Announcing pg_statviz

Jimmy Angelakos

Senior Solutions Architect

EDB

PGDay Chicago

2023-04-20

Announcing pg_statviz

- A minimalist **extension and utility pair**
- Time series analysis and visualization of **PostgreSQL** internal statistics

PostgreSQL internal statistics

- The **Cumulative Statistics System** (FKA Statistics Collector)
  - Postgres subsystem that collects info about system activity
- Dynamic statistics (right now)
- Cumulative statistics, but can be reset
- Table/index information on row & disk block levels
- This info can be reported via views

Motivation

- Why?
  - Track PostgreSQL performance over time and potentially perform tuning or troubleshooting
- Yes, but why?
  - So that people can understand their system better at a glance 👀

Motivation ii

- Working with customers
  - Who often have no idea how their database is performing
  - Or why it’s not working well
- Their monitoring tools don’t give them insights

How?

- Created for:
  - Snapshotting **cumulative and dynamic** statistics
  - Performing **time series analysis** on them
- Utility can produce visualizations for selected time ranges on the stored stats snapshots

Design Philosophy

- **K.I.S.S.** and **UNIX** philosophies
- Tool aims to be:
  - Modular
  - Minimal
  - Unobtrusive
- Does only what it's meant for: create snapshots of PostgreSQL statistics for visualization and analysis.

Design Philosophy ii

- Not for live monitoring displays
  - But one could...
- Open schema, clearly defined
  - Data easily exportable
- No built-in scheduler
- No built-in data retention policy mechanism

Design

- Components
  - PostgreSQL extension
  - Python utility for retrieving stored snapshots & creating simple visualizations using **Matplotlib**
- Nothing to put in **shared_preload_libraries**
- No need to restart Postgres

Installation

- Extension installation:

  CREATE EXTENSION pg_statviz;

- Utility installation:

  $ pip install pg_statviz

Usage

- Extension can be used by superusers, or any user that has **pg_monitor** role privileges
- To take a snapshot, e.g. from **psql**:

SELECT pgstatviz.snapshot();

```
psql (15.1 (Ubuntu 15.1-1.pgdg22.04+1))
Type "help" for help.

faf=> SELECT pgstatviz.snapshot();
NOTICE:  created pg_statviz snapshot
          snapshot
---------------------------
 2023-04-20 14:15:14.5869+01
(1 row)

faf=>
```

Usage ii

```
vyruss@rancor:~$ pg_statviz --help
usage: pg_statviz [--help] [--version] [-d DBNAME] [-h HOSTNAME] [-p PORT] [-U USERNAME] [-W]
                  [-D FROM TO] [-O OUTPUTDIR]
                  {analyze,buf,cache,checkp,conn,tuple,wait,wal} ...

run all analysis modules

positional arguments:
  {analyze,buf,cache,checkp,conn,tuple,wait,wal}
    analyze             run all analysis modules
    buf                 run buffers written analysis module
    cache               run cache hit ratio analysis module
    checkp              run checkpoint analysis module
    conn                run connection count analysis module
    tuple               run tuple count analysis module
    wait                run wait events analysis module
    wal                 run WAL generation analysis module

options:
  --help
  --version             show program's version number and exit
  -d DBNAME, --dbname DBNAME
                        database name to analyze (default: 'vyruss')
  -h HOSTNAME, --host HOSTNAME
                        database server host or socket directory (default: '/var/run/postgresql')
  -p PORT, --port PORT  database server port (default: '5432')
  -U USERNAME, --username USERNAME
                        database user name (default: 'vyruss')
  -W, --password        force password prompt (should happen automatically) (default: False)
  -D FROM TO, --daterange FROM TO
                        date range to be analyzed in ISO 8601 format e.g. 2023-01-01T00:00
                        2023-01-01T23:59 (default: [])
  -O OUTPUTDIR, --outputdir OUTPUTDIR
                        output directory (default: -)
vyruss@rancor:~$
```

