Yandex Cloud

GiST news and prospects

Andrey Borodin

Open source software is
made and maintained
by people, for people

Slides ->

Open source software is
made and maintained
by people, for people

in the beginning was the word… scientific paper

Generalized Search Trees for Database Systems

(Extended Abstract)

Joseph M. Hellerstein
University of Wisconsin, Madison
jmh@cs.berkeley.edu

Jeffrey F. Naughton*
University of Wisconsin, Madison
naughton@cs.wisc.edu

Avi Pfeffer
University of California, Berkeley
avi@cs.berkeley.edu

Abstract

This paper introduces the Generalized Search Tree (GiST), an index structure supporting an extensible set of queries and data types. The GiST allows new data types to be indexed in a manner supporting queries natural to the types; this is in contrast to previous work on tree extensibility which only supported the traditional set of equality and range predicates. In a single data structure, the GiST provides all the basic search tree logic required by a database system, thereby unifying disparate structures such as B+-trees and R-trees in a single piece of code, and opening the application of search trees to general extensibility.

To illustrate the flexibility of the GiST, we provide simple method implementations that allow it to behave like a B+-tree, an R-tree, and an *RD-tree*, a new index for data with set-valued attributes. We also present a preliminary performance analysis of RD-trees, which leads to discussion on the nature of tree indices and how they behave for various datasets.

1 Introduction

An efficient implementation of search trees is crucial for any database system. In traditional relational systems, B+-trees [Com79] were sufficient for the sorts of queries posed on the usual set of alphanumeric data types. Today, database systems are increasingly being deployed to support new applications such as geographic information systems, multimedia systems, CAD tools, document libraries, sequence databases, developing domain-specific search trees is problematic. The effort required to implement and maintain such data structures is high. As new applications need to be supported, new tree structures have to be developed from scratch, requiring new implementations of the usual tree facilities for search, maintenance, concurrency control and recovery.

2. *Search Trees For Extensible Data Types:* As an alternative to developing new data structures, existing data structures such as B+-trees and R-trees can be made *extensible* in the data types they support [Sto86]. For example, B+-trees can be used to index any data with a linear ordering, supporting equality or linear range queries over that data. While this provides extensibility in the data that can be indexed, it does not extend the set of queries which can be supported by the tree. Regardless of the type of data stored in a B+-tree, the only queries that can benefit from the tree are those containing equality and linear range predicates. Similarly in an R-tree, the only queries that can use the tree are those containing equality, overlap and containment predicates. This inflexibility presents significant problems for new applications, since traditional queries on linear orderings and spatial location are unlikely to be apropos for new data types.

In this paper we present a third direction for extending search tree technology. We introduce a new data structure

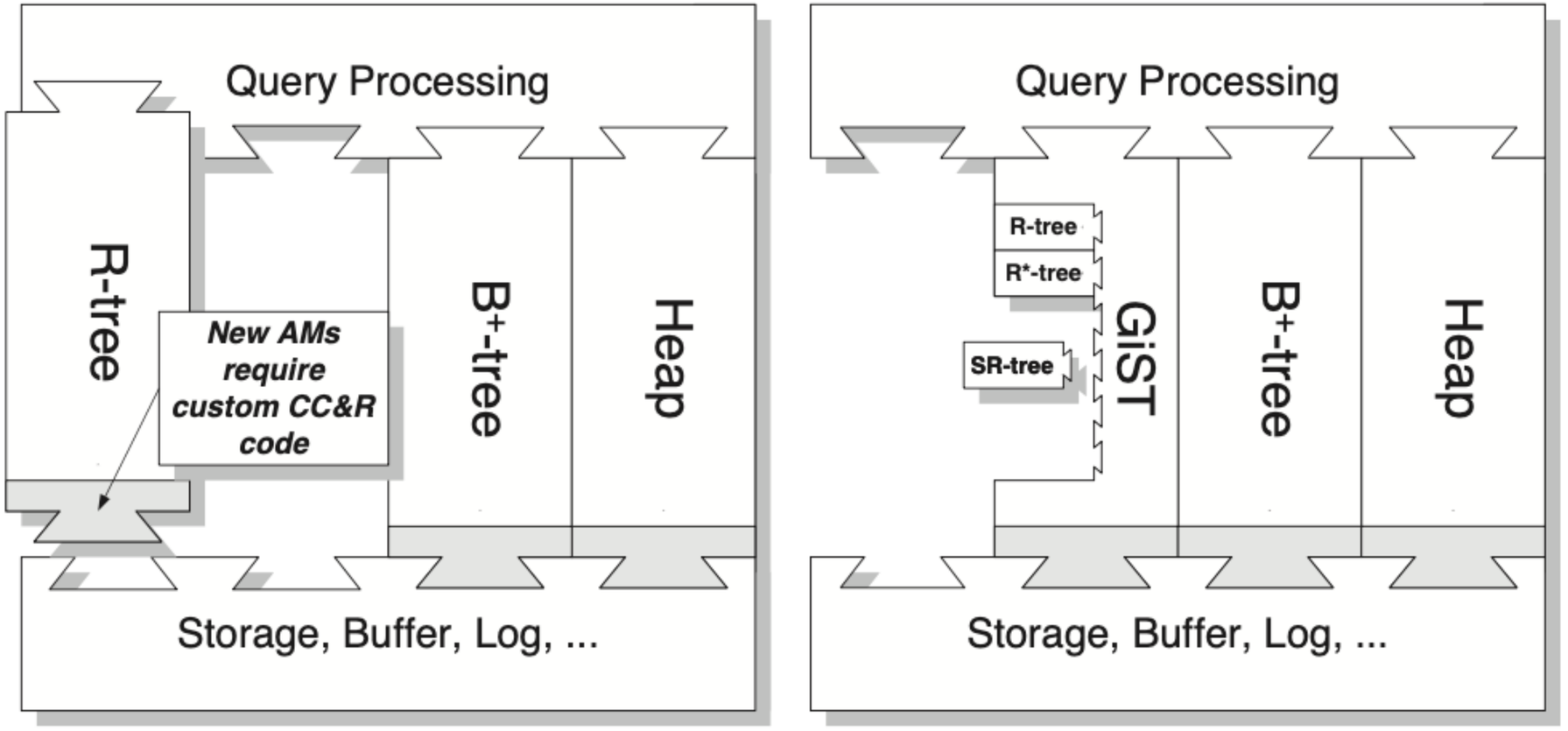
Access Methods for Next-Generation Database Systems

by

Marcel Kornacker

Diplom Informatik (Universität Hamburg) 1995
M.S. (University of California, Berkeley) 1997

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy



(a) Standard ORDBMS

(b) ORDBMS with GiST

Figure 2.1: Access method interfaces – the database extender's perspective.

