B-Tree deduplication in PostgreSQL 13: design and background

ISS2020 — October 14, 2020

https://speakerdeck.com/peterg/postgresql-deduplication-2020

Peter Geoghegan

@petervgeoghegan

https://speakerdeck.com/peterg/postgresql-deduplication-2020

B-Tree deduplication

New feature in Postgres 13 (which was just released): Deduplication

- Work by Anastasia Lubennikova and myself (Peter Geoghegan)
- Targets standard B-Tree access method (default index type in Postgres)
- **Simple idea**: only store key once — not once per index entry

  - Before: ‘KeyA’, (1,22), ‘KeyA’, (1,23), ‘KeyA’, (1,24)

  - After: ‘KeyA’, (1,22)(1,23)(1,24)

https://speakerdeck.com/peterg/postgresql-deduplication-2020

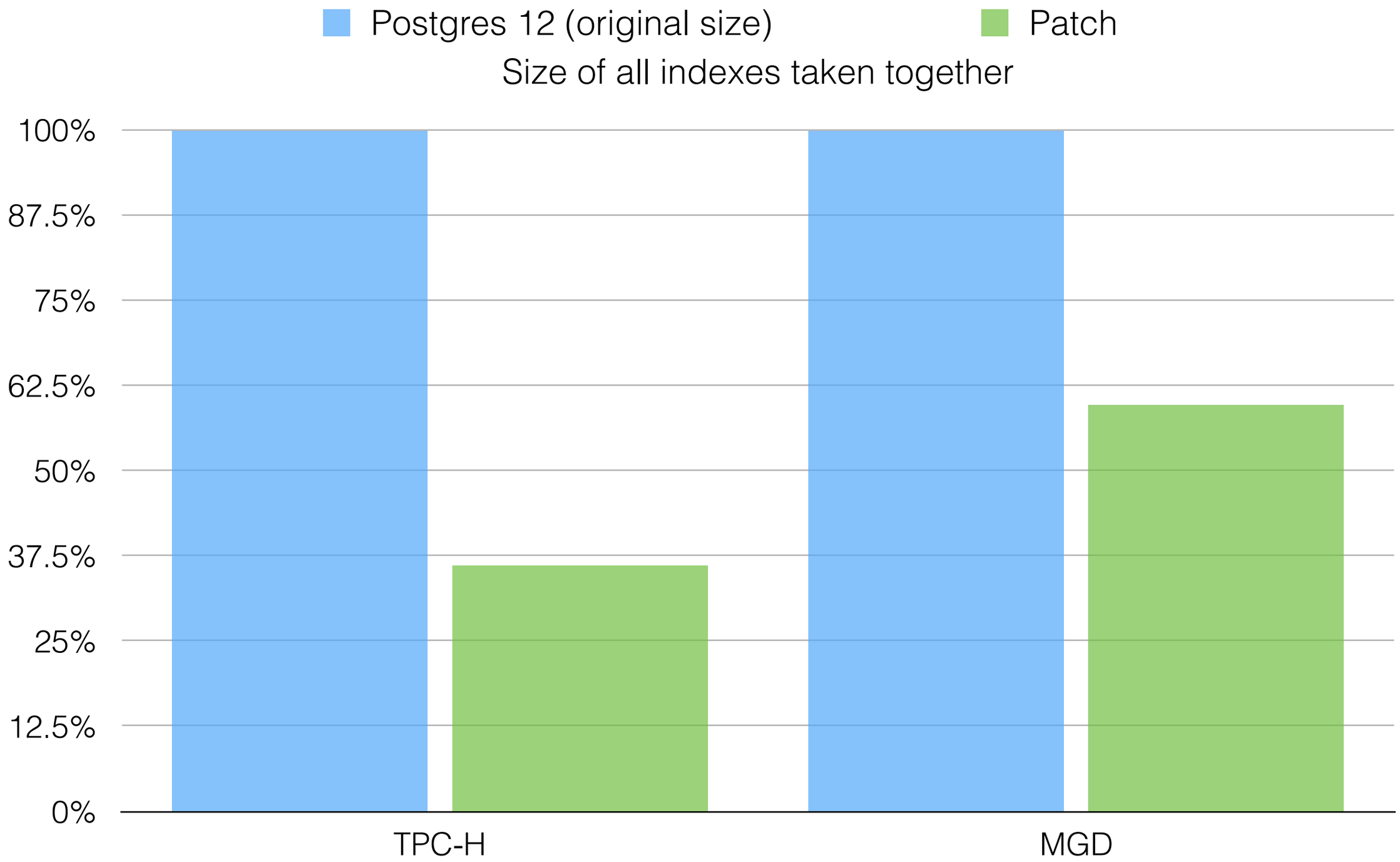
Why deduplicate?

