Injection Points

PGConf.dev 2025
2025/05/15

Michael Paquier (he/him)

Senior Development Engineer
AWS - RDS

© 2025, Amazon Web Services, Inc. or its affiliates.

The lecturer

- French, based in Tokyo.
- At AWS since 2022.
- PostgreSQL contributor since 2009
  - Patches, reviews and bug fixes.
  - Blogging.
- Committer since 2018.

aws © 2025, Amazon Web Services, Inc. or its affiliates.

Agenda

Concept

Core

Test Facilities

Examples

© 2025, Amazon Web Services, Inc. or its affiliates.

Concept

© 2025, Amazon Web Services, Inc. or its affiliates.

Quote

- https://www.postgresql.org/message-id/20250413021039.bb.nmisch%40google.com

From:	Noah Misch <noah(at)leadboat(dot)com>
To:	Wolfgang Walther <walther(at)technowledgy(dot)de>
Cc:	PostgreSQL Hackers <pgsql-hackers(at)postgresql(dot)org>
Subject:	Re: Buildfarm: Enabling injection points on basilisk/dogfish (Alpine / musl)
Date:	2025-04-13 02:10:39
Message-ID:	20250413021039.bb.nmisch@google.com
Views:	Raw Message | Whole Thread | Download mbox | Resend email
Lists:	pgsql-hackers

```
> I see that coverage in the build farm is not very big. IIUC, those are a
> development tool, so might not be relevant, because nobody is developing on
> Alpine / musl?

No, whether anyone develops on the platform is not a factor.  One hasn't fully
tested PostgreSQL until one builds with injection points.
```

© 2025, Amazon Web Services, Inc. or its affiliates.

Postgres makes testing hard

- Multi-process infrastructure.
- Cannot really synchronize actions.
- Some tricks
  - Locking (LOCK pg_database?)
  - Hooks, limited by code paths and states.
  - Extensions, limited in backend to external libraries.

© 2025, Amazon Web Services, Inc. or its affiliates.

Requirements

- Developer tool.
  - Run custom code
  - At custom location
- Properties
  - Flexible: core, extension, backpatch
  - Customizable, extensible
  - Cheap
  - Maintenance

© 2025, Amazon Web Services, Inc. or its affiliates.

Test Designs

- Race Conditions
- States: ERROR, FATAL, PANIC
- Cross-process interactions:
  - Wait and Wake
  - Monitoring
- Corner cases: stack manipulation, cheap, reproducibility rate
- Print information: NOTICE, LOG.

aws © 2025, Amazon Web Services, Inc. or its affiliates.

Past Proposals

- Better infra for concurrency (2020) https://postgr.es/m/CAPpHfdtSEOHX8dSk9Qp+Z++i4BGQoffKip6JDWngEA+g7Z-XmQ@mail.gmail.com
- Forks of upstream:
  - PROBE_POINT() by 2ndQ: https://postgr.es/m/CAGRY4nyiBqP=nNmYbee1q2rcyWROM68N36DiEF4ttv3Zr9Wocw@mail.gmail.com
  - Greenplum (CloudberryDB), proposed once upstream.
  - Usually required by products, different languages/tools.

aws © 2025, Amazon Web Services, Inc. or its affiliates.

Core

© 2025, Amazon Web Services, Inc. or its affiliates.

Code

- Disabled by default, enabled in CI:
  - configure --enable-injection-points
  - meson -Dinjection_points=true
- Sources
  - src/backend/utils/misc/injection_point.c
  - src/include/utils/injection_point.h
- Shared memory array: point, library, function, private area.
- Process cache with hash table: loaded function pointer.
- Documentation: https://www.postgresql.org/docs/devel/xfunc-c.html#XFUNC-ADDIN-INJECTION-POINTS

aws © 2025, Amazon Web Services, Inc. or its affiliates.

Basics

- INJECTION_POINT(“hoge”)

```
#ifdef USE_INJECTION_POINTS
#define INJECTION_POINT(name, arg) InjectionPointRun(name, arg)
#else
#define INJECTION_POINT(name, arg) ((void) name)
#endif
```

```
extern void InjectionPointAttach(const char *name,
                                 const char *library,
                                 const char *function,
                                 const void *private_data,
                                 int private_data_size);
extern bool InjectionPointDetach(const char *name);
extern void InjectionPointRun(const char *name, void *arg);
```

aws © 2025, Amazon Web Services, Inc. or its affiliates.