GiST for PostgreSQL

Oleg Teodor

PostgreSQL: Documentation: 16

https://www.postgresql.org/docs/16/gist-intro.html

25th May 2023: PostgreSQL 16 Beta 1 Released!

Documentation → PostgreSQL 16
Supported Versions: Current (15) / 14 / 13 / 12 / 11
Development Versions: 16 / devel
Unsupported versions: 10 / 9.6 / 9.5 / 9.4 / 9.3 / 9.2 / 9.1 / 9.0 / 8.4 / 8.3 / 8.2

Search the documentation for...

This documentation is for an unsupported version of PostgreSQL.
You may want to view the same page for the **current** version, or one of the other supported versions listed above instead.

68.1. Introduction
Prev Up Chapter 68. GiST Indexes Home Next

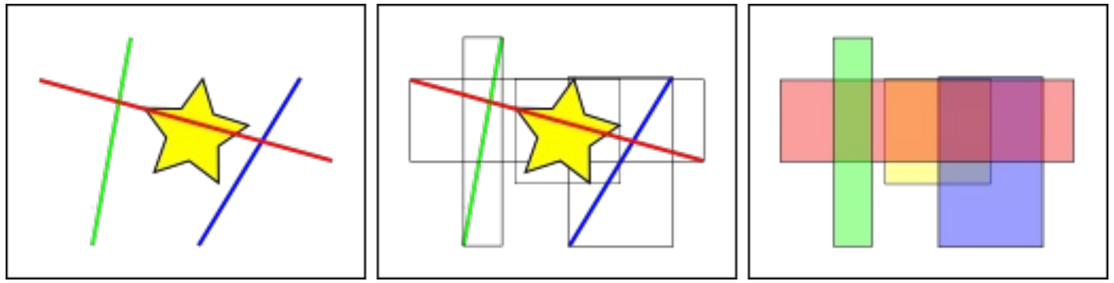
68.1. Introduction

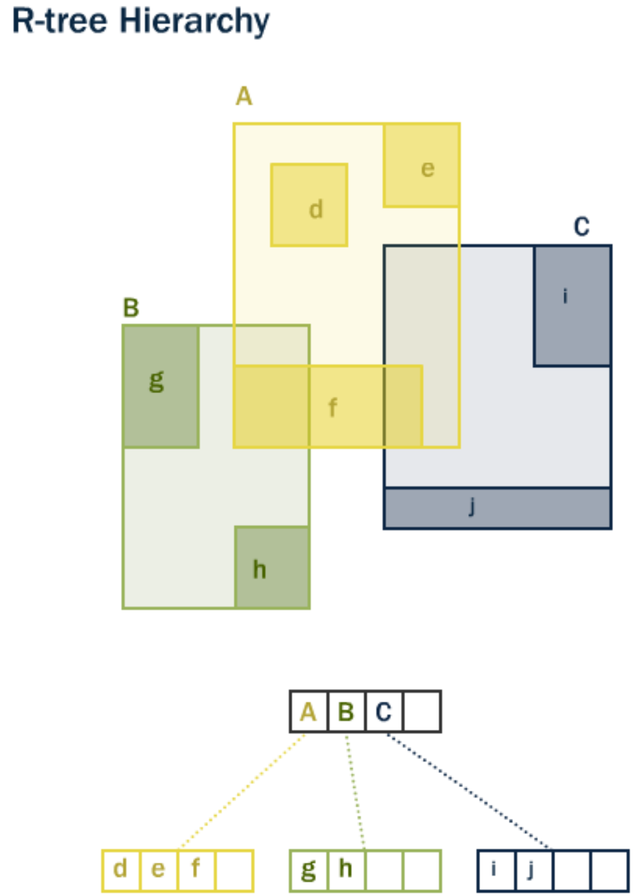
GiST stands for Generalized Search Tree. It is a balanced, tree-structured access method, that acts as a base template in which to implement arbitrary indexing schemes. B-trees, R-trees and many other indexing schemes can be implemented in GiST.

One advantage of GiST is that it allows the development of custom data types with the appropriate access methods, by an expert in the domain of the data type, rather than a database expert.

Some of the information here is derived from the University of California at Berkeley's GiST Indexing Project web site and Marcel Kornacker's thesis, Access Methods for Next-Generation Database Systems. The GiST implementation in PostgreSQL is primarily maintained by Teodor Sigaev and Oleg Bartunov, and there is more information on their web site.

PostGIS





Fast forward to our days

Postgres 11: predicate locking

Predicate locking in Gist index

Edit | Comment/Review ▾ | Change Status ▾

Title	Predicate locking in Gist index
Topic	Server Features
Created	2017-06-16 17:01:13
Last modified	2018-03-27 12:44:58 (5 years, 2 months ago)
Latest email	2018-03-27 12:44:48 (5 years, 2 months ago)
Status	2018-03: Committed 2018-01: Moved to next CF 2017-11: Moved to next CF 2017-09: Moved to next CF
Target version	
Authors	Shubham Barai (shubham)
Reviewers	Fedor Sigaev (sigaev), Alexander Korotkov (smagen), Andrey Borodin (x4m)
Committer	Fedor Sigaev (sigaev)

Postgres 12: covering GiST

```
CREATE INDEX on tbl USING gist(c4) INCLUDE (c3);
```

https://commitfest.postgresql.org/22/1615/

Home / Commitfest 2019-03 / Covering GiST indexes

Covering GiST indexes

Edit | Comment/Review | Change Status

Title	Covering GiST indexes
Topic	Server Features
Created	2018-04-13 13:38:52
Last modified	2019-03-15 10:52:09 (4 years, 2 months ago)
Latest email	2019-03-10 16:42:12 (4 years, 2 months ago)
Status	2019-03: Committed 2019-01: Moved to next CF 2018-11: Moved to next CF 2018-09: Moved to next CF 2018-07: Moved to next CF
Target version	12
Authors	Andrey Borodin (x4m)
Reviewers	Andreas Karlsson (kandreas)
Committer	Alexander Korotkov (smagen)

