Yandex

Yandex

Odyssey

Advanced multi-threaded PostgreSQL connection pooler and request router

Andrey Borodin, software engineer

Andrey Borodin

- Contributing to Postgres since 2016
- Yekaterinburg database meetup organizer

Working on

- disaster recovery system WAL-G
- connection pooler Odyssey
- interested in anything related to indexing

Yandex and PostgreSQL

Yandex.Mail

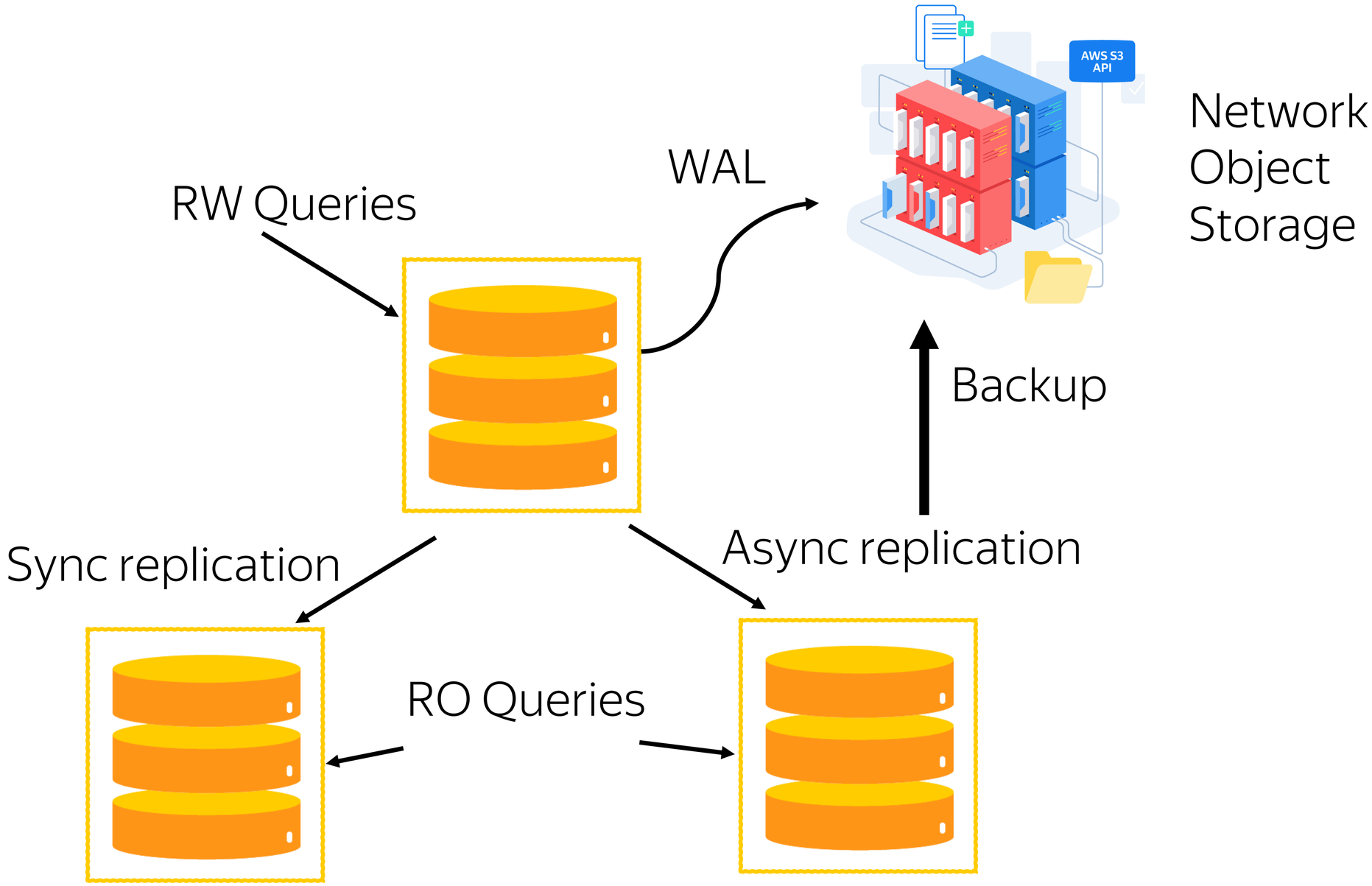
- some hundreds of millions of users
- 1+ trillion rows, 1+ million requests per second

Yandex.Cloud

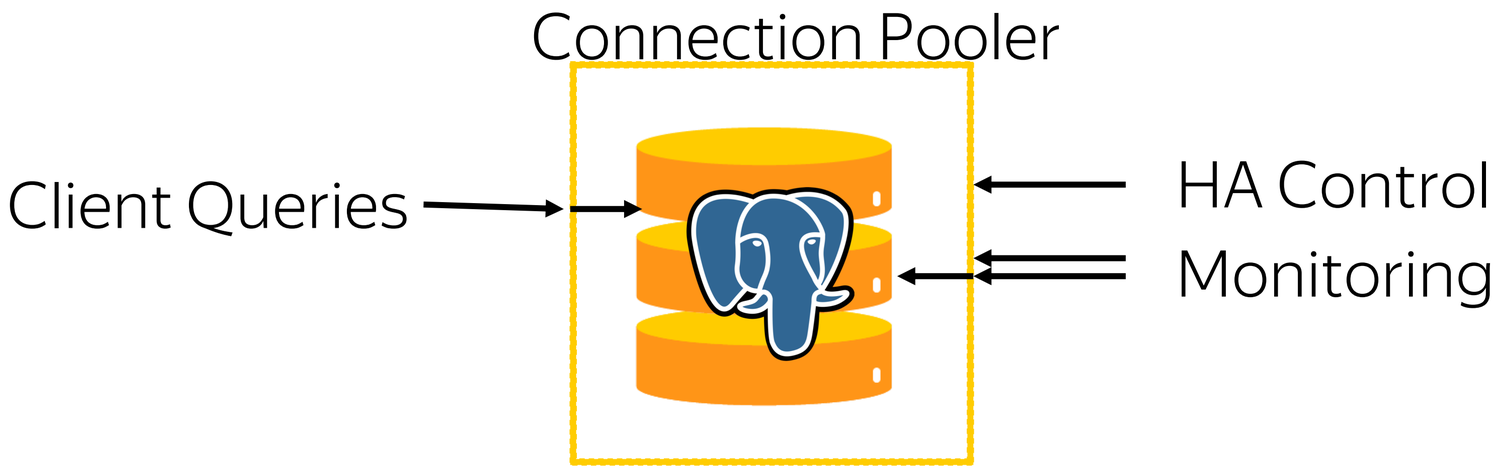
- ~2Pb of Postgres (May 2019)

And many other services like taxi, maps, weather forecast, carsharing, food delivery etc.

Cluster in the cloud



Node in a cluster



Why should we pool connections?

Why should we pool connections?

- 1 backend == 1 process

Why should we pool connections?

1 backend == 1 process
Caches per backend

- Relations cache
- Compiled PL\pgSQL
- Plans

Why should we pool connections?

1 backend == 1 process
Caches per backend

- Relations cache
- Compiled PL\pgSQL
- Plans

HA node fencing

OLTP throughput

```
Snapshot
GetSnapshotData(Snapshot snapshot)
{
        ...
                /*
                 * Spin over procArray checking xid, xmin, and subxids.  The goal is
                 * to gather all active xids, find the lowest xmin, and try to record
                 * subxids.
                 */
                numProcs = arrayP->numProcs;
                for (index = 0; index < numProcs; index++)
                {
                        int                     pgprocno = pgprocnos[index];
                        PGXACT     *pgxact = &allPgXact[pgprocno];
                        TransactionId xid;
```

Where can we pool connections?

