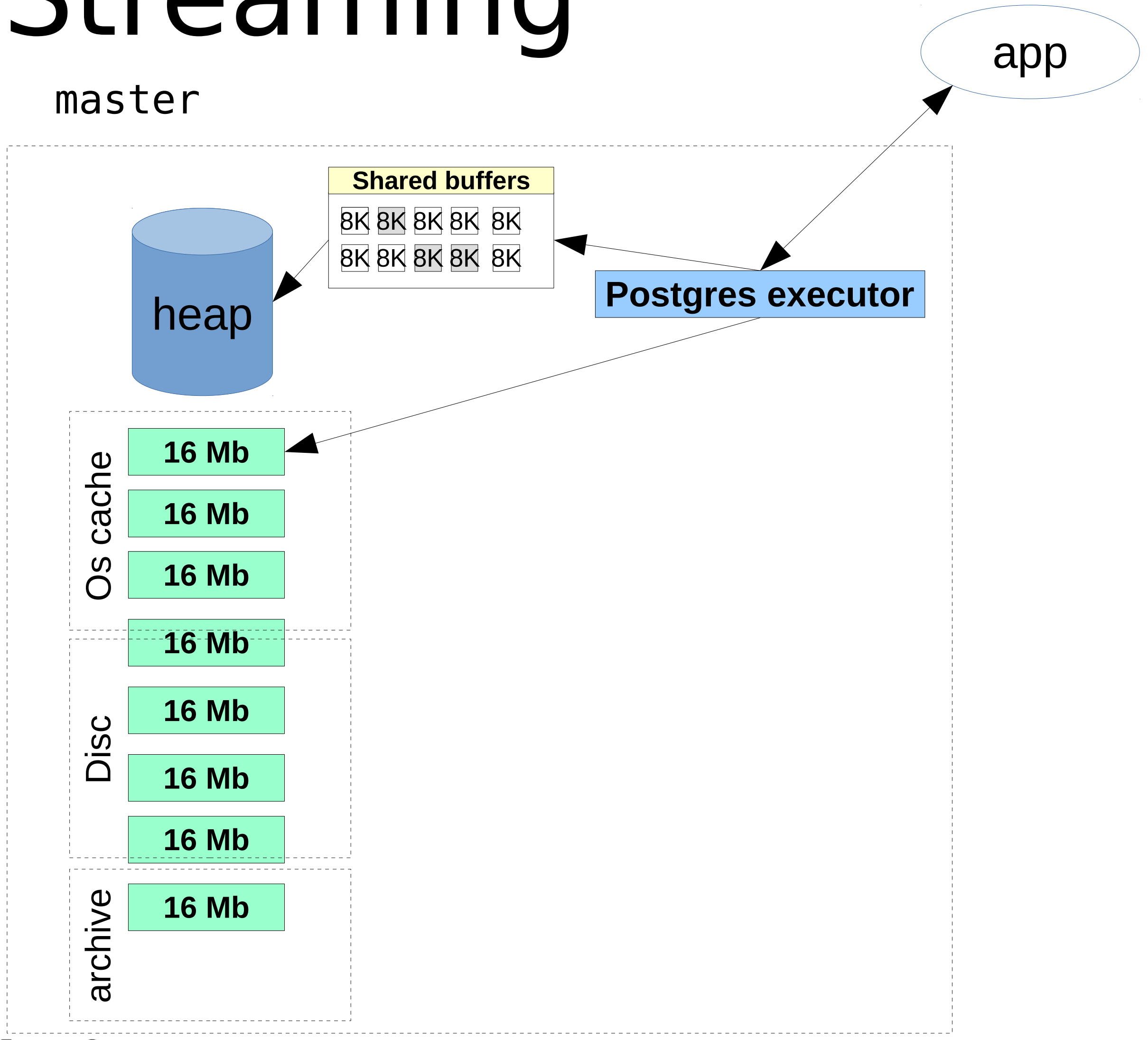

master
app
Shared buffers
8K 8K 8K 8K 8K
8K 8K 8K 8K 8K
heap
Postgres executor

Streaming



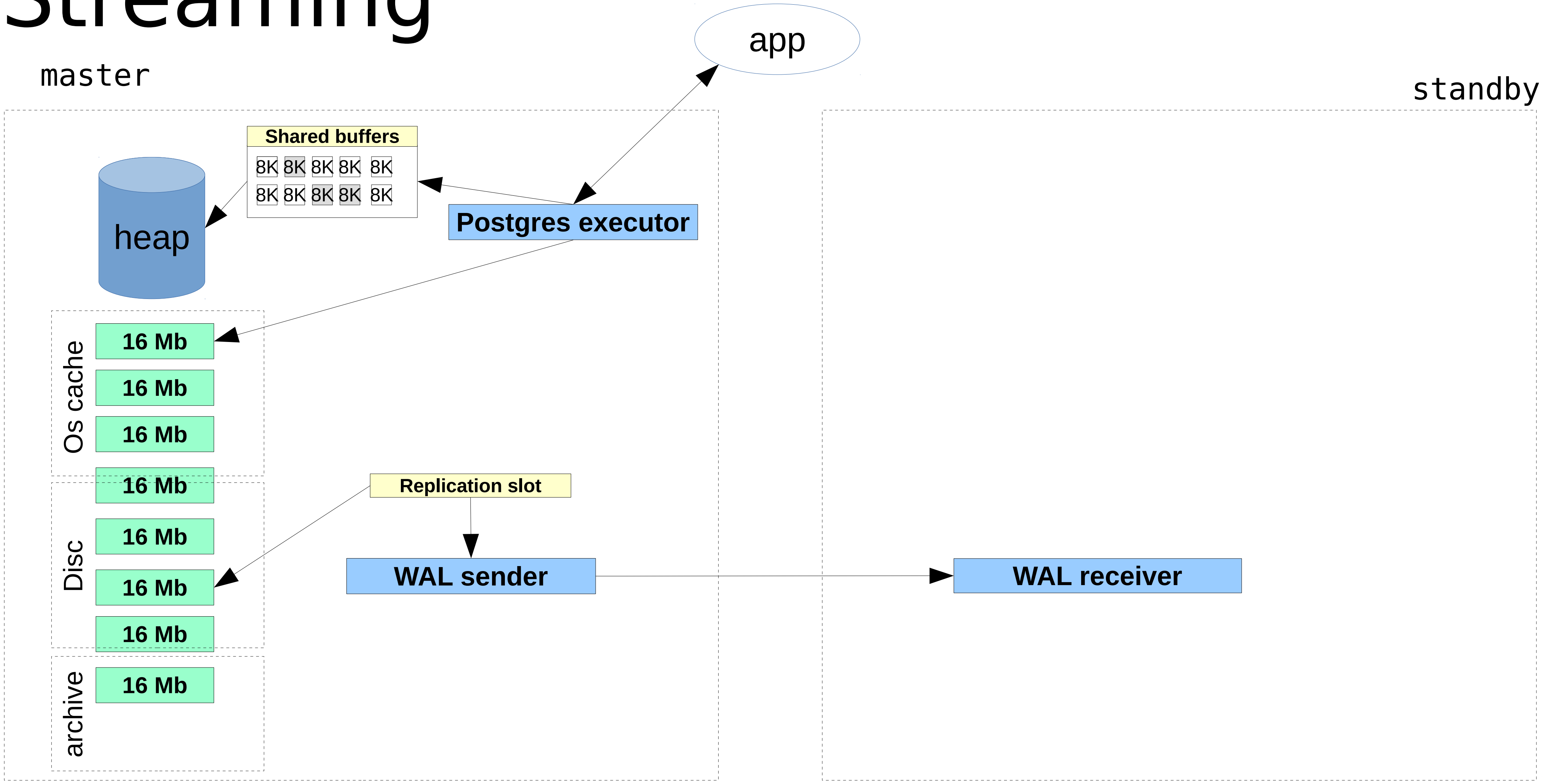
master
app
Shared buffers
8K 8K 8K 8K 8K
8K 8K 8K 8K 8K
heap
Postgres executor
Os cache
16 Mb
16 Mb
16 Mb
16 Mb

Streaming



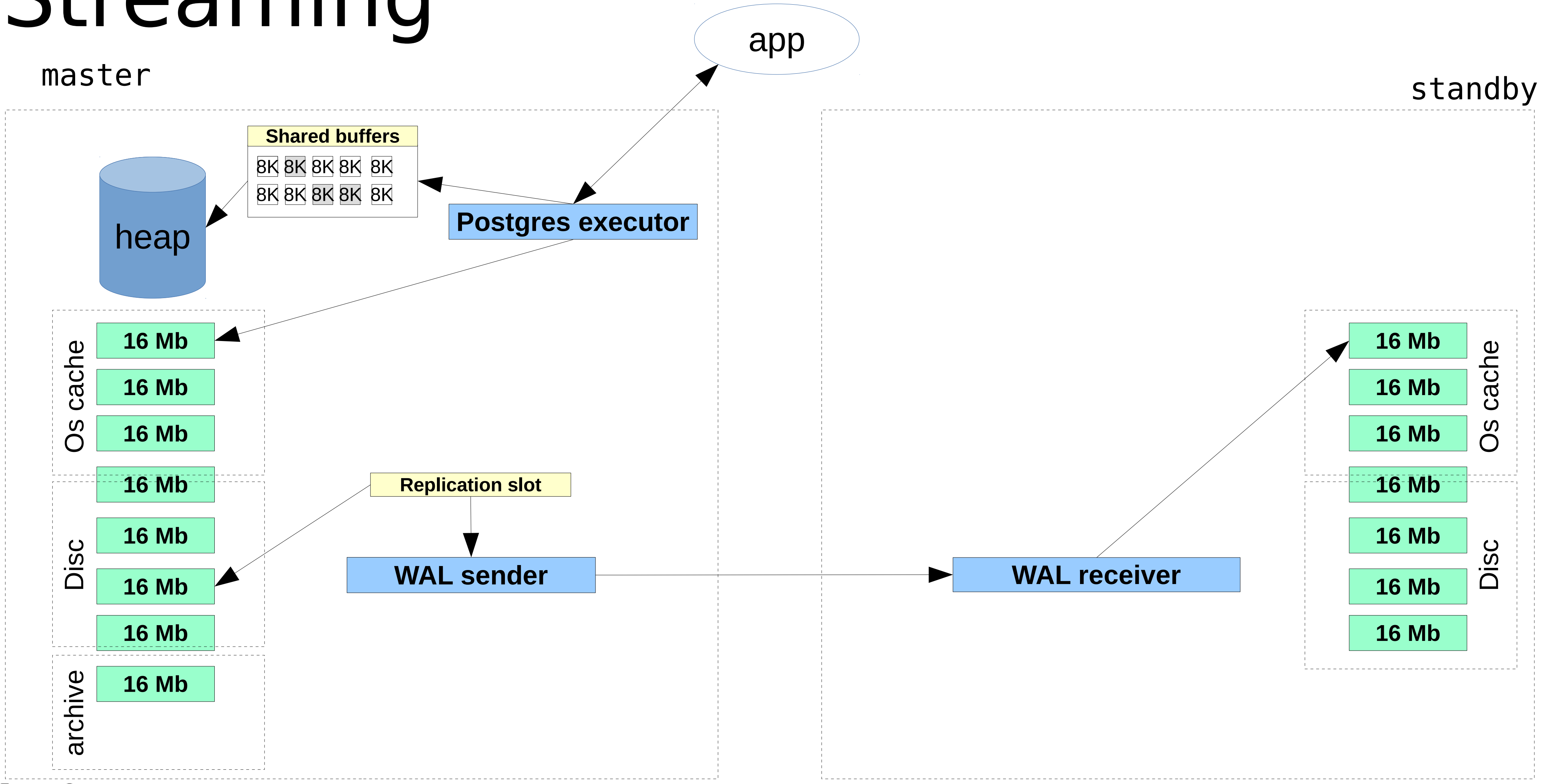
Avito

Streaming



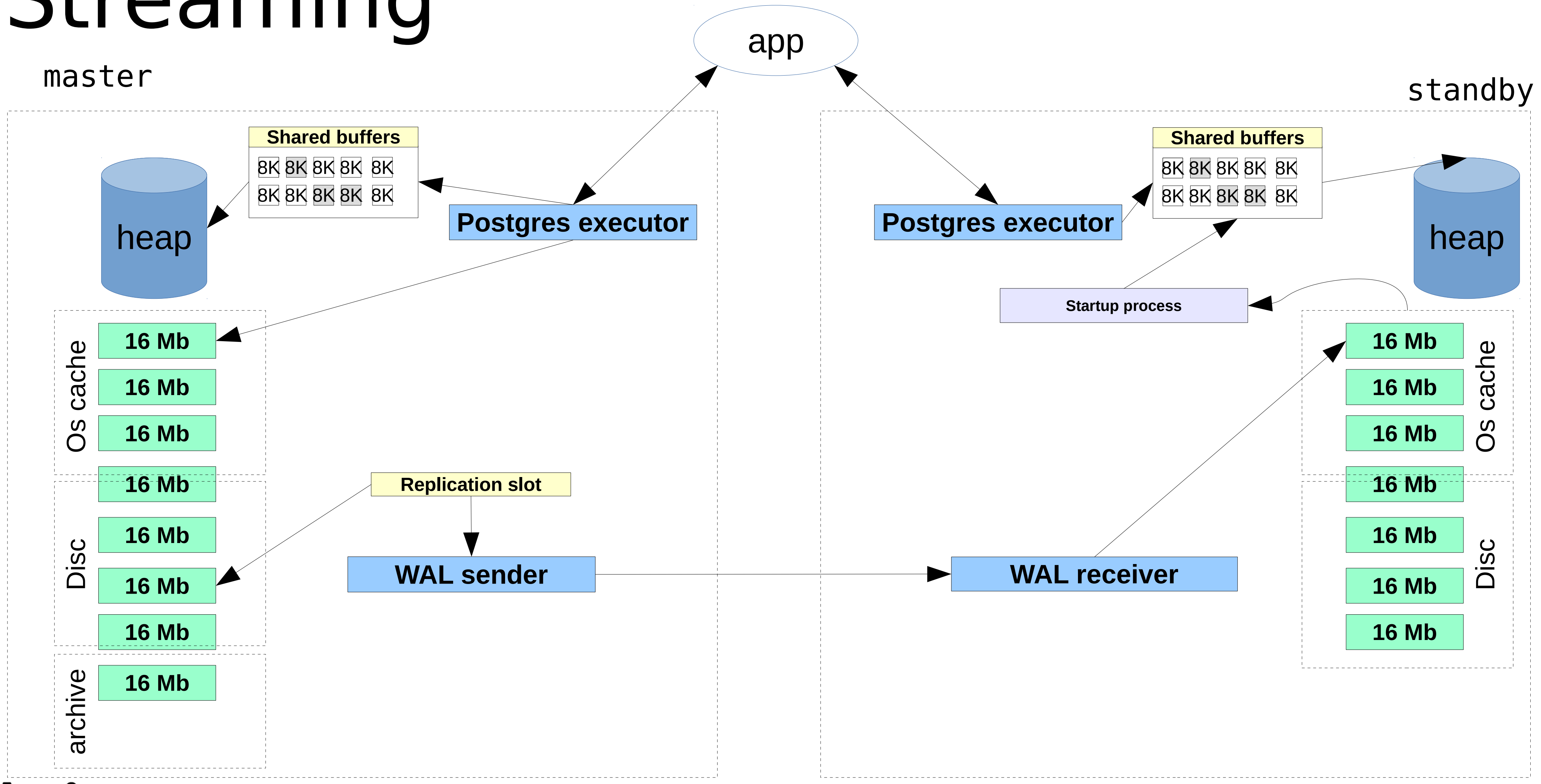
app
master
standby
Shared buffers
8K 8K 8K 8K 8K
8K 8K 8K 8K 8K
heap
Postgres executor
Os cache
16 Mb
16 Mb
16 Mb
16 Mb
Disc
16 Mb
16 Mb
16 Mb
archive
16 Mb
Replication slot
WAL sender
WAL receiver