Tom was involved in every single feature in this talk

Fix GiST buffering build to work when there are included columns.

```
gistRelocateBuildBuffersOnSplit did not get the memo about which
attribute count to use.  This could lead to a crash if there were
included columns and buffering build was chosen.  (Because there
are random page-split decisions elsewhere in GiST index build,
the crashes are not entirely deterministic.)

Back-patch to v12 where GiST gained support for included columns.

Pavel Borisov

Discussion: https://postgr.es/m/CALT9ZEECCV5m7wvxg46PC-7x-EybUmnpupBGH
```

master
REL_16_BETA1 ... REL_14_BETA1

tglsfdc committed on Oct 13, 2020

✓ **Fix dereference of dangling pointer in GiST index buffering build.**

```
gistBuildCallback tried to fetch the size of an index tuple that
might have already been freed by gistProcessEmptyingQueue.
While this seems to usually be harmless in production builds,
in principle it could result in a SIGSEGV, or more likely a bogus
value for indtuplesSize leading to poor page-split decisions later
in the build.

Patch by Alexander Lakhin (commentary by Pavel Borisov and me).
Back-patch to all supported branches.

Discussion: https://postgr.es/m/16329-7a6aa9b6fa1118a1@postgresql.org
Discussion: https://postgr.es/m/17874-63ca6c7ce42d2103@postgresql.org
```

master
REL_16_BETA1

tglsfdc committed on Mar 29

Feature can be further improved

CREATE INDEX on tbl USING gist(column1) INCLUDE (column1);

-- if column1 has no IndexOnlyScan-supporting opclass, planner will not use IndexOnlyScan

Postgres 13: faster VACUUM

- Phisically-ordered scan
- Actual page deletion

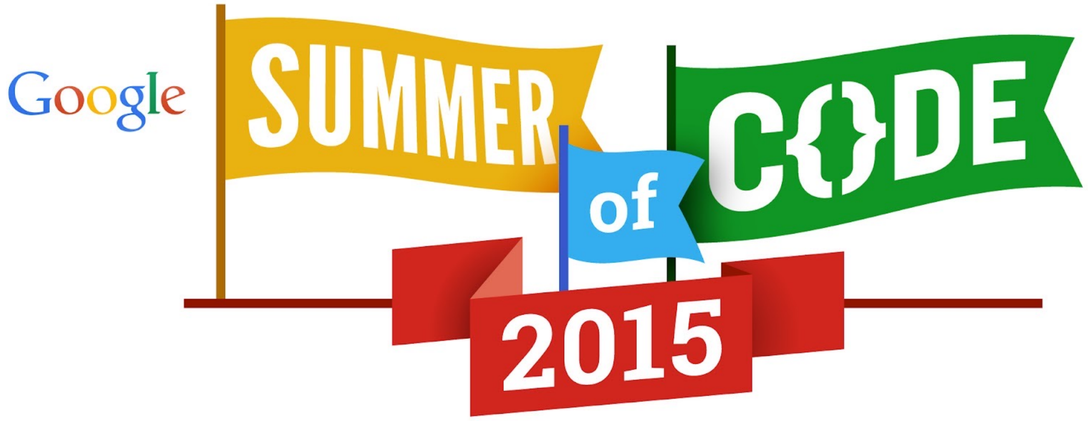
https://commitfest.postgresql.org/7/371/

Home / Commitfest 2015-11 / New gist vacuum

New gist vacuum

Edit | Comment/Review | Change Status

Title	New gist vacuum
Topic	Performance
Created	2015-09-14 11:36:00
Last modified	2018-02-25 13:56:56 (5 years, 3 months ago)
Latest email	2018-03-02 17:00:41 (5 years, 2 months ago)
Status	2016-01: Returned with feedback 2015-11: Moved to next CF
Target version	
Authors	Constantine Kuznetsov (chapaev28)
Reviewers	Jeff Janes (jjanes)
Committer	Fedor Sigaev (sigaev)
Links	Wiki
Emails	**New gist vacuum.** × First at 2015-09-10 22:52:30 by Костя Кузнецов <chapaev28 at ya.ru> Latest at 2018-03-02 17:00:41 by Andrey Borodin <x4mmm at yandex-team.ru> Latest attachment (0001-Fix-GiST-stats-for-partial-indexes-v3.patch) at 2018-03-02 05:33:40 from Andrey Borodin <x4mmm at yandex-team.ru> +



GiST VACUUM

Edit | Comment/Review ▾ | Change Status ▾

Title	GiST VACUUM
Topic	Performance
Created	2018-03-07 07:42:52
Last modified	2019-03-25 16:34:02 (4 years, 2 months ago)
Latest email	2019-04-05 05:39:19 (4 years, 1 month ago)
Status	2019-03: Committed 2019-01: Moved to next CF 2018-11: Moved to next CF 2018-09: Moved to next CF 2018-07: Moved to next CF
Target version	12
Authors	Constantine Kuznetsov (chapaev28), Andrey Borodin (x4m)
Reviewers	Heikki Linnakangas (heikki)
Committer	Heikki Linnakangas (heikki)

From:	Jeff Janes <jeff(dot)janes(at)gmail(dot)com>
To:	Heikki Linnakangas <hlinnaka(at)iki(dot)fi>
Cc:	Andrey Borodin <x4mmm(at)yandex-team(dot)ru>, Michael Paquier <michael(at)paqu Кузнецов <chapaev28(at)ya(dot)ru>
Subject:	Re: GiST VACUUM
Date:	2019-03-15 19:20:06
Message-ID:	CAMkU=1wOqJgjXMfO_R_rUjZ6dvba3x-xeCPBLYQZ1pTV1utLow@mail.gmail.com
Views:	Raw Message | Whole Thread | Download mbox | Resend email
Lists:	pgsql-hackers

On Tue, Mar 5, 2019 at 8:21 AM Heikki Linnakangas <hlinnaka(at)iki(dot)fi> wrote:

Thank you. This is a transformational change; it will allow GiST indexes
larger than RAM to be used in some cases where they were simply not
feasible to use before. On a HDD, it resulted in a 50 fold improvement in
vacuum time, and the machine went from unusably unresponsive to merely
sluggish during the vacuum. On a SSD (albeit a very cheap laptop one, and
exposed from Windows host to Ubuntu over VM Virtual Box) it is still a 30
fold improvement, from a far faster baseline. Even on an AWS instance with
a "GP2" SSD volume, which normally shows little benefit from sequential
reads, I get a 3 fold speed up.