Critical Sections

- Problem with internal_load_library()@dfmgr.c
- Allocation when running a point the first time!
- Crashes:

```
START_CRIT_SECTION();
INJECTION_POINT(“hoge”, NULL);    /* PANIC */
END_CRIT_SECTION();
```

© 2025, Amazon Web Services, Inc. or its affiliates.

Cached Points

- Two-step process:
  - Load a point, heat process cache.
  - Run callback from cache.
- Works:

```
INJECTION_POINT_LOAD(“hoge”);
START_CRIT_SECTION();
INJECTION_POINT_CACHED(“hoge”, NULL); /* okay */
END_CRIT_SECTION();
```

aws © 2025, Amazon Web Services, Inc. or its affiliates.

Stack Manipulation

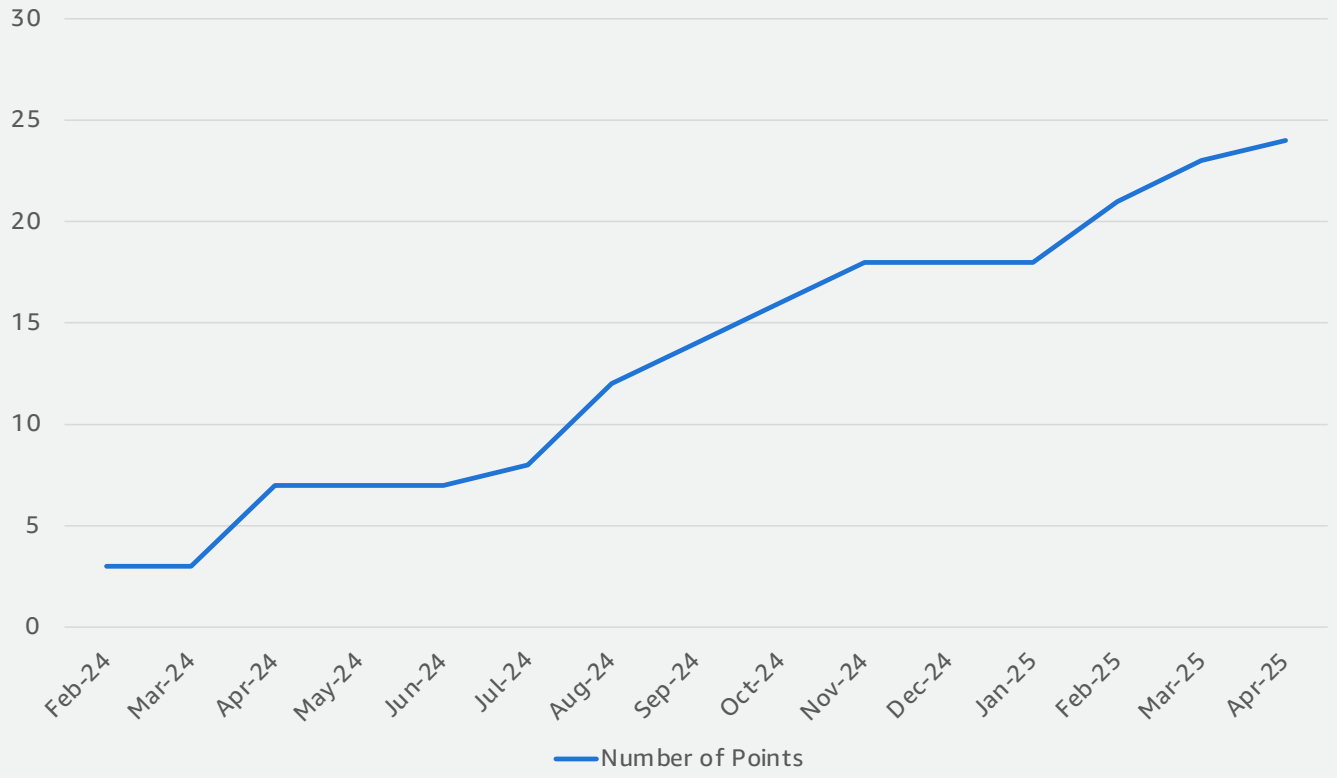
- Runtime arguments, see 371f2db8b05e for AIO.
- Conditional states:

```
If (IS_INJECTION_POINT_ATTACHED(“hoge”))
{
   /* Force a custom state */
   some_var = 199;

   /* Execute callback */
   INJECTION_POINT_CACHED(“hoge”, NULL);
}
```

aws © 2025, Amazon Web Services, Inc. or its affiliates.