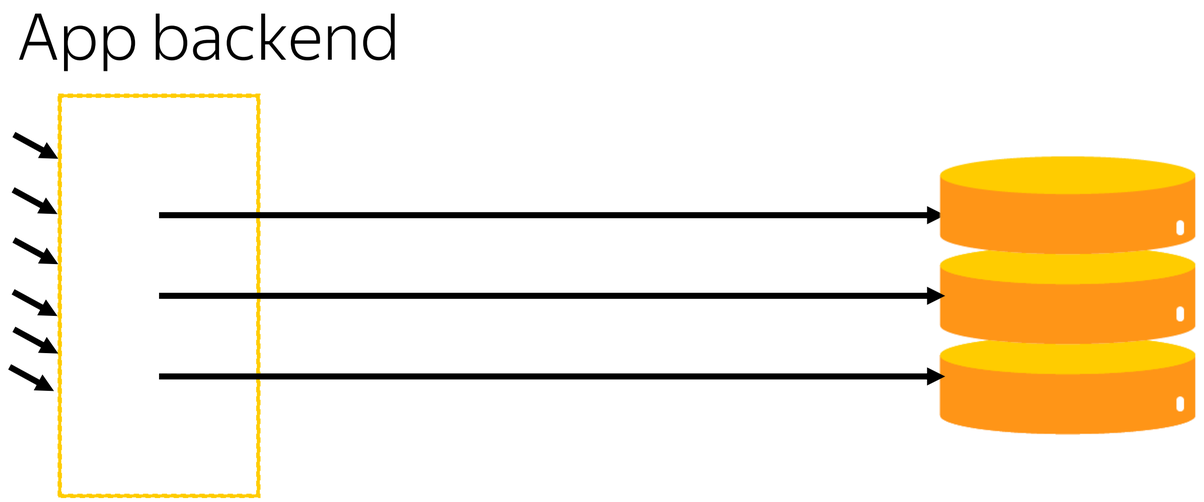
1 | Application-side pool

2 | Between app and DB

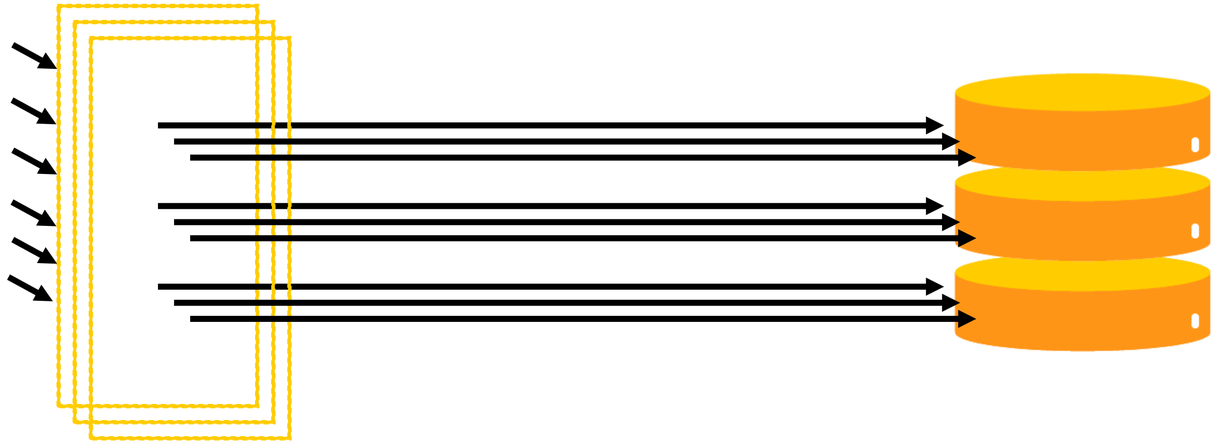
4 | DB built-in pooling

7 | Combinations

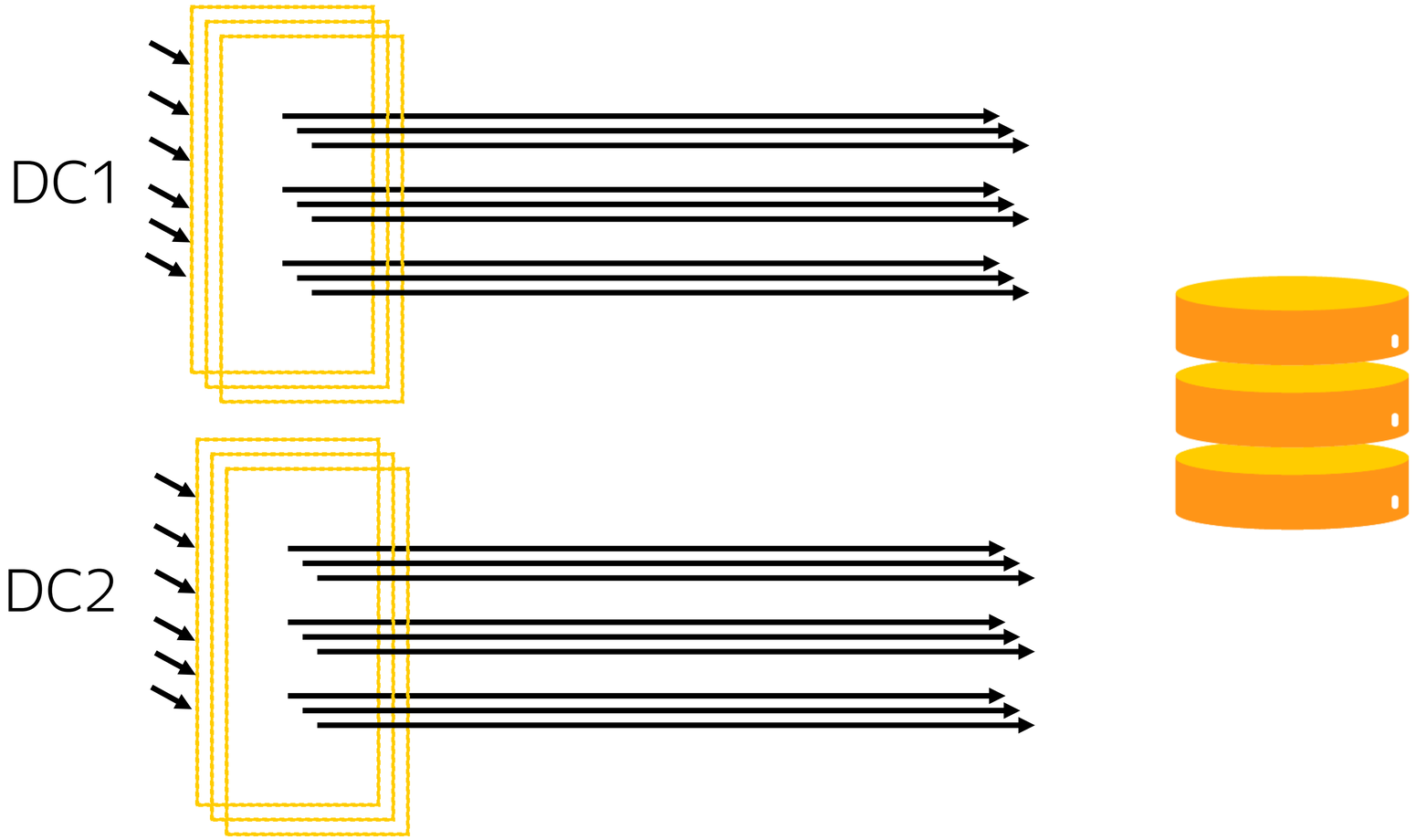
Application-side connection pooler



Backend under load balancer



In multiple availability zones



Proxy poolers

Pgpool II
Crunchy-Proxy

- Diverse functionality beyond pooling
- Only session pooling

PgBouncer

- Lightweight tool
- Transaction pooling

PgBouncer FTW

on our workload

Houston,

we have a problem

Diagnostics is complicated

```
miscdb01d/postgres M # SELECT client_addr, count(*)
FROM pg_stat_activity GROUP BY client_addr;
 client_addr | count
-------------+-------
 127.0.0.1   |   127
 ::1         |   136
(2 rows)
Time: 2.209 ms
miscdb01d/postgres M #
```

