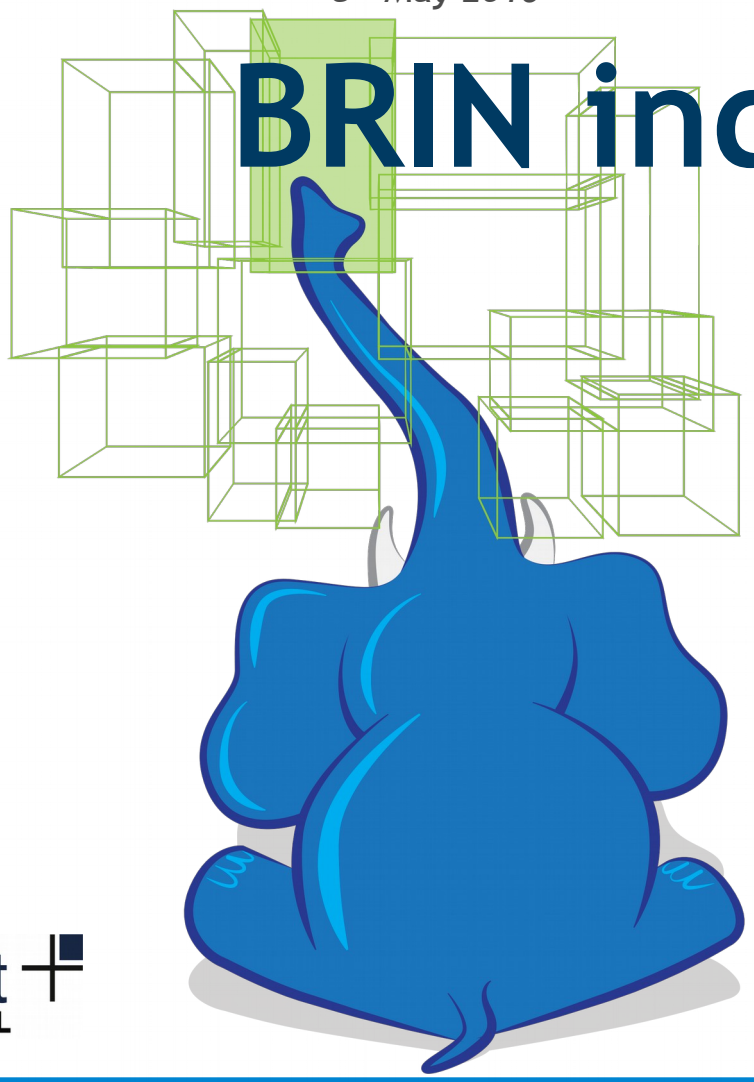
PGDay@FOSS4G.NA 2016

Raleigh, NC

3rd May 2016

FOSS4G NORTH AMERICA 2016

BRIN indexes on geospatial databases



2ndQuadrant
Professional PostgreSQL

www.2ndquadrant.com

Giuseppe Broccolo - giuseppe.broccolo@2ndquadrant.it

~$ whoami

2ndQuadrant
Professional PostgreSQL

Giuseppe Broccolo, PhD

PostgreSQL & PostGIS consultant

@giubro gbroccolo7 gbroccolo gemini__81

giuseppe.broccolo@2ndquadrant.it

Indexes & PostgreSQL

- Binary structures on disc:
  - Indexing organised in **nodes** into 8kB pages
  - Speed up data access: ~O(log N)
  - High performance until the index can be contained in RAM
    - Size: ~O(N)

Giuseppe Broccolo - giuseppe.broccolo@2ndquadrant.it

Geospatial indexes in PostgreSQL

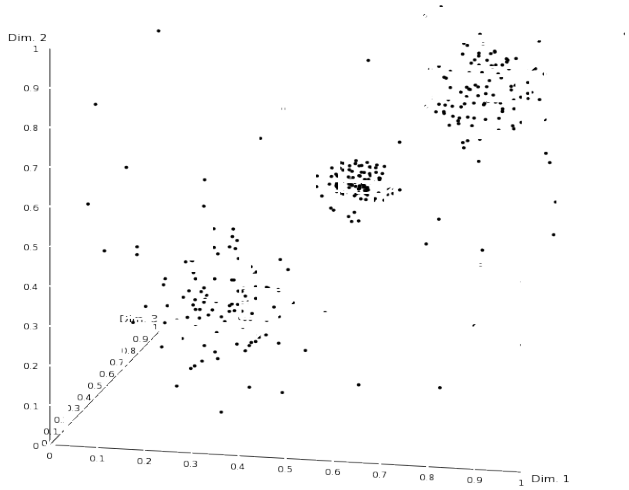
- All datatypes can be indexed in principle...
  - ...just define the needed **OpClass**!
    - Operators & Support functions

- Geospatial data can be larger than 8kB
  - A single node cannot be contained into a single page
  - GiST indexing allows to reduce the information that has to be indexed ***compressing*** the PostgreSQL in-core geospatial objects into its ***bounding box***

Giuseppe Broccolo - giuseppe.broccolo@2ndquadrant.it

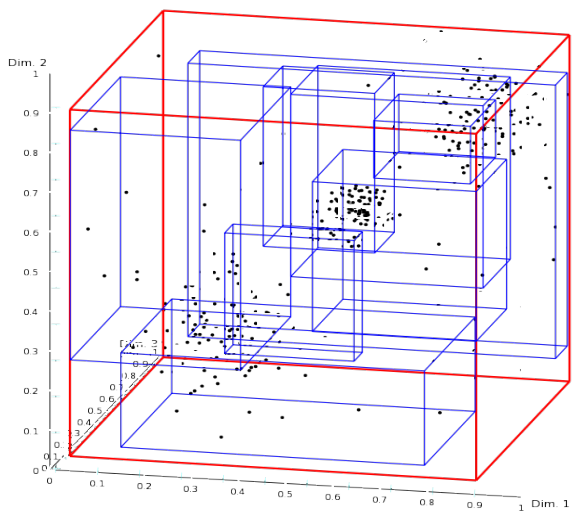
PostGIS datatypes

In PostGIS (since v0.6) **`geometry`**/**`geography`** are converted into their float-precision bounding boxes

