



Standby in Production

Konstantin Evteev

Ottawa 2018

Standby in Production

(1) Deadlock on standby

(2) DDL (statement_timeout and deadlock_timeout)

(3) Vacuum replaying on standby and truncating data file



master

standby

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



master

standby

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

Begin;

```
alter table options add v2 int;
```



master



standby

```
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;
```



master



standby

```
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;
```

Begin;

```
select * from items;
```



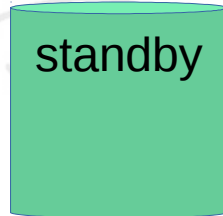
master

```
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 :

Standby in Production

(1) Deadlock on standby

(2) DDL (statement_timeout and deadlock_timeout)

(3) Vacuum replaying on standby and truncating data file



master

standby

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

master

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

standby

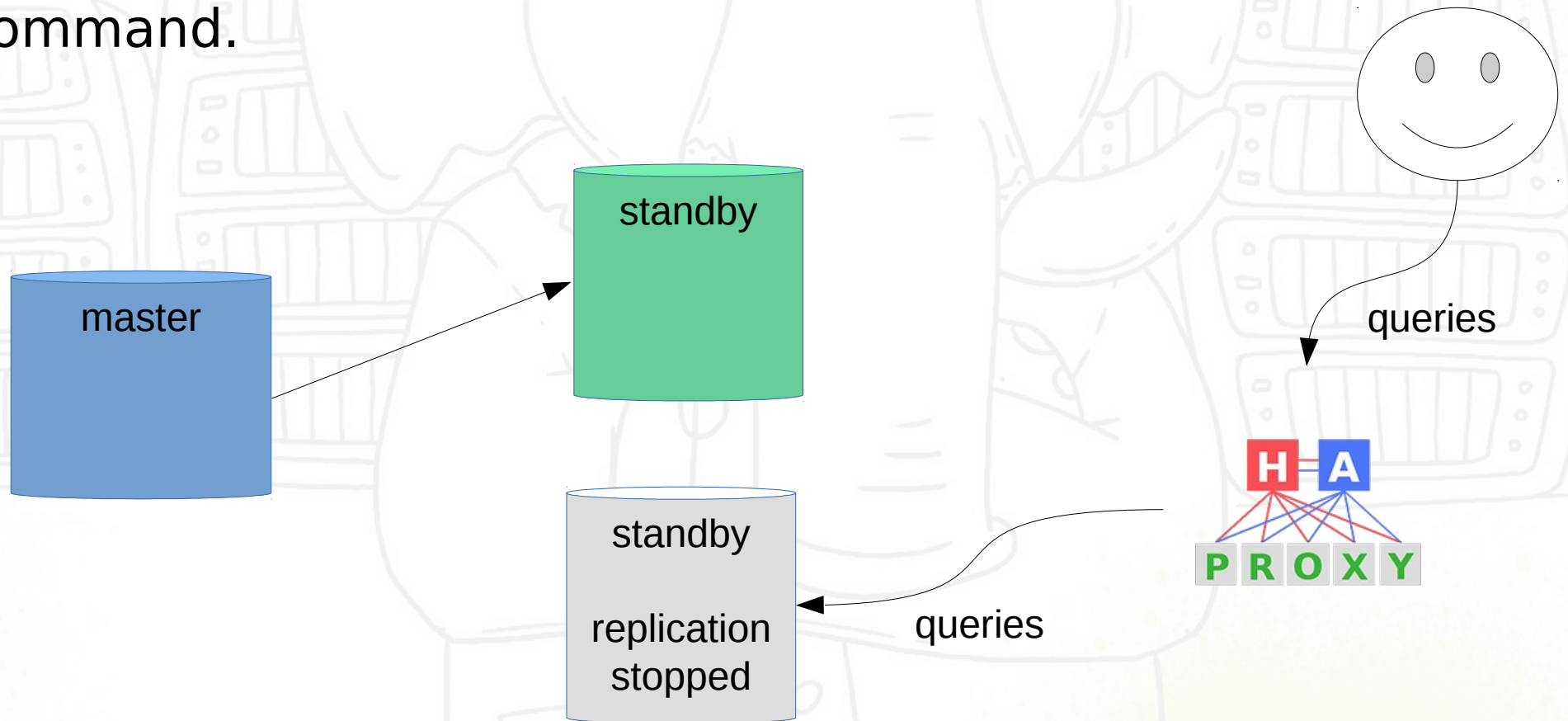
```
select * from options;  
...  
select * from options;
```