INJECTION_POINT() vs Time



© 2025, Amazon Web Services, Inc. or its affiliates.

Test Facilities

© 2025, Amazon Web Services, Inc. or its affiliates.

injection_points

- src/test/modules/injection_points/
  - Not contrib/
  - May evolve in stable branches
  - Flexible for bug fixes (perhaps critical)
- SQL functions:
  - Attach, Detach
  - Wakeup
  - Run, Load
  - Callbacks: ‘error’, ‘notice’, ‘wait’.

aws © 2025, Amazon Web Services, Inc. or its affiliates.

injection_points - Objects

```
=# CREATE EXTENSION injection_points;
=# \dx+ injection_points
   Objects in extension "injection_points"
            Object description
-----------------------------------------------
 function injection_points_attach(text,text)
 function injection_points_cached(text,text)
 function injection_points_detach(text)
 function injection_points_load(text)
 function injection_points_run(text,text)
 function injection_points_set_local()
 function injection_points_wakeup(text)
(7 rows)
```

© 2025, Amazon Web Services, Inc. or its affiliates.

injection_points – Local points

- SQL tests.
- injection_points_set_local() at top of SQL script
- Concurrent-safe
- Automatic detach at process shutdown.

```
SELECT injection_points_set_local();
SELECT injection_point_attach('foo', 'error');
SELECT injection_point_run('foo'); -- ERROR!
-- Or insert here SQL sequence able to run 'foo'
\q -- detach
```

aws © 2025, Amazon Web Services, Inc. or its affiliates.

Isolation Tests

- isolationtester, relies on wait&wakeup.

```
setup       { CREATE EXTENSION injection_points; }
teardown { DROP EXTENSION injection_points; }

session s1
setup   {
  SELECT injection_points_set_local();
  SELECT injection_points_attach('injection-points-wait', 'wait');
}
step wait1 { SELECT injection_points_run('injection-points-wait'); }
step noop1  { }

session s2
step wakeup2    { SELECT injection_points_wakeup('injection-points-wait'); }
step detach2     { SELECT injection_points_detach('injection-points-wait'); }

# Detach after wait and wakeup.
permutation wait1 wakeup2 noop1 detach2
```

© 2025, Amazon Web Services, Inc. or its affiliates.

TAP tests

- Wait and Wake, mostly!
- Monitoring: pg_stat_activity, server logs.

```
$node->start;
[...]
# Wait for specific event.
$node->wait_for_event($backend_type, $event_name);
[...]
# Hold session, then wait for specific event.
my $observer = $node->background_psql('postgres');
$observer->query_until(
  qr/start/,
  q{\echo start
    SELECT injection_points_run($event_name);});
$node->wait_for_event('client backend', $event_name);
```

aws © 2025, Amazon Web Services, Inc. or its affiliates.

Examples

© 2025, Amazon Web Services, Inc. or its affiliates.

SQL - REINDEX bug

- Index corruption: “safe”? Not really.
  - Expressions and predicates.
  - Possible access to other tables.
  - Snapshot waits missed.
- reindex_conc.sql in injection_points module.
- Commit 5bbdfa8a18dc.

aws © 2025, Amazon Web Services, Inc. or its affiliates.