I also ran this through a lot of crash-recovery testing using simulated
torn-page writes using my traditional testing harness with high concurrency
(AWS c4.4xlarge and a1.4xlarge using 32 concurrent update processes) and
did not encounter any problems. I tested both with btree_gist on a scalar
int, and on tsvector with each tsvector having 101 tokens.

I did notice that the space freed up in the index by vacuum doesn't seem to
get re-used very efficiently, but that is an ancestral problem independent
of this change.

Cheers,

Jeff

Fix data loss on crash after sorted GiST index build.

```
If a checkpoint happens during sorted GiST index build, and the system
crashes after the checkpoint and after the index build has finished,
the data written to the index before the checkpoint started could be
lost. The checkpoint won't fsync it, and it won't be replayed at crash
recovery either. Fix by calling smgrimmedsync() after the index build,
just like in B-tree index build.

Backpatch to v14 where the sorted GiST index build was introduced.

Reported-by: Melanie Plageman
Discussion: https://www.postgresql.org/message-id/CAAKRu_ZJJynimxKj5xYBSzi
```

master
REL_16_BETA1 ... REL_15_BETA1

hlinnaka committed on Feb 24, 2022

PostGIS hacky hacks by Komzpa

GiST penalty for geometry

```
431 + #ifdef ADVANCED_PENALTY
432 + static float
433 + pack_float(const float value, const int realm)
434 + {
435 +   union {
436 +     float f;
437 +     struct { unsigned value:31, sign:1; } vbits;
438 +     struct { unsigned value:29, realm:2, sign:1; } rbits;
439 +   } a;
440
441 +   a.f = value;
442 +   a.rbits.value = a.vbits.value >> 2;
443 +   a.rbits.realm = realm;
444 +
445 +   return a.f;
446 + }
447 + #endif
```

https://github.com/x4m/postgres_g/commit/76972490adb756e02df07f28723951f269729fcc

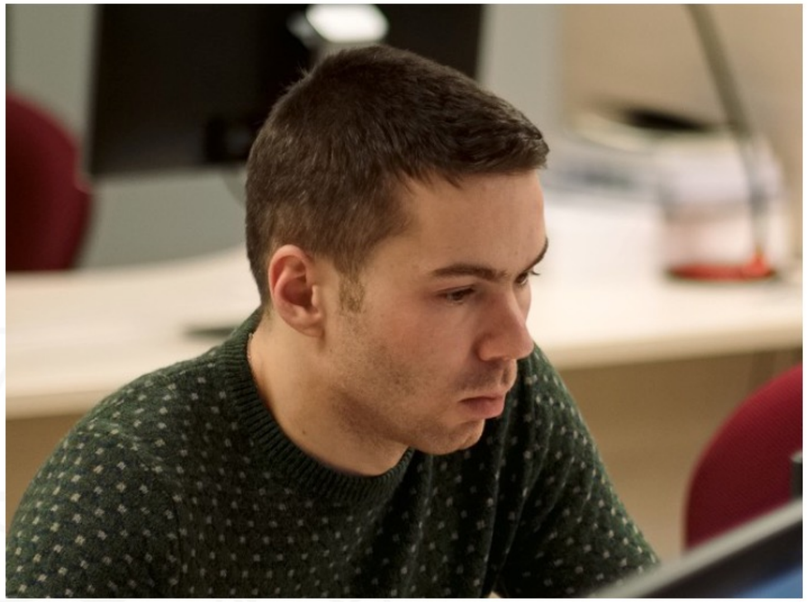
gist_btree_bui...

Commits on Feb 14, 2019

Add Z-order curve functions for points and boxes
Nikita Glukhov committed on Feb 14, 2019

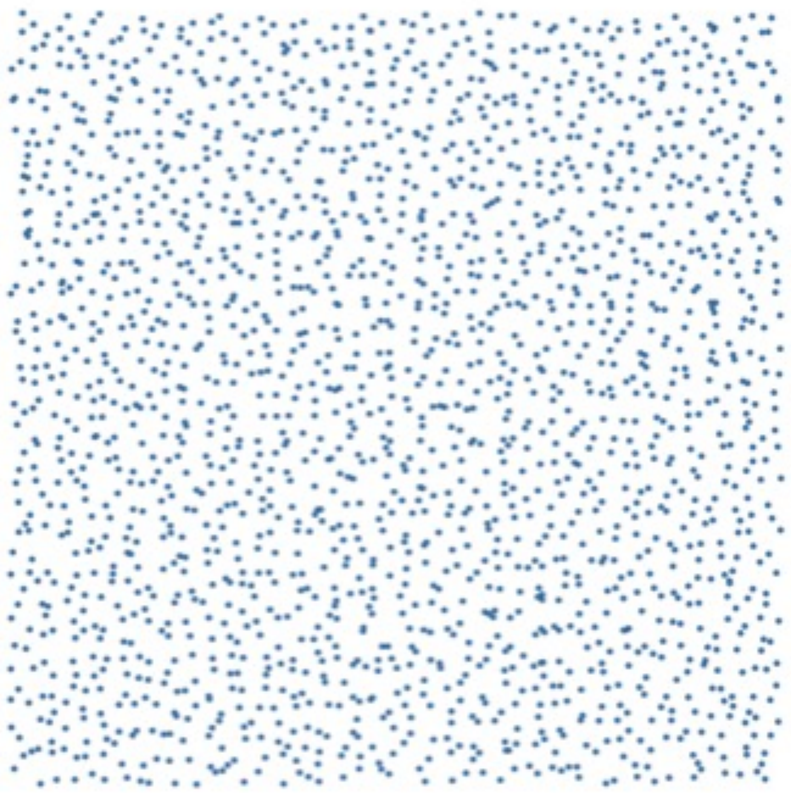
Compress GiST attributes before B-Tree sort
Nikita Glukhov committed on Feb 14, 2019

