MAKE BTREE_GIST FASTER

BERND HELMLE

www.cybertec-postgresql.com

CYBERTEC
POSTGRESQL SERVICES & SUPPORT

PROUD CONTRIBUTOR TO
PostgreSQL
the world's most advanced open source database

WHAT'S IT ALL ABOUT?

- GiST (Generalized Search Tree)
- Framework for somehow specialized indexes
- `btree_gist:` extension available in PostgreSQL
- Implements btree-like operators/behavior with GiST

USAGE EXAMPLE (1)

nearest-neighbour-search

```
1| CREATE TABLE test (a int4);
2| -- create index
3| CREATE INDEX testidx ON test USING GIST (a);
4| ...
5| -- nearest-neighbor search: find the ten entries closest to "42"
6| SELECT *, a <-> 42 AS dist FROM test ORDER BY a <-> 42 LIMIT 10;
```

USAGE EXAMPLE (2)

Exclusion Constraints

```
1| CREATE TABLE zoo (
2|    cage INTEGER,
3|    animal TEXT,
4|    EXCLUDE USING GIST (cage WITH =, animal WITH <>)
5| );
```

WE CAN’T HAVE NICE THINGS WITHOUT CAVEATS

- Two build methods for GiST: sorted and buffered
- First one is used, if sortsupport for opclasses used by the index is available
- If not, use buffered: inserts each tuple one by one into the new index
- btree_gist opclass lacks sortsupport

SOLUTION

- `sortsupport` API: Implement `sortsupport` for `btree_gist` opclass.
- Extend that idea to builtin range types. . .

THE PATCH (1)

- Original idea implemented by Andrey Borodin in 2020 (https://commitfest.postgresql.org/patch/2824/)
- Committed in PostgreSQL, but reverted
- Some unresolved problems with correct handling of specific datatypes.

THE PATCH (2)

- New attempt by Christoph Heiss and me 2022
- Similar problems
- Needed some time to figure out everything

FINALLY

```
commit e4309f73f698851a2f7d49ca5e98e3e188400891
Author: Heikki Linnakangas <heikki.linnakangas@iki.fi>
Date: Thu Apr 3 13:46:35 2025 +0300

    Add support for sorted gist index builds to btree_gist

    This enables sortsupport in the btree_gist extension for faster builds
    of gist indexes.

    Sorted gist index build strategy is the new default now. Regression
    tests are unchanged (except for one small change in the 'enum' test to
    add coverage for enum values added later) and are using the sorted
    build strategy instead.

    One version of this was committed a long time ago already, in commit
    9f984ba6d2, but it was quickly reverted because of buildfarm
    failures. The failures were presumably caused by some small bugs, but
    we never got around to debug and commit it again. This patch was
    written from scratch, implementing the same idea, with some fragments
    and ideas from the original patch.

    Author: Bernd Helmle <mailings@oopsware.de>
    Author: Andrey Borodin <x4mmm@yandex-team.ru>
    Discussion: https://www.postgresql.org/message-id/64d324ce2a6d535d3f0f3baeeea7b25beff82ce4.camel@oopsware.de
```

FINALLY

```
commit e9e7b66044c9e3dfa76fd1599d5703acd3e4a3f5
Author: Heikki Linnakangas <heikki.linnakangas@iki.fi>
Date: Wed Apr 2 19:51:28 2025 +0300

    Add GiST and btree sortsupport routines for range types

    For GiST, having a sortsupport function allows building the index
    using the "sorted build" method, which is much faster.

    For b-tree, the sortsupport routine doesn't give any new
    functionality, but speeds up sorting a tiny bit. The difference is not
    very significant, about 2% in cursory testing on my laptop, because
    the range type comparison function has quite a lot of overhead from
    detoasting. In any case, since we have the function for GiST anyway,
    we might as well register it for the btree opfamily too.

    Author: Bernd Helmle <mailings@oopsware.de>
    Discussion:https://www.postgresql.org/message-id/64d324ce2a6d535d3f0f3baeeea7b25beff82ce4.camel@oopsware.de
```

SOME NUMBERS (1)

```
 1| init.pgbench script for performance testing
 2|
 3| BEGIN;
 4|
 5| DROP TABLE IF EXISTS test_dataset;
 6| CREATE TABLE test_dataset(keyid integer not null, id uuid not null,
 7| block_range int4range);
 8| CREATE TEMP SEQUENCE testset_seq;
 9| INSERT INTO test_dataset SELECT nextval('testset_seq'), id, block_range
10| FROM test ORDER BY random() LIMIT 10000;
11| CREATE UNIQUE INDEX ON test_dataset(keyid);
12|
13| COMMIT;
```