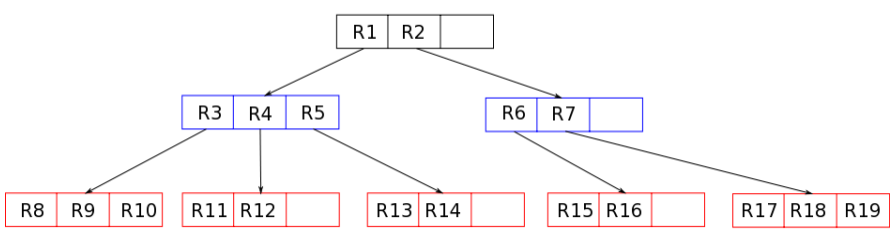


PostGIS datatypes

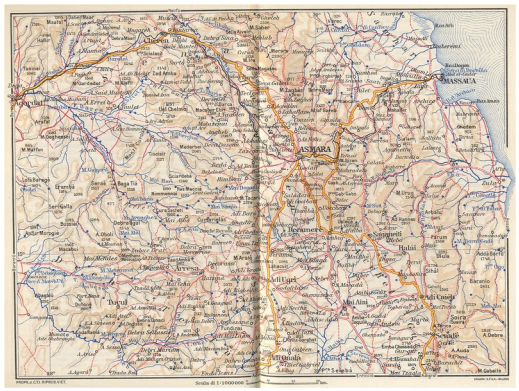
In PostGIS (since v0.6) **`geometry`**/**`geography`** are converted into their float-precision bounding boxes



`gidx`/**`box2df`** are then used as storage datatype to populate the index's nodes

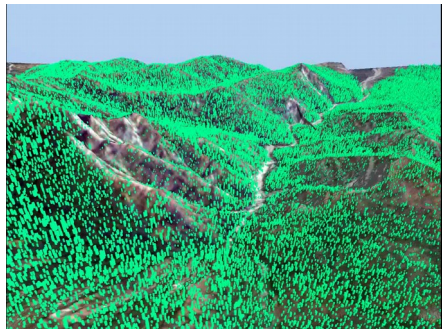


GIS & BigData



Maps of entire countries

LiDAR surveys



It's relatively easy to have to work with terabytes of geospatial data…

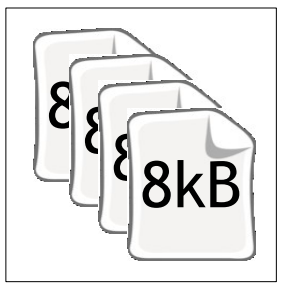
…can indexes be contained in RAM?

Giuseppe Broccolo - giuseppe.broccolo@2ndquadrant.it

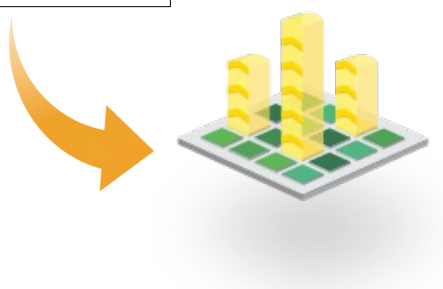
A new index: Block Range INdexing

2ndQuadrant
Professional PostgreSQL

S. Riggs, A. Herrera

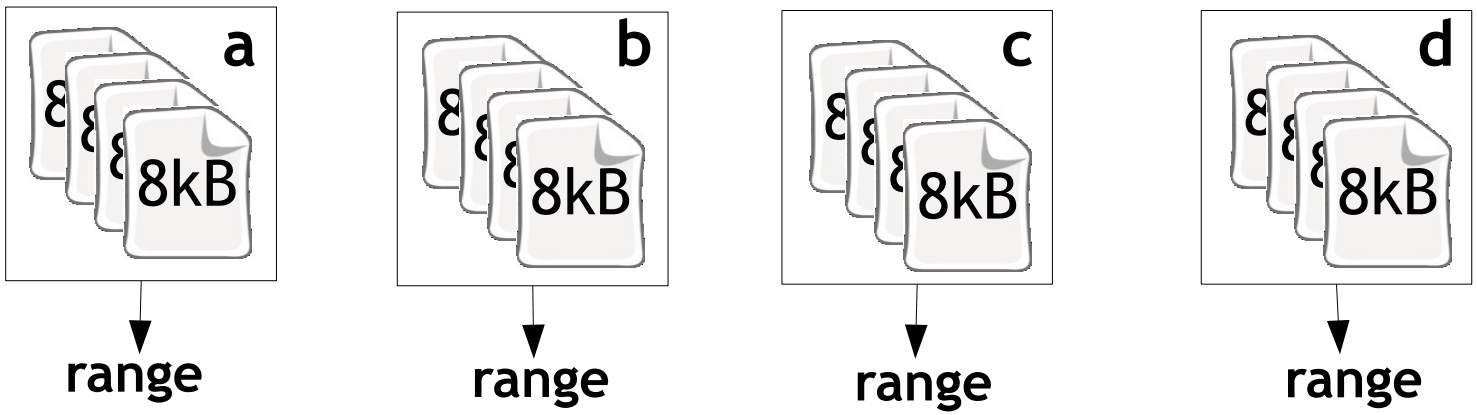


- **BLOCK**: set of physically adjacent 8kB pages
- **RANGE**: boundaries of data contained in the block
- **granularity: `pages_per_range`**
  - default: blocks of 128 pages