Usage

iii

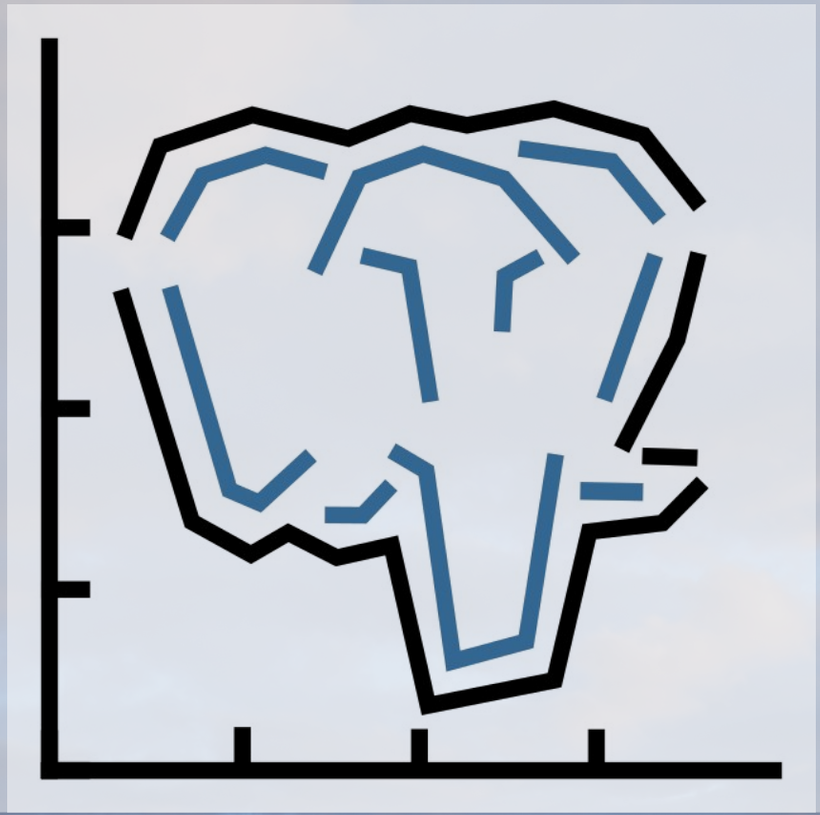
```
vyruss@rancor:~$ pg_statviz -d faf -U postgres -D 2023-01-17T23:00 2024-01-01
INFO:pg_statviz.modules.buf:Running buffers written analysis
INFO:pg_statviz.modules.buf:Saving pg_statviz_rancor_5432_buf.png
INFO:pg_statviz.modules.buf:Saving pg_statviz_rancor_5432_buf_rate.png
INFO:pg_statviz.modules.checkp:Running checkpoint analysis
INFO:pg_statviz.modules.checkp:Saving pg_statviz_rancor_5432_checkp.png
INFO:pg_statviz.modules.checkp:Saving pg_statviz_rancor_5432_checkp_rate.png
INFO:pg_statviz.modules.cache:Running cache hit ratio analysis
INFO:pg_statviz.modules.cache:Saving pg_statviz_rancor_5432_cache.png
INFO:pg_statviz.modules.conn:Running connection count analysis
INFO:pg_statviz.modules.conn:Saving pg_statviz_rancor_5432_conn_status.png
INFO:pg_statviz.modules.conn:Saving pg_statviz_rancor_5432_conn_user.png
INFO:pg_statviz.modules.tuple:Running tuple count analysis
INFO:pg_statviz.modules.tuple:Saving pg_statviz_rancor_5432_tuple.png
INFO:pg_statviz.modules.wait:Running wait events analysis
INFO:pg_statviz.modules.wait:Saving pg_statviz_rancor_5432_wait.png
INFO:pg_statviz.modules.wal:Running WAL generation analysis
INFO:pg_statviz.modules.wal:Saving pg_statviz_rancor_5432_wal.png
INFO:pg_statviz.modules.wal:Saving pg_statviz_rancor_5432_wal_rate.png
vyruss@rancor:~$
```