SOME NUMBERS (2)

```
1| \set keyid random(1, 10000)
2| SELECT id, block_range FROM test_dataset WHERE keyid = :keyid Ägset
3| SELECT id, block_range FROM test WHERE id = ':id' AND block_range &&
4| ':block_range';
```

SOME NUMBERS (3)

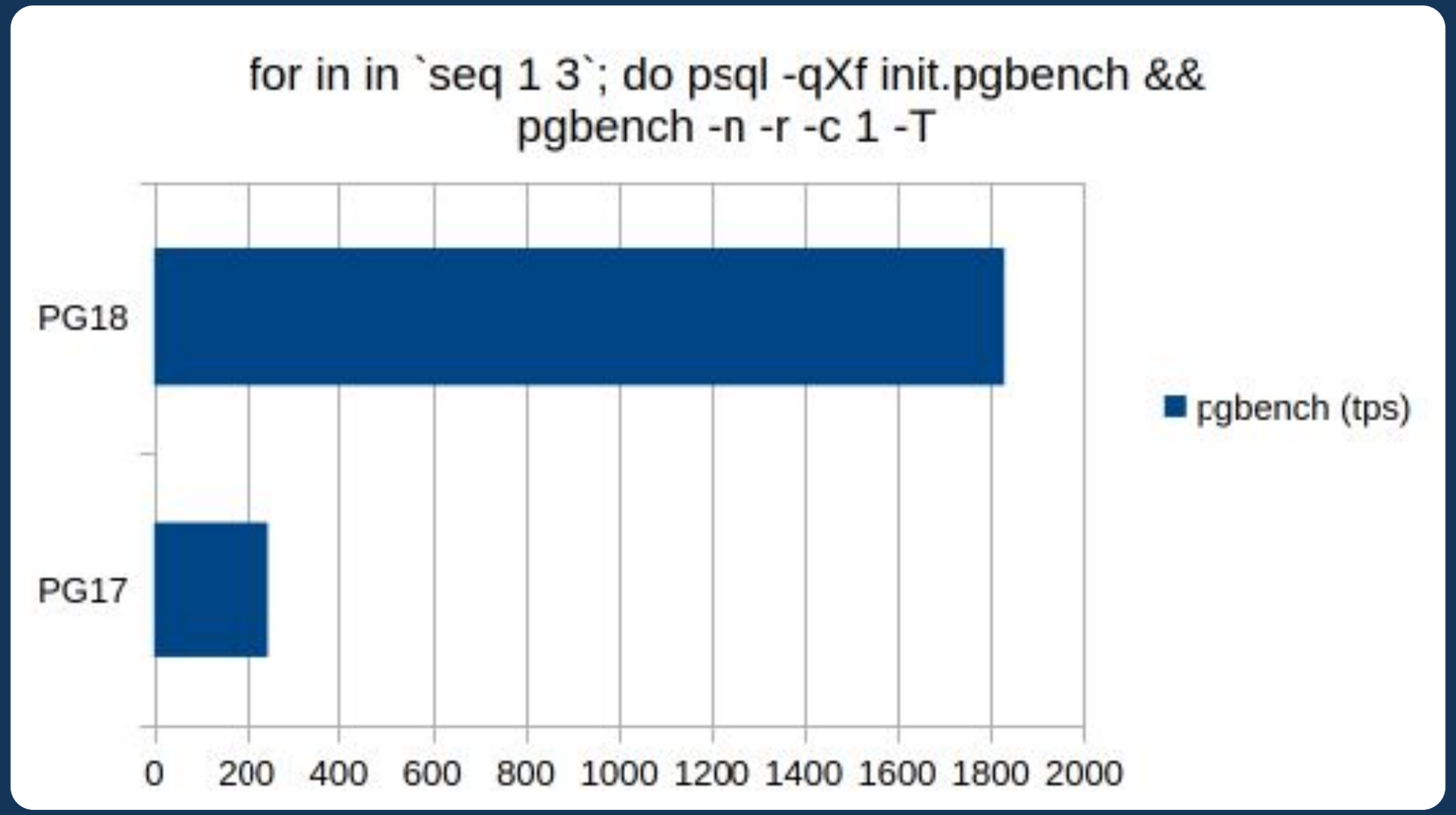

Figure 1: GiST Build timing

SOME NUMBERS (4)