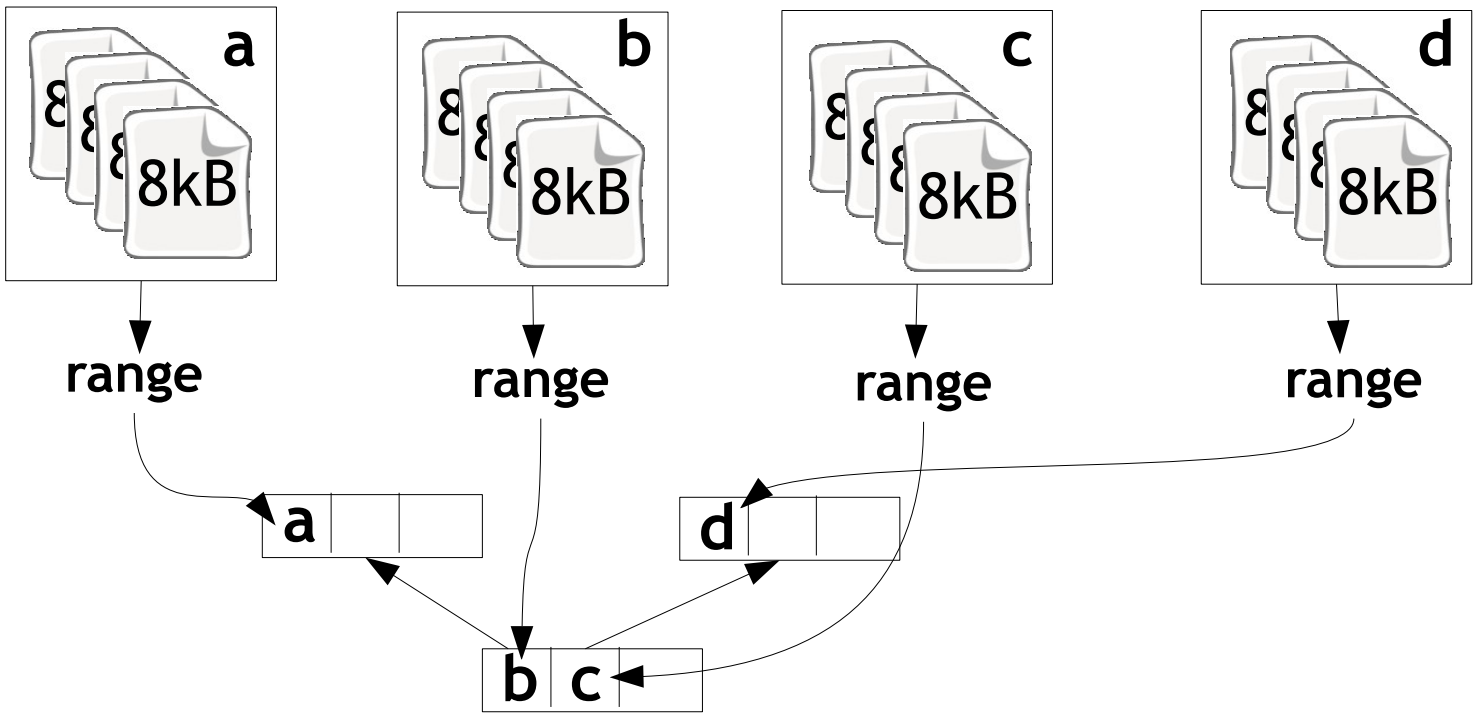


```
CREATE INDEX idx ON table USING brin(column) WITH (pages_per_range=10);
```