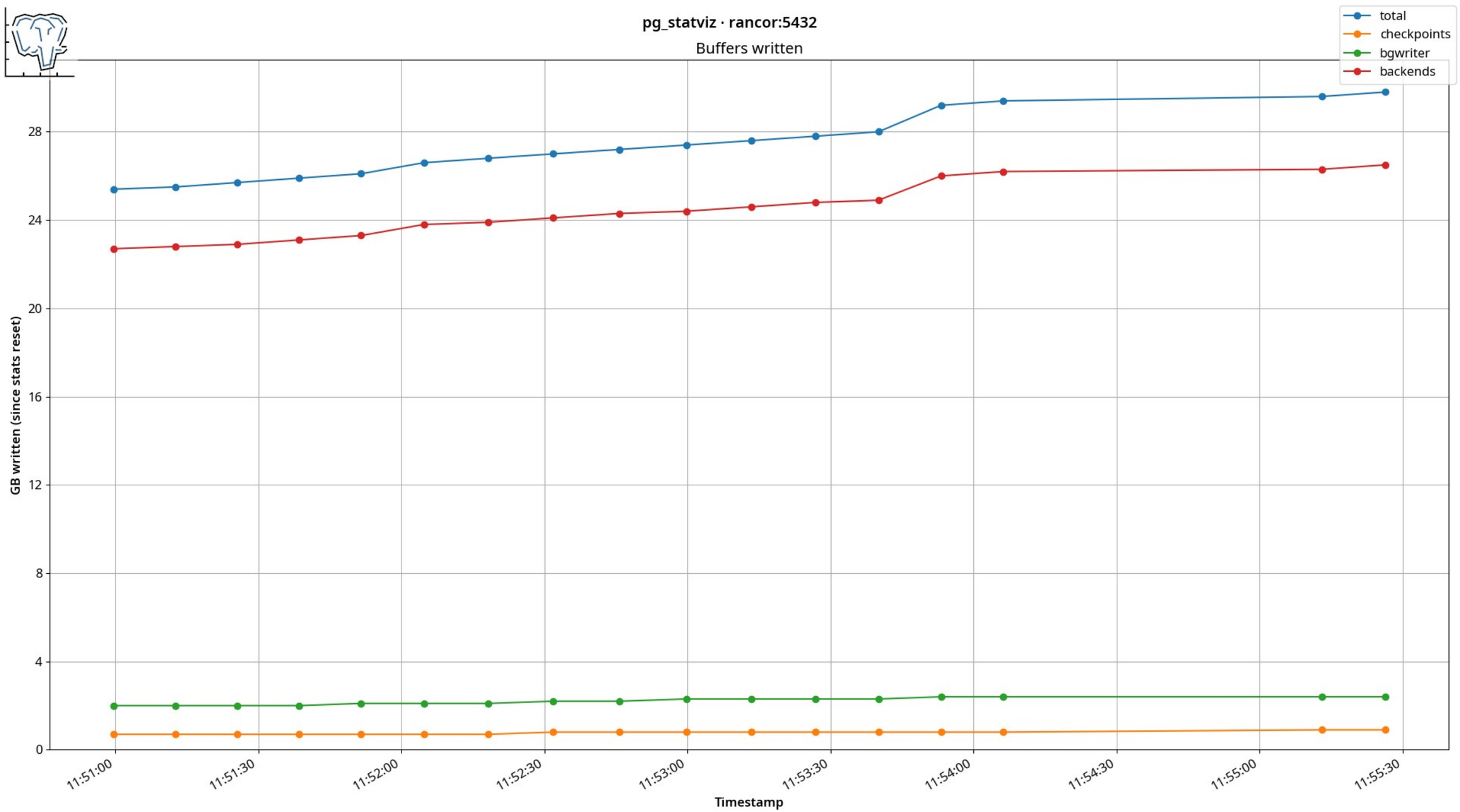
Modules

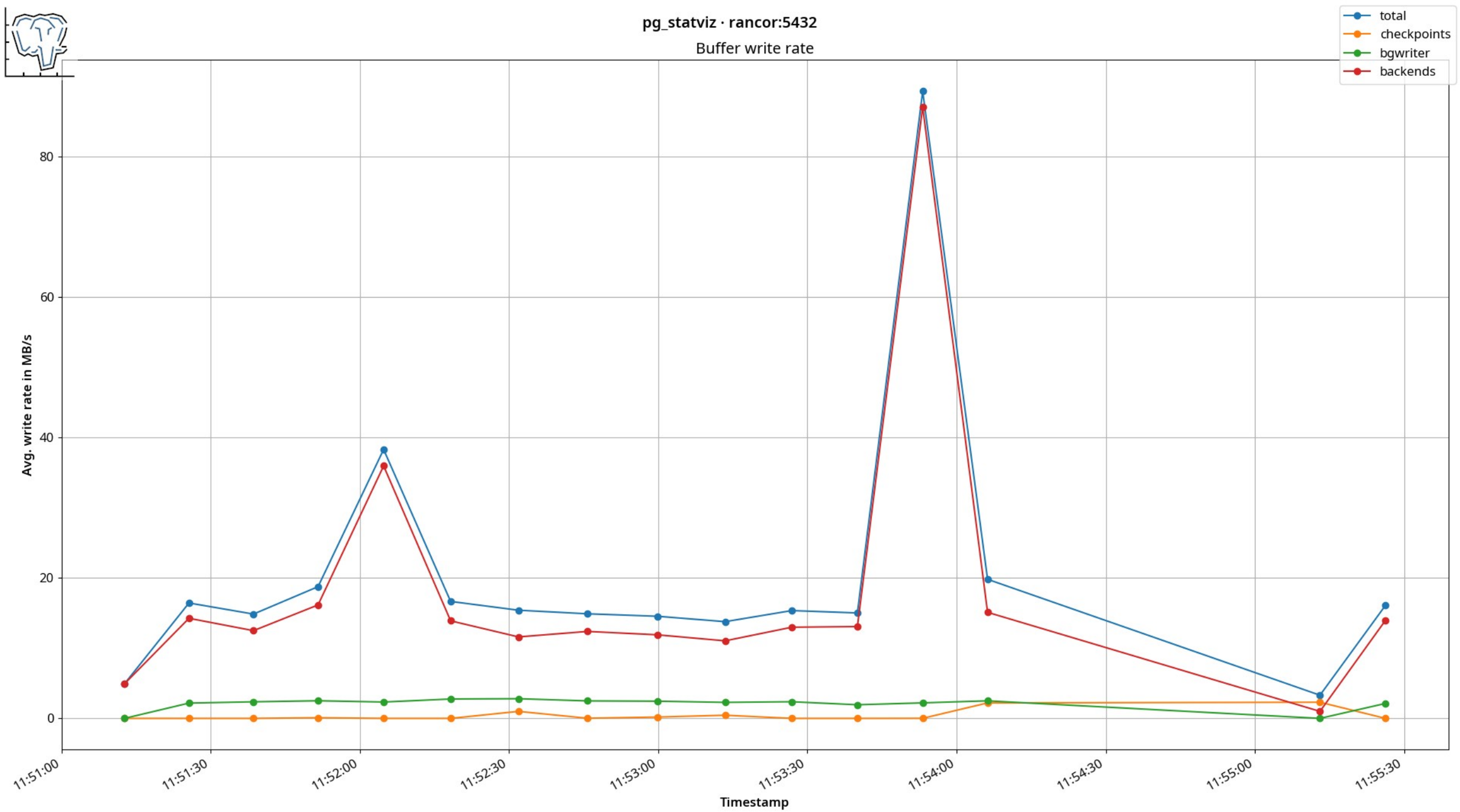
- **analyze** (default) - run all analysis modules
- **buf** - buffers written, buffer write rate
- **cache** - cache hit ratio
- **checkp** - checkpoint analysis, checkpoint rate
- **conn** - connection count, by status and by user
- **tuple** - tuple count analysis
- **wait** - wait events analysis
- **wal** - WAL generation analysis