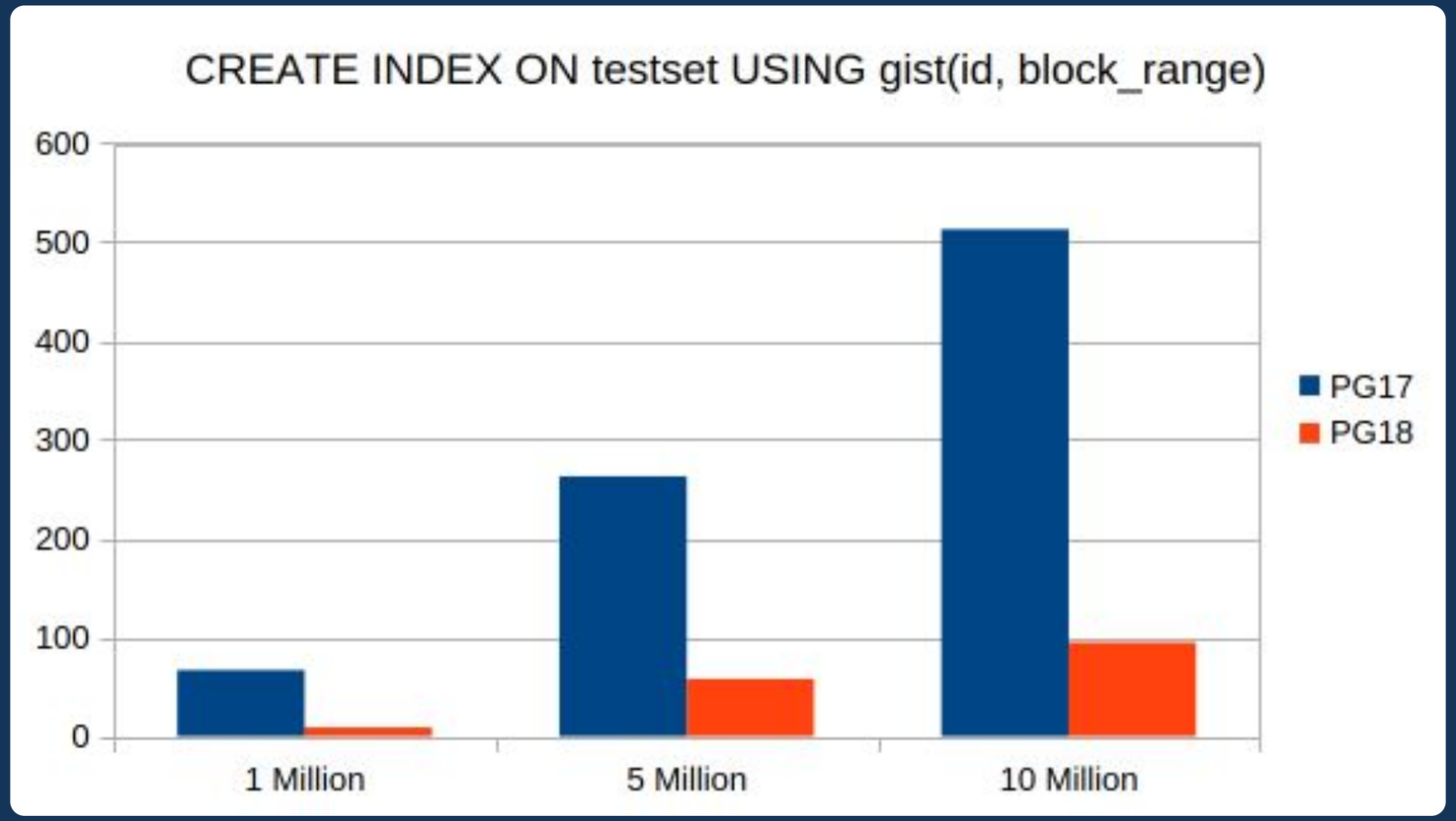

Figure 2: GiST Build timing

BERND HELMLE

SENIOR DATABASE ENGINEER

EMAIL

bernd.helmle@cybertec.at

PHONE

+4917680133292

www.cybertec-postgresql.com

@cybertec-postgresql

www.youtube.com/@cybertecpostgresql

PROUD CONTRIBUTOR TO

PostgreSQL
the world's most advanced open source database