Design changes very little about existing B-Tree indexing

- Increases performance, decreases storage costs
- Occurs only when needed, to avoid a page split
- Makes B-Tree indexes like **bitmap indexes** — though *only* when that makes sense
  - Made possible by existing “logical MVCC” design
  - No changes to **locking**, despite making indexes with duplicates much smaller

https://speakerdeck.com/peterg/postgresql-deduplication-2020

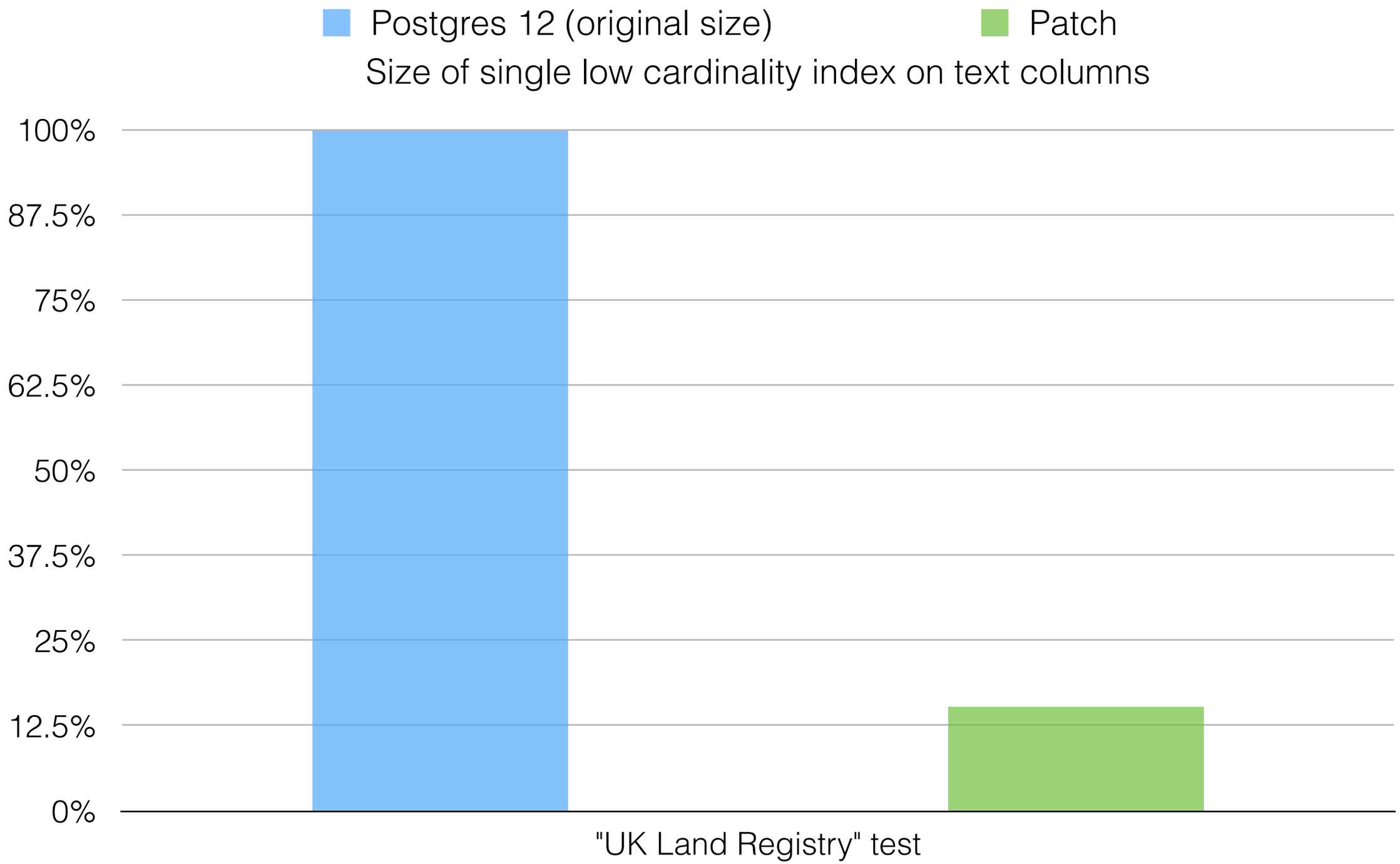
Index size reductions



https://speakerdeck.com/peterg/postgresql-deduplication-2020

crunchy data

Index size reductions



https://speakerdeck.com/peterg/postgresql-deduplication-2020

crunchy data

Design goals for deduplication

Standard B-Trees support **high concurrency** access

- Deduplication is enabled by default now, so it must not hurt performance for workloads/indexes with few duplicates
- “Best of both worlds” performance
- Posting lists (Heap TID lists) are *not* compressed
  - Limits low-level contention
  - Very little downside compared to no deduplication

https://speakerdeck.com/peterg/postgresql-deduplication-2020

The challenge of write amplification

Write amplification is a traditional problem when using Postgres — at least with some workloads

- Indexes require new entries for MVCC versions
- *All* indexes on a table receive new tuples — **not just** the indexes on columns that an UPDATE changed.
- This *only* happens when **at least one indexed column** is touched by an update, though

https://speakerdeck.com/peterg/postgresql-deduplication-2020

A solution: B-Tree “deletion deduplication”

Traditional complaint about Postgres is that sometimes indexes require new entries for MVCC versions, **even when UPDATEs do not change the keys**

- New mechanism extends dedup to delete garbage tuples to avoid a page split
- Reliably prevents “logically unnecessary” page splits. These are splits caused by version churn alone.

https://speakerdeck.com/peterg/postgresql-deduplication-2020

Conclusions

- There are significant differences among all major systems that implement MVCC

- Versioning of logical rows in PostgreSQL (as oppose to versioning of tables or of individual pages) enables innovation around indexing

  - B-Tree deduplication

  - Transactional full-text search (using GIN)

  - Geospatial indexing (using GiST)

https://speakerdeck.com/peterg/postgresql-deduplication-2020

Thanks!

https://speakerdeck.com/peterg/postgresql-indexing-2020