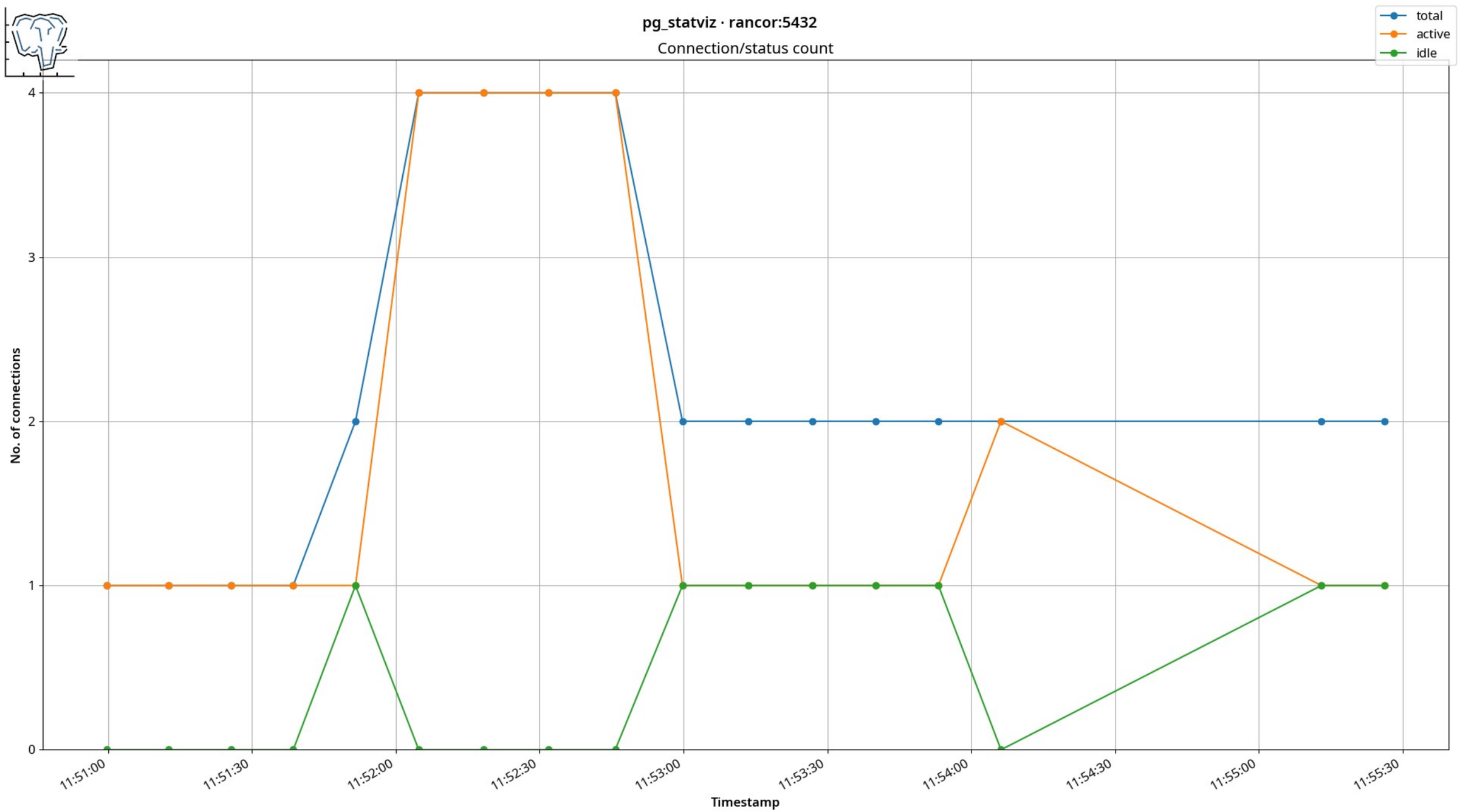
Obligatory

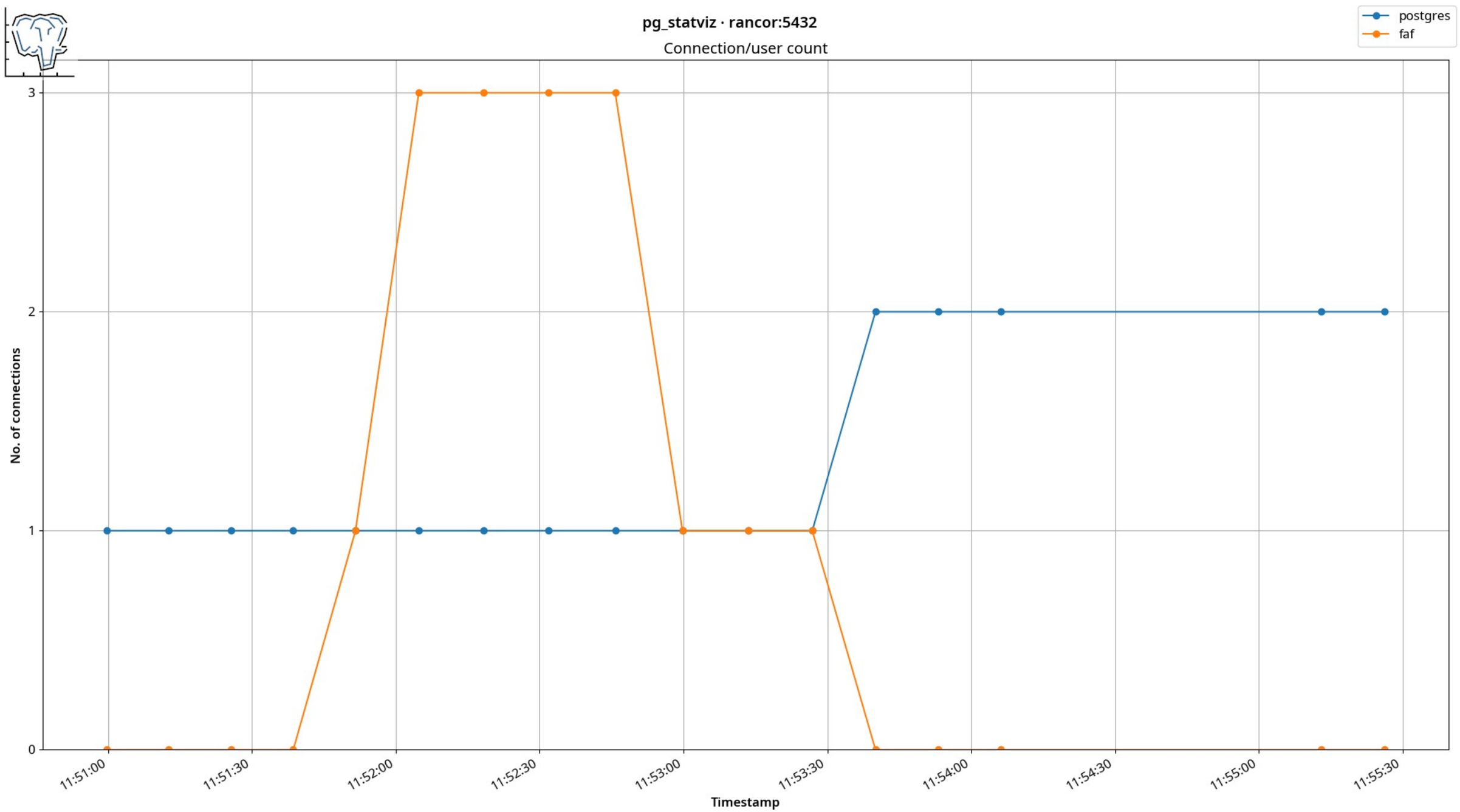
- And yes, it has a logo :)