Diagnostics is complicated

Hard to trace

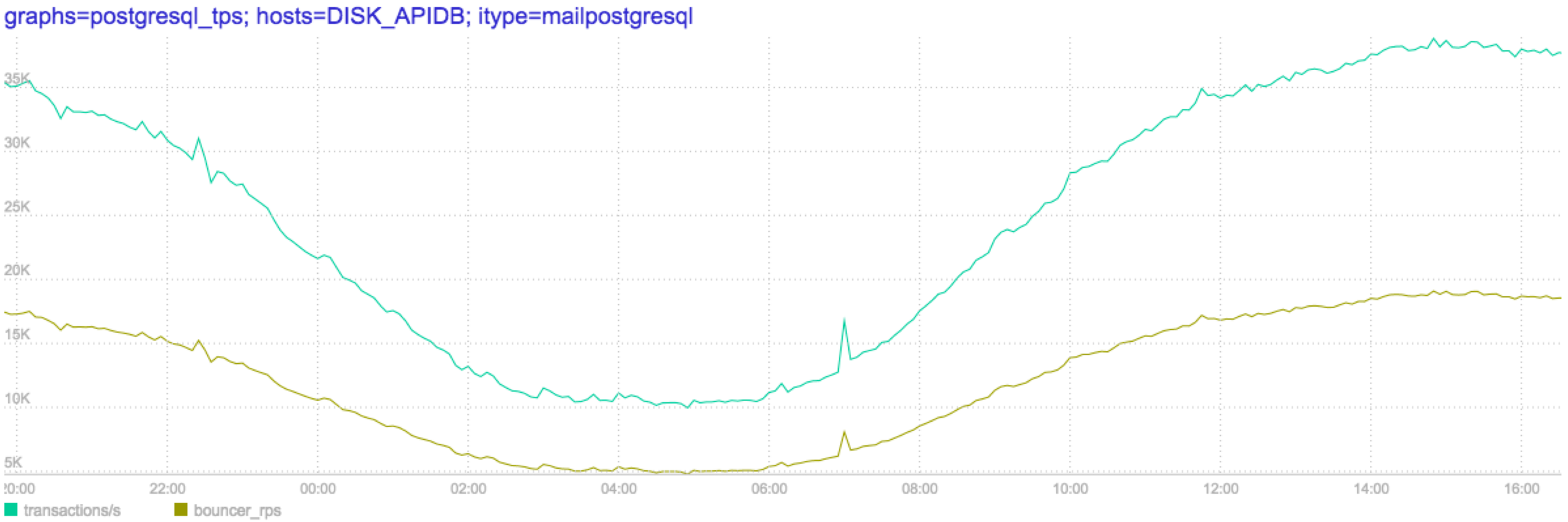
- Network problems
- Client driver problems

Hard to trace events of single session

application_name_add_host

```
miscdb01d/postgres M # SELECT client_addr, client_port, application_name
FROM pg_stat_activity LIMIT 1;
-[ RECORD 1 ]----+-----------------------------------------
client_addr      | 127.0.0.1
client_port      | 42051
application_name | app - [2a02:6b8:0:f12:225:90ff:fe94:155c]:50184
Time: 2.716 ms
miscdb01d/postgres M #
```

application_name_add_host



max_client_pool_conn

No way to limit connection count for specific database+user

```
        key         | value
--------------------+-------
 max_client_conn    | 20000
 default_pool_size  | 500
 min_pool_size      | 0
 reserve_pool_size  | 0
```

max_client_pool_conn

One client is opening `max_client_conn` connections and others will wait

```
2017-03-13 10:36:11.671 28152 LOG C-0x1350dd0:
  (nodb)/(nouser)@[2a02:6b8:0:1a71::21a0]:55760 closing because: no more
  connections allowed (max_client_conn) (age=0)

2017-03-13 10:36:11.671 28152 WARNING C-0x1350dd0:
  (nodb)/(nouser)@[2a02:6b8:0:1a71::21a0]:55760 Pooler Error: no more
  connections allowed (max_client_conn)
```

max_client_pool_conn

So, we patched PgBouncer

```
          key            | value
-------------------------+-------
 max_client_conn         | 20000
 max_client_pool_conn    | 4000
 default_pool_size       | 500
 min_pool_size           | 0
 reserve_pool_size       | 0
```

Pgbouncer cannot connect to server

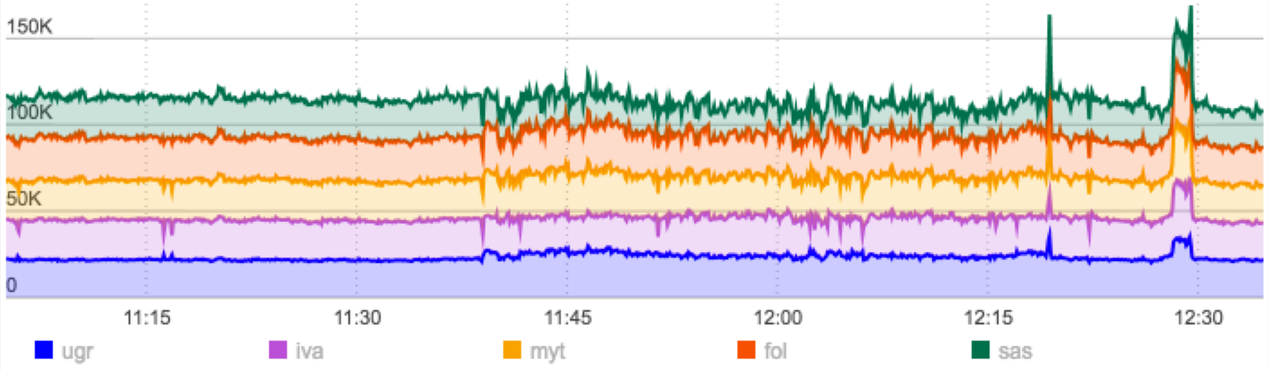
We can limit user in PostgreSQL:

```
› ALTER ROLE XXX WITH CONNECTION LIMIT 200;
› ALTER ROLE YYY WITH CONNECTION LIMIT 10;
```

Pgbouncer cannot connect to server

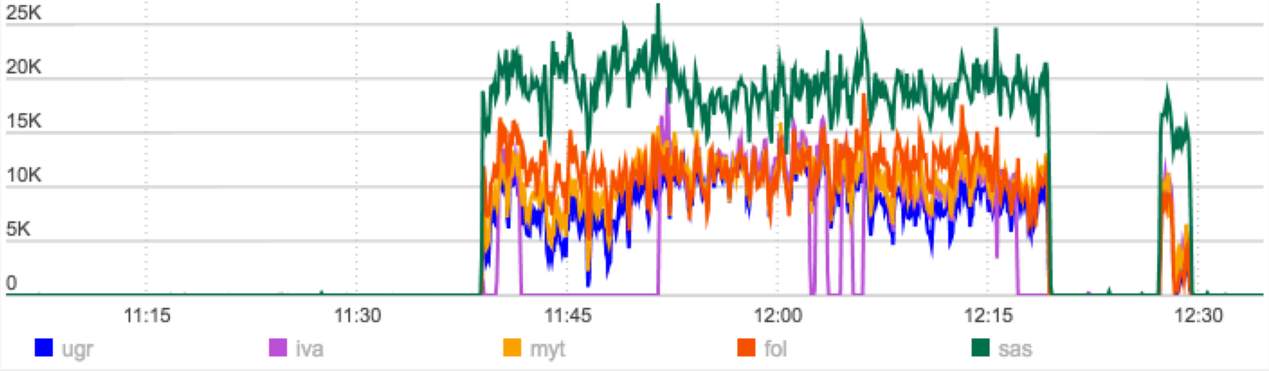
```
2017-03-13 10:48:23.995 24408 ERROR S: login failed: FATAL: too many connections for role
"YYY"

psycopg2.OperationalError: ERROR:  pgbouncer cannot connect to server

>>> try:

...     conn = psycopg2.connect("port=6432 ...")

... except psycopg2.Error as e:

...     print(e.pgcode)

...

None

>>>
```