```
2018-01-12 16:54:40.208 MSK  
pid=20949,user=user_3,db=test,host=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,host=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)

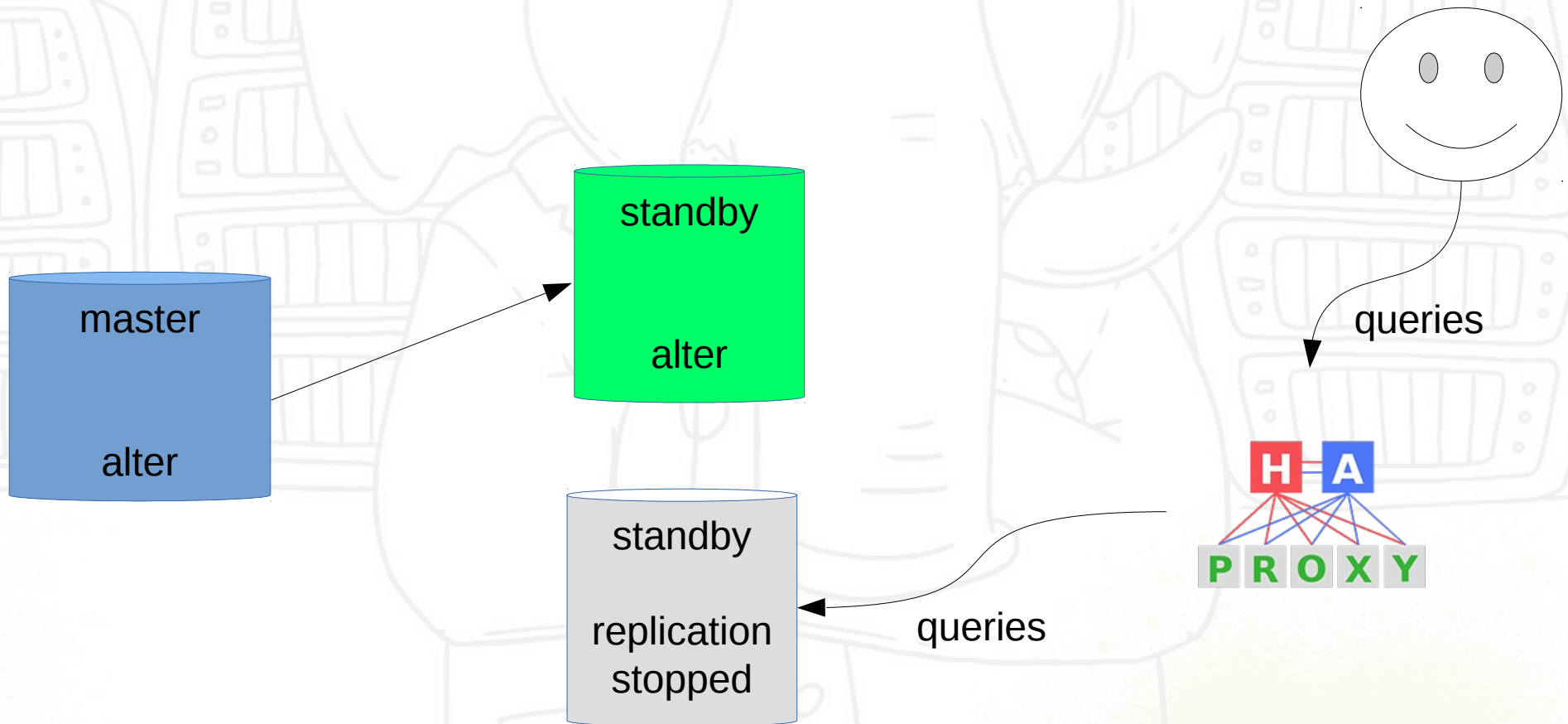
Script for HAProxy to implement external control
(switching your traffic from all nodes)

Stop the replication on active standby before ALTER
command.