Streaming



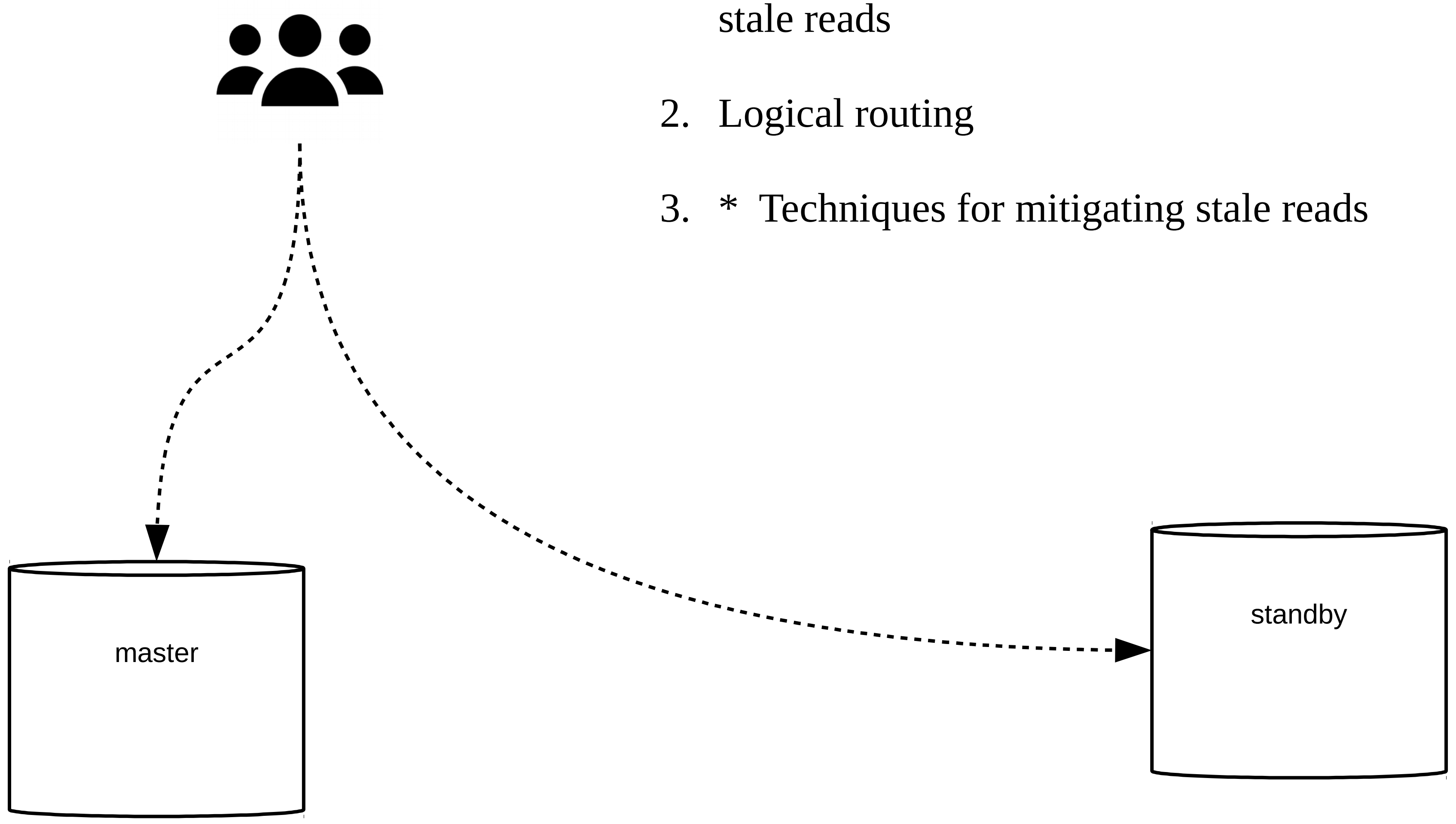
Avito

Streaming



Avito

Scaling reads

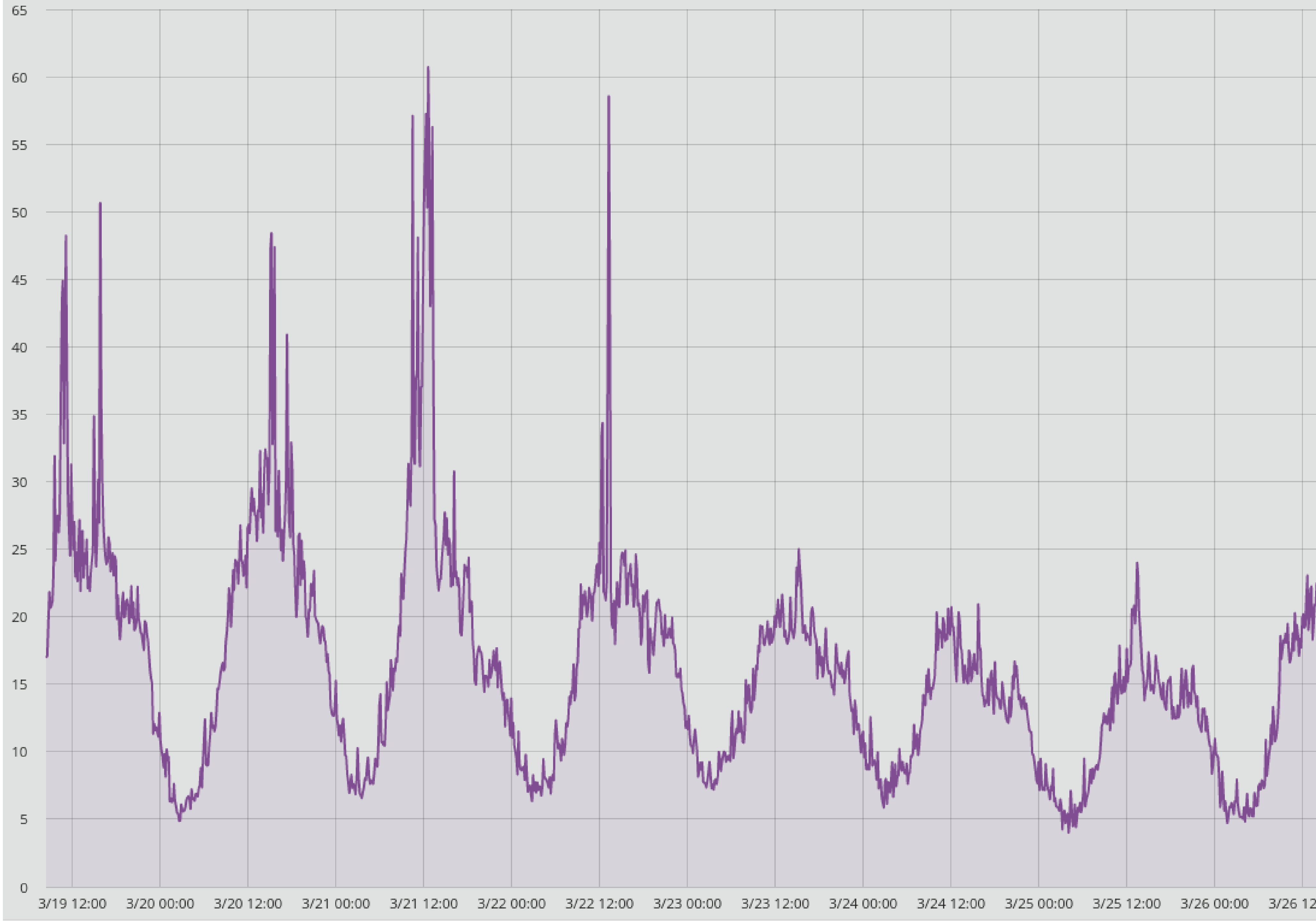


1. Using read replicas without any techniques for mitigating stale reads

2. Logical routing

3. * Techniques for mitigating stale reads

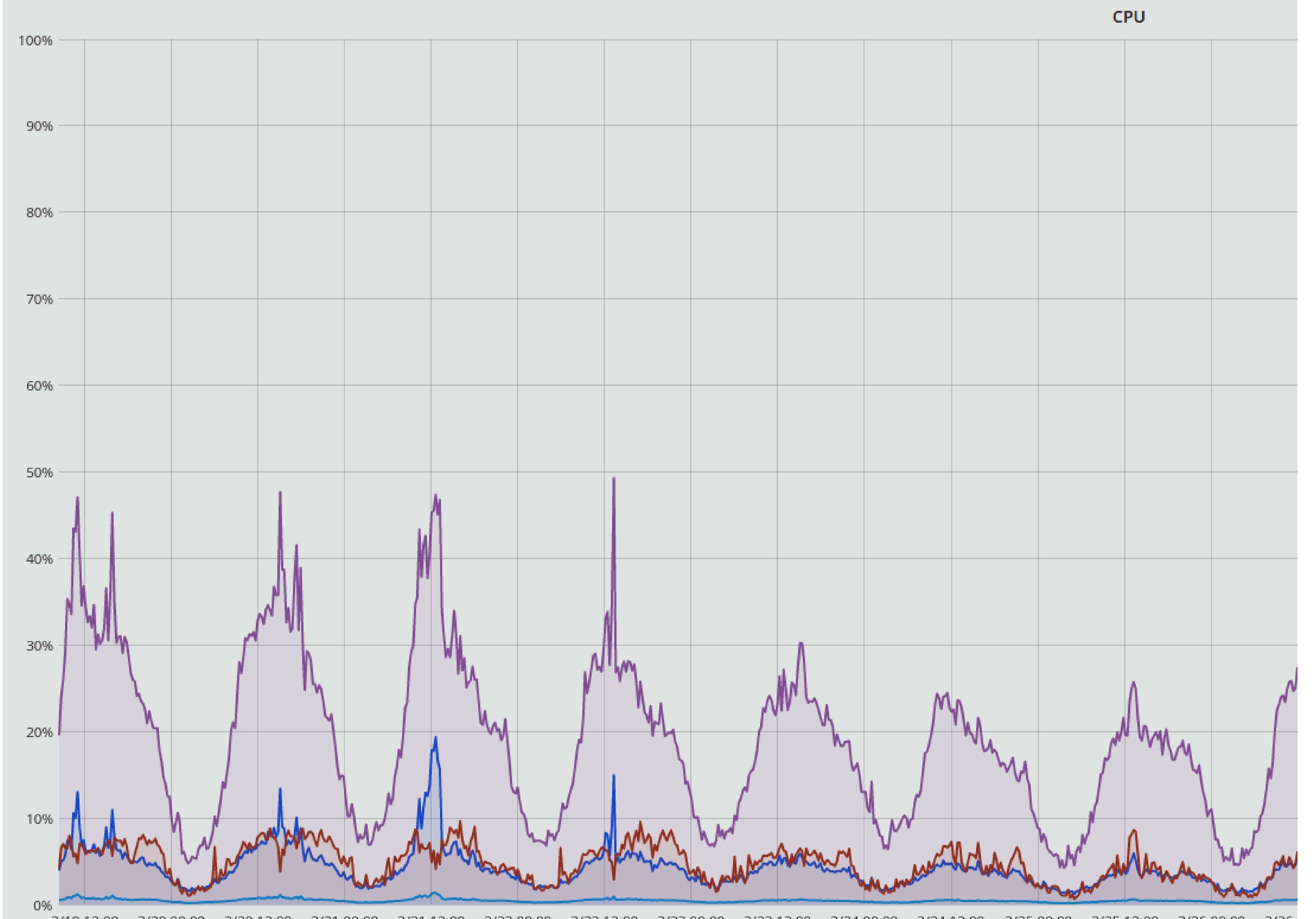
Load Average



** 28 physical cores*

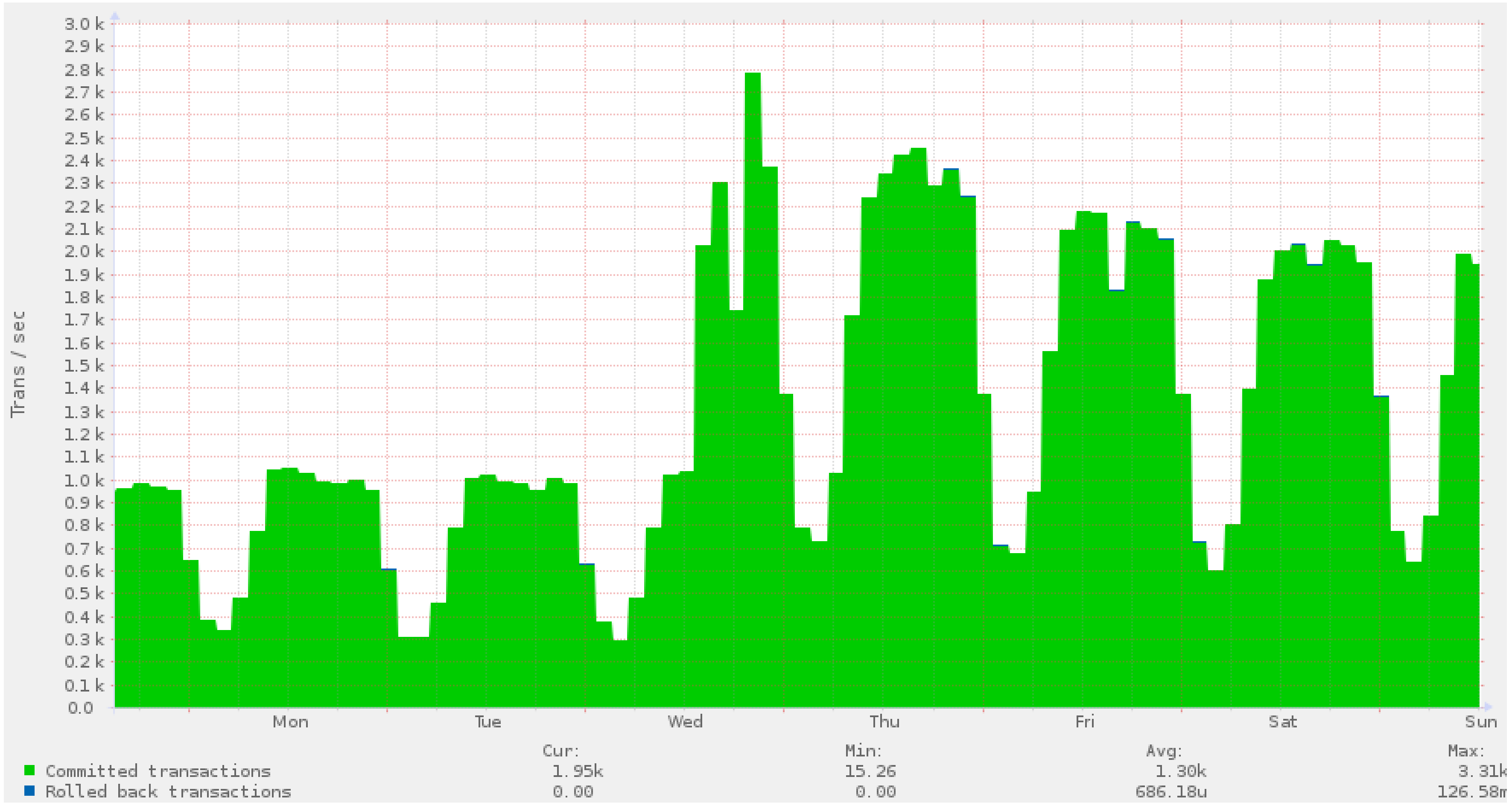
Avito

CPU



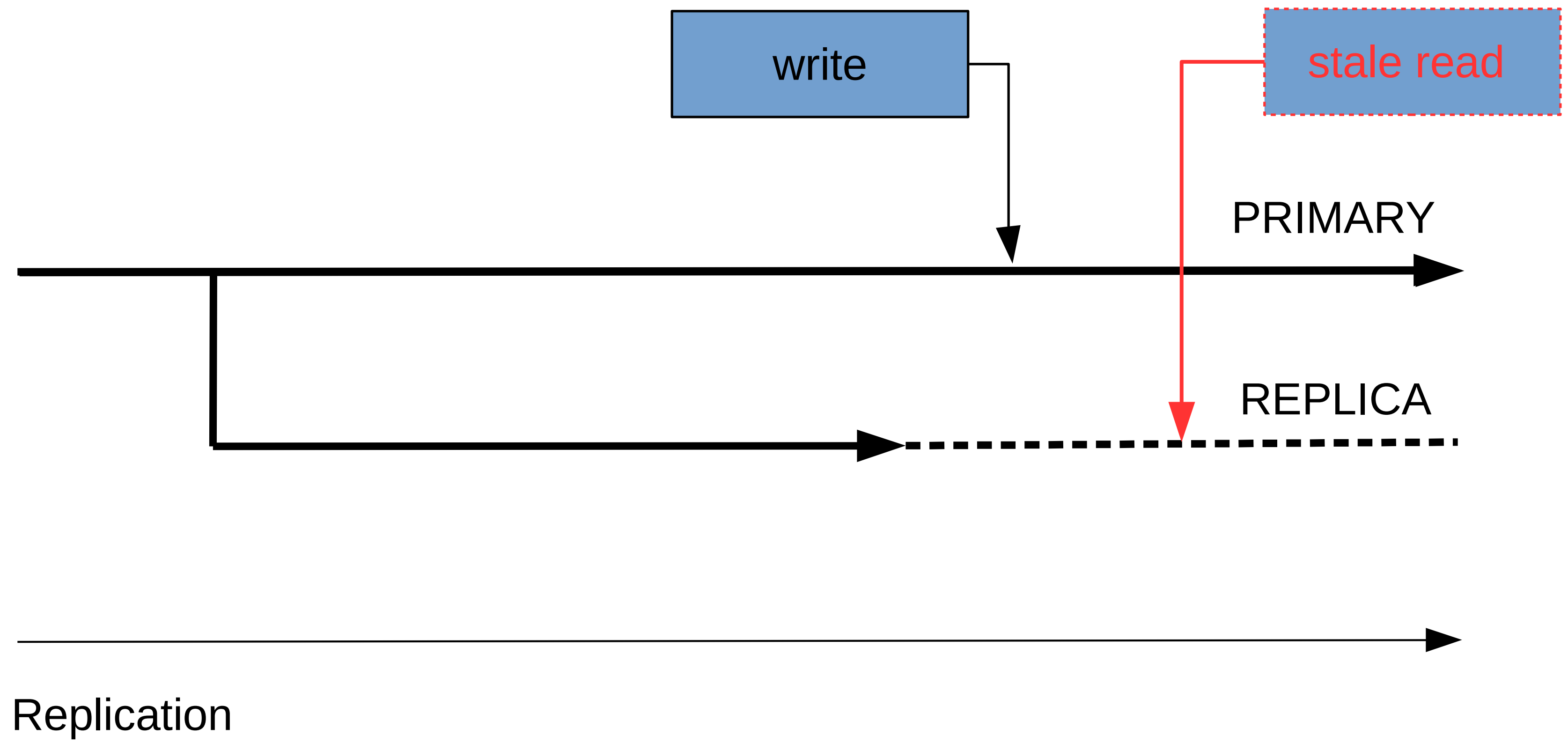
* *28 physical cores*

TPS



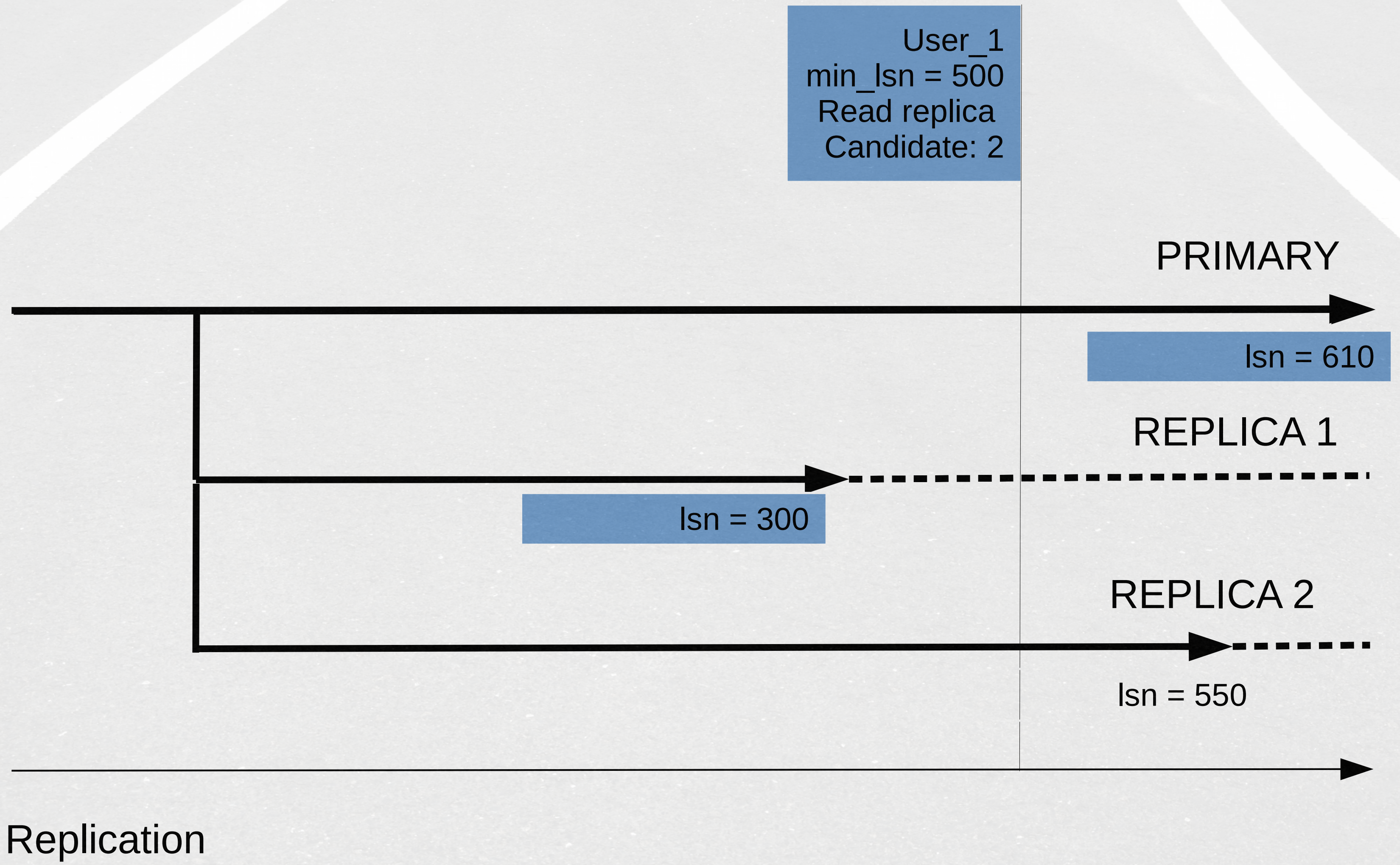
	Cur:	Min:	Avg:	Max:
🟩 Committed transactions	1.95k	15.26	1.30k	3.31k
🟦 Rolled back transactions	0.00	0.00	686.18u	126.58m

Stale Reads



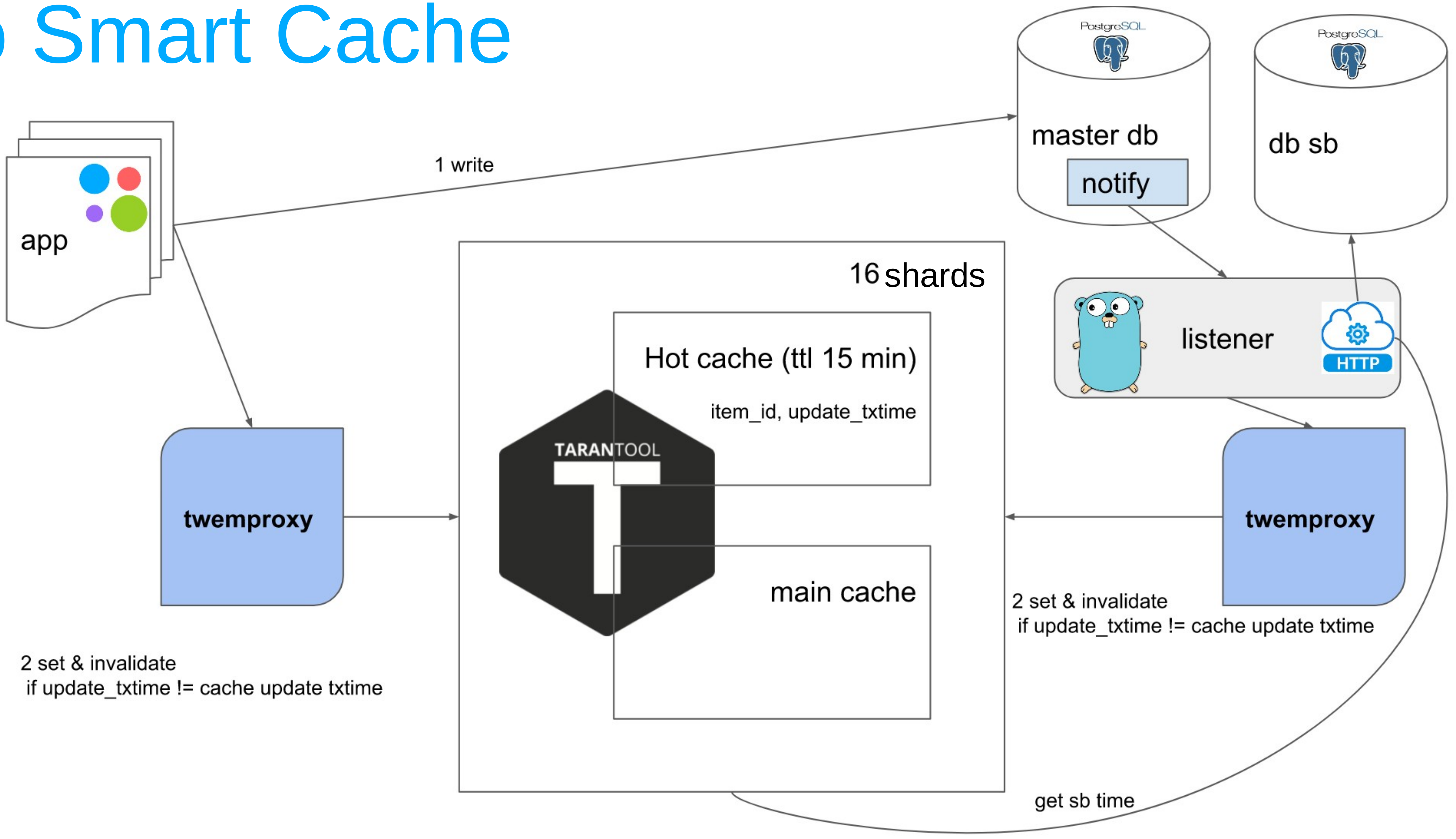
https://brandur.org/postgres-reads
https://about.gitlab.com/2017/10/02/scaling-the-gitlab-database/#sticky-connections

Routing



https://brandur.org/postgres-reads