HTTP 20x codes by DC

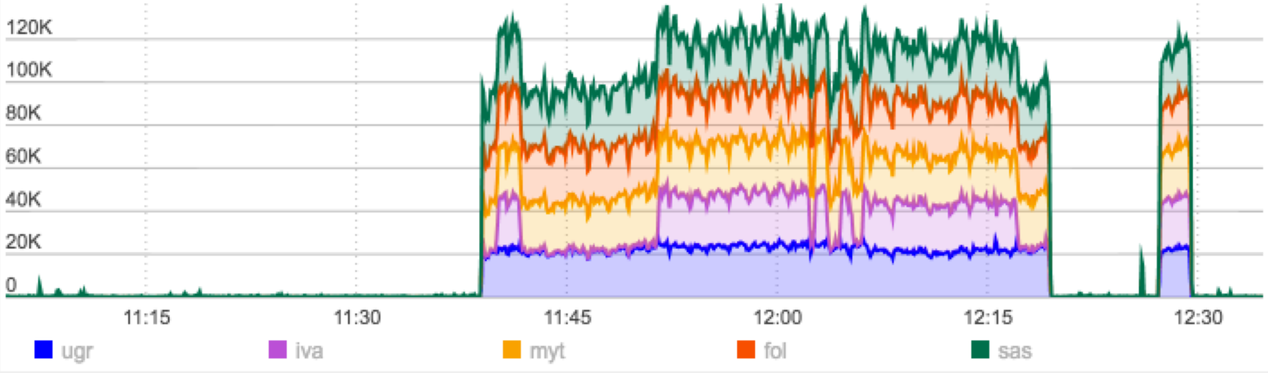


HTTP timings

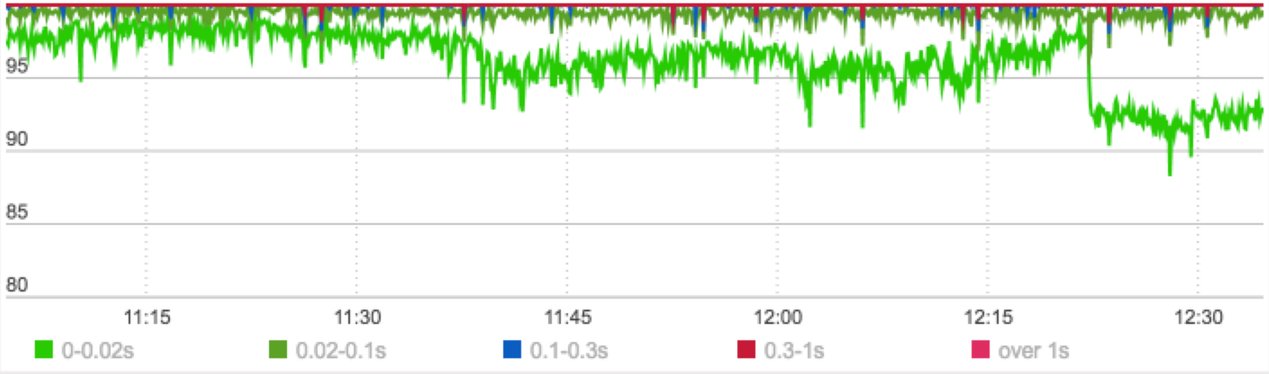
HTTP 5xx errors by DC



HTTP slow queries by DC



Passport timings



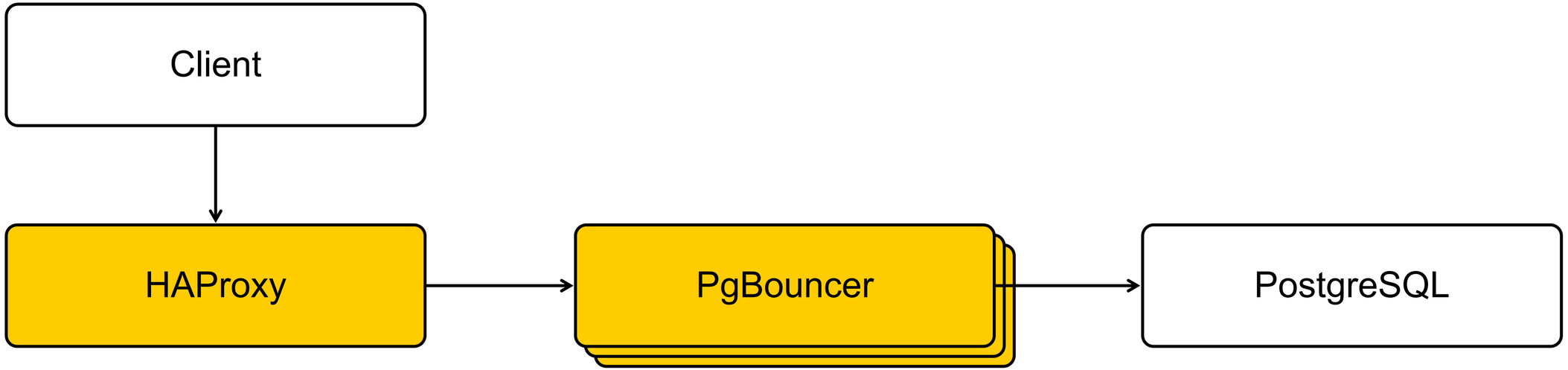
PostgreSQL timings

What’s going on?

```
  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 3548 pgbounce  10 -10 55344 16860   904 R 97.0  0.0 58h15:40 /usr/bin/pgbouncer -d -q /etc/pgbouncer/pgbouncer.ini
18561 postgres  20   0 66.1G  1804   712 S  7.0  0.0  5h51:48 postgres: wal writer process
21655 postgres  20   0 66.1G  3484  1876 S  5.0  0.0 50:00.07 postgres: wal sender process repl xivadb04d.mail.yandex.net(48473)
21688 postgres  20   0 66.1G  3484  1876 S  5.0  0.0 49:56.82 postgres: wal sender process repl xivadb04g.mail.yandex.net(36924)
26749 root      20   0 15968  1836  1048 R  3.0  0.0  0:02.06 htop
```

We need more ~~gold~~ PgBouncers

HAProxy



HAProxy

+ Pros

- Transparent for client
- Existing tools

- Cons

- No client IP again
- One more moving part
- HAProxy does not speak **proto3**
- **Problems with depleted sockets**

SO_REUSEPORT

https://lwn.net/Articles/542629/

```
+   if (af != AF_UNIX && cf_listen_reuseport == 1) {
+       int val = 1;
+       errpos = "setsockopt";
+       res = setsockopt(sock, SOL_SOCKET, SO_REUSEPORT, &val, sizeof(val));
+       if (res < 0)
+           goto failed;
+   }
```