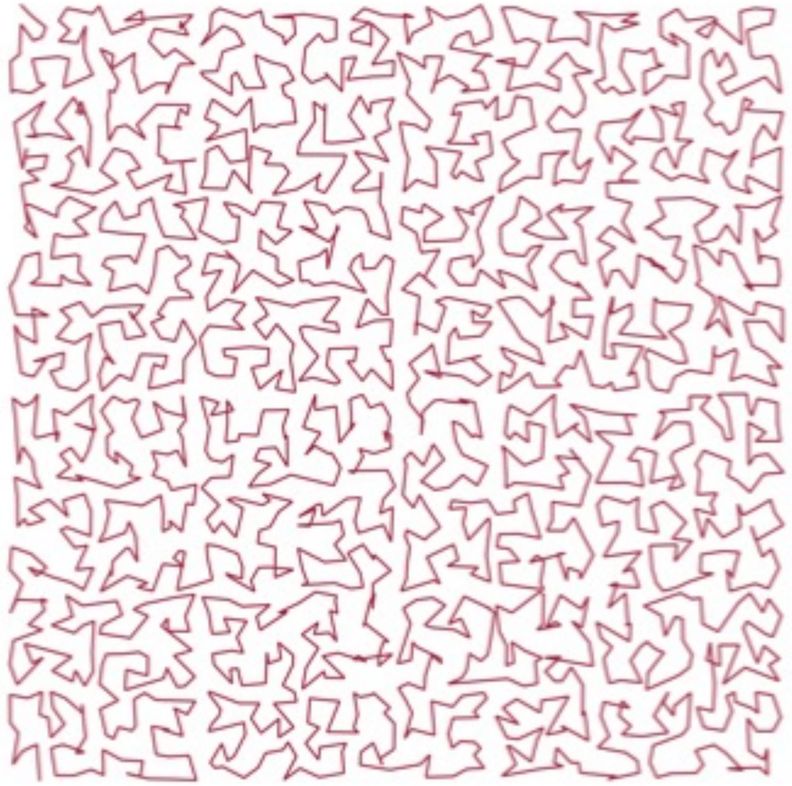
Build GiST index as B-Tree
Nikita Glukhov committed on Feb 14, 2019



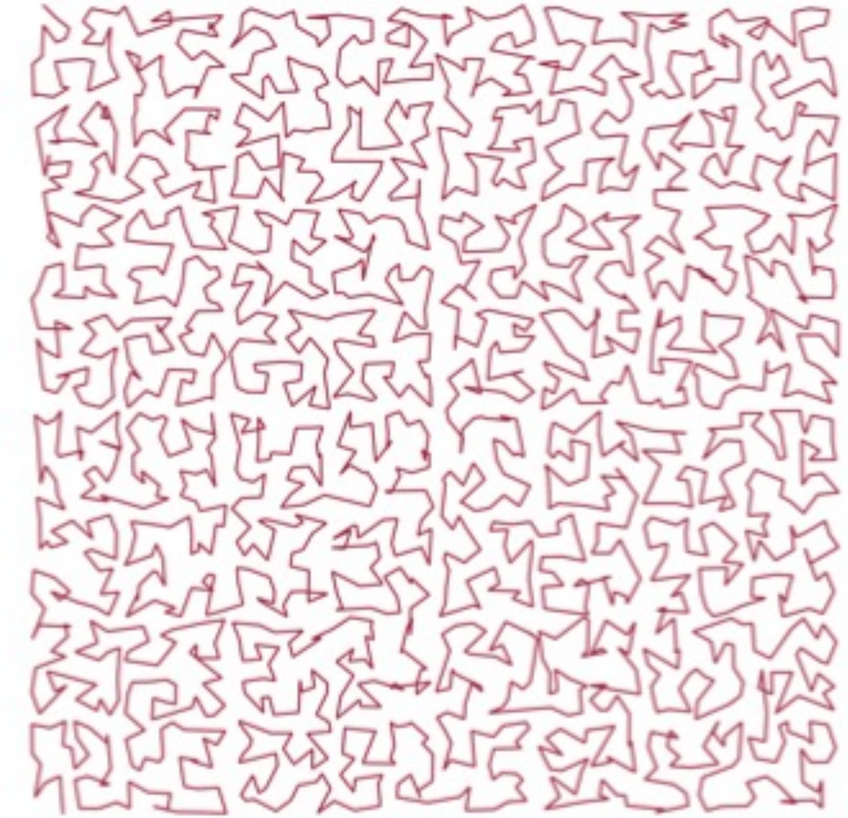
```
27  /*
28   * amproc indexes for GiST indexes.
29   */
30  #define GIST_CONSISTENT_PROC            1
31  #define GIST_UNION_PROC                 2
32  #define GIST_COMPRESS_PROC              3
33  #define GIST_DECOMPRESS_PROC            4
34  #define GIST_PENALTY_PROC               5
35  #define GIST_PICKSPLIT_PROC             6
36  #define GIST_EQUAL_PROC                 7
37  #define GIST_DISTANCE_PROC              8
38  #define GIST_FETCH_PROC                 9
39  #define GIST_OPTIONS_PROC               10
40  #define GIST_SORTSUPPORT_PROC           11
41  #define GISTNProcs                  11
42
```