https://about.gitlab.com/2017/10/02/scaling-the-gitlab-database/#sticky-connections

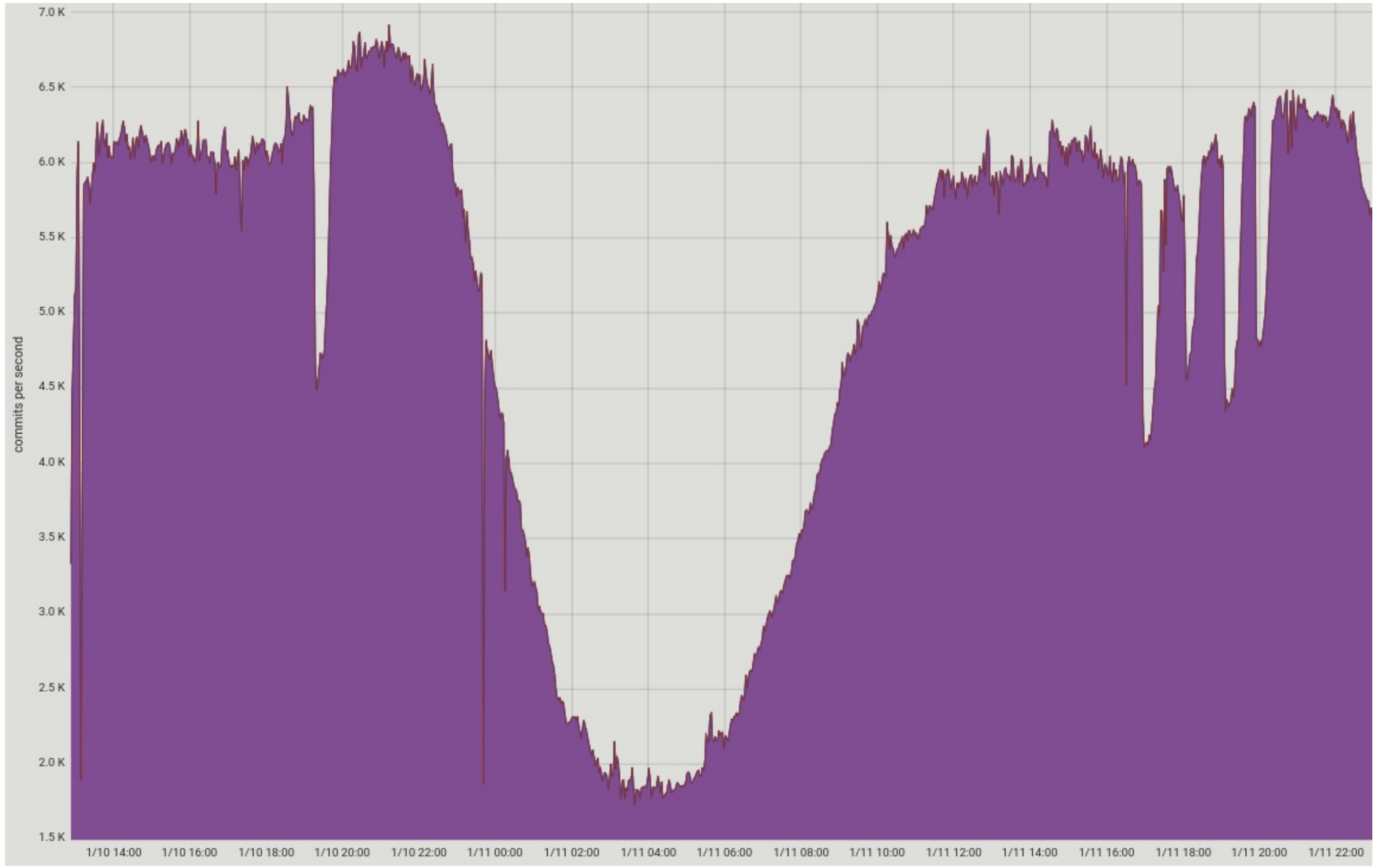
Avito Smart Cache



Two levels of cache

```
app.get_item(key):
        data = main_cache.get(key) // try to get data from cache

        if found then return

        hot_cache.get(key)          // get data from hot level of cache

        if found then               // if date was recently changed then route to master
            data =db_master.get_item(key)
            main_cache.set(key, data) // ttl 1 hour
        else if sb too old then     // if standby is falling behind – route to master
                data =db_master.get_item(key)
            main_cache.set(key, data) // ttl 24 hours
        else                       // in other cases we can rout to standby
            data =db_slave.get_item(key)
            main_cache.set(key, data) // ttl 24 hours
        end if
```

Everything seems fine but ...