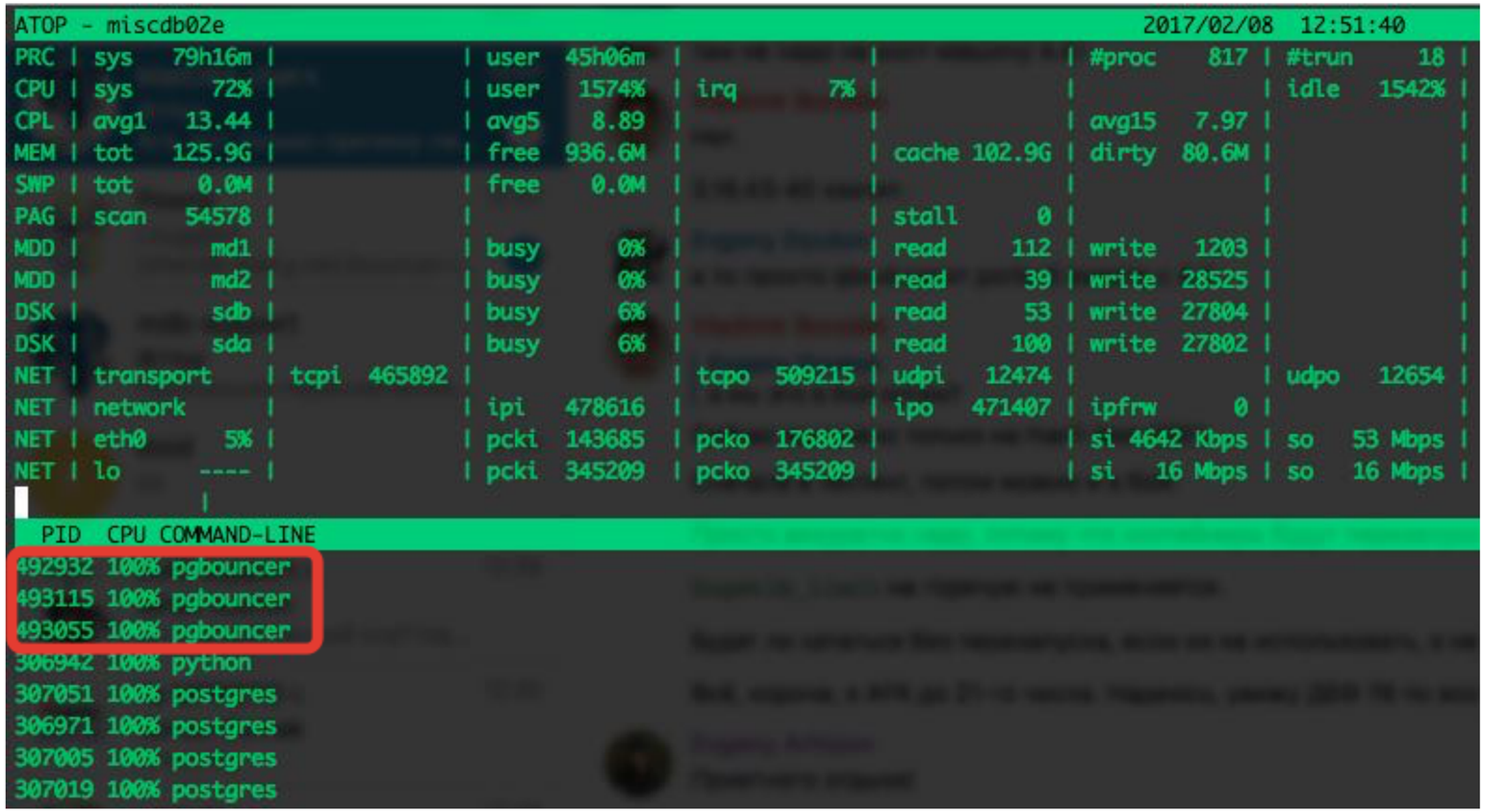
SO_REUSEPORT

SO_REUSEPORT

- Transparent for clients
- No extra moving parts

- Fragmentation of idle connections among PgBouncers

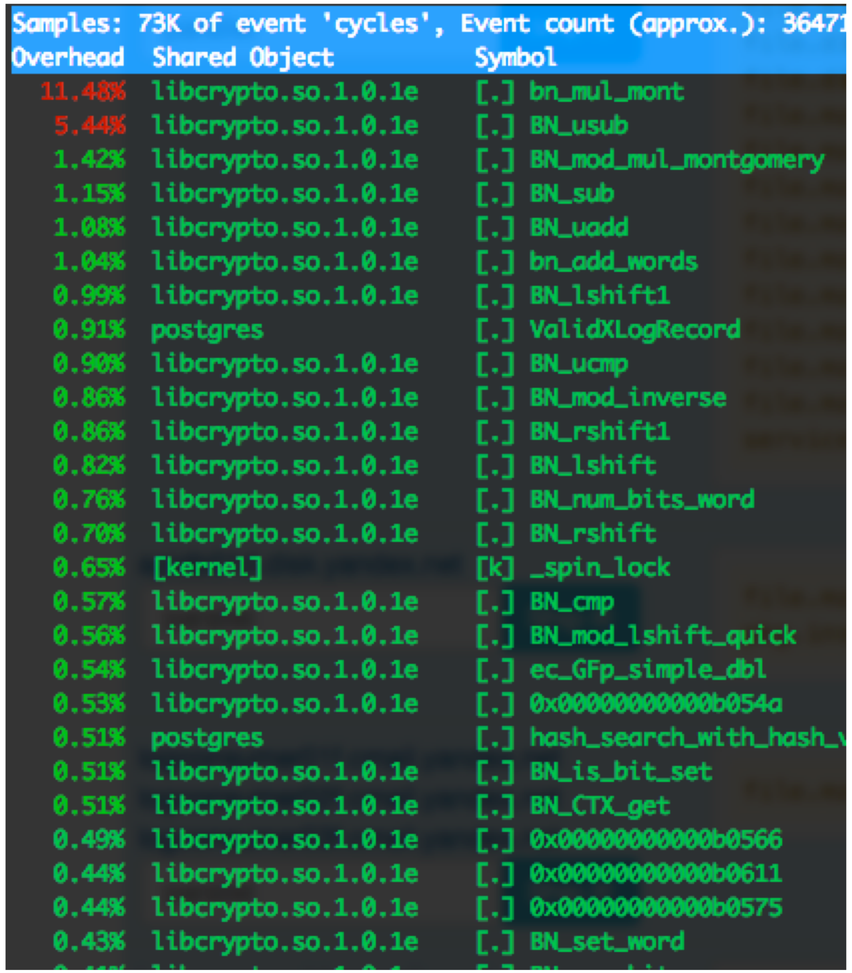
TLS


ATOP - miscdb02e 2017/02/08 12:51:40
PRC | sys 79h16m | user 45h06m | #proc 817 | #trun 18 |
CPU | sys 72% | user 1574% | irq 7% | idle 1542% |
CPL | avg1 13.44 | avg5 8.89 | avg15 7.97 |
MEM | tot 125.9G | free 936.6M | cache 102.9G | dirty 80.6M |
SWP | tot 0.0M | free 0.0M |
PAG | scan 54578 | stall 0 |
MDD | md1 | busy 0% | read 112 | write 1203 |
MDD | md2 | busy 0% | read 39 | write 28525 |
DSK | sdb | busy 6% | read 53 | write 27804 |
DSK | sda | busy 6% | read 100 | write 27802 |
NET | transport | tcpi 465892 | tcpo 509215 | udpi 12474 | udpo 12654 |
NET | network | ipi 478616 | ipo 471407 | ipfrw 0 |
NET | eth0 5% | pcki 143685 | pcko 176802 | si 4642 Kbps | so 53 Mbps |
NET | lo ---- | pcki 345209 | pcko 345209 | si 16 Mbps | so 16 Mbps |
PID CPU COMMAND-LINE
492932 100% pgbouncer
493115 100% pgbouncer
493055 100% pgbouncer
306942 100% python
307051 100% postgres
306971 100% postgres
307005 100% postgres
307019 100% postgres

TLS

```
$ pgbench -C -T 30 -j 300 -c 300 -S
  postgresql://127.0.0.1:6432/pgbench?sslmode=disable
<…>
latency average: 26.101 ms
tps = 11484.521542 (including connections establishing)

$ pgbench -C -T 30 -j 300 -c 300 -S
  postgresql://127.0.0.1:6432/pgbench?sslmode=require
<…>
latency average: 523.895 ms
tps = 566.809760 (including connections establishing)
```