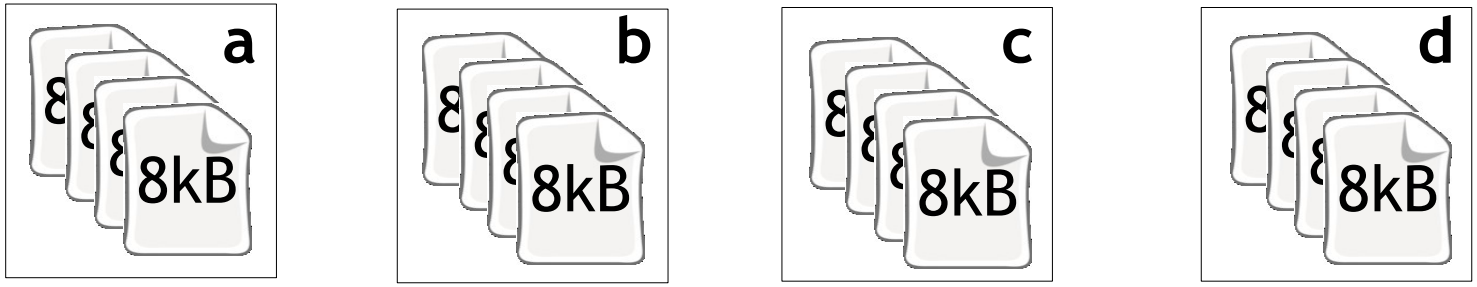
BRIN - indexing

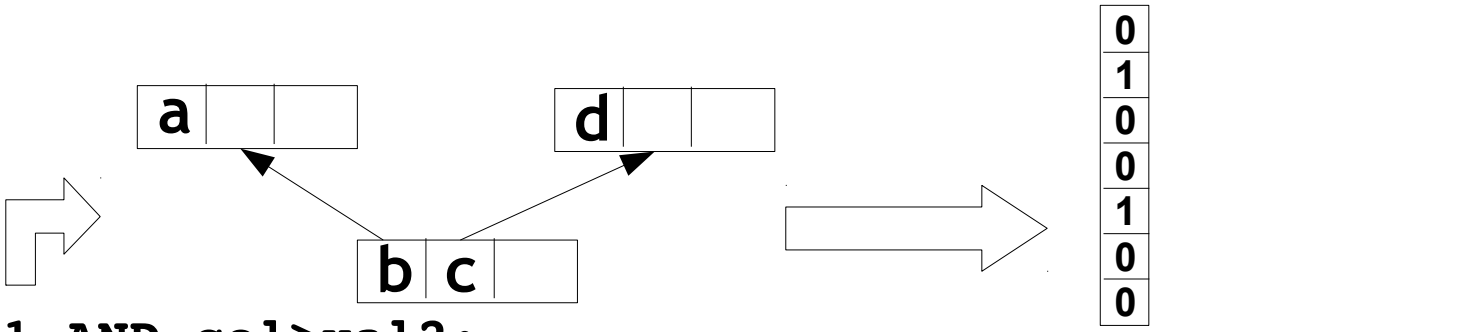


BRIN - indexing



BRIN - scan execution

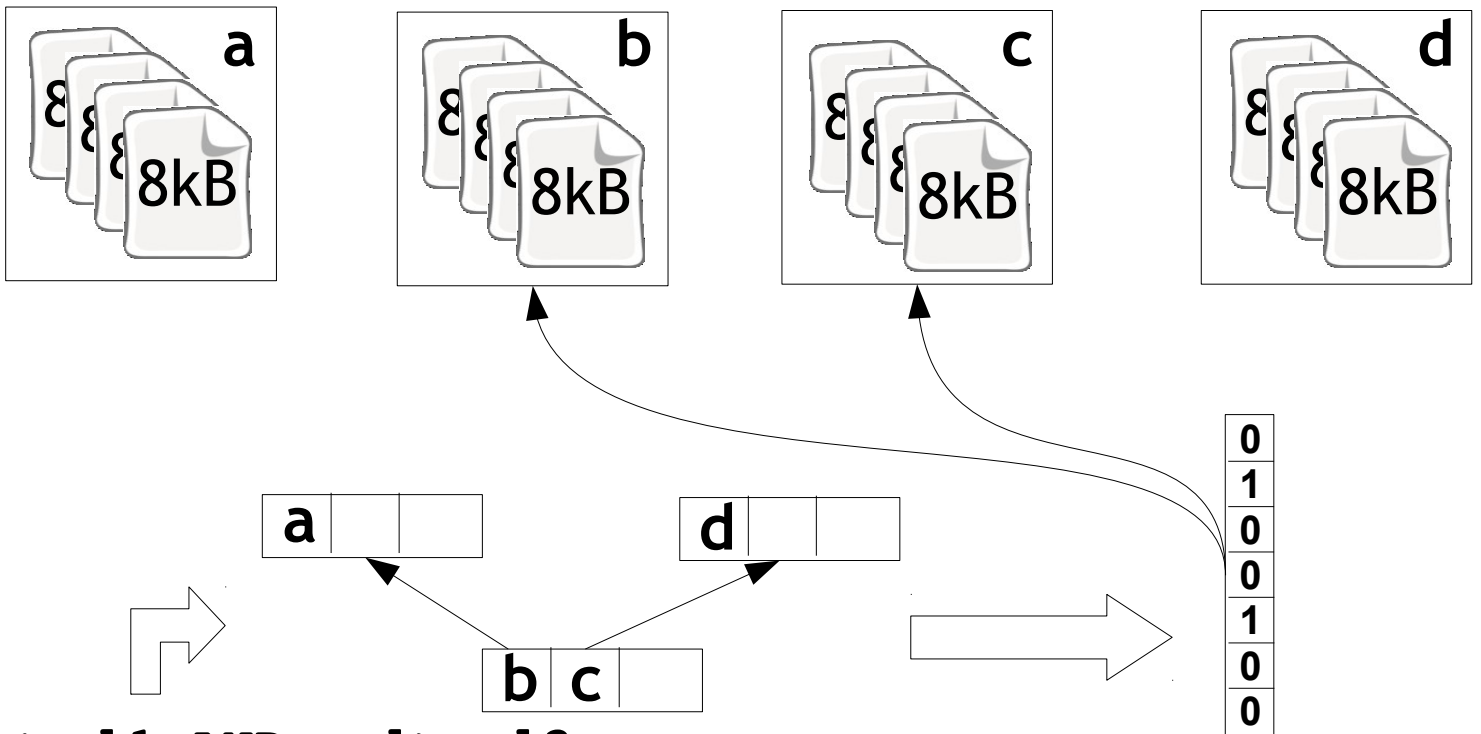





```
...WHERE col<val1 AND col>val2;
```