Caveats

(1) Deadlock on standby

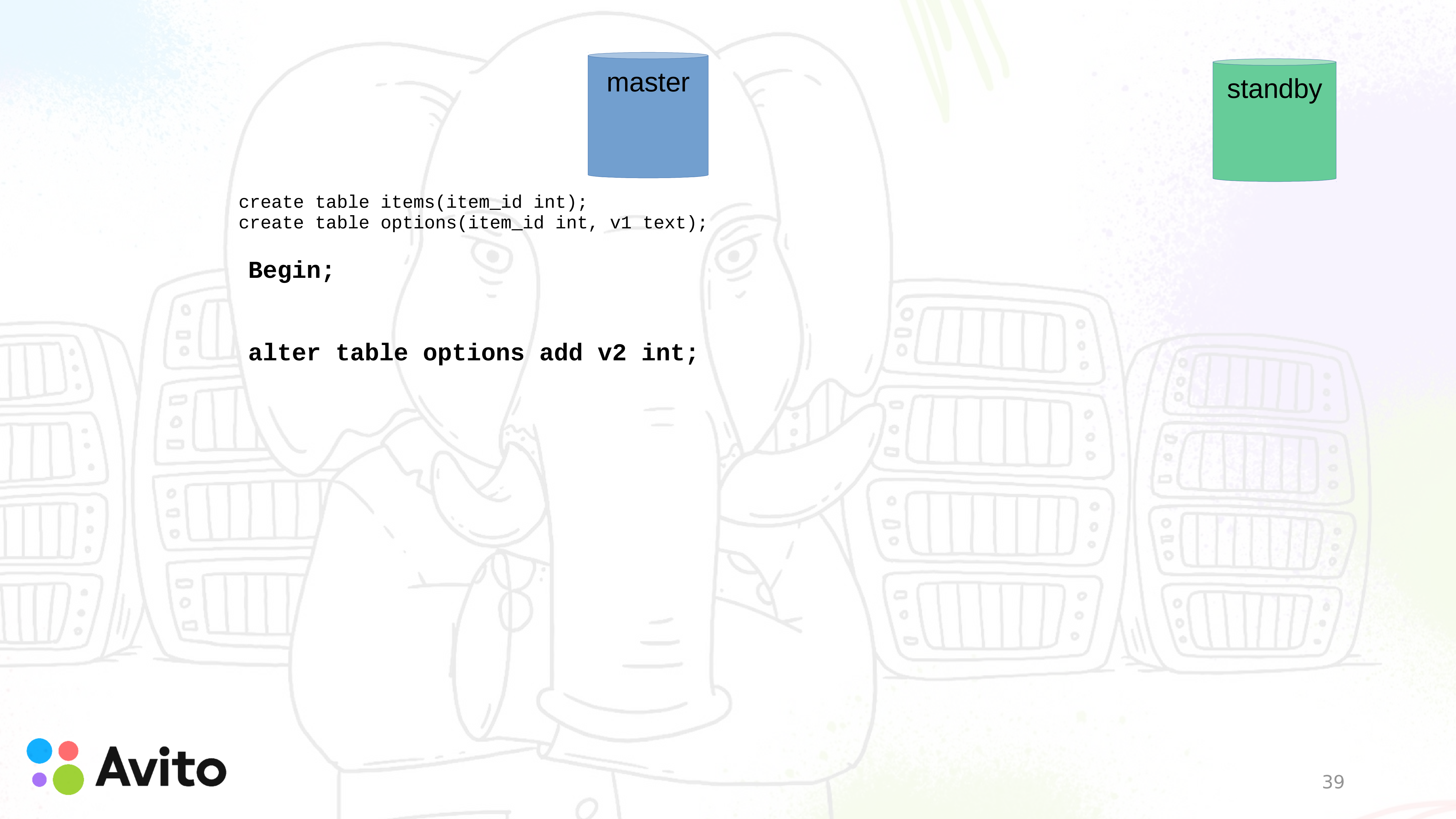
(2) DDL (statement_timout and deadlock_timeout)

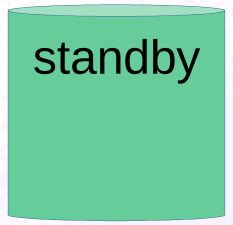
(3) Vacuum replaying on standby and truncating data file

(4) Restoring WAL from archive

master
standby

```
create table items(item_id int);
create table options(item_id int, v1 text);
```

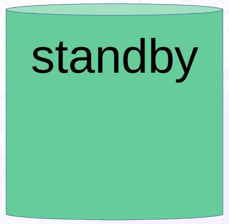




```
create table items(item_id int);
create table options(item_id int, v1 text);
```

```
Begin;


alter table options add v2 int;
```

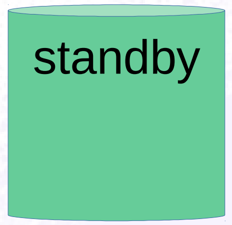
```
create table items(item_id int);
create table options(item_id int, v1 text);

Begin;


alter table options add v2 int;
```



```
Begin;


select * from items;
```

```
create table items(item_id int);
create table options(item_id int, v1 text);
```

```
Begin;


alter table options add v2 int;
```



```
Begin;


select * from items;
```

```
alter table items add a text;
```

master

```
create table items(item_id int);
create table options(item_id int, v1 text);
```

```
Begin;

alter table options add v2 int;

alter table items add a text;
```

standby

```
Begin;

select * from items;
```

`select * from options;`
`--DEADLOCK is not detected`

PostgreSQL 10

```
ERROR:  deadlock detected
LINE 1: select * from options;
                      ^
DETAIL:  Process 25364 waits for
AccessShareLock on relation 10000
of database 9000; blocked by
process 25322.
Process 25322 waits for
AccessExclusiveLock on relation
10000 of database 9000; blocked
by process 25364.
HINT:  See server log for query
details.
```

25322 is the PID of the apply process

Caveats

(1) Deadlock on standby

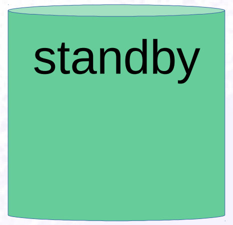
(2) DDL (statement_timout and deadlock_timeout)

(3) Vacuum replaying on standby and truncating data file

(4) Restoring WAL from archive

master
standby

```
alter table options add v2 int;
statement_timeout + deadlock_timeout
```



```
alter table options add v2 int;
statement_timeout + deadlock_timeout
```

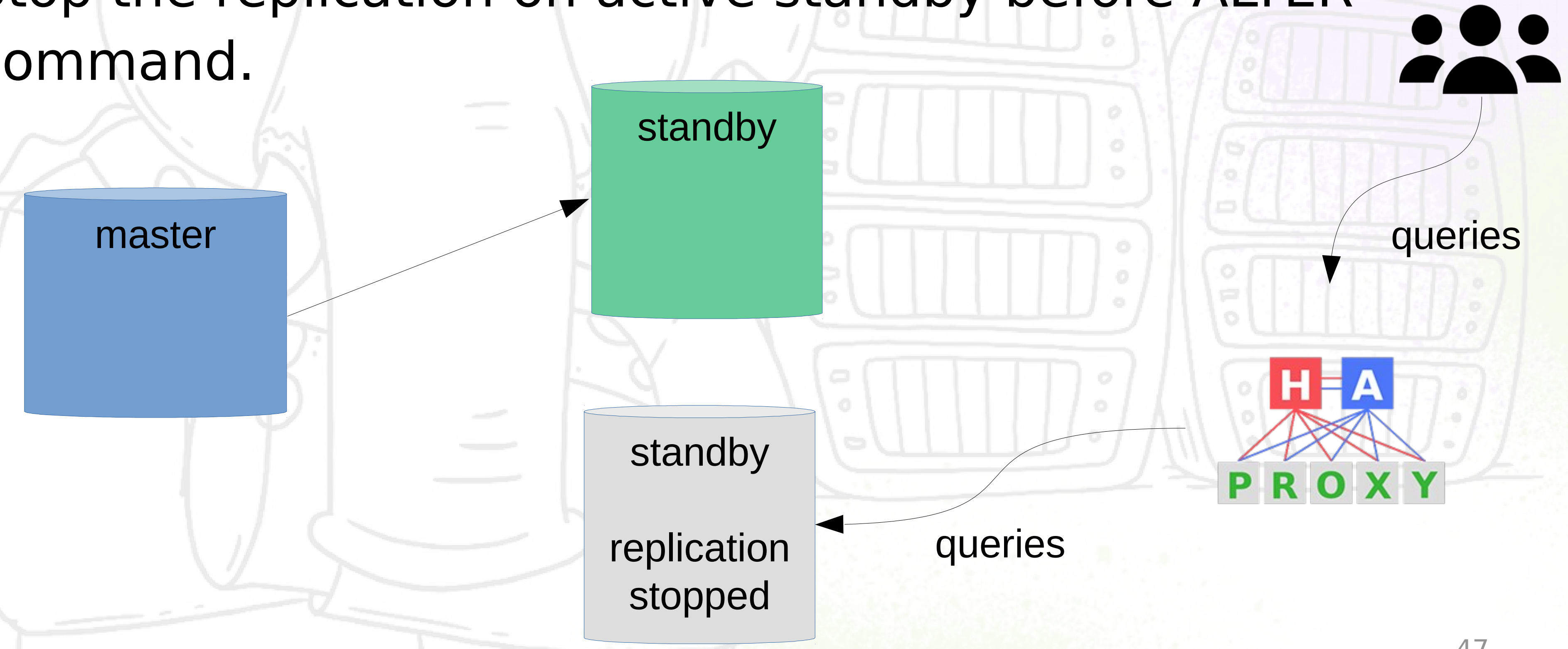
```
select * from options;
….
select * from options;
```

```
2018-01-12 16:54:40.208 MSK
pid=20949,user=user_3,db=test,hos
t=127.0.0.1:55763 LOG: process
20949 still waiting for
AccessShareLock on relation 10000
of database 9000 after 5000.055
ms
2018-01-12 16:54:40.208 MSK
pid=20949,user=user_3,db=test,hos
t=127.0.0.1:55763 DETAIL: Process
holding the lock: 46091. Wait
queue: 18639, 20949, 53445,
20770, 10799, 47217, 37659, 6727,
37662, 25742, 20771,
```

(2) DDL (statement_timeout and deadlock_timeout)

Script for HAProxy to implement external control (switching your traffic from all nodes)

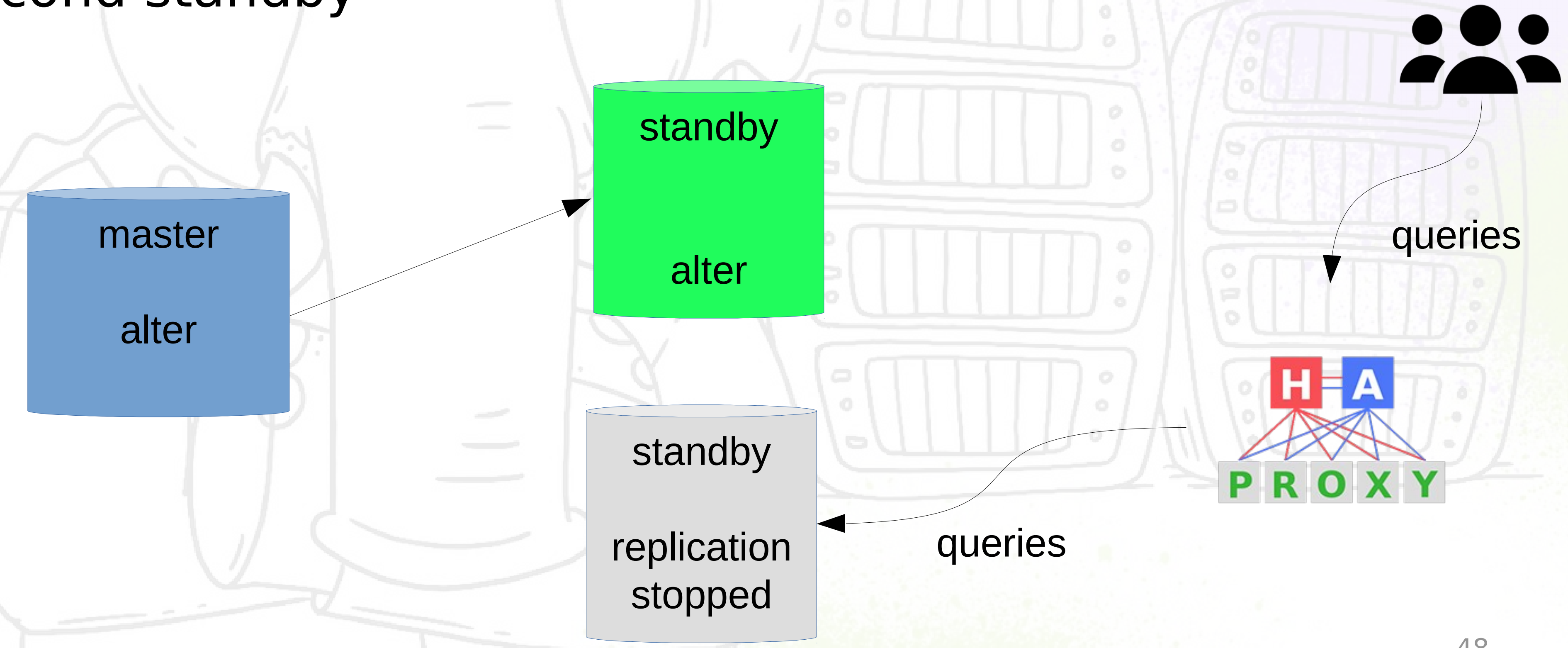
Stop the replication on active standby before ALTER command.