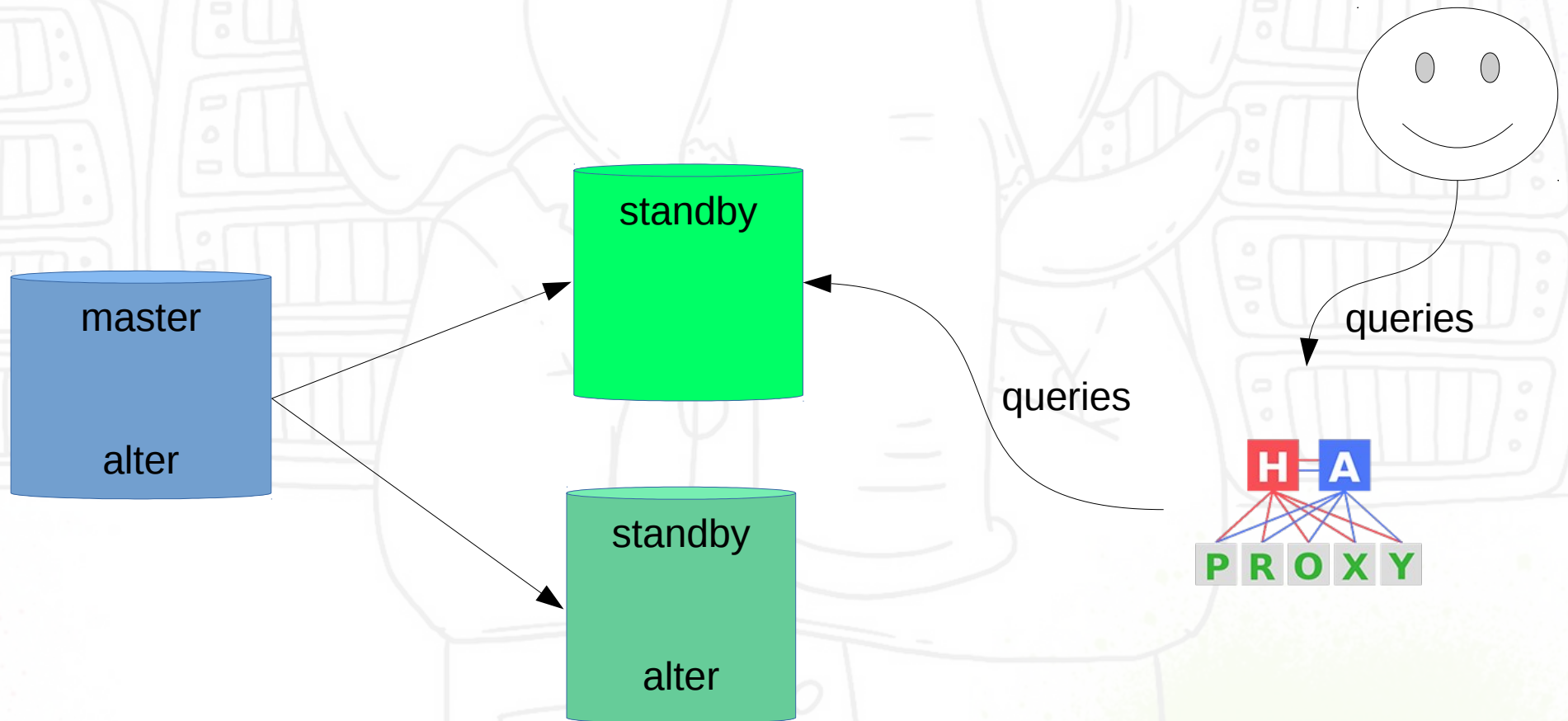
(2) DDL (statement_timeout and deadlock_timeout)