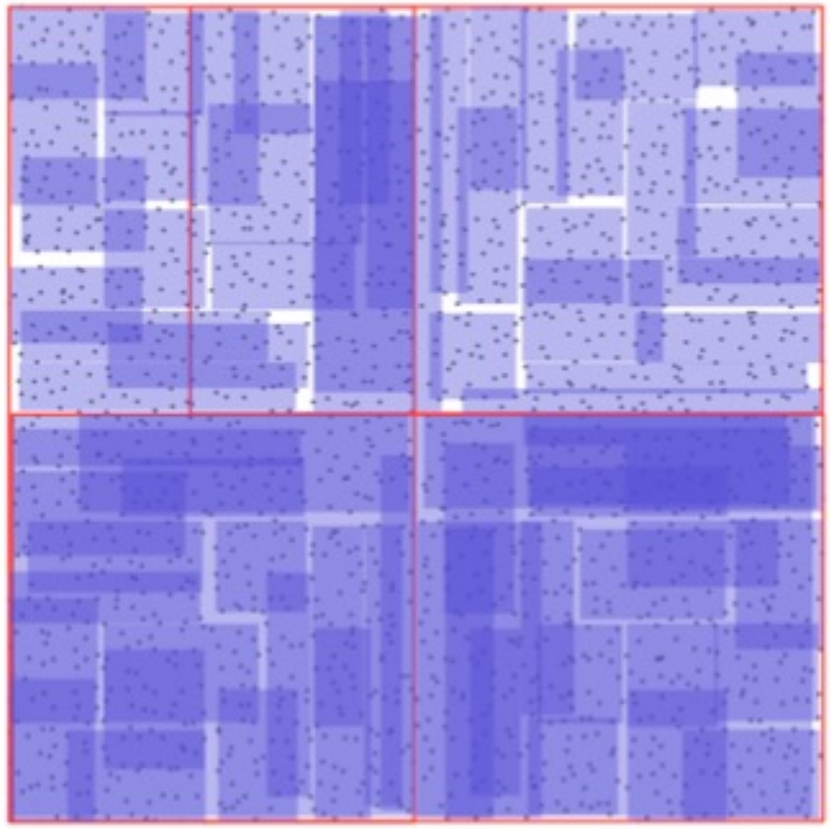
Sorting using Hilbert curve





Filling pages by sorted tuples





```
create index roads_rdr_idx on roads_rdr using gist(geom);
```

PG14 / WITH SORTSUPPORT / CREATE 200 ms
PG14 / NO SORTSUPPORT / CREATE 1705 ms

```
select pg_relation_size('roads_rdr_idx');
```

PG14 / WITH SORTSUPPORT / IDXSIZE 2940928 kb

PG14 / NO SORTSUPPORT / IDXSIZE 5439488 kb

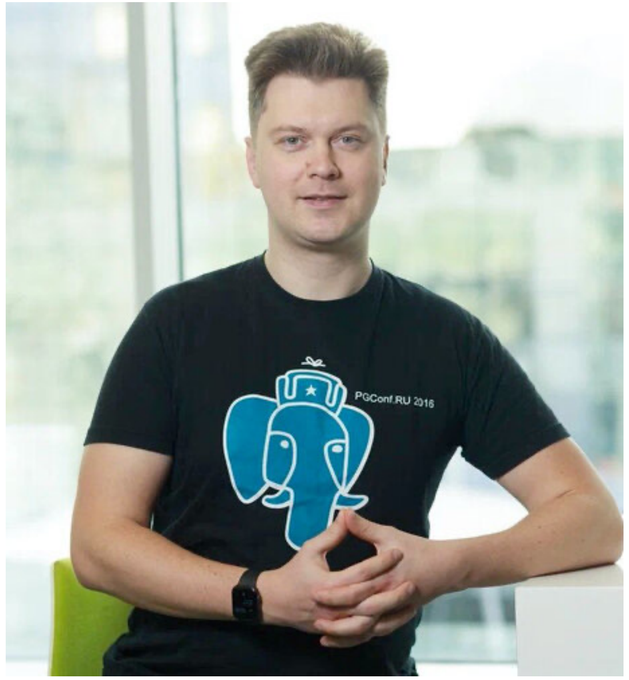
```
select count(*) from roads_rdr a, roads_rdr b where a.geom && b.geom;
```

PG14 / WITH SORTSUPPORT / QUERY 11200 ms
PG14 / NO SORTSUPPORT / QUERY **4700 ms**

GiST sorting build

Edit | Comment/Review ▾ | Change Status ▾

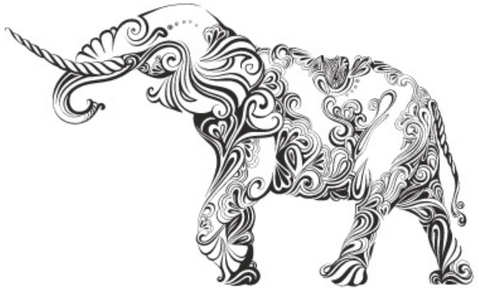
Title	GiST sorting build
Topic	Server Features
Created	2019-09-02 05:36:43
Last modified	2020-11-05 16:15:23 (2 years, 6 months ago)
Latest email	2021-12-16 09:18:52 (1 year, 5 months ago)
Status	2020-09: Committed 2020-07: Moved to next CF 2020-03: Moved to next CF 2020-01: Moved to next CF 2019-11: Moved to next CF
Target version	
Authors	Andrey Borodin (x4m)
Reviewers	Heikki Linnakangas (heikki), Pavel Borisov (pborisov)
Committer	Heikki Linnakangas (heikki)



Building gist by default GiST

- PG 13- : buffered build
- PG 14+ : sorted build
- PostGIS 3.2 adds sorted build for 2D geometry