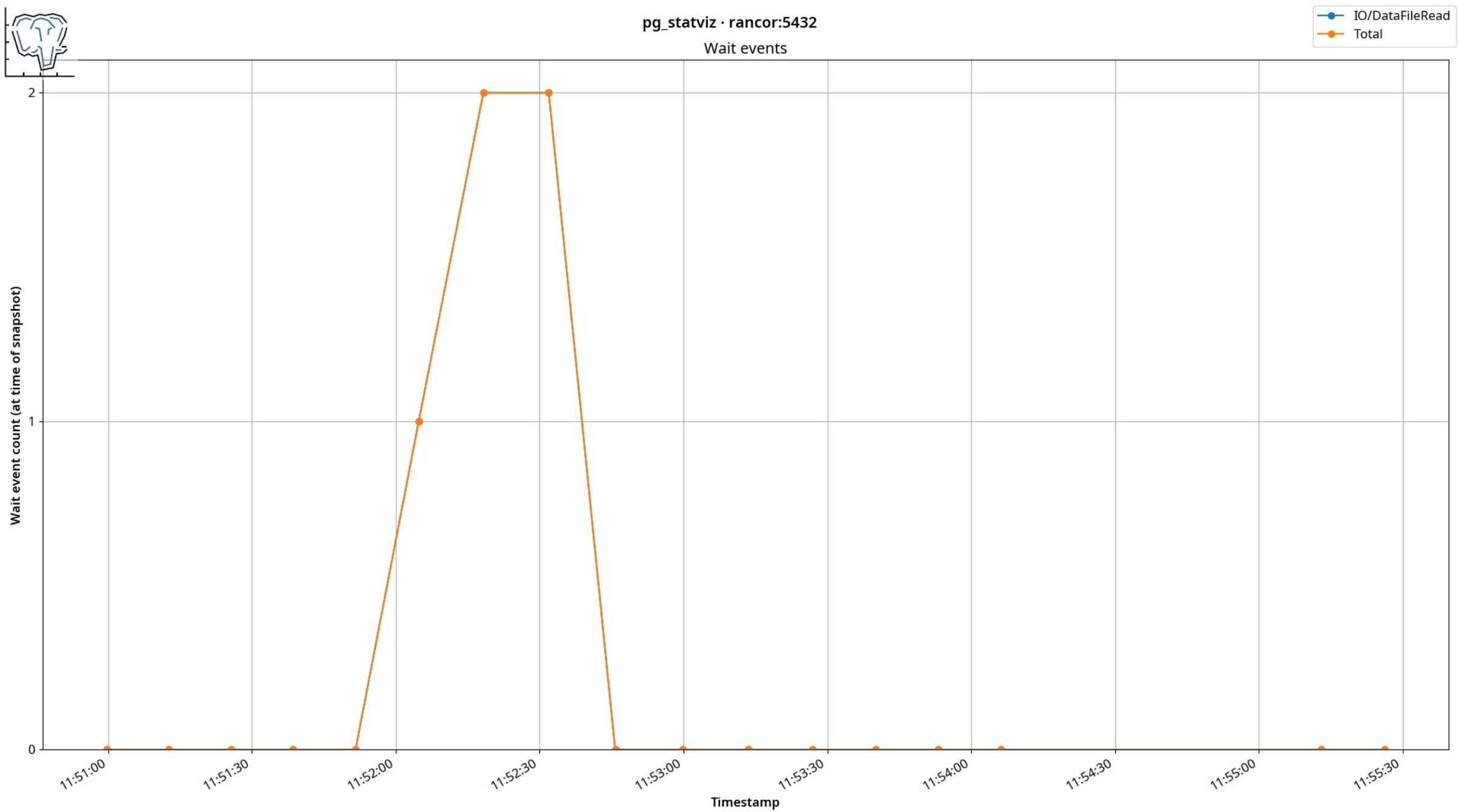



pg_statviz · rancor:5432
Buffers written
total
checkpoints
bgwriter
backends
GB written (since stats reset)
0
4
8
12
16
20
24
28
11:51:00
11:51:30
11:52:00
11:52:30
11:53:00
11:53:30
11:54:00
11:54:30
11:55:00
11:55:30
Timestamp


pg_statviz · rancor:5432
Buffer write rate
total
checkpoints
bgwriter
backends
Avg. write rate in MB/s
0
20
40
60
80
11:51:00
11:51:30
11:52:00
11:52:30
11:53:00
11:53:30
11:54:00
11:54:30
11:55:00
11:55:30
Timestamp


pg_statviz · rancor:5432
Connection/status count
total
active
idle
No. of connections
0
1
2
3
4
11:51:00
11:51:30
11:52:00
11:52:30
11:53:00
11:53:30
11:54:00
11:54:30
11:55:00
11:55:30
Timestamp


pg_statviz · rancor:5432
Connection/user count
postgres
faf
No. of connections
0
1
2
3
11:51:00
11:51:30
11:52:00
11:52:30
11:53:00
11:53:30
11:54:00
11:54:30
11:55:00
11:55:30
Timestamp


pg_statviz · rancor:5432
Wait events
IO/DataFileRead
Total
Wait event count (at time of snapshot)
2
1
0
11:51:00
11:51:30
11:52:00
11:52:30
11:53:00
11:53:30
11:54:00
11:54:30
11:55:00
11:55:30
Timestamp

Use cases

- "Black box" database
  - Deploy and let the developers wreak havoc
  - Identify users/components
- Performance troubleshooting
- Observe and monitor DB behaviour over a long period
  - During a stress test run
  - 8 hours (working hours) / 24 hours (complete day cycle)
  - A month / years (?)