TLS


Samples: 73K of event 'cycles', Event count (approx.): 3647
Overhead Shared Object Symbol
11.48% libcrypto.so.1.0.1e [.] bn_mul_mont
5.44% libcrypto.so.1.0.1e [.] BN_usub
1.42% libcrypto.so.1.0.1e [.] BN_mod_mul_montgomery
1.15% libcrypto.so.1.0.1e [.] BN_sub
1.08% libcrypto.so.1.0.1e [.] BN_uadd
1.04% libcrypto.so.1.0.1e [.] bn_add_words
0.99% libcrypto.so.1.0.1e [.] BN_lshift1
0.91% postgres [.] ValidXLogRecord
0.90% libcrypto.so.1.0.1e [.] BN_ucmp
0.86% libcrypto.so.1.0.1e [.] BN_mod_inverse
0.86% libcrypto.so.1.0.1e [.] BN_rshift1
0.82% libcrypto.so.1.0.1e [.] BN_lshift
0.76% libcrypto.so.1.0.1e [.] BN_num_bits_word
0.70% libcrypto.so.1.0.1e [.] BN_rshift
0.65% [kernel] [k] _spin_lock
0.57% libcrypto.so.1.0.1e [.] BN_cmp
0.56% libcrypto.so.1.0.1e [.] BN_mod_lshift_quick
0.54% libcrypto.so.1.0.1e [.] ec_GFp_simple_dbl
0.53% libcrypto.so.1.0.1e [.] 0x00000000000b054a
0.51% postgres [.] hash_search_with_hash_
0.51% libcrypto.so.1.0.1e [.] BN_is_bit_set
0.51% libcrypto.so.1.0.1e [.] BN_CTX_get
0.49% libcrypto.so.1.0.1e [.] 0x00000000000b0566
0.44% libcrypto.so.1.0.1e [.] 0x00000000000b0611
0.44% libcrypto.so.1.0.1e [.] 0x00000000000b0575
0.43% libcrypto.so.1.0.1e [.] BN_set_word

TLS

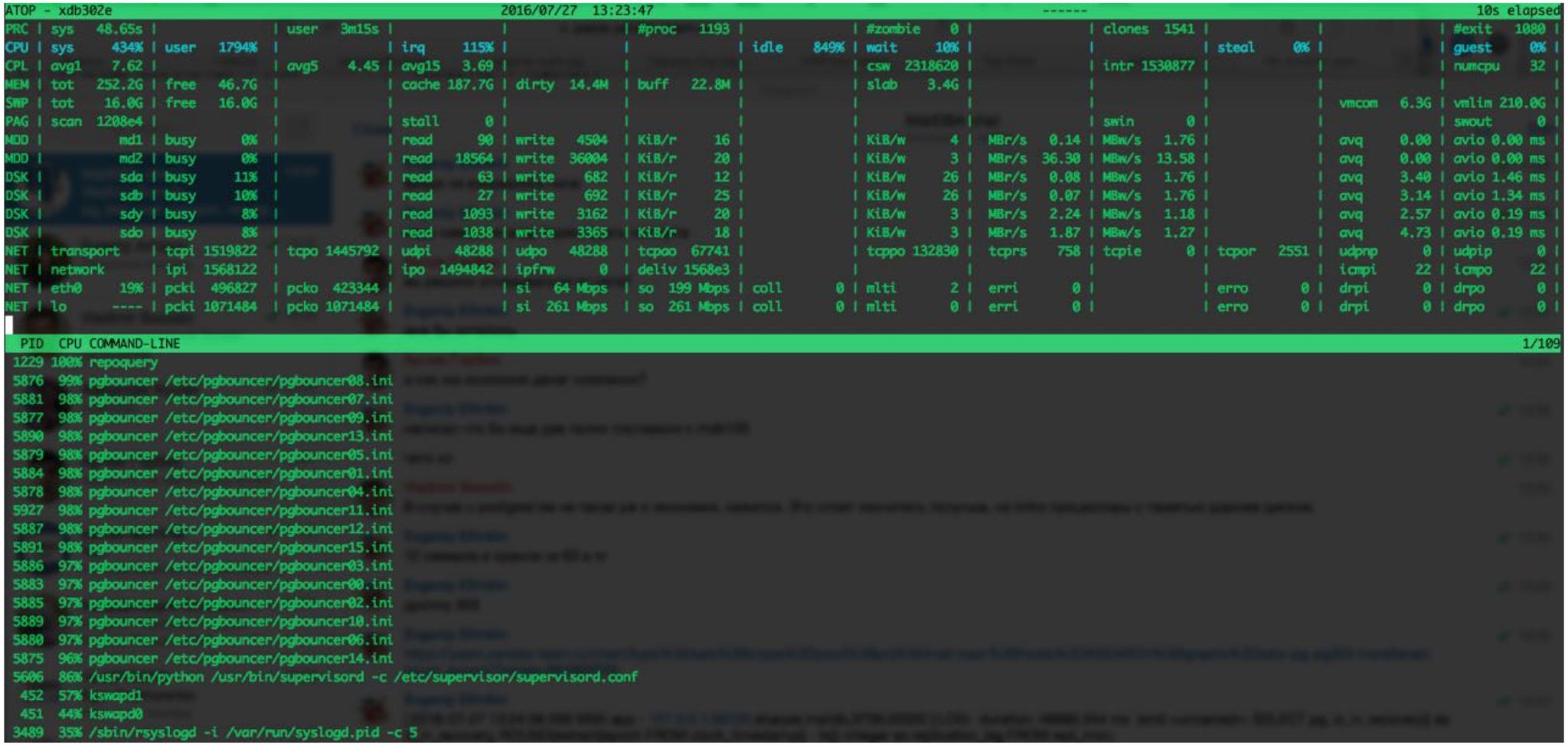
When the node is opened – connections startups are coordinated

- TLS hadshake explosion

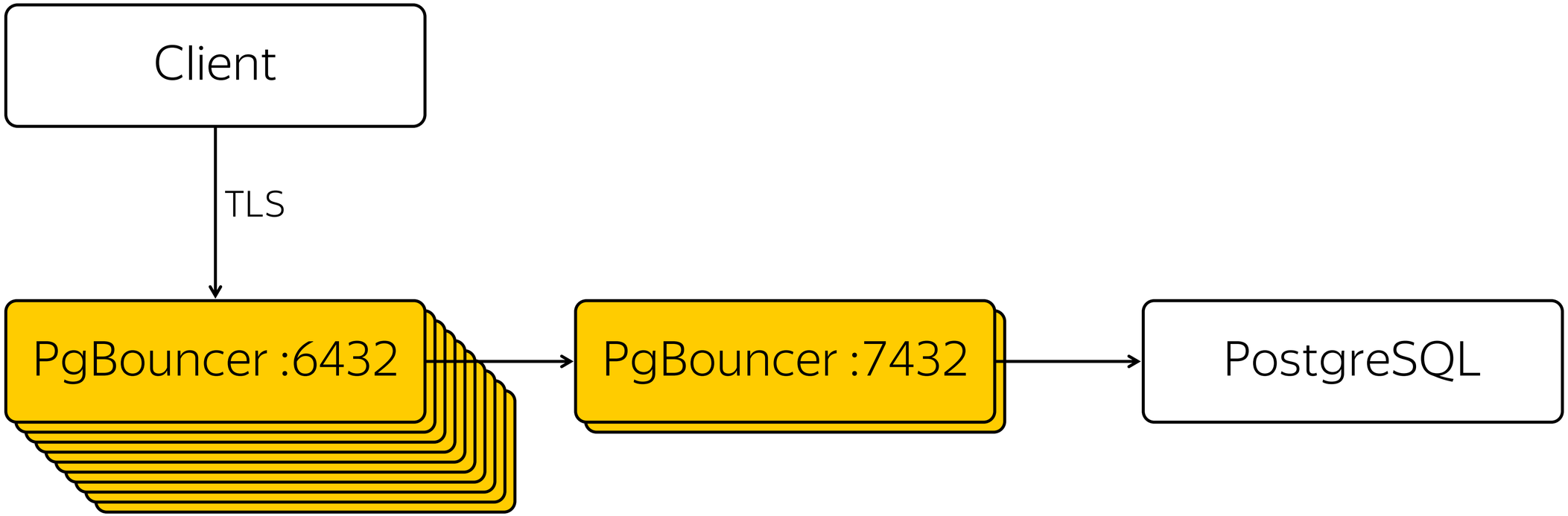
Some clients have small **connect_timeout**