(2) DDL (statement_timeout and deadlock_timeout)

Run ALTER command

Wait till the ALTER command has been replayed on the second standby

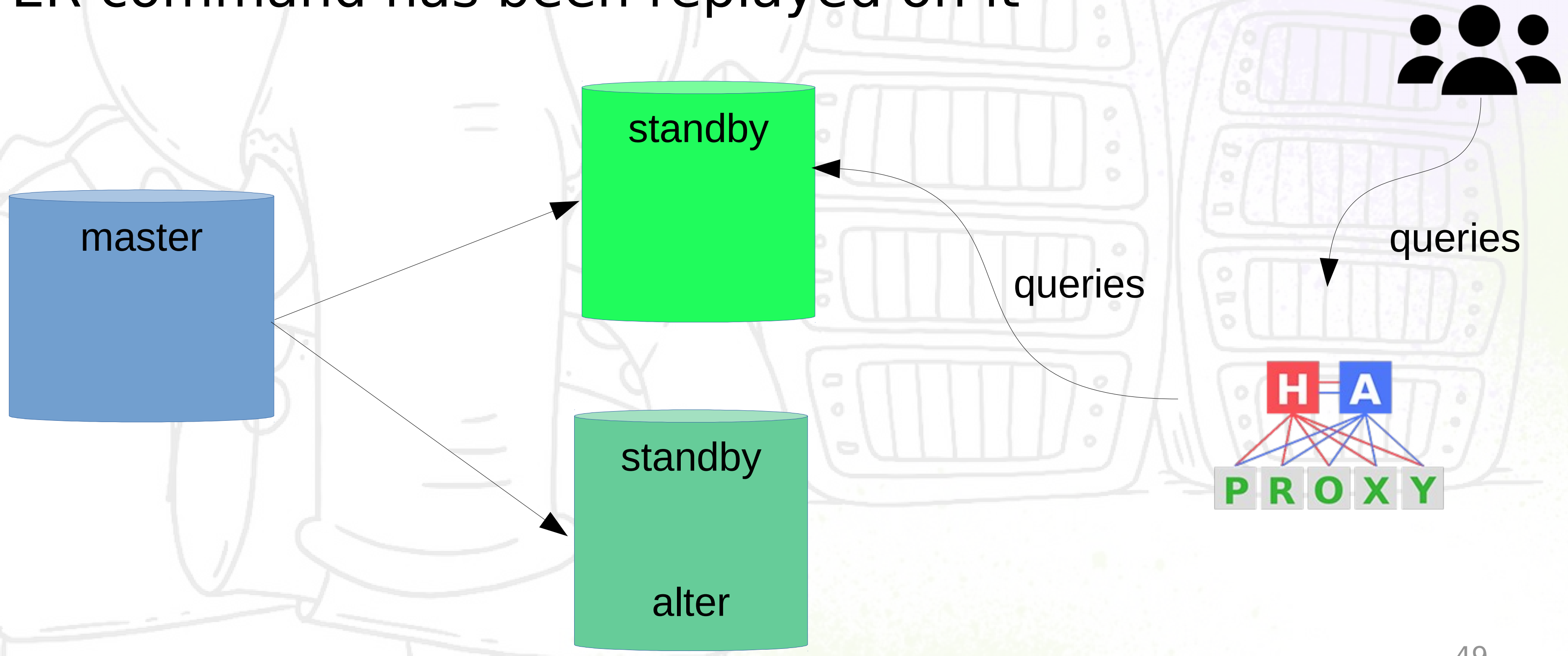


Avito

(2) DDL (statement_timeout and deadlock_timeout)

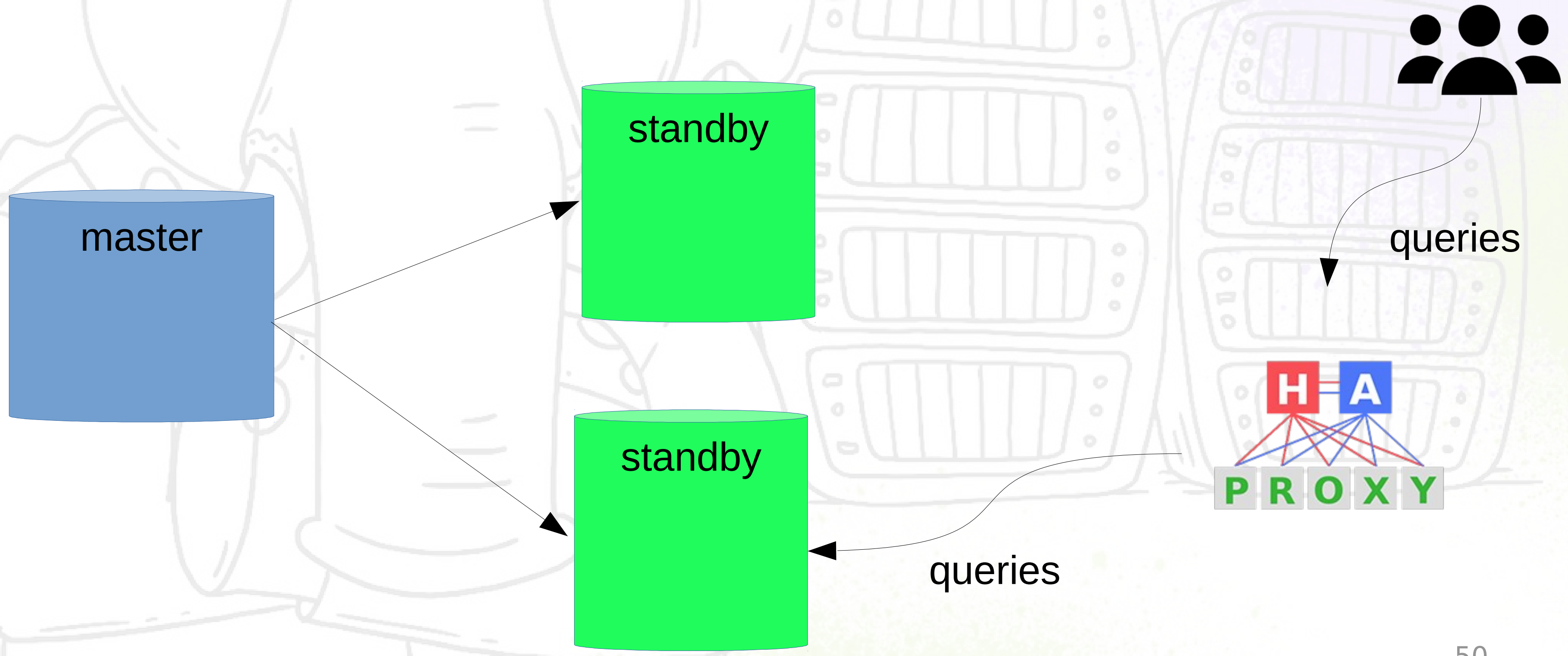
Switch traffic on the second standby

Start replication on the first standby and wait till the ALTER command has been replayed on it



(2) DDL (statement_timeout and deadlock_timeout)

Return the first standby to the pool of active standbys



Caveats

(1) Deadlock on standby

(2) DDL (statement_timout and deadlock_timeout)

(3) Vacuum replaying on standby and truncating data file

(4) Restoring WAL from archive

(3) Vacuum replaying on standby and truncating data file

- Vacuum can truncate the end of data file — the exclusive lock is needed for this action. At this moment on standby long locks between read only queries and recovery process occur
- It happens because some unlock actions are not written to WAL .
- On next slide you can see few AccessExclusive locks in one xid 920764691, and not a single unlock...
- Unlock happens much later. When standby replays commit

Avito

(3) Vacuum replaying on standby and truncating data file

```
tx: 920764691, lsn: 73CE0/10605980, desc: AccessExclusive locks: xid 920764691 db 16445 rel 3326466
tx: 920764691, lsn: 73CE0/10694568, desc: file truncate: base/16445/3326466 to 1965248 blocks
tx: 920764691, lsn: 73CE0/1105AB98, desc: AccessExclusive locks: xid 920764691 db 16445 rel 3326466
tx: 920764691, lsn: 73CE0/11116A88, desc: file truncate: base/16445/3326466 to 1965152 blocks
tx: 920764691, lsn: 73CE0/116C89C0, desc: AccessExclusive locks: xid 920764691 db 16445 rel 3326466
tx: 920764691, lsn: 73CE0/117211E0, desc: file truncate: base/16445/3326466 to 1965088 blocks
tx: 920764691, lsn: 73CE0/128DFF00, desc: AccessExclusive locks: xid 920764691 db 16445 rel 3326466
tx: 920764691, lsn: 73CE0/129A5DD0, desc: file truncate: base/16445/3326466 to 1964960 blocks
tx: 920764691, lsn: 73CE0/1315C4E8, desc: AccessExclusive locks: xid 920764691 db 16445 rel 3326466
tx: 920764691, lsn: 73CE0/134CF9E0, desc: file truncate: base/16445/3326466 to 1964832 blocks
```

In our example there is 75 WAL files interval between first lock and success truncate (unlock relation)

(3) Vacuum replaying on standby and truncating data file

The solution can be like:

```
* alter table ... disable truncate
         (autovacuum_truncate = disable) ?

* Decrease the number of locks on standby?
                  (Postgres Professional)
```

Avito

Caveats

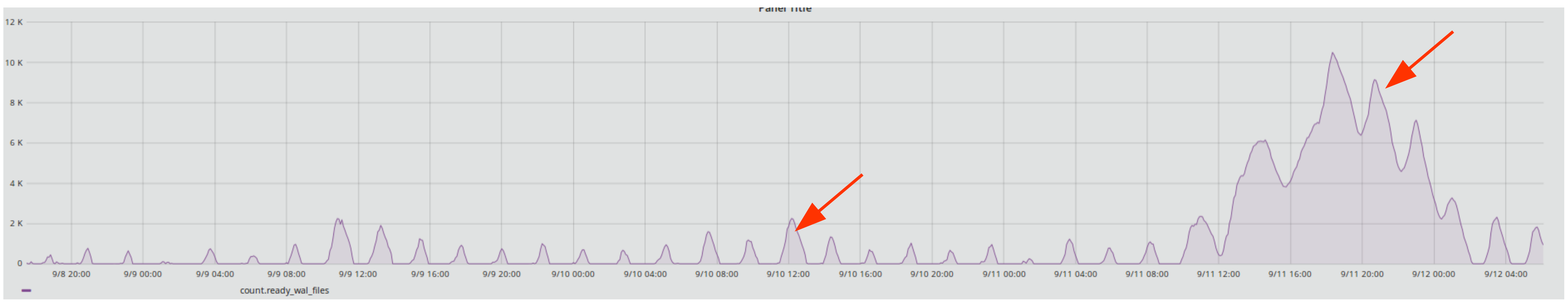
(1) Deadlock on standby

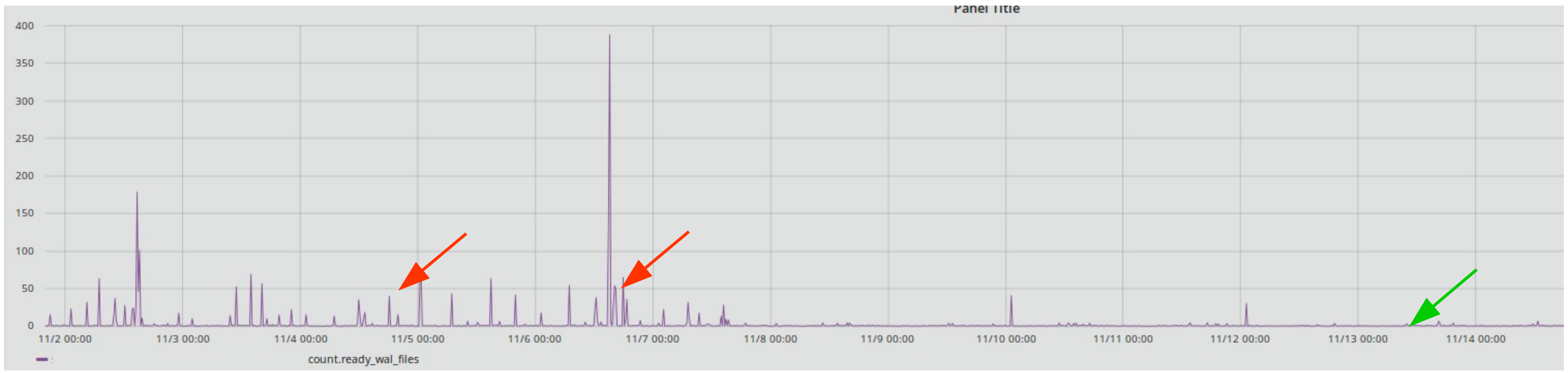
(2) DDL (statement_timout and deadlock_timeout)

(3) Vacuum replaying on standby and truncating data file

(4) Restoring WAL from archive

More and more WAL



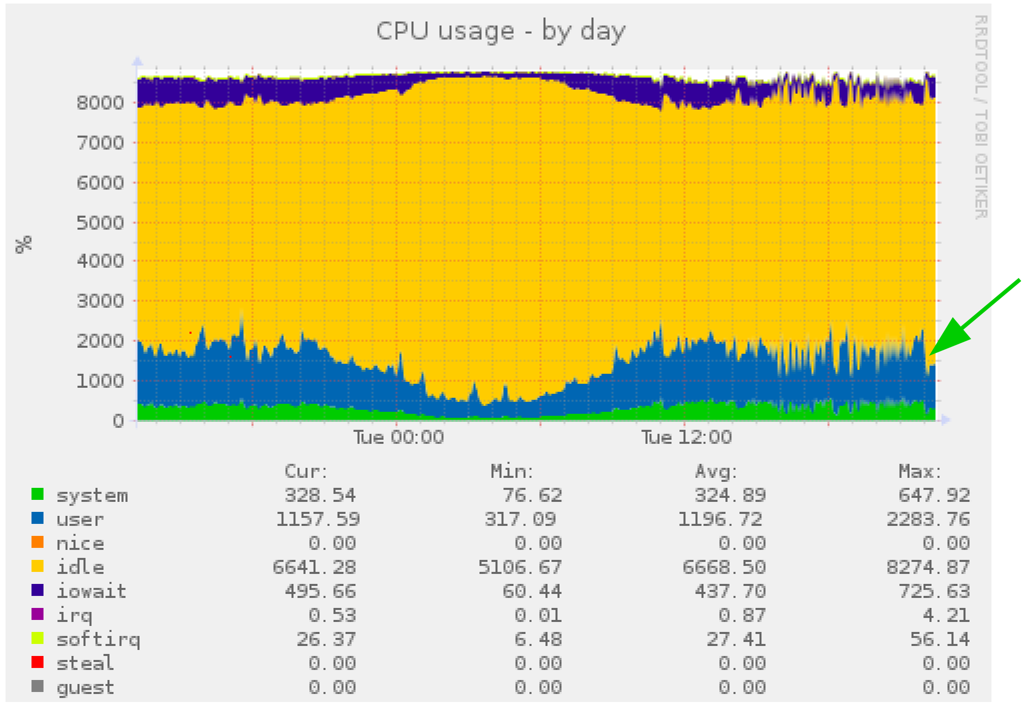


Avito archive 2016

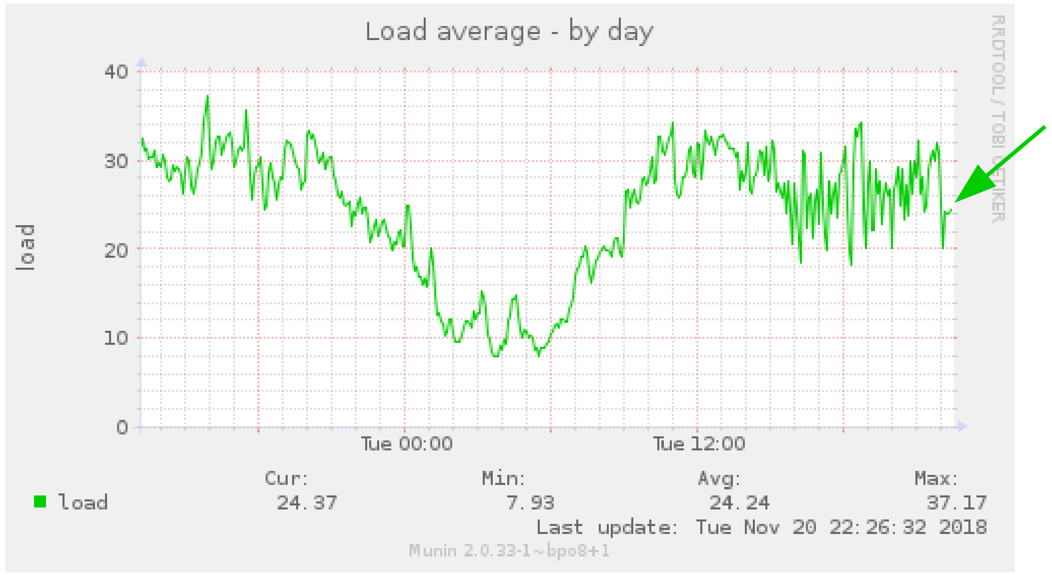
https://github.com/avito-tech/dba-utils/tree/master/pg_archive

```
archive_command = '/usr/local/bin/archive_cmd HOSTNAME /postgresql/walldir/logs.complete %p %f'

usage: archive_cmd DST-HOSTNAME DST-DIR SRC-WAL-FILENAME-WITH-PATH SRC-WAL-FILENAME

DST-HOSTNAME               - for scp

DST-DIR                    - archive directory for WALs

SRC-WAL-FILENAME-WITH-PATH - %p (file name with path)

SRC-WAL-FILENAME           - %f (file name)


# archive in one thread if

# - ready WAL files lower then threshold ready_wals_for_parallel
```