BitMapIndexScan

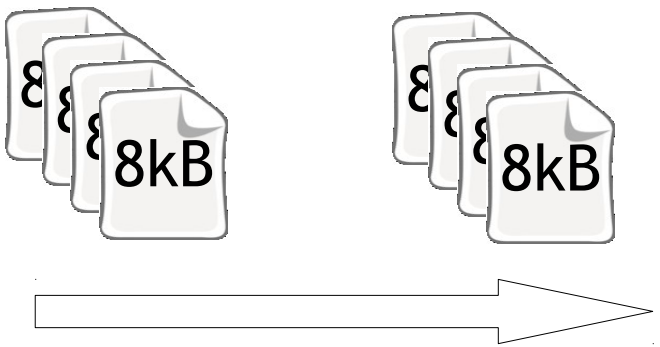
BRIN - scan execution




```
...WHERE col<val1 AND col>val2;
```

BitMapHeapScan

BRIN - scan execution



"ordered" **TidScan**

(recheck cond)

```
...WHERE col<val1 AND col>val2;
```

BRIN - an example of execution plan

```
                                QUERY PLAN
 ---------------------------------------------------------------------------
 -> Bitmap Heap Scan on column  (cost=113.50..26741.01 rows=10000
width=20) (actual time=6.513..3094.020 rows=2499193 loops=1)
     Recheck Cond: column = value
     Rows Removed by Index Recheck: 7500807
     Heap Blocks: lossy=63695
    ->  Bitmap Index Scan on brin_index  (cost=0.00..111.00 rows=10000
width=0) (actual time=6.200..6.200 rows=637440 loops=1)
          Index Cond: column = value
```

Giuseppe Broccolo - giuseppe.broccolo@2ndquadrant.it

BRIN support and extensibility

- Two kind of support functions required in the **`OpClass`**:
  - **`minmax`** key values are added following sorting criteria + sorting operators
  - **`inclusion`** key values are added following inclusion criteria + inclusion operators
  - No kNN support currently available!
- **Extensibility**
  - provide support functions + operators to define the **`OpClass`**

`http://www.postgresql.org/docs/9.5/static/brin-extensibility.html`

Giuseppe Broccolo - giuseppe.broccolo@2ndquadrant.it

2ndQuadrant
Professional PostgreSQL

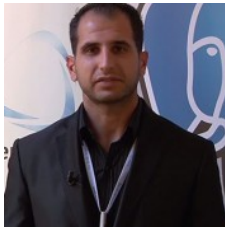
PGDay@FOSS4G.NA 2016
Raleigh, NC
3rd May 2016

FOSS4G NORTH AMERICA 2016

BRIN support for PostGIS

Ronan Dunklau

Julien Rouhaud



me

DALIBO
L'expertise PostgreSQL

2ndQuadrant
Professional PostgreSQL

OSGeo
Code Sprint
Paris 2016

https://github.com/dalibo/postgis/tree/for_upstream

Giuseppe Broccolo - giuseppe.broccolo@2ndquadrant.it

BRIN support for PostGIS

- Different **`OpClass`** (**`inclusion`** support) for
  - 2D (default), 3D, 4D **`geometry`**
  - **`geography`**
  - **`box2d/box3d`** (cross-operators defined in the **`OpFamily`**)
- Storage datatype: float-precision (such as in GiST)
  - **`gidx`** (3D, 4D **`geometry`**)
  - **`box2df`** (2D **`geometry`**, **`geography`**)
- **`brin_inclusion_add_value()`** has been redefined
- Operators: **`&&`**, **`@`**, **`~`** (2D) - **`&&&`** (3D, 4D)

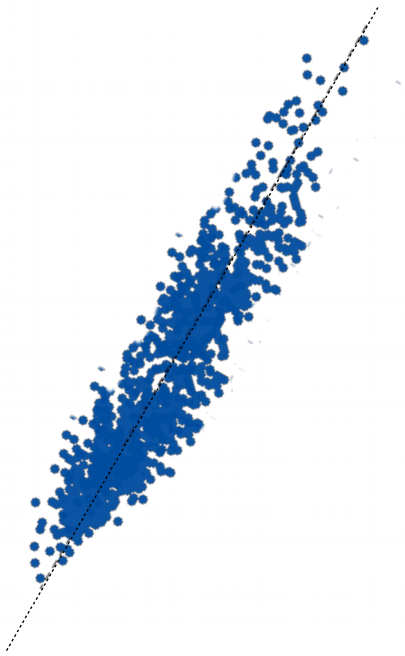
NO kNN SUPPORT!!

Now it's time to test our patch!

Giuseppe Broccolo - giuseppe.broccolo@2ndquadrant.it

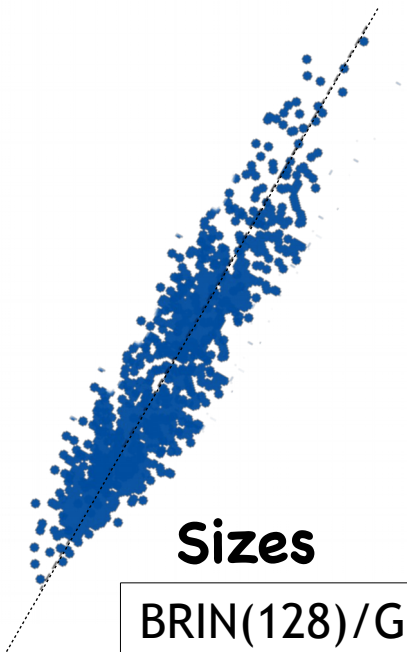
1st example: "sorted" geometry points

~10M, 3D, SRID=0, distribution range: 0÷100kunits



Giuseppe Broccolo - giuseppe.broccolo@2ndquadrant.it

1st example: "sorted" geometry points



~10M, 3D, SRID=0, distribution range: 0÷100kunits

```
=# CREATE INDEX idx_gist ON points USING gist
-# (geom gist_geometry_ops_nd);

=# CREATE INDEX idx_brin_128 ON points USING brin
-# (geom brin_geometry_inclusion_ops_3d);

=# CREATE INDEX idx_brin_10 ON points USING brin
-# (geom brin_geometry_inclusion_ops_3d)
-# WITH (pages_per_range=10);
```