Extension implementation i

- The basis of everything

```
3
4
5   CREATE TABLE IF NOT EXISTS @extschema@.snapshots(
6       snapshot_tstamp timestamptz PRIMARY KEY
7   );
8
9
```

Extension implementation ii

- Everything follows from that, code is modular

```
38
39  -- Buffers
40  CREATE TABLE IF NOT EXISTS @extschema@.buf(
41      snapshot_tstamp timestamptz REFERENCES @extschema@.snapshots(snapshot_tstamp) ON DELETE CASCADE PRIMARY KEY,
42      checkpoints_timed bigint,
43      checkpoints_req bigint,
44      checkpoint_write_time double precision,
45      checkpoint_sync_time double precision,
46      buffers_checkpoint bigint,
47      buffers_clean bigint,
48      maxwritten_clean bigint,
49      buffers_backend bigint,
50      buffers_backend_fsync bigint,
51      buffers_alloc bigint,
52      stats_reset timestamptz);
53
```

Extension implementation iii

```
faf=> \dt pgstatviz.*
              List of relations
  Schema   |   Name    | Type  |  Owner
-----------+-----------+-------+----------
 pgstatviz | buf       | table | postgres
 pgstatviz | conf      | table | postgres
 pgstatviz | conn      | table | postgres
 pgstatviz | db        | table | postgres
 pgstatviz | snapshots | table | postgres
 pgstatviz | wait      | table | postgres
 pgstatviz | wal       | table | postgres
(7 rows)
```

Extension implementation iv

- Snapshot function

```
268
269  -- Snapshots
270  CREATE OR REPLACE FUNCTION @extschema@.snapshot()
271  RETURNS timestamptz
272  AS $$
273      DECLARE ts timestamptz;
274      BEGIN
275          ts := clock_timestamp();
276          INSERT INTO @extschema@.snapshots
277          VALUES (ts);
278          PERFORM @extschema@.snapshot_buf(ts);
279          PERFORM @extschema@.snapshot_conf(ts);
280          PERFORM @extschema@.snapshot_conn(ts);
281          PERFORM @extschema@.snapshot_db(ts);
282          PERFORM @extschema@.snapshot_wait(ts);
283          PERFORM @extschema@.snapshot_wal(ts);
284          RAISE NOTICE 'created pg_statviz snapshot';
285          RETURN ts;
286      END
287  $$ LANGUAGE PLPGSQL;
288
```

Extension implementation v

- Buffers

```
53
54  CREATE OR REPLACE FUNCTION @extschema@.snapshot_buf(snapshot_tstamp timestamptz)
55  RETURNS void
56  AS $$
57      INSERT INTO @extschema@.buf (
58          snapshot_tstamp,
59          checkpoints_timed,
60          checkpoints_req,
61          checkpoint_write_time,
62          checkpoint_sync_time,
63          buffers_checkpoint,
64          buffers_clean,
65          maxwritten_clean,
66          buffers_backend,
67          buffers_backend_fsync,
68          buffers_alloc,
69          stats_reset)
70      SELECT
71          snapshot_tstamp,
72          checkpoints_timed,
73          checkpoints_req,
74          checkpoint_write_time,
75          checkpoint_sync_time,
76          buffers_checkpoint,
77          buffers_clean,
78          maxwritten_clean,
79          buffers_backend,
80          buffers_backend_fsync,
81          buffers_alloc,
82          stats_reset
83      FROM pg_stat_bgwriter;
84  $$ LANGUAGE SQL;
85
```

Extension implementation vi

- Connections

```
97
98  CREATE OR REPLACE FUNCTION @extschema@.snapshot_conn(snapshot_tstamp timestamptz)
99  RETURNS void
100 AS $$
101     WITH
102         pgsa AS (
103             SELECT *
104             FROM pg_stat_activity
105             WHERE datname = current_database()
106             AND state IS NOT NULL),
107         userconns AS (
108             SELECT jsonb_agg(uc)
109             FROM (
110                 SELECT usename AS user, count(*) AS connections
111                 FROM pgsa
112                 GROUP BY usename) uc)
113     INSERT INTO @extschema@.conn (
114         snapshot_tstamp,
115         conn_total,
116         conn_active,
117         conn_idle,
118         conn_idle_trans,
119         conn_idle_trans_abort,
120         conn_fastpath,
121         conn_users)
122     SELECT
123         snapshot_tstamp,
124         count(*) AS conn_total,
125         count(*) FILTER (WHERE state = 'active') AS conn_active,
126         count(*) FILTER (WHERE state = 'idle') AS conn_idle,
127         count(*) FILTER (WHERE state = 'idle in transaction') AS conn_idle_trans,
128         count(*) FILTER (WHERE state = 'idle in transaction (aborted)') AS conn_idle_trans_abort,
129         count(*) FILTER (WHERE state = 'fastpath function call') AS conn_fastpath,
130         (SELECT * from userconns) AS conn_users
131     FROM pgsa;
132 $$ LANGUAGE SQL;
133
```

Extension implementation vii

```
faf=> \df pgstatviz.*
                                        List of functions
  Schema   |       Name       |     Result data type     |           Argument data types            | Type
-----------+------------------+--------------------------+------------------------------------------+------
 pgstatviz | delete_snapshots | void                     |                                          | func
 pgstatviz | snapshot         | timestamp with time zone |                                          | func
 pgstatviz | snapshot_buf     | void                     | snapshot_tstamp timestamp with time zone | func
 pgstatviz | snapshot_conf    | void                     | snapshot_tstamp timestamp with time zone | func
 pgstatviz | snapshot_conn    | void                     | snapshot_tstamp timestamp with time zone | func
 pgstatviz | snapshot_db      | void                     | snapshot_tstamp timestamp with time zone | func
 pgstatviz | snapshot_wait    | void                     | snapshot_tstamp timestamp with time zone | func
 pgstatviz | snapshot_wal     | void                     | snapshot_tstamp timestamp with time zone | func
(8 rows)
```