SQL - REINDEX bug

```
--- a/src/backend/commands/indexcmds.c
+++ b/src/backend/commands/indexcmds.c
[...]
@@ -3782,8 +3783,16 @@ ReindexRelationConcurrently(...)
+#ifdef USE_INJECTION_POINTS
+       if (idx->safe)
+           INJECTION_POINT("reindex-conc-index-safe", NULL);
+       else
+           INJECTION_POINT("reindex-conc-index-not-safe", NULL);
+#endif
```

```
-- Attach a NOTICE to both points
SELECT injection_points_attach('reindex-conc-index-safe', 'notice');
SELECT injection_points_attach('reindex-conc-index-not-safe', 'notice');
-- Check the state with the commands
REINDEX INDEX CONCURRENTLY reindex_inj.ind_simple;
NOTICE:  notice triggered for injection point reindex-conc-index-safe
REINDEX INDEX CONCURRENTLY reindex_inj.ind_expr;
NOTICE:  notice triggered for injection point reindex-conc-index-not-safe
```

© 2025, Amazon Web Services, Inc. or its affiliates.

Isolation - GRANT/VACUUM

- Data loss and index corruption.
  - In-place UPDATE for stats at the end of VACUUM.
  - GRANT TABLE, DATABASE
- Test “inplace” in injection_points module.
- Commit a07e03fd8fa7.

aws © 2025, Amazon Web Services, Inc. or its affiliates.

Isolation - GRANT/VACUUM

```
--- a/src/backend/access/index/genam.c
+++ b/src/backend/access/index/genam.c
@@ -6089,6 +6194,19 @@ heap_inplace_update(...)
[...]
+       INJECTION_POINT("inplace-before-pin");
```

```
session s1
setup   {
  SELECT injection_points_set_local();
  SELECT injection_points_attach('inplace-before-pin', 'wait');
}
step vac1      { VACUUM vactest.orig50;  -- wait during inplace update }

# Transactional updates of the tuple vac1 is waiting to inplace-update.
session s2
step grant2     { GRANT SELECT ON TABLE vactest.orig50 TO PUBLIC; }
step revoke2    { REVOKE SELECT ON TABLE vactest.orig50 FROM PUBLIC; }
```

aws © 2025, Amazon Web Services, Inc. or its affiliates.

TAP - Promote and checkpoint

- Race condition:
  - Restart point still running after promotion.
  - PANIC after restart.
- Wait for restartpoint in startup process, wakeup.
- Recovery test 041_checkpoint_at_promote.
- Bugfix at 7863ee4def65.
- Test at 6782709df81f, 2~3min vs 5s.

aws © 2025, Amazon Web Services, Inc. or its affiliates.

TAP - Promote and checkpoint

```
--- a/src/backend/access/transam/xlog.c
+++ b/src/backend/access/transam/xlog.c
@@ -7528,6 +7529,12 @@ CreateRestartPoint(int flags)
[...]
+   INJECTION_POINT("create-restart-point");
```

```
$node_standby->safe_psql('postgres',
  "SELECT injection_points_attach('create-restart-point', 'wait');");
[...]
my $psql_session =
  $node_standby->background_psql('postgres', on_error_stop => 0);
$psql_session->query_until(
  qr/starting_checkpoint/, q(\echo starting_checkpoint
  CHECKPOINT;));
[...]
$node_standby->wait_for_event('checkpointer', 'create-restart-point');
[...]
$node_standby->promote;
[...]
$node_standby->safe_psql('postgres',
  "SELECT injection_points_wakeup('create-restart-point');");
```

© 2025, Amazon Web Services, Inc. or its affiliates.

Bypass

- Avoid random WAL records for standby snapshots.
- See 105b2cb33617, for 035_standby_logical_decoding.pl

```
XLogRecPtr
LogStandbySnapshot(void)
{
[…]
#ifdef USE_INJECTION_POINTS
   if (IS_INJECTION_POINT_ATTACHED("skip-log-running-xacts"))
   {
      /*
       * This record could move slot's xmin forward during decoding, leading
       * to unpredictable results, so skip it when requested by the test.
       */
      return GetInsertRecPtr();
   }
#endif
```

© 2025, Amazon Web Services, Inc. or its affiliates.

Next plans

- Injection point persistence
  - Integration with injection_points, flush via SQL.
  - Re-create at start via shared_preload_libraries.
  - Very early startup conditions.
  - https://postgr.es/m/aBG2rPwl3GE7m1-Q%40paquier.xyz
- Backpatch to stable branches?

aws © 2025, Amazon Web Services, Inc. or its affiliates.

Thank you!

Michael Paquier
@paquier

© 2025, Amazon Web Services, Inc. or its affiliates.