Sizes

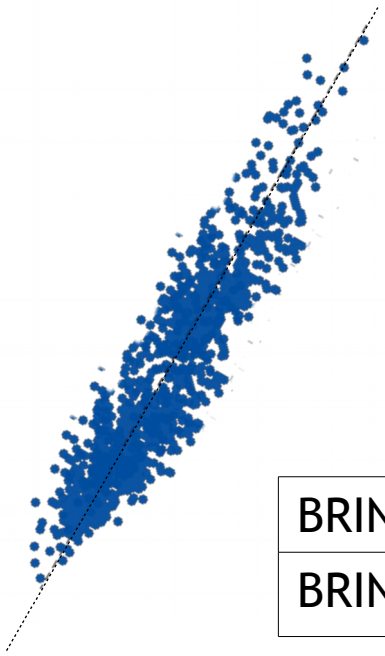
BRIN(128)/GiST	1/2000
BRIN(10)/GiST	1/1000

Building times

BRIN(128)/GiST	1/30
BRIN(10)/GiST	1/20

Giuseppe Broccolo - giuseppe.broccolo@2ndquadrant.it

1st example: "sorted" geometry points



~10M, 3D, SRID=0, distribution range: 0÷100kunits

```
=# SELECT * FROM points
-# WHERE 'BOX3D(10. 10. 10., 12., 12., 12.)'::box3d &&& geom;
```

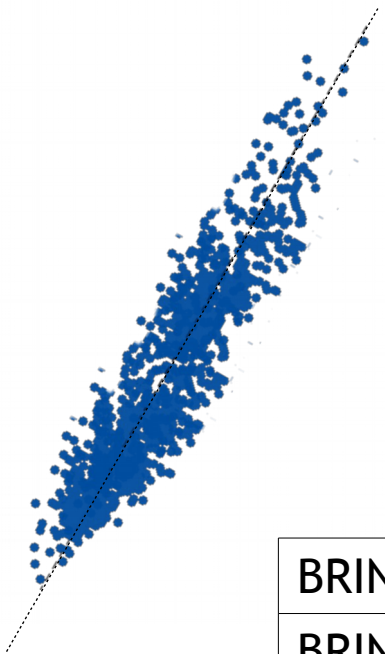
Execution times

BRIN(128)/GiST	50/1
BRIN(10)/GiST	6/1

BRIN(128)/SeqScan	1/60
BRIN(10)/SeqScan	1/350

Giuseppe Broccolo - giuseppe.broccolo@2ndquadrant.it

2nd example: "unsorted" geometry points



~10M, 3D, SRID=0, distribution range: 0÷100kunits

```
=# CREATE TABLE unsorted_points AS
-# SELECT * FROM points ORDER BY random();

=# SELECT * FROM unsorted_points
-# WHERE 'BOX3D(10. 10. 10., 12., 12., 12.)'::box3d &&& geom;
```

Execution times

BRIN(128)/GiST	2800/1
BRIN(10)/GiST	400/1

BRIN(128)/SeqScan	1/1
BRIN(10)/SeqScan	1/12

Giuseppe Broccolo - giuseppe.broccolo@2ndquadrant.it

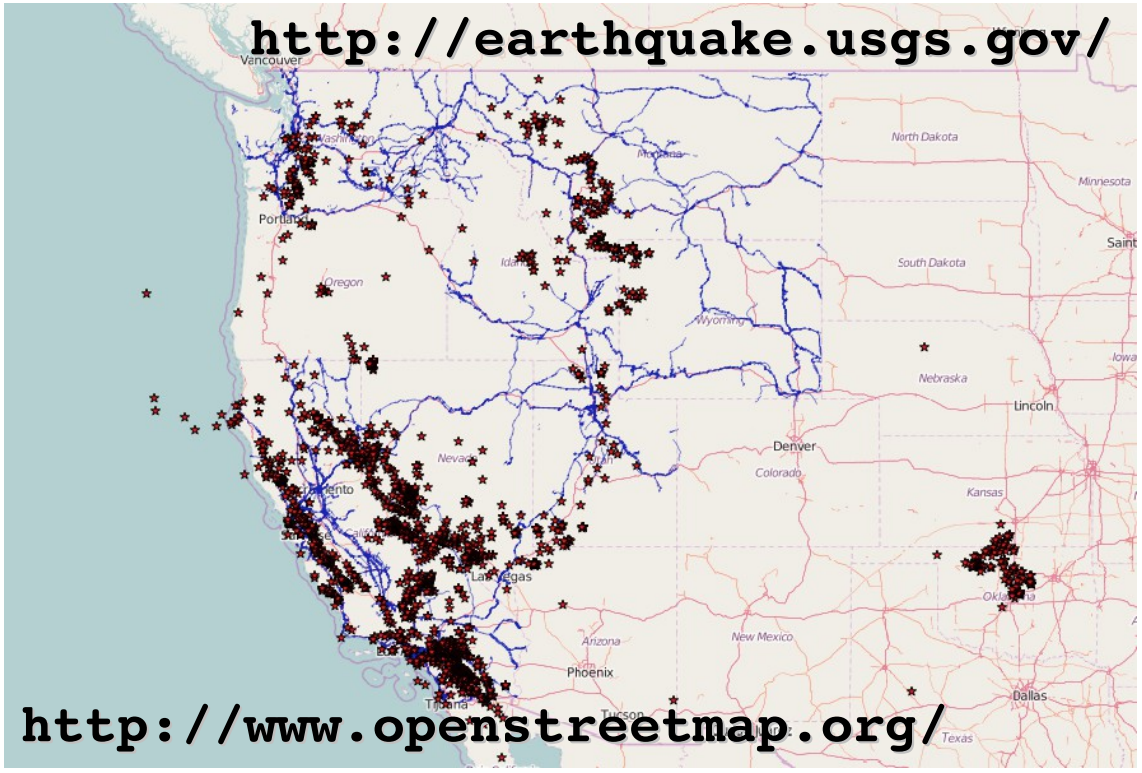
2nd example: "unsorted" geometry points

```
=# CREATE EXTENSION pageinspect;

=# SELECT blknum, value FROM brin_page_items(get_raw_page('idx_brin_128', 2), 'idx_brin_128')
-# LIMIT 5;
 blknum |                                        value
--------+-------------------------------------------------------------------------
      0 | {GIDX( 1.11394846439 1.64980053902 1.11335003376, 99982.2578125 99982.640625
99982.1171875 ) .. f .. f}
    128 | {GIDX( 0.224781364202 0.184096381068 0.798862099648, 99995.859375 99995.171875
99995.0390625 ) .. f .. f}
    256 | {GIDX( 6.25802087784 6.50119876862 6.8031873703, 99993.15625 99993.8203125
99993.2109375 ) .. f .. f}
    384 | {GIDX( 1.9203556776 1.11907613277 1.55043530464, 99995.421875 99995.234375
99995.421875 ) .. f .. f}
    512 | {GIDX( 3.44181084633 3.4120452404 3.41762590408, 99996.3125 99996.7109375 99996.59375
) .. f .. f}
(5 rows)
```