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



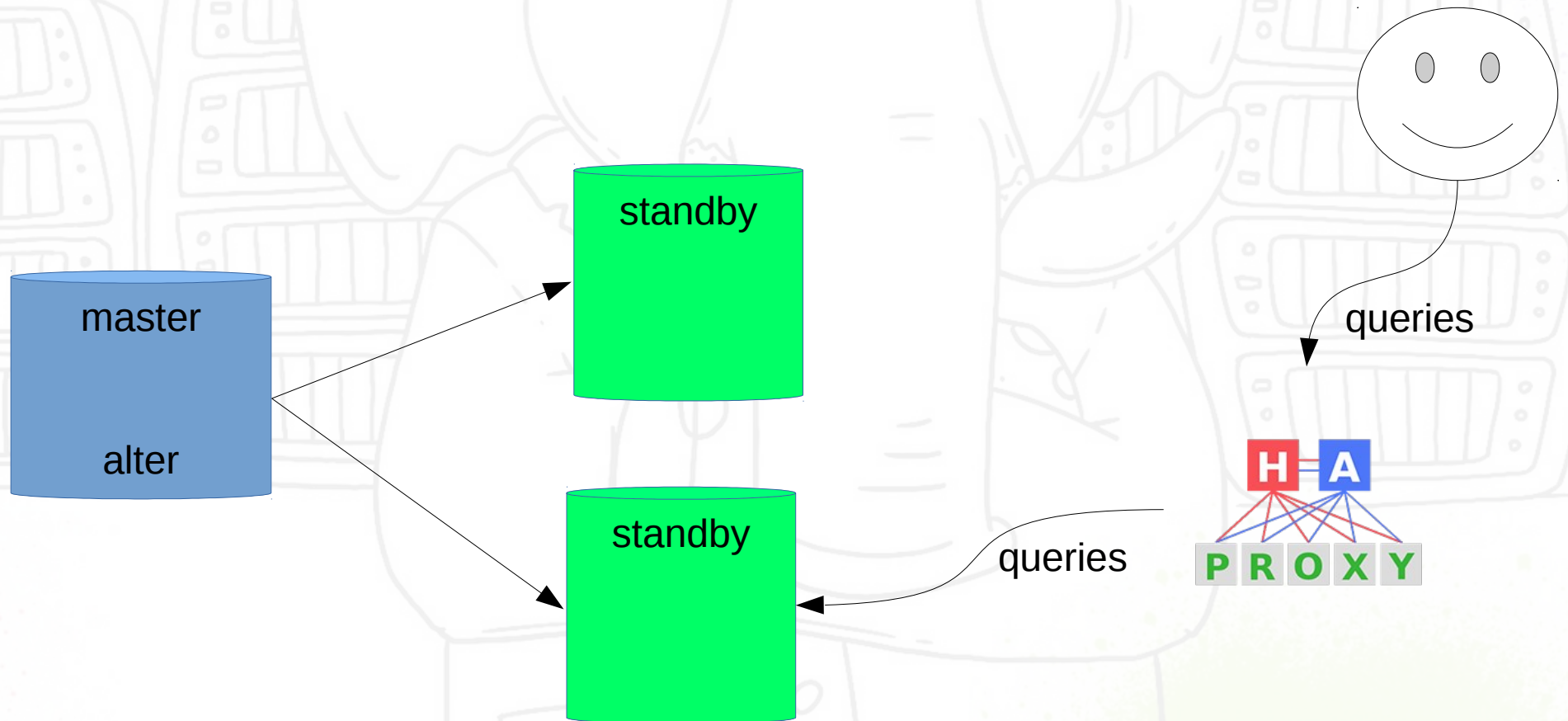
(2) DDL (statement_timeout and deadlock_timeout)

- 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



Standby in Production

(1) Deadlock on standby

(2) DDL (statement_timeout and deadlock_timeout)

(3) Vacuum replaying on standby and truncating data file

(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 unlock actions (for failed attempts of taking lock on master) 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