Utility implementation i

- Modular code in Python

```
10
11  import sys
12  from argh import ArghParser
13  from pg_statviz.modules.analyze import analyze
14  from pg_statviz.modules.buf import buf
15  from pg_statviz.modules.cache import cache
16  from pg_statviz.modules.checkp import checkp
17  from pg_statviz.modules.conn import conn
18  from pg_statviz.modules.tuple import tuple
19  from pg_statviz.modules.wait import wait
20  from pg_statviz.modules.wal import wal
21
22
23  # Python version check
24  if sys.version_info < (3, 7):
25      raise SystemExit("This program requires Python 3.7 or later")
26
```

Utility implementation ii

- "Buffers written" data retrieval and preparation

```
61
62      # Retrieve the snapshots from DB
63      cur = conn.cursor()
64      cur.execute("""SELECT buffers_checkpoint, buffers_clean, buffers_backend,
65                            stats_reset, snapshot_tstamp
66                     FROM pgstatviz.buf
67                     WHERE snapshot_tstamp BETWEEN %s AND %s
68                     ORDER BY snapshot_tstamp""",
69                  (daterange[0], daterange[1]))
70      data = cur.fetchmany(MAX_RESULTS)
71      if not data:
72          raise SystemExit("No pg_statviz snapshots found in this database")
73
74      tstamps = [t['snapshot_tstamp'] for t in data]
75      blcksz = int(info['block_size'])
76
77      # Gather buffers and convert to GB
78      total = [round((b['buffers_checkpoint']
79                      + b['buffers_clean']
80                      + b['buffers_backend'])
81                     * blcksz / 1073741824, 1) for b in data]
82      checkpoints = [round(b['buffers_checkpoint']
83                           * blcksz / 1073741824, 1) for b in data]
84      bgwriter = [round(b['buffers_clean']
85                        * blcksz / 1073741824, 1) for b in data]
86      backends = [round(b['buffers_backend']
87                        * blcksz / 1073741824, 1) for b in data]
88
```

Utility implementation iii

- "Buffers written rate" data preparation

```
113
114     # Buffer diff generator - yields 3-tuple list of the 3 rates in buffers/s
115     def bufdiff(data):
116         yield (numpy.nan, numpy.nan, numpy.nan)
117         for i, item in enumerate(data):
118             if i + 1 < len(data):
119                 if data[i + 1]['stats_reset'] == data[i]['stats_reset']:
120                     s = (data[i + 1]['snapshot_tstamp']
121                          - data[i]['snapshot_tstamp']).total_seconds()
122                     yield ((data[i + 1]['buffers_checkpoint']
123                             - data[i]['buffers_checkpoint']) / s,
124                            (data[i + 1]['buffers_clean']
125                             - data[i]['buffers_clean']) / s,
126                            (data[i + 1]['buffers_backend']
127                             - data[i]['buffers_backend']) / s)
128                 else:
129                     yield (numpy.nan, numpy.nan, numpy.nan)
130     buffers = list(bufdiff(data))
131
132     # Normalize and round the rate data
133     total = [round((b[0] + b[1] + b[2]) * blcksz / 1048576,
134                    1 if b[0] >= 100 else 2)
135              for b in buffers]
136     checkpoints = [round(b[0] * blcksz / 1048576, 1 if b[0] >= 100 else 2)
137                    for b in buffers]
138     bgwriter = [round(b[1] * blcksz / 1048576, 1 if b[0] >= 100 else 2)
139                 for b in buffers]
140     backends = [round(b[2] * blcksz / 1048576, 1 if b[0] >= 100 else 2)
141                 for b in buffers]
142
```

Utility implementation iv

- Plotting

```
142
143     # Plot buffer rates
144     plt, fig = plot.setup()
145     plt.suptitle(f"pg_statviz · {info['hostname']}:{port}",
146                  fontweight='semibold')
147     plt.title("Buffer write rate")
148     plt.plot_date(tstamps, total, label="total", aa=True,
149                   linestyle='solid')
150     plt.plot_date(tstamps, checkpoints, label="checkpoints", aa=True,
151                   linestyle='solid')
152     plt.plot_date(tstamps, bgwriter, label="bgwriter", aa=True,
153                   linestyle='solid')
154     plt.plot_date(tstamps, backends, label="backends", aa=True,
155                   linestyle='solid')
156
157     plt.xlabel("Timestamp", fontweight='semibold')
158     plt.ylabel("Avg. write rate in MB/s", fontweight='semibold')
159     fig.legend()
160     fig.tight_layout()
161     outfile = f"""{outputdir.rstrip("/") + "/" if outputdir
162         else ''}pg_statviz_{info['hostname']
163         .replace("/", "-")}_{port}_buf_rate.png"""
164     _logger.info(f"Saving {outfile}")
165     plt.savefig(outfile)
166
```

Utility implementation v

- Some charts are not so easy (wait events)

```
76      # Determine all kinds of wait event for plotting
77      waitkinds = []
78      for w in wevents:
79          for e in w:
80              if 'wait_event' in e:
81                  wk = {'wait_event_type': e['wait_event_type'],
82                        'wait_event': e['wait_event']}
83                  if wk not in waitkinds:
84                      waitkinds += wk,
85
86      # Plot as many of each wait event kind we have per snapshot
87      plt, fig = plot.setup()
88      plt.suptitle(f"pg_statviz · {info['hostname']}:{port}",
89                   fontweight='semibold')
90      plt.title("Wait events")
91      for wk in waitkinds:
92          wc = []
93          for w in wevents:
94              if not w:
95                  wc += 0,
96              else:
97                  found = False
98                  for e in w:
99                      if wk.items() <= e.items():
100                         wc += e['wait_event_count'],
101                         found = True
102                 if not found:
103                     wc += 0,
104         if not all(c == 0 for c in wc):
105             plt.plot_date(tstamps, wc,
106                           label=f"{wk['wait_event_type']}/{wk['wait_event']}",
107                           aa=True, linestyle='solid')
108     # Plot total wait events
```