Archiving 1 WAL ~ 60ms



	Cur:	Min:	Avg:	Max:
🟩 system	328.54	76.62	324.89	647.92
🟦 user	1157.59	317.09	1196.72	2283.76
🟧 nice	0.00	0.00	0.00	0.00
🟨 idle	6641.28	5106.67	6668.50	8274.87
🟦 iowait	495.66	60.44	437.70	725.63
🟪 irq	0.53	0.01	0.87	4.21
🟩 softirq	26.37	6.48	27.41	56.14
🟥 steal	0.00	0.00	0.00	0.00
⬜ guest	0.00	0.00	0.00	0.00

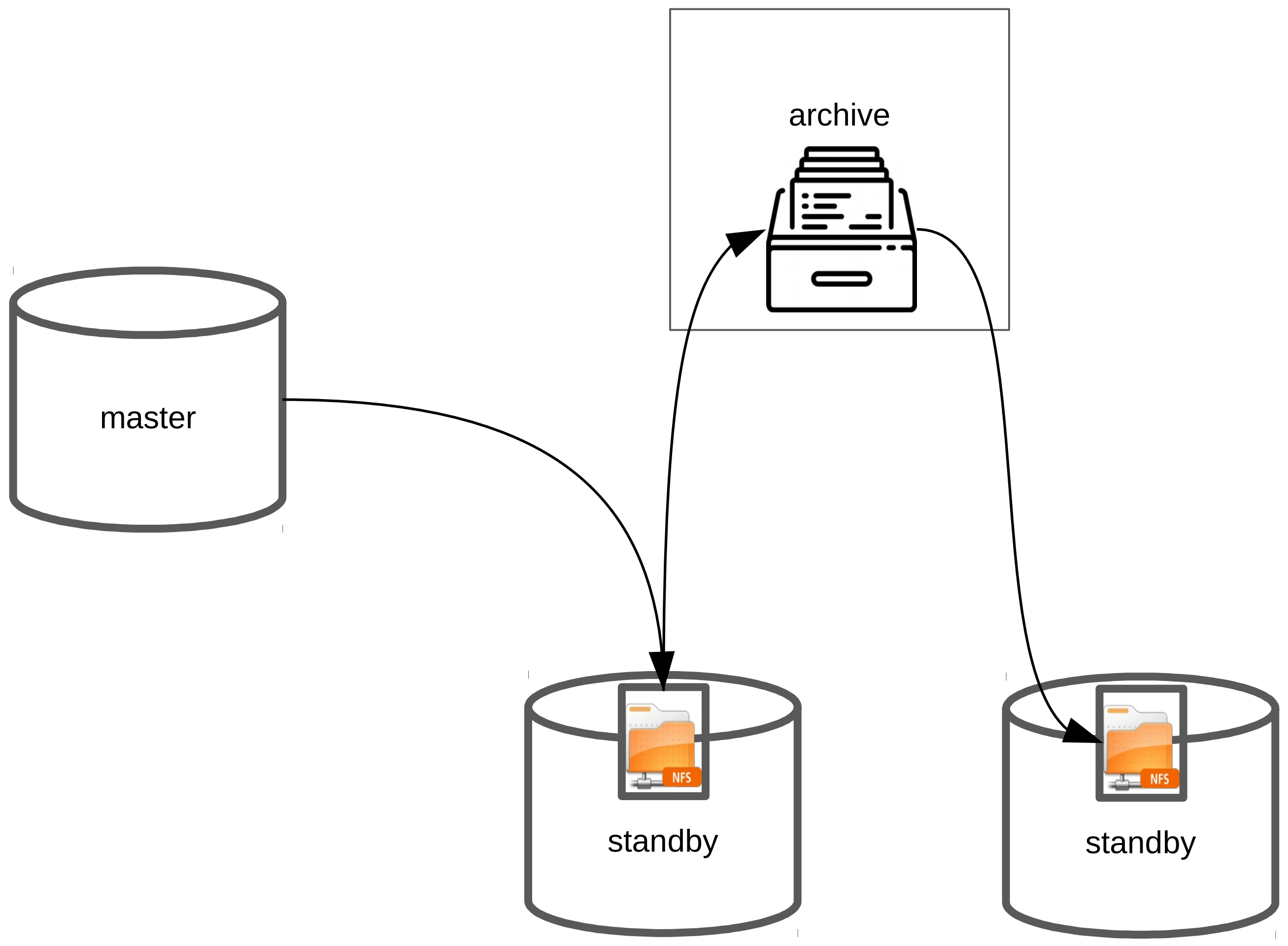


	Cur:	Min:	Avg:	Max:
🟩 load	24.37	7.93	24.24	37.17

Last update: Tue Nov 20 22:26:32 2018

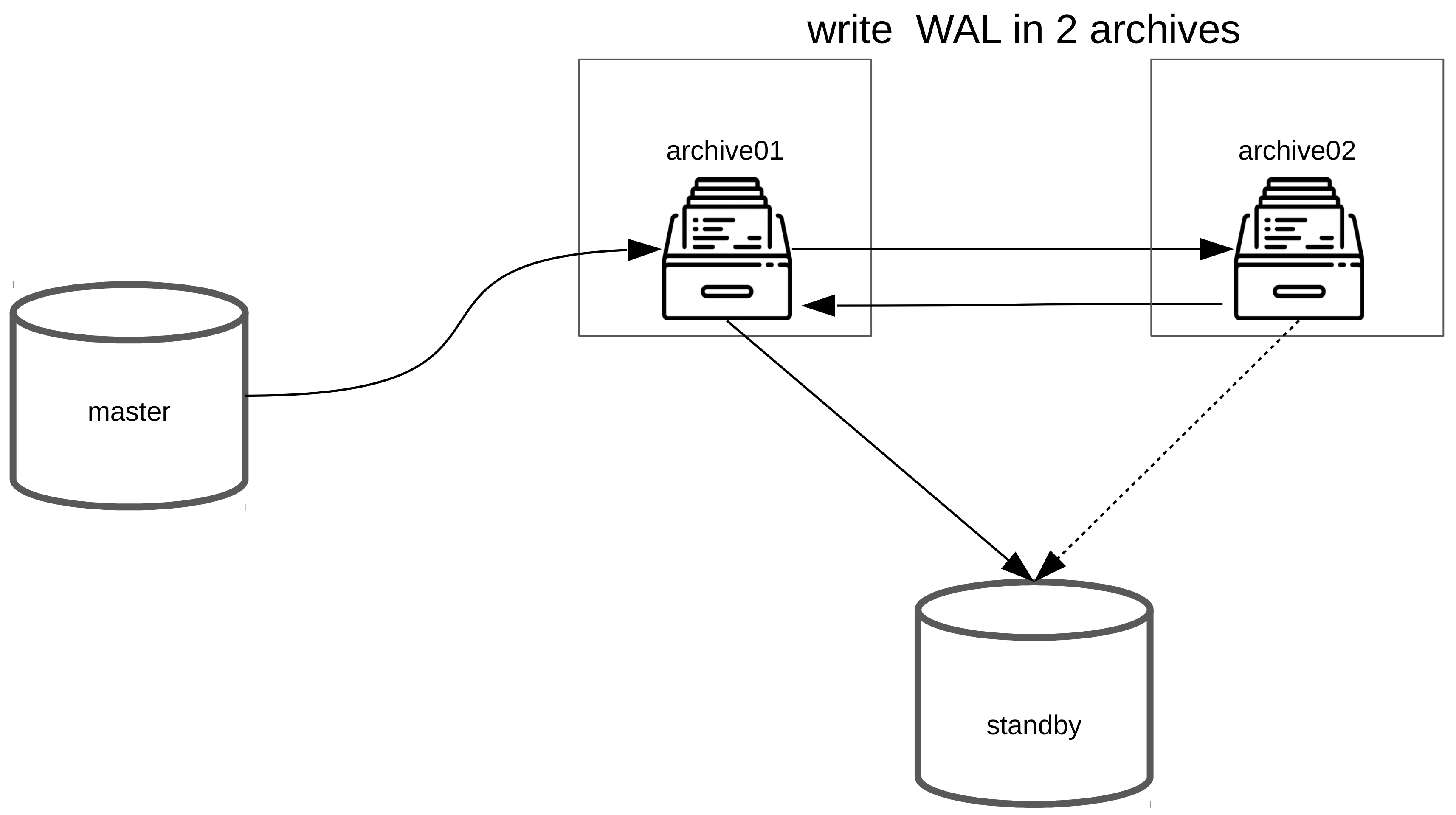
Avito

Old archive schema



New archive schema

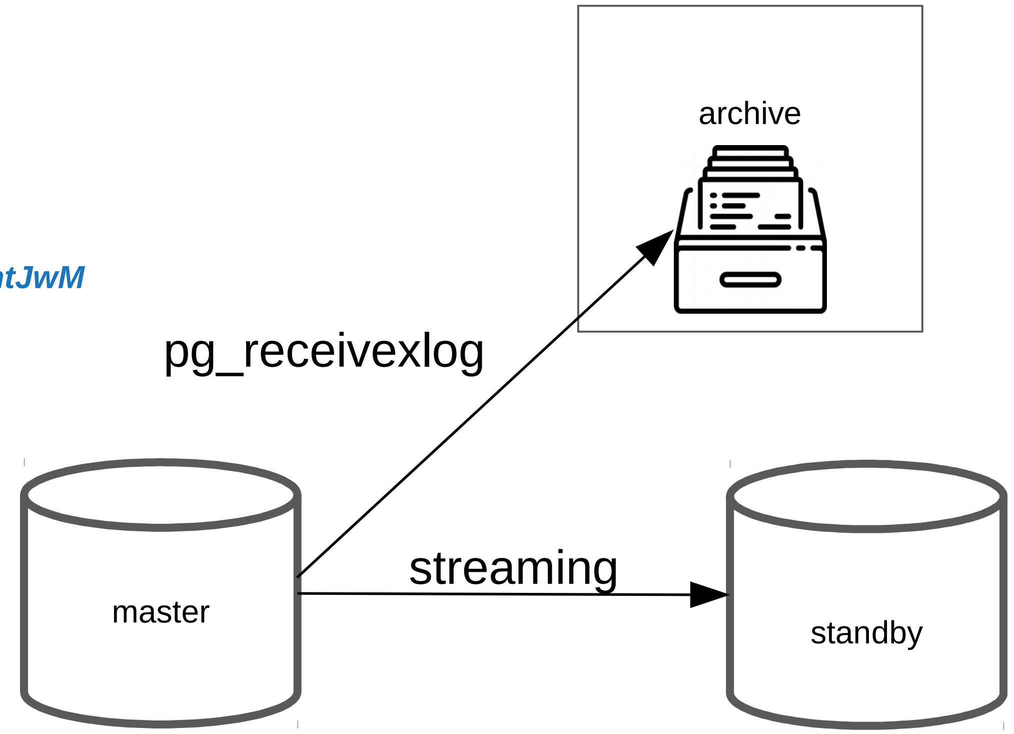
https://github.com/avito-tech/dba-utils/tree/master/pg_archive2



Streaming

WARM standby done right
Heikki Linnakangas

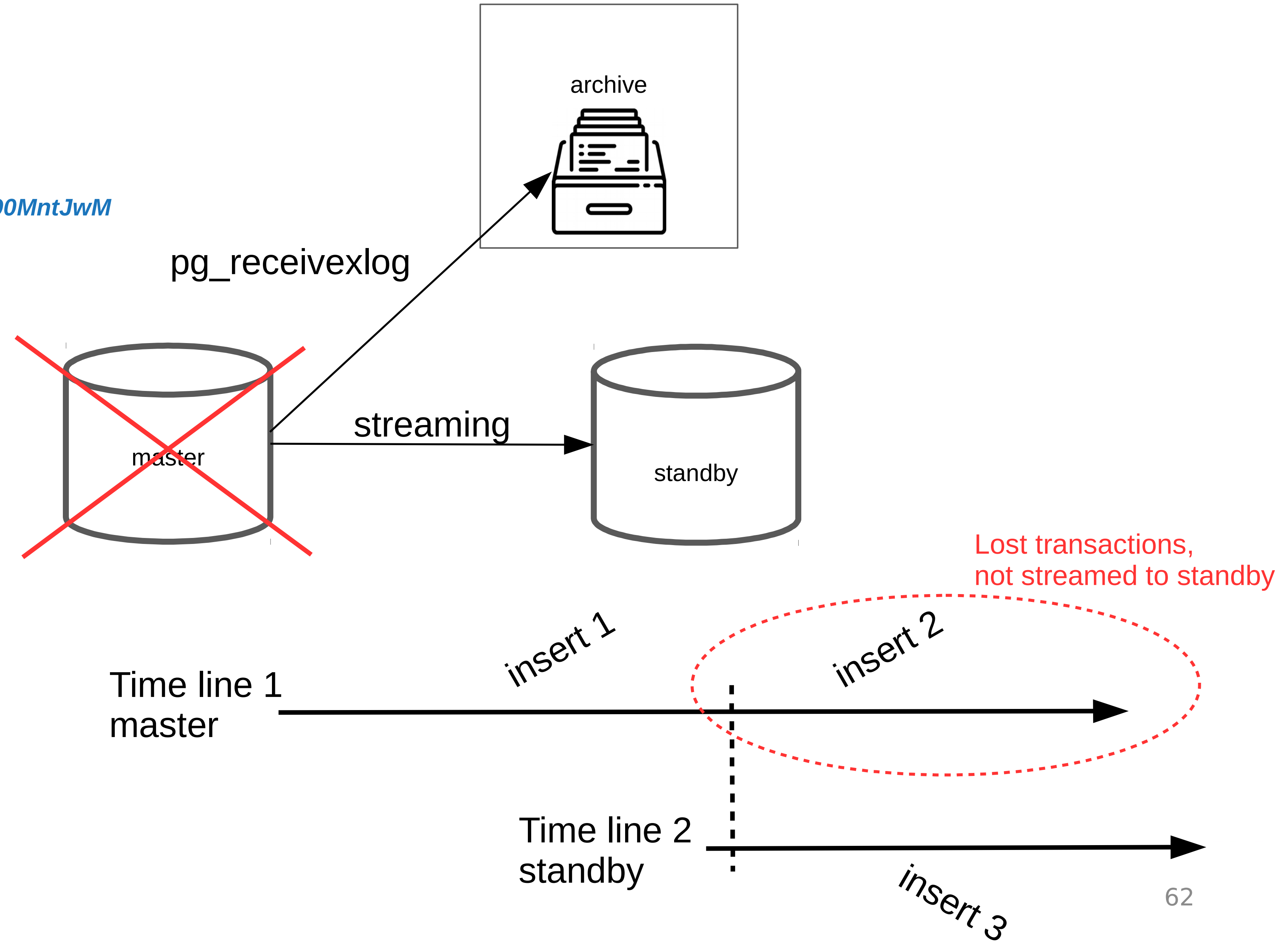
https://pgday.ru/ru/2015/papers/8
https://www.youtube.com/watch?v=mlQ90MntJwM