(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
```

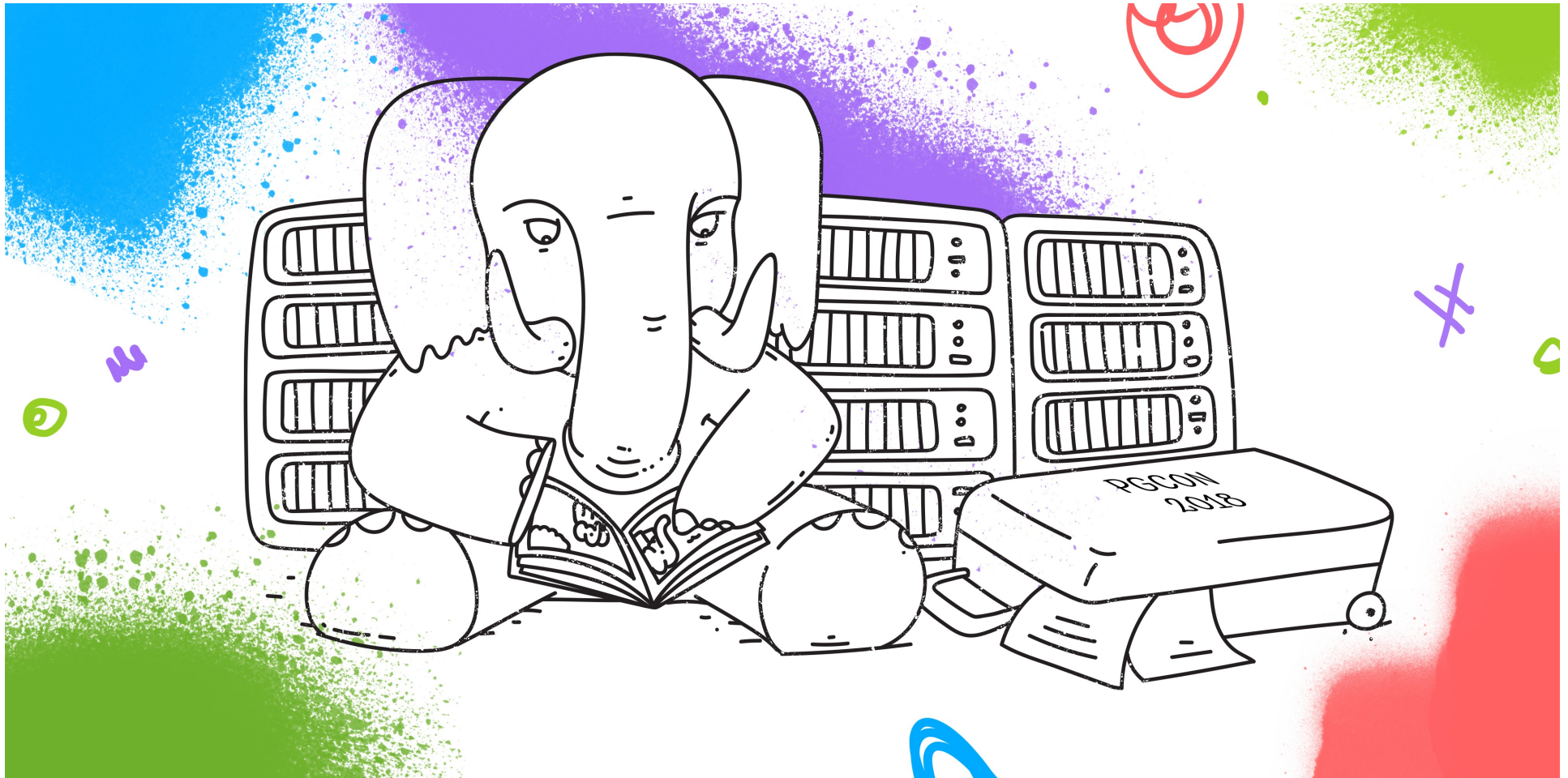
In our example there is 75 WAL files interval between first lock and success truncate (unlock relation)

(3) Vacuum replaying on standby and truncating data file

The solution can be like:

```
alter table xxx set (autovacuum_truncate = disable)
```

Thank you!



<https://github.com/avito-tech>

kevteev@avito.ru