Giuseppe Broccolo - giuseppe.broccolo@2ndquadrant.it

A real case: earthquakes in the USA in 2016

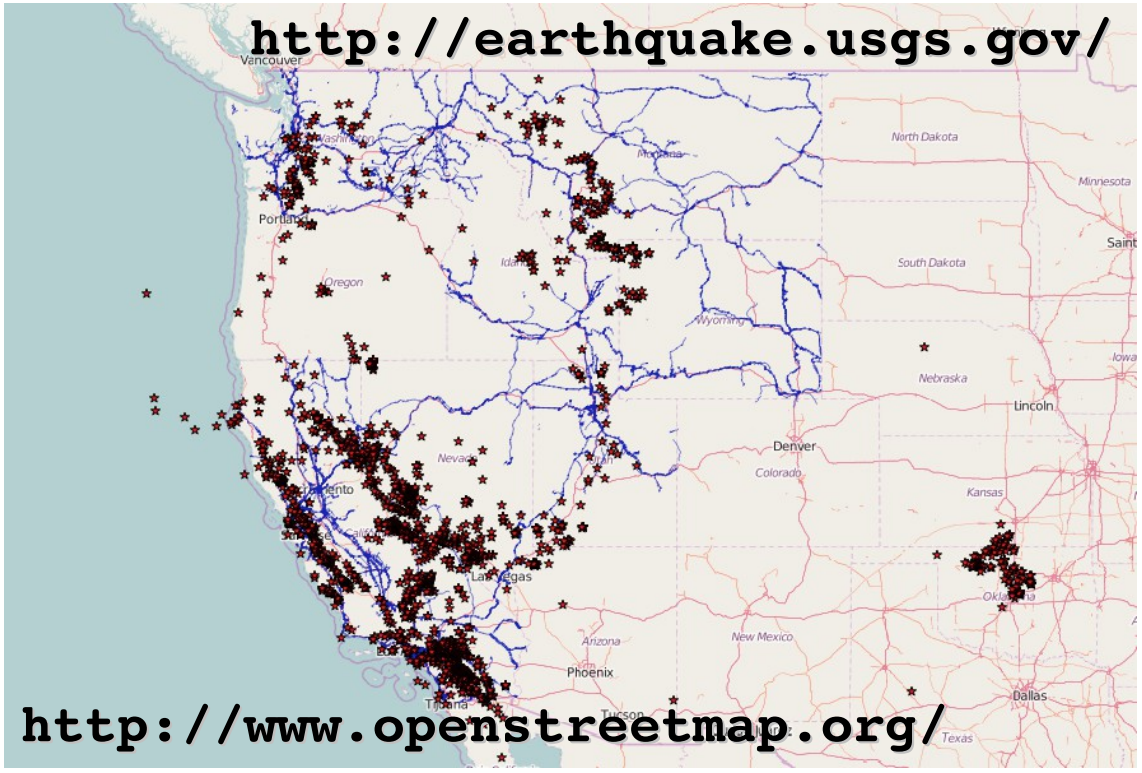


~100k lat/lon WGS84 points
(earthquakes in the world)

~1M SRID 102008 linestrings
(railways in the west side, not Colorado)

Giuseppe Broccolo - giuseppe.broccolo@2ndquadrant.it

A real case: earthquakes in the USA in 2016



~100k lat/lon WGS84 points (earthquakes in the world)

~1M SRID 102008 linestrings (railways in the west side, not Colorado)

Which railway was close (<10km) to an earthquake epicenter?