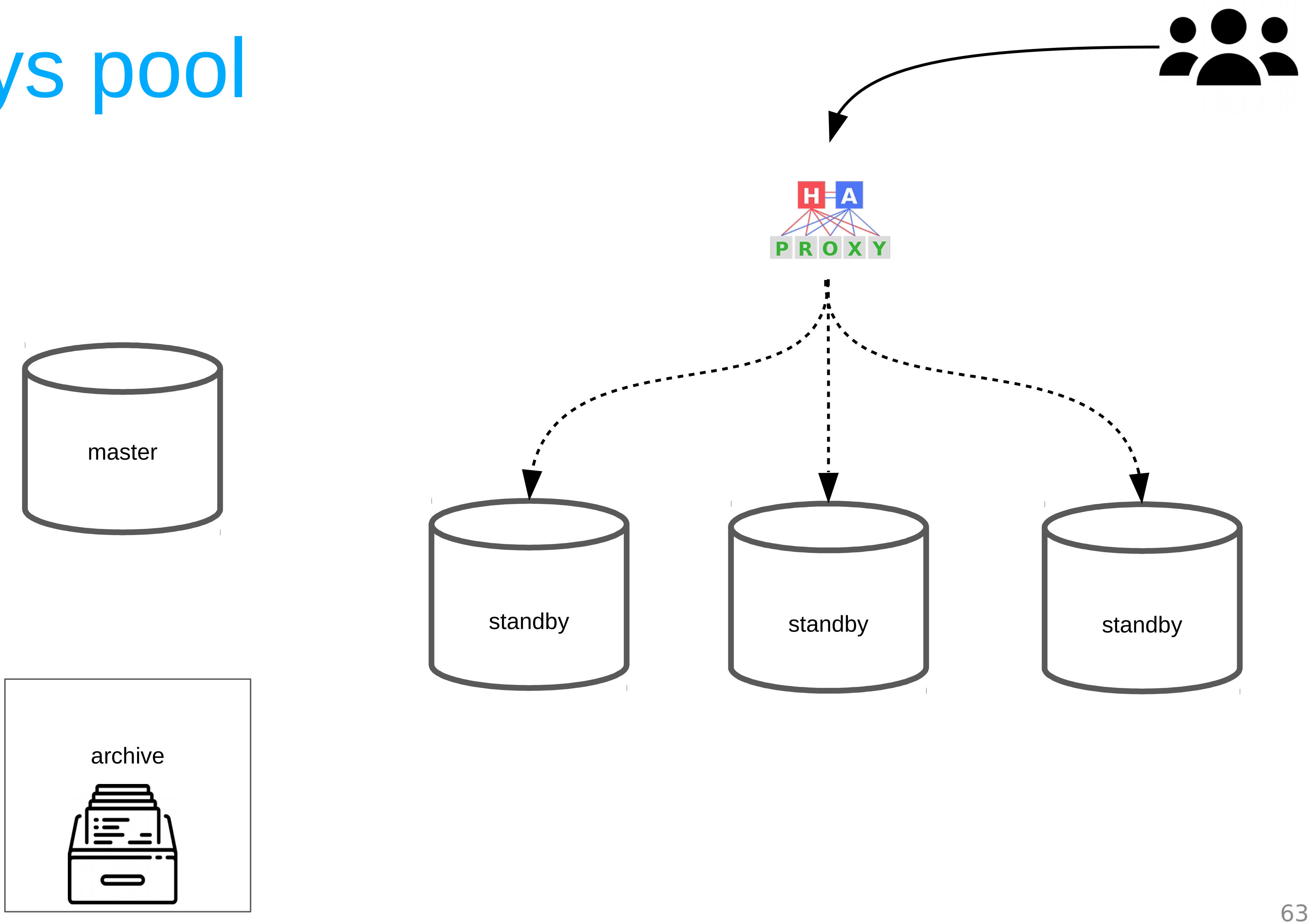
Streaming

WARM standby done right
Heikki Linnakangas

https://pgday.ru/ru/2015/papers/8
https://www.youtube.com/watch?v=mlQ90MntJwM



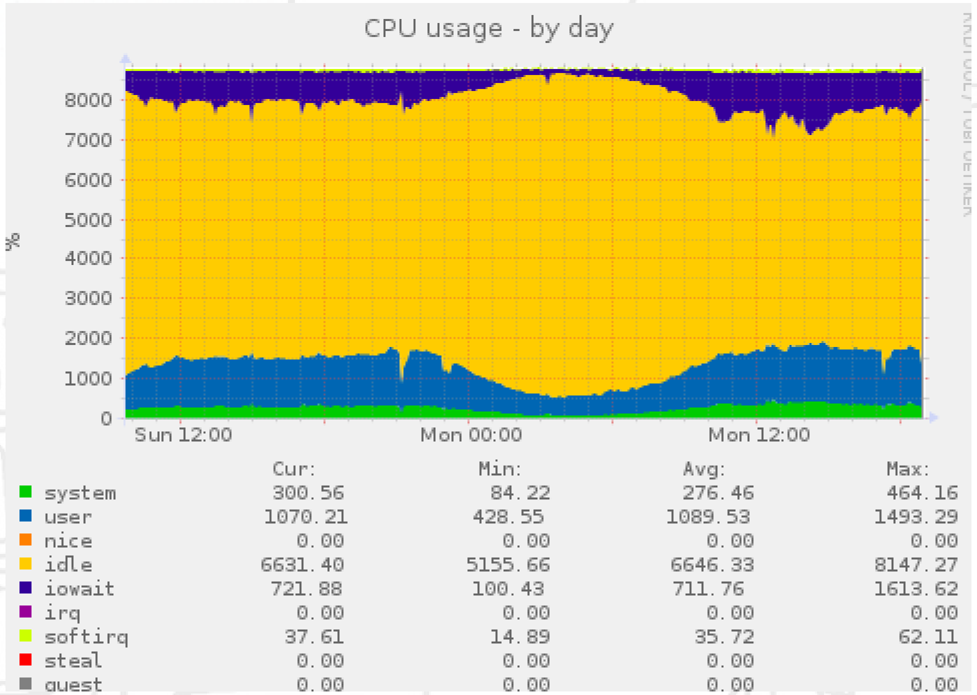
Standbys pool



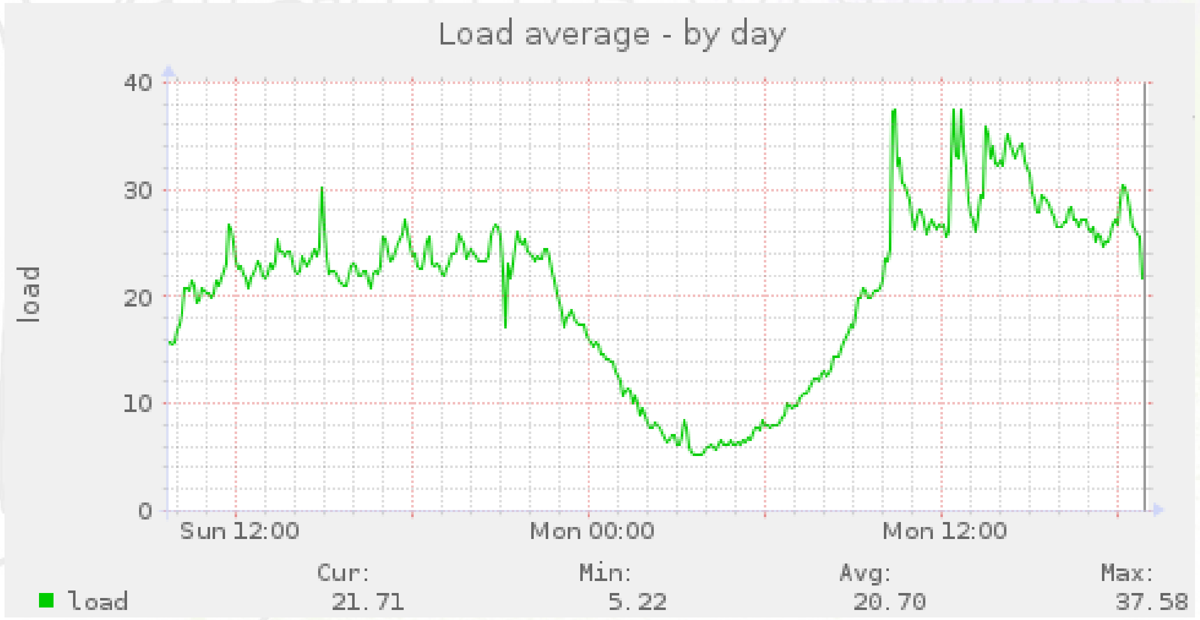
HAProxy check function

```
if master
     then false
if lag > max
     then create file and return false
if lag > min and file exists
     then return false
if lag < min and file exists
     then remove file and return true
else
     true
```

Monitoring CPU



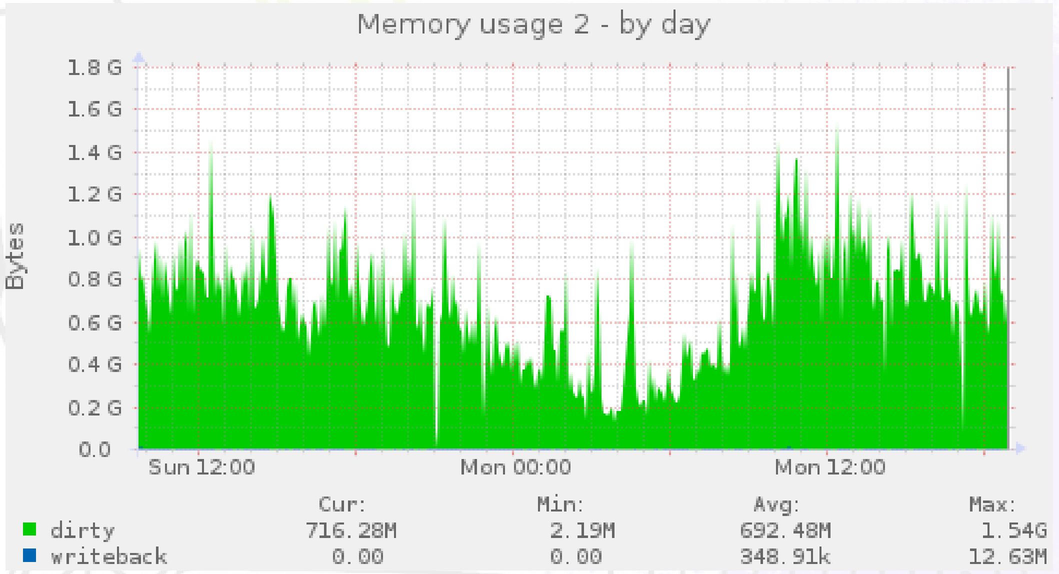
	Cur:	Min:	Avg:	Max:
🟩 system	300.56	84.22	276.46	464.16
🟦 user	1070.21	428.55	1089.53	1493.29
🟧 nice	0.00	0.00	0.00	0.00
🟨 idle	6631.40	5155.66	6646.33	8147.27
🟦 iowait	721.88	100.43	711.76	1613.62
🟪 irq	0.00	0.00	0.00	0.00
🟩 softirq	37.61	14.89	35.72	62.11
🟥 steal	0.00	0.00	0.00	0.00
⬛ guest	0.00	0.00	0.00	0.00



	Cur:	Min:	Avg:	Max:
🟩 load	21.71	5.22	20.70	37.58

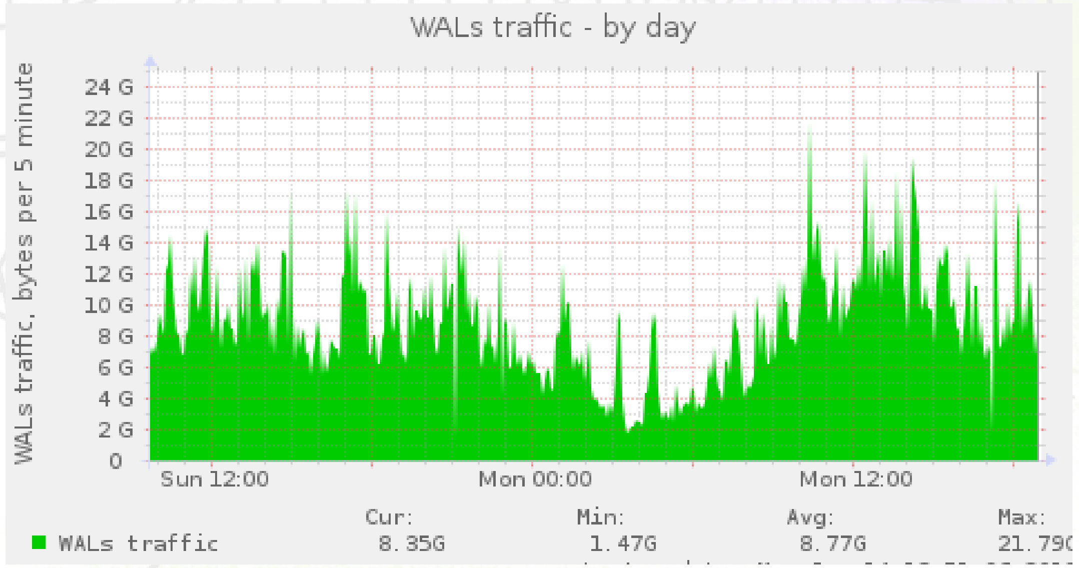
Monitoring

Memory



```
                Cur:       Min:       Avg:      Max:
■ dirty       716.28M     2.19M    692.48M     1.54G
■ writeback     0.00      0.00     348.91k    12.63M
```

WALs traffic

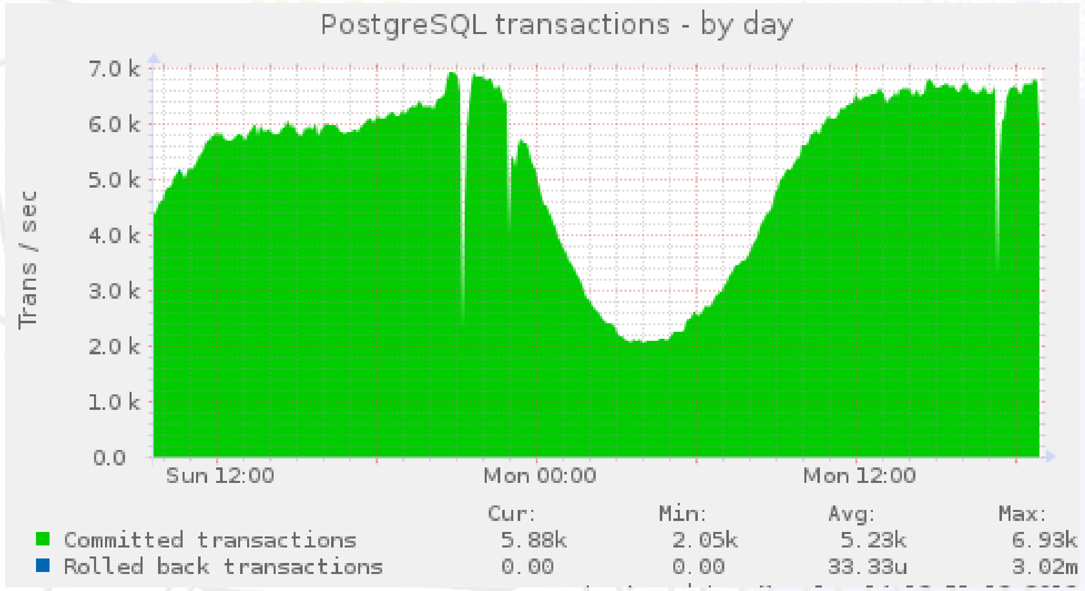


```
                Cur:       Min:       Avg:      Max:
■ WALs traffic  8.35G     1.47G      8.77G     21.79
```