Utility implementation iii

- Easy to change matplotlib settings

```
13
14  def setup():
15      for f in ["NotoSans-Regular.ttf", "NotoSans-SemiBold.ttf"]:
16          f = importlib.resources.path("pg_statviz.libs", f)
17          fnt.fontManager.addfont(f)
18      plt.rcParams['font.family'] = 'Noto Sans'
19      plt.rcParams['font.size'] = 12
20      im = plt.imread(importlib.resources.path("pg_statviz.libs",
21                                               "pg_statviz.png"))
22      height = im.shape[0]
23      fig = plt.figure(figsize=(19.2, 10.8))
24      fig.figimage(im, 5, (fig.bbox.ymax - height - 6), zorder=3)
25      plt.grid(visible=True)
26      plt.ticklabel_format(axis='y', style='plain')
27      plt.gcf().autofmt_xdate()
28      return plt, fig
29
```

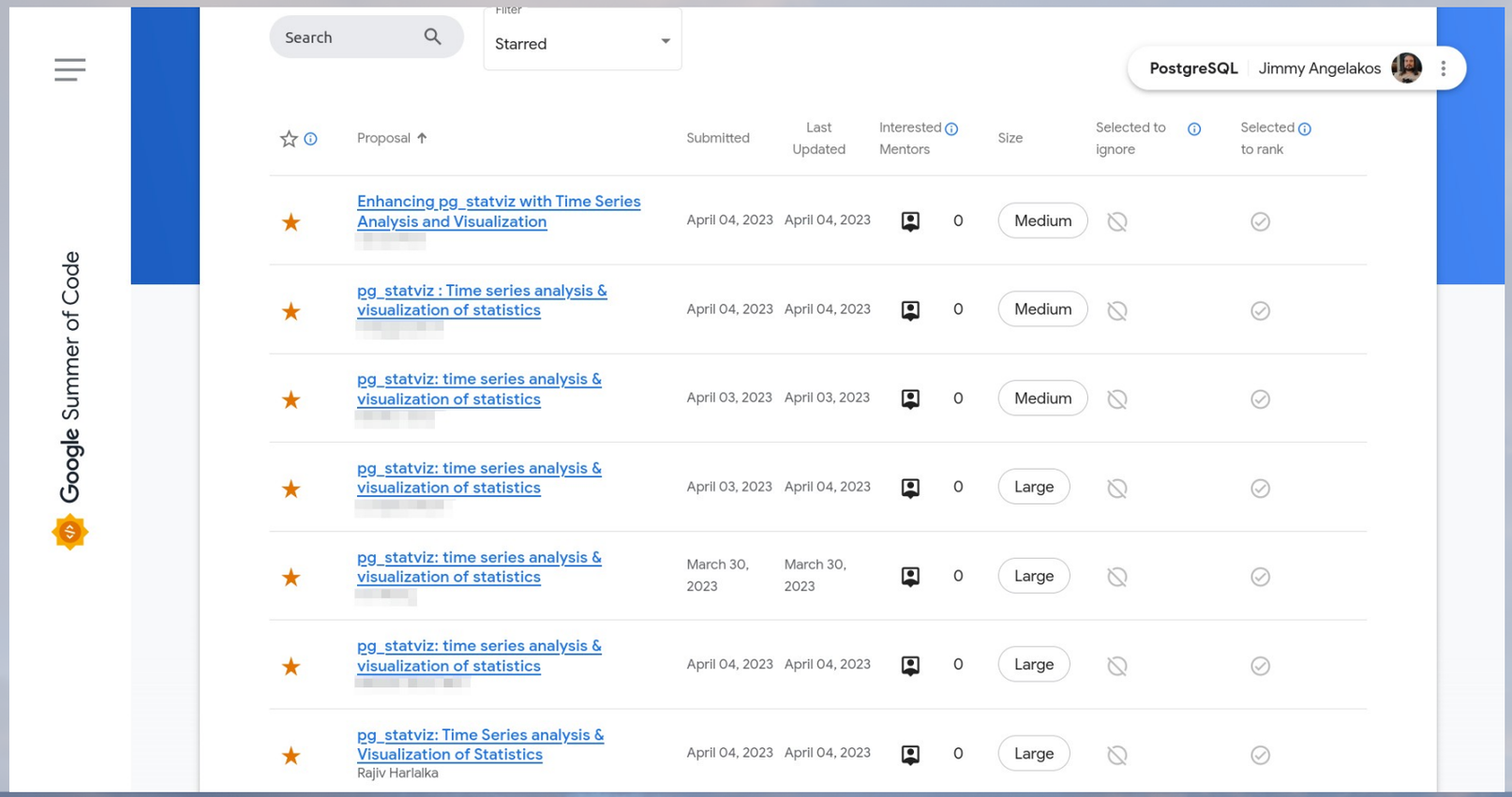
The Future

- Code is currently at "feature-complete alpha" maturity
- Needs:
  - Packaging for PGDG repositories and Linux distributions
  - Additional modules for stats to monitor (such as replication, I/O (pg_stat_io), tables/indexes)
  - Data management/retention functions
  - Writing of Python regression tests

The Future

ii

- Google Summer of Code PostgreSQL Project



Thank you!

- Project:

  https://github.com/vyruss/pg_statviz

- You can find me on Mastodon:

  @vyruss@fosstodon.org

Thank you!


PG Day Chicago 2023 Sponsors
PLATINUM SPONSOR
DRW
GOLD SPONSORS
pgEdge
datAvail
Microsoft
aws
Quest
Where Next Meets Now.
Silver Sponsors
DBeaver
PGX
POSTGRESQL
EXPERTS, INC.
FUJITSU
instaclustr
Now part of Spot by NetApp
Community
Partner
Chicago
PostgreSQL
User Group