- Clients retry, pgbouncer burns CPU

TLS


ATOP - xdb302e 2016/07/27 13:23:47 ------ 10s elapsed
PRC | sys 48.65s | user 3m15s | #proc 1193 | #zombie 0 | clones 1541 | #exit 1080 |
CPU | sys 434% | user 1794% | irq 115% | idle 849% | wait 10% | steal 0% | guest 0% |
CPL | avg1 7.62 | avg5 4.45 | avg15 3.69 | csw 2318620 | intr 1530877 | numcpu 32 |
MEM | tot 252.2G | free 46.7G | cache 187.7G | dirty 14.4M | buff 22.8M | slab 3.4G | vmcom 6.3G | vmlim 210.0G |
SWP | tot 16.0G | free 16.0G | swin 0 | swout 0 |
PAG | scan 1208e4 | stall 0 |
MDD | md1 | busy 0% | read 90 | write 4504 | KiB/r 16 | KiB/w 4 | MBr/s 0.14 | MBw/s 1.76 | avq 0.00 | avio 0.00 ms |
MDD | md2 | busy 0% | read 18564 | write 36004 | KiB/r 20 | KiB/w 3 | MBr/s 36.30 | MBw/s 13.58 | avq 0.00 | avio 0.00 ms |
DSK | sda | busy 11% | read 63 | write 682 | KiB/r 12 | KiB/w 26 | MBr/s 0.08 | MBw/s 1.76 | avq 3.40 | avio 1.46 ms |
DSK | sdb | busy 10% | read 27 | write 692 | KiB/r 25 | KiB/w 26 | MBr/s 0.07 | MBw/s 1.76 | avq 3.14 | avio 1.34 ms |
DSK | sdy | busy 8% | read 1093 | write 3162 | KiB/r 20 | KiB/w 3 | MBr/s 2.24 | MBw/s 1.18 | avq 2.57 | avio 0.19 ms |
DSK | sdo | busy 8% | read 1038 | write 3365 | KiB/r 18 | KiB/w 3 | MBr/s 1.87 | MBw/s 1.27 | avq 4.73 | avio 0.19 ms |
NET | transport | tcpi 1519822 | tcpo 1445792 | udpi 48288 | udpo 48288 | tcpao 67741 | tcppo 132830 | tcprs 758 | tcpie 0 | tcpor 2551 | udpnp 0 | udpip 0 |
NET | network | ipi 1568122 | ipo 1494842 | ipfrw 0 | deliv 1568e3 | icmpi 22 | icmpo 22 |
NET | eth0 19% | pcki 496827 | pcko 423344 | si 64 Mbps | so 199 Mbps | coll 0 | mlti 2 | erri 0 | erro 0 | drpi 0 | drpo 0 |
NET | lo ---- | pcki 1071484 | pcko 1071484 | si 261 Mbps | so 261 Mbps | coll 0 | mlti 0 | erri 0 | erro 0 | drpi 0 | drpo 0 |
PID CPU COMMAND-LINE 1/109
1229 100% repoquery
5876 99% pgbouncer /etc/pgbouncer/pgbouncer08.ini
5881 98% pgbouncer /etc/pgbouncer/pgbouncer07.ini
5877 98% pgbouncer /etc/pgbouncer/pgbouncer09.ini
5890 98% pgbouncer /etc/pgbouncer/pgbouncer13.ini
5879 98% pgbouncer /etc/pgbouncer/pgbouncer05.ini
5884 98% pgbouncer /etc/pgbouncer/pgbouncer01.ini
5878 98% pgbouncer /etc/pgbouncer/pgbouncer04.ini
5927 98% pgbouncer /etc/pgbouncer/pgbouncer11.ini
5887 98% pgbouncer /etc/pgbouncer/pgbouncer12.ini
5891 98% pgbouncer /etc/pgbouncer/pgbouncer15.ini
5886 97% pgbouncer /etc/pgbouncer/pgbouncer03.ini
5883 97% pgbouncer /etc/pgbouncer/pgbouncer00.ini
5885 97% pgbouncer /etc/pgbouncer/pgbouncer02.ini
5889 97% pgbouncer /etc/pgbouncer/pgbouncer10.ini
5880 97% pgbouncer /etc/pgbouncer/pgbouncer06.ini
5875 96% pgbouncer /etc/pgbouncer/pgbouncer14.ini
5606 86% /usr/bin/python /usr/bin/supervisord -c /etc/supervisor/supervisord.conf
452 57% kswapd1
451 44% kswapd0
3489 35% /sbin/rsyslogd -i /var/run/syslogd.pid -c 5

Cascading PgBouncers



Cascading PgBouncers

- Still transparent for client
- Withstand any load peak
- Control over idle connection count
- Smooth restart

- Maintenance is difficult
- No control over distribution of load by instances of PgBouncers

Looks OK.
How to open source this?

Cancel running query

Client of healthy user

- Opens new connection w\o auth
- Call PQcancel, with secret token from backend
- postgresql.org/docs/current/static/libpq-cancel.html

Smoker's client

- Just send TCP reset

github.com/pgbouncer/pgbouncer/pull/79

What do we want?

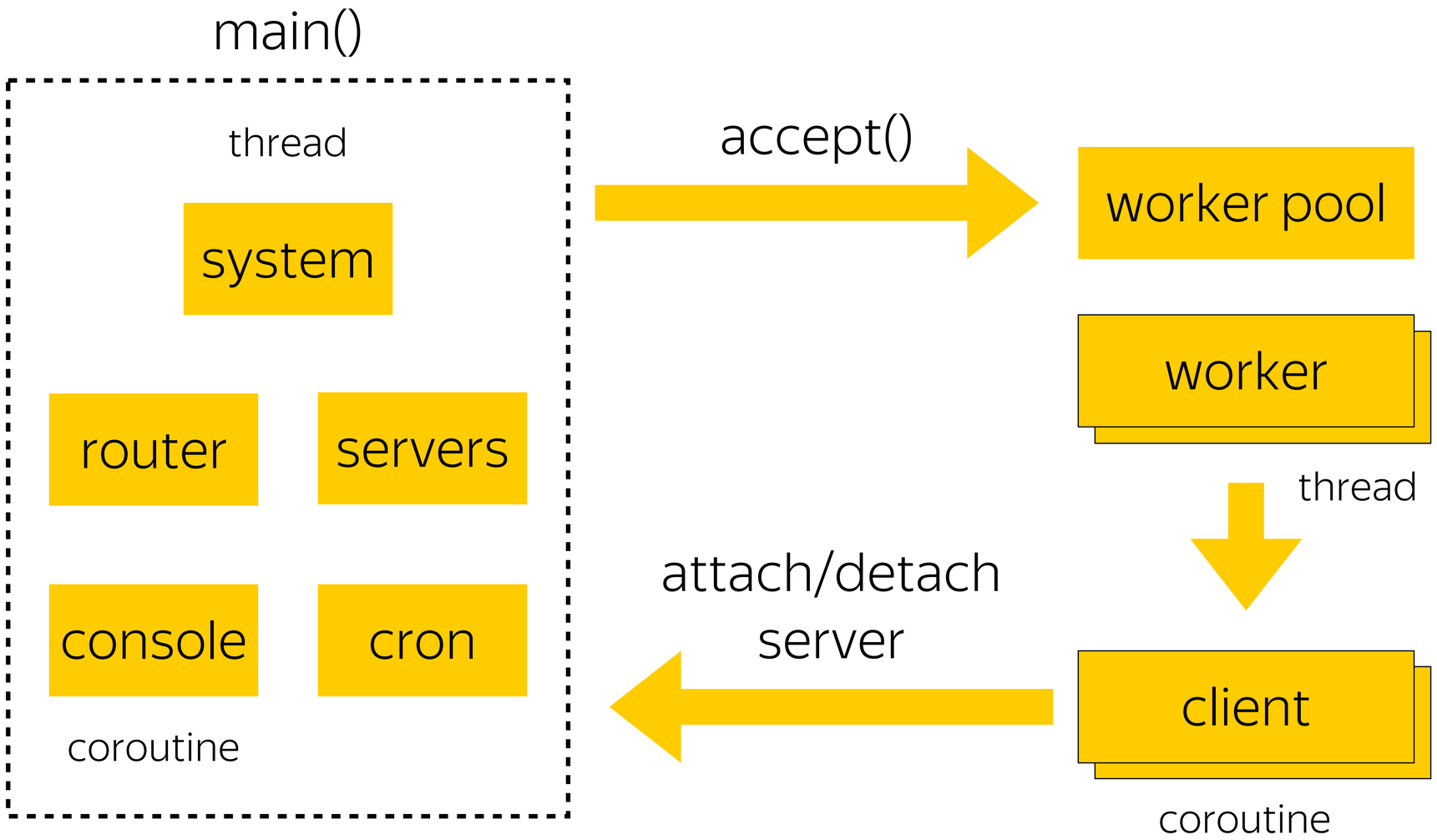
- Controllable CPU scaling
- Flexible tuning
- Tracing client session
- Mixed pooling types
- Better error codes forwarding