Giuseppe Broccolo - giuseppe.broccolo@2ndquadrant.it

A real case: earthquakes in the USA in 2016

GiST

```
=# CREATE INDEX idx_rl_gist ON westrailways USING gist
-# (ST_Buffer(GEOGRAPHY(ST_Trasform(geom), 4326), 10000));

=# CREATE INDEX idx_eq_brin ON worldearthquakes USING gist (coord);
```

BRIN

```
=# CREATE INDEX idx_rl_brin ON westrailways USING brin
-# (ST_Buffer(GEOGRAPHY(ST_Trasform(geom), 4326), 10000))
-# WITH (pages_per_range=10);

=# CREATE INDEX idx_eq_brin ON worldearthquakes USING brin (coord)
-# WITH (pages_per_range=10);
```

BRIN/GiST Sizes	~1/100
BRIN/GiST Times	~1/200

Giuseppe Broccolo - giuseppe.broccolo@2ndquadrant.it

A real case: earthquakes in the USA in 2016

```
=# WITH us_eq AS (
-#     SELECT coord FROM world
-#     WHERE coord && 'BOX2D(-126.90 49.73, -65.83 24.73)'::box2d
-# ) SELECT * FROM railways r, us_eq u
-#   WHERE ST_Buffer(GEOGRAPHY(ST_Trasform(geom), 4326), 10000) && u.coord;
```

- without indexes: ~60s
- with GiST: ~20ms
- with BRIN: ~400ms

...so, is GIS data "naturally" ordered?

Giuseppe Broccolo - giuseppe.broccolo@2ndquadrant.it

It's not a matter of "natural" order...

- both indexes are used: **`bitmapindexscan/bitmapheapscan`**
  - indexes pages has to be read many times!
- the query plan is "complex" - more "recheck" nodes in the plan

```
Recheck Cond: (ST_Buffer(GEOGRAPHY(ST_Trasform(geom), 4326), 10000) && u.coord)
              Rows Removed by Index Recheck: 10546
              Heap Blocks: lossy=512
```

BRIN

```
Recheck Cond: (ST_Buffer(GEOGRAPHY(ST_Trasform(geom), 4326), 10000) && u.coord)
              Rows Removed by Index Recheck: 5424
              Heap Blocks: lossy=42
```

GiST

Giuseppe Broccolo - giuseppe.broccolo@2ndquadrant.it

...what about INSERT's?

```
INSERT INTO railways SELECT * FROM colorado_railways;
```

GiST	147ms
BRIN (10)	79ms
BRIN (128)	63ms
No indexes	23ms

Giuseppe Broccolo - giuseppe.broccolo@2ndquadrant.it

A really good testbed: LiDAR data

PostgreSQL extension for LiDAR data: **pg_pointcloud**

http://www.slideshare.net/GiuseppeBroccolo/gbroccolo-foss4-geugeodbindex

GiST performances:

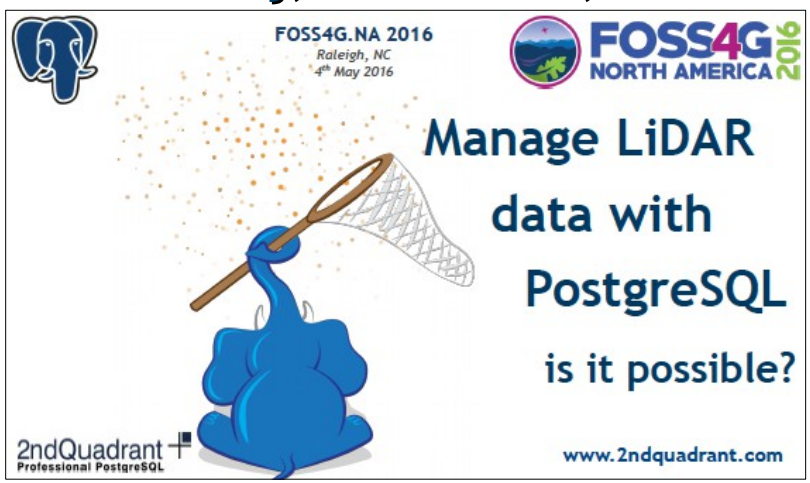
- storage: 1TB RAID1, RAM 16GB, 8 CPU @3.3GHz, PostgreSQL9.3
- index size ~O(table size)
- Index was used:
  - up to ~300M points in bbox inclusion searches
  - up to ~10M points in kNN searches

LiDAR size: ~O($10^9 \div 10^{11}$) → less than 10% can be properly indexed!

Is the patch compatible with `pg_pointcloud`?

1.6TB, ~250G points in ~560M patches

4th May, room 302C, 5:20PM



Here just a summary:

- BRIN/GiST ~O(10MB) not ~O(10GB)
- BRIN/GiST ~O(1h) not ~O(1day)
- bbox inclusion searches:
  - GiST = 20x faster than BRIN
  - BRIN = 200x faster than SeqScan
  - BRIN work better with low selectivity queries
    - All BRIN index pages can be contained in RAM
    - `SELECT`s on more than ~50% of the dataset through BRIN scan

Giuseppe Broccolo - giuseppe.broccolo@2ndquadrant.it

Conclusions

- BRINs can be successfully used in geo DB based on PostgreSQL
  - totally support PostGIS datatype
    - A patch almost ready for the next PostGIS release (2.3.0)
  - easier indexes maintenance, low indexing time, low impact on concurrent **`INSERT`**s
  - less specific than GiST...but not to much!
- Really small indexes!
  - GiST performances drop as well as it cannot be totally contained in RAM
  - BRIN can be successfully used in LiDAR dataset, at least for bbox inclusion searches

Giuseppe Broccolo - giuseppe.broccolo@2ndquadrant.it

Special thanks to...

- Julien Rouhaud
- Ronan Dunklau

- Alvaro Herrera
- Emre Hesegeli

- ...and to 2ndQuadrant!

Creative Commons license