Avito

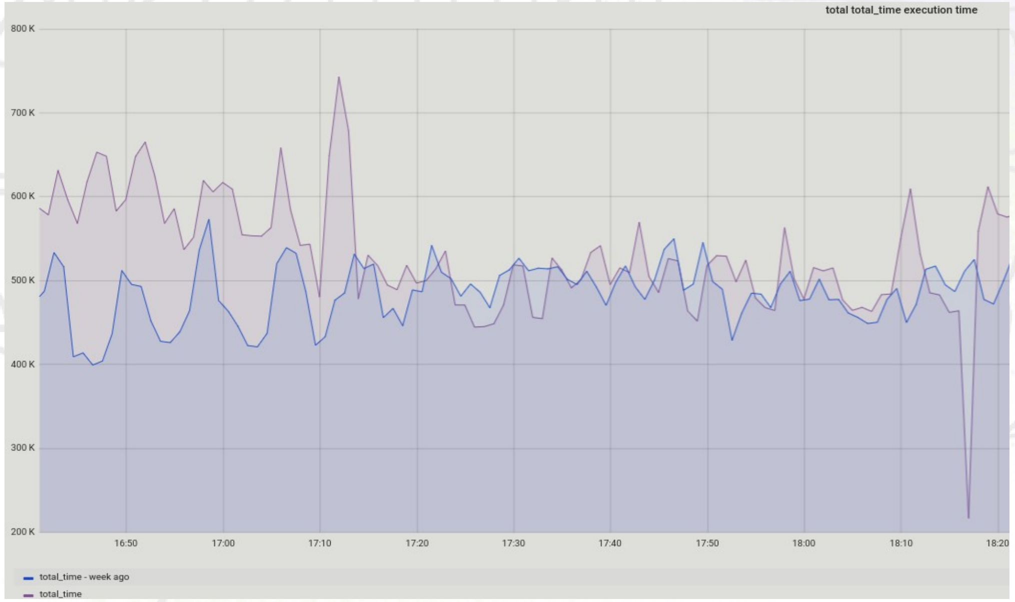
Monitoring transactions

TPS

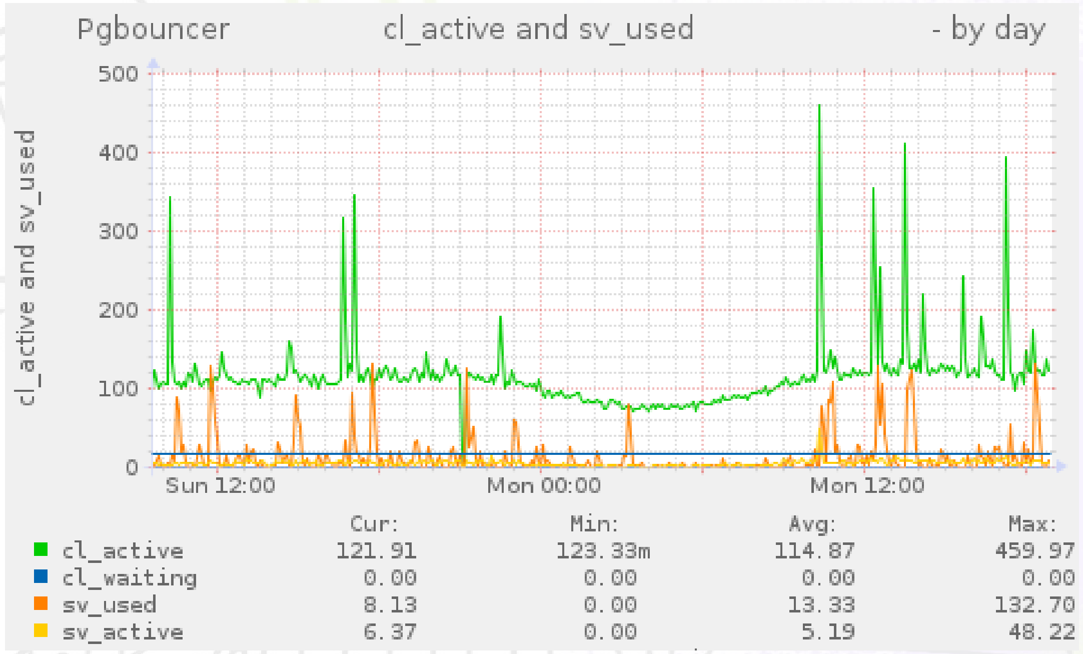


	Cur:	Min:	Avg:	Max:
🟩 Committed transactions	5.88k	2.05k	5.23k	6.93k
🟦 Rolled back transactions	0.00	0.00	33.33u	3.02m

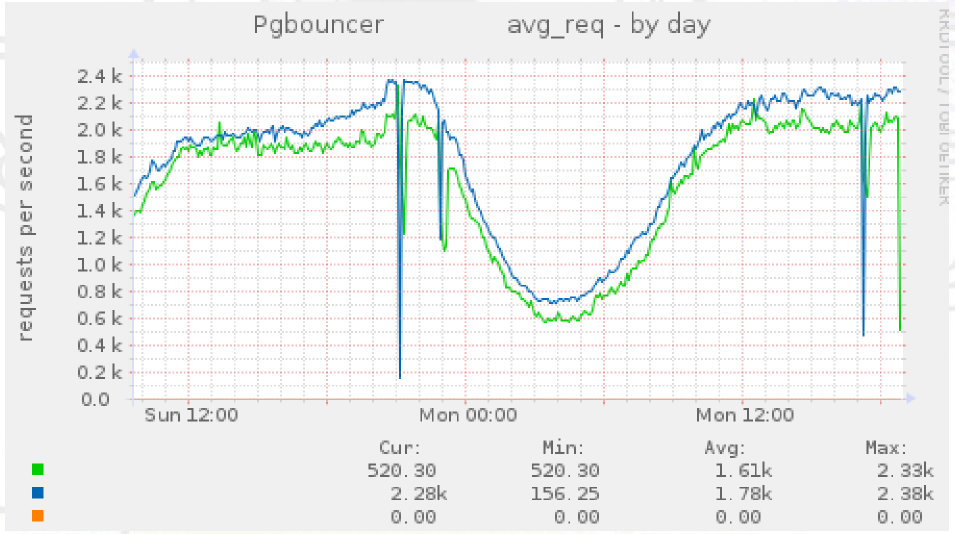
pg stat statements



Monitoring PgBouncer



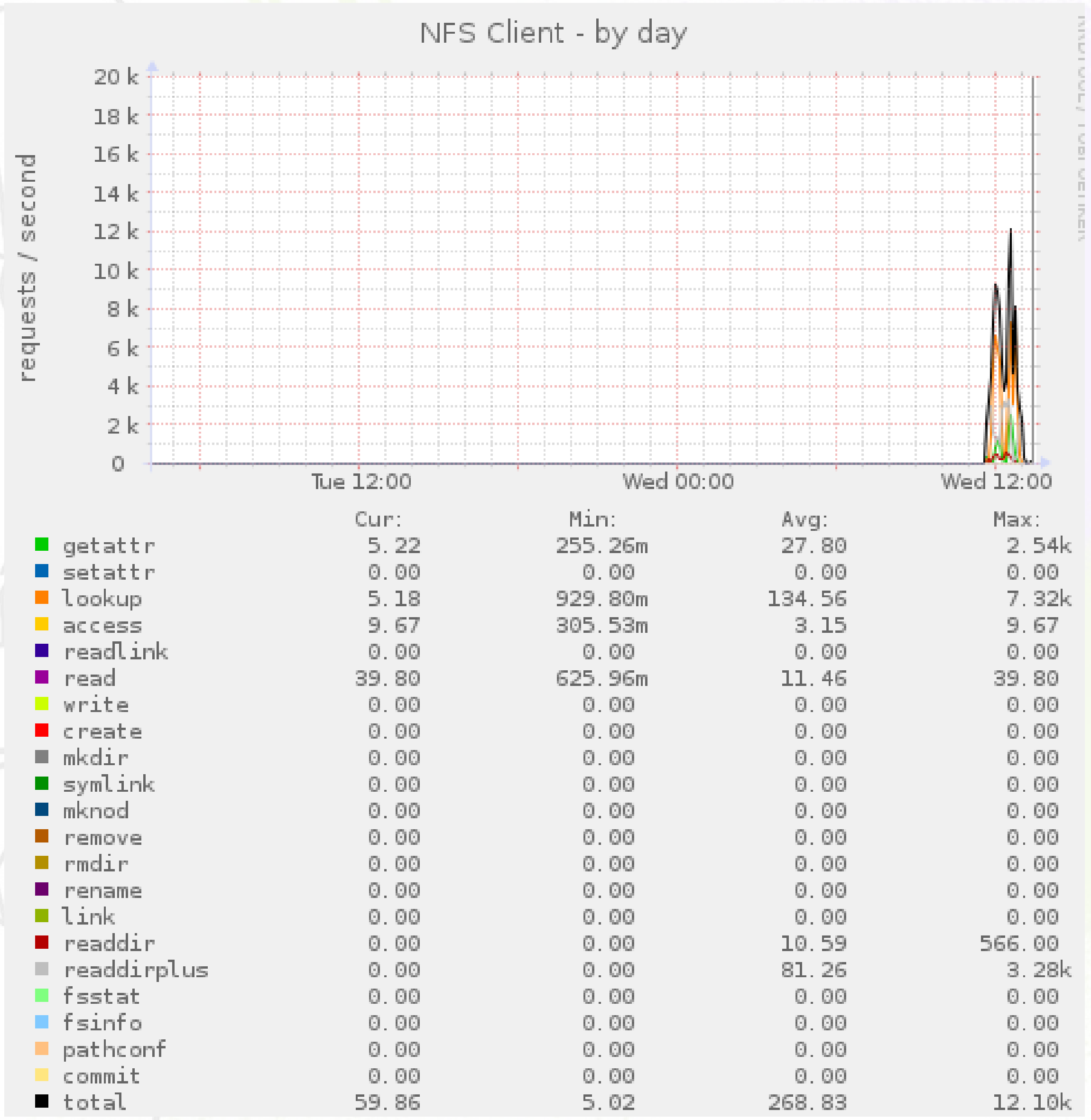
	Cur:	Min:	Avg:	Max:
🟩 cl_active	121.91	123.33m	114.87	459.97
🟦 cl_waiting	0.00	0.00	0.00	0.00
🟧 sv_used	8.13	0.00	13.33	132.70
🟨 sv_active	6.37	0.00	5.19	48.22



	Cur:	Min:	Avg:	Max:
🟩	520.30	520.30	1.61k	2.33k
🟦	2.28k	156.25	1.78k	2.38k
🟧	0.00	0.00	0.00	0.00

Avito

Monitoring archive/nfs



		Cur:	Min:	Avg:	Max:
🟩	getattr	5.22	255.26m	27.80	2.54k
🟦	setattr	0.00	0.00	0.00	0.00
🟧	lookup	5.18	929.80m	134.56	7.32k
🟨	access	9.67	305.53m	3.15	9.67
🟪	readlink	0.00	0.00	0.00	0.00
🟪	read	39.80	625.96m	11.46	39.80
🟩	write	0.00	0.00	0.00	0.00
🟥	create	0.00	0.00	0.00	0.00
■	mkdir	0.00	0.00	0.00	0.00
🟩	symlink	0.00	0.00	0.00	0.00
🟦	mknod	0.00	0.00	0.00	0.00
🟫	remove	0.00	0.00	0.00	0.00
🟫	rmdir	0.00	0.00	0.00	0.00
🟪	rename	0.00	0.00	0.00	0.00
🟩	link	0.00	0.00	0.00	0.00
🟥	readdir	0.00	0.00	10.59	566.00
⬜	readdirplus	0.00	0.00	81.26	3.28k
🟩	fsstat	0.00	0.00	0.00	0.00
🟦	fsinfo	0.00	0.00	0.00	0.00
🟧	pathconf	0.00	0.00	0.00	0.00
🟨	commit	0.00	0.00	0.00	0.00
⬛	total	59.86	5.02	268.83	12.10k

Conclusion

- There are few kinds of standbys

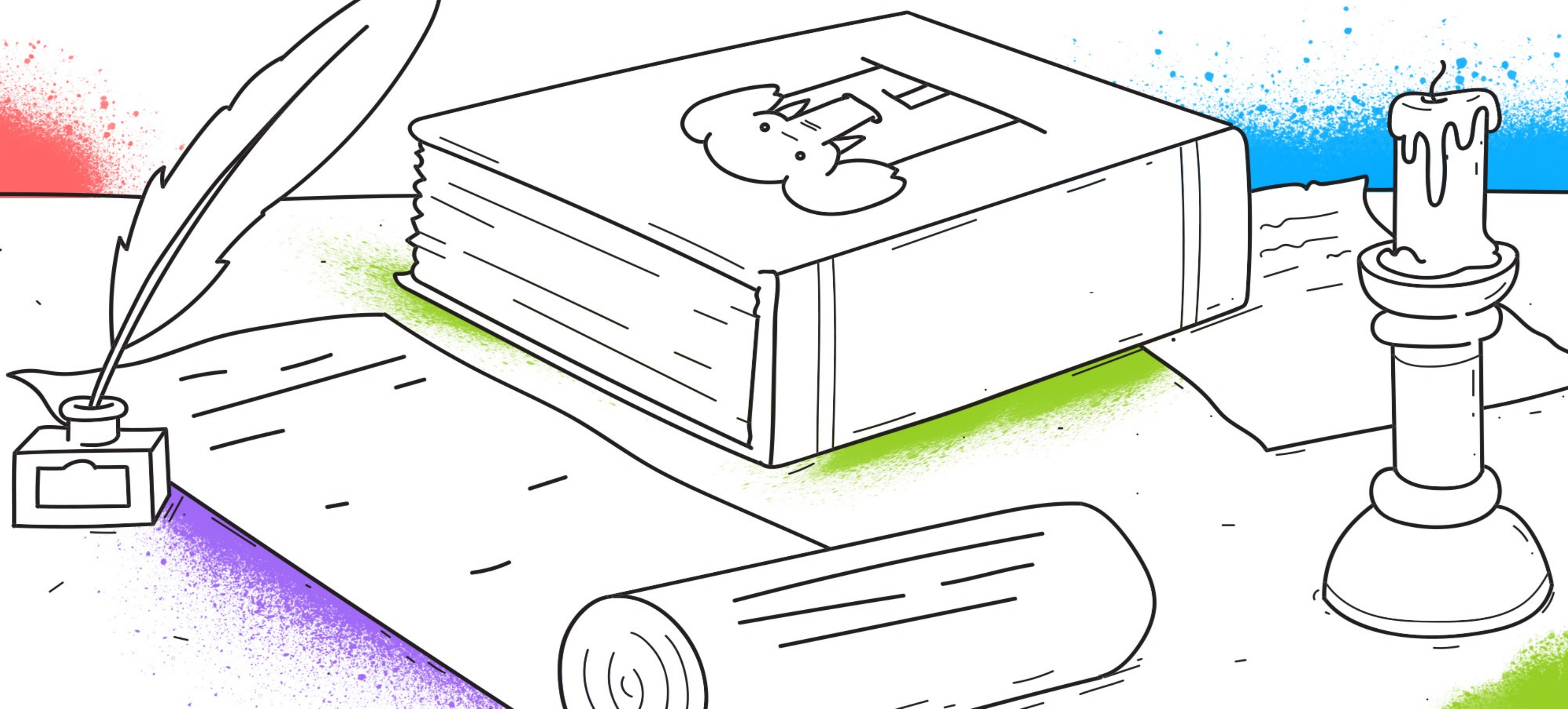
- We can scale reads with the help of standby
    - ignore stale reads
    - logical routing
    - hot cache

- There are some caveats with standby in production

- Archive and backup depends on your DRP

- Major upgrade with standby also needs advanced manipulations

****** https://www.postgresql.org/message-id/15615-a64615b9b466c18f%40postgresql.org***



https://www.avito.ru/company/job/dp-eng

kevteev@avito.ru