Odyssey

Compilation

- Linux x86, x86_64
- C99
- cmake, gcc/clang
- Depends only on openssl
- One config file
  `./odyssey <config_file>`

Internal architecture



Multithreading

- Machinarium: workers and coroutines
- Independent epoll(7) context for each worker

Multithreading details

- Accept(2) in separate thread
- Pipelining small packets
- Cache-friendly pipelining
- Optimization for special case workers = 1

Odyssey features

- Enhanced transaction pooling
- **CANCEL queries that no one waits**

Enhanced transaction pooling

- Trying to keep server connection
- Automatic ROLLBACK
- Automatic CANCEL
- Optimization of parameter setup (SET, DISCARD)

Odyssey features

- Replication support
- Clients can migrate FROM your cloud managed services

Odyssey features

- PgBouncer console compatibility

Does your monitoring look into ‘SHOW SERVERS’?

Odyssey features

- Error forwarding
- Easier to handle overload

Logging and error forwarding

```
client_fwd_error off
$ psql "dbname=test host=localhost port=6432"
psql: ERROR:  odyssey: c9259d96414b9: failed to connect to
remote server sce469f2305d9


client_fwd_error on
$ psql "dbname=test host=localhost port=6432"
psql: FATAL:  odyssey: cbde3e23d9aa2: database "test"
does not exist
```

Logging and error forwarding

```
log_format "%p %t %l [%i %s] (%c) %m \n"
```

```
4249 17 Jun 17:32:27.604 info [cbde3e23d9aa2 none] (startup) new client connection [::1]:50676
4249 17 Jun 17:32:27.604 info [cbde3e23d9aa2 none] (startup) route 'test.pmwkaa' to 'default.default'
4249 17 Jun 17:32:27.604 info [cbde3e23d9aa2 sa6a53e6ec6d7] (setup) new server connection
127.0.0.1:5432
4249 17 Jun 17:32:27.607 error [cbde3e23d9aa2 sa6a53e6ec6d7] (startup) FATAL 3D000 database "test"
does not exist
```

Logging and error forwarding

```
client_fwd_error off
$ psql "dbname=test host=localhost port=6432"
psql: ERROR:  odyssey: c9259d96414b9: failed to connect to
remote server sce469f2305d9


client_fwd_error on
$ psql "dbname=test host=localhost port=6432"
psql: FATAL:  odyssey: cbde3e23d9aa2: database "test"
does not exist
```

Route settings

```
storage "postgres_server" {
    type "remote"
    host "127.0.0.1"
    port 5432
    tls "disable"
}
```

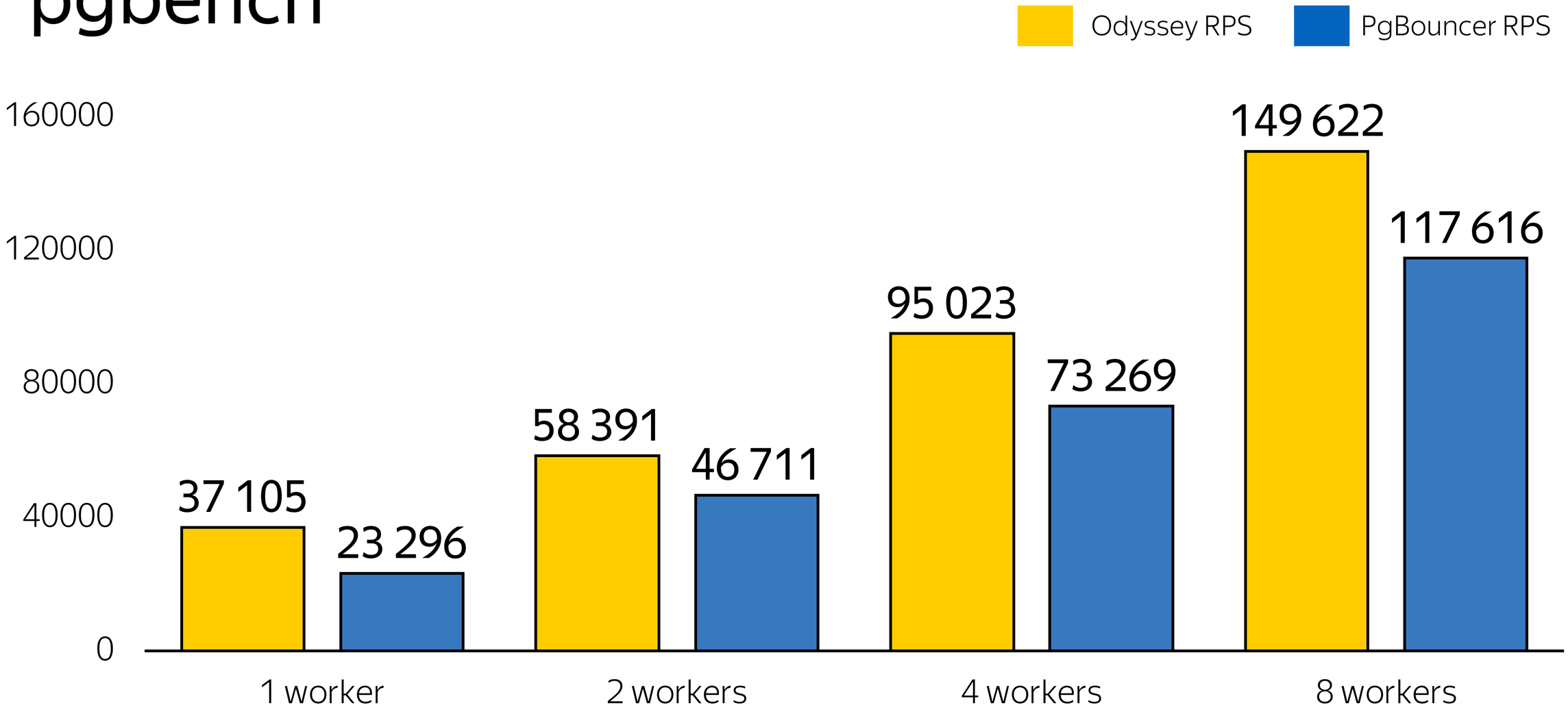
Route settings

```
database "test" {
    user "test" {
        storage "postgres_server"
        authentication "none"
        client_max 100
        pool "transaction"
        pool_size 10
        pool_cancel yes
        pool_rollback yes
    }
    user default {
         authentication "block"
    }
}
```

Route settings

```
database default {
    user default {
        authentication "block"
    }
}
```

pgbench



*Benchmark results depend on software, hardware and weather on the moon. Do not trust them.
**We optimized scaling, not throughput.

How we test

- PostgreSQL make install-check
- Drivers tests: pq, node-postgres, pgjdbc, psycopg2
- Unit-tests

How we test

- make install-check -> Odyssey -> PostgreSQL
- make install-check -> PgBouncer -> Odyssey -> PostgreSQL

Roadmap

- SCRAM authentication
- Forward read-only queries to replica
- Online restart
- Pause server
- ...
- **Pull requests are welcome!**

Andrey Borodin

Waiting for questions ☺

 x4mmm@yandex-team.ru

 x4mmm