K-Nearest Neighbors Search in PostgreSQL



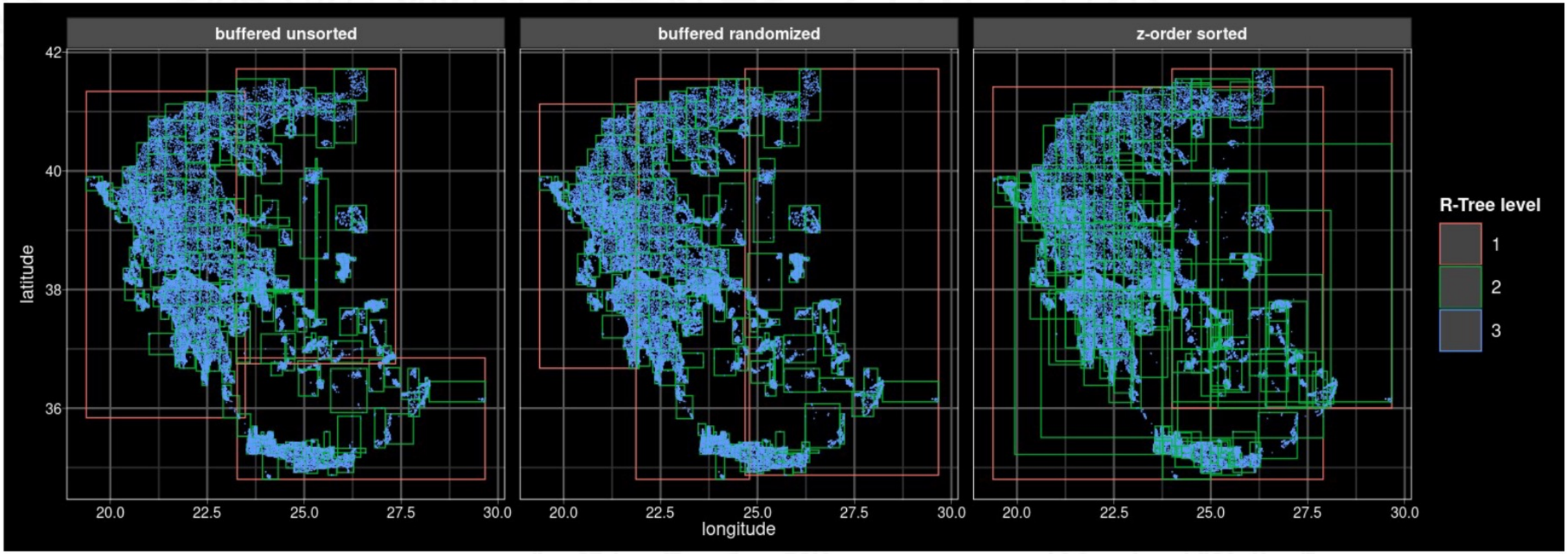
Oleg Bartunov, Nikita Glukhov
Postgres Professional

http://www.sai.msu.su/~megera/postgres/talks/knn-pgconf-2021.pdf

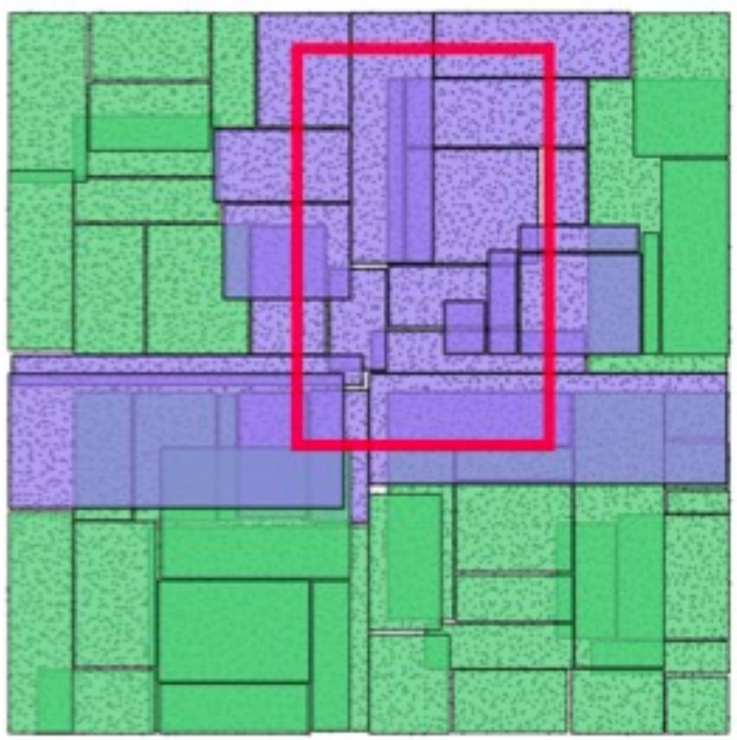
PostgresPro

Example: GiST R-Tree for Greece

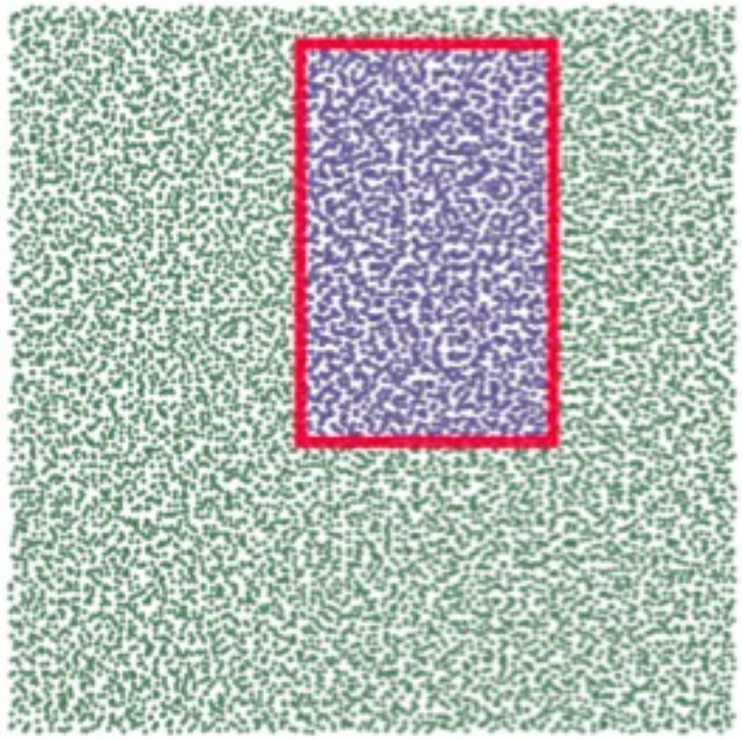
Z-order index boxes overlaps more strongly



What’s the problem of a sorted build?



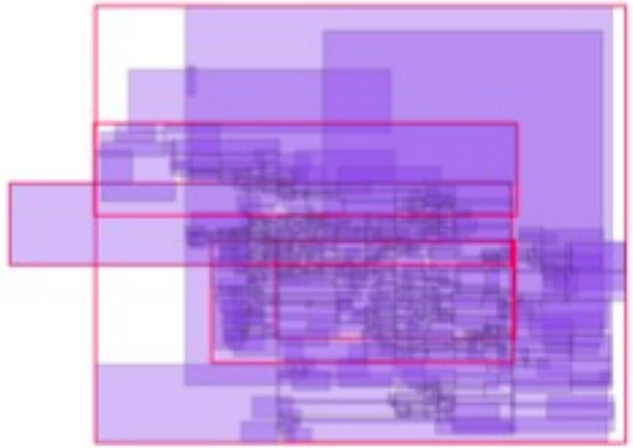
Level 1



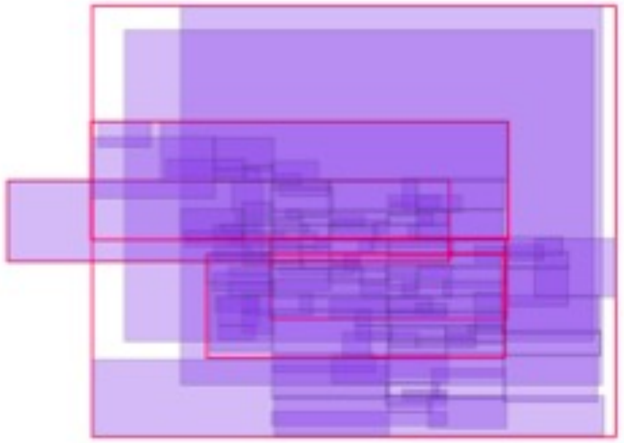
Level 2

fillfactor

fillfactor = 100
self-join = 3120.143 ms
index size = 2.6 mb



fillfactor = 50
self-join = 2346.177 ms
index size = 5.3 mb



fillfactor = 10
self-join = 1019.454 ms
index size = 29.2 mb

What’s the problem of a sorted build?

We use neither Split() nor Penalty() from opclass

New algorithm with Split()

	SORT SUPPORT (PG14)	SORT SUPPORT (PG15)	NO SORT SUPPORT
CREATE INDEX	51.2429	165.6421	273.2494
SELECT, SELF-JOIN	4100.5645	1752.5885	1709.3702
SELECT, TILING	458.5242	418.7391	424.466
SELECT, KNN (K = 1)	401.8068	242.7706	248.6685
SIZE OF INDEX (MB)	2.940928	4.95616	5.496832

https://commitfest.postgresql.org/36/3488/

Small nuance

```
#define GIST_SORTED_BUILD_PAGE_NUM 4
```

New build:

- Available in PG 15
- Default in PostGIS 3.3

Postgres 15: better GiST build

reduce page overlap of GiST indexes built using sorted method

Edit | Comment/Review ▾ | Change Status ▾

Title	reduce page overlap of GiST indexes built using sorted method
Topic	Performance
Created	2021-12-30 11:31:48
Last modified	2022-02-08 05:19:31 (1 year, 3 months ago)
Latest email	2022-02-04 00:52:56 (1 year, 3 months ago)
Status	2022-03: Committed 2022-01: Moved to next CF
Target version	15
Authors	Sergei Shoulbakov (sshoulbakov), Andrey Borodin (x4m), Aliaksandr Kalenik (kalenik.aliaksandr.gmail.com)
Reviewers	Regina Obe (reginaobe)
Committer	Alexander Korotkov (smagen)

TODO

Functions for sorting GiST build of gist_btree types

Edit | Comment/Review ▾ | Change Status ▾

Title	Functions for sorting GiST build of gist_btree types
Topic	Miscellaneous
Created	2020-11-05 16:09:37
Last modified	2022-04-08 15:16:25 (1 year, 1 month ago)
Latest email	2021-12-16 09:18:52 (1 year, 5 months ago)
Status	2022-03: Returned with feedback 2022-01: Moved to next CF 2021-11: Moved to next CF 2021-09: Moved to next CF 2021-07: Moved to next CF 2021-03: Moved to next CF 2021-01: Moved to next CF
Target version	
Authors	Andrey Borodin (x4m)
Reviewers	Heikki Linnakangas (heikki), Emre Hasegeli (hasegeli)
Committer	Heikki Linnakangas (heikki)

Add sortsupport for range types and btree_gist

Edit | Comment/Review ▾ | Change Status ▾

Title	Add sortsupport for range types and btree_gist
Topic	Performance
Created	2022-06-15 10:46:24
Last modified	2023-04-09 03:21:36 (1 month, 3 weeks ago)
Latest email	2022-11-30 17:25:25 (6 months ago)
Status	2023-07: Needs review 2023-03: Moved to next CF 2023-01: Moved to next CF 2022-11: Moved to next CF 2022-09: Moved to next CF 2022-07: Moved to next CF
Target version	
Authors	Christoph Heiss (christoph.heiss)
Reviewers	Bernd Helmle (psoo)
Committer	

Amcheck verification of GiST and GIN

Edit | Comment/Review ▾ | Change Status ▾

Title	Amcheck verification of GiST and GIN
Topic	Monitoring & Control
Created	2022-06-30 06:40:35
Last modified	2023-04-09 02:55:16 (1 month, 3 weeks ago)
Latest email	2023-04-06 04:00:01 (1 month, 3 weeks ago)
Status	2023-07: Ready for Committer 2023-03: Moved to next CF 2023-01: Moved to next CF 2022-11: Moved to next CF 2022-09: Moved to next CF 2022-07: Moved to next CF
Target version	
Authors	Andrey Borodin (x4m), Heikki Linnakangas (heikki), Grigory Kryachko (gskryachko)
Reviewers	José Villanova (azlev), Aleksander Alekseev (aalekseev)

Ambitious ideas

- Spatial Join (pg_sphere-like)
- Better GiST API (to suite RR*-tree index)
- Intra-page indexing (to match B-tree perfrormance)
- Move GiST to contrib (needs improved generic WAL)

Open source software is made and maintained by people, for people

Thank you for working on Postgres!

Andrey Borodin
open source RDBMS development team lead
Yandex Cloud

telegram: x4mmm
mailto: x4mmm@yandex-team.ru